SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0001 $01.25/0

NUMERICAL EXPERIMENTS ON SOLVING THEODORSEN'S INTEGRAL EQUATION FOR CONFORMAL MAPS WITH THE FAST FOURIER TRANSFORM AND VARIOUS NONLINEAR ITERATIVE METHODS*

MARTIN H. GUTKNECHT†

Abstract. In [M. H. Gutknecht, Numer. Math., 36 (1981), pp. 405–429] we investigated several iterative methods for solving discretized versions of Theodorsen's nonlinear integral equation. Here we present corresponding numerical experiments and discuss some related questions, such as the application of a continuation method, the evaluation of the approximate mapping function, the selection of the appropriate type of approximation, and the use of preliminary maps to eliminate corners. Both the nonlinear SOR method and a related second order Euler method are seen to be very effective, even if the assumptions of the theory do not hold.

1. Introduction. Theodorsen's integral equation [4, p. 65] is a well-known basis for the numerical computation of the conformal mapping g of the unit disk D onto a starlike region Δ given by the polar coordinates τ, $\rho(\tau)$ of its boundary Γ. The mapping g is normalized by $g(0)=0$, $g'(0)>0$. It is uniquely determined by the boundary correspondence function θ, which is defined implicitly by

$$g(e^{it}) = \rho(\theta(t))\, e^{i\theta(t)} \quad (\forall t \in \mathbb{R}), \qquad \int_0^{2\pi} \theta(t)\, dt = 2\pi^2. \tag{1.0}$$

Theodorsen's integral equation simply states that $Y: t \mapsto \theta(t) - t$ is the conjugate periodic function of $X: t \mapsto \log \rho(\theta(t))$. A bibliography up to 1964 is contained in Gaier [4]; more recent work including [10], [15], [16], [21], [22] is referenced in [12].

Upon discretization—briefly described below—Theodorsen's integral equation becomes a fixed point equation

$$\mathbf{y} = \boldsymbol{\Phi}(\mathbf{y}) := \mathbf{K}_\Sigma \log \rho(\mathbf{t}+\mathbf{y}) \tag{1.1}$$

for $\mathbf{y} \in \mathbb{R}^{2N}$. Here $\mathbf{t} := (k\pi/N)_{k=0}^{2N-1}$ is constant, $\mathbf{x} := \log \rho(\mathbf{t}+\mathbf{y})$ is the vector obtained by componentwise evaluation, $\mathbf{y}$ approximates $Y(\mathbf{t})$, and $\mathbf{K}_\Sigma$ is a circulant skew-symmetric Toeplitz matrix. The product $\mathbf{K}_\Sigma \mathbf{x}$ can be calculated in $O(N \log N)$ arithmetic operations by using the fast Fourier transform (FFT) [11], [15].

If the discretization is based on trigonometric interpolation, $\mathbf{K}_\Sigma = \mathbf{K}$ is called *Wittich's matrix*. $\mathbf{K}$, $\mathbf{y}$, and $\mathbf{t}$ can be brought to the form

$$\mathbf{P}\mathbf{K}\mathbf{P}^T = \begin{pmatrix} \mathbf{0} & -\mathbf{L}^T \\ \mathbf{L} & \mathbf{0} \end{pmatrix}, \quad \mathbf{P}\mathbf{y} = \begin{pmatrix} \mathbf{y}'' \\ \mathbf{y}' \end{pmatrix}, \quad \mathbf{P}\mathbf{t} = \begin{pmatrix} \mathbf{t}'' \\ \mathbf{t}' \end{pmatrix}$$

by permuting columns and rows, simultaneous in such a way that all even indexed ones come before all odd indexed ones. Then $\mathbf{P}\mathbf{K}\mathbf{P}^T$ is a consistently ordered weakly cyclic matrix of index 2, and—as pointed out by Niethammer [22]—we may apply the *nonlinear SOR method* to (1.1):

$$\begin{aligned} \mathbf{y}''_{m+1} &:= -\omega \mathbf{L}^T \log \rho(\mathbf{t}' + \mathbf{y}'_m) + (1-\omega)\mathbf{y}''_m, \\ \mathbf{y}'_{m+1} &:= \omega \mathbf{L} \log \rho(\mathbf{t}'' + \mathbf{y}''_{m+1}) + (1-\omega)\mathbf{y}'_m, \end{aligned} \qquad m = 0, 1, 2, \cdots.$$

* Received by the editors January 24, 1980, and in revised form February 22, 1982.

† Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule Zürich, CH-8092 Zürich, Switzerland. This work is part of a thesis submitted for Habilitation at Eidgen. Tech. Hochschule, Zürich, Switzerland.

($\mathbf{y}_0'$, $\mathbf{y}_0''$, and the relaxation factor ω must be given.) As shown both by the author [11] and by Hübner [16], each iteration step still requires only $O(N \log N)$ operations.

In [12] we established new theoretical results on this and several other iterative methods for solving (1.1), namely the *nonlinear Jacobi iteration* with relaxation (JOR)

$$\mathbf{y}_{m+1} := \omega\boldsymbol{\Phi}(\mathbf{y}_m) + (1-\omega)\mathbf{y}_{m-1}, \qquad m = 0, 1, 2, \cdots$$

($\mathbf{y}_0$ and ω given), a *nonlinear second order Euler iteration*

$$\mathbf{y}_{m+1} := \omega\boldsymbol{\Phi}(\mathbf{y}_m) + (1-\omega)\mathbf{y}_{m-1}, \qquad m = 0, 1, 2, \cdots$$

($\mathbf{y}_0 = \mathbf{y}_{-1}$ and ω given), the related *nonlinear Chebyshev iteration* (CSI) using an ω depending on m, and the *nonlinear cyclic Chebyshev iteration* (CCSI) so related to SOR. Both SOR and CSSI require $\mathbf{K}_\Sigma = \mathbf{K}$, while the other methods allow other discretization (i.e., approximation) techniques.

The main purpose of this paper is to present numerical experiments with SOR, JOR, and Euler iteration, the latter with various types of approximation. We also discuss some related questions that turn out to be important in practice, such as the use of a continuation method to avoid convergence to useless solutions of (1.1), the evaluation of the approximate mapping function and its derivative, the behavior of the boundary correspondence function θ at a corner of Γ in case Γ is piecewise analytic, and the elimination of corners by preliminary maps.

Our numerical experiments on the rate of convergence indicate that the range of application of our results in [12] is far wider than one would expect from the theory. Most theorems in [12] require either assumption (D) or assumption (SD)—as we called them—which could be replaced by the following more convenient conditions:

Assumption (D′): ρ is continuously differentiable.

Assumption (SD′): Γ is symmetric about the real axis and ν-fold rotationally symmetric about 0, where $\nu \geqq 1$ is a divisor of N; ρ is weakly monotone and continuously differentiable in $(0, \pi/\nu)$.

Moreover, we always assume that $\mathbf{y}^*$ is a solution of (1.1) and that, in case assumption (SD′) holds, both $\mathbf{y}^*$ and its initial approximation $\mathbf{y}_0$ (and thus all iterates $\mathbf{y}_m$) lie in the $(N/\nu - 1)$-dimensional subspace $\mathcal{S}$ of vectors $\mathbf{y} \in \mathbb{R}^{2N}$ satisfying

$$y_0 = y_N = 0, \qquad y_k = -y_{2N-k} \qquad (k = 1, \cdots, N-1), \tag{1.2a}$$

$$y_k = y_{k+2N/\nu} \qquad (k = 0, \cdots, 2N - 2N/\nu - 1 \text{ if } \nu > 1). \tag{1.2b}$$

(The existence of $\mathbf{y}^*$ was established in [10], [16].) Consequently, (D′) implies (D), while (SD′) and the additional conditions $\mathbf{K}_\Sigma = \mathbf{K}$ and

$$\theta_k := t_k + y_k \in (0, \pi/\nu), \qquad k = 1, \cdots, N/\nu - 1, \tag{1.3}$$

for $\mathbf{y} = \mathbf{y}^*$ imply (SD).

The classical discretization [4], [12] of Theodorsen's integral equation, where $\mathbf{K}_\Sigma = \mathbf{K}$ is Wittich's matrix, is based on interpolating the data $(\mathbf{t}, \mathbf{x})$, i.e.,

$$t_k := k\pi/N, \qquad x_k := \log \rho(t_k + y_k), \qquad k = 0, \cdots, 2N-1, \tag{1.4}$$

by a trigonometric polynomial T and evaluating the conjugate polynomial KT at every t_k, two operations done by a fast Fourier transform (FFT) [11], [15]. As we show in [13], [34], it does not cost much more to interpolate or approximate the data (1.4) (or rather its periodic extension) by a function $P\mathbf{x}$, where $P: \Pi_{2N}^R \to \mathcal{A}$ is an operator with known attenuation factors $\boldsymbol{\tau} = (\tau_k)_{k=-\infty}^{\infty} \in l_1$ [7], [13]. Here, $\mathcal{A}$ denotes the space of real 2π-periodic functions having absolutely converging Fourier series,

and Π_{2N}^R (Π_{2N}) is the space of $2N$-periodic real (complex) sequences. If we further denote the $2N$-point discrete Fourier transform (DFT) [3], [15] by $\mathscr{F}_{2N}: \Pi_{2N} \to \Pi_{2N}$, then the DFT coefficients $\mathbf{c} := \mathscr{F}_{2N}\mathbf{x}$ of $\mathbf{x}$ and the usual Fourier coefficients $\mathbf{C} := \mathscr{F}\mathbf{P}\mathbf{x}$ are related by $c_k = \tau_k C_k (\forall k)$ [7]. In particular, interpolation by periodic spline functions of degree $2m-1$ ($m \geqq 1$), which is of this type, proved to be optimal (both in the sense of Sard and Schoenberg and in the sense of Golomb and Weinberger) for data corresponding to values of functions $X \in H^m$, where

$$H^m := \{X \in C^{m-1}(\mathbb{R}, \mathbb{R}) : X \ 2\pi\text{-periodic}, d^{m-1}X/dt^{m-1} \text{ abs. cont.}, d^m X/dt^m|_{[0,2\pi]} \in L_2[0, 2\pi]\}. \tag{1.5}$$

We also allow for various classes of smoothed approximants such as trigonometric polynomials with Cesaro (Fejér) or with Lanczos smoothing and splines interpolating smoothed data (so-called smoothing by spline functions) [13].

To account for these modifications algebraically, we have to replace Wittich's matrix $\mathbf{K}$ by $\mathbf{K}_\Sigma := \mathbf{K}\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is a symmetric circulant Toeplitz matrix, for which $\|\mathbf{K}_\Sigma\| \leqq \|\boldsymbol{\Sigma}\| \leqq 1$, but very close to 1, in all cases of interest. Moreover, one can then replace

$$\varepsilon_\rho := \operatorname*{ess\,sup}_{0 \leqq \tau \leqq 2\pi} \left| \frac{\rho'(\tau)}{\rho(\tau)} \right|$$

by $\varepsilon := \varepsilon_\rho \|\mathbf{K}_\Sigma\|$ when choosing the relaxation factor ω (of one of the nonlinear iterative methods mentioned) according to the formulas given in [12] for the case when assumption (SD) is satisfied. Here we use these formulas even for asymmetric regions, and we always identify ε and ε_ρ, i.e., we neglect that $\|\mathbf{K}_\Sigma\|$ may be slightly smaller than $\|\mathbf{K}\|$, which equals 1.

Our experiments on the accuracy of the approximate mapping function show that the classical Theodorsen method (where $\mathbf{K}_\Sigma = \mathbf{K}$) becomes inaccurate if Γ is not smooth. Often the components of $\boldsymbol{\theta} := \mathbf{t} + \mathbf{y}$ are no longer monotone. Although more suitable types of approximation yield much more useful results, it seems that the best idea is to eliminate the corners by preliminary maps. If the resulting boundary is not starlike or if its ε_ρ is large, one can try additional preliminary maps such as, e.g., those incorporated in osculation methods [2], [4], [8], [14], [26]. Apparently, such preliminary maps were used successfully in the precomputer era of conformal mapping, but experiments with a sufficiently flexible computer program for an osculation method have only recently been published by Grassmann [8]. For Example 5.2 (below), his program has been linked with a new general program to eliminate corners.

If Γ is smooth, the discrete Theodorsen equation can also be solved by a combination of Newton's method [4, p. 90] with a linear iterative method, e.g., by Newton–SOR [22], or by Newton–Euler. Though these methods are not treated explicitly here or in [12], it is clear from the general theory [23] that their asymptotic behaviors are similar to those of the corresponding nonlinear iterative methods. They are particularly efficient if the evaluation of ρ is expensive and that of ρ' is not much more expensive.

There are also a number of *related numerical methods* [4, pp. 105–115], [31], [33], for computing the conformal map of the disk D onto the given region Δ. They are all based on the fact that $t \mapsto Y(t) := \operatorname{Im} G(e^{it})$ is the conjugate periodic function KX of $t \mapsto X(t) := \operatorname{Re} G(e^{it})$ if G is analytic in D, continuous in $\bar{D}$, and real valued at 0. (There are also a great number of numerical methods for the inverse map.) If we let $G(w)$ be a suitable branch of $\log [g(w)/w]$, the relation $Y = KX$ becomes

$$\operatorname{Im} \log \{e^{-it} g(e^{it})\} = K[\operatorname{Re} \log \{e^{i\cdot} g(e^{i\cdot})\}](t), \tag{1.6}$$

and in view of (1.0) this leads easily to Theodorsen's integral equation

$$\theta(t) - t = K[\log \rho(\theta(\cdot))](t) \qquad (\forall t \in \mathbb{R}). \tag{1.7}$$

(1.7) is obviously restricted to starlike regions Δ, but one must be aware that relation (1.6) persists if this property does not hold. If we assume that $\Gamma = \{\zeta(\sigma); 0 \leqq \sigma \leqq 1\}$, relation (1.6) in conjunction with the condition $g(e^{it}) = \zeta(\theta(t))$ defining θ becomes upon discretization a nonlinear system of equations for $\theta(t)$. Similarly, one could use $G(w) = g(w)/w$, as in the method of Kulisch and Melentjew [4, pp. 106–109], or $G(w) = g(w)$ as in the methods of Fornberg [33], and Chakravarthy and Anderson [31], or even $G(w) = \log g'(w)$ as basically in the methods of Timman [4, pp. 110–111], Woods [39], James [35], Bauer, Garabedian, Korn, and Jameson [30], which were designed for mapping the exterior of an airfoil profile and allow for an open trailing edge [30], [35], [39] and for various other less idealized situations [35], [39, p. 302]. In all these cases one can apply the FFT. (This was in fact done in [30], [33].)

The discretized Theodorsen equation is particularly simple since it is in a natural way a fixed point equation of a function which is a contraction if $\varepsilon < 1$. Even for $\varepsilon \geqq 1$ the various methods to solve it are well understood if assumption (SD) holds [12]. This is not really true for any of the other methods mentioned.[1] Its drawback—the limitation to starlike domains—can be overcome in practice by preliminary maps, cf. §§ 4 and 5. Among the more recent methods, the one in [31] is not competitive since conjugation is done extremely inefficiently there. (The matrix $\mathbf{A}$ in [31] could be replaced by our $\mathbf{L}$ defined above, and multiplication by $\mathbf{L}$ is an $O(N \log N)$ operation [11].) In contrast, the methods in [30] and [33] are also "fast". They require that Γ be smooth. The first one has the advantage that the derivative of the approximate mapping function never vanishes. Fornberg [33] chooses $G(w) = g(w)$ and solves the nonlinear system (mentioned above) in the frequency domain, i.e., by requiring that the Fourier coefficients of $X + iY$ with negative index vanish. Apart from his different choice for G he determines in principle the same approximate mapping function as we do by solving (1.1) with $\mathbf{K}_\Sigma = \mathbf{K}$, or, generally, the correspondingly discretized version of (1.6). Therefore, our discussion of accuracy in § 7 is also relevant to Fornberg's method and to some of the other methods. (In fact, only a small part of our material is strictly limited to Theodorsen's equation.) It is still an open question which method is best in which situation. But the numerical experiments presented here show that our methods are definitely among the fastest.

2. Numerical experiments on the rate of convergence. The underrelaxation factors $\omega^*(\varepsilon)$ given in [12] for JOR, SOR, and Euler iteration are optimal in the following sense: If assumption (D′) holds and the spectrum of $\mathbf{J} := \mathbf{\Phi}'(\mathbf{y}^*)$ (known to satisfy $\Lambda(\mathbf{J}) \leqq \varepsilon$) is purely imaginary and contains $\pm i\varepsilon$, or if assumption (SD′) holds (possibly up to the monotonicity of ρ) and the spectrum of the restriction $\mathbf{J}|_{\mathcal{S}}$ of $\mathbf{J}$ to $\mathcal{S}$ has these properties, then the spectral radius $\Lambda(\mathbf{J}_\omega)$ of the Fréchet derivative of the respective (nonlinear) iteration function becomes minimal for $\omega = \omega^*$. This means that ultimately the rate of convergence becomes best for $\omega = \omega^*$ (if $\mathbf{y}_m \to \mathbf{y}^*$ at all). Practically, it is important to know whether ω^* is still a good choice if some of these conditions do not hold. In [12] we proved that the spectrum of $\mathbf{J}|_{\mathcal{S}}$ is purely imaginary if $\mathbf{K}_\Sigma = \mathbf{K}$ and (SD′) and (1.3) hold; but we must expect that actually $\Lambda(\mathbf{J}|_{\mathcal{S}}) < \varepsilon$.

[1] However, a new theoretical understanding of Fornberg's method arises from recent results presented by O. Widlund at the Workshop on Computational Problems in Complex Analysis at Stanford (September 1981).

Moreover, we need a starting vector $\mathbf{y}_0$ so that $\mathbf{y}_m \to \mathbf{y}^*$. Many questions may be raised: Is $\mathbf{y}_0 = 0$ good enough? And, if $\mathbf{y}_m \to \mathbf{y}^*$, are we sure that $\mathbf{y}^*$ is a useful solution? How fast is the convergence at the beginning compared with the asymptotic convergence rate?

In this section we describe a few of our experiments with the nonlinear JOR, SOR, and Euler iterations. We gradually relax assumption (SD'). Fortunately, it turns out that the convergence behavior is not much affected.

Experiments with the nonlinear cyclic Chebyshev semi-iteration (CCSI) and with the Chebyshev semi-iteration (CSI) showed that they are usually slightly less efficient than SOR and Euler iteration, respectively. In particular, the first few iterations often yield only a moderate improvement. But there are also examples where CCSI or CSI reach the stopping criterion within fewer iterations.

Example 2.1. The reflected ellipse

$$\rho(\tau) := [1-(1-\alpha^2)\cos^2\tau]^{1/2} \qquad (0<\alpha\leqq 1) \tag{2.1}$$

obtained by reflecting an ellipse with semiaxes $1/\alpha$ and 1 across the unit circle satisfies assumption (SD') with $\nu = 2$. Its exact boundary correspondence function is elementary,

$$\theta(t) = \tan^{-1}(\alpha \tan t), \tag{2.2}$$

and, owing to the fast convergence of the Fourier series of $Y(t) = \theta(t) - t$, high-accuracy results may be obtained with moderate N (unless α is small). These two features also explain the popularity of this example [4], [21], [22], [28].

Some of our results are summarized in Tables 2.1 and 2.2. Except for the last two lines of Table 2.2, a total of $2N = 64$ boundary points have been computed, but due to the symmetries, the number of variables was only 16 (for JOR and Euler) or 32 (for SOR). In addition to $\varepsilon = \frac{1}{2}(1-\alpha^2)/\alpha$, $\omega^* := \omega^*(\varepsilon)$, and the corresponding asymptotic factor of convergence σ^*, we list in Table 2.1 the actual spectral radius $\Lambda(\mathbf{J})$ of $\mathbf{J}$, the true optimal relaxation factor $\omega^{\text{opt}} := \omega^*(\Lambda(\mathbf{J}))$, and the corresponding asymptotic convergence factor σ^{opt}. The subscripts J and S refer to JOR and SOR iteration, respectively. Recall that for the Euler iteration $\omega_E^*(\varepsilon) = \omega_S^*(\varepsilon)$ and $\sigma_E^*(\varepsilon) = [\sigma_S^*(\varepsilon)]^{1/2}$, cf. [12, Thm. 6.2]. Finally, we display

$$i^* := -1/\log_{10}\sigma^*, \tag{2.3}$$

i.e., the number of iterations ultimately needed to get one additional decimal of the result if σ^* were the correct asymptotic factor of convergence. This may be compared in Table 2.2 with the number

$$m(l) := \inf\{m : \|\mathbf{y}_m - \Phi(\mathbf{y}_m)\|_\infty \leqq 10^{-l}\} \tag{2.4}$$

TABLE 2.1
Comparison of nearly optimal and optimal underrelaxation (Ex. 2.1).

α	ε	$\Lambda(\mathbf{J})$	ω_J^*	ω_J^{opt}	σ_J^*	σ_J^{opt}	ω_S^*	ω_S^{opt}	σ_S^*	σ_S^{opt}	i_J^*	i_S^*
.8	.225	.221	.952	.954	.220	.215	.988	.988	.0123	.0119	1.52	.524
.6	.533	.523	.779	.785	.471	.464	.937	.940	.0625	.0604	3.06	.830
.4	1.050	1.030	.476	.485	.724	.717	.816	.821	.184	.179	7.13	1.36
.3	1.517	1.488	.303	.311	.835	.830	.710	.716	.290	.284	12.75	1.86
.2	2.400	2.354	.148	.153	.923	.920	.556	.562	.444	.438	28.77	2.84
.1	4.950	4.549	.039	.047	.980	.979	.331	.354	.669	.646	115.12	5.74
.05	9.975	9.783	.010	.010	.995	.995	.181	.185	.819	.815	460.52	11.50

TABLE 2.2
Actual convergence of JOR, SOR, *and Euler iteration* (*Ex.* 2.1).

N	α	$m_J(3)$	$m_J(6)-m_J(3)$	r_J	e_J
32	.8	3	4	$7.87_{10^{-7}}$	$1.45_{10^{-7}}$
	.6	6	9	$7.35_{10^{-7}}$	$3.18_{10^{-7}}$
	.4	17	20	$8.00_{10^{-7}}$	$3.70_{10^{-7}}$
	.3	30	37	$8.52_{10^{-7}}$	$2.71_{10^{-6}}$
	.2	75	78	$9.83_{10^{-7}}$	$1.14_{10^{-4}}$
	.1	311	317	$8.99_{10^{-7}}$	$6.63_{10^{-3}}$
128	.2	77	82	$8.85_{10^{-7}}$	$3.38_{10^{-7}}$
	.1	331	324	$9.32_{10^{-7}}$	$2.44_{10^{-7}}$
N	α	$m_S(3)$	$m_S(6)-m_S(3)$	r_S	e_S
32	.8	2	2	$5.26_{10^{-9}}$	$3.29_{10^{-10}}$
	.6	3	3	$8.26_{10^{-8}}$	$4.79_{10^{-9}}$
	.4	5	4	$3.42_{10^{-7}}$	$8.95_{10^{-8}}$
	.3	7	5	$7.31_{10^{-7}}$	$2.68_{10^{-6}}$
	.2	10	9	$9.86_{10^{-7}}$	$1.14_{10^{-4}}$
	.1	[31]	[23]	$8.86_{10^{-7}}$	1.12
128	.2	11	8	$8.15_{10^{-7}}$	$3.92_{10^{-7}}$
	.1	[34]	[45]	$6.33_{10^{-7}}$	1.60
N	α	$m_E(3)$	$m_E(6)-m_E(3)$	r_E	e_E
32	.8	3	3	$2.35_{10^{-7}}$	$2.68_{10^{-8}}$
	.6	5	5	$3.32_{10^{-7}}$	$8.13_{10^{-8}}$
	.4	8	8	$8.67_{10^{-7}}$	$2.58_{10^{-7}}$
	.3	12	11	$8.48_{10^{-7}}$	$2.98_{10^{-6}}$
	.2	19	18	$8.22_{10^{-7}}$	$1.14_{10^{-4}}$
	.1	[59]	[46]	$9.15_{10^{-7}}$	1.12
128	.2	20	17	$9.44_{10^{-7}}$	$4.90_{10^{-7}}$
	.1	[∞]		1.91	

of iterations actually needed to satisfy the discrete Theodorsen equation to within a maximum error of 10^{-l} with the JOR (m_J), the SOR (m_S), or the Euler iteration (m_E). Here, $\mathbf{y}_0 = \mathbf{0}$ throughout. Brackets $[\cdot]$ indicate that the resulting vector $\boldsymbol{\theta}_{m(l)+1} := \mathbf{t} + \mathbf{y}_{m(l)+1}$ does not satisfy (1.3). We also list the residual $r = r(m(6))$ generally defined by

$$r(m) := \|\mathbf{y}_m - \boldsymbol{\Phi}(\mathbf{y}_m)\|_\infty. \tag{2.5}$$

(So, $r(m(l)) \leqq 10^{-l}$ by definition.) Finally, $e = e(m(6)+1)$ is the actual error

$$e(m) := \|\mathbf{y}_m - Y(\mathbf{t})\|_\infty \tag{2.6}$$

of the next iterate. (Since the evaluation of $\boldsymbol{\Phi}$ needed in (2.4) adds to the cost, one is inclined to finish the $(m(l)+1)$st step that is in progress.)

Table 2.1 shows that the differences between ε and $\Lambda(\mathbf{J})$, between ω^* and ω^{opt}, and between σ^* and σ^{opt} are small and cannot affect the convergence essentially. Comparing $m(3)$ in Table 2.2 with $3i^*$, we note that at the beginning the JOR iteration is more effective, while the SOR iteration is slightly less effective than suggested by the corresponding asymptotic rate. Nevertheless, SOR is faster than JOR and Euler even at the beginning, and even in the case of small ε. Later its superiority is out of

the question: as predicted by the theory, it then converges twice as fast as Euler, which is itself much faster than JOR.

Surprisingly, for $\alpha = .1$ ($\varepsilon = 4.950$) the JOR method still yields a reasonable solution with $e_J(629) \doteq 8.99 \times 10^{-7}$, while both SOR and Euler converge to a solution violating (1.3) and with an error of about 1.12. Since the difference between r and e shows that the discretization error becomes important when $\alpha \leqq .3$, one might hope—at least if the JOR result were not known—that replacing $N = 32$ by $N = 128$ would lead to a better solution. However, while e_J, e_S, and e_E become in fact essentially smaller for $\alpha = .2$, the SOR iteration still converges for $\alpha = .1$ to a useless solution, and the Euler iteration does not converge at all. An inspection of the Euler data reveals that the odd indexed and the even indexed iterates converge to different limits. So, we actually get a solution of system (5.12) in [12],

$$\mathbf{y}^{(1)} = \mathbf{\Phi}(\mathbf{y}^{(2)}), \qquad \mathbf{y}^{(2)} = \mathbf{\Phi}(\mathbf{y}^{(1)}), \tag{2.7}$$

which has double size. As we noted there, Euler's iteration is equivalent to an SOR iteration for (2.7); and (2.7) may have solutions $\mathbf{y}^{(1)} \neq \mathbf{y}^{(2)}$ that do not satisfy $\mathbf{y} = \mathbf{\Phi}(\mathbf{y})$. In fact, here $\|\mathbf{y}^{(1)} - \mathbf{y}^{(2)}\|_\infty = r_E(\infty) \doteq 1.91$.

We also note that in the three cases where $\theta_{m(l)}$ violates (1.3), the convergence is definitely slower than predicted by i^*. So, the eigenvalues of $\mathbf{J}|_{\mathcal{S}}$ are probably no longer purely imaginary. At least, this is motivated by

LEMMA 1. *Suppose that either assumption* (D′) *or assumption* (SD′) *holds except that* ρ *may be nonmonotone. Assume further that* $\mathbf{J}$ *or* $\mathbf{J}_{\mathcal{S}}$, *respectively, has purely imaginary eigenvalues, and that the iterates* $\mathbf{y}_m$ *created by the nonlinear* SOR *method converge to* $\mathbf{y}^*$. *Finally, let* $\sigma := \sigma_S^*(\Lambda(\mathbf{J}^S))$ *or* $\sigma_S^*(\Lambda(\mathbf{J}^S|_{\mathcal{S}}))$, *respectively, where* $\sigma_S^*(\Lambda) := \Lambda^2/[1+(1+\Lambda^2)^{1/2}]^2$ (*cf.* [12, *Thm.* 5.1]). *Then in any norm*

$$\limsup_{m\to\infty} \frac{\|\mathbf{y}_m - \mathbf{y}^*\|}{\|\mathbf{y}_{m-1} - \mathbf{y}^*\|} \geqq \limsup_{m\to\infty} \|\mathbf{y}_m - \mathbf{y}^*\|^{1/m} = \sigma, \tag{2.8a}$$

$$\limsup_{m\to\infty} \frac{\|\mathbf{y}_{m+1} - \mathbf{y}_m\|}{\|\mathbf{y}_m - \mathbf{y}_{m-1}\|} \geqq \limsup_{m\to\infty} \|\mathbf{y}_{m+1} - \mathbf{y}_m\|^{1/m} = \sigma. \tag{2.8b}$$

The same relations hold for the nonlinear Euler iteration if σ *is replaced by* $\sigma^{1/2}$.

Proof. The inequality in (2.8a) is just [23, Statement 9.3.1]. The equality at right is established as [23, Statement 10.1.4]; additionally, one has to take into account that all eigenvalues of $\mathbf{J}_\omega^S$ have moduli $\sigma = \Lambda(\mathbf{J}_\omega^S)$. In case of assumption (SD′), only the restrictions to $\mathcal{S}$ must be considered [12]. A further inspection of the proofs in [23] shows that (2.8b) also holds, since $\mathbf{J}_\omega^S$ is a strong F-derivative if ρ is continuously differentiable [23]. For Euler's iteration, $(\mathbf{J}_\omega^E)^2 = \mathbf{J}_\omega^S$ [12]. □

Example 2.2. Let Γ consist of the right half of the reflected ellipse (2.1) and of the left half of the ellipse

$$\rho(\tau) = [1 - (1-\alpha^2)\cos^2\tau]^{-1/2} \tag{2.9}$$

with semiaxes α and 1. Assumption (SD′) is nearly satisfied with $\nu = 1$ except that ρ is not monotone. The question is whether this affects the asymptotic rate of convergence. As an indicator we determine

$$q(l) := \left[\frac{\|\mathbf{y}_{m(l)+1} - \mathbf{y}_{m(l)}\|_2}{\|\mathbf{y}_{m(l-1)+1} - \mathbf{y}_{m(l-1)}\|_2}\right]^{1/[m(l)-m(l-1)]}, \tag{2.10}$$

which may be considered as an approximate upper bound for $\Lambda(\mathbf{J}_\omega^S|_{\mathcal{S}})$ or $\Lambda(\mathbf{J}_\omega^E|_{\mathcal{S}})$, respectively, if $\mathbf{J}|_{\mathcal{S}}$ has purely imaginary eigenvalues. ((2.8b) suggests that one compute

$\|\mathbf{y}_{m+1}-\mathbf{y}_m\|_2^{1/m}$ instead, which usually turns out to be larger than $q(l)$, however.) From our experiments with $N=128$ and $\mathbf{y}_0=\mathbf{0}$, we list, on the top three lines of Table 2.3, $q_S(6)$, $q_E^T(6)$, and $q_E^{S3}(6)$, where the superscripts T and S3 denote trigonometric and cubic spline interpolation [13, Ex. 5.1], respectively. Note that q is close to the corresponding σ^* except for $\alpha=0.2$ ($\varepsilon=2.4$), where the SOR solution and the Euler solution with trigonometric interpolation (E/T) yield a solution $\boldsymbol{\theta}$ satisfying (1.3) but having nonmonotone components indicated by braces $\{\cdot\}$. In contrast, third degree spline interpolation (E/S3) leads to a more useful solution.

TABLE 2.3
Convergence in case of a symmetric region with nonmonotone ρ (Ex. 2.2) and in case of an asymmetric region (Ex. 2.3).

Ex.	α	σ_S^*	$m_S(6)$	$q_S(6)$	σ_E^*	$m_E^T(6)$	$q_E^T(6)$	$m_E^{S3}(6)$	$q_E^{S3}(6)$
2.2	.8	.01235	4	.01235	.11111	6	.11114	6	.11113
	.4	.18367	9	.18742	.42857	18	.42897	18	.41575
	.2	.44444	{24}	.48569	.66667	{47}	.70880	39	.68274
2.3	.8	.01235	4	.01282	.11111	6	.11377	6	.11377
	.4	.18367	10	.18580	.42857	19	.42618	20	.42057
	.2	.44444	23	.54849	.66667	44	.70252	43	.66878

Example 2.3. Let Γ consist of the left half of the unit circle, of the part of the reflected ellipse (2.1) in the first quadrant, and of the part of the ellipse (2.9) in the fourth quadrant. So, Γ is again a piecewise analytic curve with straight angles, but now it is asymmetric. Nevertheless, we use $\omega=\omega^*(\varepsilon)$ as defined for symmetric curves. Results are listed on the lower three lines of Table 2.3. ($N=128$ and $\mathbf{y}_0=\mathbf{0}$ again.) Here, the components of $\boldsymbol{\theta}_{m(6)}$ are always monotone, and only $q_S(6)$ in the case $\alpha=.2$ indicates a slower convergence. Yet, $q_E^T(6)$ turns out to be close to σ_E^*, although the iterates tend to the same solution as with the SOR method. On the other hand, $q_S(9)\doteq .38048$, and an experiment with an alternate starting point than $\mathbf{y}_0=\mathbf{0}$ yields $q_S(6)\doteq .49168$, $q_S(9)\doteq .38513$. We must conclude that in some examples $q(l)$ is not a reliable estimate of $\Lambda(\mathbf{J}_\omega)$.

Example 2.4. A square with the center at the origin satisfies assumption (SD'). As we will see in § 7 trigonometric interpolation of $\log\rho(\mathbf{t}+\mathbf{y}^*)$ leads to an approximate mapping function g_N for which $g_N(\partial D)$ has ripples. We use this example to demonstrate other approximation techniques. In addition to the trigonometric interpolation (T), we consider Cesàro (C) and Lanczos (L) smoothing [13, Ex. 6.1], interpolation by a first degree spline [13, Ex. 5.1], and smoothing by a first degree and by a cubic spline, both with smoothing parameter $\tilde{\rho}=10^4$ (S1/10^4 and S3/10^4) [13, Ex. 6.2]. See Table 2.4. Surprisingly, the data $m(6)$, $r(m(6))$, and $q_E(m(6))$ are nearly the same in all cases except for trigonometric interpolation, where it takes one iteration more to satisfy

TABLE 2.4
Convergence of iterates based on various types of approximation (Ex. 2.4).

	T	C	L	S1	S1/10^4	S3/10^4
$m(6)$	15	14	14	14	14	14
$r(m(6))\times 10^6$	.51198	.42876	.43405	.43412	.43353	.43272
$q_E(6)$	.41854	.41404	.41416	.41418	.41420	.41331

the stopping criterion $r(m(6)) \leqq 10^{-6}$ and where $r(m(6))$ is nevertheless larger than for the other five.

To enable the reader to estimate the computing time of any of the examples we list in Table 2.5—as a function of N—the time (in seconds) for *one* JOR or Euler iteration step with the boundary function $\rho(\tau) = \exp(\alpha \cos \beta\tau)$. For comparison we measured this time also when **Kx** was computed by multiplication with Wittich's matrix—as suggested in earlier texts—instead of by applying the FFT. In both cases it is possible to take advantage of symmetries, cf. [11]. The last two columns show the time for one step with axial symmetry.

TABLE 2.5
Computing time for one JOR *or one Euler iteration step (seconds on* CDC 6400/6500).

N	no symmetry		axial symmetry	
	Wittich	FFT	Wittich	FFT
16	.03	.04	.02	.02
32	.09	.08	.05	.04
64	.30	.15	.16	.08
128	1.04	.31	.55	.16
256	3.72	.64	2.08	.34
512	14.06	1.34	8.04	.69
1,024	54.41	2.79	31.05	1.46
2,048		6.02		3.00

(These execution times have been measured on the CDC 6400/6500 at Eidgen. Tech. Hochschule, Zurich, where an addition or subtraction takes 1.1 μsec and a multiplication or division, 5.7 μsec. The FFT program used here was a general purpose ALGOL60 implementation of the radix-2 algorithm and thus not most efficient for conjugation, where permutation of data to reverse binary order could be omitted. Most of our other experiments were done instead with a hand-optimized version of Singleton's FORTRAN mixed radix FFT program [24], which proved to be essentially faster in case N is a large power of 2.)

These execution times are hardly affected if **K** is replaced by a more general conjugation process $\mathbf{K}_\Sigma$, but the preparations may cause some overhead (cf. [13]).

The computing time for one SOR iteration is essentially the same too if the program published in [11] is used for the conjugation; however, one cannot capitalize on axial symmetry easily.

3. Continuation methods. There is strong experimental evidence that even for quite simple boundaries Γ and a suitable N, the discrete Theodorsen equation (1.1) may have more than one solution. In this section we state conditions assuring the existence of a unique special solution, which is also the only reasonable one. This solution is obtained from the unique solution for $\Gamma = \partial D$ (viz., $\mathbf{y}^* = \mathbf{0}$) by a homotopy. Practically, it can therefore be computed by means of a continuation method [23, pp. 230–234]. For the theory we either need assumption (D′) or assumption (SD′).

Under assumption (D′) we consider the following initial value problem:

$$\mathbf{y}'(\eta) = [\mathbf{I} - \eta\Phi'(\mathbf{y}(\eta))]^{-1}\Phi(\mathbf{y}(\eta)), \tag{3.1a}$$

$$\mathbf{y}(0) = \mathbf{0}. \tag{3.1b}$$

If we assume that $\mathbf{I}-\eta\Phi'(\mathbf{y}(\eta))$ remains regular for $0 \leqq \eta < \eta_0$, it is easy to conclude that the solution of (3.1) exists there, is continuously differentiable and satisfies

$$\mathbf{y}(\eta) = \eta\Phi(\mathbf{y}(\eta)), \tag{3.2a}$$

i.e.,

$$\mathbf{y}(\eta) = \mathbf{K}_\Sigma \log \rho^\eta(\mathbf{t}+\mathbf{y}(\eta)), \tag{3.2b}$$

cf. [23, p. 233]. (In fact, (3.1a) is obtained by differentiating (3.2a).) So, $\mathbf{y}(\eta)$ is the solution of the discrete Theodorsen equation corresponding to the boundary Γ_η with the polar coordinates τ, $\rho^\eta(\tau)$. In particular, if $\eta_0 > 1$, $\mathbf{y}(1)$ is a solution of the given equation (1.1).

By [12, Thm. 3.1], we know that $\mathbf{y}(\eta)$ is the only solution of (3.2) as long as $\eta \in [0, 1/\varepsilon)$. Along the path $\eta \mapsto \mathbf{y}(\eta)$ this solution can be continued uniquely until $\mathbf{I}-\eta\Phi'(\mathbf{y}(\eta))$ becomes singular. This does not exclude the existence of other solutions in case $1/\varepsilon \leqq \eta < \eta_0$, nor does it guarantee the monotonicity of the components of $\boldsymbol{\theta}(\eta) := \mathbf{t}+\mathbf{y}(\eta)$.

Under assumption (SD'), $\Phi(\mathcal{S}) \subset \mathcal{S}$ (cf. [12, App. 2]). If $\mathbf{y} \in \mathcal{S}$ satisfies (1.3) and if $\mathbf{K}_\Sigma = \mathbf{K}$, the eigenvalues of $\Phi'(\mathbf{y})|_{\mathcal{S}}$ are purely imaginary [12, Thm. 3.4], and hence the restricted operator

$$[\mathbf{I}-\eta\Phi'(\mathbf{y})]|_{\mathcal{S}} \tag{3.3}$$

is regular and defines an automorphism of $\mathcal{S}$. Consequently, $\mathbf{y}(\eta) \in \mathcal{S}$, $0 \leqq \eta < \eta_0$, and the path $\eta \mapsto \mathbf{y}(\eta)$ is well defined as long as $\mathbf{y}(\eta)$ satisfies (1.3).

In this case of symmetry, Hübner [16] proposed to replace ρ by

$$\hat{\rho}(\tau) := \begin{cases} \rho(0) & \text{if } \tau \leqq 0, \\ \rho(\tau) & \text{if } 0 < \tau < \pi/\nu, \\ \rho(\pi/\nu) & \text{if } \tau \geqq \pi/\nu, \end{cases} \tag{3.4a}$$

when $\rho(t_k + y_k)$, $k = 0, \cdots, N/\nu$, is evaluated, while the other components of $\hat{\rho}(\mathbf{t}+\mathbf{y})$ are determined by symmetry. We let

$$\hat{\boldsymbol{\Phi}}(\mathbf{y}) := \mathbf{K} \log \hat{\rho}(\mathbf{t}+\mathbf{y}) \qquad (\forall \mathbf{y} \in \mathcal{S}). \tag{3.4b}$$

By [12, App. 2, Lemma 2] $\hat{\boldsymbol{\Phi}}'(\mathbf{y})|_{\mathcal{S}}$ has purely imaginary eigenvalues. By the general Theorem 5.3.9 in [23], the mapping $\mathbf{y} \mapsto \mathbf{y} - \eta\hat{\boldsymbol{\Phi}}(\mathbf{y})$ is a homeomorphism of $\mathcal{S}$ (for fixed η); thus, exactly one point $\mathbf{y} = \hat{\mathbf{y}}(\eta)$ is mapped onto $\mathbf{0}$. This is an alternate proof of

THEOREM 2 (Hübner [16]). *The equation* $\mathbf{y} = \eta\hat{\boldsymbol{\Phi}}(\mathbf{y})$ *has exactly one solution* $\hat{\mathbf{y}}(\eta)$. *If it satisfies* (1.3), *it is also a solution of* (3.2); *otherwise*, (3.2) *has no solution satisfying* (1.3).

Hübner's original proof is based on the above-mentioned Lemma 2 and Brouwer's fixed point theorem; it only requires that $\hat{\rho}$ be absolutely continuous, while we need $\hat{\rho}$ continuously differentiable in the above proof.

Of course, $\hat{\mathbf{y}}(\eta)$ satisfies (3.1) with Φ replaced by $\hat{\boldsymbol{\Phi}}$. So, $\hat{\mathbf{y}}(1)$ could be computed by any standard numerical method for solving initial value problems. However, evaluating $\hat{\mathbf{y}}'$ would require calculating $\hat{\boldsymbol{\Phi}}'$ (i.e. ρ'), and we do not consider such methods in this paper. Likewise, $\mathbf{y}(1)$ could be computed through solving (3.1) if $\mathbf{I}-\eta\Phi'(\mathbf{y}(\eta))$ is regular for $0 \leqq \eta \leqq 1$.

The following continuation method [23, § 10.4], which does not require evaluating ρ', is another simple means for computing $\mathbf{y}(1)$ or $\hat{\mathbf{y}}(1)$: Let

$$0 = \eta_0 < \eta_1 < \eta_2 < \cdots < \eta_I = 1 \tag{3.5}$$

be a partition of $[0, 1]$. Denote by $\Phi_{\omega,\eta}$ the iteration function defining one step of either the JOR, the SOR, or the Euler iteration applied to the fixed point equation $\mathbf{y} = \eta\Phi(\mathbf{y})$ (i.e., to the boundary Γ_η). Then, define the iterates $\mathbf{y}_{i,m}$ by

$$
\begin{aligned}
&\mathbf{y}_{1,0} := \mathbf{0}, \\
&\mathbf{y}_{i+1,0} := \mathbf{y}_{i,m_i+1}, \qquad i = 1, \cdots, I-1, \\
&\mathbf{y}_{i,m+1} := \Phi_{\omega^*(\varepsilon\eta_i),\eta_i}(\mathbf{y}_{i,m}), \qquad m = 0, \cdots, m_i, \quad i = 1, \cdots, I-1, \\
&\mathbf{y}_{I,m+1} := \Phi_{\omega^*(\varepsilon),1}(\mathbf{y}_{I,m}), \qquad m = 0, 1, 2, \cdots.
\end{aligned} \tag{3.6}
$$

If (SD′) holds, a variant 2 of this algorithm may be defined by our replacing ρ by $\hat{\rho}$ and $\mathbf{K}_\Sigma$ by $\mathbf{K}$, i.e., Φ by $\hat{\Phi}$. A general theorem by Avila [23, 10.4.1] and the theory in [12] yield

THEOREM 3. *Under assumptions* (SD′) *and* (D′), *there exist a partition* (3.5) *and integers* $m_1, \cdots, m_{I-1}$ *such that the variant* 2 *of algorithm* (3.6) *yields a sequence* $\{\mathbf{y}_{I,m}\}$ *converging to the fixed point* $\hat{\mathbf{y}}(1)$ *of* $\hat{\Phi}$.

Of course, if $\hat{\mathbf{y}}(\eta)$ satisfies (1.3) for $0 \leqq \eta \leqq 1$, we may as well use variant 1. Moreover, variant 1 proved practically useful even for asymmetric boundaries Γ. In any case one has to choose the partition (3.5) and the integers m_i somehow. A risky choice may lead to no solution or the wrong one; a too cautious choice is expensive.

Our experiments have been done with the following values, which depend on two parameters I_0 and l_0:

$$
\begin{aligned}
&I := \lfloor I_0\varepsilon \rfloor + 1, \qquad \eta_i := i/I, \\
&m_i := m_i(l_0) := \inf\{m : \|\mathbf{y}_{i,m} - \eta_i\Phi(\mathbf{y}_{i,m})\|_\infty \leqq 10^{-l_0}\}, \qquad i = 1, \cdots, I-1.
\end{aligned} \tag{3.7}
$$

Example 3.1. In Example 2.1 (cf. Table 2.2), we always obtain the unique fixed point of $\hat{\Phi}$ except when $\alpha = .1$, in which case SOR and Euler iteration applied to (1.1) converge to another fixed point of Φ, while the JOR method still yields the common fixed point of Φ and $\hat{\Phi}$. However, we can compute this fixed point now more efficiently with the continuation method (3.6) (variant 1 or 2) based on SOR or Euler. Table 3.1 shows for the SOR method with $N = 128$, variant 1 of (3.6), and the two problems with $\alpha = .1$ and $\alpha = .05$, respectively, the total number

$$
M_P := \sum_{i=1}^{I-1} [m_i(l_0) + 1] \tag{3.8}
$$

TABLE 3.1
Number of steps required by the continuation method (Ex. 3.1: reflected ellipse).

		$\alpha = .1$ ($\varepsilon = 4.950$)			$\alpha = .05$ ($\varepsilon = 9.975$)		
I_0	l_0	I	M_P	$M_P + m_S(6)$	I	M_P	$M_P + m_S(6)$
.5	1	3	[11]	46	5	[51]	[ ]
.5	2	3	16	51	5	[ ]	[ ]
.5	3	3	22	57	5	[ ]	[ ]
1	1	5	[17]	50	10	[45]	[124]
1	2	5	28	61	10	97	159
1	3	5	38	71	10	149	211
2	1	10	[27]	58	20	[65]	126
2	2	10	53	84	20	168	228
2	3	10	80	111	20	275	336

of evaluations of Φ required to determine the starting point $\mathbf{y}_{I,0}$ of the last secondary iteration in (3.6), and the total number $M_P + m_S(6)$ of steps done until $\mathbf{y}_{I,m}$ satisfies (2.4) with $l = 6$. For $\alpha = .1$ we always end up with the correct solution, although $\mathbf{y}_{I,0}$ violates (1.3) if $l_0 = 1$. However, if $\alpha = .05$ and $I_0 = .5$, the path $\mathbf{y}(\eta)$ is always left, and this still happens for $\alpha = .05$, $I_0 = 1$, $l_0 = 1$. On the other hand, in case of convergence to the correct solution, the number of steps required varies strongly.

Example 3.2. Theorem 2 does not apply to Example 2.2, and so there could exist another solution with monotone $\boldsymbol{\theta}$ in the case $\alpha = .2$ (cf. Table 2.3). However, our experiments with the continuation method have always delivered the same solution. Moreover, $N = 512$ and $\mathbf{y}_0 = \mathbf{0}$ immediately lead to a monotone $\boldsymbol{\theta}$ (see Ex. 7.2).

These two examples show that further investigations on the appropriate choice of N, I, the partition $\{\eta_i\}$ and the numbers m_i of secondary iterations are worthwhile. Ultimately one needs a programmable strategy containing feedback loops for the determination of these parameters.

4. The behavior of the boundary correspondence function at a corner of a piecewise analytic boundary. While many other constructive methods for conformal mappings [4] require that the boundary Γ be differentiable or need to be modified to allow for corners, Theodorsen's method is less restrictive (cf. [12, Thms. 4.3, 7.3]). To be sure, for some results on local convergence, we had to assume in [12] that ρ' exists (except at points on a symmetry axis). But as we pointed out, differentiability seems not to be crucial for the local convergence in practice. However, if the approximation of $\log \rho(\theta(t))$ is done by trigonometric interpolation, corners strongly infect the accuracy of the resulting approximate values of the boundary correspondence function θ. In order to select a more appropriate class of approximants, we cite some results on the behavior of θ. We restrict ourselves to the most important case, namely to a piecewise analytic boundary Γ.

Assume first that Δ is a Jordan region with $0 \in \Delta$ and a piecewise analytic boundary Γ. Let $g : \bar{D} \to \bar{\Delta}$ be the continuous extension of a conformal map with $g(0) = 0$, $g'(0) > 0$ of the unit disk D onto Δ. Denote by $\zeta_j := g(w_j)$, $j = 1, \cdots, J$, the corners (or, rather, the breakpoints), which are assumed to be regular points of the analytic arcs meeting there under the inner angles $\pi\alpha_j$, where $0 < \alpha_j \leqq 2$. Let $\zeta_0 = g(w_0)$ be any of these corners, $\pi\alpha$ the corresponding angle, and set $h := w - w_0$. Then, according to a theorem by Lehman [19] generalizing earlier work of Lichtenstein, Kellog, Warschawski, and Lewy, the following asymptotic expansion holds for $h \to 0$ if $\arg h$ remains bounded for some continuous branch of the argument:

$$
\begin{aligned}
g(w_0 + h) &\sim \zeta_0 + \sum_{k,l,m} a_{klm} h^{k+l\alpha} (\log h)^m \\
&= \zeta_0 + a_{010} h^{\alpha} + a_{111} h^{1+\alpha} \log h + a_{110} h^{1+\alpha} + O(h^{2\alpha}),
\end{aligned} \tag{4.1}
$$

where $a_{010} \neq 0$, and the sum runs over

$$
k \geqq 0, \quad l \geqq 1, \quad m = 0 \quad \text{if } \alpha \text{ is irrational,}
$$

$$
k \geqq 0, \quad 1 \leqq l \leqq L, \quad 0 \leqq m \leqq \frac{k}{M} \quad \text{if } \alpha = \frac{M}{L} \text{ is rational.}
$$

(M and L have no common divisor, and coefficients a_{klm} not needed are set at 0.) The terms in (4.1) are supposed to be arranged in appropriate order, which may deviate from the one shown explicitly. The expansion for g' is obtained by termwise differentiation and subsequent reordering [19]. For a simple deduction of the dominant term

in (4.1) see Warschawski [27], who, however, discusses the inverse map also covered by Lehman [19]. In case of a straight angle, i.e., $\alpha = 1$, (4.1) reduces to Lewy's expansion [20].

As in the usual deduction of Theodorsen's integral equation [4, p. 64], we note that there exists a continuous branch of $\log[g(w)/w]$ in $\bar{D}$ which takes the real value $\log g'(0)$ at 0. Consequently, the real continuous 2π-periodic functions X and Y defined by

$$X(t) + iY(t) := \log[e^{-it} g(e^{it})] \qquad (\forall t \in \mathbb{R}) \tag{4.2}$$

are conjugate periodic functions: $Y = KX$.

In particular, if we now assume that Γ is starlike with respect to 0 and given by its polar coordinate τ, $\rho(\tau)$, then $\theta(t) := Y(t) + t$ is the boundary correspondence function satisfying $g(e^{it}) = \rho(\theta(t))\, e^{i\theta(t)}$. Moreover, $\log \rho(\theta(t)) = X(t)$, and $Y = KX$ becomes Theodorsen's integral equation (1.7).

Whenever θ' exists, (4.2) yields

$$\theta'(t) = \operatorname{Im} \frac{ie^{it} g'(e^{it})}{g(e^{it})} = \operatorname{Re} \frac{e^{it} g'(e^{it})}{g(e^{it})}. \tag{4.3}$$

If we let $w = e^{it}$, $w_0 = e^{it_0}$ and insert (4.1) and the termwise differentiated series into (4.3), we get for $t \to t_0$ in view of $h = e^{it} - e^{it_0} = i(t - t_0)w_0 + O(|t - t_0|^2)$ and by choosing $\arg[i(t - t_0)w_0] = \arg(e^{it} - e^{it_0}) + O(|t - t_0|^2)$,

$$\theta'(t) \sim \operatorname{Re}\{b_{010}[(t - t_0)iw_0]^{\alpha - 1}\} = O(|t - t_0|^{\alpha - 1}), \tag{4.4}$$

where $b_{010} := \alpha a_{010} w_0 / g(w_0) \neq 0$, and similarly

$$\theta''(t) = O(|t - t_0|^{\alpha - 2}). \tag{4.5}$$

If $\alpha \neq 1$ and $\alpha \neq 2$, then $\operatorname{Re}\{\cdots\} \neq 0$, and the order stated is actually attained. In case $\alpha = 1$ we get

$$\theta'(t) \to \operatorname{Re} b_{010}, \qquad \theta''(t) = O(\log|t - t_0|), \tag{4.6}$$

while for $\alpha = 2$ we have $\operatorname{Re}\{b_{010} h^{\alpha - 1}\} = O(|t - t_0|^2)$, since Γ is starlike, so that even

$$\begin{aligned} \theta'(t) &= O(|t - t_0|^2 \log|t - t_0|), \\ \theta''(t) &= O(|t - t_0| \log|t - t_0|), \\ \theta'''(t) &= O(\log|t - t_0|). \end{aligned} \tag{4.7}$$

By making use of $K(H^m) \subset H^m$ [13, Lemma 9.1] and the fact that KY equals $-X$ up to a constant, we finally conclude:

$$\begin{aligned} &\theta,\ Y \text{ are absolutely continuous,} \\ &\theta',\ Y' \text{ are absolutely continuous iff } \alpha_j \geqq 1\ (\forall j), \\ &X,\ Y \in H^1 \text{ iff } \alpha_j > \tfrac{1}{2}\ (\forall j), \\ &X,\ Y \in H^2 \text{ iff } \alpha_j > \tfrac{3}{2} \text{ or } \alpha_j = 1\ (\forall j), \\ &X,\ Y \in H^3 \text{ iff } \alpha_j = 2\ (\forall j). \end{aligned} \tag{4.8}$$

As we have shown in [13], the function KS conjugate to the interpolating spline of degree $2m - 1$ is in a certain sense an optimal approximation for KX if only the values $X(t_k)$ and the information $X \in H^m$ are given. (4.8) allows us to choose the appropriate degree of the spline. In particular, we conclude that interpolating splines are not

appropriate if $\alpha_j \leqq \frac{1}{2}$ at some corner. However, smoothing splines and other smoothing approximants proved quite successful in our experiments, which are partly described in § 7. In case $\alpha_j \geqq 1$ $(\forall j)$ one might also try quadratic spline interpolation (with knots halfway between the data points).

5. Elimination of corners by preliminary maps. Under the initial assumptions of § 4, the mapping

$$\gamma_{\zeta_0,\alpha}: \zeta \in \bar{\Delta} \mapsto \hat{\zeta} := \zeta_0[1-(1-\zeta/\zeta_0)^{1/\alpha}] \tag{5.1}$$

offers itself as a means for eliminating the corner with interior angle $\alpha\pi (0<\alpha\leqq 2)$ at ζ_0. Of course, a continuous argument of $1-\zeta/\zeta_0$ has to be chosen in $\bar{\Delta}\backslash\{\zeta_0\}$ for the definition of $\gamma_{\zeta_0,\alpha}$, and we require that it be 0 at $\zeta = 0$. Then

$$\gamma_{\zeta_0,\alpha}(0) = 0, \qquad \gamma'_{\zeta_0,\alpha(0)} = \frac{1}{\alpha} > 0. \tag{5.2}$$

Moreover, $\gamma_{\zeta_0,\alpha}(\zeta_0) = \zeta_0$ and, if $\gamma_{\zeta_0,\alpha}(\bar{\Delta})$ is placed on a Riemann surface,

$$\gamma_{\zeta_0,1/\alpha} \circ \gamma_{\zeta_0,\alpha}(\zeta) = \zeta \qquad \forall \zeta \in \bar{\Delta}. \tag{5.3}$$

If the projection of $\gamma_{\zeta_0,\alpha}(\bar{\Delta})$ on $\mathbb{C}$ is not injective (as it may happen if $\alpha < 1$), a preliminary map $\gamma_{\zeta^*,1/\alpha}$ with a suitably chosen $\zeta^* \notin \bar{\Delta}$ usually enables us to avoid this situation. One could also replace $\gamma_{\zeta_0,\alpha}$ by a single but more complicated Kármán–Trefftz transformation,

$$\frac{\hat{\zeta}-\zeta_0}{\hat{\zeta}-\zeta^*} = \left(\frac{\zeta-\zeta_0}{\zeta-\zeta^*}\right)^{1/\alpha} \qquad \forall \zeta \in \bar{\Delta} \tag{5.4}$$

in this case, which occurs in practice less often than one would expect (cf. [36]). So, by a composition of conformal maps of the type (5.1), it is very often possible to eliminate all corners of Γ and to end up with a region Δ^* with boundary Γ^* in the plane. Yet, Γ^* is piecewise analytic only in a weaker sense, since its breakpoints are in general singular points of the analytic arcs joining there.

For particular applications the map (5.1), and many other preliminary conformal maps such as the Joukowski and the Kármán–Trefftz transformation, have been used for a long time (see [4, p. 257], [17], [26], [29]). For example, the success of Theodorsen's method in airfoil design would have been impossible without them [6]. But we would like to point out that one can nowadays develop a general and fairly reliable computer program for the successive elimination of all corners of many piecewise analytic curves. The main programming problem is choosing the correct argument of $1-\zeta/\zeta_0$. Another approach requiring the evaluation of indefinite integrals was described by Landweber and Miloh [18], but the composition of maps (5.1) is much simpler, its range of application is wider, and the inverse map is of the same type. It does not take care of axial symmetry, but the Kármán–Trefftz transformation could be applied instead of (5.1) to eliminate two symmetric corners at once. At the end, it is often profitable to apply additionally a Moebius transform T that keeps the origin fixed and maps the smallest circle containing Δ^* onto the unit disk. (A corresponding algorithm is described in [9].)

If the resulting region $T(\Delta^*)$ is not suitable for Theodorsen's method (e.g., because either it is not starlike or ε_ρ is too large) one could apply further auxiliary conformal maps such as those proposed by Koebe, Ringleb, Heinhold, and Albrecht for osculation methods (see [2], [4], [14], [26] and references given there). Grassmann [8] has reported on a recent computer implementation of such an osculation method. Our

examples given below (and many others) do not require such further preparatory maps, but it might be worthwhile to apply a few of them. The main handicap put by preliminary maps on the subsequent use of Theodorsen's method is that the boundary Γ^* [or $T(\Gamma^*)$] is no longer given in polar coordinates. So, one has to map a sufficient number of boundary points first, say the points $\zeta_{j,k}$, $k=0,\cdots,k_j$, on the arc from $\zeta_j=\zeta_{j,0}$ to $\zeta_{j+1}=\zeta_{j+1,0}=\zeta_{j,k_j}$, $j=1,\cdots,J$ (where $\zeta_{J+1}:=\zeta_1$). We denote the image points by $\zeta^*_{k,j}$, and define $\arg\zeta^*$ continuously on Γ^* (assumed to be starlike with respect to 0) such that $\arg\zeta_1^*\leqq\arg\zeta^*\leqq\arg\zeta_1^*+2\pi$. Then, we propose to interpolate the data $(\arg\zeta^*_{j,k},\log|\zeta^*_{j,k}|)$, $k=0,\cdots,j_k$, by a cubic spline s_j with double knots at the end points $\arg\zeta_j$ and $\arg\zeta_{j+1}$ (i.e. s_j interpolates there the derivative too). The global interpolation function

$$s(\tau):=s_j(\tau)\quad\text{if }\arg\zeta_j\leqq\tau\leqq\arg\zeta_{j+1}\quad(j=1,\cdots,J) \tag{5.5}$$

replaces the function $\log\rho(\tau)$ in (1.1). Note that the $2N$ evaluations of s required afterwards in each iteration step for solving (1.1) are very cheap since the coefficients of s are computed only once.

One would like the abscissae $\arg\zeta^*_{k,j}$ roughly equidistant, but unfortunately the maps (5.1) cause strong distortions in arc length. When choosing $\zeta_{j,k}$, $k=1,\cdots,k_j-1$, it is at least possible to take the distortions at the two nearby breakpoints ζ_j^* and ζ_{j+1}^* approximately into account. In fact, $\zeta_{j,k}$ can be chosen in such a way that the points $\zeta^*_{j,k}$, $k=0,\cdots,k_j$, would be equidistant if the arc from ζ_j to ζ_{j+1} were straight and these two corners were the only ones.

Of great importance to Theodorsen's method is the effect of the mapping (5.1) on the behavior of the boundary correspondence θ at some corner ζ_0. From (4.1) we get

$$\begin{aligned}\hat g(w_0+h)&:=\zeta_0\{1-[1-g(w_0+h)/\zeta_0]^{1/\alpha}\}\\&\sim\zeta_0+(-\zeta_0)^{1-1/\alpha_h}\Big[\sum_{k,l,m}a_{klm}h^{k+l\alpha-\alpha}(\log h)^m\Big]^{1/\alpha}\\&=\zeta_0+(-\zeta_0)^{1-1/\alpha}h\{a_{010}^{1/\alpha}+O(h^\alpha)+O(h\log h)\},\end{aligned} \tag{5.6}$$

$$\hat g'(w_0+h)=(-\zeta_0)^{1-1/\alpha}a_{010}^{1/\alpha}+O(h^\alpha)+O(h\log h), \tag{5.7}$$

$$\hat g''(w_0+h)=O(h^{\alpha-1})+O(\log h)\quad\text{as } h\to 0. \tag{5.8}$$

If α is irrational, $\log h$ can be replaced by 1 here.

If we eliminate all corners of Γ by preliminary maps of the type (5.1) and end up with a region Δ^* whose boundary Γ^* is starlike with respect to the origin, then—by the arguments used in § 4—the boundary correspondence function θ^* of a conformal map $g^*:D\to\Delta^*$ with $g^*(0)=0$ and the related functions $Y^*(t):=\theta^*(t)-t$, $X^*(t):=\log\rho^*(\theta^*(t))$ satisfy in analogy to (4.8)

$$\begin{aligned}&d\theta^*/dt,\ dY^*/dt\quad\text{are absolutely continuous,}\\&X^*,\ Y^*\in H^1,\\&X^*,\ Y^*\in H^2\text{ if }\alpha_j>\tfrac12(\forall j).\end{aligned} \tag{5.9}$$

So, approximating X^* by a cubic spline is appropriate if $\alpha_j>\frac12$ at every breakpoint of the piecewise analytic boundary Γ; otherwise, one may use a broken line interpolant, a slightly smoothed cubic spline approximant [13], a smoothed trigonometric interpolant, or, maybe, a quadratic spline interpolant.

Example 5.1. To the Swiss cross we first apply the map $\zeta \mapsto \zeta^4$ to account for the fourfold rotational symmetry. The resulting domain in Fig. 5.1(a) has three corners, which are removed by a composition of three maps of type (5.1) (see Fig. 5.1(b)–(d)). (The scale in these figures is not constant. The small squares mark removed corners.) An additional Moebius transform yields a boundary $T(\Gamma^*) \subset \bar{D}$ having three points in common with ∂D (cf. Fig. 5.1 (e)). Its point (marked by a square) nearest to 0 has modulus $\mu \doteq .903$, and ε is obviously small, so that a high accuracy approximation of $D \to T(\Delta^*)$ is easily computed by Theodorsen's method.

Example 5.2. The L-shape of Fig. 5.2(a) has six corners to eliminate (cf. Fig. 5.2(b)–(g)). After the Moebius transform, $\mu \doteq .593$ (see Fig. 5.2(h)). Twelve steps with Grassmann's osculation method program would lead to a boundary with $\mu \doteq .981$, but the inverse image of the unit circle under the corresponding composition of elementary transforms still deviates considerably from the given

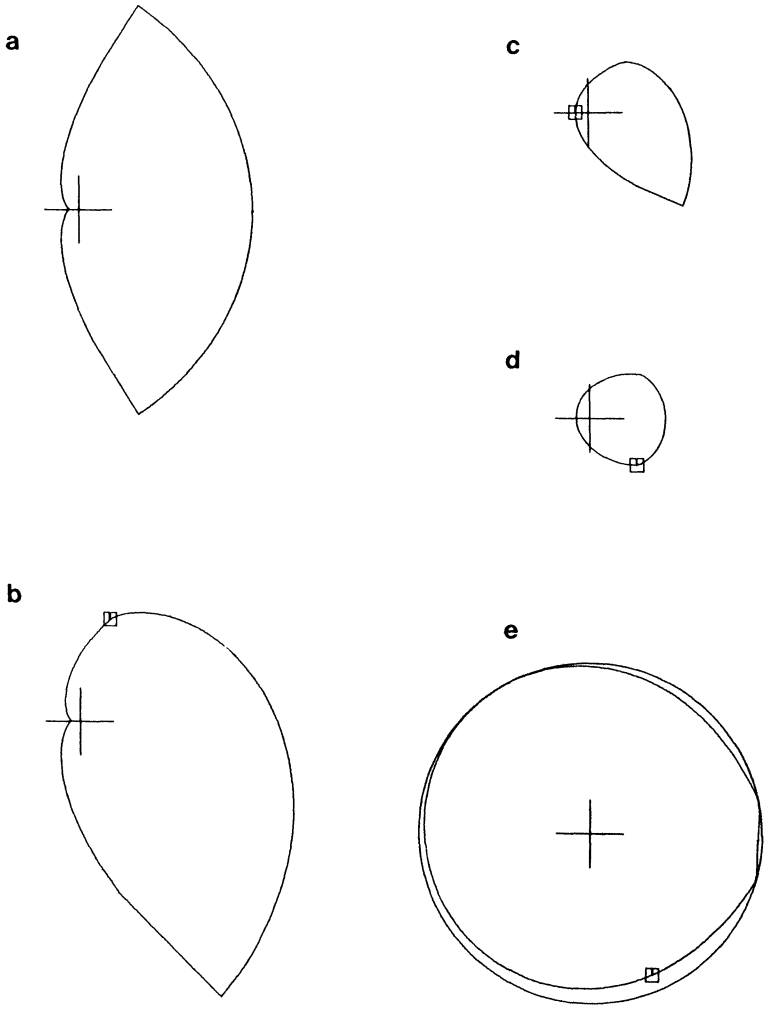


FIG. 5.1. *Elimination of the corners of a Swiss cross preliminarily mapped by* $\zeta \to \zeta^4$. *The three corners in* (a) *are removed one after another, cf.* (b)–(d), *then a Moebius transform is used to fit the region into the unit disk. The squares mark removed corners and, in* (e), *the point nearest to* 0, *respectively.*

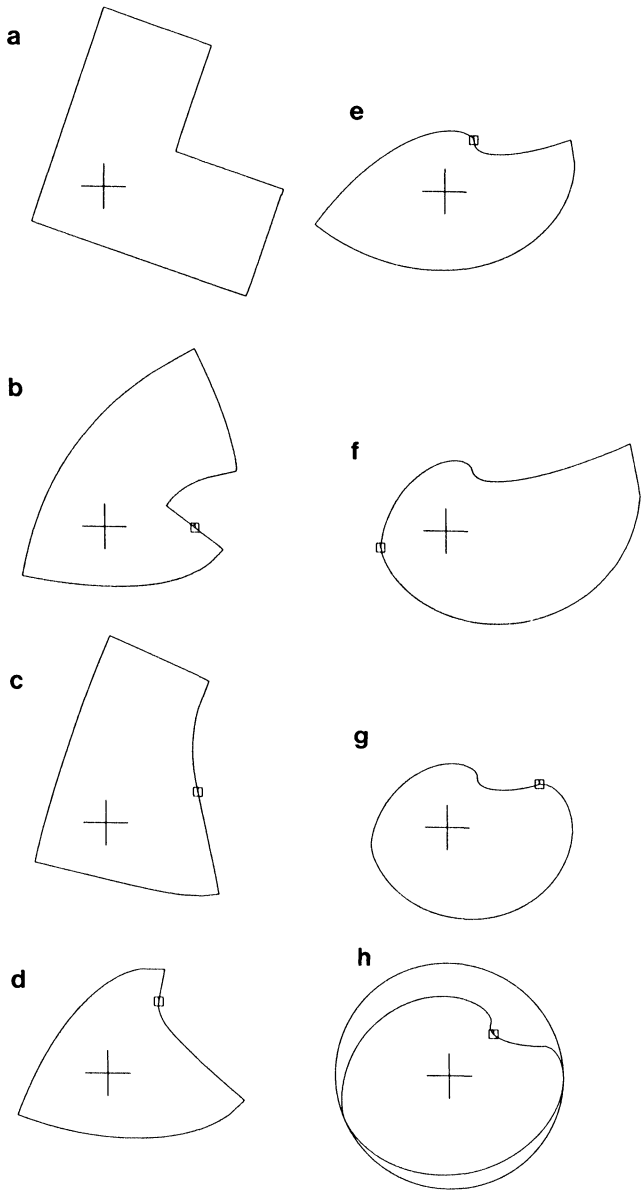


FIG. 5.2. *Elimination of the corners of an L-shape.*

L-shape. See Fig. 5.3, where the inverse image of 1,000 equidistant points on the unit circle is plotted. So, one still has to apply a more accurate method afterwards, such as e.g., Theodorsen's one, which can also be applied to the domain of Fig. 5.2(h) directly.

6. The evaluation of the mapping function and its derivative. Assume that $\mathbf{y}$ is a solution of the (original or modified) discrete Theodorsen equation (1.1). Then the components of $\boldsymbol{\theta} := \mathbf{t} + \mathbf{y}$ are approximate values of the boundary correspondence function θ. But how can we compute an approximate value of the conformal map $g : \bar{D} \to \bar{\Delta}$ at an arbitrary point $w \in \bar{D}$?

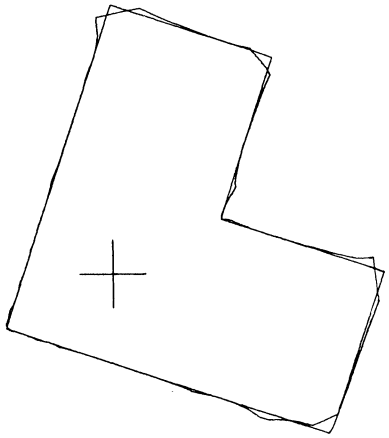


FIG. 5.3. *Inverse of the conformal mapping composed of six eliminations of corners, a Moebius transform, and twelve osculation maps.*

Again, denote by $P: \Pi_{2N}^{R} \to \mathscr{A}$ the operator upon which our discrete equation is based, and by $\boldsymbol{\tau} = (\tau_k) \in l_1$, its attenuation factors [13]. Let

$$\begin{aligned} \mathbf{x} &:= \log \rho(\mathbf{t}+\mathbf{y}), & \mathbf{c} &:= \mathscr{F}_{2N}\mathbf{x}, \\ X_N &:= P\mathbf{x}, & \mathbf{C} &:= \mathscr{F}X_N, \end{aligned} \tag{6.1}$$

so that $C_k = \tau_k c_k = \tau_{-k}\bar{c}_{-k} = \bar{C}_{-k} (\forall k \in \mathbb{Z})$ and

$$X_N(t) = \sum_{k=-\infty}^{\infty} C_k e^{ikt}, \qquad X_N(t) + i(KX_N)(t) = 2 \sum_{k=0}^{\infty}{}' C_k e^{ikt},$$

where the prime indicates that the term with index 0 has weight $\frac{1}{2}$. We conclude that

$$p_N(w) := 2 \sum_{k=0}^{\infty}{}' c_k \tau_k w^k \qquad (w \in \bar{D}) \tag{6.2}$$

is analytic in D, continuous in $\bar{D}$, and satisfies

$$\operatorname{Re} p_N(e^{it}) = X_N(t), \quad \operatorname{Im} p_N(e^{it}) = (KX_N)(t), \quad \operatorname{Im} p_N(0) = 0.$$

Due to the discrete Theodorsen equation (1.1),

$$(KX_N)(\mathbf{t}) = (KP\mathbf{x})(\mathbf{t}) = \mathbf{K}_{\Sigma}\mathbf{x} = \mathbf{y}. \tag{6.3}$$

If we define

$$g_N(w) := w \exp(p_N(w)), \tag{6.4}$$

we obtain

$$g_N(\exp(i\mathbf{t})) = \exp[(P\mathbf{x})(\mathbf{t}) + i\mathbf{y} + i\mathbf{t}]. \tag{6.5}$$

Consequently, if P is interpolatory, i.e., $(P\mathbf{x})(\mathbf{t}) = \mathbf{x}$,

$$g_N(\exp(it_k)) = \rho(t_k + y_k) \exp(it_k + iy_k) \in \Gamma, \qquad k = 0, \cdots, 2N-1. \tag{6.6}$$

So, $g_N : \bar{D} \to \mathbb{C}$ is a function that is analytic in D and has $2N$ image points $g_N(\exp(it_k))$ lying on Γ (if P is interpolatory). We may therefore hope that g_N is a reasonable approximation of g. However, in general we cannot guarantee this. For example, g_N may not be conformal due to loops of $g_N(\partial D)$ (see Example 7.4). Error bounds for

$g_N - g$ have only been given in case P stands for trigonometric interpolation and $\varepsilon < 1$ (see Gaier [4, pp. 92 ff.]).

For evaluating g_N at a single arbitrary point $w \in \bar{D}$ we might approximate p_N by a partial sum, which can be evaluated, e.g., by Horner's algorithm. However, we would like to point out a special formula in case w is of the form

$$w = r e_{2\nu N}^{l} \qquad (0 < r \leqq 1,\ \nu \in \mathbb{Z},\ \nu > 0,\ l = 1, \cdots, 2\nu N) \tag{6.7}$$

where $e_{2\nu N} := \exp(i\pi/\nu N)$. Moreover, if r and ν are fixed, we can evaluate (or at least approximate) g_N simultaneously on the $2\nu N$ points of this type by using one FFT $\mathscr{F}_{2\nu N}^{-1}$. We think that in practice this property is an important advantage of Theodorsen's method. Also, an additional FFT yields the corresponding values of g_N', and we might proceed to compute higher derivatives.

Since $e_{2\nu N}^{2\nu N l} = 1$ $(\forall l \in \mathbb{Z})$, $\mathbf{c} \in \Pi_{2N} \subset \Pi_{2\nu N}$, and $\tau_{2jN} = 0$ for $j \neq 0$ [13, Eq. (2.5)], we obtain by inserting (6.7) into (6.2):

$$p_N(r e_{2\nu N}^{l}) = 2 {\sum_{k=0}^{2\nu N-1}}' c_k \chi_k^{(0)}(r) e_{2\nu N}^{kl}, \qquad l = 1, \cdots, 2\nu N, \tag{6.8}$$

where

$$\chi_k^{(0)}(r) := \sum_{j=0}^{\infty} \tau_{k+2\nu Nj} r^{k+2\nu Nj} \qquad (0 \leqq k < 2\nu N). \tag{6.9}$$

Similarly, if $(k\tau_k)_{k=-\infty}^{\infty} \in l_1$ or $r < 1$, differentiation of (6.2) yields

$$r e_{2\nu N}^{l} p_N'(r e_{2\nu N}^{l}) = 2 \sum_{k=1}^{2\nu N-1} c_k \chi_k^{(1)}(r) e_{2\nu N}^{kl}, \qquad l = 1, \cdots, 2\nu N, \tag{6.10}$$

where

$$\chi_k^{(1)}(r) := {\sum_{j=0}^{\infty}}' (k + 2\nu Nj) \tau_{k+2\nu Nj} r^{k+2\nu Nj} \qquad (1 \leqq k < 2\nu N). \tag{6.11}$$

The sums in (6.8) or (6.10) can be evaluated simultaneously at all $2\nu N$ points by applying $\mathscr{F}_{2\nu N}^{-1}$. Afterwards, (6.4) and

$$g_N'(w) = [1 + w p_N'(w)] \exp(p_N(w)) = [1 + w p_N'(w)] \frac{g_N(w)}{w} \tag{6.12}$$

yield the values of g_N and g_N', respectively, at these points. There remains the problem of summing up (6.9) and (6.11). We discuss two cases:

Example 6.1. Assume $\tau_k = 0$ for all $k > N$ as for *trigonometric interpolation* [13, Ex. 5.1], *Cesàro smoothing* or *Lanczos smoothing* [13, Ex. 6.1]. Then trivially

$$\chi_k^{(0)}(r) = \tau_k r^k, \qquad \chi_k^{(1)}(r) = k\tau_k r^k. \tag{6.13}$$

Example 6.2. If $r < 1$ and $2\nu N$ is sufficiently large, we may always approximate the series (6.9) and (6.11) by their first terms, i.e.,

$$\chi_k^{(0)}(r) \approx \tau_k r^k, \quad \chi_k^{(1)}(r) \approx k\tau_k r^k \quad \text{if } r < 1. \tag{6.14}$$

On the other hand, if $r = 1$ and P correspond to *interpolation by a spline function* S of degree $2m-1$ ($m \geqq 1$) [13, Ex. 5.2], we can sum up explicitly and obtain

$$\chi_k^{(0)}(1) = \begin{cases} \tau_0 & \text{if } k = 0, \\ \tau_k\left[1 + \psi_{2m-1}\left(\frac{k}{2\nu N}\right)\right] & \text{if } 0 < k < 2\nu N, \end{cases} \tag{6.15}$$

$$\chi_k^{(1)}(1) = k\tau_k\left[1 + \psi_{2m-2}\left(\frac{k}{2\nu N}\right)\right] \quad \text{if } 0 < k < 2\nu N \text{ and } m \geqq 2, \tag{6.16}$$

where the ψ_l are functions introduced in [13] and closely related to the polygamma functions $\psi^{(l)}$ [1, § 6.3]:

$$\psi_l(z) := \sum_{j=1}^{\infty} \left(\frac{z}{j+z}\right)^{l+1} = \frac{(-1)^{l+1}}{l!} z^{l+1} \psi^{(l)}(z) - 1$$

($z \in \mathbb{C}\backslash\{0, -1, -2, \cdots\}$, $l = 1, 2, \cdots$). Evaluating ψ_l is also discussed in [13, Ex. 5.2].

The formulas (6.14)–(6.16) also hold for smoothing by spline functions if we modify either c_k in (6.8) and (6.10) or τ_k in (6.14)–(6.16), cf. [13, § 6].

7. Numerical experiments on the accuracy of the approximate mapping function. As we have seen in the last section (cf. (6.6)), the $2N$ points $g_N(\exp(it_k))$ lie exactly on Γ if P is interpolatory and $\mathbf{y}$ is a solution of (1.1). So, for $t = t_k$

$$\delta_g(t) := \log|g_N(e^{it})| - \log\rho(\arg g_N(e^{it})) \tag{7.1}$$

should be 0, and thus, in addition to the residual defined by (2.5),

$$\delta_{g,0} := \max_k |\delta_g(t_k)| \tag{7.2}$$

is another measure for the accuracy of $\mathbf{y}$. For estimating the relative discretization error

$$\max_t |1 - g_N(e^{it})/g(e^{it})|, \tag{7.3}$$

we also compute

$$\delta_{g,1} := \max_k |\delta_g(t_k + t_1/2)|. \tag{7.4}$$

This estimate does not allow a direct comparison with the discretization error

$$\max_{\zeta \in \Gamma} |1 - |f_N(\zeta)|| \tag{7.5}$$

of approximations f_N to the inverse mapping $f = g^{[-1]}$, which is the aim of many other numerical mapping techniques [4], [5], [25]. But since

$$|\delta f| \approx |f'||\delta g| \approx \frac{|g|}{|g'|}|\delta \log g|,$$

we may consider

$$\delta_f(t) := \frac{|g_N(e^{it})|}{|g_N'(e^{it})|}\delta_g(t), \tag{7.6}$$

$$\delta_{f,0} := \max_k |\delta_f(t_k)|, \qquad \delta_{f,1} := \max_k |\delta_f(t_k + t_1/2)| \tag{7.7}$$

as estimates comparable to (7.5). They take into account that it is difficult to make $\delta_{g,1}$ small if $|wg'_N(w)|^{-1}$, the field strength of the potential field

$$\Psi_N(\zeta) := \log |g_N^{[-1]}(\zeta)| \quad \forall \zeta \in g_N(\bar{D})\backslash\{0\} \tag{7.8}$$

becomes small on $g_N(\partial D)$.

Yet another crude accuracy test consists in plotting $g_N(\partial D)$ to compare it with Γ. In addition, one can easily plot equipotential lines $\Psi_N(\zeta) = \log r = \text{const}$ since many points on them can be computed with one FFT (cf. (6.8)). The corresponding field lines are crudely approximated by broken lines in the following plots, however.

Comparing $\delta_{g,0}$ or the residual with $\delta_{g,1}$ allows one to judge whether the discretization error is small enough, i.e., whether N is large enough. The quantities

$$v_{\min} := \min_t \left| \frac{g_N(e^{it})}{g'_N(e^{it})} \right|, \qquad v_{\max} := \max_t \left| \frac{g_N(e^{it})}{g'_N(e^{it})} \right| \tag{7.9}$$

(where usually only $t \in \{t_k, t_k + t_0/2, k = 0, \cdots, 2N-1\}$ in our tests) indicate whether the mapping problem is difficult to solve. Finally, $\delta_{f,0}$ and $\delta_{f,1}$ show whether $\boldsymbol{\theta} = \mathbf{y} + \mathbf{t}$ can serve as an accurate basis for determining the boundary correspondence function $t(\theta)$ of the inverse map by inverse interpolation. However, we must be aware that $v_{\min}$ and $v_{\max}$ may deviate essentially from the corresponding quantities for the exact mapping function g since g'_N may be an unsatisfactory approximation of g'. For the same reason, δf_0 and δf_1 may be poor estimates for (7.5).

Example 7.1. Results on the accuracy of the mapping function g_N corresponding to the SOR iterate $\mathbf{y}_{m(l)+1}$ for the inverse ellipse are given in Table 7.1. (For $\alpha = .1$ and $\alpha = .05$ the continuation method has been used.) If α is large (i.e., ε is small), $\delta_{g,0}$ and $\delta_{g,1}$ are of the same order, and they mainly depend on l, i.e., on the stopping criterion for the iteration (cf. (2.4)). If α is small, a reduction of the error $\delta_{g,1}$ requires a larger N. The quantities $v_{\min}$ and $v_{\max}$ have also been computed. At least five digits coincide with the correct values $v_{\min} = \alpha$, $v_{\max} = 1/\alpha$, except if $\alpha = .05$, where we obtain $v_{\min} = .050010$, $v_{\max} = 22.036$ (for $N = 128$), which indicates an error of at least 10% in g'. Plots of equipotential lines $\Psi_N(\zeta) = \log(k/8)$, $k = 1, \cdots, 8$, are shown in Fig. 7.1 for $\alpha = .2, .1$, and, partly, but five times magnified, for $\alpha = .05$.

TABLE 7.1
Accuracy of g_N^T for reflected ellipses (Ex. 7.1).

	$N = 32, l = 6$		$N = 128, l = 6$		$N = 128, l = 12$	
α	$\delta_{g,0}$	$\delta_{g,1}$	$\delta_{g,0}$	$\delta_{g,1}$	$\delta_{g,0}$	$\delta_{g,1}$
.8	$5.51_{10^{-11}}$	$4.42_{10^{-11}}$	$6.31_{10^{-11}}$	$5.25_{10^{-11}}$	$1.87_{10^{-14}}$	$1.60_{10^{-14}}$
.6	$1.90_{10^{-9}}$	$1.79_{10^{-9}}$	$2.43_{10^{-9}}$	$2.09_{10^{-9}}$	$3.91_{10^{-14}}$	$3.55_{10^{-14}}$
.4	$4.55_{10^{-8}}$	$1.65_{10^{-7}}$	$7.28_{10^{-8}}$	$7.07_{10^{-8}}$	$1.24_{10^{-13}}$	$1.39_{10^{-13}}$
.3	$2.87_{10^{-7}}$	$8.03_{10^{-6}}$	$3.12_{10^{-7}}$	$3.36_{10^{-7}}$	$1.99_{10^{-13}}$	$2.10_{10^{-13}}$
.2	$8.50_{10^{-7}}$	$5.43_{10^{-4}}$	$7.94_{10^{-7}}$	$9.18_{10^{-7}}$	$5.76_{10^{-13}}$	$1.14_{10^{-12}}$
.1	$1.52_{10^{-6}}$	$5.41_{10^{-2}}$	$1.36_{10^{-6}}$	$6.59_{10^{-7}}$	$3.54_{10^{-12}}$	$9.93_{10^{-7}}$
.05	$[3.59_{10^{-7}}$	$3.55_{10^{0}}]$	$2.78_{10^{-6}}$	$2.41_{10^{-3}}$	$1.65_{10^{-10}}$	$2.40_{10^{-3}}$

Examples 7.2 *and* 7.3 correspond to Examples 2.2 and 2.3, respectively. As shown in Table 7.2, $\delta_{g,1}$ is roughly of the same magnitude in the cases of trigonometric (T) and cubic spline interpolation (S3). The advantage of S3 becomes evident by

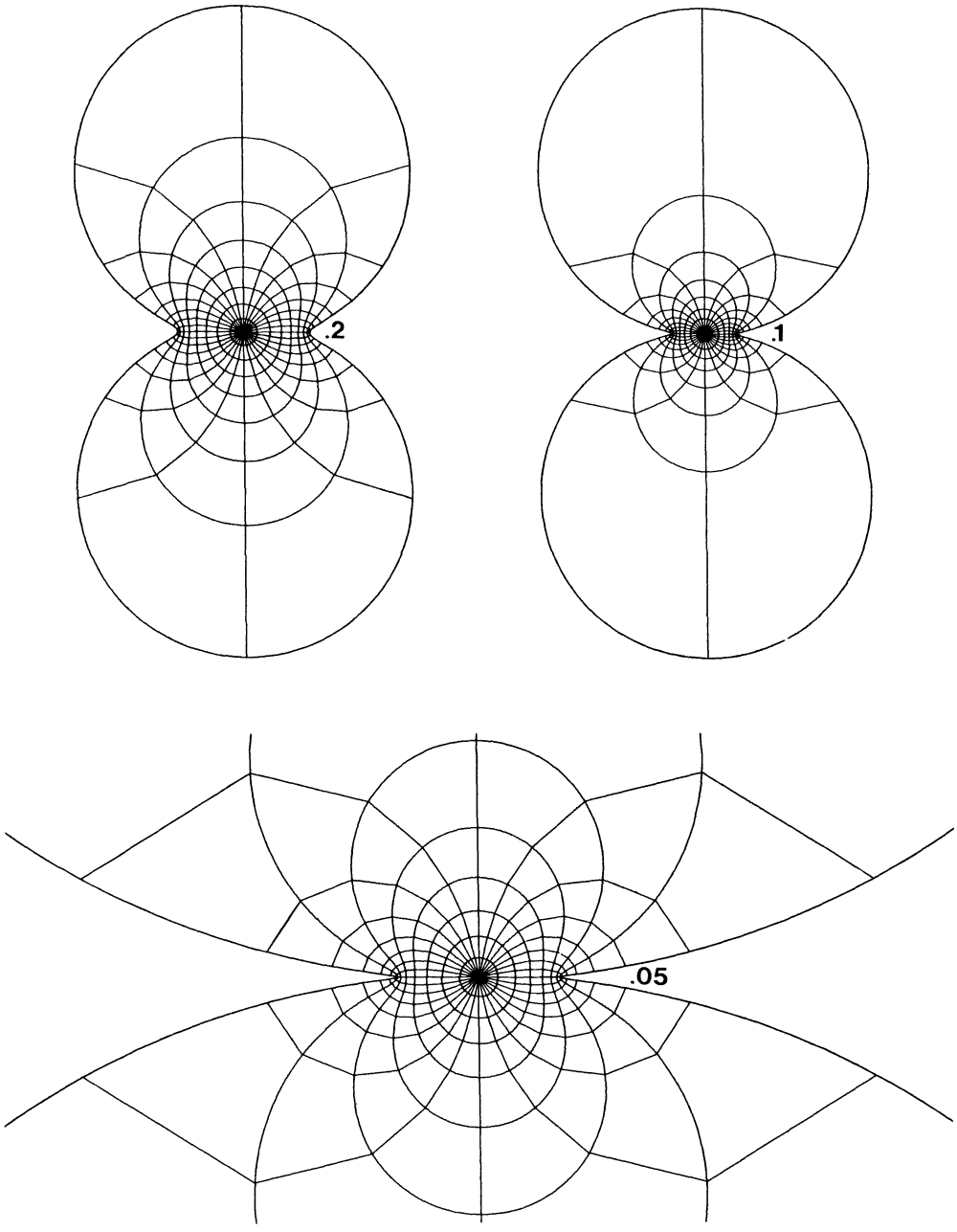


FIG. 7.1. *Conformal maps on reflected ellipses* ($\alpha = .2, .1,$ *and* $.05$).

inspection of g'_N. While $v^{\mathrm{T}}_{\min} = v^{\mathrm{S3}}_{\min}$, $v^{\mathrm{T}}_{\max} = v^{\mathrm{S3}}_{\max}$ for $\alpha = .4, .8$, we obtain for $\alpha = .2$, $N = 512$ in

$$\text{Ex. 2.1:} \qquad v^{\mathrm{T}}_{\min} = .037, \qquad v^{\mathrm{T}}_{\max} = 29.86,$$

$$\text{Ex. 2.2:} \qquad v^{\mathrm{T}}_{\min} = .093, \qquad v^{\mathrm{T}}_{\max} = 4.29.$$

TABLE 7.2
Accuracy of g_N^T and g_N^{S3} in case of a symmetric and in case of an asymmetric piecewise analytic boundary (Exs. 7.2 and 7.3).

		$N = 128$		$N = 512$					
Ex.	α	$\delta_{g,1}^T$	$\delta_{g,1}^{S3}$	$\delta_{g,1}^T$	$\delta_{g,1}^{S3}$	$\delta_{f,1}^T$	$\delta_{f,1}^{S3}$	$v_{\min}^{S3}$	$v_{\max}^{S3}$
7.2	.8	$2.52_{10^{-6}}$	$1.89_{10^{-6}}$	$3.09_{10^{-7}}$	$3.48_{10^{-7}}$	$2.40_{10^{-7}}$	$2.71_{10^{-7}}$	.774	1.25
	.4	$1.78_{10^{-3}}$	$1.47_{10^{-3}}$	$7.97_{10^{-5}}$	$1.02_{10^{-4}}$	$1.85_{10^{-5}}$	$2.34_{10^{-5}}$	.204	2.38
	.2	$\{1.02_{10^{-1}}\}$	$1.05_{10^{-1}}$	$9.78_{10^{-3}}$	$7.69_{10^{-3}}$	$2.27_{10^{-2}}$	$3.92_{10^{-4}}$	.049	5.02
7.3	.8	$6.37_{10^{-6}}$	$7.27_{10^{-6}}$	$4.30_{10^{-7}}$	$3.86_{10^{-7}}$	$3.85_{10^{-7}}$	$3.45_{10^{-7}}$	.847	1.27
	.4	$5.17_{10^{-4}}$	$6.43_{10^{-4}}$	$3.06_{10^{-5}}$	$3.15_{10^{-5}}$	$1.21_{10^{-5}}$	$1.24_{10^{-5}}$	.343	2.37
	.2	$2.18_{10^{-2}}$	$1.75_{10^{-2}}$	$3.35_{10^{-3}}$	$2.76_{10^{-3}}$	$4.31_{10^{-4}}$	$3.14_{10^{-4}}$	.096	4.04

The plots for these two examples are hardly distinguishable from the corresponding ones with S3 approximation shown in Fig. 7.2. However, the difference between T and S3 becomes visible if we reduce N to 128 as in Fig. 7.3. Here, g_N' is very inaccurate for both T and S3, of course.

Example 7.4. Results for the square and ten types of approximation (cf. Ex. 2.4) are compiled in Table 7.3. They have been computed by Euler iteration with $\mathbf{y}_0 = \mathbf{0}$, $l = 6$, $N = 1{,}024$ (actually, 256 variables are involved). The smallest error $\delta_{g,\max} := \max\{\delta_{g,0}, \delta_{g,1}\}$ is obtained for $S1/10^5$, the first degree smoothing spline with damping parameter $\tilde{\rho} = 10^5$. Unfortunately, in general g_N' has singularities on ∂D at the breakpoints e^{it_k} of the first degree spline. The best result among the

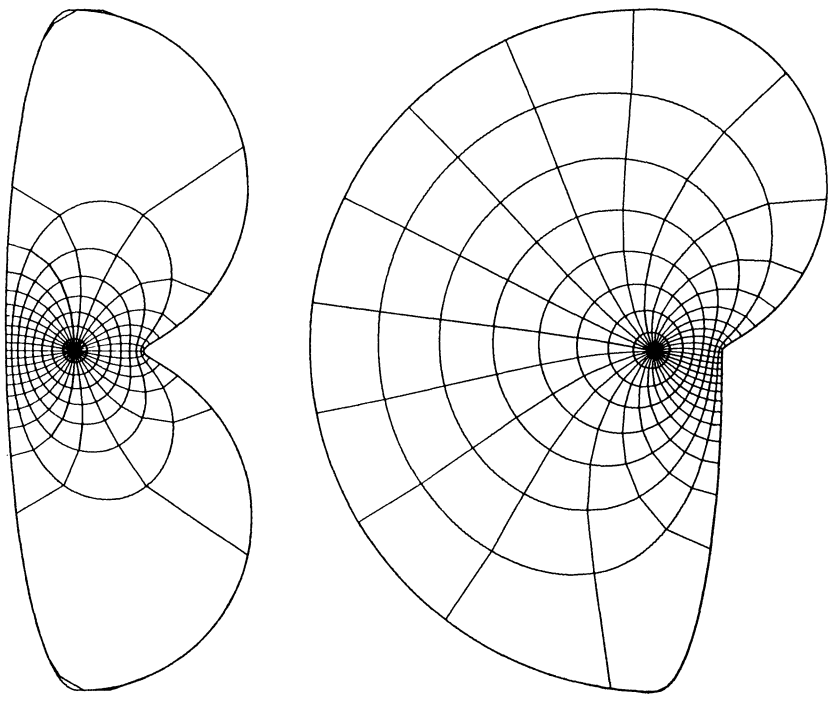


FIG. 7.2. *Examples* 7.2 *and* 7.3: *cubic spline interpolation with* $N = 512$.

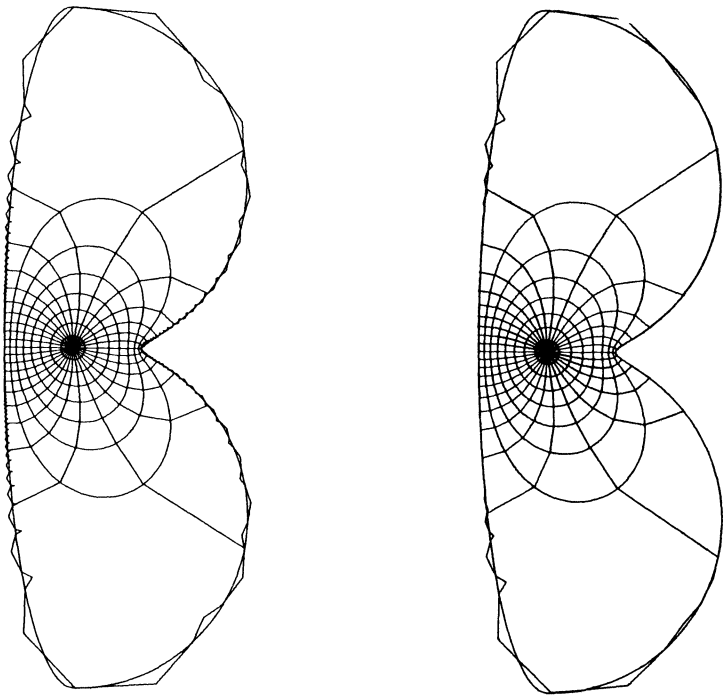


FIG. 7.3. *Example* 7.2: *trigonometric* (*left*) *and cubic spline* (*right*) *interpolation.*

TABLE 7.3
Accuracy of mapping functions based on various types of approximation (*Ex.* 7.4: *square*, $N = 1{,}024$).

	T	C	L	S3/10^8	S3/10^{10}	S3/10^{12}
$\delta_{g,0}$	$5.62_{10^{-8}}$	$1.92_{10^{-2}}$	$1.29_{10^{-2}}$	$1.64_{10^{-2}}$	$1.98_{10^{-3}}$	$6.81_{10^{-5}}$
$\delta_{g,1}$	$1.65_{10^{-2}}$	$5.81_{10^{-3}}$	$2.14_{10^{-3}}$	$5.65_{10^{-3}}$	$1.37_{10^{-2}}$	$1.56_{10^{-2}}$
$\delta_{f,0}$	$6.36_{10^{-8}}$	$2.03_{10^{-3}}$	$1.23_{10^{-3}}$	$1.63_{10^{-3}}$	$1.82_{10^{-4}}$	$6.17_{10^{-5}}$
$\delta_{f,1}$	$2.69_{10^{-3}}$	$6.32_{10^{-4}}$	$2.68_{10^{-4}}$	$7.03_{10^{-4}}$	$1.30_{10^{-3}}$	$1.48_{10^{-3}}$
$v_{\min}$	.0928	.1058	.0956	.0996	.0921	.0922
$v_{\max}$	1.3587	1.3374	1.3482	1.3281	1.3603	1.3604

	S1	S1/10^4	S1/10^5	S1/10^6
$\delta_{g,0}$	$3.00_{10^{-5}}$	$1.59_{10^{-2}}$	$3.70_{10^{-3}}$	$4.65_{10^{-4}}$
$\delta_{g,1}$	$9.33_{10^{-3}}$	$5.51_{10^{-3}}$	$6.18_{10^{-3}}$	$8.97_{10^{-3}}$

differentiable approximations considered is found for Lanczos smoothing. This has also been observed for some other regions with salient corners. Plots of the first quadrant are shown in Fig. 7.4. The one with trigonometric interpolation is magnified and based on 2,048 points of $g_N(\partial D)$ in this quadrant; $2N/4 = 512$ of these points should lie on Γ. This plot reveals that $g_N(\partial D)$ has loops. The other plots, for which only half as many points have been computed, exhibit that smoothing involves rounding off salient corners. The plot of S3/10^8, which is not shown, looks very similar to the one for L, and is therefore superior to the case S3/10^{10}, showing already a few ripples typical for S3.

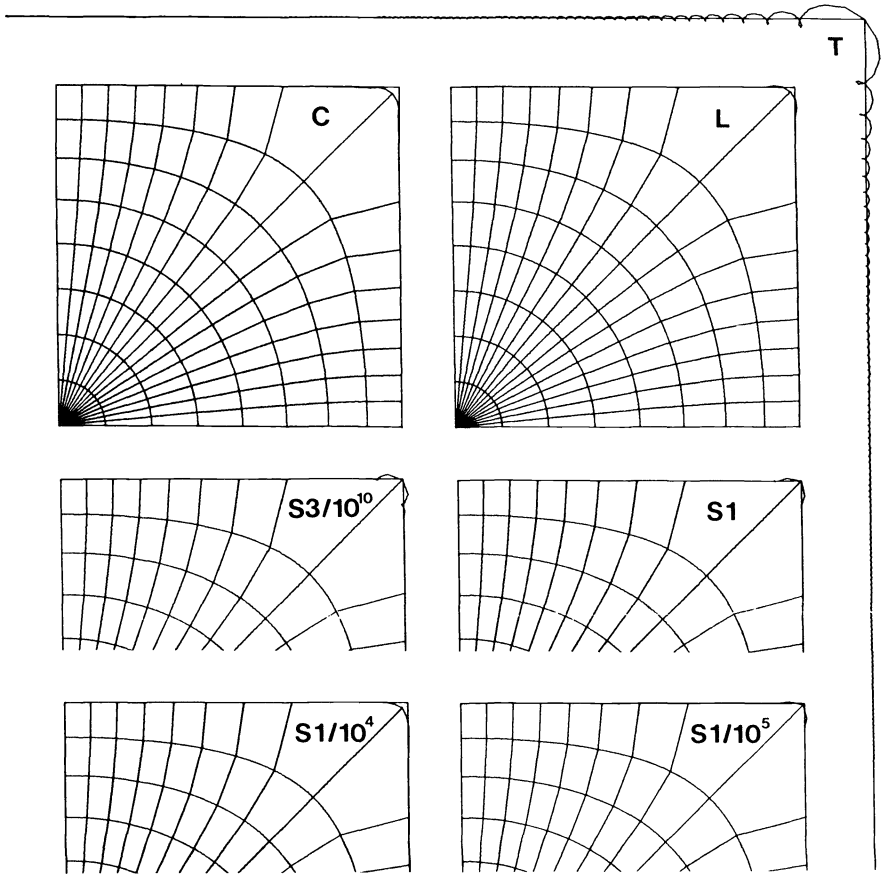


FIG. 7.4. *Conformal maps—based on various types of approximation—on a square.*

The behavior of $X(t) := \log \rho(\theta(t))$ in case of a piecewise analytic boundary (with $\varepsilon_\rho < \infty$, cf. § 4) suggests the conjecture that the loops in case of trigonometric interpolation can be avoided by choosing N large enough if $\alpha_j > \frac{1}{2}$ at all corners. (In fact, there are no loops if $\theta'_N(t) = Y'_N(t) + 1 > 0$ $(\forall t)$. Since $\theta' \in L_2[0, 2\pi]$ is piecewise continuous and positive except at corners with $\alpha_j > 1$, where $\theta'(t) = O(|t - t_j|^{\alpha_j - 1})$, one may expect that θ'_N is nonnegative and vanishes at most at these t_j if N is sufficiently large. As yet, a proof has not been given.)

In practice loops are a problem not only near corners with $\alpha_j \leqq \frac{1}{2}$. While theoretically no loops occur if Γ is analytic and N is sufficiently large, there are simple examples where it is in practice nearly impossible to avoid these loops:

Example 7.5. It is easy to map the disk fairly accurately onto an ellipse with semiaxes 1 and $\alpha = .4$. Yet, for $\alpha = .2$ we obtain the disappointing results listed in Table 7.4 and displayed in Fig. 7.5. Note that $\delta_{f,\max} := \max\{\delta_{f,0}, \delta_{f,1}\}$ is much smaller than $\delta_{g,\max} := \max\{\delta_{g,0}, \delta_{g,1}\}$ here. There is a marked similarity with the results for the square (and boundaries with acute corners). The best results are obtained by Lanczos smoothing and appropriately smoothed splines. Here, $S3/10^{10}$ is nearly as accurate as $S1/10^5$ (which is not differentiable). To obtain better results, one might try two preliminary Kármán–Trefftz transformations (cf. § 5 and [8]).

TABLE 7.4
Accuracy of mapping functions based on various types of approximation (Ex. 7.5: ellipse with $\alpha = .2$).

N		T	C	L	S3	S3/10^6	S3/10^8	S3/10^{10}
512	$\delta_{g,max}$	{.4348}	.2182	.1836	{.4165}	.3400	.2187	{.3487}
1024	$\delta_{g,max}$	{.3235}	.1673	.1385	{.3086}	.3400	.2240	.1170
	$\delta_{f,max}$	.0571	.0677	.0303	.0187	.0189	.0054	.0019
	v_{min}	.0167	.0179	.0167	.0165	.0555	.0242	.0163
	v_{max}	1.8248	1.7476	1.8066	1.8350	1.5497	1.5552	1.8402

N		S1	S1/10^4	S1/10^5	S1/10^6
512	$\delta_{g,max}$	{.2705}	.1912	{.1824}	{.2600}
1024	$\delta_{g,max}$	{.1876}	.2159	.0975	{.1563}

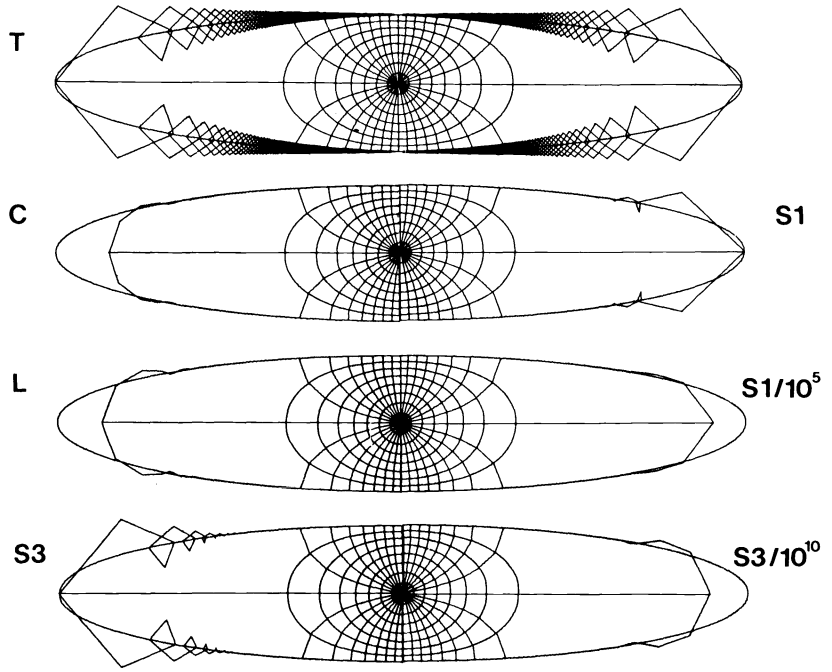


FIG. 7.5. *Seven types of approximate conformal maps on an ellipse.*

Example 7.6. Results for the Swiss cross, which is an example with a reentrant corner, are compiled in Table 7.5 ($N = 2{,}048$). The two examples in the top part of Fig. 7.6 demonstrate that $N = 512$ (i.e., 128 variables) is enough for a rough but reasonable approximation. All six other solutions in Fig. 7.6 are done with $N = 2{,}048$.

Example 7.7. In Fig. 7.7 we display the Lanczos type solution with $N = 512$ for an asymmetric L-shape. It confirms that the method also works for asymmetric regions.

So far, we have done no tests on the accuracy of the conformal mapping defined by composition of g_N, optional inverse osculation maps, and the maps inverse to those

TABLE 7.5
Accuracy of mapping functions based on various types of approximation (Ex. 7.6: Swiss cross, $N = 2{,}048$).

	T	C	L	S3/10^8	S3/10^{10}	S3/10^{12}
$\delta_{g,\max}$	$7.66_{10^{-3}}$	$2.30_{10^{-2}}$	$2.02_{10^{-2}}$	$2.80_{10^{-2}}$	$2.10_{10^{-2}}$	$6.79_{10^{-3}}$
$\delta_{f,\max}$	$2.86_{10^{-3}}$	$1.27_{10^{-2}}$	$9.28_{10^{-4}}$	$1.69_{10^{-3}}$	$9.40_{10^{-4}}$	$1.48_{10^{-3}}$
$v_{\min}$	.0442	.0464	.0459	.0602	.0448	.0443
$v_{\max}$	54.43	27.08	39.53	26.10	47.47	64.58

	S1	S1/10^5	S1/10^6	S1/10^7	S1/10^8
$\delta_{g,\max}$	$1.59_{10^{-2}}$	$2.05_{10^{-2}}$	$1.88_{10^{-2}}$	$1.62_{10^{-2}}$	$1.60_{10^{-2}}$

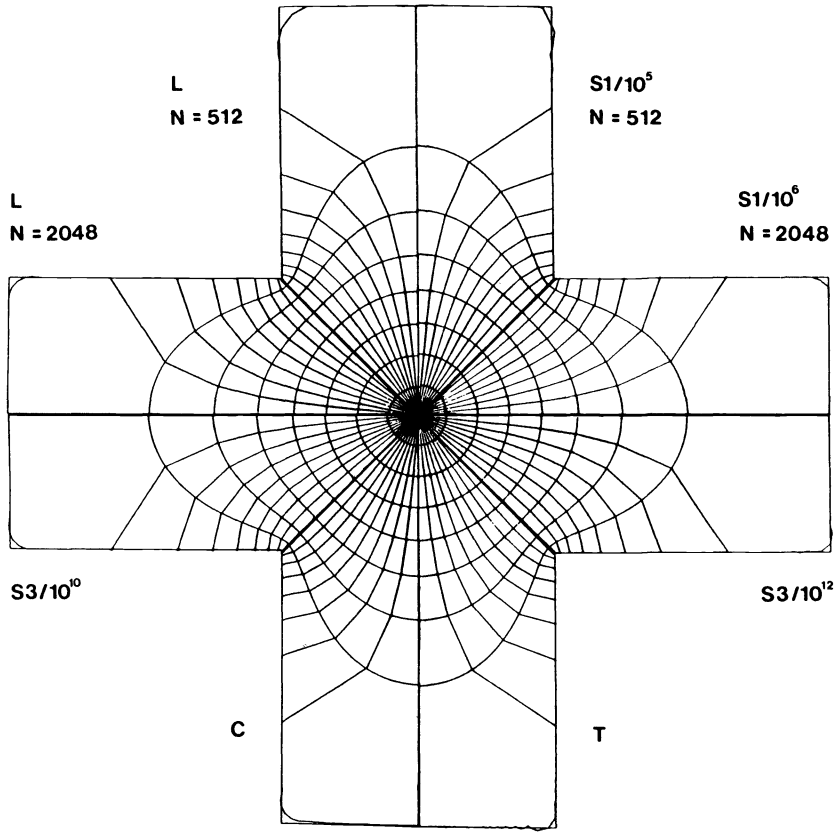


FIG. 7.6. *Eight types of approximate conformal maps on a Swiss cross. (Inaccuracies on the symmetry axes are due to inaccurate plotting and gluing.)*

used for corner elimination (cf. § 5). However, the following should be kept in mind: If $\delta_g(t_0)$ is the error of g_N at a point t_0, where a corner with angle $\alpha\pi$ has been eliminated, then the inverse transformation, which contains a map of type $\gamma_{\zeta_0,1/\alpha}$, yields an error proportional to $[\delta_g(t_0)]^\alpha$. So, if α is small, much of the accuracy of Theodorsen's method (or any other one) gets lost.

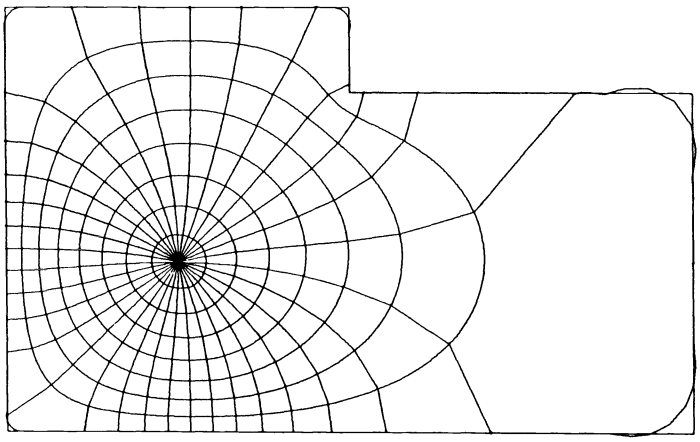


FIG 7.7. *Conformal map (with Lanczos smoothing) on an L-shape.*

8. Conclusions. The nonlinear SOR iteration and the nonlinear second order Euler iteration proposed in [12] are efficient means for solving the classical or the modified discrete Theodorsen equation, respectively, for most regions that are starlike with respect to the origin. If $\varepsilon_\rho > 1$, one may have to apply the continuation method of § 3 as an outer iteration. But then, in all of our tests convergence was achieved if N and the parameters I_0 and l_0 of the continuation method were large enough.

However, the classical discretization based on trigonometric interpolation may yield approximate mapping functions g_N which are useless since they are not one-to-one. Not only does this happen for boundaries with acute corners, but in practice even for certain simple analytic boundaries where the mapping problem is ill-conditioned, such as, e.g., flat ellipses. Discretizations based on other types of approximation involving some smoothing yield much better results in these cases. Smoothing has the (in practice often negligible) effect of rounding off salient corners and other salient parts of the region, where the Green's function nearly vanishes. In particular, Cesaro and Lanczos smoothing proved very useful. (For similar smoothing techniques not tested here, see [32], [37], [38].) Smoothing splines, which are more difficult to implement but nearly as efficient in execution, may yield even better results if the smoothing parameter is chosen appropriately (which was done by trial and error in our tests). First degree splines are not suitable if g'_N is requested too.

If we cannot allow salient corners to be rounded off, we can usually eliminate these corners by preliminary conformal maps, which have also been implemented in a general program. Additional preliminary osculation maps could be used to deal with regions that are not starlike or that present other difficulties. More tests in this direction are certainly worthwhile.

Acknowledgment. I am indebted to Prof. P. Henrici for his continuous interest in and support for this project. I have also benefited from valuable discussions with the late Prof. E. Grassmann, whose work in numerical conformal mapping was abruptly terminated by his tragic premature death.

Furthermore, I am very grateful to Miss B. Knecht, who typed this and several related manuscripts with utmost care, and to my wife Ursula for the patience and understanding which allowed me to complete this work. Finally, L. N. Trefethen and an unknown referee were kind enough to suggest a number of linguistic improvements.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, DC, 1964.

[2] R. ALBRECHT, *Zum Schmiegungsverfahren der konformen Abbildung*, Z. Angew. Math. Mech., 32 (1952), pp. 316–318.

[3] J. W. COOLEY, P. A. W. LEWIS AND P. D. WELCH, *The finite Fourier transform*, IEEE Trans. Audio Electroacoust. AU-17 (1969), pp. 77–85.

[4] D. GAIER, *Konstruktive Methoden der konformen Abbildung*, Springer, Berlin/Göttingen/Heidelberg, 1964.

[5] ———, *Integralgleichungen erster Art und konforme Abbildung*, Math. Z., 147 (1976), pp. 113–129.

[6] I. E. GARRICK, *Conformal mapping in aerodynamics, with emphasis on the method of successive conjugates*, National Bureau of Standards Appl. Math. Ser., 18 (1952), pp. 137–147.

[7] W. GAUTSCHI, *Attenuation factors in practical Fourier analysis*, Numer. Math., 18 (1972), pp. 373–400.

[8] E. GRASSMANN, *Numerical experiments with a method of successive approximation for conformal mapping*, Z. Angew. Math. Phys., 30 (1979), pp. 873–884.

[9] E. GRASSMANN AND J. G. ROKNE, *The range of values of a circular complex polynomial over a circular complex interval*, Computing, 23 (1979), pp. 139–169.

[10] M. H. GUTKNECHT, *Existence of a solution to the discrete Theodorsen equation for conformal mappings*, Math. Comp., 31 (1977), pp. 478–480.

[11] ———, *Fast algorithms for the conjugate periodic function*, Computing, 22 (1979), pp. 79–91.

[12] ———, *Solving Theodorsen's integral equation for conformal maps with the fast Fourier transform and various nonlinear iterative methods*, Numer. Math., 36 (1981), pp. 405–429.

[13] ———, *Solving Theodorsen's integral equation for conformal maps with the fast Fourier transform, Part III: The evaluation of the conjugate function of a periodic spline on a uniform mesh*, Report 79-05, Seminar f. Angew. Math., Eidgen. Tech. Hochschule, Zürich, Oct. 1979.

[14] J. HEINHOLD AND R. ALBRECHT, *Zur Praxis der konformen Abbildung*, Rend. Circ. Mat. Palermo Ser. 2, 3 (1954), pp. 130–148.

[15] P. HENRICI, *Fast Fourier methods in computational complex analysis*, SIAM Rev., 21 (1979), pp. 481–527.

[16] O. HÜBNER, *Zur Numerik der Theodorsenschen Integralgleichung in der konformen Abbildung*, Mitt. Math. Sem. Giessen, 140 (1979), pp. 1–32.

[17] H. KOBER, *Dictionary of Conformal Representation*, Dover, New York, 1952.

[18] L. LANDWEBER AND T. MILOH, *Elimination of corners in the mapping of a closed curve*, J. Engrg. Math., 6 (1972), pp. 369–375.

[19] R. S. LEHMAN, *Development of the mapping function at an analytic corner*, Pacific J. Math., 7 (1957), pp. 1437–1449.

[20] H. LEWY, *Developments at the confluence of analytic boundary conditions*, Univ. of California Publ. in Math., 1 (1950), pp. 247–280.

[21] C. LUNDWALL-SKAAR, *Konforme Abbildung mit Fast Fourier Transformationen*, Diplomarbeit, Eidgen. Tech. Hochschule, Zürich, 1975.

[22] W. NIETHAMMER, *Iterationsverfahren bei der konformen Abbildung*, Computing, 1 (1966), pp. 146–153.

[23] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[24] R. C. SINGLETON, *An algorithm for computing the mixed radix fast Fourier transform*, IEEE Trans. Audio Electroacoust., AU-17 (1969), pp. 93–103.

[25] G. T. SYMM, *An integral equation method for conformal mapping*, Numer. Math., 9 (1966), pp. 250–258.

[26] W. VON KOPPENFELS AND F. STALLMANN, *Praxis der konformen Abbildung*, Springer, Berlin/Göttingen/Heidelberg, 1959.

[27] S. E. WARSCHAWSKI, *On a theorem of L. Lichtenstein*, Pacific J. Math., 5 (1955), pp. 835–840.

[28] R. WEGMANN, *Ein Iterationsverfahren zur konformen Abbildung*, Numer. Math., 30 (1978), pp. 453–466.

[29] J. R. WHITEMAN AND S. SCHEFFLER, *Conformal transformation techniques for the numerical solution of Poisson problems*, Report BICOM 78-3, Institute of Computational Mathematics, Brunel University, Uxbridge, England, June 1978.

[30] F. BAUER, P. GARABEDIAN, D. KORN AND A. JAMESON, *Supercritical Wing Sections* II, Springer, Berlin/Heidelberg/New York, 1975.

[31] S. CHAKRAVARTHY AND D. ANDERSON, *Numerical conformal mapping*, Math. Comp., 33 (1979), pp. 953–969.

[32] A. K. CHAN AND C. K. CHUI, *Nonrecursive digital filter designs by means of Korovkin kernels*, IEEE Trans. Acoust. Speech Signal Process., ASSP-27 (1979), pp. 218–222.
[33] B. FORNBERG, *A numerical method for conformal mappings*, this Journal, 1 (1980), pp. 386–400.
[34] M. H. GUTKNECHT, *Two applications of periodic splines*, in Approximation Theory III, E. W. Cheney, ed., Academic Press, New York, 1980.
[35] R. M. JAMES, *A new look at two-dimensional incompressible airfoil theory*, Report MDC-J0918/01, Douglas Aircraft Company, 1971.
[36] G. MORETTI, *Conformal mappings for computations of steady, three-dimensional, supersonic flows*, in Numerical/Laboratory Computer Methods in Fluid Mechanics, American Society of Mechanical Engineers, New York, 1976.
[37] A. H. NUTTAL, *Some windows with very good sidelobe behavior*, IEEE Trans. Acoust. Speech Signal Process., ASSP-29 (1981), pp. 84–91.
[38] C. ROSSI, *Window functions for nonrecursive digital filters*, Electron. Lett., 3 (1967), pp. 559–561.
[39] L. C. WOODS, *The Theory of Supersonic Plane Flow*, Cambridge Univ. Press, Cambridge, 1961.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

EXTRAPOLATED ADAPTIVE QUADRATURE*

D. KAHANER† AND J. STOER‡

Abstract. In this paper we consider algorithms for numerical quadrature in one dimension which combine global adaption and extrapolation. We analyze the convergence of one specific algorithm in terms of the amount of work as a function of the input accuracy request. The main result is that asymptotically the expected amount of work is unaffected by the adaption. This is illustrated by numerical examples. An alternative algorithm is also suggested.

Key words. automatic quadrature, mathematical software, adaptive quadrature, extrapolation, Romberg, Wynn's ε-algorithm

1. Introduction. Extrapolation in Romberg's method has long been an elegant algorithm for the numerical evaluation of definite integrals [1, p. 45]. Early investigations centered on the rate of convergence for smooth integrands [2], numerical stability of the mesh sequences [2], [3], [4], stopping criteria [5], and new asymptotic expansions for nonsmooth integrands [6]. In more than one dimension many of these same ideas carry over, but crucial adjustments must be made for the specific region of interest, singularity structure of the integrand, zero coefficients in the asymptotic expansion, etc. [7]. A number of subroutines have been written for automatic quadrature based upon (usually) the trapezoidal rule in one dimension [1, p. 376]. For general smooth functions these algorithms have extremely rapid convergence. However, when implemented as programs several persistent problems remained.

a) The usual process of halving the mesh lengths at each level led to very rapid increase in the number of evaluation points used. Fewer function values can be required if the mesh lengths do not decrease so rapidly [9], but in practice the improvement is very modest. This "surcharge" feature is common to all automatic quadrature routines [8] but is extremely costly for extrapolation codes.

b) The familiar Euler–MacLaurin formula, which is the asymptotic error expansion for the trapezoidal rule, only applies to smooth integrands. The practical use of a code based on such an expansion for an integrand with a singularity results in essentially no improvement from the extrapolation. If the nature and position of the singularity is known (this is the case in most problems), the correct expansion can be used, and some programs do alter the expansion to account for this situation. This can be done either by requiring the user to describe the singularity explicitly or by deducing it from numerical evidence [10]. An alternative technique uses a nonlinear extrapolation procedure, the ε-algorithm, to handle more general expansions automatically at the cost of an even higher surcharge [11].

Concurrent with the development of extrapolation procedures was the growth in popularity of adaptive quadrature algorithms [12]. These share many of the same surcharge and other problems inherent to any automatic quadrature routine [8]. Nevertheless, the best of them can successfully integrate smooth and nonsmooth integrands. Furthermore, the nonuniform distribution of function values which the typical algorithms generate is satisfying to many users. In the popularity contest, at least, adaptive algorithms have clearly "won".

* Received by the editors March 9, 1982.

† Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234.

‡ Institut für Angewandte Mathematik und Statistik der Universität Wurzburg Am Hubland, D-8700 Wurzburg, West Germany.

In the quest for even better methods several attempts have been made to combine extrapolation and adaptation [10], [13], [14]. The most recent of these [14] attempts to get the outstanding convergence rates associated with extrapolation along with the nonuniform mesh associated with adaptation. The resulting codes appear to be among the best available today.

Our goal in this paper is to provide an analysis of one specific model adaptive extrapolation algorithm, which we name GAX. This algorithm shares many features with some existing programs, albeit in rather simplified form. In particular we leave out all questions relating to finite word computer arithmetic, even though these issues are of vital importance in any real implementation. Additionally, we make numerous simplifying assumptions about the local estimates that are used even though the success of most adaptive quadratures depends heavily upon these estimates. In § 2 we define the terms we need to associate with extrapolation. Section 3 describes what we mean by adaptive quadrature and gives the algorithm GAX. A convergence theorem, Theorem 4.2, is given in § 4. Some numerical examples and implementation details are in § 5. Finally we present a modified algorithm, GAXI, which we believe to be an improvement.

2. Extrapolation. We consider the problem of the approximation of

$$\mathcal{T} := \mathcal{T}_f := \int_a^b f(x)\,dx, \qquad -\infty < a < b < \infty \tag{2.1}$$

by use of the Romberg T-table [2]. Let I denote a generic interval $[\alpha, \beta]$, and

$$\mathrm{Tr}\,(I) := \frac{\beta-\alpha}{2}[f(\alpha)+f(\beta)], \tag{2.2}$$

i.e., the primitive trapezoidal rule for I. When $[\alpha, \beta]$ is uniformly subdivided into subintervals of length h, the compound trapezoidal rule for I is

$$T(h; I) := h\left[\frac{f(\alpha)}{2}+f(\alpha+h)+\cdots+f(\beta-h)+\frac{f(\beta)}{2}\right]. \tag{2.3}$$

Instead of $T(h; [a, b])$ we write briefly $T(h)$.

Thus

$$\mathrm{Tr}\,([a, b]) = T(b-a) := T_{0,0}. \tag{2.4}$$

Romberg's method to calculate T starts with the computation of (2.3) for subdivisions of the full interval into 1, 2, 4, 8, $\cdots$ equal parts. We use the notation

$$T_{k,0} := T(h_k), \qquad h_k := \frac{b-a}{2^k}. \tag{2.5}$$

These values are arranged vertically and completed to a triangular array

$$\begin{array}{ccccc} T_{0,0} & & & & \\ & T_{1,1} & & & \\ T_{1,0} & & T_{2,2} & & \\ & T_{2,1} & \vdots & \cdots & T_{m,m} \\ T_{2,0} & \vdots & & & \\ \vdots & & T_{m,2} & & \\ & T_{m,1} & & & \\ T_{m,0} & & & & \end{array} \tag{2.6}$$

which is usually called the T-table, with

$$T_{m,k} := \frac{4^k T_{m,k-1} - T_{m-1,k-1}}{4^k - 1}, \qquad k > 0. \tag{2.7}$$

The elements $T_{m,k}$ with $m \geqq k$, k, fixed, form the kth column of (2.6). Then we have [2]

$$T_{m,k} = \sum_{j=0}^{k} c_{k,j} T_{m-k+j,0} \tag{2.8}$$

with

$$c_{k,j} = \prod_{\substack{i=0 \\ i \neq j}}^{k} \frac{h_i^2}{h_i^2 - h_j^2} = \prod_{\substack{i=0 \\ i \neq j}}^{k} \frac{1}{1 - 4^{i-j}},$$

$$\sum_{j=0}^{k} c_{k,j} = 1, \tag{2.9}$$

$$\sum_{j=0}^{k} |c_{k,j}| = \prod_{l=1}^{k} \frac{1+4^{-l}}{1-4^{-l}} \leqq 1.969\cdots < 2 \quad \text{for all } k \geqq 0.$$

It has been shown [2] that for $f \in C^{2k+2}[a, b]$

$$T_{m,k} - \int_a^b f(x)\,dx = O(4^{-m(k+1)}), \qquad m \to \infty, \tag{2.10}$$

and also that if f is complex analytic in an open domain containing $[a, b]$ then any diagonal of (2.6) converges superlinearly, i.e., asymptotically faster than any geometric series.

In a practical program for Romberg quadrature the T-table (2.6) is generated in a stepwise manner—a new $T_{m,0}$ element is computed by direct function evaluation and then $T_{m,1}, T_{m,2}, \cdots, T_{m,m}$ are computed in turn by (2.7). The process produces as output in addition a number

$$|T_{m,m} - T_{m,m-1}| + |T_{m,m} - T_{m-1,m-1}| \tag{2.11}$$

which represents a conservative estimate of the error in $T_{m,m}$. The amount of work necessary to compute $T_{m,k}$ measured in evaluations of $f(x)$ is $2^m + 1$. Thus (2.10) may be restated to show that n units of work achieve an error ε in the kth column of the T-table which is

$$\varepsilon = O(n^{-2k-2}). \tag{2.12}$$

Most programs have an upper limit on the number of columns which can be computed.

3. GAX—Global adaptive extrapolation algorithm. Our adaptive algorithm begins with the initial interval $[a, b]$, the integrand function and absolute accuracy requirement ε and generates a, generally, nonuniform partition when it finally terminates. By an LQM (local quadrature module) we mean a rule which associates a pair of numbers LQE (local quadrature estimate for $\int_I f(x)\,dx$) and LEE (local error estimate for $|\mathrm{LQE}(I) - \int_I f(x)\,dx|$) with a given function f and interval I. To be specific we take an LQM which uses

$$\mathrm{LQE} := \mathrm{Tr}\,(I). \tag{3.1}$$

Since

$$\mathrm{Tr}\,(I)-\int_I f(x)\,dx=\frac{|I|^3}{12}f''(\zeta),\qquad \zeta\in I$$

if $f\in C^2(I)$ ($|I|=$ length of I), we take as LEE any rule, which is asymptotically consistent with this error expression in the following sense.

DEFINITION 3.2. A LEE is said to be asymptotically consistent with the trapezoid rule if whenever $f\in C^2[a,b]$, there exist constants c, $\varepsilon_0>0$ such that for all subintervals $I\subset[a,b]$

(1) $0\leqq \mathrm{LEE}\,(I)\leqq c|I|^3$;

and

(2) for all $0<\varepsilon<\varepsilon_0$, if $\mathrm{LEE}\,(I)\leqq\varepsilon$,

then

$$\left|T\left(\frac{|I|}{n};I\right)-\int_I f(x)\,dx\right|<\frac{\varepsilon}{n^2},\qquad n=1,2,\cdots. \tag{3.2}$$

This says, roughly, that the LEE we use will have the same qualitative behavior as the actual error in Tr (I).

By a CELL we mean an interval, more specifically the data defining the interval such as its endpoints or an endpoint and length, and at least the two numbers LQE, LEE. (Either LQE(I) or LQE_I are used for readability.)

$$\mathrm{CELL}:=\{[\alpha,\beta],\ \mathrm{LQE}_{[\alpha,\beta]},\ \mathrm{LEE}_{[\alpha,\beta]}\}. \tag{3.3}$$

A practical implementation may include other information such as the level of the cell,

$$\mathrm{LEVEL}_{[\alpha,\beta]}:=\log_2\left|\frac{b-a}{\beta-\alpha}\right|. \tag{3.4}$$

These data are combined to give certain global estimates

$$\mathrm{TEE}:=\sum_{\mathrm{CELLS}}\mathrm{LEE}, \tag{3.5}$$

$$\mathrm{Q}:=\sum_{\mathrm{CELLS}}\mathrm{LQE}, \tag{3.6}$$

the total error and quadrature estimates respectively.

These values change as the calculation progresses—new cells are created and old ones destroyed. We may then consider the following model calculation GAQ for global adaptive quadrature.

GAQ (global adaptive quadrature algorithm)
Input: $f,[a,b],\varepsilon,$ MAXLEV
Initialize: $[a,b]\to$ CELL
$\mathrm{Q}\leftarrow\mathrm{LQE}_{[a,b]}$
$\mathrm{TEE}\leftarrow\mathrm{LEE}_{[a,b]}$
$\mathrm{LEVEL}\leftarrow\mathrm{LEVEL}_{[a,b]}:=0$
While: TEE $>\varepsilon$ and LEVEL $<$ MAXLEV:
SUBDIVIDE WORST CELL IN HALF
UPDATE Q, TEE, LEVEL
RETURN Q, TEE, LEVEL

Here we have augmented the input to include MAXLEV which functions as a bound on the individual $\text{LEVEL}_{[\alpha,\beta]}$ and prevents subdivision beyond acceptable limits. Usually this can be calculated internally since it is related to machine precision but we include it as input for a subsequent algorithm, GAXI, which is introduced in § 5. The global parameter LEVEL is of course

$$\text{LEVEL} := \max \{\text{LEVEL}_{[\alpha,\beta]}\}. \tag{3.7}$$

The initialization step forms a cell from the initial interval and sets Q, TEE and LEVEL to the corresponding cell values. In an actual implementation there would be no need to have a single interval as input. Allowing more than one would permit a user to isolate singularities at endpoints and generalize "restarting" schemes which are already well known [15].

The iteration loop continues until TEE is driven below ε (success) or the maximum level is reached (failure). In this latter case we choose to terminate although practical implementation might accept the estimates on these small cells and continue. The subdivision step finds the cell with largest LEE, bisects it, and forms cells of each half, i.e., computes LQE and LEE for each subcell. The update amounts to reducing Q and TEE by the contribution due to the parent and adding in the contributions due to the two new cells. The larger cell is then destroyed.

Programs like GAQ are easy to write [16], [17]. In the absence of rounding error (on an infinite precision machine) their properties have been analyzed in detail [19]. The essential result for one-dimensional integrals is that the work required to obtain an estimate Q to absolute accuracy $\varepsilon \geqq \text{TEE}$ is proportional to $1/\sqrt{\varepsilon}$ using (3.1) as our LQM. Since work is usually measured in number of evaluations of the integrand, n, we have

$$\varepsilon \leqq \frac{K_f}{n^2} \tag{3.8}$$

where K_f is a constant independent of n.

This is the same order estimate that one would get using (2.3) with $h = (b-a)/n$, although the constant K_f in (3.8) is much better, a fact borne out in practice. One of the most interesting results of [19], however, is that (3.8) holds for a very wide class of functions including those with algebraic and branch point singularities. The error in (2.3), on the other hand, is well known to be of the form (3.8) only if $f \in C^2[a, b]$. In this paper we are only interested in smooth integrands. In fact we assume that f is sufficiently differentiable so that all derivatives appearing in expressions are continuous on $[a, b]$ unless otherwise specified. (For less smooth functions (3.8) must be replaced by more slowly decreasing expressions [6].) This combination of good error coefficient for smooth functions and convergence for poorly behaved functions accounts to a major extent for the popularity of the methods.

Compared to the results (2.10) this convergence is rather slow. Attempting to combine the adaptive features of GAQ with the convergence properties of Romberg quadrature suggests the following modifications.

At any point during the calculation a cell, x, is said to be SMALL if no other cell is smaller,

$$\text{CELL } x = \text{SMALL} \Leftrightarrow \text{LEVEL}_x \geqq \max_{\substack{y \in \text{CELL} \\ y \neq x}} \{\text{LEVEL}_y\}. \tag{3.9}$$

Any cell which is not SMALL is BIG. Thus after the initialization step the cell $[a, b]$ is SMALL. Figure 1 illustrates the concept by listing the cells and their level at one specific point in the course of a calculation.

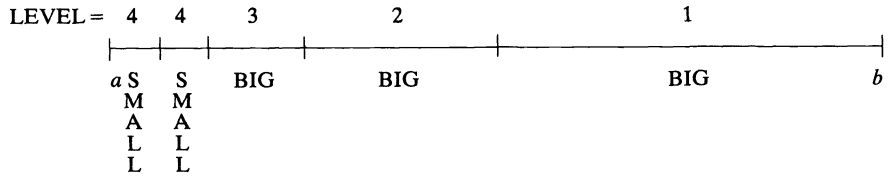


FIG. 1. *Big and small cells.*

We now define big cell error estimate (BEE) and extrapolated error estimate (EXE):

$$\text{BEE} := \sum_{\substack{\text{BIG}\\ \text{CELLS}}} \text{LEE}, \qquad \text{EXE} := \left| T_{m,m} - \int f(x)\,dx \right|. \tag{3.10}$$

In practice EXE is not computable. A realistic program might actually use (2.11) as an approximate value for EXE. The GAX algorithm terminates whenever either TEE (3.5) or EXE falls below the input ε.

The basic goal in GAX is to somehow reduce the number of integrand evaluations which are normally required to compute the elements of the first column of the Romberg table, $\{T_{m,0}\}$. This is done by replacing these numbers by approximations to them, denoted $\{\hat{T}_{m,0}\}$. $T_{m,0}$ estimates $\mathcal{T}$ by summing 2^m uniform trapezoidal rules, requiring 2^m+1 integrand evaluations in total. We will define $\hat{T}_{m,0}$ to also estimate $\mathcal{T}$ but now by a sum of nonuniform trapezoidal rules using (in general) a subset of the 2^m+1 evaluation points. Thus $T_{m,0}-\hat{T}_{m,0}$ will be given by a sum over the intervals used by $\hat{T}_{m,0}$, which we have termed the BIG cells in (3.9). To decide if the $\hat{T}_{m,0}$ is a good enough approximation to $T_{m,0}$ it will only be necessary to ask if the BIG intervals have been calculated accurately enough. If they appear to be sufficiently accurate we use $\hat{T}_{m,0}$ rather than $T_{m,0}$ as the element in the first Romberg column. In GAX we define "accurately enough" to be "BEE$\leqq\varepsilon$". If the BEE is too large we try to reduce it by doing an adaptive quadrature on the BIG cells. Ultimately BEE$<\varepsilon$. This is guaranteed because the number of BIG cells eventually decreases as the subdivision progresses. In fact if we go far enough the only cells are SMALL and $\hat{T}_{m,0}=T_{m,0}$. In that case there is no reduction in the number of function evaluations.

An exact description of the algorithm is given below.

GAX (global adaptive extrapolation algorithm)
Input: f, $[a,b]$, ε, MAXLEV, K
Initialize: $[a,b]\rightarrow$CELL
$Q\leftarrow \text{LQE}_{[a,b]}$, TEE$\leftarrow \text{LEE}_{[a,b]}$
$\text{LEVEL}\leftarrow \text{LEVEL}_{[a,b]}=0$, $m\leftarrow 0$
BEE$\leftarrow 0$, EXE$\leftarrow$TEE, $\hat{T}_{0,0}\leftarrow Q$
While TEE$>\varepsilon$ **and** EXE$>\varepsilon$ **and** LEVEL$<$MAXLEV:
Subdivide worst CELL in half
Update Q, TEE, BEE, LEVEL
If a new LEVEL introduced:
While BEE$>\varepsilon$ and TEE$>\varepsilon$:
Subdivide worst BIG CELL in half
Update Q, TEE, BEE
$m\leftarrow m+1$, $\hat{T}_{m,0}\leftarrow$Q
Compute the diagonal $\{\hat{T}_{m,k}\,1\leqq k\leqq \bar{k}\}$, $\bar{k}:=\min(K,m)$ of T-table.
Find an estimate EXE for $|\hat{T}_{m,\bar{k}}-\int f(x)\text{dx}|$

If TEE > EXE: Return LEVEL, $\hat{T}_{m,\bar{k}}$, EXE
Else: Return LEVEL, Q, TEE
End

Note that since the higher level subdivide step creates new cells and destroys an old one it may create cells smaller than any seen before. In that case all the remaining cells become BIG. That is, the concepts of BIG/SMALL are not inherent to the cell but change during the algorithm.

The parameter K of GAX limits the number of columns that can be computed by GAX. Usually $K <$ MAXLEV.

Our hope with this algorithm is that where f doesn't change much, a few function values will suffice for that part of the contribution to $T_{m,0}$ but that the high rate of convergence of the T-table will be retained. The FORTRAN subroutine QAGS by deDoncker et al. [14] motivated much of our work. GAX is similar in spirit (we hope) to QAGS but as the latter is a library quality program, a great many important differences exist.

The following simple example may be helpful. The initial interval $[a, b]$ is input. It is SMALL and its LQE is exactly $T_{0,0}$. (See Step (i) in Fig. 2). If LEE (=TEE) $> \varepsilon$ we subdivide (step (ii)), enabling us to compute $T_{1,0}$ and $T_{1,1}$. Suppose the right half has the larger LEE. Hence we subdivide it (iii). The left half is now BIG, but supposing its LEE $< \varepsilon$ we compute $\hat{T}_{2,0}$, pretend it is equivalent to $T_{2,0}$ and use it to compute $\hat{T}_{2,1}$ and $\hat{T}_{2,2}$. The worst cell is now the right quarter interval leading to (iv). Now if the two BIG cells have BEE $> \varepsilon$ we adapt on them, choosing the worst to subdivide (v), at which point BEE $< \varepsilon$ and we compute $\hat{T}_{3,0} \cdots \hat{T}_{3,3}$.

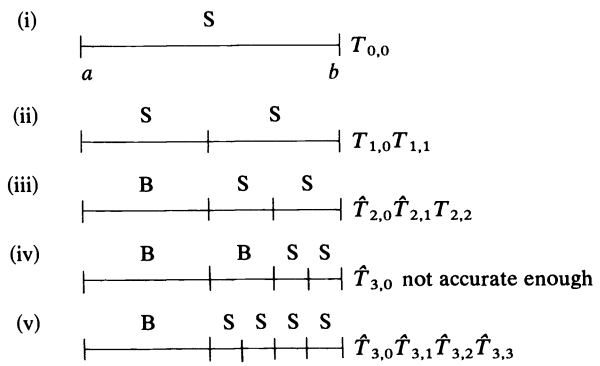


FIG. 2.

As a final remark note that when $[(a+b)/2, (a+3b)/4]$ is subdivided we could go back and update $\hat{T}_{2,0}$, $\hat{T}_{2,1}$, $\hat{T}_{2,2}$. GAX does not do this.

Several other possibilities exist in addition to the test "BEE $> \varepsilon$?" Among these are

$$\text{(3.11)} \qquad \begin{gathered} \text{"BEE} > \alpha\varepsilon\text{?"}, \qquad 0 < \alpha \leqq 1, \\ \text{"BEE} > \varepsilon \cdot \frac{\text{Sum of lengths of BIG intervals}}{b-a}\text{?"} \end{gathered}$$

Using either of these means that we are *more* likely to subdivide the BIG cells.

4. Convergence of GAX. Here we consider the effect of using GAX with a fixed smooth function as $\varepsilon \downarrow 0$. In addition to (3.1), (3.2) we also assume that GAX uses the exact error of $\hat{T}_{m,\bar{K}}$ as EXE:

$$\text{EXE} := \left| \hat{T}_{m,\bar{K}} - \int_a^b f(x)\,dx \right| \tag{4.1}$$

and that the test $\text{BEE} > \varepsilon$ of GAX is replaced by the slightly sharper test $\text{BEE} > \alpha\varepsilon$, with $0 < \alpha \leqq \frac{1}{4}$.

In particular we examine the finiteness of GAX and the amount of work (measured in terms of f evaluations) needed up to termination.

THEOREM 4.2. *Let* MAXLEV $:= \infty$, $f \in C^{2N+2}[a, b]$, $N \geqq 0$ *and let* $K \leqq N$ *be the maximum number of columns* GAX *is allowed to compute. Then*

1. *for all* $\varepsilon > 0$ GAX *terminates with an approximation to* $\int_a^b f(x)dx$ *whose error is at most* ε. *The total number* n *of function evaluations needed by* GAX *until termination is related to* ε *by*

$$\varepsilon = O\left(\frac{1}{n^{2K+2}}\right). \tag{4.3}$$

2. *If in addition* $f''(x)$ *does not vanish in* $[a, b]$, *then for all sufficiently small* $\varepsilon > 0$ *the T-tableau generated by* GAX *is equal to the Romberg T-tableau,* $\hat{T}_{m,0} = T_{m,0}$ *for all* $m \geqq 0$.

Theorem 4.2 has a positive and a negative aspect: Part 1 says that GAX is at least as efficient as Romberg's method for all sufficiently smooth functions; but according to part 2, it is not better than Romberg's method for a broad subclass of these functions, if $\varepsilon > 0$ is sufficiently small.

Proof. Let MAXLEV $:= \infty$ and $f \in C^{2N+2}[a, b]$, $0 \leqq K \leqq N$ and $\varepsilon > 0$. Denote by $M(\varepsilon)$ the maximal cell level constructed by GAX. Since by (3.1), (3.2) and (3.5)

$$\text{LEE}(I) = O(|I|^3), \quad \text{TEE} = \sum_{x \in \text{CELL}} \text{LEE}_x,$$

it follows that the condition $\text{TEE} \leqq \varepsilon$ is eventually satisfied, if by the successive subdivisions of the worst cells the size $|I|$ of these cells has become sufficiently small, $|I| = O(\sqrt{\varepsilon})$. Hence for any $\varepsilon > 0$, $M(\varepsilon)$ is finite; in particular,

$$2^{M(\varepsilon)} = O(1/\sqrt{\varepsilon}), \tag{4.4}$$

and the program terminates. Clearly, because of (3.1), (4.1), GAX computes the exact integral $\int_a^b f(x)\,dx$ up to an absolute error ε.

Since the number $n = n(\varepsilon)$ of function evaluations needed until termination is related to $M(\varepsilon)$ by

$$n \leqq 2^{M(\varepsilon)+1},$$

equation (4.4) implies the crude relation

$$\varepsilon = O(1/n^2).$$

In order to prove the better estimate (4.3), we have to study the behavior of GAX more closely. Consider the state of GAX at the point in time t_m, at which LEVEL $= m$

and $\hat{T}_{m,0}$ has just been computed and let

$$I_m^B := \{\text{BIG intervals at time } t_m\},$$
$$I_m^S := \{\text{SMALL intervals at time } t_m\},$$

$$|I_m^B| := \sum_{I \in I_m^B} |I|, \qquad |I_m^S| := \sum_{I \in I_m^S} |I|, \tag{4.5}$$

$$J_m^B := \sum_{I \in I_m^B} \int_I f(x)\,dx, \qquad J_m^S := \sum_{I \in I_m^S} \int_I f(x)\,dx.$$

Then by definition of $\hat{T}_{m,0}$

$$\hat{T}_{m,0} = \sum_{I \in I_m^B} \mathrm{Tr}\,(I) + \sum_{I \in I_m^S} \mathrm{Tr}\,(I), \tag{4.6}$$

whereas the corresponding element $T_{m,0}$ of the Romberg tableau (4.6) is given by (see 2.3)

$$T_{m,0} = T(h_m) = \sum_{I \in I_m^B} T(h_m; I) + \sum_{I \in I_m^S} \mathrm{Tr}\,(I). \tag{4.7}$$

We now estimate the difference $F_m := \hat{T}_{m,0} - T_{m,0}$. Since for each $I \in I_m^B$,

$$|h_m| \leqq \frac{|I|}{2}, \qquad 0 \leqq \mathrm{LEE}\,(I) \leqq \mathrm{BEE} = \sum_{I' \in I_m^B} \mathrm{LEE}\,(I') \leqq \alpha\varepsilon \leqq \frac{\varepsilon}{4}$$

and LEE is asymptotically consistent (3.2), it follows for sufficiently small ε and all $I \in I_m^B$ that

$$\left| T(h_m, I) - \int_I f(x)\,dx \right| \leqq \mathrm{LEE}\,(I) \left(\frac{h_m}{|I|}\right)^2 \leqq \frac{\mathrm{LEE}\,(I)}{4}$$

and therefore

$$\begin{aligned} |F_m| = |\hat{T}_{m,0} - T_{m,0}| &= \left| \sum_{I \in I_m^B} (\mathrm{Tr}\,(I) - T(h_m, I)) \right| \\ &\leqq \left| \sum_{I \in I_m^B} \mathrm{Tr}\,(I) - J_m^B \right| + \left| \sum_{I \in I_m^B} (T(h_m, I) - \int_I f(x)\,dx) \right| \\ &\leqq \mathrm{BEE} + \sum_{I \in I_m^B} \mathrm{LEE}\,(I)/4 \\ &\leqq \tfrac{5}{4} \cdot \mathrm{BEE} \leqq \tfrac{5}{16}\varepsilon < 0.3\varepsilon. \end{aligned} \tag{4.8}$$

Because of (2.8),

$$\hat{T}_{m,k} = \sum_{j=0}^{k} c_{kj} \hat{T}_{m-k+j,\,0}, \; T_{m,k} = \sum_{j=0}^{k} c_{kj} T_{m-k+j,\,0},$$

so that by (2.9), (4.8) the error $\Delta_{m,k}$ of $\hat{T}_{m,k}$ satisfies

$$\begin{aligned} \Delta_{m,k} := \hat{T}_{m,k} - \mathscr{T} &= \sum_{j=0}^{k} c_{kj} F_{m-k+j} + T_{m,k} - \mathscr{T}, \\ |\Delta_{m,k}| &\leqq 0.6\varepsilon + |T_{m,k} - \mathscr{T}|. \end{aligned} \tag{4.9}$$

Now for $f \in C^{2N+2}[a, b]$ and fixed $k \leqq N$, the error $T_{m,k} - \mathcal{T}$ of $T_{m,k}$ satisfies a bound of the form ([2])

$$|T_{m,k} - \mathcal{T}| \leqq \frac{c_k}{(4^m)^{k+1}} \quad \text{for all } m \geqq k,$$

with certain constants c_k; hence by (4.9)

$$|\Delta_{m,k}| \leqq 0.6\varepsilon + c_k/4^{(k+1)m}, \qquad m \geqq k. \tag{4.10}$$

GAX will terminate as soon as $|\Delta_{m,k}| \leqq \varepsilon$, so that by (4.10) the terminating level $M(\varepsilon)$ of GAX satisfies $M(\varepsilon) \leqq m(\varepsilon)$, where $m(\varepsilon)$ is the smallest integer $m \geqq 0$ with

$$2^m \geqq (c_k/(0.4\varepsilon))^{1/(2K+2)}. \tag{4.11}$$

Since the number of function evaluations needed for the construction of level $M(\varepsilon)$ is $n = n(\varepsilon) = 2^{M(\varepsilon)} + 1$, it follows from (4.11) that

$$\varepsilon = O(1/n^{2K+2}),$$

which proves (4.3) and the first part of Theorem 4.2. In order to prove the second part, assume that $f''(x)$ does not vanish in $[a, b]$. We will show that for sufficiently small ε, there are no BIG cells up to termination, $I_m^B = \varnothing$ for all $m \leqq M(\varepsilon)$, so that $T_{m,0} = \hat{T}_{m,0}$.

We assume $0 < \varepsilon \leqq c_k/0.4$ and $\varepsilon \leqq \varepsilon_0$ (3.2). Then by definition of $m(\varepsilon)$

$$\left(\frac{c_k}{0.4\varepsilon}\right)^{1/(2K+2)} \geqq \tfrac{1}{2} 2^{m(\varepsilon)}. \tag{4.12}$$

Suppose that there is a level $m \leqq M(\varepsilon) \leqq m(\varepsilon)$ such that $I_m^B \neq \varnothing$. Then for any $I \in I_m^B$ by (3.2) and (4.12)

$$\begin{aligned}
\frac{\varepsilon}{4} \geqq \mathrm{BEE} \geqq \mathrm{LEE}(I) &\geqq \left| \mathrm{Tr}(I) - \int f(x)\,dx \right| \\
&\geqq \frac{|I|^3}{12}\mu \qquad \left(\mu := \min_{\xi \in [a,b]} |f''(\xi)| > 0\right) \\
&\geqq \left(\frac{b-a}{2^m}\right)^3 \cdot \frac{\mu}{12} \\
&\geqq \frac{(b-a)^3}{12} \cdot \mu \cdot \frac{1}{8}\left(\frac{0.4}{c_k}\right)^{3/(2K+2)} \cdot \varepsilon^{3/(2K+2)} \\
&> \varepsilon
\end{aligned}$$

for small enough ε. This contradiction shows $I_m^B = \varnothing$ for all $m \leqq M(\varepsilon)$, which completes the proof of the theorem. □

5. Numerical results. We have programmed a version of GAX in FORTRAN double precision for the Univac 1108. The actual implementation required use of a LEE and EXE different from (3.2) and (3.10). Rather, we used (2.11) for EXE and a consistent, locally third order estimate for LEE. Our simplified model program made no attempt to save and reuse f values. Additionally we used heap subroutines [17] to maintain the cells as it was easy to write the programs with them. In our tests

we considered two smooth problems:

$$\int_0^1 \frac{dx}{1+25x^2}, \tag{5.1}$$

$$\int_0^1 e^{5x}\,dx. \tag{5.2}$$

We attempted to integrate these for a sequence of ε's. In Table 1 we list the results of the calculation for $\varepsilon = 1.E\text{-}09$. The computer output is provided when each new extrapolation takes place, that is, after BEE$\leqq\varepsilon$. We list the total number of cells, $|\mathcal{T}\text{-EXE}|$ and TEE. In both cases the number of cells at the mth level is 2^m as predicted by Theorem 4.2.

For comparison we also integrated 3 nonsmooth functions

$$\int_0^1 \sqrt{x}\,dx, \tag{5.3}$$

$$\int_0^1 \frac{dx}{\sqrt{x}}, \tag{5.4}$$

$$\int_0^1 x \ln x\,dx. \tag{5.5}$$

For these TEE and EXE go to zero at roughly the same rate. In fact TEE should decrease faster, but apparently we are not yet in the asymptotic region. For these functions we have also that, almost, 2^m cells exist at the mth level.

The rather negative result of Theorem 4.2 seems to fly in the face of user experience with programs such as [14]: a substantial body of users have concluded that these programs are very efficient for solving their problems. To try and glean some insight into these matters, we conducted two additional experiments with suitably modified versions of GAX, closer in spirit to these production programs.

First we replaced the trapezoidal rule LQM with one of much higher polynomial order. In particular we used QUARUL [14], an LQM often used in production software. Its LQE is "exact" for polynomials through degree 19 [18]. As a result the smooth integrands (5.1) and (5.2) are essentially done exactly, in one step. Thus, no extrapolation is necessary. The singular functions (5.3)–(5.5) are also done more accurately than with the trapezoidal rule but there is no improvement via the Romberg extrapolation because of Difficulty b) in § 1.

The second alteration to GAX involved replacing the linear Romberg extrapolation by the nonlinear ε-algorithm as was also used in [14]. This method is known to be useful for extrapolating sequences of quadrature estimates of singular functions in much the same way that Romberg is useful for extrapolating estimates of smooth functions. The combination of a high order LQM and the ε-algorithm make the modified program much closer to those described in [14], and Table 2 illustrates the results on the integrands (5.3)–(5.5). (Integrands (5.1), (5.2) are still done almost exactly at the first step.) We see there a very dramatic improvement—a rapid decrease in the error due to the extrapolation on the nonuniform mesh. Notice that each subdivision produces a new level. What happens is this: for any cell that does not include the point $x = 0$, the LQM produces a very accurate LQE and a small LEE. Thus BEE$<\varepsilon$, the only adaptation is done by subdividing the interval with the origin as its left boundary, and each such subdivision produces a new level and allows an

TABLE 1

Level	# Subintervals	$\lvert\mathcal{I}$-EXE$\rvert$	EXE Romberg error est.	TEE Total error est.
	1./(1.+25*X**2)			
1.	2	.129-01	.835-01	.167-01
2.	4	.575-03	.131-01	.327-02
3.	8	.906-04	.676-03	.213-02
4.	16	.102-05	.919-04	.524-03
5.	32	.595-08	.101-05	.130-03
6.	64	.451-10	.600-08	.326-04
	EXP (5.*X)			
1.	2	.336+00	.170+02	.340+01
2.	4	.306-02	.353+00	.929+00
3.	8	.745-05	.310-02	.237+00
4.	16	.458-08	.747-05	.598-01
5.	32	.708-12	.459-08	.149-01
	DSQRT (X)			
1.	2	.101-01	.662-01	.132-01
2.	4	.315-02	.742-02	.494-02
3.	8	.108-02	.210-02	.181-02
4.	16	.379-03	.704-03	.659-03
5.	32	.133-03	.245-03	.237-03
6.	64	.473-04	.866-04	.849-04
7.	128	.167-04	.306-04	.302-04
8.	253	.592-05	.108-04	.107-04
9.	483	.210-05	.382-05	.383-05
	1./DSQRT (X)			
1.	2	.629+00	.516+00	.112+00
2.	4	.432+00	.208+00	.804-01
3.	8	.303+00	.130+00	.571-01
4.	16	.214+00	.897-01	.404-01
5.	32	.151+00	.629-01	.286-01
6.	64	.107+00	.444-01	.202-01
7.	128	.758-01	.314-01	.143-01
8.	255	.536-01	.222-01	.101-01
9.	502	.379-01	.157-01	.716-02
	X*LOG (X)			
1.	2	.479-02	.899-01	.179-01
2.	4	.962-03	.406-02	.539-02
3.	8	.229-03	.744-03	.157-02
4.	16	.566-04	.173-03	.449-03
5.	32	.141-04	.425-04	.126-03
6.	64	.352-05	.105-04	.351-04
7.	128	.881-06	.264-05	.967-05
8.	255	.231-06	.650-06	.264-05
9.	505	.610-07	.170-06	.721-06

TABLE 2

Level	# Subintervals	$\lvert\mathcal{I}$-EXE$\rvert$	EXE Romberg error est.	TEE total error est.
	DSQRT (X)			
1.	2	.169−05	.200+01	.175−02
2.	3	.598−06	.133+01	.618−03
3.	4	.346−17	.666+00	.218−03
4.	5	.173−17	.229−05	.773−04
5.	6	.173−17	.598−06	.273−04
6.	7	.260−17	.578−17	.966−05
	1./DSQRT (X)			
1.	2	.229−01	.593+01	.673+00
2.	3	.162−01	.397+01	.476+00
3.	4	.173−16	.203+01	.336+00
4.	5	.242−16	.391−01	.238+00
5.	6	.346−17	.162−01	.168+00
6.	7	.346−17	.485−16	.119+00
	X*LOG (X)			
1.	2	.906−07	.750+00	.173−03
2.	3	.226−07	.500+00	.296−04
3.	4	.130−17	.250+00	.594−05
4.	5	.173−17	.113−06	.127−05
5.	6	.173−17	.226−07	.283−06
6.	7	.130−17	.216−17	.643−07

immediate extrapolation. Of course, if ε were small enough, BEE would be $\geqq \varepsilon$ and additional subdivision would then take place.

Theorem 4.2 cannot be expected to apply to either of these modifications or to production software. But these numerical and theoretical results together do suggest the following hypotheses:

H1. The outstanding performance of production extrapolated adaptive quadrature programs is due largely to the very high order LQM used therein.

H2. For small enough ε these programs will exhibit behavior similar to that predicted by Theorem 4.2 (level $m \to 2^m$ cells).

H3. No fixed order LQM will alter the asymptotic results of Theorem 4.2, but the higher the order the smaller will ε have to be before these results appear in practice.

Examination of the innermost "WHILE" loop in GAX will easily show that this is just GAQ applied to the BIG cells. The convergence rate of the LQM governs the rate of convergence of GAQ [16], [19]. This rate is slower than that of the Romberg algorithm and will limit the performance of the code. Thus, to effect an overall asymptotic improvement, we ought to replace GAQ with something having Romberg-like convergence, for example GAX. Thus we consider GAXI:

GAXI (a modification of GAX)
replace
"Subdivide worst BIG CELL in half"
with
CALL GAXI with input:
f, worst BIG CELL, ε, LEVEL-LEVEL$_{\text{(worst BIG CEL}}$

H4. GAXI has improved asymptotic convergence.

REFERENCES

[1] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Academic Press, New York, 1975.

[2] F. L. BAUER, H. RUTISHAUSER AND E. L. STIEFEL, *New aspects in numerical quadrature* in Experimental Arithmetic, High Speed Computing and Mathematics, American Mathematical Society, Providence, RI, 1963, pp. 199–217.

[3] J. N. LYNESS, *An algorithm for Gauss Romberg integration*, BIT, 12 (1972), pp. 194–203.

[4] R. BULIRSCH, *Bemerkungen zur Romberg-Integration*, Numer. Math., 6 (1964), pp. 6–16.

[5] T. HAVIE, *On the practical application of the modified Romberg algorithm*, BIT, 7 (1967), pp. 103–113.

[6] J. N. LYNESS AND B. W. NINHAM, *Numerical quadrature and asymptotic expansions*, Math. Comp., 21 (1967), pp. 162–178.

[7] J. N. LYNESS, *Applications of extrapolation techniques to multidimensional quadrature of some integrand functions with a singularity*, J. Comp. Phys., 20 (1976), pp. 346–363.

[8] ———, *When not to use an automatic quadrature routine*, SIAM Rev., to appear.

[9] R. BULIRSCH AND J. STOER, *Handbook series numerical integration: Numerical quadrature by extrapolation*, Numer. Math., 9 (1967), pp. 271–278.

[10] C. DE BOOR, *On writing an automatic integration algorithm*, in Mathematical Software, J. R. Rice, ed., Academic Press, New York, 1971, pp. 201–209.

[11] D. K. KAHANER, *Numerical quadrature by the ε-algorithm*, Math. Comp., 26 (1972), pp. 689–693.

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

[13] A. GENZ, *An adaptive multidimensional quadrature procedure*, Comp. Phys. Comm., 4 (1972), pp. 11–15.

[14] E. DE DONCKER, *An adaptive extrapolation algorithm for automatic integration*, SIGNUM Newsl., 13 (1978), pp. 189–198.

[15] D. K. KAHANER AND M. B. WELLS, *An experimental algorithm for N-dimensional adaptive quadrature*, ACM Trans. Math. Software 5 (1979), pp. 86–96.

[16] M. A. MALCOLM AND R. B. SIMPSON, *Local versus global strategies for adaptive quadrature*, ACM Trans. Math. Software, 1 (1975), pp. 129–146.

[17] D. K. KAHANER, *Algorithm 561—Fortran implementation of heap programs for efficient table maintenance*, ACM Trans. Math. Software, 6 (1980), pp. 444–449.

[18] P. RABINOWITZ, *The exact degree of precision of generalized Gauss–Kronrod integration rules*, Math. Comp., 35 (1980), pp. 1275–1283.

[19] C. DE BOOR AND J. R. RICE, *An adaptive algorithm for multivariate approximation giving optimal convergence rates*, J. Approx. Theory, 25 (1979), pp. 337–359.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0003 $01.25/0

A NUMERICAL METHOD FOR SOLVING INVERSE REAL SYMMETRIC EIGENVALUE PROBLEMS*

J. Y. WANG† AND B. S. GARBOW†

Abstract. We present a method using Newton iteration and least squares techniques to solve inverse real symmetric eigenvalue problems. A matching algorithm is provided for cases where the number of prescribed eigenvalues is less than the dimension of the matrix. We consider the convergence of the method and present some numerical examples.

Key words. eigenvalue, inverse, least squares, symmetric

1. Introduction. Inverse eigenvalue problems for real symmetric tridiagonal matrices have been studied extensively [2], [3], [5], [9]. Among these works, [5] provides a direct method that is also shown to be stable. These methods require an additional set of specified eigenvalues associated with a corresponding principal submatrix so that certain three-term recurrence relations can be applied. Similar ideas have been extended in [2] to solve inverse eigenvalue problems for band matrices. In this report we formulate the inverse eigenvalue problem as one of nonlinear optimization, whereby general real symmetric matrices can be handled.

We consider the following inverse problem. Determine parameters p_i^*, $i = 1, 2, \cdots, m$, such that the composite matrix A, expressible as

$$A = A_0 + \sum_{i=1}^{m} p_i^* A_i, \tag{1}$$

has a prescribed set of eigenvalues λ_j^*, $j = 1, 2, \cdots, m1$, where A_i are prescribed real symmetric $n \times n$ matrices, p_i^* are real numbers, and $m \leqq m1 \leqq n$.

This problem arises frequently in applied physics. For instance [16], the A_i may be energy matrices and the eigenvalues of A energy levels obtained by conducting some physical experiments. Some special cases of this problem have been discussed in [6], [13] and [19].

There are situations where only certain eigenvalues are obtainable; others are either too expensive to determine or are mixed together with no easy way to separate them. In this report we also consider the case where the number of prescribed eigenvalues is less than the dimension of the matrix. We develop an algorithm for matching the computed eigenvalues with the prescribed eigenvalues. A Fortran program for numerical computation has been written and is available upon request.

2. General properties. In [1], Z. Bohte proved that a simple eigenvalue λ_k of A defined by (1) can be expressed as a convergent power series of $p_1, \cdots, p_m$ in some neighborhood of $p_1 = \cdots = p_m = 0$. Moreover, if μ_k is the corresponding simple eigenvalue of A_0 with associated normalized eigenvector y_k and

$$M = \max_{1 \leqq i \leqq m} \|A_i\|, \qquad d = (n-1)/\min_{s \neq k} |\mu_s - \mu_k|, \tag{2}$$

* Received by the editors August 2, 1979, and in revised form August 12, 1981. This work was supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under contract W-31-109-Eng-38.

† Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois 60439.

he showed that if $|p_i| \leqq 1/(4mMd)$, then

$$\lambda_k = \mu_k + \sum_{i=1}^{m} y_k^T A_i y_k p_i + G_k(p_1, \cdots, p_m), \qquad k = 1, 2, \cdots, n, \tag{3}$$

where G_k contains only nonlinear terms in $p_1, \cdots, p_m$, and further that

$$\left(\frac{\partial \lambda_k}{\partial p_i}\right)^{(0)} = y_k^T A_i y_k \tag{4}$$

and

$$\left|\left(\frac{\partial^2 \lambda_k}{\partial p_j \partial p_i}\right)^{(0)}\right| \leqq 2 \cdot M^2 \cdot d = L, \qquad i, j = 1, \cdots, m, \quad k = 1, \cdots, n. \tag{5}$$

In this paper we assume that the eigenvectors are normalized (l_2 norm) and that the eigenvalues are stored in nonincreasing order.

If the number of prescribed eigenvalues is equal to the dimension of A and $m = n$, formulate the inverse eigenvalue problem as a system of nonlinear equations

$$f_k(p_1, \cdots, p_m) = 0, \qquad k = 1, \cdots, n, \tag{6}$$

where

$$f_k(p_1, \cdots, p_m) = \lambda_k - \lambda_k^*, \qquad \lambda_k = x_k^T A x_k, \tag{7}$$

λ_k is an eigenvalue of A, and x_k is the normalized eigenvector of A associated with λ_k.

In the case where the number of prescribed eigenvalues is less than the dimension of A, we formulate the problem as follows. Let $\{\lambda_j^*\} = \{\lambda_1^* > \cdots > \lambda_{m1}^*\}$ be the prescribed eigenvalues and $\{\lambda_k\} = \{\lambda_1 \geqq \cdots \geqq \lambda_n\}$ be the computed eigenvalues of A, where $m \leqq m1 < n$. Let

$$H = \{h_1 \geqq \cdots \geqq h_{m1}\} \tag{8}$$

be the subset of $\{\lambda_k\}$ that matches the set $\{\lambda_j^*\}$ in the sense that

$$\sum_{j=1}^{m1} (\lambda_j^* - h_j)^2$$

is no greater than the corresponding sum for any other subset of $\{\lambda_k\}$. We formulate the inverse eigenvalue problem to minimize the system of nonlinear equations

$$f_j(p_1, \cdots, p_m), \qquad j = 1, \cdots, m1, \tag{9}$$

where

$$f_j(p_1, \cdots, p_m) = h_j - \lambda_j^*, \qquad h_j = x_j^T A x_j,$$

and x_j is the normalized eigenvector of A associated with the computed eigenvalue h_j.

3. Numerical methods. In the case where $m = m1 = n$, we use the general Newton method on (6) and arrive at the following linear system:

$$\sum_{i=1}^{m} b_{ki}^{(\nu)} \cdot \Delta p_i = \lambda_k^* - \lambda_k^{(\nu)}, \qquad k = 1, \cdots, n, \tag{10}$$

where

$$b_{ki}^{(\nu)} = (x_k^{(\nu)})^T \cdot A_i \cdot x_k^{(\nu)} \tag{11}$$

and $\{\lambda_k^{(\nu)}, x_k^{(\nu)}\}$ is an eigenpair of the matrix

$$A^{(\nu)} = A_0 + \sum_{i=1}^{m} p_i^{(\nu)} A_i. \tag{12}$$

Here $\{\lambda_k^{(\nu)}\}$ are arranged in nonincreasing order.

The new approximation to the parameters is then defined by

$$p_i^{(\nu+1)} = p_i^{(\nu)} + \Delta p_i, \qquad i = 1, \cdots, m, \tag{13}$$

where Δp_i is obtained by solving linear system (10). This procedure is repeated until either the Δp_i become sufficiently small or the number of iterations exceeds a preset maximum.

If $m < m1 = n$, we consider the problem of finding $P^* = (p_1^*, \cdots, p_m^*)$ which minimizes

$$\sum_{i=1}^{n} |f_i(p)|^2,$$

where $f_i(p)$ is defined in (7). By using the Gauss–Newton method on (6), we obtain the following linear system:

$$\sum_{j=1}^{m} \left(\sum_{k=1}^{n} b_{ki}^{(\nu)} b_{kj}^{(\nu)} \right) \cdot \Delta p_j = \sum_{k=1}^{n} b_{ki}^{(\nu)} (\lambda_k^* - \lambda_k^{(\nu)}), \qquad i = 1, \cdots, m, \tag{14}$$

where $b_{ki}^{(\nu)}$ is defined in (11). The new approximation to the parameters is then defined by (13).

If $m \leqq m1 < n$, we use the matching algorithm given in § 5 to determine the subset H in (8) and then use the Gauss–Newton method to obtain the following linear system:

$$\sum_{j=1}^{m} \left(\sum_{k=1}^{m1} b_{ki}^{(\nu)} b_{kj}^{(\nu)} \right) \cdot \Delta p_j = \sum_{k=1}^{m1} b_{ki}^{(\nu)} (\lambda_k^* - h_j^{(\nu)}), \qquad i = 1, \cdots, m, \tag{15}$$

where $b_{ki}^{(\nu)}$ is defined in (11) and $(h_k^{(\nu)}, x_k^{(\nu)})$ is an eigenpair of $A^{(\nu)}$. The new approximation to the parameters is then defined by (13).

It is possible that during the iterative procedure two of the computed eigenvalues may be exactly identical, e.g., $h_{m1}^{(\nu)} = h_{m1+1}^{(\nu)}$. In this case, we merely use one of the two associated eigenvectors and hope that the iteration will move away from this situation.

4. Convergence of the method. If we express system (6) as

$$F(P) = 0, \tag{16}$$

where $P = (p_1, \cdots, p_m)$, then the derivative of F is represented by the Jacobian matrix defined in (11). Define the natural norm of a matrix A, denoted by $\|A\|$, as

$$\|A\| = \max \{\|Ax\|; \|x\| = 1\}, \tag{17}$$

where x is a vector. We make use of the following lemma.

LEMMA 1. *Assume that conditions* (2), (3) *hold. Then there exist* γ, δ *such that*

$$\|F'(x) - F'(y)\|_2 \leqq \gamma \|x - y\|_2 \tag{18}$$

for all $x, y \in E_0$, *where* $E_0 = \{z \in R^m : \|z - P^{(0)}\|_2 < \delta\}$.

Proof. It follows from (5) that

$$\|F''(P^{(0)})\|_1 \leqq n \cdot L \text{ and } \|F''(P^{(0)})\|_\infty \leqq m \cdot L.$$

Since $\|F''(P^{(0)})^2\|_2 \leqq \|F''(P^{(0)})\|_1 \cdot \|F''(P^{(0)})\|_\infty \leqq m \cdot n \cdot L^2$, we have

$$\|F''(P^{(0)})\|_2 \leqq n \cdot L.$$

Then by [17, p. 78], the proof of the lemma is completed. □

Using the Newton–Kantorovich theorem stated in [17], we next prove the following theorem.

THEOREM 1. *Assume that the conditions of Lemma* 1 *hold. Define* $B = (b_{ki}^{(0)})$ *as in* (11) *with* $\|B^{-1}\|_2 = N$. *Then if*

$$N^2 \cdot \gamma \cdot l \leqq \tfrac{1}{2}, \tag{19}$$

the Newton iteration procedure (10) *converges quadratically to a solution in* E_0, *where* E_0, γ *are defined in Lemma* 1, *and* $l = \sqrt{n} \cdot \max_i |\lambda_i^* - \lambda_i^{(0)}|$ *where* $\lambda_i^{(0)}$ *are the eigenvalues of* $A^{(0)}$.

Proof. From the conditions of the theorem,

$$\|F'(P^{(0)})^{-1}\|_2 = N,$$

and

$$\|F'(P^{(0)})^{-1}F(P^{(0)})\|_2 \leqq \|F'(P^{(0)})^{-1}\|_2 \cdot \|F(P^{(0)})\|_2 \leqq N \cdot l.$$

The proof then follows by using a theorem in [17, p. 421]. □

In the case $m < m1 = n$, the problem is solved in the least-squares sense. As an application of the theorems in [4], the following theorem is true.

THEOREM 2. *Assume that the conditions of Lemma* 1 *hold. If for some* $P^* \in E_0$, $J^T \cdot F(P^*) = 0$ *and* $\gamma \cdot \|F(P^*)\|_2$ *is a strict lower bound for the spectrum of* $J^T \cdot J$, *then the procedure* (14) *converges locally to* P^* *and the convergence is of order at least one, where* J *is the Jacobian of* F *evaluated at the point* P^* *and* γ, E_0 *are defined in Lemma* 1. *Moreover, if* $F(P^*) = 0$, *i.e.,* P^* *is a zero of* F, *then* (14) *converges quadratically.*

The above theorems demonstrate that the domain of convergence may be very small if λ_s is close to λ_k for $s \neq k$. However, in practice, the process converges from much poorer initial approximations.

In the case $m1 < n$, a matching algorithm should be used. Since such an algorithm is a discrete mapping, the error analysis is difficult; but our numerical experience indicates that if the initial guess is sufficiently close to a solution, then procedure (15) converges to the solution. In general $F(P^*) \neq 0$, in which case the convergence rate is less than quadratic.

5. A matching algorithm. In the case where the number of prescribed eigenvalues is less than the dimension of matrix A, we have to match the computed eigenvalues to the prescribed eigenvalues. A matching algorithm is necessary for the procedure to ensure that the right-hand side of (15) approaches zero.

Matching problem. Given two sets of nonincreasing real numbers

$$C = (c_1, \cdots, c_{m1}), \qquad D = (d_1, \cdots, d_n), \tag{20}$$

where $n \geqq m1$, determine a subset D' of D,

$$D' = \{d'_1, \cdots, d'_{m1}\},$$

such that

$$\sum_{j=1}^{m1} (c_j - d'_j)^2 \tag{21}$$

is minimal.

It follows from [17, p. 108] that, if $m1 = n$, then the set D' with $d'_j = d_j$, $j = 1, \cdots, n$, is a minimizer of (21). If $m1 < n$, it takes at most ${}_nC_{m1}$ tries to find a minimizer of (21) by exhaustion.

In our case the matching problem occurs in an iterative procedure. We propose a relatively fast matching algorithm to gradually reduce the right-hand side of (15) to zero.

Matching algorithm. Let C, D be the two ordered sequences of (20). Let J and K be the pointers for the sequences C and D, respectively.

Step 1: (Initialization) $K \leftarrow 1, J \leftarrow 1$;
Step 2: If $(N-K)$ .EQ. $(M1-J)$, match $C(J)$ with $D(K)$ and go to Step 7;
Step 3: If $D(K)$ .LE. $C(J)$, match $C(J)$ with $D(K)$ and go to Step 7;
Step 4: If $D(K+1)$ .GE. $C(J)$, then (advance) $K \leftarrow K+1$ and go to Step 2;
Step 5: Compute $A1 = D(K) - C(J)$ and $A2 = C(J) - D(K+1)$.
Step 6: If $A1$ .LE. $A2$, then match $C(J)$ with $D(K)$; otherwise (advance) $K \leftarrow K+1$ and go to Step 2;
Step 7: (Advance) $K \leftarrow K+1, J \leftarrow J+1$;
Step 8: If J is greater than $M1$, then STOP; otherwise go to Step 2.

6. Numerical examples. The method described in this report has been applied to a number of problems. To demonstrate that the method works properly, numerical examples were selected where the exact solutions are known. Let $e_i^{(\gamma)} = |p_i^* - p_i^{(\gamma)}|$ be the difference between the true solution and the computed solution at the γth iteration. A termwise, a posteriori numerical estimate of the order of convergence is given by

$$\text{EOC}_i^{(\gamma)} = \log\,(e_i^{(\gamma)}/e_i^{(\gamma+1)}). \tag{22}$$

The computations were done in double precision on the IBM System 370 Model 195 computer at Argonne National Laboratory. In each iteration, the approximate eigensystem was computed by the *QL* method (EISPACK; see [15]). The resulting linear systems were solved using Gaussian elimination with partial pivoting (LINPACK; see [7]).

Example 1. Consider the inverse eigenvalue problem defined in (1) with symmetric matrices A_i whose lower triangular parts are given in Table 1.

TABLE 1

	0.39427				
	0.79289	0.47550			
A_0:	0.29431	0.39107	0.64510		
	1.00517	0.37370	0.96357	0.27282	
	0.98080	0.31333	0.33449	1.04439	0.40766
	0.870				
	0.327	0.593			
A_1:	0.145	0.931	0.743		
	0.437	0.201	0.808	0.986	
	0.451	0.037	0.300	0.882	0.926
	0.733				
	0.823	0.142			
A_2:	0.109	0.564	0.768		
	0.905	0.414	0.562	0.736	
	0.114	0.852	0.639	0.342	0.788
	0.212				
	0.634	0.891			
A_3:	0.421	0.456	0.720		
	0.542	0.577	0.812	0.345	
	0.882	0.123	0.420	0.786	0.743

This problem is of order five with all eigenvalues prescribed and three parameters to be determined. The prescribed eigenvalues are $\lambda_1^* = 4.0216090$, $\lambda_2^* = 0.61568326$, $\lambda_3^* = 0.42495309$, $\lambda_4^* = -0.65946669$, $\lambda_5^* = -1.0619386$. For this problem, a solution is $p_1^* = 0.1$, $p_2^* = 0.11$, $p_3^* = 0.12$.

With the initial guess

$$p_1^{(0)} = p_2^{(0)} = p_3^{(0)} = 0, \tag{23}$$

the computed eigenvalues agree with the prescribed eigenvalues to seven digits after three iterations. In Table 2, we list the errors $e_i^{(\gamma)}$ and the convergence rates $\mathrm{EOC}_i^{(\gamma)}$ of the parameters p_i.

TABLE 2
Errors in computed parameters p_i for $m1 = n = 5$, $m = 3$ with initial guess (23).

γ	$e_1^{(\gamma)}$	$\mathrm{EOC}_1^{(\gamma)}$	$e_2^{(\gamma)}$	$\mathrm{EOC}_2^{(\gamma)}$	$e_3^{(\gamma)}$	$\mathrm{EOC}_3^{(\gamma)}$
1	4.86D−3		3.53D−3		2.92D−3	
2	1.36D−5	2.6	1.70D−5	2.3	1.38D−6	3.3
3	3.23D−8	2.6	1.85D−8	2.9	1.47D−8	1.9

Next, with the same matrices we let $m1 = 4$, i.e., only $\lambda_1^*, \lambda_2^*, \lambda_3^*, \lambda_4^*$ are given. Using the initial guess (23), the rate of convergence is again quadratic. A similar result obtains for $m1 = 3$. We list the errors in the computed eigenvalues in Table 3. For $m1 = 5$ or 4, it takes four iterations to obtain the results; for $m1 = 3$, it takes five iterations.

TABLE 3
Errors in computed eigenvalues of Example 1 *for $m1 = 5, 4, 3$.*

$m1$	$\lvert\lambda_1^* - \lambda_1\rvert$	$\lvert\lambda_2^* - \lambda_2\rvert$	$\lvert\lambda_3^* - \lambda_3\rvert$	$\lvert\lambda_4^* - \lambda_4\rvert$	$\lvert\lambda_5^* - \lambda_5\rvert$
5	1.4D−9	6.5D−9	4.7D−9	3.0D−9	1.6D−8
4	9.1D−10	7.1D−9	8.5D−10	1.1D−9	1.7D−8
3	9.9D−11	1.9D−9	6.0D−10	6.5D−8	1.0D−8

Example 2. Consider the inverse eigenvalue problem [8] for $m1 = n = 6$, $m = 3$ with $A_0 = (a_{jk})$, $a_{jk} = j + k - 1$ for $j \neq k$, $a_{11} = a_{22} = a_{33} = 0$, $a_{44} = 37.841526$, $a_{55} = 66.072172$, $a_{66} = 91.024067$. A_i, $i = 1, 2, 3$, are zero matrices except for a 1 in the ith diagonal position. The prescribed eigenvalues are 100, 64, 36, 16, 4, 0. For this problem, a solution is $p_1^* = 1.198616$, $p_2^* = 5.810801$, $p_3^* = 18.052819$. Using the procedures described in this paper with initial guess $p_1 = 1$, $p_2 = 5$, $p_3 = 20$, it takes four iterations to get seven-digit accuracy in the computed solution. We have rerun the problem, reducing $m1$ successively to 5, 4 and 3. We list the errors in the computed eigenvalues in Table 4.

TABLE 4
Errors in computed eigenvalues of Example 2 *for $m1 = 6, 5, 4, 3$.*

$m1$	$\lvert\lambda_1^* - \lambda_1\rvert$	$\lvert\lambda_2^* - \lambda_2\rvert$	$\lvert\lambda_3^* - \lambda_3\rvert$	$\lvert\lambda_4^* - \lambda_4\rvert$	$\lvert\lambda_5^* - \lambda_5\rvert$	$\lvert\lambda_6^* - \lambda_6\rvert$
6	1.8D−7	6.7D−7	3.1D−7	2.0D−8	7.3D−9	3.9D−9
5	2.1D−7	5.7D−7	1.1D−7	2.0D−9	2.0D−9	5.2D−5
4	9.1D−8	5.5D−7	1.1D−7	8.6D−10	1.1D−4	1.3D−4
3	1.2D−11	5.4D−12	3.0D−11	3.8D−4	3.3D−3	4.0D−3

REFERENCES

[1] Z. BOHTE, *Numerical solution of the inverse algebraic eigenvalue problem*, Comput. J., 10 (1968), pp. 385–388.

[2] D. BOLEY AND G. H. GOLUB, *Inverse eigenvalue problems for band matrices*, SU 326 P30-55, Computer Science Dept., Stanford Univ., Stanford, CA, 1977.

[3] ———, *The matrix inverse eigenvalue problem for periodic Jacobi matrices*, SU 326 P30-64, Computer Science Dept., Stanford Univ., Stanford, CA, 1978.

[4] K. BROWN AND J. DENNIS, *Derivative free analogues of the Levenberg–Marquardt and Gauss algorithms for nonlinear least squares approximation*, Numer. Math., 18 (1972), pp. 289–297.

[5] C. DE BOOR AND G. H. GOLUB, *The numerically stable reconstruction of a Jacobi matrix from spectral data*, Linear Algebra Appl., 21 (1978), pp. 245–260.

[6] G. N. DE OLIVEIRA, *Note on an inverse characteristic value problem*, Numer. Math., 15 (1970), pp. 345–347.

[7] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[8] K. P. HADELER, *Newton-Verfahren für Inverse Eigenwertaufgaben*, Numer. Math., 12 (1968), pp. 35–39.

[9] O. H. HALD, *Inverse eigenvalue problem for Jacobi matrices*, Linear Algebra Appl., 14 (1976), pp. 63–85.

[10] A. HOFFMAN AND H. WIELANDT, *Recent Advances in Matrix Theory*, H. Schneider, ed., North American Press, Milwaukee, WI, 1964.

[11] L. V. KANTOROVICH AND G. P. AKILOV, *Functional Analysis in Normed Spaces*, Pergamon Press, New York, 1964.

[12] D. E. KNUTH, *Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[13] P. LANCASTER, *On eigenvalues of matrices dependent on a parameter*, Numer. Math., 6 (1964), pp. 377–387.

[14] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[15] P. MOREL, *Des algorithmes pour le problème inverse des valeurs propres*, Linear Algebra Appl., 13 (1976), pp. 251–273.

[16] B. NOBLE, *Applied Linear Algebra*, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[17] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, London, 1970.

[18] 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*, Springer-Verlag, New York, 1976.

[19] S. TOMAN AND J. PLIVA, *Multiplicity of solutions of the inverse secular problem*, J. Molecular Spectroscopy, 21 (1966), pp. 362–371.

[20] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0004 $01.25/0

SOME GENERAL BIFURCATION TECHNIQUES*

RALPH BAKER KEARFOTT†

Abstract. The problem $H(y) = H(x, \lambda) = \theta$, where $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is considered. Numerical techniques for locating bifurcation points $\bar{y}$ and for following arcs leading from $\bar{y}$ are presented. These techniques are valid for primary and secondary bifurcation points, and at multiple bifurcation points, regardless of whether there is a change in the sign of the determinant or the Jacobi matrix H_x at $\bar{y}$; they can also possibly be used when arcs intersect tangentially. The techniques do not require computation of second partial derivatives, although Jacobi matrices are computed using finite differences in neighborhoods of bifurcation points.

Details for incorporation into a derivative-free arc-following method, developed in a previous work, are given. Computational results for five test examples appear. Directions for further investigations and improvements are listed.

The stepsize control and Jacobi matrix update techniques may be improved for large, sparse problems, when first partial derivatives are easy to compute, or in the absence of bifurcation points.

Key words. arc-following method, bifurcation, numerical method, rank-1 updates, steplength algorithms

1. Introduction.

We desire to quantitatively describe the solution set of

$$H(y) = H(x, \lambda) = \theta, \tag{1}$$

where $H : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and $y = (x, \lambda) \in \mathbb{R}^n \times \mathbb{R}$.

Solution sets of (1) usually consist of intersecting sets of codimension-1 manifolds in $\mathbb{R}^{n+1}$. Under sufficient smoothness and regularity assumptions, the solution set within the region $\{(x, \lambda), \|x\| \leqq M,\ 0 \leqq \lambda \leqq 1\}$ consists of a finite number of smooth intersecting arcs. In that situation, the numerical problem can be resolved into the following components: (i) following the individual solution arcs of (1) accurately and efficiently; (ii) finding the bifurcation points (points where the arcs intersect); and (iii) successfully following one or more arcs away from a bifurcation point.

Much work on these problems has been done by H. B. Keller, W. C. Rheinboldt, and others. For a survey of arc-following techniques, see e.g. [2]. For a survey of techniques for bifurcation problems, see e.g. [16]. The reader may also consult [17], where application of simplicial methods to these problems is treated. Each of these surveys contains lists of further references.

Despite the richness of techniques and methods, further research is desirable. Improvements and further evaluation of both arc-following and bifurcation techniques are possible. Also, bifurcation techniques are of an ad hoc nature; primary, secondary and multiple bifurcation points are treated separately (cf. [16]); little work has been done in detecting "even order" bifurcation points (where an odd number of arcs intersect); and continuation away from a bifurcation point may fail when the arcs intersect tangentially. Other undesirable characteristics include the need for higher order derivatives or double iteration processes.

The purpose of this paper is to present some new and more general techniques for handling bifurcation problems. We implement the techniques with a derivative-free predictor-corrector-type arc-following method, which we review in § 2. We base

* Received by the editors February 4, 1981 and in revised form March 29, 1982.

† Department of Mathematics and Statistics, The University of Southwestern Louisiana, Lafayette, Louisiana 70504.

detection of each bifurcation point $\bar{y}$ on analysis of the singular values of an approximate Jacobi matrix $H'(\bar{y})$; details are given in § 3.

We obtain directions for arcs emanating from $\bar{y}$ by locating minima of $\|H(y)\|_2$ for y on the boundary of a small ball in the affine space containing $\bar{y}$ and defined by the null space of an approximate $H'(\bar{y})$. This technique is most similar to that of H. B. Keller [14] in that direction vectors at the bifurcation point are given explicitly; it differs in the method of obtaining those directions. Also, Keller's directions are tangent to the arcs at $\bar{y}$, whereas our directions are approximately secant to the projections of the arcs onto the above affine space. Our approach and its implementation are detailed in § 4 below.

Results and analyses of computer runs are given in § 5. A summary, conclusions and directions for improvements are given in § 6.

2. The arc-following method. Since the basic arc-following method is explained in [11], only a general outline will be given here.

We assume that solution arcs of (1) can be parametrized with respect to arclength s, and we suppose $y_0 \in \mathbb{R}^n \times \mathbb{R}$ is such that $H(y_0) = \theta$. Then:

$$H'(y(s))y'(s) = \theta, \qquad \|y'(s)\| = 1, \quad y(0) = y^{(0)} \tag{2}$$

may be integrated to find an arc $y(s)$ with $H(y(s)) = \theta$, where H' is the n by $n+1$ Jacobi matrix of H, and $y'(s)$ is the componentwise derivative of $y \in \mathbb{R}^{n+1}$ relative to s.

Our method is based on "predictor-corrector" techniques (cf. e.g. [1], [6], [15]). The predictor step corresponds to an implicit Euler step for (2) and is defined by:

$$z_0^{(k+1)} = y^{(k)} + \delta_k b^{(k)} \tag{3}$$

where $\delta_k \in \mathbb{R}$ is suitably small, $H(y^{(k)}) \doteq \theta$, and $b^{(k)}$, $\|b^{(k)}\| = 1$ is approximately tangent to $y(s)$ at $y^{(k)}$. The next iterate $y^{(k+1)}$ is then obtained by solving:

$$G(z) = \begin{pmatrix} H(z) \\ b^{(k)t}(z - z_0^{(k+1)}) \end{pmatrix} = \theta \tag{4}$$

with a generalized secant method.

In [11] a derivative-free predictor-corrector method for (2) is given. There, H' is computed by updating with a least-change secant (Broyden) technique. Such techniques allow computation of a new approximate derivative matrix H' each time a new value of H is obtained, without any additional function or derivative evaluations. To assure that H' is approximately correct, Powell's idea of special correction updates is used ([19] and [11]); the effect is to supply a matrix H' which reflects, as a linear transformation, the action of the true H' in a region containing the past $2n+3$ points $z_j^{(l)}$ and $y^{(l)}$. (Powell corrections require, on the average, less than one additional function evaluation per step).

Stepsize algorithms for adaptively choosing δ_k are also given in [11], in addition to publications of several other researchers. In the absence of bifurcation points, such stepsize controls are usually based on keeping the number of corrector iterations required to solve (4) within bounds, or upon keeping the cosine of the angle between successive approximate tangent directions $b^{(k)}$ and $b^{(k+1)}$ within bounds; to do this, δ_{k+1} is set to δ_k, $2\delta_k$, or $\frac{1}{2}\delta_k$ as appropriate.

Numerical tests in [11] indicate that the predictor-corrector technique, combined with Broyden updates of H', is competitive with general algorithms such as the Chow–Yorke algorithm in [25], provided one assumes evaluation of a Jacobi matrix from scratch would require n function evaluations. This fact has also been verified

by Kurt Georg [8], who independently investigated a similar predictor-corrector scheme with Broyden updates.

Our basic idea for handling bifurcation points, in addition to a framework for its implementation, are also given in [11]. Below, we expand, in refined form, these ideas; we then present the results of numerical tests and outline possible improvements.

3. Detection of the bifurcation points. If $\bar{y}$ is a bifurcation point of (1), then the dimension of the null space $N(\bar{y})$ of $H'(\bar{y})$ must be greater than 1 (cf. [20]). In particular, the determinant $\det(H_x(x, \lambda))$ must vanish at the point $\bar{y} = (x^*, \lambda_*)$ with corresponding arclength s_*. Let $\{\tau_i(s)\}_{i=1}^{i=n+1}$ be the eigenvalues of the Jacobi matrix $G'(y(s))$, where G is as in (4). Then two or more $\tau_i(s)$ must vanish at $s = s^*$.

If an odd number of $\tau_i(s)$ cross the axis $\tau = 0$ at $s = s^*$, then $\det(G'(y(s))$ must change sign at $s = s^*$. Some techniques for locating bifurcation points are based on detection of such sign changes (see the references in the surveys) or on detection of changes in a topological degree (cf. e.g. [10]). Here, we deal with the more difficult problem of detecting all bifurcation points.

In general, any quantity $q(s)$ which is a continuous function of arclength s and which equals zero at values of s for which $H'(y(s))$ is rank deficient may be used to detect bifurcation points. In the initial experiments presented here, we elect to examine the reciprocal of the condition number of H' relative to the 2-norm, expressed as the ratio of the smallest nonzero singular value $\sigma_n(s)$ to the largest nonzero singular value $\sigma_1(s)$ of $H'(y(s))$. To obtain this quantity, we actually did a singular value decomposition of the approximate $H'(y^{(k)})$; though not optimal from the point of view of overhead, this stable technique gave us valuable additional information concerning the behavior of our algorithms and test functions.

Bifurcation points are found by detecting minima of $\sigma_n(s)/\sigma_1(s)$ (or, more generally, minima of $|q(s)|$). Besides having the properties listed in § 2 of this paper, our predictor stepsize control should decrease δ_k before encountering a bifurcation point, in such a manner that the position of each bifurcation point may be efficiently bracketed and refined. If exact derivative matrices $H'(y^{(k)})$ are assumed, linear convergence to a bifurcation point can be achieved by allowing $|q(s_{k+1})| - |q(s_k)| > -\alpha q(s_k)$, for some $0 < \alpha < 1$, until $|q(s_k)|$ reaches some threshold value, after which the step δ_k is unchanged until a minimum of $|q(s)|$ is bracketed.

As was mentioned in § 2, the matrices H'_k, and hence the $q(s_k)$, formed from Broyden updates with Powell correction steps are not exact Jacobi matrices at $y^{(k)}$, but depend upon the action of the exact linear transformations $H'(y)$ in a region containing the past $2n+3$ corrector iterates and points $y^{(l)}$. In our experiments, we handled this problem by setting δ_{k+1} to be a small fraction of δ_k whenever the average decrease in $|q(s)|$ over the past $2n+3$ points $y^{(l)}$ exceeded $\alpha|q(s_k)|$; δ_{k+1} was not allowed to be greater than δ_k except after a cycle of $2n+3$ predictor steps over which δ_k was not decreased due to $q(s)$.

The above step control in the neighborhood of bifurcation points is inefficient when n is large, since $2n+3$ times as many points $y^{(k)}$ are computed as in the case where "exact" H' are available. A possible improvement, in later computations, would be to discard the Powell update procedure, use Broyden (or other quasi-Newton) updates only for the corrector steps, and compute H' using finite differences at each $y^{(k)}$. The H' could then be considered exact, with the step control threshold set according to the accuracy of the differencing scheme. (Of course, exact H' would be even better, if they are easy to obtain).

If we assume $H'_k = H'(y^{(k)})$ is an exact Jacobi matrix, minima of $q(s)$ are bracketed whenever $|q(s_{k-1})| < |q(s_{k-2})|$ and $|q(s_{k-1})| < |q(s_k)|$. If H'_k is the result of applying

quasi-Newton updates and Powell correction steps, the computed $q(s_k)$ will vary irregularly near a minimum of the actual $q(s)$. In our experiments, we handled this by comparing average values of $|q(s_k)|$ over the first third, middle third, and last third of the past $2n+5$ iterates. As in stepsize control, this method is cumbersome and undesirable for large n, but worked reliably in our experiments.

Once the minimum has been bracketed, it may be refined using successive quadratic or linear interpolation, etc. Standard techniques may be used, but linear combinations of previous approximate singular points $\bar{y}$ are taken to produce new $\bar{y} = y(\bar{s})$; in this process, each new $\bar{y}$ is corrected via formula (4) in order to lie on an arc $H(y(s)) = \theta$, until the condition of H' renders this inadvisable. In our experiments, relatively exact H' obtained from finite differences were used during such a "line search" (over arclength).

In our experiments, the smallest singular value $\sigma_n(s)$ typically had an absolute-value-type singularity in its derivative at points $\bar{s}$ where $\sigma_n(\bar{s}) = 0$. Since we implemented a hybrid quadratic interpolation, linear interpolation line search, refinement of the point $\bar{y}$ typically consisted of successive linear interpolation.

Occasionally, minima of $|q(s)|$ not corresponding to $q(s) = 0$ are computed. This must occur since we do not assume $q(s)$ changes sign at singular points $\bar{y} = y(\bar{s})$. These may be eliminated in the stepsize control (setting of δ_k), in the line search, or after each $\bar{y}$ is located. In our experiments, we chose the conservative method of analyzing the null space of each $H'(\bar{y})$ once $\bar{y}$ has been refined.

It is desirable to compute the singular points $\bar{y}$ accurately. This is because minima of $\|H\|$ over a low-dimensional manifold are computed to obtain tangent directions leading from $\bar{y}$, and distinct minima are relatively close together when $\bar{y}$ is given inaccurately. (See the discussion in the next section and Fig. 1.) Thus, the tolerance in the line search is set according to the accuracy to which $q(s)$ is being computed.

4. Determining arc directions at a bifurcation point. The ansatz and basic structure of the algorithm employed appear in the introduction and in [11, § 4]. Here, we give a summary, more details, and improvements.

We assume that the component arcs $y(s) \subset H^{-1}(\theta)$ are smooth functions of arclength s; then any such arcs passing through a bifurcation point $\bar{y}$ must have tangent vectors at $y = \bar{y}$ in the null space of $H'(\bar{y})$. Suppose $\{v^{(1)}, \cdots, v^{(k)}\}$ is an orthonormal basis for this null space, and let Π be the affine space given by $\Pi = \{\bar{y} + \sum_{j=1}^{k} \alpha_j v^{(j)} | \alpha_j \in \mathbb{R}\}$. Let $\mathscr{A} \subset \Pi$ be a small region containing $\bar{y}$ in its interior, and let $\{m^{(1)}, m^{(2)}, \cdots, m^{(q)}\}$ be the locations of the minima of $\|H\| | \partial\mathscr{R}$. Then direction vectors for arcs intersecting at $\bar{y}$ are given by a subset of $\{(m^{(i)} - \bar{y})/\|m^{(i)} - \bar{y}\| \,|\, 1 \leqq i \leqq q\}$.

To proceed, three tasks must be undertaken: (i) determination of an orthonormal basis for the null space of $H'(\bar{y})$; (ii) choice of a size and shape for the region $\mathscr{R}$; and (iii) choice of and execution of a method for finding the minima $m^{(i)}$ of $\|H\| | \partial\mathscr{R}$.

In the experiments reported here, we form a basis for the null space of $H'(\bar{y})$ from the right singular vectors of $H'(\bar{y})$ corresponding to singular values equal to zero. Less general techniques and exploitation of special structure in H' can improve the efficiency of this computation.

The region $\mathscr{R}$ was chosen here to be a k-ball S_δ of radius δ centered at $\bar{y}$, so that $\partial\mathscr{R}$ is without boundary and an unconstrained minimizer can be used. We parametrize ∂S_δ in terms of the standard spherical angular coordinates: $\{(\phi_1, \cdots, \phi_{k-2}, \psi) | -\pi/2 \leqq \phi_i \leqq \pi/2, \ 1 \leqq i \leqq k-2 \text{ and } -\pi \leqq \psi \leqq \pi\}$. The region $[-\pi/2, \pi/2]^{k-2} \times [-\pi, \pi]$ was divided into $(2p)^{k-1}$ subregions, defined by dividing

each of the spherical coordinate intervals into $2p$ subintervals. The unconstrained minimizer was then applied $(2p)^{k-1}$ times, with starts in each subregion.

Choice of the radius δ requires some thought for numerically difficult problems. If $y^{(j)} = \bar{y}$ is a bifurcation point, then approximations $z^{(j+1)}$ to the next iterate $y^{(j+1)}$ will be given by:

$$z^{(j+1)} = m^{(i)} = y^{(j)} + (m^{(i)} - y^{(j)})$$

for some i, $1 \leqq i \leqq q$. Thus, it is important that $H'(m^{(i)})$ be of rank n to within the computational precision of H'. Define:

$$\mathcal{M}_j = \{y \in \mathbb{R}^n \times [0, 1] \mid \sigma_j(y) = 0\}$$

for $1 \leqq j \leqq n$, where $\sigma_j(y)$ is the jth singular value of $H'(y)$. Then, under appropriate regularity conditions, $\mathcal{M}_j$ is a finite collection of codimension-1 manifolds in $\mathbb{R}^n \times [0, 1]$, and $\bar{y} \in \mathcal{M}_j$, $n - k + 1 \leqq j \leqq n$. Also, if there are q' arcs in $H^{-1}(\theta)$ intersecting at $\bar{y}$, define $\mathcal{A}_{j'}$, $1 \leqq j' \leqq q'$, to be the projection of the jth such arc onto the affine space Π corresponding to the null space of $H'(\bar{y})$. Then $H'(m^{(i)})$ can be made nonsingular by increasing δ provided the $\mathcal{A}_{j'}$ and the $\mathcal{M}_j$ intersect sufficiently transversally at $\bar{y}$.

In our experiments, the following test was used to decide nonsingularity of H'. Suppose the machine can represent m decimal digits, and suppose H is computed by differences with relative stepsize equal to 10^{-s}, so that H is accurate to approximately $m - s$ digits. Then, if H' is computed with forward differences with optimal stepsize, H' will be accurate to approximately $(m - s)/2$ decimal digits. Let β be the number of decimal digits accuracy in σ_n, where σ_j is the jth singular value of H', $1 \leqq j \leqq n$. then, since $\|H'\|_2 = \sigma_1$, we have

$$\beta \doteq (m - s)/2 - \log_{10}(\sigma_1/\sigma_n) = (m - s)/2 - L - l,$$

where $L = \log_{10} \sigma_1$ and $l = -\log_{10} \sigma_n$ (cf. e.g. [22, p. 321]). Since H' is singular when $\sigma_n = 0$, H' will be singular to within the computational precision whenever $l \geqq \beta$. Thus, H' is considered singular whenever:

$$l \geqq (m - s)/4 - L/2. \tag{5}$$

As an additional consideration in choosing δ large, all arcs emanating from $\bar{y}$ can be detected only when $\bar{y}$ is in the interior of S_δ; this will not be the case if $\bar{y}$ is given only approximately and δ is too small; numerically, $\bar{y}$ relatively near ∂S_δ will make distinct arcs more difficult to detect (see Fig. 1).

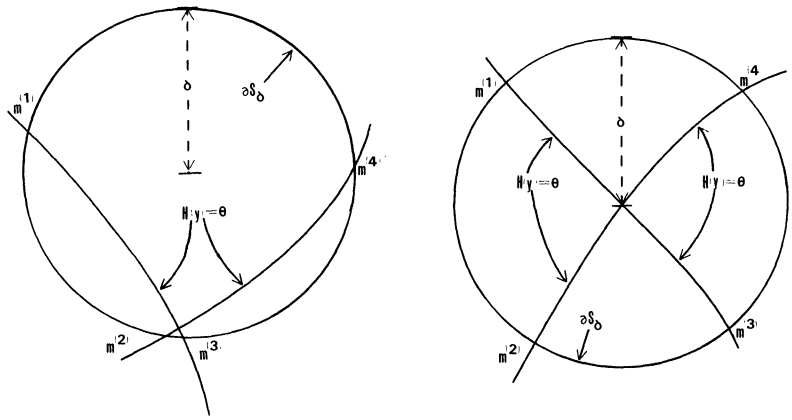


FIG. 1. *Minima of $\|H\|$ $|\partial S_\delta$ are easier to distinguish when the center of S_δ is near the bifurcation point.*

On the other hand, if δ is chosen too large, the arcs intersecting at $\bar{y}$ do not lie approximately in Π. This can cause the surface of $\|H\|$ to be "flatter" near the $m^{(i)}$ and can also cause corrector iterations starting with $m^{(i)}$ to fail. In view of this, the smallest δ consistent with easy detection of the $m^{(i)}$ and with $H'(m^{(i)})$ being nonsingular should be used.

In our experiments, we initially set δ equal to the maximum of the minimum allowable predictor stepsize and $\sqrt{n+1}$ times ε_f, where ε_f is the relative accuracy to which $\bar{y}$ has been located. As the $m^{(i)}$ were computed, inequality (5) was checked; if (5) was false for any i, δ was replaced by the minimum of $M\delta$ and the maximum allowable predictor stepsize. After increasing δ, the minimization process was restarted with $i = 1$. We mention that (5) and this process are very conservative, and improvements in efficiency are possible.

We have found the simplex method of Nelder and Mead [18] suitable for direct location of the $m^{(i)}$. A descent method similar to the method of steepest descent, the simplex method of Nelder and Mead begins with a starting simplex; this simplex is then changed one vertex at a time by "reflection", "expansion", and "contraction". The iteration ceases when the diameter of the resulting simplex has become smaller than some tolerance ε_d.

The values of the minimization search accuracy ε_d which are reasonable to demand depend upon which δ is selected, and both ε_d and δ depend on the machine accuracy ε_m and the order of accuracy in the values of H. It can be shown that a tolerance of ε_d in the $(k-1)$-dimensional parameter space corresponds roughly to a distance of $\sqrt{k-1}\pi\delta\varepsilon_d$ in $\mathbb{R}^{n+1}$. Since the relative accuracy of coordinates of $m^{(i)} \in \mathbb{R}^{n+1}$ cannot exceed the machine epsilon ε_m, it is prudent to have:

$$\varepsilon_d > \frac{\|y\|\varepsilon_m}{\sqrt{k-1}\pi\delta}. \tag{6}$$

In practice, ε_d is chosen somewhat larger than the right-hand side of (6), since the computational precision is somewhat larger than ε_m and also since the minimization is less costly with larger stopping tolerances. In particular, ε_m in (6) may be replaced by a fraction of the arc-following tolerance. In the tests, ε_d was automatically doubled every time a location $m^{(i)}$ was found such that the scaled magnitude of $\|H(m^{(i)})\|$ was less than .1 times the corresponding tolerance in the arc-following method, and was decreased proportionally whenever δ was increased.

5. Numerical experiments. Here, we present results from several test examples; our goal is to demonstrate feasibility of the techniques as alternative approaches, to give an idea of their versatility, to describe the algorithms' behavior and to point out possible difficulties.

Five examples were treated. All involve special H of the form

$$H(x, \lambda) = \lambda g(x) + (1-\lambda)g^{(0)}(x), \tag{7}$$

where $g: \mathbb{R}^n \to \mathbb{R}^n$ and $g^{(0)}: \mathbb{R}^n \to \mathbb{R}^n$. The continuation method was begun at $(\theta, 0)$, and continuation along each individual arc was halted when either the $\lambda = 1$ hyperplane or $\lambda = 0$ hyperplane was reencountered. The goal of the algorithms was to detect all bifurcation points and follow all arcs.

In the first four examples,

$$g(x) = Ax - f(x), \tag{8}$$

where A is the matrix corresponding to discretization of the boundary value problem

$$-u'' = 0; \; u(0) = u(1) = 0, \tag{9}$$

with central differences and n internal meshpoints, and the ith component of f is given by

$$f_i(x) = x_i^3.$$

Examples 1, 2 and 3 correspond to $g(x)$ as in (8), with $g^{(0)}$ defined by: $g_i^{(0)}(x) = -x_i$, and with $n = 2, 4$ and 7, respectively. In Example 4, we took $n = 3$ and we took g as in (8), but with $g^{(0)}(x) = -Ax$. In the final example, we took $n = 2$, we took $g^{(0)}$ to be the identity function, and we took the components of g to be

$$g_1(x) = 2x_1(x_1^2 + x_2^2) - .5x_1, \qquad g_2(x) = x_2(x_1^2 + x_2^2) - .5x_2.$$

(A variant of this appears in [23] in an example of a multiple bifurcation point.)

For the first three examples, it has been shown in [10] that there are exactly n primary bifurcation points, occurring at

$$\lambda_i = 1 - m_i/(1 + m_i), \qquad i = 1, 2, \cdots, n. \tag{10}$$

Here, m_i is the ith eigenvalue of the matrix A and is given by

$$m_i = 2(n+1)^2[1 + \cos(i\pi/(n+1))] \tag{11}$$

(cf. [9]). In these examples, the primary bifurcation points are all simple, and the arcs intersect transversally. Furthermore, it can be shown that $\sigma_n(s)$ is exactly linear immediately to the left and immediately to the right of each primary bifurcation point, where s is arclength. Thus, the successive linear interpolation scheme (cf. § 3) for refinement of $\bar{y}$ will give, in theory, exact bifurcation points $\bar{y}$, without necessity of line searches. However, the primary bifurcation points are spaced unevenly and are very close together for n large (cf. formulas (10) and (11)). Thus, the stepsize control scheme near bifurcation points is tested.

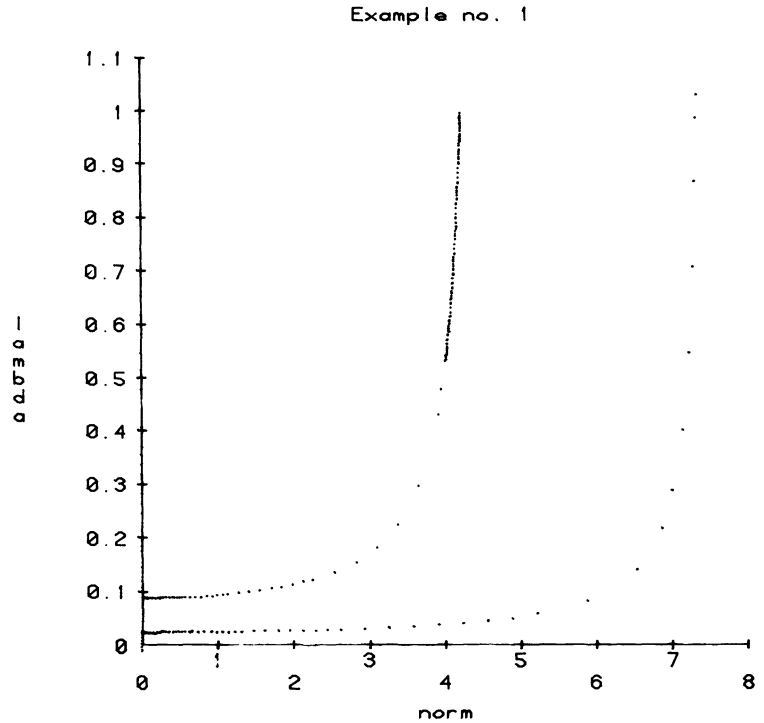


FIG. 2. *Example* 1.

In the fourth example, a single multiple bifurcation point occurs at $\lambda = \frac{1}{2}$ (cf. [10]), where H' equals the zero matrix, and four arcs intersect. Thus, it poses a test for the direction-finding algorithm in § 4.

In the fifth example, the unique primary bifurcation point for $\lambda \in [0, 1]$ occurs at $\lambda = \frac{2}{3}$. There, H' is the zero matrix and three arcs intersect.

The actual computations for Example 1 were straightforward, with no encountered difficulties. The norm and first coordinate of actual iterates are plotted with respect to λ in Figs. 2 and 3, respectively. Note that the step size on the arcs intersecting at λ_2 becomes small; this is because $\sigma_n(s)$ is decreasing and H' is singular on these arcs at $\lambda = 1$.

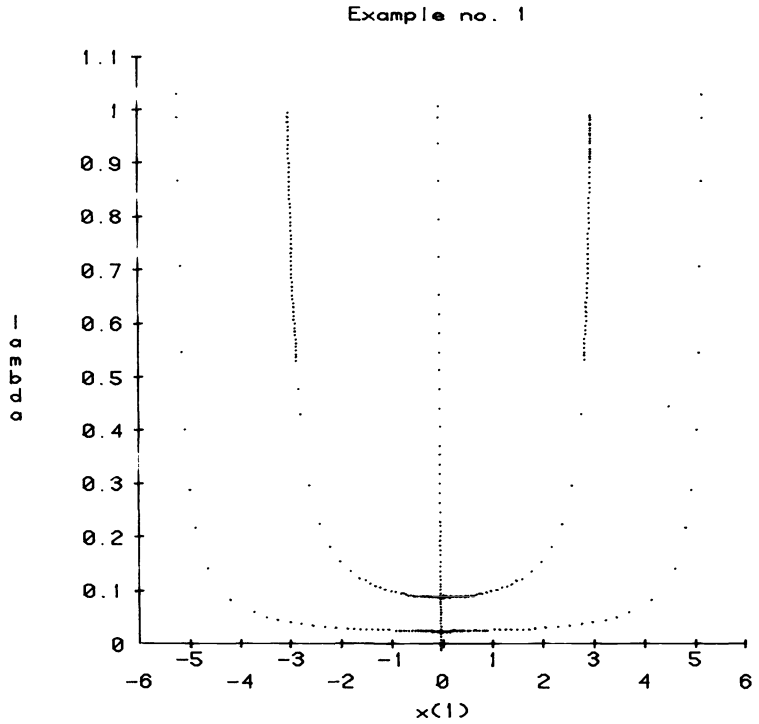


FIG. 3. *Example* 1.

The algorithms also performed satisfactorily for Example 2. However, secondary bifurcations occurred on the arcs intersecting at λ_2. Also, minima of $\sigma_n(s)$ not corresponding to bifurcation points were detected on the arcs intersecting at λ_3. See Figs. 4, 5 and 6 for graphs of the iterates.

Example 3 was straightforward, but the algorithms performed a sizable amount of computation. Numerous minima of σ_n not corresponding to bifurcation points were found.

The fourth example was by far the most difficult. The matrix H' was nearly singular in a large region, so proper automatic selection of the radius δ for S_δ was important. The hypersurface $\|H\|(y)$ seemed, in addition, to have numerous "wrinkles" near the bifurcation point; though actual tangent directions were computed accurately, the minimization gave numerous $m^{(i)}$ not corresponding to arcs $\|H\| = \theta$. These "imposters" were promptly detected when corrector iteration in the arc-following method failed, but a fine grid (i.e. p large and a large number of starts in the

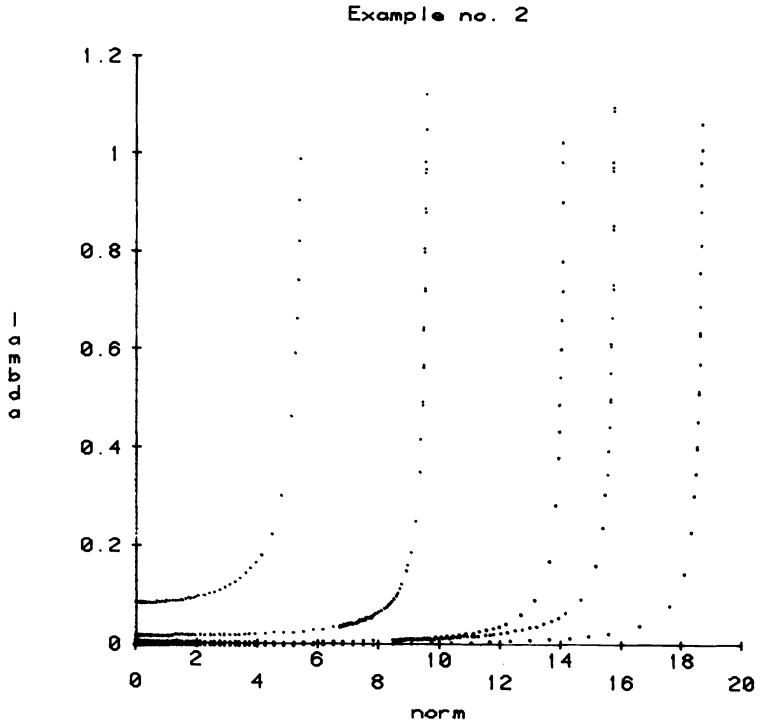


FIG. 4. *Example* 2.

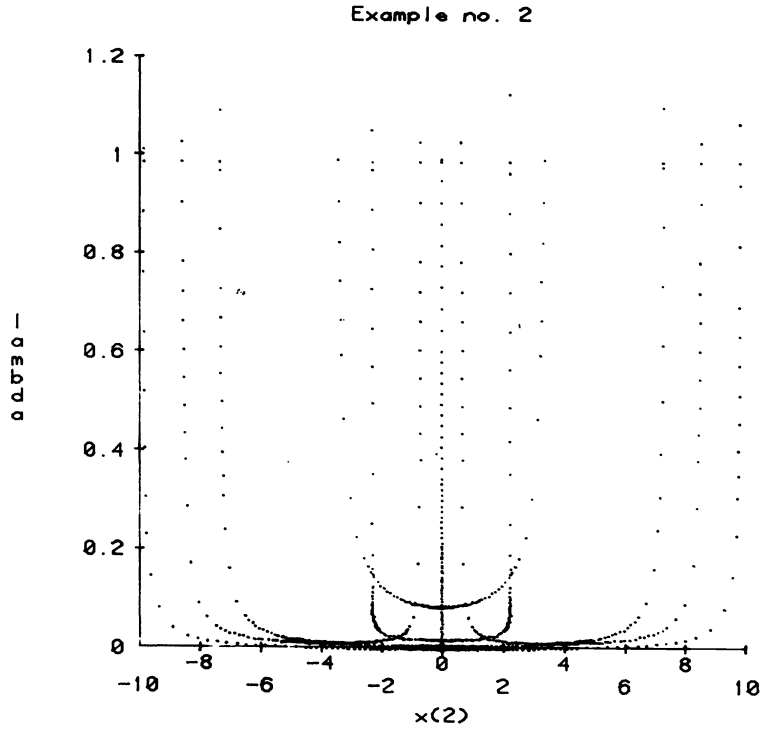


FIG. 5. *Example* 2.

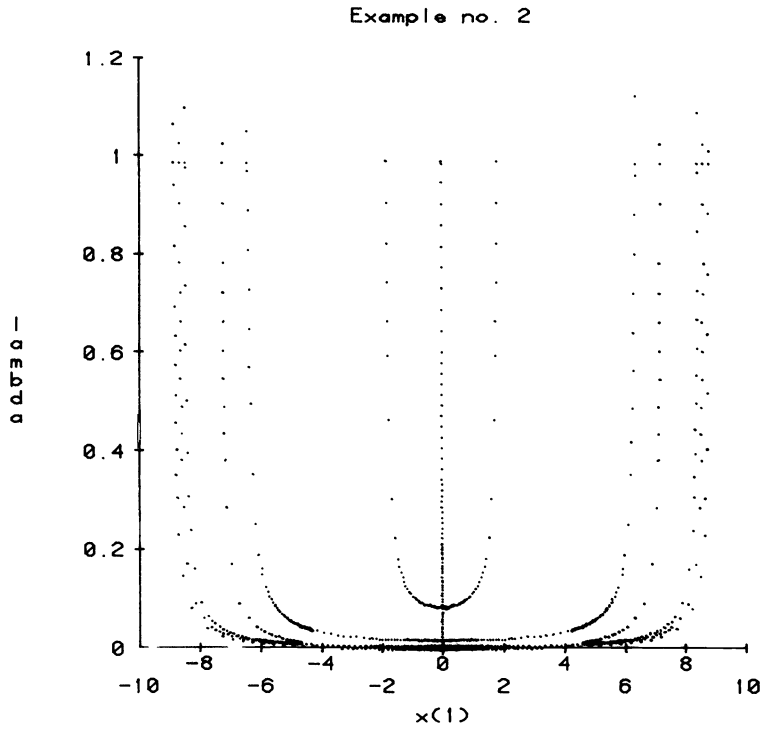


FIG. 6. *Example* 2.

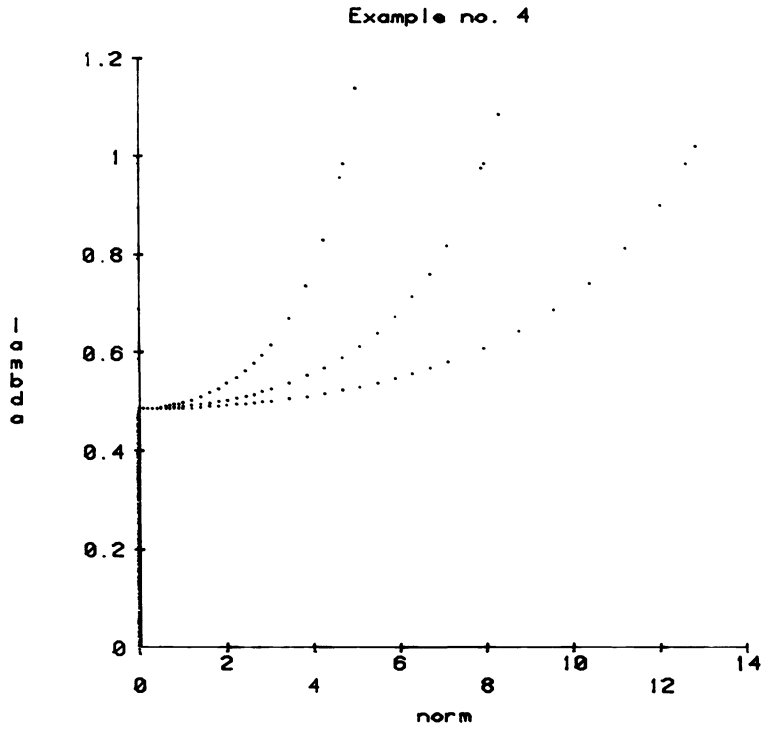


FIG. 7. *Example* 4.

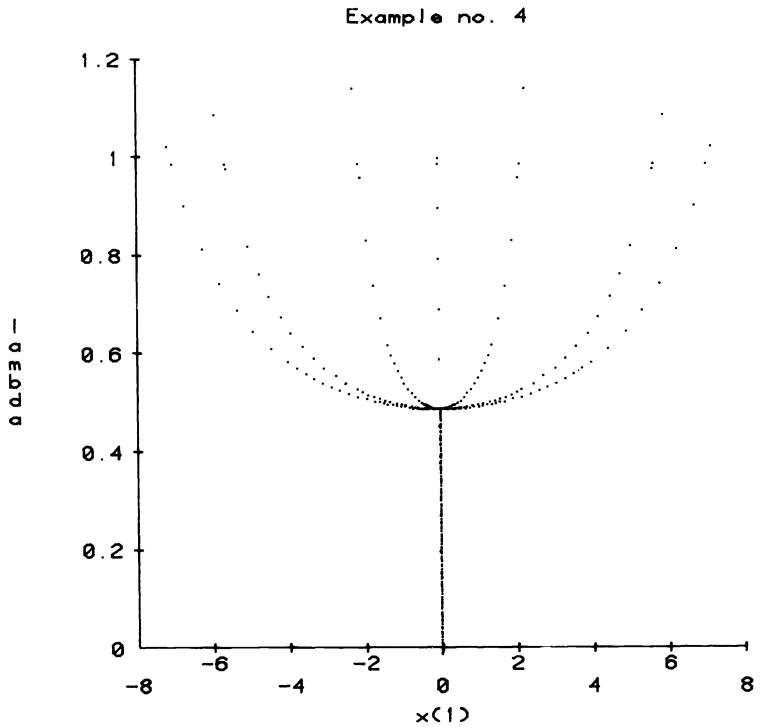


FIG. 8. *Example* 4.

minimization process) was required. With $p = 3$ (216 minimizations in 3-space), all 7 arcs proceeding away from the bifurcation point were found, but 14 false directions were also given. Plots are given in Figs. 7, 8 and 9.

The fifth example was perhaps the most straightforward. Here, there is no change in sign of the determinant of H_x at the bifurcation point on the trivial branch, since both eigenvalues pass from positive to negative; thus, the special capabilities of the stepsize control and bifurcation point refinement algorithms mentioned in § 3 are demonstrated (Figs. 10, 11).

As was indicated, the examples reveal behavior of the following facets of the algorithms: (i) the efficiency and reliability of deceleration and the "line search" when the arc-following method approaches and refines the bifurcation point; (ii) the efficiency and reliability with which directions for arcs intersecting at bifurcation points are computed, (iii) efficiency and reliability of the acceleration/deceleration scheme between and past bifurcation points. Performance data with regard to these qualities is given for Examples 1 through 5 in Tables 1 through 5, respectively.

The first part of each table deals with quality (iii): In the first column, the coordinates of the intersection of the arc in question with the $\lambda = 1$ hyperplane are given, while the coordinates of the last bifurcation point on this arc are given in the second column (BP). The total number of function evaluations required to follow the arc between these two points is given in the third column (FE). We note that this phase of the algorithms is extremely reliable; the discussion in [11] is for the most part valid, although the steps are increased somewhat more slowly here to assure that bifurcation points are not missed.

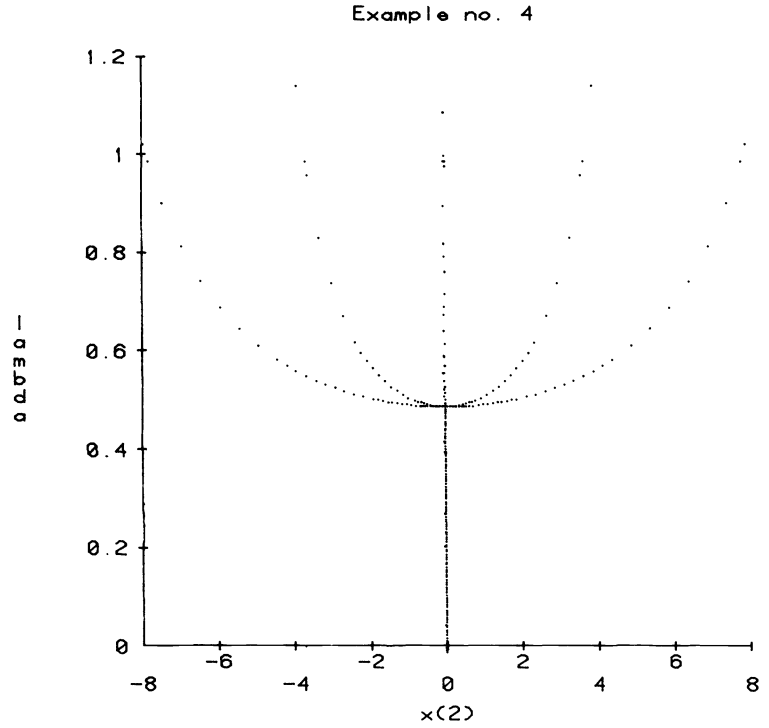


FIG. 9. *Example* 4.

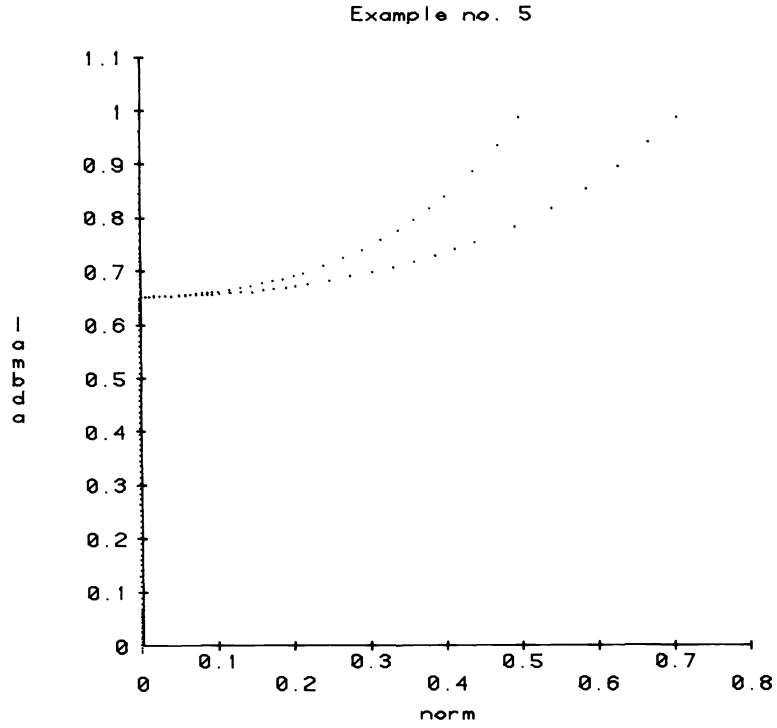


FIG. 10. *Example* 5.

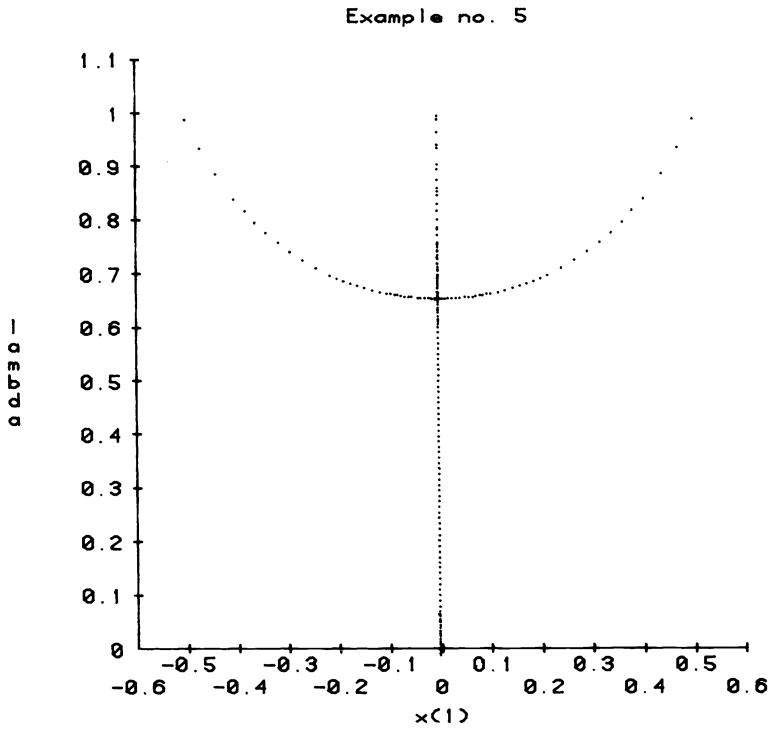


FIG. 11. *Example* 5.

The second part of each table deals with qualities (i) and (ii). The first and second columns (labeled BP and PBP, respectively) list the coordinates of the located bifurcation point and the coordinates of the previous bifurcation point (or origin), respectively. The third and fourth columns (FEP and PITL) give the total number of function evaluations required to obtain the bifurcation point from the previous one and the corresponding number of predictor steps, respectively. The final column (FEB) gives the number of function evaluations required by the minimization process to locate the $m^{(i)}$. Finally, the total number of function evaluations for the entire example, the total number of arcs intersecting at $\lambda = 1$ and the average number of function evaluations per arc are listed.

In all cases, the arc-following tolerance (ε_y in [11, Algorithm 2.1]) was set to 10^{-8}, and $\delta_{\max}$ was set equal to 1. In Examples 1, 2 and 3, $p = 6$ (cf. § 4) was adequate; $p = 3$ was adequate in Examples 4 and 5. The initial predictor step size (δ_0 in [11, Algorithm 2.1]) at $(\theta, 0)$ was in all cases set to the minimum $\delta_{\min}$. In Examples 1, 2, 4 and 5, $\delta_{\min}$ was set to 10^{-5}; in example 6, $\delta_{\min}$ was set to 10^{-6}. (The minimum stepsize $\delta_{\min}$ was judged small enough if it was less than the minimum distance between bifurcation points divided by $10(2n+3)$.) The tolerance ε_d in the simplex method of Nelder and Mead was set to $\max(\|\bar{y}\|, 1)\varepsilon_y/(10\delta)$ where δ is the radius of the ball S_δ. All other tolerances were set as in [11].

The Fortran programs were run on a Honeywell Multics 68/80 system. The Fortran programs are in experimental form, and there were some unnecessary redundant function evaluations, included for convenience in I/O, etc. Thus, the values in Tables 1 through 5 should be considered somewhat large.

TABLE 1(a)

#	root	BP	FE
1	(0, 0, 1)	(0, 0, .1)	158
2	(3, 3, 1)	,,	556*
3	(−3, −3, 1)	,,	556*
4	(−5.2, 5.2, 1)	(0, 0, .0357)	385
5	(5.2, −5.2, 1)	,,	377

* H' was singular at $\lambda = 1$ on these arcs.

TABLE 1(b)

#	BP	PBP	FEP	PITL	FEB
1	(0, 0, .0357)	(0, 0, 0)	340	219	675
2	(0, 0, .1)	(0, 0, .0357)	315	180	180

Total number of function evaluations, 4033. Total number of predictor steps, 1144. Number of function evaluations per arc, 806.

TABLE 2(a)

#	root	BP	FE
1	(0, 0, 0, 0, 1)	(0, 0, 0, 0, .0948)	223
2	(1.83, 3.41, 3.41, 1.83, 1)	,,	581
3	(−1.83, −3.41, −3.41, −1.83, 1)	,,	588
4	(−6.41, −2.3, 2.3, 6.41, 1)	(0, 0, 0, 0, .0281)	1706*
5	(6.41, 2.3, −2.3, −6.41, 1)	,,	1706*
6	(−8.47, 7.34, 7.34, −8.47, 1)	(−4.83, 3.62, 3.62, −4.83, .0216)	381
7	(−8.65, 8.56, .67, −7.23, 1)	,,	475
8	(−7.23, .67, 8.56, −8.65, 1)	,,	503
9	(8.47, −7.34, −7.34, 8.47, 1)	(4.83, −3.62, −3.62, 4.83, .0216)	408
10	(7.23, −.67, −8.56, 8.65, 1)	,,	490
11	(8.65, −8.56, −.67, 7.23, 1)	,,	471
12	(−8.83, 9.87, −9.87, 8.83, 1)	(0, 0, 0, 0, .0109)	745
13	(8.83, −9.87, 9.87, −8.83, 1)	,,	738

* Local minima of σ_n not corresponding to a singular H' were located on these arcs.

TABLE 2(b)

#	BP	PBP	FEP	PITL	FEB
1	(0, 0, 0, 0, .0109)	(0, 0, 0, 0, 0)	500	303	724
2	(0, 0, 0, 0, .0150)	(0, 0, 0, 0, .0109)	265	112	701
3	(0, 0, 0, 0, .0281)	(0, 0, 0, 0, .0150)	373	253	643
4	(0, 0, 0, 0, .0948)	(0, 0, 0, 0, .0281)	475	262	691
5	(−4.83, 3.61, 3.61, −4.83, .0216)	(0, 0, 0, 0, .0150)	1168	198	1186*
6	(4.83, −3.61, −3.61, 4.83, .0216)	,,	1165	198	1192*

Total number of function evaluations, 18,188. Total number of predictor iterations, 3341. Number of function evaluations per arc, 1399.

* These are secondary bifurcation points.

TABLE 3(a)

#	root	BP	FE
1	(0, 0, 0, 0, 0, 0, 0, 1)	(0, 0, 0, 0, 0, 0, 0, .0931)	302
2	(−1.19, −2.34, −3.3, −3.7, −3.3, −2.34, −1.19, 1)	,,	903
3	(1.19, 2.34, 3.3, 3.7, 3.3, 2.34, 1.19, 1)	,,	830
4	(−4.26, −7.31, −4.26, 0, 4.26, 7.31, 4.26, 1)	(0, 0, 0, 0, 0, 0, 0, .0260)	1037
5	(4.26, 7.31, 4.26, 0, −4.26, −7.31, −4.26, 1)	,,	1043
6	(−10.1, −4.2, 2.82, 9.49, 2.82, −4.2, −10.1, 1)	(0, 0, 0, 0, 0, 0, 0, .0125)	2989*
7	(10.1, 4.2, −2.82, −9.49, −2.82, 4.2, 10.1, 1)	,,	2946*
8	(11.3, 0, −11.3, 0, −11.3, 0, 11.3, 1)	(0, 0, 0, 0, 0, 0, 0, .00775)	2265*
9	(−11.3, 0, 11.3, 0, −11.3, 0, 11.3, 0)	,,	1367*
10	(−13.8, 13.7, .0977, −11.8, .0977, 13.7, −13.8, 1)	(0, 0, 0, 0, 0, 0, 0, .00562)	2969†
11	(13.8, 13.7, −.0977, 11.8, −.0977, −13.7, 13.8, 1)	,,	2947†
12	(−14.1, 15.6, −14.1, 0, 14.1, 15.6, −14.1, 1)	(0, 0, 0, 0, 0, 0, 0, 00456)	4577†
13	(14.1, −15.6, 14.1, 0, −14.1, 15.6, −14.1, 1)	,,	4577†
14	(14.1, −15.8, 15.9, −16, 15.9, −15.8, 14.1, 1)	(0, 0, 0, 0, 0, 0, 0, .00404)	2006†
15	(−14, 15.8, −15.9, 16, −15.9, 15.8, −14.1, 1)	,,	1986†

* Local minima not corresponding to singular H' were found on these arcs.

† Due to a too-large tolerance, local minima not corresponding to a singular H' were identified as bifurcation points on these arcs; the minimization process correctly gave only one direction.

TABLE 3(b)

#	BP	PBP	FEP	PITL	FEB
1	(0, 0, 0, 0, 0, 0, 0, .00404)	(0, 0, 0, 0, 0, 0, 0, 0)	878	531	643
2	(0, 0, 0, 0, 0, 0, 0, .00455)	(0, 0, 0, 0, 0, 0, 0, .00404)	450	197	617
3	(0, 0, 0, 0, 0, 0, 0, .00561)	(0, 0, 0, 0, 0, 0, 0, .00455)	567	278	605
4	(0, 0, 0, 0, 0, 0, 0, .00775)	(0, 0, 0, 0, 0, 0, 0, .00561)	611	250	632
5	(0, 0, 0, 0, 0, 0, 0, .0125)	(0, 0, 0, 0, 0, 0, 0, .00775)	681	336	649
6	(0, 0, 0, 0, 0, 0, 0, .0260)	(0, 0, 0, 0, 0, 0, 0, .0125)	704	365	701
7	(0, 0, 0, 0, 0, 0, 0, .0931)	(0, 0, 0, 0, 0, 0, 0, .0260)	768	419	721

Total number of function evaluations 40,467. Total number of predictor steps 8,358. Number of function evaluations per arc: 809.

TABLE 4(a)

#	root	BP	FE
1	(0, 0, 0, 1)	(0, 0, 0, .5)	16
2	(7.05, −7.8, 7.05, 1)	,,	161
3	(2.13, 3.66, 2.13, 1)	,,	156
4	(−7.05, 7.8, 2.13, 1)	,,	161
5	(5.66, 0, −5.66, 1)	,,	160
6	(−5.66, 0, 5.66, 1)	,,	157
7	(−2.13, −3.66, −2.13, 1)	,,	132

TABLE 4(b)

#	BP	PBP	FEP	PITL	FEB
1	(0, 0, 0, .5)	(0, 0, 0, 0)	662	437	235, 732

Total number of function evaluations, 240,624. (Some function evaluations occurred while rejecting false directions at the bifurcation point.) Total number of predictor iterations, 1194. Number of function evaluations per branch, 34,374.

TABLE 5(a)

#	root	BP	FE
1	(0, 0, 1)	(0, 0, .667)	57
2	(0, .707, 1)	,,	205
3	(.5, 0, 1)	,,	193
4	(0, −.707, 1)	,,	204
5	(−.5, 0, 1)	,,	185

TABLE 5(b)

#	BP	PBP	FEP	PITL	FEB
1	(0, 0, .667)	(0, 0, 0)	468	315	54, 492

Total number of function evaluations, 55,804. Total number of predictor steps, 535. Number of function evaluations per arc, 11,160.

6. Summary, conclusions and possible improvements. We have explained various techniques for locating general bifurcation points and for following all arcs intersecting at such bifurcation points. These techniques were tried on five test examples.

The test results indicate the acceleration/deceleration scheme (§ 3) reliably finds bifurcation points, regardless of whether there is a change in the determinant of H_x, yet does not force excessive computation to be done where it is unnecessary. The efficiency can undoubtedly be further improved with a better choice of multiplication parameters for increasing and decreasing δ, etc.

The direction-finding algorithm (explained in § 4) seemed to work well on simple bifurcation points (where only two arcs intersect), but comparison with other methods would be desirable. Larger numbers of function evaluations were required when the dimension of the null space of $H'(\bar{y})$ was greater than 2, but arc directions were given very accurately, and restarting the arc following method caused no problems. Perhaps the technique of adjusting the radius of S_δ will allow computation of arc directions when the arcs intersect tangentially; this needs more investigation.

The excessive numbers of function evaluations in finding the $m^{(i)}$ when the dimension of the null space of H' is greater than 2 are due partially to the small tolerance ε_d used in the minimization routine; additional experiments are necessary to determine the effects of choosing ε_d larger. Perhaps a better way of finding all minima of a function of a small number of variables could be implemented.

There was an intrinsic difficulty in Example 4: numerous minima of $\|H\|$ not corresponding to arcs occurred on ∂S_δ. This phenomenon needs further study.

As presented above, each predictor-corrector step of the arc-following method will run in $o(n^3)$ algebraic operations; this is because solution of the system in formula

(4) and a singular value decomposition are both required. Appropriate use of matrix factorization and updating techniques will allow solution of the system in $o(n^2)$ operations [24]. Also, it may be possible to find minima of determinants related to this factorization in place of finding minima of σ_n, so that only $o(n^2)$ operations are required overall in a predictor-corrector step. The special structure of the problem can also be used to give more accurate H' without additional computations [24]. This will be reported on in the future.

Finally, we emphasize that the examples in § 5 were meant merely as tests of the techniques; not much can be said about the underlying differential equation when using a discretization with $n = 7$.

REFERENCES

[1] E. ALLGOWER, *A survey of homotopy methods for smooth mappings*, 1980 (in [17]).

[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] P. A. BUSINGER AND G. H. GOLUB, *Singular value decomposition of a complex matrix, Algorithm* 358, Comm. ACM, 12 (1960), pp. 564–565.

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

[5] A. EIGER, K. SIKORSKI AND F. STENGER, *A method of bisections for solving n nonlinear equations*, ACM Trans. Math. Software, submitted.

[6] C. B. GARCIA AND T. Y. LI, *On a path-following method for systems of equations*, MRC Tech. Summ. Rep. 1983, Univ. Wisconsin, Madison, 1979.

[7] C. B. GARCIA AND W. I. ZANGWILL, *Finding all solutions to polynomial systems and other systems of equations*, Math. Programming, 16 (1979), pp. 159–176.

[8] K. GEORG, *On tracing an implicitly defined curve by quasi-Newton steps and calculating bifurcation by local perturbations*, this Journal, 2 (1981), pp. 35–50.

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

[10] H. JÜRGENS, H. O. PEITGEN AND D. SAUPE, *Topological perturbations in the numerical nonlinear eigenvalue and bifurcation problems*, Rep. 7, Forschungsschwerpunkt Dynamische Systeme, Fachbereich Mathematik, Universität Bremen, 1979.

[11] R. B. KEARFOTT, *A derivative-free arc continuation method and a bifurcation technique*, 1980 (in [17]).

[12] ———, *An efficient degree-computation method for a generalized method of bisection*, Numer. Math., 32 (1979), pp. 109–127.

[13] ———, *An improved program for generalized bisection*, to appear.

[14] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Mathematics Research Center Publication 8, Academic Press, New York, 1977.

[15] T. Y. LI AND J. A. YORKE, *A simple reliable numerical algorithm for following homotopy paths*, in Analysis and Computation of Fixed Points, S. M. Robinson, ed., Academic Press, New York, 1979.

[16] H. D. MITTLEMAN AND H. WEBER, *Numerical methods for bifurcation problems, a survey and classification*, in Bifurcation Problems and Their Numerical Solution, H. D. Mittleman and H. Weber, eds., Birkhäuser, Basel, 1980.

[17] E. L. ALLGOWER, K. GLASHOFF AND H.-O. PEITGEN, eds, *Numerical Solution of Nonlinear Equations*, Lecture Notes in Mathematics 878, Springer, New York, 1981.

[18] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, Comput. J., 7 (1965), pp. 308–313.

[19] M. J. D. POWELL, *A Fortran subroutine for solving systems of nonlinear algebraic equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, New York, 1970.

[20] P. H. RABINOWITZ, *Some aspects of nonlinear eigenvalue problems*, Rocky Mountain J. Math., 3 (1973), pp. 161–202.

[21] F. STENGER, *Computing the topological degree of a mapping in R^n*, Numer. Math., 25 (1975), pp. 23–38.

[22] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[23] N. M. TEMME, ed. *Nonlinear Analysis*, vol. 1, Mathematisch Centrum, Amsterdam, 1976.

[24] MICHAEL TODD, private correspondence.

[25] L. T. WATSON, *A globally convergent algorithm for computing fixed points of C^2 maps*, Appl. Math. Comput., 5 (1979), pp. 297–311.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0005 $01.25/0

AN EXACT PENALTY METHOD FOR CONSTRAINED, DISCRETE, LINEAR l_∞ DATA FITTING*

BARRY JOE† AND RICHARD BARTELS†

Abstract. This paper presents an algorithm for solving linearly constrained, discrete, linear l_∞ approximation problems which makes use of a penalty linear programming approach. An implementation for small, dense problems has been prepared and tested against two other codes, one published and the other not. Results of this testing are given. The paper is concluded with a short summary.

Key words. *l*-infinity, Chebyshev, data fitting, regression, exact penalty method, optimization, linear programming, constraints.

1. Introduction and formulation.

We wish to solve the following problem:

Given a vector $c=[c_1, c_2, \cdots, c_n]^T$, an $m\times n$ matrix A, an $m\times p$ matrix E, an $m\times q$ matrix G, a p-vector f and a q-vector h,

$$\begin{aligned} &\text{minimize} && \|c-A^Ty\|_\infty \\ &\text{subject to} && E^Ty=f \\ &\text{and} && G^Ty\geqq h, \end{aligned} \tag{1.1}$$

where $y=[y_1, y_2, \cdots, y_m]^T$ is a vector of unknown parameters. We will not be assuming anything about the nonnegative integers n, m, p or q.

It is assumed that all vectors and matrices are real. None of the matrices A, E, or G are required to be of full rank—indeed, E and/or G may be vacuous—nor is it assumed that vectors y satisfying the constraints actually exist, so the problem as posed is completely general.

Problem (1.1) can be formulated as the linear programming problem:

$$\begin{aligned} &\text{minimize} && e_1^T\begin{bmatrix} y_0 \\ y \end{bmatrix} \\ &\text{subject to} && \begin{bmatrix} e & A^T \\ e & -A^T \\ 0 & E^T \\ 0 & -E^T \\ 0 & G^T \end{bmatrix} \begin{bmatrix} y_0 \\ y \end{bmatrix} \geqq \begin{bmatrix} c \\ -c \\ f \\ -f \\ h \end{bmatrix} \\ &\text{and} && y_0\geqq 0, \end{aligned} \tag{1.2}$$

where y_0 is a variable representing $\|c-A^Ty\|_\infty$, e_1 is the vector $[1, 0]^T$ of dimension $m+1$, i.e. the first unit vector, and e is the n-vector containing a 1 in all components.

Problem (1.2) can be converted to its dual linear programming form:

$$\text{maximize} \quad [c^T \quad -c^T \quad f^T \quad -f^T \quad h^T \quad 0 \quad 0^T] \begin{bmatrix} t \\ u \\ v \\ w \\ z \\ s_0 \\ s \end{bmatrix} \tag{1.3}$$

* Received by the editors November 9, 1980, and in revised form April 12, 1982.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

(1.3)

$$\text{subject to } \begin{bmatrix} e^T & e^T & 0^T & 0^T & 0^T & 1 & 0^T \\ A & -A & E & -E & G & 0 & I \end{bmatrix} \begin{bmatrix} t \\ u \\ v \\ w \\ z \\ s_0 \\ s \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = e_1$$

$$\text{and} \qquad t \geqq 0,\ u \geqq 0,\ v \geqq 0,\ w \geqq 0,\ z \geqq 0,\ s_0 \geqq 0,\ 0 \geqq s \geqq 0,$$

where t and u are n-vectors, v and w are p-vectors, z is a q-vector, s_0 is a slack variable, s is an m-vector of artificial variables, and I is the order-m identity matrix. The artificial variables $s_1, s_2, \cdots, s_m$ are added to ensure that the matrix has full row rank and to give an initial basis.

Recently Bartels [5] has described a penalty linear programming method using reduced gradient, basis exchange techniques, which is a variant of the piecewise linear penalty function approach of Conn [9]. In this paper, we describe an algorithm for solving problem (1.1) by way of applying that approach to (1.3). In § 2 we cover the important issues of [5] with a view to solving (1.3). In § 3 we present our algorithm. In § 4 we lay out some preliminary comparisons of this algorithm with one of several from the literature (for reference) and with one which applies the method of [9] directly to (1.2). In § 5 we give brief comments relating some aspects of our approach with those of two other methods in the recent literature.

2. The penalty linear programming method. In this section we review the penalty linear programming method of [5] in a form useful to us. Let $\bar{c}$ be a vector of the form $[c'^T\ 0^T]^T$, where c' is an n-vector and 0 is the zero vector of length m. Let $\bar{A}$ have the form $[A'\ I]$, where A' is an $m \times n$ matrix, and I is the identity of order m. Let b be an m-vector, and let x have the form $[x'^T\ x''^T]^T$, where x' and x'' have n and m components respectively. The components of x'' play the role of artificial variables. Consider the problem

(2.1)

$$\begin{aligned} &\text{maximize} && \bar{c}^T x \\ &\text{subject to} && \bar{A}x = b \\ &\text{and} && x' \geqq 0,\ 0 \geqq x'' \geqq 0. \end{aligned}$$

We note that (1.3) is a special case of this problem.

The method given in [5] solves (2.1) by addressing the related problem:

(2.2)

$$\text{maximize} \quad \varphi(x) = \mu \bar{c}^T x + \sum_{j=1}^{n+m} \min(0, x_j) + \sum_i{}' \min(0, -x_i),$$

where $\mu > 0$ is sufficiently small and x is restricted to satisfy $\bar{A}x = b$.

Throughout this paper $\sum'$ will indicate summations which are to be carried out only in the range of the artificial variable indices: $(n+1 \leqq i \leqq n+m)$. *For* $\mu > 0$ sufficiently small, an optimal solution x^* of (2.2) is also an optimal solution of (2.1).

In [5], Bartels describes the penalty linear programming method for a problem similar to (2.1), but with general upper and lower bounds on the variables. Since (2.1) is simpler and we wish to expound upon some ideas involving degeneracy, we include the following brief discussion for completeness.

Let $\bar{A}$, $\bar{c}$, and x be partitioned such that

$$\bar{A}x = [B \quad N]\begin{bmatrix} x_B \\ x_N \end{bmatrix} \quad \text{and} \quad \bar{c}^T x = [\bar{c}_B^T \quad \bar{c}_N^T]\begin{bmatrix} x_B \\ x_N \end{bmatrix}, \tag{2.3}$$

where B is an $m \times m$ nonsingular matrix. The columns of $\bar{A}$ may have to be permuted to obtain such a submatrix, B, but we shall ignore this detail for the sake of clarity. In the terminology of the simplex method, B is a *basis matrix*, and its columns (the current *basic columns* of $\bar{A}$) are a *basis* for $\bar{A}$. The components of x_B are the *basic variables* associated with B. The components of x_N are the *nonbasic variables*, and the columns of N are the *nonbasic columns*.

We define x_B to be *degenerate* if it contains at least one variable with value zero, and define x_B to be *infeasible* if it contains at least one variable with value less than zero or at least one artificial variable with value greater than zero. We will establish the following conventions: Initially the basic variables are the m artificial variables, $B = I$, $x_B = b$, and x_N will be set to the zero vector at each step of the solution process. Furthermore, whenever an artificial variable becomes nonbasic, we will drop it from the problem. This means that the number of components in x_N may change from step to step during the solution process.

Let $\bar{A}x = b$ and $\bar{A}v = [B\,N]\begin{bmatrix} v_B \\ v_N \end{bmatrix} = 0$, where the vector v is an ascent direction for φ at the point x. Let $g_N = \mu\bar{c}_N$ and g_B be defined as follows:

$$g_{B_i} = \begin{cases} \mu\bar{c}_{B_i} & \text{if } x_{B_i} = 0 \quad \text{(any variable)}, \\ \mu\bar{c}_{B_i} & \text{if } x_{B_i} > 0 \quad \text{(nonartificial variable)}, \\ \mu\bar{c}_{B_i} + 1 & \text{if } x_{B_i} < 0 \quad \text{(any variable)}, \\ \mu\bar{c}_{B_i} - 1 & \text{if } x_{B_i} > 0 \quad \text{(artificial variable)}, \end{cases} \tag{2.4}$$

and let π be the solution to $B^T\pi = g_B$. It can be shown, for $\beta > 0$ small enough, that $\varphi(x + \beta v) = \varphi(x) + \beta(S_1 + S_2)$, where $S_1 = (g_N^T - \pi^T N)v_N + \sum_{\text{nonbasic}} \min(0, v_{N_i})$ and $S_2 = \sum_{x_{B_i}=0} \min(0, v_{B_i}) + \sum_{x_{B_i}=0} \min(0, -v_{B_i})$. We note that $S_2 \leqq 0$ and $S_2 = 0$ if x_B is nondegenerate.

An algorithm for solving (2.2) for a fixed value of μ, in which nondegeneracy is not assumed, consists of a repetition of the following steps (starting with an initial x which satisfies $\bar{A}x = b$):

(2.5)
(1) Find $v \neq 0$ satisfying $\bar{A}v = 0$ such that $\varphi(x) + \beta S_1 > \varphi(x)(S_1 > 0)$ for $\beta > 0$ small enough. Optimality occurs if no such v exists.
(2) Choose $\beta \geqq 0$ so as to maximize $\varphi(x + \beta v) = \varphi(x) + \beta(S_1 + S_2)$ as a function of β, or discover that φ is unbounded.
(3) Replace x by $x + \beta v$.

If optimality occurs and x is infeasible, or if it is discovered that φ is unbounded, then μ is reduced, and steps 1–3 are repeated, assuming that μ is not negligibly small. (In a computational sense we will say that μ is negligibly small, if the magnitude of the computed value $\mu\bar{c}^T x$ is smaller than machine epsilon times the magnitude of the computed value of $\varphi(x)$.) Termination of the maximization process comes when optimality occurs and x_B is feasible (which indicates an optimal solution to (2.1)) or μ is negligibly small (which indicates that the solution to (2.1) is infeasible or unbounded).

The above three steps correspond to the following steps in the simplex method: (1) the selection of a nonbasic column to enter the basis, (2) the selection of a basic column to leave the basis and (3) the basis exchange step.

As in the simplex method, it is suitable to restrict the consideration of v_N in step (1) of (2.5) to vectors having only one nonzero component. Let $z_N = g_N - N^T\pi$ be the *reduced cost* vector. Suppose z_{N_σ} is the most *out of kilter* reduced cost, that is, σ is an index which provides the maximum:

$$\max\{\max_{z_{N_j}>0}(z_{N_j}),\ \max_{z_{N_j}<-1}(-z_{N_j}-1)\}. \tag{2.6}$$

Following the results of [5]: if $-1 \leqq z_{N_j} \leqq 0$ for all j, then $S_1 \leqq 0$ for all choices of v_N, and x is a maximizer of φ. Suppose $z_{N_\sigma} > 0$ or $z_{N_\sigma} < -1$. Let $v_{N_\sigma} = \operatorname{sgn}(z_{N_\sigma})$, $v_{N_j} = 0$ for $j \neq \sigma$, and $v_B = -v_{N_\sigma} B^{-1} N_\sigma$ where N_σ denotes column σ of N. Then $S_1 = \max\{z_{N_{\sigma'}} - z_{N_\sigma} - 1\} > 0$, and the nonbasic vector N_σ is chosen to enter the basis.

There are two cases to be considered in step (2) of (2.5):

(2.7) (a) $S_1 + S_2 > 0$ which means we can choose $\beta > 0$,
(b) $S_1 + S_2 \leqq 0$ which means we must choose $\beta = 0$.

First suppose $S_1 + S_2 > 0$. Then $\varphi(x + \beta v)$ is a piecewise linear function of β with breakpoints given by the β values $-x_{B_i}/v_{B_i}$. The maximum of this piecewise linear function will be found at one of the positive ratio values. Let k be the number of these positive ratios. If $k = 0$, then φ is unbounded. Suppose $k \geqq 1$ and let the positive ratios be inspected in their nondecreasing numeric order (ties are broken arbitrarily):

$$0 < \beta^{(1)} \leqq \beta^{(2)} \leqq \cdots \leqq \beta^{(k)}, \tag{2.8}$$

where

$$\beta^{(l)} = -\frac{x_B^{(l)}}{v_B^{(l)}} = -\frac{x_{B_i}}{v_{B_i}}$$

for some index i. Let

$$\alpha^{(l)} = \begin{cases} 1 & \text{if } x_B^{(l)} \text{ is a nonartificial variable,} \\ 2 & \text{if } x_B^{(l)} \text{ is an artificial variable.} \end{cases}$$

Then the slope of the piecewise linear function of β is $t^{(0)} = S_1 + S_2$ on the interval $0 \leqq \beta \leqq \beta^{(1)}$, and it is $t^{(l)} = t^{(l-1)} - \alpha^{(l)}|v_B^{(l)}|$ on the interval $\beta^{(l)} \leqq \beta \leqq \beta^{(l+1)}$ for $l = 1, 2, \cdots, k$, where $\beta^{(k+1)} = \infty$. If there exists an l satisfying $1 \leqq l \leqq k$ such that $t^{(l)} \leqq 0$ and $t^{(l-1)} > 0$, then $\varphi(x + \beta v)$ is maximized for $\beta = \beta^{(l)}$, and basic vector B_ρ is chosen to leave the basis, where x_{B_ρ} and $x_B^{(l)}$ are the same variable. If $t^{(k)} > 0$, then φ is unbounded.

Now suppose $S_1 + S_2 \leqq 0$. This case can only occur if x_B is degenerate. As in the simplex method with a degenerate basic feasible solution, cycling can occur since $\beta = 0$ and $\varphi(x)$ is not changed. We want to choose a basic variable x_{B_ρ} with value 0 to become nonbasic, where from the definition of S_2.

$$\rho \in J = \{j | x_{B_j} = 0 \text{ and } v_{B_j} < 0 \text{ or artif. var. } x_{B_j} = 0 \text{ and } v_{B_j} < 0\}.$$

Let r be the number of indices in J. Suppose the v_{B_j}'s, where $j \in J$, are ordered such that

$$|v_B^{(1)}| \geqq |v_B^{(2)}| \geqq \cdots \geqq |v_B^{(r)}|,$$

and variable $x_{B_\rho} \equiv x_B^{(i)}$ is chosen to become nonbasic where i satisfies

$$S_1 - \sum_{l=1}^{i} |v_B^{(l)}| \leqq 0 \quad \text{and} \quad S_1 - \sum_{l=1}^{i-1} |v_B^{(l)}| > 0. \tag{2.9}$$

This choice of x_{B_ρ} is heuristic and does not guarantee that cycling will not occur, but we have had no problems with this scheme on any l_∞ problem which we have tried.

We had been using an explicit perturbation method with randomly generated perturbations to resolve degeneracies, but we have found method (2.9) to be easier to code and to require less storage for record keeping.

The proposed choice of x_{B_ρ} to be used in the case of degeneracy is, in fact, equivalent to a scheme which attempts to resolve degeneracies by a highly structured sort of perturbation. Note that if, for $j \in J$, x_{B_j} is perturbed by a small positive real number ε, so that $x_{B_j} = \varepsilon$ if $v_{B_j} < 0$ and $x_{B_j} < 0$ and $x_{B_j} = -\varepsilon$ if $v_{B_j} > 0$, then S_2 would become zero. Furthermore, the resulting positive ratios $-x_{B_j}/v_{B_j}$ satisfy

$$\frac{\varepsilon}{|v_B^{(1)}|} \leqq \frac{\varepsilon}{|v_B^{(2)}|} \leqq \cdots \leqq \frac{\varepsilon}{|v_B^{(r)}|}$$

and these ratios could be inserted in front of those in (2.8). The process of choosing a value of β, given above by case (a) of (2.7) and the ensuing paragraph, reduces to (2.9). Finally, if $\beta = \varepsilon/|v_B^{(i)}|$ maximizes $\varphi(x+\beta v)$, then $\varphi(x+\beta v) \approx \varphi(x) + \varepsilon$.

Let $\beta^{(-r+1)} = -|v_B^{(l)}|$ for $l = 1, 2, \cdots, r$. Then

$$\beta^{(-r+1)} \leqq \cdots \leqq \beta^{(0)} < 0 < \beta^{(1)} \leqq \cdots \leqq \beta^{(k)}. \tag{2.10}$$

By using (2.9), cases (a) and (b) of (2.7) can be combined to form the following loop for choosing basic vector B_ρ to leave the basis:

```
t := S1 ≡ max {z_Nσ, −z_Nσ − 1}
l := −r + 1
while (t > 0 and l ≦ k) do
    begin
    i := {index of basic variable associated with β^(l)}
    t := t − |v_Bi|
    if (x_Bi nonzero artif. var.) then t := t − |v_Bi|
    if (t ≦ 0) then
      begin β := max(β^(l), 0); ρ := i end
    l := l + 1
  end
```

(2.11)

If no basic index ρ is found, then unboundedness is indicated. This loop also accommodates the cases in which $k = 0$ and/or $r = 0$.

3. A constrained l_∞ linear approximation algorithm. In this section, we describe an algorithm for solving (1.3) using the penalty linear programming method described in the previous section. Since problem (1.3) is a special case of problem (2.1), the algorithm of [5] can be used for finding a solution. But, due to the special structure of $\bar{A}$, $\bar{c}$, and $b = e_1$ as they appear in (1.3), some modifications can be made for the sake of efficiency.

First we make some definitions and observations. let the columns of $\bar{A}$ be indexed (from left to right):

$$\begin{aligned} &1 \text{ to } n, \\ &-1 \text{ down to } -n, \\ &n+1 \text{ to } n+p, \\ &-(n+1) \text{ down to } -(n+p), \\ &n+p+1 \text{ to } n+p+q+m+1. \end{aligned} \tag{3.1}$$

Let the vectors $\bar{c}$, g, x, and z be indexed in the same way, where

$$g_j = \begin{cases} \mu\bar{c}_j + 1 & \text{if } x_j \text{ is basic and } x_j < 0, \\ \mu\bar{c}_j - 1 & \text{if } x_j \text{ is basic, artificial and } x_j > 0, \\ \mu\bar{c}_j & \text{if } x_j \text{ is nonbasic or } x_j \text{ is basic and } x_j = 0, \\ & \text{or } x_i \text{ is basic, nonartificial and } x_j > 0. \end{cases}$$

Let $\bar{A}$, $\bar{c}$, g, and x be partitioned into

$$[B \quad N], \quad \begin{bmatrix} c_B \\ c_N \end{bmatrix}, \quad \begin{bmatrix} g_B \\ g_N \end{bmatrix}, \quad \begin{bmatrix} x_B \\ x_N \end{bmatrix}$$

as in (2.3). Consistent with our use of z_N in the last section, we define $z = g - \bar{A}^T\pi$, and regard this vector to be partitioned into

$$z = \begin{bmatrix} z_B \\ z_N \end{bmatrix},$$

with $z_N = g_N - N^T\pi$ as before, and with $z_B = g_B - B^T\pi$. Let

$\bar{A}_j$ denote the jth column of $\bar{A}$, $1 \leqq |j| \leqq n+p$ and $n+p+1 \leqq j \leqq n+p+q+m+1$,
A_j denote the jth column of A, $1 \leqq j \leqq n$,
E_j denote the jth column of E, $1 \leqq j \leqq p$,
G_j denote the jth column of G, $1 \leqq j \leqq q$.

We note the following facts:

$$\begin{aligned} &\bar{c}_{-j} = -\bar{c}_j \text{ for } 1 \leqq |j| \leqq n+p, \\ &\bar{A}_j + \bar{A}_{-j} = 2e_1 \text{ for } 1 \leqq |j| \leqq n, \\ &\bar{A}_j + \bar{A}_{-j} = 0 \text{ for } n+1 \leqq |j| \leqq n+p, \\ &\bar{A}_j = \begin{bmatrix} 1 \\ A_j \end{bmatrix} \text{ and } \bar{A}_{-j} = \begin{bmatrix} 1 \\ -A_j \end{bmatrix} \text{ for } 1 \leqq j \leqq n, \\ &\bar{A}_{n+j} = \begin{bmatrix} 0 \\ E_j \end{bmatrix} \text{ and } \bar{A}_{-n-j} = \begin{bmatrix} 0 \\ -E_j \end{bmatrix} \text{ for } 1 \leqq j \leqq p, \\ &\bar{A}_{n+p+j} = \begin{bmatrix} 0 \\ G_j \end{bmatrix} \text{ for } 1 \leqq j \leqq q. \end{aligned} \tag{3.2}$$

Note that the optimal variables π^* satisfy $\pi^* = \mu\begin{bmatrix} y_0^* \\ y^* \end{bmatrix}$, where $\begin{bmatrix} y_0^* \\ y^* \end{bmatrix}$ is an optimal solution of (1.2). Thus, it makes sense to define $\begin{bmatrix} y_0 \\ y \end{bmatrix}$, as

$$\frac{\pi}{\mu} \equiv \begin{bmatrix} \dfrac{\pi_1}{\mu} \\ \dfrac{\pi'}{\mu} \end{bmatrix}$$

where $\begin{bmatrix} y_0 \\ y \end{bmatrix}$ is the vector of variables in (1.2). Since $\pi^T = g_B^T B^{-1}$ and $Bx_B = e_1$, we note that

$$\pi_1 = \pi^T e_1 = g_B^T x_B = \mu c_B^T x_B + \sum_{x_{B_i} < 0} x_{B_i} + \sum_{x_{B_i} > 0}' (-x_{B_i}) = \varphi(x). \tag{3.3}$$

Finally, we define the vector r as follows:

$$
\begin{aligned}
&r_j = \mu c_j - A_j^T \pi' && \text{for } 1 \leqq j \leqq n, \\
&r_{n+j} = \mu f_j - E_j^T \pi' && \text{for } 1 \leqq j \leqq p, \\
&r_{n+p+j} = \mu h_j - G_j^T \pi' && \text{for } 1 \leqq j \leqq q, \\
&r_{-j} = -r_j && \text{for } 1 \leqq j \leqq n+p.
\end{aligned} \tag{3.4}
$$

We will show that the computation of the reduced cost vector z can be simplified for our problem. Assume, for now, that $0 \leqq \pi_1 \leqq 0.5$. Consider the computation of z_j, where x_j is a nonbasic variable, and recall that x_j may be considered for becoming basic if $z_j > 0$ or $z_j < -1$. Artificial variables are dropped from the problem as they become nonbasic so they are not considered in the computation of z. Also, the slack variable does not have to be considered when it becomes nonbasic, since $z_j = -\pi_1$ or $-0.5 \leqq z_j \leqq 0$ for $j = n+p+q+1$. So the three cases to be considered are:

(a) $1 \leqq |j| \leqq n$ (equations),
(b) $n+1 \leqq |j| \leqq n+p$ (equality constraints),
(c) $n+p+1 \leqq j \leqq n+p+q$ (inequality constraints).

For nonbasic variable x_j,

$$
z_j = g_j - \bar{A}_j^T \pi = \begin{cases} r_j - \pi_1 & \text{if } 1 \leqq |j| \leqq n, \\ r_j & \text{if } n+1 \leqq |j| \leqq n+p \text{ or } n+p+1 \leqq j \leqq n+p+q. \end{cases}
$$

[We note that π_1 is r_i for basic variables x_i with nonnegative values where $1 \leqq |i| \leqq n$ (refer to (3.4)).]

Case (a). First suppose x_j and x_{-j} are both nonbasic variables. Then $z_j = r_j - \pi_1$ and $z_{-j} = -r_j - \pi_1$ using (3.4). Assume, without loss of generality, that $r_j \geqq 0$ (since j may be positive or negative). Since $\pi_1 \leqq 0.5$, it follows that $z_j \geqq -z_{-j} - 1$ if $z_j > 0$, and $z_j \geqq -0.5$ and $0 \geqq z_{-j} \geqq -1$ if $z_j \leqq 0$. If (2.6) is used to choose x_σ to become basic, then x_{-j} will never be chosen. Therefore, we only have to consider x_j to become basic if $z_j > 0$ ($r_j > \pi_1$).

Now suppose x_j is nonbasic and x_{-j} is basic. If $x_{-j} \geqq 0$, then $z_{-j} = 0$ and $z_j = -z_{-j} - 2\pi_1 = -2\pi_1$ or $-1 \leqq z_j \leqq 0$, so x_j is not considered for becoming basic. If $x_{-j} < 0$, then $z_{-j} = r_{-j} - \pi_1 + 1 = 0$ and $z_j - 2\pi_1 - 1 = 1 - 2\pi_1 \geqq 0$, so x_j can be considered for becoming basic.

Case (b). First suppose x_j and x_{-j} are both nonbasic variables. Then $z_j = r_j$ and $z_{-j} = -z_j$. Assume, without loss of generality, that $r_j \geqq 0$. We note that $z_j > -z_{-j} - 1 = z_j - 1$, so x_{-j} is not considered for becoming basic, and x_j can be considered for becoming basic if $r_j > 0$. Now suppose x_j is nonbasic and x_{-j} is basic. If $x_{-j} \geqq 0$, then $z_{-j} = 0$ and $z_j = 0$, so x_j is not considered for becoming basic. If $x_{-j} < 0$, then $z_{-j} = r_{-j} + 1 = 0$ and $z_j = -r_{-j} = 1$, so x_j can be considered for becoming basic.

Case (c). Suppose x_j is nonbasic. Then $z_j = r_j$ and x_j can be considered for becoming basic if $r_j > 0$ or $r_j < -1$.

Summarizing these three cases, nonbasic variables x_j satisfying one of the following conditions can be considered for becoming basic:

(i) $r_j - \pi_1 > 0$, $1 \leqq |j| \leqq n$,
(ii) $r_j > 0$, $n+1 \leqq |j| \leqq n+p$ or $n+p+1 \leqq j \leqq n+p+q$,
(iii) $r_j < -1$, $n+p+1 \leqq j \leqq n+p+q$,
(iv) x_{-j} is basic and $x_{-j} < 0$, $1 \leqq |j| \leqq n+p$. (3.5)

We show that x_j and x_{-j} will never be simultaneously basic, assuming $0 \leqq \pi_1 \leqq 0.5$, by proving the following statement:

(3.6) If x_j is chosen to become basic due to (3.5,iv), then x_{-j} is chosen to become nonbasic using (2.11).

Assume, then, that x_j has been chosen to become basic.

First suppose $1 \leqq |j| \leqq n$. Then $z_j = 1 - 2\pi_1 > 0$, $v_j = 1$, and $Bv_B = -\bar{A}_j$. Let $B_1, B_2, \cdots, B_{m+1}$ denote the columns of B and $B_k = \bar{A}_{-j}$. Since $\frac{1}{2}\bar{A}_j + \frac{1}{2}\bar{A}_{-j} = e_1$ and $Bx_B = e_1$,

$$x_{B_1}B_1 + \cdots + x_{B_k}B_k + \cdots + x_{B_{m+1}}B_{m+1} - \tfrac{1}{2}\bar{A}_{-j} = \tfrac{1}{2}\bar{A}_j,$$

$$v_{B_1}B_1 + \cdots + v_{B_k}B_k + \cdots + v_{B_{m+1}}B_{m+1} = -\bar{A}_j.$$

Therefore

$$v_{B_i} = \begin{cases} -2x_{B_i} & \text{if } i \neq k, \\ 1 - 2x_{B_k} = 1 - 2x_{-j} & \text{if } i = k, \end{cases}$$

and the positive ratios

$$-\frac{x_{B_i}}{v_{B_i}} = \begin{cases} \frac{1}{2} & \text{if } i \neq k \text{ and } x_{B_i} \neq 0, \\ \dfrac{-x_{-j}}{1 - 2x_{-j}} < \frac{1}{2} & \text{if } i = k. \end{cases}$$

Using (2.11), $z_j - v_{B_k} = (1 - 2\pi_1) - (1 - 2x_{-j}) = 2(x_{-j} - \pi_1) < 0$, so x_{-j} is chosen to become nonbasic.

Now suppose $n + 1 \leqq |j| \leqq n + p$. Then $z_j = 1$, $v_j = 1$, and $Bv_B = -\bar{A}_j$. Since $\bar{A}_j + \bar{A}_{-j} = 0$,

$$v_{B_i} = \begin{cases} 0 & \text{if } i \neq k, \\ 1 & \text{if } i = k \text{ where } B_k = \bar{A}_{-j}. \end{cases}$$

The only positive ratio is $-x_{B_k}/v_{B_k} = -x_{-j}$. Using (2.11), $z_j - v_{B_k} = 0$, so x_{-j} is chosen to become nonbasic.

Our current algorithm considers only the first three conditions of (3.5) in choosing a nonbasic variable to become basic, but (3.5,iv) can be added as a special case. If x_j is chosen to become basic due to (3.5,iii), then inequality constraint $l = j - (n + p)$ is satisfied ($G_l^T y \geqq h_l$), so it is not necessary for x_j to become basic. In our algorithm, μ is reduced when this occurs to decrease the probability of another x_j chosen to become basic due to (3.5,iii) and also to decrease the probability of cycling when x_B is degenerate. Our goal is to arrive at a solution to (1.3) as quickly as possible, not merely to maximize φ for a fixed choice of μ.

We will now dispense with our assumption that $0 \leqq \pi_1 \leqq 0.5$. Since x_B remains at e_1 until the slack variable becomes nonbasic, $\pi_1 = 0$ if the slack variable is basic, and $\pi_1 > 0$ when the slack variable becomes nonbasic by (3.3). If μ is too large, then π_1 may become greater than 0.5. In our algorithm, if $\pi_1 > 0.5$ then μ is reduced so that when $B^T\pi = g_B$ is re-solved for π, $\pi_1 \leqq 0.5$. By (3.3), π_1 is reduced when μ is reduced, and $\pi_1 < 0$ may result if x_B is infeasible. Since $\pi_1 = \varphi(x)$ is being maximized, π_1 will increase past 0 again. We note that if $\pi_1 < 0$, then (3.6) may not hold for j such that $1 \leqq |j| \leqq n$.

As a result of the above discussion, a good initial choice of μ is one satisfying $\mu\|c\|_\infty \approx 1$, due to the computation of g_B in (2.4) and since $y_0^* \leqq \|c\|_\infty$ implies that $\pi_1 = \mu y_0 \leqq 1$. Assuming this initial choice of μ, decreasing μ by $\frac{1}{2}$ is sufficient whenever μ has to be reduced.

The magnitude of μ affects how infeasible x_B becomes during the iterations of the algorithm. Since g_B, g_N, and π depend on μ, the magnitude of z_σ in (2.6) depends on μ. But $v_B = -v_{N_\sigma} B^{-1} N_\sigma$ does not depend on μ, so the magnitude of β and the number of basic variables skipped in (2.11) are affected by the magnitude of μ. These variables become negative for nonartificial variables and nonzero for artificial variables if $\beta > 0$. Therefore, if μ is too large unboundedness will occur, and if μ is sufficiently small x_B will not become infeasible.

For problem (1.3), the initial basic variables are the slack variable and the m artificial variables, and initially $x_B = e_1$ is degenerate. The basic solution x_B remains at e_1, although the basic variables change, until the slack variable becomes nonbasic, since the slack variable is the only variable that has a nonzero value initially. Since the artificial variables have value 0 initially, a slight modification is made to (2.10) to allow the artificial variables to become nonbasic earlier, which usually results in fewer iterations of the algorithm. During the first m iterations,

(3.7) if x_{B_i} is not an artificial variable, $x_{B_i} = 0$ and $v_{B_i} < 0$
then β_i is set to 0 instead of $-|v_{B_i}|$.

The constrained l_∞ linear approximation algorithm using the penalty linear programming method is summarized below. Input to the algorithm are the matrix $[A \quad E \quad G]$, the vector $[c^T \quad f^T \quad h^T]^T$, and the scalars m, n, p, q and μ.

$B := I$
$x_B := e_1$
iter := 0
repeat
  {Compute g_B as in (2.4)}
  {Solve $B^T\pi = g_B$ for π}
  if $(\pi_i > 0.5)$ **then**
    begin $\mu := \mu/2$; **go to** TEST **end**
  iter := iter + 1
  for $j := 1$ **until** n **do**
    if {x_j and x_{-j} are nonbasic} **then** $z_j := |g_j - A_j^T\pi'| - \pi_1$
  for $j := n+1$ **until** $n+p$ **do**
    if {x_j and x_{-j} are nonbasic} **then** $z_j := |g_j - E_{j-n}^T\pi'|$
  for $j := n+p+1$ **until** $n+p+q$ **do**
    if {x_j is nonbasic} **then** $z_j := g_j - G_{j-n-p}^T\pi'$
  {Find nonbasic index σ using (2.6)}
  if {no σ found} **then**
(3.8) **begin**
      if {x_B is infeasible} **then** $\mu := \mu/2$
      go to TEST
    end
  if $(\sigma \leqq n$ **and** $g_\sigma - A_\sigma^T\pi' < 0)$ **or**
    $(n+1 \leqq \sigma \leqq n+p$ **and** $g_\sigma - E_{\sigma-n}^T\pi' < 0)$ **then** $\sigma := -\sigma$
  if $(\sigma > n+p$ **and** $z_\sigma < -1)$ **then** $\mu := \mu/2$
  $v_\sigma := \text{sgn}(z_\sigma)$
  {Solve $Bv_B = -v_\sigma \bar{A}_\sigma$ for v_B}
  {Compute the β's as in (2.10)
  with modification (3.7) if *iter* $\leq m$}
  {Execute (2.11) to find basic index ρ}
  if {no ρ found} **then**

begin $\mu := \mu/2$; **go to** TEST **end**
{Set $x_B := x_B + \beta v_B$ and $x_\sigma := \beta v_\sigma$}
{Replace $\bar{A}_\rho$ in B by $\bar{A}_\sigma$}
TEST:
until {x_B is optimal for (1.3)} **or** {μ is too small}

If there exists a solution to (1.1), (1.2), (1.3) then this algorithm will find an optimal solution $\begin{bmatrix} y_0 \\ y^* \end{bmatrix} = \pi^*/\mu$ after a finite number of iterations (usually $O(m)$ iterations). If $y_0^* = 0$, then termination should occur after at most $m+1$ iterations. If the matrix $[A \quad E \quad G]$ has rank $k < m$, then $m-k$ artificial variables will remain basic with value 0, the corresponding $m-k$ components of y will have value 0, and y^* will likely be a nonunique solution. If problem (1.3) does not have a solution, then this algorithm will detect unboundedness or infeasibility (μ is negligibly small).

4. Numerical results. An experimental FORTRAN subroutine has been prepared to implement the method described here for small, dense problems. To strive for numerical stability, the code maintains the basis matrix B in LU-factored form and updates this form using Gauss elimination with partial pivoting after each exchange of a basic for a nonbasic variable.

This code was tested against the implementation given in [4] for the Barrodale–Phillips algorithm [3] and against an unpublished code which had been written by R. H. Bartels and A. R. Conn to investigate the applicability of using Conn's linear programming method, presented in [9], on the l_∞ problem in form (1.2). The Barrodale–Phillips code handles only the uncontrained version of (1.1), though nothing would prevent it from being extended to handle constraints, and it uses a compressed tableau version of the simplex method on a linear programming equivalent to (1.1) which corresponds roughly to our (1.3). The Bartels–Conn code was not constructed to handle equality constraints, though nothing would prevent its extension to this case. It maintains a separate matrix of active equation/constraint data in QR-factored form which it updates through the use of "slow" Givens transformations.

The method of this paper and the Bartels-Conn method represent the two most straightforward approaches to solving (1.1) using piecewise linear exact penalty functions joined with some linear programming formulation of the l_∞ problem. It was of interest to us to see which linear programming formulation would be most effective for use with the penalty function approach, the "direct" formulation (1.2) or the "dual" formulation (1.3). The algorithm by Barrodale and Phillips was included as a reference, since it appears to be a reasonably effective general purpose method appearing in the literature for small dense problems, and its code is easily accessible. These computational results are only intended to be preliminary and to support the overall impression that (a) the proposed method shows no obvious disadvantages to the standard approaches, and (b) with respect to the two obvious penalty approaches to solving (1.1)—namely via [5] applied to (1.3) or via [9] applied to (1.2)—the proposed method seems to be currently the more promising.

We have used the method described in [7] to generate random test problems with known solutions for comparing the number of iterations and central processor times used by these three programs. Some of the problems were purely random; that is, the elements of the matrix A were drawn directly from a random number generator. Others of the problems were randomly generated data fitting problems; that is, the elements of the matrix A were derived from random evaluations of a Chebyshev system of basis functions $\varphi_1(x), \cdots, \varphi_m(x)$. The types of random problems included some which were unconstrained and of full rank, some which were unconstrained and

rank deficient, some which had constraints and were full rank and some which were constrained but of deficient rank. Finally, we included several nonrandom problems which involved determining the best polynomial appoximations to various sets of given data. Most of these last problems were taken from [3].

From now on, we refer to the three codes by their FORTRAN subroutine names. Our algorithm is called CLINF, the algorithm by Barrodale and Phillips is called CHEB, and the algorithm by Bartels and Conn is called CL8. The computations were performed in double precision (18 decimal digits) on a Honeywell 66/60. For subroutines CLINF and CHEB, there is a tolerance parameter used to indicate zero equivalence. This parameter was set to 10^{-17}. The machine epsilon is 0.217×10^{-18}. For subroutine CL8, there is no tolerance parameter and the machine epsilon is used to indicate zero equivalence.

The results from the random unconstrained problems are given in Table 4.1. The elements of matrix A and l_∞ solution $\begin{bmatrix} y_0^* \\ y^* \end{bmatrix}$ are of magnitude $O(1)$. An initial value of $\mu = 1.0$ was used for CLINF.

TABLE 4.1
Average number of iterations and average CPU *time in milliseconds over* 12 *problems for each value of* m ($n = 10m$).

		CLINF		CHEB		CL8	
m	rank	iterations	time	iterations	time	iterations	time
5	5	14.6	302	12.9	268	11.4	600
10	10	32.3	1617	31.1	1705	24.6	2627
15	15	56.3	5055	56.3	5791	41.9	7214
5	4	11.9	241	11.6	233	7.8	433
10	8	28.2	1369	23.6	1222	19.0	2095
15	12	43.0	3749	46.9	4382	-	-

For the full rank problems, the accuracy of the solutions obtained from the three subroutines was comparable. For the rank deficient problems, less than 12 problems were included in the average ($m = 5$: 11 problems; $m = 10$: 9 problems; $m = 15$: 7 problems) since the subroutines did not obtain the correct solutions for some of the problems. Specifically, CLINF did not obtain the correct solution for 1 problem (for $m = 10$), CHEB did not obtain the correct solutions for 9 problems (1 for $m = 5$, 3 for $m = 10$, and 5 for $m = 15$), and CL8 did not obtain the correct solutions for 11 problems (1 for $m = 10$ and 10 for $m = 15$). For these problems, the subroutines determined the ranks to be larger than the correct ranks and the computed solutions had elements of $O(10^{16})$.

The results from the random constrained problems are given in Table 4.2. (Recall that CHEB solves only unconstrained problems and CL8 solves constrained problems with inequality constraints only.) The elements of $[A \quad G]$ and $\begin{bmatrix} y_0^* \\ y^* \end{bmatrix}$ are of magnitude $O(1)$. An initial value of $\mu = 1.0$ was used for CLINF. For the rank deficient problems with $m = 10$, 8 problems were included in the average, since CL8 did not obtain the correct solutions for 4 problems.

In Table 4.3, we show for $m = \text{rank} = 10$ that the number of iterations for CL8 depends upon the magnitude of y_0^*. The arrays A and y^* in these problems are the same as those from Table 4.1; only y_0^* has been changed. An initial value of $\mu = 1.0$ was used for CLINF.

TABLE 4.2
Average number of iterations and average CPU *time in milliseconds over 12 problems for each value of* m ($n = 10m$, $p = 0$, $q = m$)

m	rank	CLINF iterations	CLINF time	CL8 iterations	CL8 time
5	5	15.4	339	13.3	699
10	10	38.3	2013	33.9	3667
15	15	65.2	6190	52.6	9313
5	3	10.3	214	9.3	527
10	7	20.9	1069	24.9	2777

TABLE 4.3
Average number of iterations and average CPU *time in milliseconds over 12 problems for each order of magnitude of* y_0^* ($m = \text{rank} = 10$)

y_0^*	CLINF iterations	CLINF time	CHEB iterations	CHEB time	CL8 iterations	CL8 time
$O(10^2)$	32.5	1644	28.3	1545	11.1	1198
$O(10^1)$	30.2	1542	29.8	1637	15.7	1694
$O(10^0)$	32.3	1617	31.1	1705	24.6	2627
$O(10^{-1})$	37.3	1850	31.7	1731	29.5	3119
$O(10^{-2})$	37.6	1859	34.1	1855	33.9	3644

The number of iterations for CLINF varies with different initial values of μ. The variation follows no detectable pattern, but any reasonable initial value of μ will give a correct solution. We have found no a priori way of selecting the initial value of μ which leads to the fewest number of iterations.

The remainder of the tests we carried out were derived mainly from some data fitting problems given in [3]. They require the best polynomial l_∞ approximation of degree $m-1$ to various functions sampled over a finite set of values t_j. In each problem, $n = 101$ data points (t_j, c_j) were generated from some given function $f(t)$, where c_j represents $f(t_j)$, using $t = 0(0.01)1$. To test the behavior of the algorithms under study, we perturbed $c_j = f(t_j)$ by random numbers in the interval $[-\alpha, \alpha]$. That is, for $1 \leqq j \leqq 101$ $t_j = 0.01(j-1)$ and $c_j = f(t_j) + r_j$ where $-\alpha \leqq r_j \leqq \alpha$. By doing this we could adjust the norm of y_0^* to be $O(\alpha)$ in magnitude. The results are given in Tables 4.4 and 4.5. The five functions $f(t)$ used in Table 4.4 are $\sqrt{1+t}$, $\sin(\pi t/2)$, $\exp t$, $\log(1+t)$, and $\exp(-t^2/2)$. The function used in Table 4.5, with five different sets of parameters $\{a, b\}$, is

$$f(t) = \begin{cases} t/a & \text{for } 0 \leqq t \leqq a, \\ 1 & \text{for } a \leqq t \leqq b, \\ (1-t)/(1-b) & \text{for } b \leqq t \leqq 1. \end{cases}$$

For CLINF, an initial value of $\mu = 1.0$ was used for the problems from the first four rows of Table 4.4, and $\mu = 0.0001$ was used for the other problems. The same solutions were obtained from the three subroutines for all problems.

Most of the problems in which an initial value of $\mu = 0.0001$ was used for CLINF were those in which the components of y^* had magnitudes of $O(10^3)$ or $O(10^4)$. If an initial value of $\mu = 1.0$ is used for these problems, then the number of iterations

TABLE 4.4
Average number of iterations and average CPU *time in milliseconds over* 5 *problems for each value of* m ($n = 101$)

m	α	y_0^*	CLINF iterations	CLINF time	CHEB iterations	CHEB time	CL8 iterations	CL8 time
5	0	$O(10^{-4})$	13.6	415	11.4	480	14.8	1447
6	0	$O(10^{-5})$	14.6	507	12.4	564	30.8	2900
7	0	$O(10^{-6})$	19.0	734	13.8	652	29.4	2959
8	0	$O(10^{-7})$	19.6	835	15.4	787	36.6	3781
8	0.01	$O(10^{-2})$	21.6	906	18.2	915	20.2	2079
8	0.1	$O(10^{-1})$	19.8	835	17.4	876	16.2	1641

which CLINF takes is much greater. We observed in such cases that the slack variable became nonbasic early while some artificial variables became nonzero and remained basic until μ was reduced to about 10^{-3} or 10^{-4}. In all events, $\|y^*\|_\infty = O(10^k)$ implied that $\mu = O(10^{-k})$ was needed by CLINF (at least for $k = 3$ or 4). There is a reason for this. The components of y^* represent the optimal dual variables to problem (1.3), which we are solving by use of the penalty function given in (2.2). The theory behind this penalty approach (referred to in [9]) tells us that a suitable value of μ, one which would not have to be changed by CLINF, would be

$$\mu = \frac{1}{\max_i |y_i^*|}.$$

TABLE 4.5
Average number of iterations and average CPU *time in milliseconds over* 5 *problems for each value of* α ($m = 8$, $n = 101$)

α	y_n^*	CLINF iterations	CLINF time	CHEB iterations	CHEB time	CL8 iterations	CL8 time
0	$O(10^{-1})$	19.0	806	15.4	789	38.0	3892
0.01	$O(10^{-1})$	17.2	729	14.8	763	23.2	2329
0.1	$O(10^{0})$	21.2	898	18.8	943	18.8	1885

It is just this fact which is making its appearance in our observations.

The numerical results presented in this section show, for the test examples at least, that the number of iterations and CPU time for CLINF, using a reasonable initial choice of μ, is comparable to that for CHEB (the CPU time per iteration for CLINF is slightly less than that for CHEB). The number of iterations for CL8 depends upon the magnitude of the l_∞ error y_0^* and the "smoothness" of the vector c. This dependence works very much to the detriment of CL8, and it was decidedly inferior to CHEB and CLINF on many problems.

5. Summary and selected further comparisons. The algorithm which has been presented compares well, for unconstrained problems, with the one proposed by Barrodale and Philips [3], [4]. It represents an extension in functionality over that algorithm in that it has been constructed to handle linear constraints. The algorithm of this paper represents one natural approach (via the "dual" linear programming formulation, (1.3), to the l_∞ problem) for applying the material in [5], [9] to solve (1.1). The other approach (via the "primal" formulation, (1.2)) is clearly inferior without further developments, except possibly for large residual problems.

We have looked at some other algorithms for linear l_∞ data fitting which appear in the recent literature, comparing them from a mathemetical standpoint rather than a computational one. Three of interest are [1], [2], [8], and we include some observations below.

Armstrong and Kung [2] propose an algorithm with much the flavor of the one we have just described, though it is confined to the unconstrained case. Their discussion takes place in "primal space", i.e., in terms of the variables y_0, y (λ and β respectively in their terminology). They consider problem (1.2)—with E, G vacuous—in an interval-constraints format:

$$\begin{aligned} &\text{minimize} && e_1^T\begin{bmatrix} y_0 \\ y \end{bmatrix} \\ &\text{subject to} && c - ey_0 \leqq A^T y \leqq c + ey_0, \end{aligned} \tag{5.1}$$

and they deal with a basis of the form

$$\begin{bmatrix} 1, & \cdots, & 1 & 1, & \cdots, & 1 \\ a_{i_1}, & \cdots, & a_{i_l} & a_{i_{l+1}}, & \cdots, & a_{i_{m+1}} \end{bmatrix}, \tag{5.2}$$

and with a corresponding reference subsystem. We would deal with the same reference subsystem via the basis:

$$\begin{bmatrix} 1, & \cdots, & 1 & 1, & \cdots, & 1 \\ a_{i_1}, & \cdots, & a_{i_l} & -a_{i_{l+1}}, & \cdots, & -a_{i_{m+1}} \end{bmatrix}, \tag{5.3}$$

and via the dual variables $x_{i_1}, \cdots, x_{i_l}, x_{i_{l+1}}, \cdots, x_{i_m}$. In effect, this means that we work with the dual to (5.1), having split interval constraints into their positive and negative components.

If $x_{i_j} > 0 (1 \leqq j \leqq l)$, then Armstrong and Kung would observe that $y_0 = a_{i_j}^T y - c_{i_j}$, i.e., that the i_jth constraint of their reference subproblem is at its upper bound. If $x_{i_j} < 0 (1 \leqq j \leqq l)$, then the i_jth constraint of the reference subproblem is at its lower bound. The reverse (">0"$\leftrightarrow$"<0" on "upper" and "lower") would be true for $l+1 \leqq j \leqq m$.

Armstrong and Kung choose a nonreference, violated constraint from (5.1) at each step, while our criterion for selecting a nonbasic component of the problem is that $y_0 \geqq |a_s^T y - c_s|$ is violated, which amounts to the same thing. We proceed to change x so as to bring a_s into our reference system (basis) at the expense of some a_k, which will be removed. Implicitly our action will cause y_0, y to change, though we do not need to compute precisely how. Armstrong and Kung, on the other hand, determine how to change y_0, y, which implicitly causes a change in the dual variables x. Armstrong and Kung do not need to compute precisely what this change in x will be. As we change x, certain components of x_{i_j} may cross zero, which implicitly signals that an interval constraint $c_{i_j} - y_0 \leqq a_{i_j}^T y \leqq c_{i_j} + y_0$ has gone from one bound to another in the primal space. Armstrong and Kung account for this in the same manner as we do. Consequently, our line search (2.11) bears the character of step 4 through step 7 of their algorithm given in [2]. It is tempting to conjecture that our algorithm in the unconstrained case is effectively the one given by Armstrong and Kung, but with computations translated into the dual space. If this is so, our algorithm constitutes an extension of theirs to the constrained linear l_∞ data fitting problem.

In [1] Abdelmalek considers the special case of (1.1) given by:

$$\begin{aligned} &\text{minimize} && \|c - A^T y\|_\infty \\ &\text{subject to} && h_1 \leqq A^T y \leqq h_2, \end{aligned} \tag{5.4}$$

that is,

$$\text{(5.5)} \qquad \begin{aligned} &\text{minimize} && \|c - A^T y\|_\infty \\ &\text{subject to} && G^T y \geqq h, \end{aligned}$$

where

$$G = \begin{bmatrix} A^T \\ -A^T \end{bmatrix} \quad \text{and} \quad h = \begin{bmatrix} h_1 \\ -h_2 \end{bmatrix}.$$

This is converted to a format equivalent to (1.3):

$$\text{(5.6)} \qquad \begin{aligned} &\text{maximize} && [c^T \quad -c^T \quad h_1 T \quad h_2^T]x \\ &\text{subject to} && \begin{bmatrix} e^T & e^T & 0^T & 0^T \\ A & -A & A & -A \end{bmatrix} x = e_1 \\ &\text{and} && x \geqq 0, \end{aligned}$$

and this is then considered in the compressed format:

$$\begin{aligned} &\text{maximize} && \bar{c}^T x \\ &\text{subject to} && \bar{A}^T x = e_1 \\ &\text{and} && x \leqq 0. \end{aligned}$$

Then, with respect to a suitable indexing scheme for $\bar{A}$, Abdelmalek lays out relationships and makes observations similar (but not identical) to those we have laid out in § 3.

The discussion in [1] predates the work on penalty approaches to linear programming [5], [9]; hence our formulas in § 3 constitute the sort of generalization over the material in [1, § 2] that [5] constitutes over the classical simplex method. Similarly, the algorithm given in [1] constitutes the classical simplex method adjusted to the special structure of (5.6). Hence, Abdelmalek's algorithm requires a separate phase 1 and phase 2, and it is restricted to vertex-to-adjacent-vertex transit. In contrast, our algorithm (and that of [2]) are single-phase and make use of a line search to pass several vertices at each step.

The method of Barrodale and Phillips, in fact, also proceeds from the formulation of (1.1)—without constraints—in the format of (1.3). It, too, applies classical simplex techniques to the resulting linear program in a phase-1/phase-2 and vertex-to-vertex fashion. The method of [2] is reported to be somewhat faster than that of [3], [4], and we take heart at the similarities noted above between that approach and our own.

In [8] Cline proposes an algoithm for the unconstrained version of (1.1). A brief development of Cline's method is to be found in [6] which begins with the analog of (1.2):

$$\text{(5.7)} \qquad \begin{aligned} &\text{minimize} && e_1^T \begin{bmatrix} y_0 \\ y \end{bmatrix} \\ &\text{subject to} && \begin{bmatrix} e & A^T \\ e & -A^T \end{bmatrix} \geqq \begin{bmatrix} c \\ -c \end{bmatrix}, \end{aligned}$$

and compresses this linear problem into a piecewise linear problem of one lower dimension by expressing the variable y_0 as a function of y. It is noted in [6] that Cline's method does not appear to be competitive with the method of [3]. The evidence

is empirical, but the behavior of Cline's method is reminiscent of that which we have observed for CL8—it requires a great many more iterations on some data-fitting problems than do "direct" (or "dual") approaches.

REFERENCES

[1] N. N. ABDELMALEK, *The discrete linear restricted Chebyshev approximation*, BIT, 17 (1977), pp. 249–261.

[2] R. D. ARMSTRONG AND D. S. KUNG, *A dual method for discrete Chebyshev curve fitting*, Math. Programming, 19 (1980), pp. 186–199.

[3] I. BARRODALE AND C. PHILLIPS, *An improved algorithm for discrete Chebyshev linear approximation*, in Proc. Fourth Manitoba Conference on Numerical Mathematics, Univ. of Manitoba, Winnipeg, Canada, 1974, pp. 177–190.

[4] ———, *ALGORITHM 495—Solution of an overdetermined system of linear equations in the Chebyshev norm*, ACM Trans. Math. Software, 1 (1975), pp. 264–270.

[5] R. H. BARTELS, *A penalty linear programming method using reduced-gradient basis-exchange techniques*, Linear Algebra and Appl., 29 (1980), pp. 17–32.

[6] R. H. BARTELS, A. R. CONN AND C. CHARALAMBOUS, *On Cline's direct method for solving overdetermined linear systems in the l_∞ sense*, SIAM J. Numer. Anal., 15 (1978), pp. 255–270.

[7] R. H. BARTELS AND B. JOE, *On generating test data for discrete l_∞* problems, this Journal, to appear.

[8] A. K. CLINE, *A descent method for the uniform solution to overdetermined systems of linear equations*, SIAM J. Numer. Anal., 13 (1976), pp. 293–309.

[9] A. R. CONN, *Linear programming via a nondifferentiable penalty function*, SIAM J. Numer. Anal., 13 (1976), pp. 145–154.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0006 $01.25/0

A DIAGONAL MODIFICATION FOR THE DOWNDATING ALGORITHM*

ACHIYA DAX†

Abstract. Let U be an $n \times n$ upper-triangular matrix with unit diagonal, D an $n \times n$ diagonal positive-definite matrix and $\mathbf{z}$ an n-vector. The downdating problem is to compute the new triangular factors of the matrix $U^TDU - \mathbf{z}\mathbf{z}^T$. A common way to start the downdating process is to compute a vector $\mathbf{y}$ such that $U^TDU - \mathbf{z}\mathbf{z}^T = U^T(D - \mathbf{y}\mathbf{y}^T)U$. If $\mathbf{y}^TD^{-1}\mathbf{y} \leqq 1$ there are many suitable ways to carry out the downdating. This paper is concerned with the case when, theoretically, $U^TDU - \mathbf{z}\mathbf{z}^T$ should be positive (semi) definite but, because of computer rounding errors, $\mathbf{y}^TD^{-1}\mathbf{y} > 1$. In this case the modified algorithm replaces D by $D + T$ where $T = \text{diag}\{t_1, \cdots, t_n\}$ solves the following problem:

$$\text{minimize } \sum_{i=1}^{n} t_i$$

$$\text{subject to: } \mathbf{y}^T(D+T)^{-1}\mathbf{y} = 1 \text{ and } t_i \geqq 0 \text{ for } i = 1, \cdots, n.$$

The computation of T is simple and needs only a few operations. The elements of T satisfy $\sum_{i=1}^{n} t_i \leqq n|\lambda|$ where λ is the negative eigenvalue of $D - \mathbf{y}\mathbf{y}^T$. This ensures that the correction is not much larger than necessary. The case when U has fewer rows than columns is also discussed.

Key words. symmetric factorization, downdating algorithms

1. The downdating problem. Let U be an $m \times n$ UTUD matrix and let D be an $m \times m$ DPD matrix. (UTUD stands for upper triangular with unit diagonal while DPD stands for diagonal positive definite). Let $\mathbf{z}$ be an n-vector. The downdating problem is to compute the new triangular factors of $U^TDU - \mathbf{z}\mathbf{z}^T$. In other words we are looking for a new $m \times n$ UTUD matrix, U^* say, and a new $m \times m$ DPD matrix, D^* say, such that

$$U^{*T}D^*U^* = U^TDU - \mathbf{z}\mathbf{z}^T. \tag{1.1}$$

In this paper we consider the case when $U^TDU - \mathbf{z}\mathbf{z}^T$ has a negative eigenvalue. It is assumed that with exact arithmetic $U^TDU - \mathbf{z}\mathbf{z}^T$ would be positive (semi) definite and that the negative eigenvalue is due to rounding errors in previous computations. This problem arises in least squares calculations and also in some active set algorithms for quadratic programming (see Powell (1980)). Our aim is, therefore, to construct a factorization of the form

$$U^{*T}D^*U^* = U^TDU - \mathbf{z}\mathbf{z}^T + E, \tag{1.2}$$

where $\|E\|_F$, the Frobenius norm of the correction matrix, is as small as possible. Let η denote the negative eigenvalue of $U^TDU - \mathbf{z}\mathbf{z}^T$ and let $\mathbf{v}$, $\|\mathbf{v}\|_2 = 1$, denote the corresponding eigenvector. Then the rank-one matrix $E = -\eta\mathbf{v}\mathbf{v}^T$ solves the following problem: Minimize $\|E\|_F$ where E is such that $U^TDU - \mathbf{z}\mathbf{z}^T + E$ is a positive semidefinite matrix. In practice, however, the computation of η and $\mathbf{v}$ is usually too expensive. Thus, instead of looking for the "minimal" correction, we only require that $\|E\|_F$ is not much larger than $|\eta|$.

Let ε denote the relative precision of the computer arithmetic. The magnitude of the rounding errors in the basic downdating process is expected to be about $\varepsilon\|U^TDU\|$. (See Stewart (1979).) Therefore there is no need to insist that $\|E\|_F$ will be smaller than this quantity.

* Received by the editors September 18, 1981, and in revised form April 1, 1982.
† Hydrological Service, P.O. Box 6381, Jerusalem 91060, Israel.

At first we shall consider the case when U has an equal number of rows and columns, i.e., when $m = n$. In this case it is possible to compute an n-vector $\mathbf{y}$ such that

$$U^T D U - \mathbf{z}\mathbf{z}^T = U^T (D - \mathbf{y}\mathbf{y}^T) U. \tag{1.3}$$

If $\mathbf{y}^T D^{-1} \mathbf{y} \leqq 1$ then there are many suitable ways to carry out the downdating. (See Gill et al. (1974) and Fletcher and Powell (1974) for instance.) Difficulties arise when $\mathbf{y}^T D^{-1} \mathbf{y} > 1$, i.e., when $U^T D U - \mathbf{z}\mathbf{z}^T$ has a negative eigenvalue. Gill and Murray (1972) suggested that in this case the algorithm can be modified in one of the following ways:

1) $\mathbf{y}$ can be replaced by $\alpha \mathbf{y}$ where α is a parameter such that $\alpha^2 \mathbf{y}^T D^{-1} \mathbf{y} \leqq 1$, or:

2) The size of the diagonal elements of D can be increased.

The "composite t-method" of Fletcher and Powell (1974) follows the first approach and sets α such that $\alpha^2 \mathbf{y}^T D^{-1} \mathbf{y} = 1 - \varepsilon$. Later Powell (1978), (1981) observed that this approach may make $\|E\|_F$ unnecessarily large. Instead he suggested computing the triangular factors of $U^T D U - \mathbf{z}\mathbf{z}^T + \mathbf{x}\mathbf{x}^T$, where $\mathbf{x}\mathbf{x}^T$ is a small rank-one matrix that makes it positive definite. (A brief description of Powell's method is given in the last section.)

The new algorithm follows the second approach of Gill and Murray. We replace D by $D + T$, where $T = \operatorname{diag}\{t_1, \cdots, t_n\}$ solves the following problem:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} t_i \\ &\text{subject to: } \mathbf{y}^T (D + T)^{-1} \mathbf{y} = 1 \text{ and } t_i \geqq 0 \text{ for } i = 1, \cdots, n. \end{aligned} \tag{1.4}$$

A simple way to solve this problem is given in § 3. It is shown there that the elements of T satisfy

$$\sum_{i=1}^{n} t_i \leqq n |\lambda|, \tag{1.5}$$

where λ is the negative eigenvalue of $D - \mathbf{y}\mathbf{y}^T$. Section 4 gives the details of the basic downdating process. In § 5 we show that

$$|\lambda| \leqq |\eta| \, \|U^{-1}\|_2^2 \tag{1.6}$$

and that (1.5) together with (1.6) yield the bound

$$\|U^T T U\|_F \leqq n |\eta| \, \|U\|_2^2 \, \|U^{-1}\|_2^2. \tag{1.7}$$

In other words, if $\|U\|_2^2 \, \|U^{-1}\|_2^2$ is not much larger than one, then the norm of the correction matrix is not much larger than its least value.

The case when U has fewer rows than columns, i.e., when $m < n$, is discussed in § 6. Here it happens quite often, because of rounding errors, that the overdetermined system

$$U^T \mathbf{y} = \mathbf{z} \tag{1.8}$$

has no solution. We give a simple way to overcome this difficulty. Apart from this point the other parts of the algorithm are almost unchanged. Thus in practice the two cases are easily combined into one algorithm.

Solving (1.4) is not the only way to compute a diagonal matrix T such that $D + T - \mathbf{y}\mathbf{y}^T$ is positive semidefinite. This and other aspects of the downdating problem are briefly discussed in § 7.

2. The modified algorithm. In this section we describe the algorithm when $m = n$. The algorithm is composed of the following steps:

Step A: Solve the linear system $U^T\mathbf{y} = \mathbf{z}$.

Step B: Compute $\sigma = \sum_{i=1}^{n} y_i^2/d_i$.
If $\sigma > 1$ skip to step C; otherwise we have

$$U^TDU - \mathbf{z}\mathbf{z}^T = \tilde{U}^T(\tilde{D} - \tilde{\mathbf{y}}\tilde{\mathbf{y}}^T)\tilde{U}$$

where $\tilde{D} = \operatorname{diag}\{d_1, \cdots, d_n, 1\}$, $\tilde{\mathbf{y}} = (y_1, \cdots, y_n, \rho)^T$, $\rho = (1-\sigma)^{1/2}$ and $\tilde{U}$ is obtained from U by adding a row of zeros. Now the method of § 4 is used to transform $\tilde{U}^T(\tilde{D} - \tilde{\mathbf{y}}\tilde{\mathbf{y}}^T)\tilde{U}$ into $U^{*T}D^*U^*$.

Step C: Solve (1.4), then use the method of § 4 to transform $U^T(D + T - \mathbf{y}\mathbf{y}^T)U$ into $U^{*T}D^*U^*$. (Here U^* has only $n-1$ rows.)

3. The minimization problem. In this section we give a simple method for solving the following problem:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} t_i \\ &\text{subject to: } t_i \geqq 0 \text{ for } i = 1, \cdots, n \text{ and } \textstyle\sum_{i=1}^{n} y_i^2/(d_i + t_i) = 1. \end{aligned} \tag{3.1}$$

The parameters y_i and d_i are given. It is assumed that $d_i > 0$ for $i = 1, \cdots, n$ and

$$\sum_{i=1}^{n} y_i^2/d_i > 1.$$

Let $\mathbf{t}^* = (t_1^*, \cdots, t_n^*)^T$ denote the desired solution of (3.1). Then, since $\mathbf{t}^* \neq \mathbf{0}$, the gradients of the constraints which are active at $\mathbf{t}^*$ are linearly independent. Hence the necessary conditions for an optimal point imply that $\mathbf{t}^*$ must satisfy

$$\begin{aligned} &1 = \lambda_i + \frac{\theta y_i^2}{(d_i + t_i^*)^2}, \\ &\lambda_i \geqq 0, \quad t_i^* \geqq 0, \quad \lambda_i t_i^* = 0, \end{aligned} \qquad i = 1, \cdots, n.$$

(θ and λ_i, $i = 1, \cdots, n$, are the Lagrange multipliers corresponding to the equality and inequality constraints of (3.1).) It follows therefore that if $t_i^* > 0$ then $\lambda_i = 0$ and $\theta y_i^2/(d_i + t_i^*)^2 = 1$. This means that $\theta > 0$. With the definition

$$\mu = (1/\theta)^{1/2},$$

the above relations imply that if $|y_i|/d_i \leqq \mu$ then $t_i^* = 0$, and if $|y_i|/d_i > \mu$ then $|y_i|/(d_i + t_i^*) = \mu$. Thus, the solution is straightforward once μ is known.

For simplicity we shall assume now that the variables are ordered so that $|y_1|/d_1 \leqq |y_2|/d_2 \leqq \cdots \leqq |y_n|/d_n$. Then, in order to compute μ, define

$$\rho_k = \sum_{i=k+1}^{n} |y_i|, \qquad k = 0, 1, \cdots, n-1,$$

$$\sigma_k = \sum_{i=1}^{k} \frac{y_i^2}{d_i} - 1, \qquad k = 1, \cdots, n-1,$$

$$\tau_k = \sigma_k + \frac{|y_k|}{d_k}\rho_k, \qquad k = 1, \cdots, n-1.$$

With this notation it is easy to verify that if $\tau_k > 0$ then $\mu < |y_k|/d_k$, and if $\tau_k \leqq 0$ then $\mu \geqq |y_k|/d_k$. This way one can easily find the index k for which $|y_k|/d_k \leqq \mu < |y_{k+1}|/d_{k+1}$. This means that $t_i^* = 0$ for $i = 1, \cdots, k$ and $t_i^* > 0$ for $i = k+1, \cdots, n$. Furthermore, μ and $\mathbf{t}^*$ must satisfy

$$\frac{|y_i|}{d_i + t_i^*} = \mu \quad \text{for } i = k+1, \cdots, n,$$

and

$$\sigma_k + \sum_{i=k+1}^{n} \frac{y_i^2}{d_i + t_i^*} = 0.$$

Hence

$$\sigma_k + \mu \rho_k = 0,$$

which gives the value

$$\mu = \frac{-\sigma_k}{\rho_k}.$$

The required solution is, therefore,

$$t_i^* = \begin{cases} 0 & \text{for } i = 1, \cdots, k, \\ |y_i|/\mu - d_i & \text{for } i = k+1, \cdots, n. \end{cases}$$

(Further, if $\tau_i > 0$ for $i = 1, \cdots, n-1$, then $t_i^* = |y_i|\rho_0 - d_i$ for $i = 1, \cdots, n$.) We sum up the results by presenting a complete algorithm for solving (3.1):

a) Preliminary preparations:
   1) Set $\xi_i = |y_i|/d_i$ for $i = 1, \cdots, n$.
   2) Interchange the variables so that $\xi_n = \max\{\xi_j; j = 1, \cdots, n\}$.
   3) Set $\rho_n = 0$ and $\sigma_n = \sum_{i=1}^{n} |y_i|\xi_i - 1$.

b) For $k = n-1, n-2, \cdots, 1$ do as follows:
   1) Interchange the variables so that $\xi_k = \max\{\xi_j; j = 1, \cdots, k\}$.
   2) Set $\rho_k = |y_{k+1}| + \rho_{k+1}$,
      $\sigma_k = \sigma_{k+1} - |y_{k+1}|\xi_{k+1}$,
      $\tau_k = \sigma_k + \xi_k \rho_k$.
   3) If $\tau_k \leqq 0$ skip to d).

c) Set $\rho_0 = \rho_1 + |y_1|$, and $t_i^* = |y_i|\rho_0 - d_i$ for $i = 1, \cdots, n$.
   Reverse the effects of the interchanges and terminate.

d) Set $t_i^* = 0$ for $i = 1, \cdots, k$, $\nu = -\rho_k/\sigma_k$ and $t_i^* = \nu|y_i| - d_i$ for $i = k+1, \cdots, n$.
   Reverse the effects of the interchanges and terminate.

The above algorithm needs fewer than $\frac{1}{2}n^2$ comparisons. The number of multiplications and divisions is fewer than $5n$. Fortunately, when this algorithm is used by the downdating algorithm then only a few of the numbers $t_1^*, t_2^*, \cdots, t_n^*$ are usually nonzero (see Dax (1981)). In such a case the amount of work is, of course, much smaller than the above figures.

We shall finish this section by giving upper and lower bounds on the elements of $\mathbf{t}^*$. Let λ denote the negative eigenvalue of $D - \mathbf{y}\mathbf{y}^T$ where $D = \operatorname{diag}\{d_1, \cdots, d_n\}$

and $\mathbf{y}=(y_1, \cdots, y_n)^T$. Let $\mathbf{u}$ denote the corresponding unit eigenvector, i.e., $\mathbf{u}^T\mathbf{u}=1$ and $(D-\mathbf{y}\mathbf{y}^T)\mathbf{u}=\lambda\mathbf{u}$. Then λ and $\mathbf{u}$ also satisfy

$$\mathbf{u}=(D-\lambda I)^{-1}\mathbf{y}(\mathbf{y}^T\mathbf{u})$$

and

$$\mathbf{y}^T(D-\lambda I)^{-1}\mathbf{y}=1.$$

In other words, λ is the only negative root of the equation

$$\sum_{i=1}^{n}\frac{y_i^2}{d_i-\lambda}=1.$$

The last equality means that $(|\lambda|, |\lambda|, \cdots, |\lambda|)^T$ is a feasible point of the minimization problem (3.1). Hence it follows that

$$\sum_{i=1}^{n} t_i^* \leqq n|\lambda| \tag{3.2}$$

and

$$\left(\sum_{i=1}^{n} t_i^{*2}\right)^{1/2} \leqq \sum_{i=1}^{n} t_i^* \leqq n|\lambda|. \tag{3.3}$$

In order to derive a lower bound note that

$$\phi(\alpha)=\sum_{i=1}^{n}\frac{y_i^2}{d_i+\alpha}$$

is a monotonic decreasing function in the interval $[0, \infty)$. Hence the inequality

$$\sum_{i=1}^{n}\frac{y_i^2}{d_i+|\lambda|}=\sum_{i=1}^{n}\frac{y_i^2}{d_i+t_i^*}\geqq\sum_{i=1}^{n}\frac{y_i^2}{d_i+t_{\max}^*},$$

where

$$t_{\max}^*=\max\{t_i^*|i=1, \cdots, n\}$$

implies that

$$|\lambda|\leqq t_{\max}^*. \tag{3.4}$$

Therefore

$$|\lambda|\leqq\sum_{i=1}^{n} t_i \leqq n|\lambda|. \tag{3.5}$$

4. The basic downdating process. Let U be an $m\times n$ UTUD matrix, D an $m\times m$ DPD matrix, and $\mathbf{y}$ an m-vector such that $\mathbf{y}^T\mathbf{D}^{-1}\mathbf{y}=1$. In this section we describe one of the basic ways to compute an $(m-1)\times n$ UTUD matrix, U^* say, and an $(m-1)\times(m-1)$ DPD matrix, D^* say, that satisfy

$$U^T(D-\mathbf{y}\mathbf{y}^T)U=U^{*T}D^*U^*.$$

The method is due to Gill et al. (1974). It uses an $m\times m$ orthogonal matrix, Q, such that

$$QD^{-1/2}\mathbf{y}=\mathbf{e}_m,$$

where $\mathbf{e}_m$ is the mth column of the $m \times m$ unit matrix. Let V denote the $(m-1)\times n$ matrix which is composed of the first $m-1$ rows of $QD^{1/2}U$. Then Q is constructed such that V is an upper trapezoidal matrix. These properties of Q yield the relations

$$U^T(D-\mathbf{y}\mathbf{y}^T)U = U^T D^{1/2} Q^T (I - \mathbf{e}_m \mathbf{e}_m^T) Q D^{1/2} U = V^T V.$$

Hence U^* and D^* are given by the equality

$$D^{*1/2} U^* = V.$$

In practice Q is a product of $m-1$ Givens rotations. At the ith step, $i = m-1, m-2, \cdots, 1$, the ith and the mth rows of both $D^{1/2}(D^{-1}\mathbf{y})$ and $D^{1/2}U$ are multiplied from the left by a Givens rotation which annihilates the ith component of $D^{1/2}(D^{-1}\mathbf{y})$.

The use of $D^{1/2}(D^{-1}\mathbf{y})$ instead of $D^{-1/2}\mathbf{y}$ enables us to carry out the Givens transformations without square roots (see Gentleman (1973)). This way the ith step needs only $3(n-i+2)$ multiplications and divisions. The total number of multiplications and divisions is, therefore, $\frac{3}{2}m^2 + 3m(n-m) + O(m)$.

5. Error analysis. In this section we shall follow the notations of § 2. If $\sigma \leqq 1$ then the new algorithm coincides with the downdating algorithm studied by Stewart (1979). (The only difference is that Stewart considers Givens transformations that use square roots.) If $\sigma > 1$, the new algorithm replaces D by $D+T$ and then continues as before. The aim of this section is, therefore, to impose a bound on the size of the correction matrix $U^T T U$. The main tool of our analysis is the following lemma.

LEMMA. *Let G be an $m \times m$ symmetric matrix with eigenvalues $\xi_1 \geqq \xi_2 \geqq \cdots \geqq \xi_{m-1} \geqq \xi_m \neq 0$, $\xi_{m-1} > 0$ but ξ_m may be negative. Let R be an $m \times n$ matrix and let S denote the $m \times m$ matrix which is composed of the first m columns of R. Assume that S is an invertible matrix and let $\nu_1 \geqq \nu_2 \geqq \cdots \geqq \nu_m$ denote the nonzero eigenvalues of $R^T G R$. Then*

$$|\xi_m| \leqq |\nu_m| \, \|S^{-T}\|_2^2.$$

Proof. First consider the following *minimal correction* problem: minimize $\|E\|_F$, where E is such that $R^T G R + E$ is a symmetric positive semidefinite matrix of rank $m-1$. The optimal E is the rank-one matrix $-\nu_m \mathbf{y}\mathbf{y}^T$ where $\mathbf{y}$ is a unit eigenvector of $R^T G R$ that corresponds to ν_m. Since $\mathbf{y}$ is an eigenvector of $R^T G R$ there exists an m-vector $\mathbf{w}$ such that

$$R^T \mathbf{w} = \mathbf{y}.$$

Hence

$$\mathbf{w} = S^{-T} \hat{\mathbf{y}},$$

where $\hat{\mathbf{y}}$ is composed of the first m components of $\mathbf{y}$. Note that the equality

$$R^T G R - \nu_m \mathbf{y}\mathbf{y}^T = R^T (G - \nu_m \mathbf{w}\mathbf{w}^T) R$$

implies that $G - \nu_m \mathbf{w}\mathbf{w}^T$ is a symmetric positive semidefinite matrix of rank $m-1$. Now let us look again on the above minimal correction problem but with G replacing $R^T G R$. The solution of the new problem is the rank-one matrix $-\xi_m \mathbf{x}\mathbf{x}^T$ where $\mathbf{x}$ is a unit eigenvector of G which corresponds to ξ_m. This gives

$$|\xi_m| = \|-\xi_m \mathbf{x}\mathbf{x}^T\|_F \leqq \|-\nu_m \mathbf{w}\mathbf{w}^T\|_F = |\nu_m| \, \|\mathbf{w}\|_2^2 \leqq |\nu_m| \, \|S^{-T}\|_2^2$$

so the lemma is proved. Q.E.D.

Let us now derive a bound on $\|U^TTU\|_F$. As before let η denote the negative eigenvalue of $U^TDU - \mathbf{z}\mathbf{z}^T$ and let λ denote the negative eigenvalue of $D - \mathbf{y}\mathbf{y}^T$. From the basic properties of matrix norms we know that

$$\|U^TTU\|_F \leqq \|U^T\|_2 \|T\|_F \|U\|_2.$$

In § 3 we have proved that

$$\|T\|_F \leqq n|\lambda|$$

while the last lemma implies that

$$|\lambda| \leqq |\eta| \|U^{-1}\|_2^2.$$

This yields the bound

$$\|U^TTU\|_F \leqq n|\lambda| \|U\|_2^2 \leqq n|\eta| \|U\|_2^2 \|U^{-1}\|_2^2. \tag{5.1}$$

6. The case $m < n$. In this case the modified algorithm is carried out as follows:
Step A: Firstly solve the linear system

$$V^T\mathbf{y} = \mathbf{b}$$

where the matrix V is composed of the first m columns of U and the vector $\mathbf{b}$ is composed of the first m components of $\mathbf{z}$. Then compute the residual vector

$$\mathbf{r} = \mathbf{c} - W^T\mathbf{y}.$$

The matrix W is composed of the last $n-m$ columns of U while $\mathbf{c}$ is composed from the last $n-m$ components of $\mathbf{z}$. The components of $\mathbf{r}$ are denoted as r_j, $j = m+1, \cdots, n$.

Step B: Interchange columns, if necessary, so that

$$|r_{m+1}| = \max\{|r_j|; j = m+1, \cdots, n\}.$$

If $|r_{m+1}| \leqq \varepsilon \max\{d_i; i = 1, \cdots, m\}$, which is unlikely because of the rounding errors, we treat it as zero. In this case branch to step B of § 2. (But use m instead of n.) Otherwise we have the relation

$$U^TDU - \mathbf{z}\mathbf{z}^T = \hat{U}^T(\hat{D} - \hat{\mathbf{y}}\hat{\mathbf{y}}^T)\hat{U}$$

where $\hat{D} = \operatorname{diag}\{d_1, \cdots, d_m, 0\}$, $\hat{\mathbf{y}} = (y_1, \cdots, \mathbf{y}_m, r_{m+1})^T$, and $\hat{U}$ is an $(m+1)\times n$ UTUD matrix. The first m rows of $\hat{U}$ are those of U while the last row is

$$(0, 0, \cdots, 0, 1, r_{m+2}/r_{m+1}, r_{m+3}/r_{m+1}, \cdots, r_n/r_{m+1}).$$

Step C: Define $y_{m+1} = r_{m+1}$ and $d_{m+1} = r_{m+1}^2$. Then solve (3.1) where $m+1$ replaces n. The solution of (3.1) is denoted as $\hat{T} = \operatorname{diag}\{t_1, \cdots, t_{m+1}\}$. Finally use the method of § 4 to transform $\hat{U}^T(\hat{D} + \hat{T} - \hat{\mathbf{y}}\hat{\mathbf{y}}^T)\hat{U}$ into the desired form of $U^{*T}D^*U^*$.

The derivation of a bound on the correction matrix is done exactly as before. This yields the inequality

$$\|\hat{U}^T\hat{T}\hat{U}\|_F \leqq (m+1)|\eta| \|\hat{U}\|_2^2 \|\hat{V}^{-1}\|_2^2, \tag{6.1}$$

where $\hat{V}$ is composed of the first $m+1$ columns of $\hat{U}$.

Although the bounds (5.1) and (6.1) are similar, numerical experiments show that for $m < n$ the stability of the algorithm is slightly less satisfactory. It is possible, however, to improve the algorithm by using an additional procedure that reduces the norm of the correction matrix. This routine needs only $O(n)$ operations and helps greatly to slow down the build up of errors. (See Dax (1981).)

7. Discussion. For $m = n$ the new algorithm is a simple extension of the standard downdating scheme. The only difference is that if $\mathbf{y}^T D^{-1}\mathbf{y} > 1$ then D is replaced by $D + T$ where T solves (1.4). Solving (1.4) is not the only suitable way to compute a diagonal correction. Let $\mathbf{u}_i^T$, $i = 1, \cdots, n$, denote the ith row of U. Then the correction matrix satisfies

$$\|U^T T U\|_F = \left\| \sum_{i=1}^{n} t_i \mathbf{u}_i \mathbf{u}_i^T \right\|_F \leqq \sum_{i=1}^{n} \|t_i \mathbf{u}_i \mathbf{u}_i^T\|_F = \sum_{i=1}^{n} t_i w_i^2,$$

where

$$w_i = \|\mathbf{u}_i\|_2.$$

These relations imply that it is probably better to minimize $\sum_{i=1}^{n} t_i w_i^2$ instead of $\sum_{i=1}^{n} t_i$. Hence another way to compute T is to solve the following problem:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} t_i w_i^2 \\ &\text{subject to: } \mathbf{y}^T (D+T)^{-1}\mathbf{y} = 1 \text{ and } t_i \geqq 0 \text{ for } i = 1, \cdots, n. \end{aligned} \tag{7.1}$$

The algorithm for solving (7.1) is almost identical to that for solving (1.4). The solution of (7.1) satisfies

$$\|U^T T U\|_F \leqq \sum_{i=1}^{n} t_i w_i^2 \leqq \sum_{i=1}^{n} |\lambda| w_i^2 \leqq |\eta| \|U\|_F^2 \|U^{-1}\|_2^2. \tag{7.2}$$

Therefore, since $\|U\|_F^2 \leqq n\|U\|_2^2$, (7.2) is a better bound than (5.1). The price paid for this advantage is the extra $\frac{1}{2}n^2$ multiplications that are needed to compute the w_i's.

Another way to reduce the bound on the correction matrix is to minimize $\|T\|_F$. This way T solves the following problem:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^{n} t_i^2 \\ &\text{subject to: } \mathbf{y}^T (D+T)^{-1}\mathbf{y} = 1 \text{ and } t_i \geqq 0 \text{ for } i = 1, \cdots, n. \end{aligned} \tag{7.3}$$

An algorithm for solving (7.3) is given in Dax (1981). This algorithm is slightly more complicated than the algorithm for solving (1.4). The solution of (7.3) satisfies

$$\|T\|_F \leqq n^{1/2} |\lambda|. \tag{7.4}$$

Hence instead of (5.1) we have

$$\|U^T T U\|_F \leqq n^{1/2} |\lambda| \|U\|_2^2 \leqq n^{1/2} |\eta| \|U\|_2^2 \|U^{-1}\|_2^2. \tag{7.5}$$

Powell (1981) suggested a different way to compute the correction matrix. His algorithm computes a vector $\mathbf{w}$ such that $\mathbf{w}^T D^{-1}\mathbf{w} = 1$ and uses the method of § 4 to downdate the matrix

$$U^T (D - \mathbf{w}\mathbf{w}^T) U.$$

Then it computes a pair of vectors, $\mathbf{v}$ and $\mathbf{x}$ say, such that

$$(U^T \mathbf{w})(U^T \mathbf{w})^T - \mathbf{z}\mathbf{z}^T = \mathbf{v}\mathbf{v}^T - \mathbf{x}\mathbf{x}^T \quad \text{and} \quad \mathbf{v}^T \mathbf{x} = 0.$$

(The second equality makes $\|\mathbf{x}\|_2$ as small as possible.) Finally it updates the triangular factors of

$$U^T (D - \mathbf{w}\mathbf{w}^T) U + \mathbf{v}\mathbf{v}^T.$$

This updating needs the same number of operations as the downdating of $U^T(D - \mathbf{w}\mathbf{w}^T)U$. Our algorithm avoids the updating and therefore needs fewer operations. The choice of $\mathbf{w}$ is made in a way which ensures that $\mathbf{x}\mathbf{x}^T$, the correction matrix, satisfies the bound

$$\|\mathbf{x}\mathbf{x}^T\|_F \leqq |\eta| \|R\|_2^2 \|R^{-1}\|_2^2, \tag{7.6}$$

where R is an $n \times n$ UTUD matrix. The first m rows of R are those of U while the last $n-m$ rows are the last $n-m$ rows of the $n \times n$ unit matrix. This bound is a little better than the bounds on the diagonal corrections.

Acknowledgment. The author is greatly indebted to Professor M. J. D. Powell for having suggested the topic and for his generous help and guidance during this research. The author also wishes to thank the referees for their constructive suggestions.

REFERENCES

[1] A. DAX (1981), *Numerical experiments with new downdating algorithms*, DAMTP 1981/NA5, Univ. of Cambridge.

[2] R. FLETCHER AND M. J. D. POWELL (1974), *On the modification of LDL^T factorizations*, Math. Comp., 28, pp. 1067–1087.

[3] W. M. GENTLEMAN (1973), *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12, pp. 329–336.

[4] P. E. GILL, G. H. GOLUB, W. MURRAY AND M. A. SAUNDERS (1974), *Methods for modifying matrix factorizations*, Math. Comp., 28, pp. 505–535.

[5] P. E. GILL AND W. MURRAY (1972), *Quasi-Newton methods for unconstrained optimization*, J. Inst. Math. Appl., 9, pp. 91–108.

[6] M. J. D. POWELL (1978), *Downdating with row reduction*, DAMPT 78/NA3, Univ. of Cambridge.

[7] ———, *An upper triangular matrix method for quadratic programming*, DAMPT 1980/NA4, Univ. of Cambridge. Presented at Nonlinear Programming 4, Madison, WI July, 1980.

[8] ——— (1981), *Downdating the Cholesky factorization*, DAMPT 1981/NA2, Univ. of Cambridge.

[9] G. W. STEWART (1979), *The effects of rounding error on an algorithm for downdating a Cholesky factorization*, J. Inst. Math. Appl, 23, pp. 203–213.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0007 $01.25/0

NUMERICAL STUDY OF INCOMPRESSIBLE FLOW ABOUT IMMERSED ELASTIC BOUNDARIES*

RAUL MENDEZ†

Abstract. In this work we study the interaction between a two-dimensional, incompressible inviscid fluid and a one-dimensional tense elastic boundary. The boundary exerts forces on the fluid but its motion is determined from the motion of the fluid. The crux of our method lies in the representation of the point-forces into which the boundary stresses are discretized using Peskin's link formalism via a collection of vortex pairs of appropriate opposite strengths (vortex dipoles). Euler's equations are then integrated: the vorticity at the boundary determines the fluid motion everywhere. The method is applied to follow the oscillatory motion of an immersed tense circular boundary which at $t = 0$ has been deformed into an ellipse.

Key words. vortex dipole algorithm, point-forces, vortex pairs

1. Introduction. In this work we introduce a numerical technique designed to calculate the interaction between a tense, one-dimensional elastic boundary and an incompressible, inviscid two-dimensional fluid. The boundary exerts forces on the fluid while the fluid's motion determines the motion of the boundary.

The method of solution introduced here relies on previous work of Peskin [1], [2] to derive the forces felt by the fluid from the position of the boundary. The crux of our method consists of the representation of these boundary forces by a vortex dipole layer of appropriate strength. (Because of this representation we call our method the *vortex dipole algorithm.*) Euler's equation, written in the vorticity transport form, can then be integrated numerically to yield the fluid's velocity everywhere. We assume the initial configuration to be an ellipse with the fluid at rest and assume the equilibrium configuration is a circle (of the same area). The ellipse overshoots the circular configuration, however, and oscillates between being elongated horizontally and vertically.

The method of solution proposed in this work applies only to inviscid fluid problems, although it can be modified and adapted to viscous flow problems. Indeed, C. Peskin, A. Wolfe and M. McCracken have used a modified vortex dipole algorithm as a component of a vortex scheme, based on Chorin's vortex scheme, [3] in their mitral flow [4] and aortic sinus flow [5], [6] calculations, all at high Reynolds number. The vortex dipole algorithm as described here is a very efficient scheme: since no tangential stresses can be applied to the fluid, the boundary forces must be along the normal to the boundary; this implies that the vortex dipole layer used to represent the boundary stresses must have dipole moment tangent to the boundary. An important consequence of this is that the vortex dipoles (vortex pairs with opposite strengths), representing the point forces into which the boundary forces are discretized, will travel with the boundary. We shall show below that this implies that the total number of dipoles remains constant throughout the calculation. Unfortunately, in the viscous case studied by Peskin and his coworkers [4], [5], [6], the boundary forces need not be normal to the boundary and the vortex dipoles need not be tangential to the boundary. Thus the vortex pairs are swept into the fluid and the scheme's efficiency is lost since the number of dipoles needed will increase with time. (Peskin has introduced in [6] a vortex merging technique in an attempt to handle this difficulty.) We shall investigate in a future paper the application of the vortex dipole algorithm, introduced here, to the oscillations of the aorta, i.e., to aortic sounds, and we shall

* Received by the editors March 17, 1981, and in revised form March 12, 1982.
† Naval Postgraduate School, Monterey, California 93940.

assume this flow to be inviscid. A comparison with frequency estimates of this oscillatory motion derived by Peskin in [7] is also planned, and a preliminary version of this work has been presented in [8].

2. Equations of motion. The fluid is assumed to be two-dimensional, incompressible and inviscid. The fluid's equation of motion in nondimensional form is as follows:

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla \mathbf{p} + \mathbf{F} \tag{1}$$

where $\mathbf{u} = \mathbf{u}(\xi, t)$ is the velocity of the fluid particle located at ξ at time t, $\mathbf{p} = \mathbf{p}(\xi, t)$ is the pressure and $\mathbf{F} = \mathbf{F}(\xi, t)$ is the external force density; the symbol ∂_t denotes differentiation with respect to time. Since we seek a vortex approximation to (1), we write this equation in the vorticity transport form (2), which is obtained by applying the curl operator to both sides of (1) and where $w = \nabla \times \mathbf{u} \cdot \hat{z}$ is the vorticity function, $\hat{z}$ being a unit vector normal to the plane of the flow. Thus

$$\partial_t w + (\mathbf{u} \cdot \nabla)\mathbf{w} = \nabla \times \mathbf{F}. \tag{2}$$

The incompressibility of the fluid requires

$$\nabla \cdot \mathbf{u} = 0. \tag{3}$$

The boundary may slip tangent to the fluid, but its normal motion must agree with that of the fluid. Hence

$$\partial_t \mathbf{x} = \mathbf{u}(\mathbf{x}, t) \cdot \boldsymbol{\eta} \tag{4}$$

where $\mathbf{x} = \mathbf{x}(s, t)$ denotes the position at time t of the material boundary point which was labelled with the parameter s at time $t = 0$, and $\boldsymbol{\eta}$ is the normal unit vector at $\mathbf{x}(s, t)$. The notation $\mathbf{x}(\cdot, t)$ will be used to denote the position of the boundary at time t (we shall often refer to the boundary's position simply as the boundary). Thus a Lagrangian description will be used for the boundary while the fluid will be described by using Eulerian coordinates.

To derive our next equation, we observe that the forces $\mathbf{f}(s, t)$ exerted by the boundary at $\mathbf{x}(s, t)$ can be shown to depend solely on the boundary's configuration $\mathbf{x}(\cdot, t)$. This is a consequence of Newton's second law and our zero-mass boundary assumption (Peskin [7]). Thus we can write:

$$\mathbf{f}(\cdot, t) = S(\mathbf{x}(\cdot, t)), \tag{5}$$

where S is an operator describing the properties of the boundary to be investigated presently. Define $T(s, t)$, $\boldsymbol{\tau}(s, t)$ and $m(s)$ to be the tension, unit tangent vector and boundary mass density at the boundary point $\mathbf{x}(s, t)$. Let us state Newton's second law for a segment $\mathbf{x}(s, t)$, $s \in [a, b]$ of the boundary:

$$\int_a^b m(s) \frac{\partial^2 x}{\partial t^2}\, ds = T(s, t)\boldsymbol{\tau}(s, t)\Big|_a^b - \int_a^b \mathbf{f}(s, t)\, ds, \tag{6}$$

$$0 = \int_a^b \frac{\partial}{\partial s}(T(s, t)\boldsymbol{\tau}(s, t) - \mathbf{f}(s, t))\, ds; \tag{7}$$

the left-hand side in (7) is zero because of our zero-mass assumption. Thus:

$$\begin{aligned} \mathbf{f}(s, t) &= \frac{\partial}{\partial s}(T(s, t)\boldsymbol{\tau}(s, t)), \\ \mathbf{f}(s, t) &= T(s, t)\frac{\partial}{\partial s}\boldsymbol{\tau}(s, t) + \frac{\partial T}{\partial s}(s, t)\boldsymbol{\tau}(s, t). \end{aligned} \tag{8}$$

Since the fluid is inviscid $\mathbf{f}(s, t)$ cannot have a component tangent to the boundary; thus

$$\frac{\partial T}{\partial s}(s, t) = 0, \tag{9}$$

$$\mathbf{f}(s, t) = T(s, t)\frac{\partial}{\partial s}\boldsymbol{\tau}(s, t). \tag{10}$$

Thus, $\mathbf{f}(s, t)$ is normal to the boundary and T is a function f t alone; for a fixed time t the tension T is constant along the boundary. We shall see that (9) gives rise to a numerical check on the calculation.

Our last equation of motion describes the force density per unit area, $\mathbf{F}(\boldsymbol{\xi}, t)$, where $\boldsymbol{\xi}$ is the location of a fluid particle. As mentioned above, $\mathbf{F}$ is a distribution supported on the boundary $\mathbf{x}(\cdot, t)$ with kernel $\delta(\boldsymbol{\xi} - \mathbf{x}(s, t))$

$$\mathbf{F}(\boldsymbol{\xi}, t) = \int_{B_0} \mathbf{f}(s, t)\delta(\boldsymbol{\xi} - \mathbf{x}(s, t))\, ds, \tag{11}$$

where B_0 is the interval of s values used to describe $\mathbf{x}(\cdot, t)$. The plausibility of (11) becomes evident by checking that the force $\mathbf{F}_1$ exerted by the boundary on a region R of the fluid depends only on that part of the boundary contained within the region R.

$$\mathbf{F}_1 = \int_R \mathbf{F}(\boldsymbol{\xi}, t)\, d\boldsymbol{\xi} = \int_R d\boldsymbol{\xi} \int_{B_0} \mathbf{f}(s, t)\delta(\boldsymbol{\xi} - \mathbf{x}(s, t))\, ds \tag{12}$$

$$= \int_{\mathbf{x}(s,t)\in R} \mathbf{f}(s, t)\delta(\boldsymbol{\xi} - \mathbf{x}(s, t))\, ds. \tag{13}$$

Equation (13) might mistakenly be interpreted to mean that the forces act only locally. To see that the forces are felt instantaneously throughout the fluid one should apply the divergence operator to both sides of Euler's equation (1) and observe that $\nabla \cdot \mathbf{F}$ acts as a source for the pressure P in the resulting equation.

3. Discretization. We introduce in this section the discretization on which our numerical methods are based.

At the outset we shall discretize the label s, used to parametrize the boundary, into a collection of tags $k = 1, \cdots, N$ and single out N points $\{\mathbf{x}_k^0\}_{k=1}^N$ from the continuum $\mathbf{x}(s, 0)$. The boundary $\mathbf{x}(s, 0)$ will be partitioned into N segments of equal length and the collection $\{\mathbf{x}_k^0\}$, $k = 1, \cdots, N$ will be chosen to be the midpoints of these N subdivisions. The time will march in steps of fixed length Δt. That is, we shall be interested in approximate solutions to the system (1)–(4), (10) only a times $t = n\Delta t$, $n = 1, 2, \cdots$. These approximations at times $t = n\Delta t$ will be denoted by means of the superscript n, e.g., $\mathbf{x}_k^n$ denotes the approximate location of the kth boundary marker at time $t = n\Delta t$. Approximations to the forces $\mathbf{f}(s, t)$ (10) applied at the points $\{\mathbf{x}_k^n\}_{k=1}^N$ will be denoted $\mathbf{f}_k^n$ and will be calculated from the boundary's position $\{\mathbf{x}_k^n\}_{k=1}^N$ by using a system of rodlike elastic links introduced by Peskin in [1] and termed by him the link formalism. Each link $l_{k,k+1}$ connects the pair of consecutive boundary points $\mathbf{x}_k^n$ and $\mathbf{x}_{k+1}^n$; the tensions along these links will be approximated by functions T_k^n; this will be the subject of § 4. In § 5 we shall discuss how to apply the forces $\mathbf{f}_k^n$ to the fluid through vortex pairs of strengths $\pm d_k^n$ and locations $\mathbf{x}_{ki}^n$, $k = 1, \cdots, N$; $i = 1, 2$, thus discretizing the vorticity functions W into

$$w^n(\boldsymbol{\xi}) = \sum_{k=1}^{N} \sum_{i=1}^{2} w_{ki}^n \delta(\boldsymbol{\xi} - \mathbf{x}_{ki}^n).$$

We shall see in § 6 that

$$w_{ki}^n = (-1)^i \sum_{j=0}^{n} d_k^j.$$

The external force density $\mathbf{F}(\boldsymbol{\xi}) = \int_{\mathbf{x}(\cdot,\tau)} \mathbf{f}(s,t)\delta(\boldsymbol{\xi}-\mathbf{x}(s,t))\,ds$ will be discretized into the approximation $\mathbf{F}^n(\boldsymbol{\xi}) = (\gamma/N)\sum_{k=1}^{N} \mathbf{f}_k^n \delta(\boldsymbol{\xi}-x_k^n)$, where γ is the length of the boundary at $t=0$. We shall set $\gamma = 1$. Section 6 describes how, by using the accumulated vorticity at the boundary, we approximately solve Euler's equation.

4. Calculation of boundary forces. We have seen that the boundary forces $\mathbf{f}(s,t)$ can be derived from the location of the boundary configuration via

$$\mathbf{f}(s,t) = T(t)\frac{\partial}{\partial s}(\boldsymbol{\tau}(s,t)), \tag{10}$$

where $T(t)$ is the tension and $\boldsymbol{\tau}(s,t)$ is the tangent unit vector at $\mathbf{x}(s,t)$. As noted above, the approximation to $\mathbf{f}(s,t)$ at the boundary markers $\mathbf{x}_k^n$ will be denoted $\mathbf{f}_k^n$. To calculate $\mathbf{f}_k^n$ we used Peskin's link formalism. This technique will be described only as it applies to the present test problem boundary, which we assume to be a simple closed curve. As seen above, the collection of markers $\{\mathbf{x}_k^n\}_{k=1}^N$ tracks the motion of the boundary; we shall presently connect each consecutive pair $\mathbf{x}_k^n, \mathbf{x}_{k+1}^n$; $k = 1, \cdots, N$, $(\mathbf{x}_{N+1}^n = \mathbf{x}_1^n)$ by means of a rodlike elastic link $l_{k,k+1}$. Notice that because our boundary is a simple closed curve, then each boundary point $\mathbf{x}_k^n$ is connected only to two boundary points, namely to its neighbors $\mathbf{x}_{k-1}^n$ and $\mathbf{x}_{k+1}^n$, by links $l_{k-1,k}$ and $l_{k,k+1}$ respectively. Since the location of the markers changes with time, tensions T_k^n will be induced along the links $l_{k,k+1}$. We shall assume that the functions T_k^n describe the elastic properties of our links, which simulate those of a rubber band (our links resist stretching but do not oppose bending). Thus:

$$T_k^n = \begin{cases} c(r_{k,k+1}^n - r^0), & r_{k,k+1} > r^0, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

where C is a constant we shall call the stiffness, $r_{k,k+1}^n = \|\mathbf{x}_k^n - \mathbf{x}_{k+1}^n\|$, $\|\cdot\|$ denotes the Euclidean norm and r^0 denotes the length of the link $l_{k,k+1}$ at $t=0$. Each force $\mathbf{f}_k^n$ can now be obtained as the resultant of the tensions acting along the links $l_{k,k+1}$ and $l_{k,k-1}$ ending at $\mathbf{x}_k^n$. Let now $\hat{a}_{ij}$ denote a unit vector in the direction of $\mathbf{x}_j^n - \mathbf{x}_i^n$. Then

$$\mathbf{f}_k^n = T_{k-1}^n \hat{a}_{k,k-1}^n + T_k^n \hat{a}_{k,k+1}^n. \tag{15}$$

One can see that (15) is indeed an approximation to (10), as follows: Let $\hat{a}_{k,k-1} = -\boldsymbol{\lambda}$. Then

$$\mathbf{f}(s,t) = \frac{\partial}{\partial s}[T\boldsymbol{\lambda}] \cong \frac{1}{\Delta s} T\boldsymbol{\lambda}\Big|_{s-\Delta/2}^{s+\Delta/2},$$

and

$$\begin{aligned} T_k^n \hat{a}_{k,k+1}^n &\cong (T\boldsymbol{\lambda})\left(s + \frac{\Delta s}{2}\right), \\ T_k^n \hat{a}_{k,k-1}^n &\cong (-T\boldsymbol{\lambda})\left(s - \frac{\Delta s}{2}\right). \end{aligned} \tag{16}$$

5. The vortex dipole algorithm. A spoon immersed in a cup of inviscid fluid induces, when pushed by a force parallel to the cup's surface, a vortex pair of opposite strengths, i.e., a vortex dipole, which is visible on the surface of the fluid. This

phenomenon renders intuitively plausible our representation of a point force via a vortex dipole. We mention at the outset that the momentum change $\Delta\mathbf{p}_k^n = \mathbf{f}_k^n \Delta t$, where $k = 1, \cdots, N$, induced by a point force acting at $\mathbf{x}_k^n$ can be imparted to the fluid by a vortex dipole, i.e., by a pair of vortices of appropriate opposite strengths $\pm d_k^n$. Although the total momentum of a point vortex is zero, that of a vortex dipole is nonzero and its magnitude is directly proportional to the vortices' strengths d_k^n, and to the distance $2h$ separating them. To establish this we proceed as follows: Assume temporarily, to simplify the notation, that the vortices which make up the dipole have locations $(0, (-1)^i y_0)$ and strengths $(-1)^i d$, where $y_0 > 0$, $d > 0$ and $i = 1, 2$. Let $\mathbf{q} = (q_1, q_2)$ be the total momentum associated with the dipole, i.e., $\mathbf{q} = \iint_\Pi \mathbf{v}(x, y)\, dx\, dy$, where $\mathbf{v}(x, y) = (v_1(x, y), v_2(x, y))$ is the velocity field induced at (x, y) by the dipole and Π is the plane. Thus, $\mathbf{v} = \mathbf{w}^1 + \mathbf{w}^2$, where $\mathbf{w}^i = (w_1^i, w_2^i)$ is the velocity field due to the dipole's ith vortex, $i = 1, 2$. Because of symmetry, $\mathbf{q}$ is normal to the axis of the dipole, i.e., $\mathbf{q}_2 = 0$. Notice now that

$$w_1^1(x, y) = \frac{1}{2\pi}\left[\frac{-d(y - y_0)}{x^2 + (y - y_0)^2}\right], \qquad w_1^2(x, y) = \frac{1}{2\pi}\left[\frac{d(y + y_0)}{x^2 + (y + y_0)^2}\right].$$

Thus

$$\int_{-\infty}^{\infty} w_1^1(x, y)\, dx = \begin{cases} -\dfrac{d}{2}, & y > y_0, \\ \dfrac{d}{2}, & y < y_0, \end{cases}$$

(17)

$$\int_{-\infty}^{\infty} w_1^2(x, y)\, dx = \begin{cases} \dfrac{d}{2}, & y > -y_0, \\ -\dfrac{d}{2}, & y < -y_0, \end{cases}$$

and

$$q_1 = \iint_\Pi v_1\, dx\, dy = \iint_\Pi (w_1^1 + w_1^2)\, dx\, dy = \frac{d}{2}\int_{-\infty}^{\infty} dy \left\{\begin{bmatrix} -1, & y > y_0 \\ 1, & y < y_0 \end{bmatrix} - \begin{bmatrix} 1, & y > -y_0 \\ -1, & y < -y_0 \end{bmatrix}\right\}$$
$$= 2dy_0.$$

Thus, the exchange of a point force for a vortex dipole will maintain the momentum balanced as long as

$$\|\Delta\mathbf{p}_k^n\| = \|\mathbf{f}_k^n\| \Delta t = 2d_k^n h. \tag{18}$$

The vortices in each dipole are located at

$$\mathbf{x}_{ki}^n = \mathbf{x}_k^n + (-1)^i h \boldsymbol{\tau}_k^n \tag{19}$$

with $\boldsymbol{\tau}_k^n$ being a unit vector normal to $\mathbf{f}_k^n$ and h being a numerical parameter. The formal exchange between point forces and vortex dipoles can be established by comparing the distributions associated with each; see [4].

Thus, the term $\Delta t \nabla \times \mathbf{F} \cdot \hat{z}$, approximated via

$$\Delta t \nabla \times \mathbf{F}^n \cdot \hat{z} = \frac{\Delta t}{N} \sum_{k=1}^{N} \nabla \times \{\mathbf{f}_k^n \delta(\boldsymbol{\xi} - \mathbf{x}_k^n)\} \cdot \hat{z},$$

can be represented by the distribution

$$\sum_{k=1}^{N} d_k^n[\delta(\boldsymbol{\xi}-\mathbf{x}_{k1}^n)-\delta(\boldsymbol{\xi}-\mathbf{x}_{k2}^n)], \tag{20}$$

which is associated with the collection of vortex dipoles representing the point forces.

6. Numerical solution of Euler's equation. Since the fluid is inviscid, vorticity travels with material points. We shall assume that the motion of the two vortices in each dipole will be close to the motion of the boundary marker which is initially at the center of the dipole. Since vorticity is confined to the boundary and dipole moments are tangential to the boundary, it follows that the motion of the vortices is the same as that of the boundary (vorticity is a fluid marker in an inviscid fluid). The tangential motion of vortex dipoles and boundary points is less clearly defined because the tangential velocity is not continuous across the boundary. In the present work, we have moved the boundary points as fluid markers in the computed velocity of the vortices, and we have moved the vortex dipoles along the boundary points where they were generated. One check on the validity of this procedure is that the tension around the boundary remains fairly constant, since this is the condition that actually determines the tangential motion of the material points of the boundary. (In future work we plan to use this condition directly.)

Our assumption above permits us to move our vortex dipoles by displacing them with the boundary point at their center, while maintaining the dipoles' axes tangent to the boundary. This approximation makes the vortex dipole algorithm a very efficient scheme: only N vortex dipoles will be required at each time step to represent the boundary stresses. Indeed, we shall not need to create a new dipole at $\mathbf{x}_k^n$ to represent $\mathbf{f}_k^n$; we shall solely update the strengths w_{ki}^n of the vortices at $\mathbf{x}_{ki}^n$, $i=1,2$. To accomplish this we write Euler's equation (1) in Lagrangian form:

$$D_t w = (\nabla \times \mathbf{F}) \cdot \hat{z}, \tag{21}$$

where D_t denotes the total derivative with respect to time. A first order approximation yields

$$w^n = w^{n-1} + \Delta t(\nabla \times \mathbf{F}^n \cdot \hat{z}). \tag{22}$$

Approximation (22) together with (18) reveals how to update the strengths of w_{ki}^n of each dipole's vortices:

$$w_{ki}^n = w_{ki}^{n-1} + (-1)^i d_k^n, \qquad i=1,2, \quad k=1,\cdots,N, \tag{23}$$

where $d_k^n = (\Delta t/2h)\|\mathbf{f}_k^n\|$. This equation yields by induction

$$w_{ki}^n = (-1)^i \sum_{j=0}^{n} d_k^j. \tag{24}$$

The fluid velocity $\mathbf{u}^n(\boldsymbol{\xi})$, a continuous variable, can now be computed by superimposing the fields $\mathbf{u}_{ki}^n(\boldsymbol{\xi})$, associated with each vortex, where

$$\mathbf{u}_{ki}^n(\boldsymbol{\xi}) = \frac{w_{ki}^n(\boldsymbol{\xi}-\mathbf{x}_{ki}^n)\times\hat{z}}{2\pi\|\boldsymbol{\xi}-\mathbf{x}_{ki}^n\| \max\{\|\boldsymbol{\xi}-\mathbf{x}_{ki}^n\|, \sigma\}}, \tag{25}$$

with $\hat{z}$ normal to the plane and σ a cutoff to be determined numerically; notice that (25) renders the speed constant within σ (Chorin [3]). Thus

$$\mathbf{u}^n(\boldsymbol{\xi}) = \sum_{k=1}^{N}\sum_{i=1}^{2} \mathbf{u}_{ki}^n(\boldsymbol{\xi}). \tag{26}$$

7. Summary of numerical methods. In this section we summarize the equations that allow us to solve the system (1)–(4), (10). As mentioned above, the time marches in steps; at the beginning of the nth time step, the locations of the boundary markers $\mathbf{x}_k^n$ are known, and in addition, so are the strengths w_{ki}^n, $i = 1, 2$, of each of the vortices centered about $\mathbf{x}_k^n$. The locations $\mathbf{x}_{ki}^n$ of these vortex pairs will be determined presently.

We first derive the forces $\mathbf{f}_k^n$ at the boundary points $\mathbf{x}_k^n$, via

$$\mathbf{f}_k^n = T_{k-1}^n \hat{a}_{k,k-1} + T_k^n \hat{a}_{k,k+1}, \tag{15}$$

where the unit vectors $\hat{a}_{k,k\pm1}$ denote directions parallel to $\mathbf{x}_{k\pm1}^n - \mathbf{x}_k^n$, and the functions T_k^n, T_{k+1}^n describe tensions along these links $l_{k-1,k}$ and $l_{k,k+1}$.

Each force $\mathbf{f}_k^n$ is applied to the fluid by updating the strengths w_{ki}^n of the vortices at each dipole by amounts $(-1)^i d_k^n$, $i = 1, 2$, where $d_k^n = (\Delta t/2h)\|\mathbf{f}_k^n\|$. From this it follows that

$$w_{ki}^n = w_{ki}^{n-1} + (-1)^i d_k^n, \qquad i = 1, 2. \tag{23}$$

The location $\mathbf{x}_{ki}^n$ of each vortex pair is then set by:

$$\mathbf{x}_{ki}^n = \mathbf{x}_k^n + (-1)^i h \boldsymbol{\tau}_k^n \tag{19}$$

where $\boldsymbol{\tau}_k^n$ is a unit vector satisfying $\boldsymbol{\tau}_k^n \cdot \mathbf{f}_k^n = 0$ (as noted above $\boldsymbol{\tau}_k^n$ is nearly tangent to the boundary at $\mathbf{x}_k^n$).

The fluid's velocity $\mathbf{u}^n(\boldsymbol{\xi})$ is then calculated by adding up the fields $\mathbf{u}_{ki}^n(\boldsymbol{\xi})$ due to the kith vortex, $i = 1, 2$ and $k = 1, \cdots, N$. Now $\mathbf{u}_{ki}^n(\boldsymbol{\xi})$ is given by

$$\mathbf{u}_{ki}^n(\boldsymbol{\xi}) = \frac{w_{ki}^n(\boldsymbol{\xi} - \mathbf{x}_{ki}^n) \times \hat{z}}{2\pi \|\boldsymbol{\xi} - \mathbf{x}_{ki}^n\| \max\{\|\boldsymbol{\xi} - \mathbf{x}_{ki}^n\|, \sigma\}} \tag{25}$$

($\hat{z}$ is normal to the plane of the flow). Notice that because of (25), nearby vortices don't induce large velocities on one another. The velocity field $\mathbf{u}^n(\boldsymbol{\xi})$ is thus

$$\mathbf{u}^n(\boldsymbol{\xi}) = \sum_{k=1}^{N} \sum_{i=1}^{2} \mathbf{u}_{ki}^n(\boldsymbol{\xi}). \tag{26}$$

The configuration of the boundary is now advanced one step through the first order approximations:

$$\mathbf{x}_k^{n+1} = \mathbf{x}_k^n + \Delta t \mathbf{u}^n(\mathbf{x}_k^n).$$

The time step is thus completed.

8. Numerical results on test problems. The method of solution described above was used to calculate the interaction for an incompressible inviscid fluid surrounding a tense, elastic one-dimensional circular boundary which at time $t = 0$ has the form of an ellipse. The initial velocities of both boundary and fluid were assumed to be zero. The vortex dipole algorithm allowed us to calculate the oscillations in time of the boundary, while respecting the physical properties of both boundary and fluid; care was taken to verify that tension remained constant along the boundary at each time step (see Fig. 1) as well as to check that the algorithm did not violate the incompressibility of the fluid (Fig. 2). Figure 3 depicts the motion of the boundary for selected time steps. Numerical parameters were as follows: Twenty-four points were used to represent the configuration of the boundary, which at the start of the calculation was taken to be an ellipse with major and minor axes of length $a_1 = 1.5$,

$b_1 = .5$, respectively. It was observed that the phenomenon of numerical instability developed for values of $\Gamma = s(\Delta t)^2$ satisfying

$$\Gamma > 4 \times 10^{-4} \tag{27}$$

while the calculation appeared to be stable for

$$\Gamma \leqq 4 \times 10^{-4}. \tag{28}$$

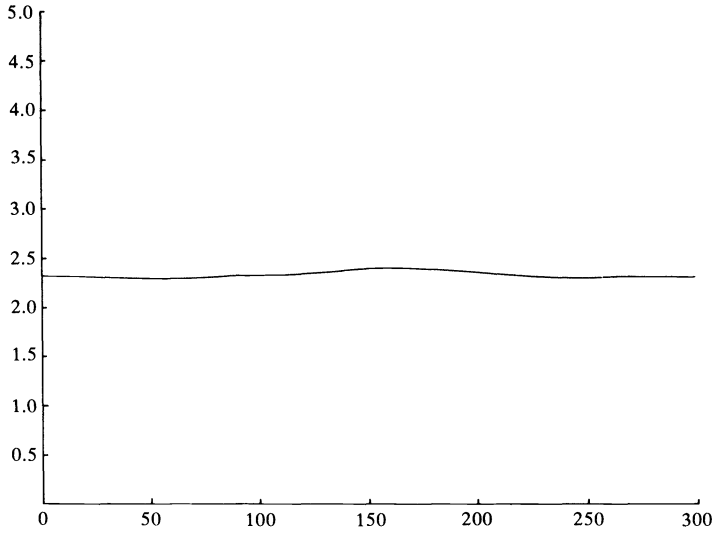


FIG. 1. *Area*, $s = 1.0$, $\Delta t = .015$, *time steps* 1–300.

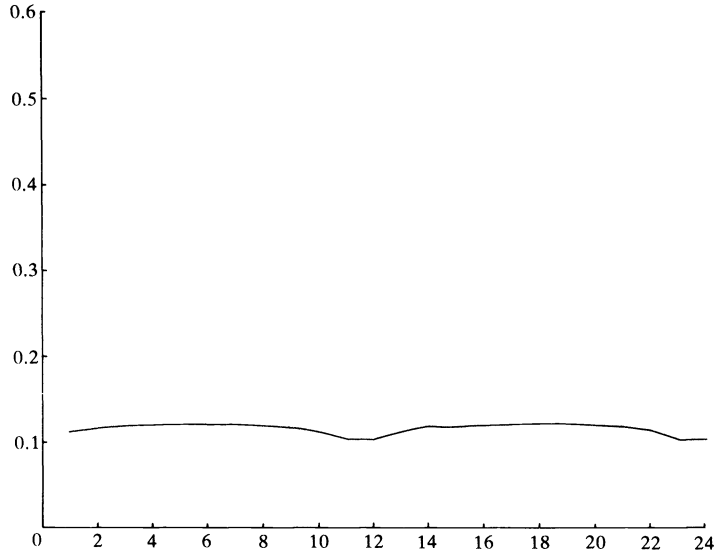


FIG. 2. *Tension of each link*, $\Delta s = 1.0$, $\Delta t = .015$, *time step* 60.

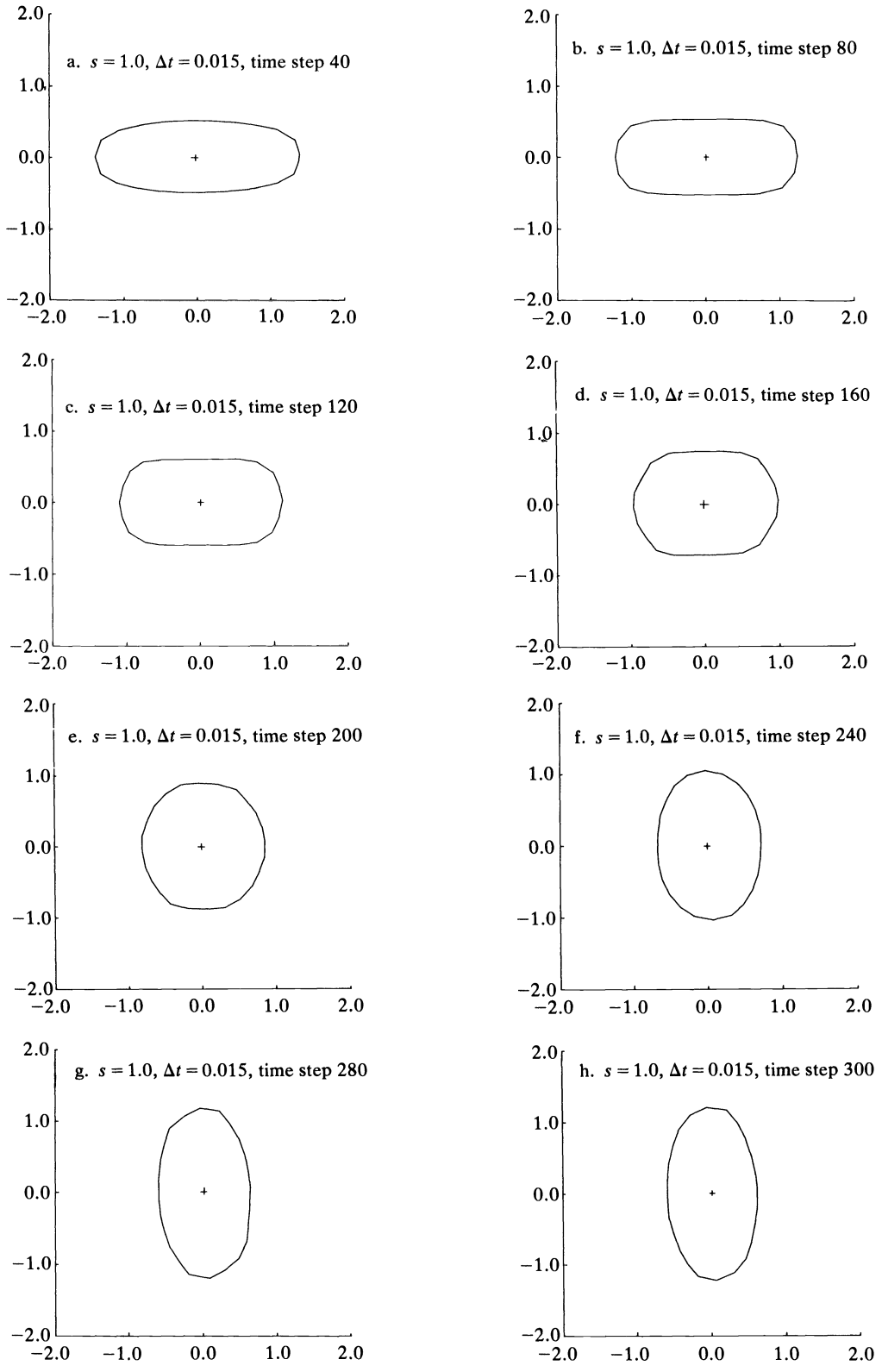


FIG. 3. *Boundary configurations.*

With the stiffness fixed at $C = 1.0$, several runs with different Δt that satisfied (28) were made to verify nondependence on the time step size. A run with $\Delta t = .02$ was made, and approximately three hundred time steps were required to calculate roughly a half-cycle of the oscillation. The ellipse bounced from being elongated along the horizontal axis at $t = 0$ to being elongated along the vertical axis, passing through the configuration of a nearly perfect cycle. Because of numerical viscosity, the final configuration was not symmetrical with the initial configuration. In fact, the final configuration was less eccentric than the initial one.

The flow $\mathbf{u}^n$ satisfied the condition $\nabla \cdot \mathbf{u}^n = 0$ everywhere. Seepage across the boundary was nonetheless present in the calculation. These two seemingly contradictory facts were reconciled as follows: The flux across the boundary is given by $G = \int_{\mathbf{x}(\cdot,t)} \mathbf{u}^n \cdot \boldsymbol{\eta}\, ds$, where $\boldsymbol{\eta} = \boldsymbol{\eta}(s, t)$ is the normal unit vector at $\mathbf{x}(s, t)$. Although the average G is zero, the numerical flux $G^n = \sum_{k=1}^{N} \mathbf{u}_k^n(\mathbf{x}_k^n) \cdot \boldsymbol{\eta}_k^n$, where $\boldsymbol{\eta}_k^n$ is the normal unit vector at $\mathbf{x}_k^n$, need not be zero. To assign a value to the parameter σ, we had to compromise between two conflicting criteria: If σ is too large, there is no resolution left in the calculation, while if σ is too small we have too much seepage. Best results were attained by assigning the value $\sigma = h = d/2$ to the cutoff σ and to each vortex dipole's semidiameter h, where d is the length of each of the subdivisions of the boundary at $t = 0$. Best results were also obtained by setting the resting length r_0 at $r_0 = d/2$ in the expression $T = s(r - r_0)$ (Fig. 1).

Conclusion. We have presented a method for simulating the interaction between a two-dimensional, incompressible, inviscid fluid and a one-dimensional, tense elastic boundary. The boundary exerts forces on the fluid, but its motion is determined by that of the fluid. We have, therefore, examined a coupled mechanical system. Our method of solution, the vortex dipole algorithm, has succeeded in resolving the complexity of the mechanical system we have considered.

The method of solution we have introduced in this paper leads to a very efficient scheme. Since each vortex pair is assumed to move with its associated boundary point, it follows that the total number of dipoles remains fixed throughout the calculation. Thus, the amount of labor required to update the location of boundary markers and vortex pairs remains the same at each time step.

In future work we intend to remove our assumption about vortex dipoles moving along the boundary points. We propose to do this as follows: At each time step we shall track the motion of the boundary via a different collection of markers, namely the centers of our vortex pairs, whose locations will not need to coincide, as in our present scheme, with the current location of the markers from the previous time step. To calculate the boundary forces at each marker, we shall obtain the tension (constant along the boundary) from the link formalism and compute the curvature vector via splines.

Acknowledgments. The author wishes to express his gratitude to Professors A. J. Chorin and C. S. Peskin, whose work made feasible the present research, and whose support and encouragement made it possible.

REFERENCES

[1] C. S. PESKIN, *Flow patterns around heart valves: A numerical method*, J. Comput. Phys., 10 (1972), pp. 252–271.

[2] ———, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.

[3] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[4] M. F. McCRACKEN AND C. S. PESKIN, *A vortex method for blood flow through heart and valves*, J. Comput. Phys., 35 (1980), pp. 183–205.
[5] C. S. PESKIN AND A. W. WOLFE, *The aortic sinus vortex*, Federation Proceedings, 37 (1978), pp. 2784–2792.
[6] C. S. PESKIN, *Numerical study of aortic flow*, in preparation.
[7] ———, *Lectures on mathematical aspects of physiology*, Lecture Notes, Courant Institute, New York Univ., New York.
[8] R. H. MENDEZ, *Numerical study of aortic sounds*, presented at SIAM National Meeting, Rensselaer Polytechnic Inst., Troy, NY, June 1981.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0008 $01.25/0

SHOCKS IN GAS PIPELINES*

D. MARCHESIN† AND P. J. PAES-LEME‡

Abstract. We use the uniform sampling method to study shocks and spikes in transient flow in gas pipelines. The method is well suited for this purpose. Both shocks and spikes tend to be masked by the use of a conventional damping term known as Moody friction. They are also masked by standard numerical schemes.

Key words. gas pipelines, pipe transients, Moody friction, Glimm's method, Riemann problem

1. Introduction. When the flow rate is changed in a gas pipeline network, pressure surges develop. In this case shocks will be created for sufficiently wide and long pipes, as we will see. The objective of designers of fluid systems is to avoid undesirable transients [17].

We use the uniform sampling method [4], [1] to perform a numerical simulation which gives good results for these spikes even for coarse grids (40 mesh blocks). This fact could contribute to the efficiency of the simulation of pipeline systems consisting of many interconnected pipes and pumping stations.

We consider the flow in pipelines governed by the partial differential equations of gas dynamics with constant temperature. This is a commonly used approximation. Other approximations are considered in [8]. We use the one-dimensional conservation laws of mass and momentum, plus a constitutive equation of state. They are

$$\begin{aligned} &\rho_t + (\rho u)_x = 0, \\ &(\rho u)_t + p_x + (\rho u^2)_x + f\frac{\rho u|u|}{2d} = 0, \\ &p = c^2\rho, \end{aligned} \tag{1.1}$$

for $0 \leqq x \leqq L$, $0 \leqq t$, where ρ is the density, u is the fluid speed, p is the pressure, t is the time, x is the distance along the pipe, c is the isothermal speed of sound, d is the diameter of the pipe and $f = f(\rho, u)$ is the "Moody friction factor" [2], [15]. We neglect any gravity effects by considering only horizontal pipes.

As boundary conditions we specify as function of time either the pressure or the flux at each end of the pipe. The initial condition at time zero is taken to be the steady state solution of (1.1).

We now indicate circumstances in which shocks occur. For wide enough pipes the friction term in (1.1) becomes negligible. We consider a smooth compression wave such that at an initial time there are two constant states separated by a length l. For isothermal flow the characteristic speeds ahead and behind are $c + u_a$ and $c + u_b$, where u_a and u_b are the respective flow speeds of the gas. Assuming that the wave moves to the right, i.e., $u_b > u_a > 0$, the shock will be formed in a time Δt such that $(c + u_b)\Delta t = l + (c + u_a)\Delta t$, provided the pipe is long enough. For example, taking the state

* Received by the editors March 4, 1981, and in revised form December 18, 1981.

† Departamento de Matemática, Pontifícia Universidade Católica do RJ, 22453 Rio de Janeiro, Brasil. The research of this author was supported in part by the Army Research Office under grant DAAG29-79-C-017 9 and by the National Science Foundation under grant PHY-80-01979.

‡ Departamento de Matemática, Pontificia Universidade Católica do RJ, 22453 Rio de Janeiro, Brasil. The research of this author was supported in part by the Department of Energy under grant DE-AC02-76ER03077-V.

approximately described in the left part of Fig. 2a as initial, with $u_a = 10$ m/s, $u_b = 240$ m/s and $l = 40$ m we have $\Delta t = 0.175$ s. The wave will have travelled 97 m by the time the shock is formed.

Pipes with a diameter of 1.0 m or 1.5 m are used to exhaust the gases generated in smelting plants. Transients are generated by sudden shut-off of exhaustion turbines caused by power failures.

Analyzing a linearized version of (1.1), one can show that by introducing the friction term, a shock has its amplitude reduced in time but its sharpness is preserved. Thus the formation of shocks due to pressure transients is inhibited by friction in extremely thin pipes.

The uniform sampling method used to advance the solution in time is well suited when discontinuities are the main feature of the problem, and it has recently been used in a variety of applications. Finite element techniques have been employed to solve (1.1) [2], [5], [6], [9]. Good resolution of shocks is obtained only through the use of meshes which are too fine for the treatment of a full network. In §2 we describe the method for solving (1.1). In §3 we solve the associated Riemann problem, and the results are presented in §4.

2. The numerical procedure. In order to advance the approximate solution (ρ, u) of (1.1) from time t_n to time $t_{n+1} = t_n + \Delta t$ we use operator splitting. Taking $\rho(t_n)$ and $u(t_n)$ as Cauchy data we march by Δt through

$$\rho_t + (\rho u)_x = 0, \qquad (\rho u)_t + c^2 \rho_x + (\rho u^2)_x = 0 \tag{2.1}$$

obtaining $(\tilde{\rho}, \tilde{u})$. This in turn is used as Cauchy data to march by Δt through

$$\rho_t = 0, \qquad (\rho u)_t + f\frac{\rho u|u|}{2d} = 0 \tag{2.2}$$

obtaining $\rho(t_{n+1})$ and $u(t_{n+1})$. For a proof of convergence of this procedure see [7]. This approach was used in [12], [13], [14].

At time t_n, the solution is assumed to be the constants $(\rho^n_{i+1/2}, u^n_{i+1/2})$ in the ith interval $i\Delta x \leqq x < (i+1)\Delta x$, for $i = 0, 1, 2, \cdots, N-1$. The time step obeys a Courant–Friedrichs–Lewy condition $\Delta t \leqq \frac{1}{2}\Delta x / \sup_i (|u^n_{i+1/2}| + c)$, where c is the (constant) sound speed $\sqrt{p/\rho}$. For $t_n \leqq t \leqq t_n + \Delta t$, the solution is then uniquely defined and is obtained by solving for each i the Riemann problems associated with equations (2.1) with Cauchy data at $t = t_n$:

$$(\rho, u) = (\rho^n_{i-1/2}, u^n_{i-1/2}) \quad \text{for } x < i\Delta x,$$

$$(\rho, u) = (\rho^n_{i+1/2}, u^n_{i+1/2}) \quad \text{for } x > i\Delta x.$$

The solution of these Riemann problems is described in the next section. Thus a solution $(\rho(x, t), u(x, t))$ is obtained for $t_n \leqq t < t_{n+1}$; this solution is not piecewise constant. We introduce a sequence $\{\theta_n\}$ of numbers equidistributed in the interval $[0, 1]$. (Here we use the fractional part of $n\sqrt{2}$.) Finally, we obtain a piecewise constant function

$$\tilde{\rho}_{i+1/2} = \rho((i+\theta_n)\Delta x, t_{n+1}),$$

$$\tilde{u}_{i+1/2} = u((i+\theta_n)\Delta x, t_{n+1}), \qquad i = 0, 1, 2, \cdots, N-1.$$

We remark in passing that if the friction term in (2.2) were zero the function above would be the computed solution at time t_{n+1}.

Since the procedure above is essentially first order accurate in time, any reasonable scheme may be employed to advance the ordinary differential equation in (2.2). We use Euler's scheme:

$$\rho_{i+1/2}^{n+1} = \tilde{\rho}_{i+1/2}, \qquad u_{i+1/2}^{n+1} = \tilde{u}_{i+1/2} - \Delta t f \frac{\tilde{u}|\tilde{u}|}{2d}\bigg|_{i+1/2}, \qquad i = 0, 1, 2, \cdots, N-1.$$

In order to initialize the simulation at time zero, we use the steady state solution of (1.1), namely,

$$(\rho u)_x = 0, \qquad c^2 \rho_x + (\rho u^2)_x + f \frac{\rho u|u|}{2d} = 0.$$

To solve this equation, ρ and u are given at the inlet end of the pipe.

3. The Riemann problem. Consider the initial value problem for (2.1) with initial data

$$(\rho, u)(x, 0) = (\rho_r, u_r), \quad x \geqq 0, \qquad (\rho, u)(x, 0) = (\rho_l, u_l), \quad x < 0,$$

where $S_r = (\rho_r, u_r)$ and $S_l = (\rho_l, u_l)$ are two constant states. Correspondingly denote by p_r, u_r and p_l, u_l the constant pressures and gas velocities of the right and left states.

For later times any left state S_l can be connected to a right state S_r through a left centered wave S_1, a constant state S_* and a right centered wave S_2 (see Fig. 1).

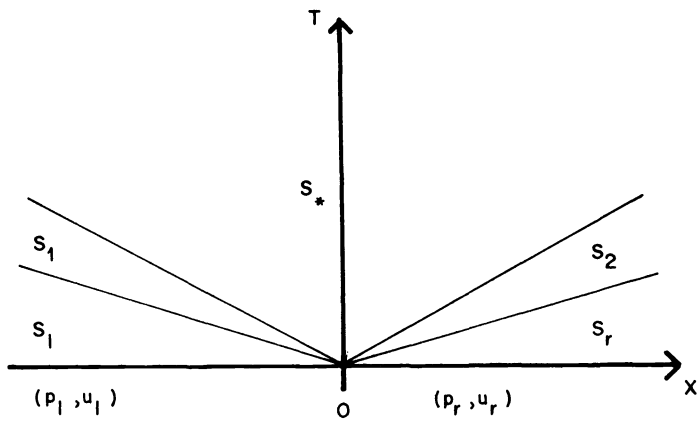


FIG. 1. *Solution of the Riemann problem.*

Denote by ρ_*, p_* and u_* respectively the density, pressure and velocity of the gas in the constant state S_*. Define the quantities.

$$M_l = \frac{p_l - p_*}{u_l - u_*}, \qquad M_r = \frac{p_r - p_*}{u_r - u_*}. \tag{3.1}$$

It is easier to describe the solution of the problem in configuration space, where S_l, S_* and S_r are points. These points are connected through rarefaction or shock curves which represent S_1 and S_2.

If the left centered wave is a rarefaction wave ($p_* < p_l$), we connect the state S_l to a state S_* by a rarefaction curve, i.e., a curve on which the Riemann invariant $u + c \log \rho$ is constant:

$$u_l + c \log \rho_l = u_* + c \log \rho_*.$$

If the right centered wave is a rarefaction wave ($p_* < p_r$), we connect a state on the right to a state S_* by a rarefaction curve on which the other Riemann invariant $u + c \log \rho$ is constant:

$$u_r + c \log \rho_r = u_* + c \log \rho_*.$$

If the wave is a shock we then connect two adjacent states by a shock curve, i.e., a curve which satisfies the Rankine–Hugoniot jump condition:

$$\frac{[\rho u]}{[\rho]} = \frac{[\rho u^2 + p]}{[\rho u]} = \mathbf{U}. \tag{3.2}$$

Here $\mathbf{U}$ equals U_l (the speed of the left shock) or U_r (the speed of the right shock) if the wave is left centered or right centered respectively. If the left wave is a shock we have $p_* > p_l$ and from (3.2) we obtain

$$M_l = \rho_l(u_l - U_l) = \rho_*(u_* - U_l). \tag{3.3a}$$

Similarly, if the right wave is a shock we have $p_* > p_r$ and

$$M_r = -\rho_r(u_r - U_r) = -\rho_*(u_* - U_r). \tag{3.3b}$$

In either case, from (3.1) we obtain

$$\begin{aligned} M_l &= M_l(p_*) = (\rho_l \rho_l)^{1/2} \phi\left(\frac{p_*}{p_l}\right), \\ M_r &= M_r(p_*) = (\rho_r p_r)^{1/2} \phi\left(\frac{p_*}{p_r}\right), \end{aligned} \tag{3.4}$$

where

$$\phi(x) = \begin{cases} \dfrac{1-x}{-\log x}, & x \leqq 1, \\ x^{1/2}, & x \geqq 1. \end{cases}$$

Upon elimination of u_* from (3.1) we obtain

$$p_* = \left(u_l - u_r + \frac{p_r}{M_r} + \frac{p_l}{M_l}\right) \Big/ \left(\frac{1}{M_l} + \frac{1}{M_r}\right). \tag{3.5}$$

The substitution of (3.4) in (3.5) yields a fixed point equation of the form

$$p_* = \psi(p_*). \tag{3.6}$$

The initial guess p_*^0 is chosen as in [1], [11] to be the average value $(p_l + p_r)/2$. We use one step of Godunov's iterative procedure on (3.6) to create a second guess p_*^1. These two guesses are used to initialize the secant method to find the zero of the function

$$p_* - \psi(p_*).$$

Replacing Godunov's iterative procedure in [1], [11], [14] by the secant method makes the corresponding scheme almost two times faster.

More details may be found in [14], where the Riemann problem for isothermal gas flow has been solved. In [14], the Riemann solver is reported to become 66% faster by using the isothermal rather than the isentropic flow equations.

We refer the reader interested in the details of the implementation of the Riemann problem to [14]. Flow charts are presented in [11], where an equation of state $p = A\rho^{\gamma}(\gamma > 1)$ is considered. The isothermal case can be obtained by the limit when γ tends to 1.

4. Results. We simulated the flow of an ideal gas at fixed temperature in a pipe with a diameter of 20 cm and length 800 m. We assumed that at time zero there was a steady state motion with the pressure of the gas entering the pipe at the left being 10^5 N/m^2 (approximately 1 atmosphere), density equal to 1 kg/m^3 and speed 10 m/sec. To simplify our computation with friction we took a constant Moody factor equal to 0.01 (see [16, p. 30, Eq. 1.2–5]). Typical mesh sizes employed were 20 m.

The pressure was increased by 10^5 N/m^2 linearly at both ends of the pipe in 0.1 sec and then kept fixed. The solution at this time is well represented in Fig. 5(a).

The results obtained using the uniform sampling method with zero friction are displayed in Fig. 2. Note the sharp resolution of the shocks and of the spike, even on the relatively coarse mesh used.

The results obtained using the Lax–Wendroff finite difference scheme [10] with zero friction are shown in Fig. 3. We note that the shocks and the spike are not well resolved. The oscillations behind the shock can be reduced, as shown in Fig. 4, through the introduction of an artificial viscosity term [10] via operator splitting. The amplitude of the oscillations is not satisfactorily reduced by using a finer mesh (see Fig. 5).

The aim of this paper is to determine the effect of the Moody friction term using a method that resolves shocks whenever they are present: namely, the uniform sampling method. Figures 6 and 7 show that the Moody friction changes dramatically certain quantitative aspects of the solution. Even for smooth initial data, shocks are still created, but their amplitudes are strongly reduced. The initial data and boundary conditions considered in this paper are not contained in [7], where it is shown that under certain conditions shocks do not develop.

Figures 6c, d show clearly that the Moody friction term reduces the amplitude of the shock as it advances. Thus, one way to prevent a pump from receiving a pressure surge caused by turning on a second pump at the other end of the pipe would be to make the pipe as long as possible. Pipeline designers seem to be aware of this consideration [3].

It is commonly believed that the use of a Moody friction term is justified when turbulent flow is simulated. For laminar flow, see [18], where a memory term is introduced to model the friction. The use of either these friction terms or the finite difference techniques tends to smooth the solution. If a simulator is used in the design of a pipeline system with the purpose of detecting transients, it is better to use the uniform sampling method.

Acknowledgments. We would like to thank J. Glimm for support and encouragement throughout this work. We are grateful to T. Dupont, E. L. Isaacson and R. Sampaio for many enlightening conversations. Most of this work was done at the Courant Institute – Department of Energy Computing Laboratory which kindly made its facilities available to us.

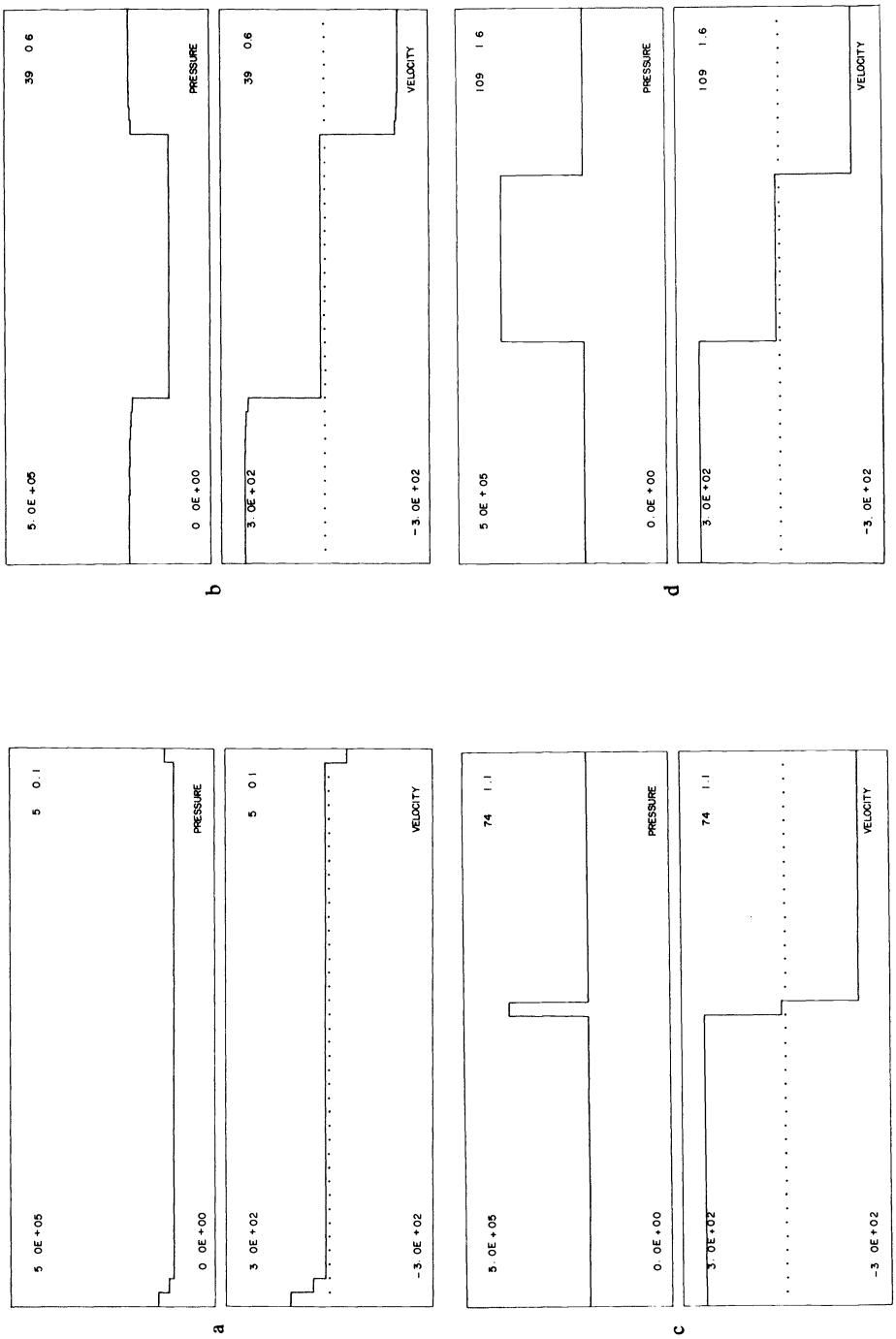


FIG. 2. *Solution using the uniform sampling method with 40 mesh points, as function of position. The scales are indicated by the numbers on the left. The time step number and the physical time are indicated at the upper right corner of the figure.* (a) *approximates a piecewise linear solution. Note the sharp resolution of the shocks and of the spike in* (b), (c) *and* (d).

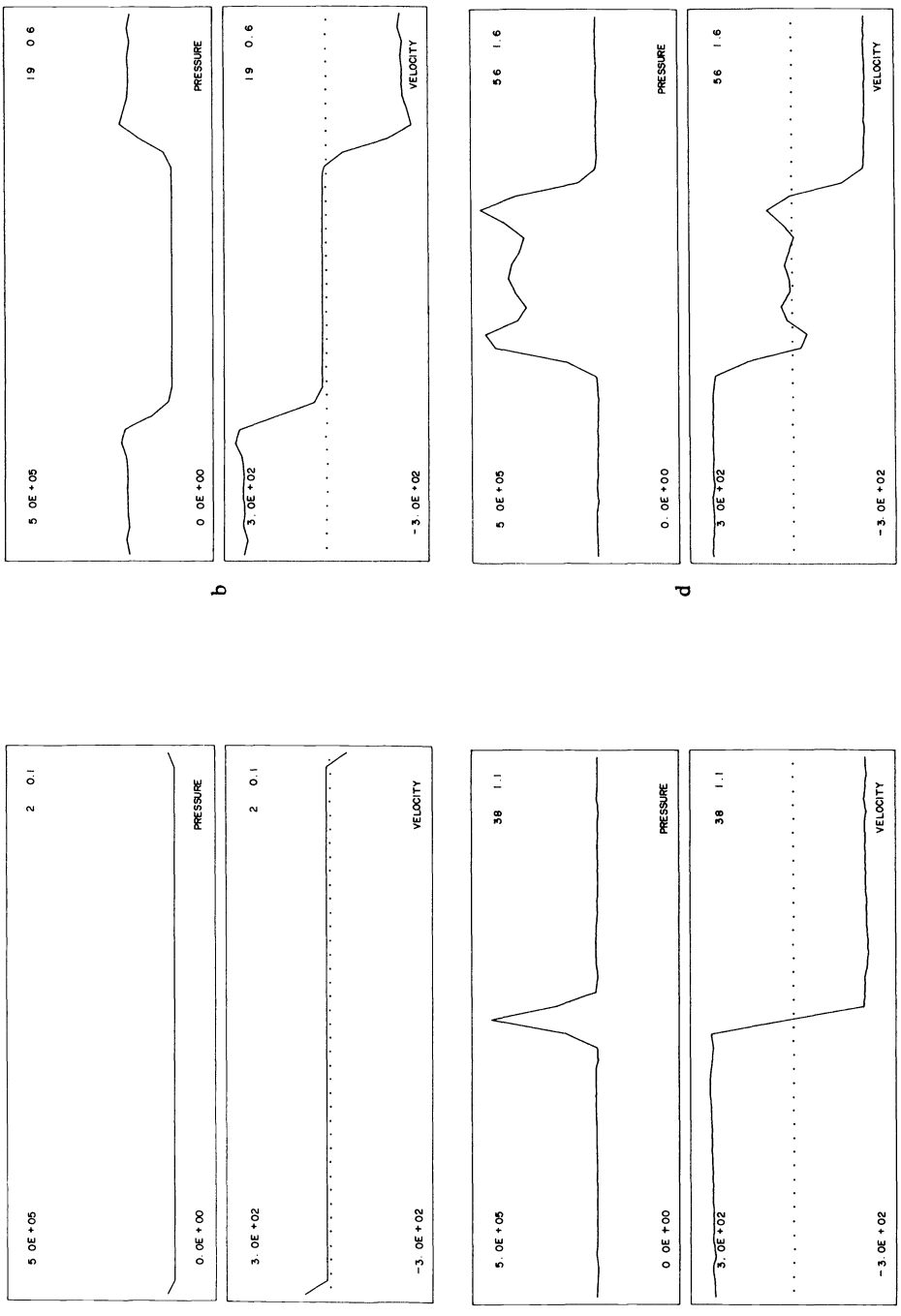


FIG. 3. *Solution using the Lax–Wendroff finite difference scheme with 40 mesh points, as function of position. Note the insufficient resolution of the shocks. There are oscillations behind the shocks and superimposed on the spike.*

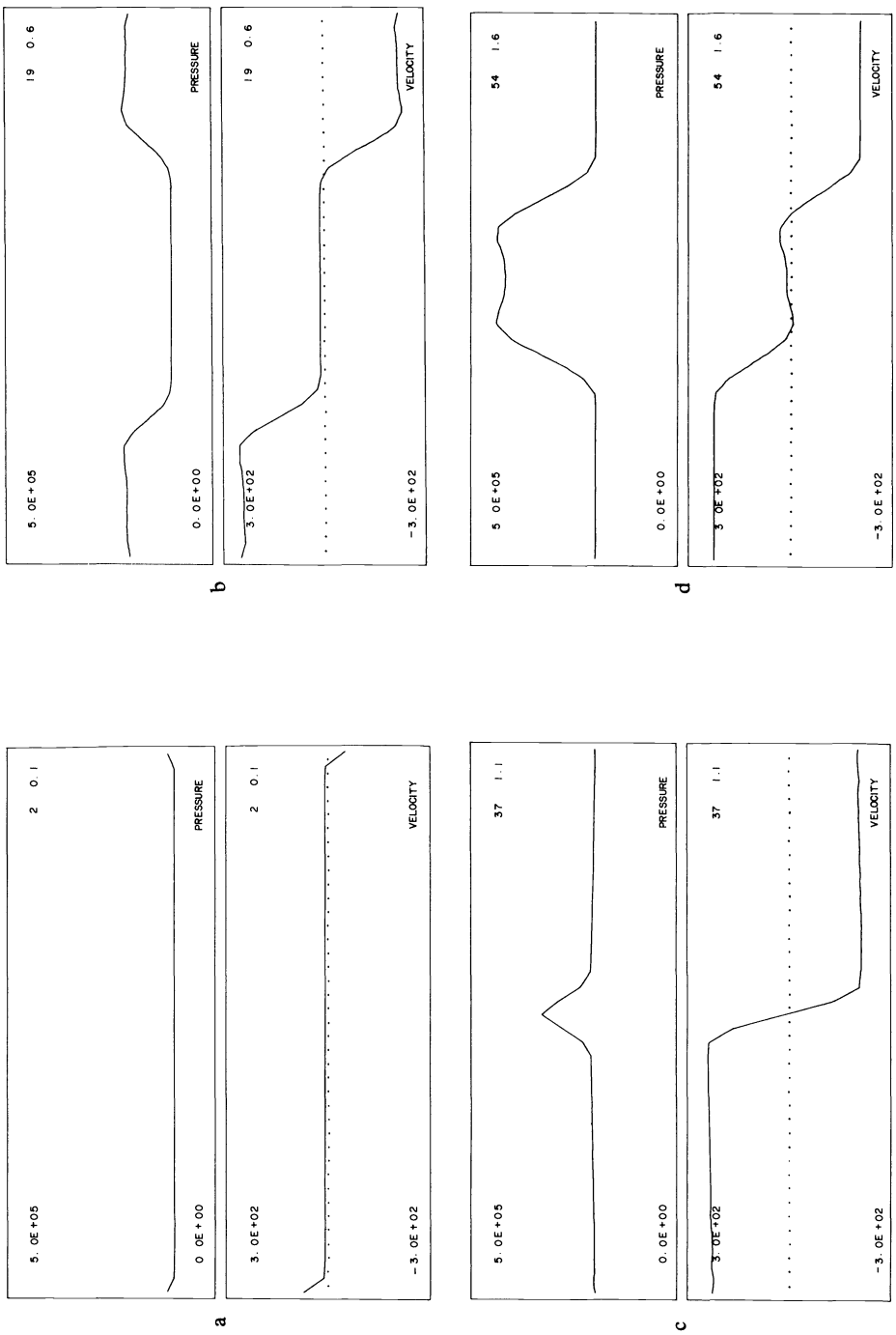


FIG. 4. *Solution using the Lax–Wendroff finite difference scheme with artificial viscosity. A comparison with Fig. 3 indicates that the spurious oscillations are reduced as well as the amplitude of the spike.*

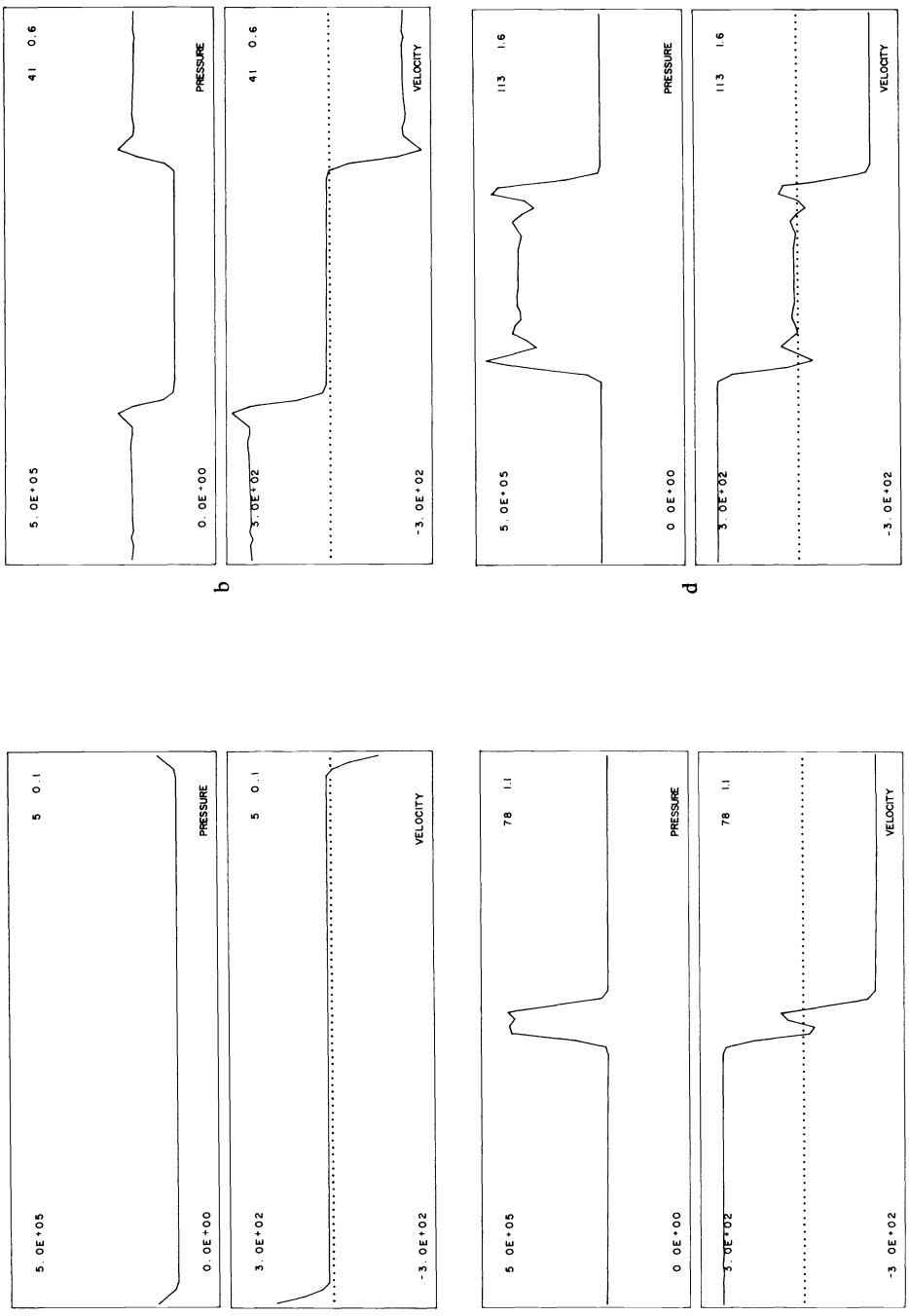


FIG. 5. *Solution using the Lax–Wendroff finite difference scheme with 80 mesh points. Compare with Fig. 3.*

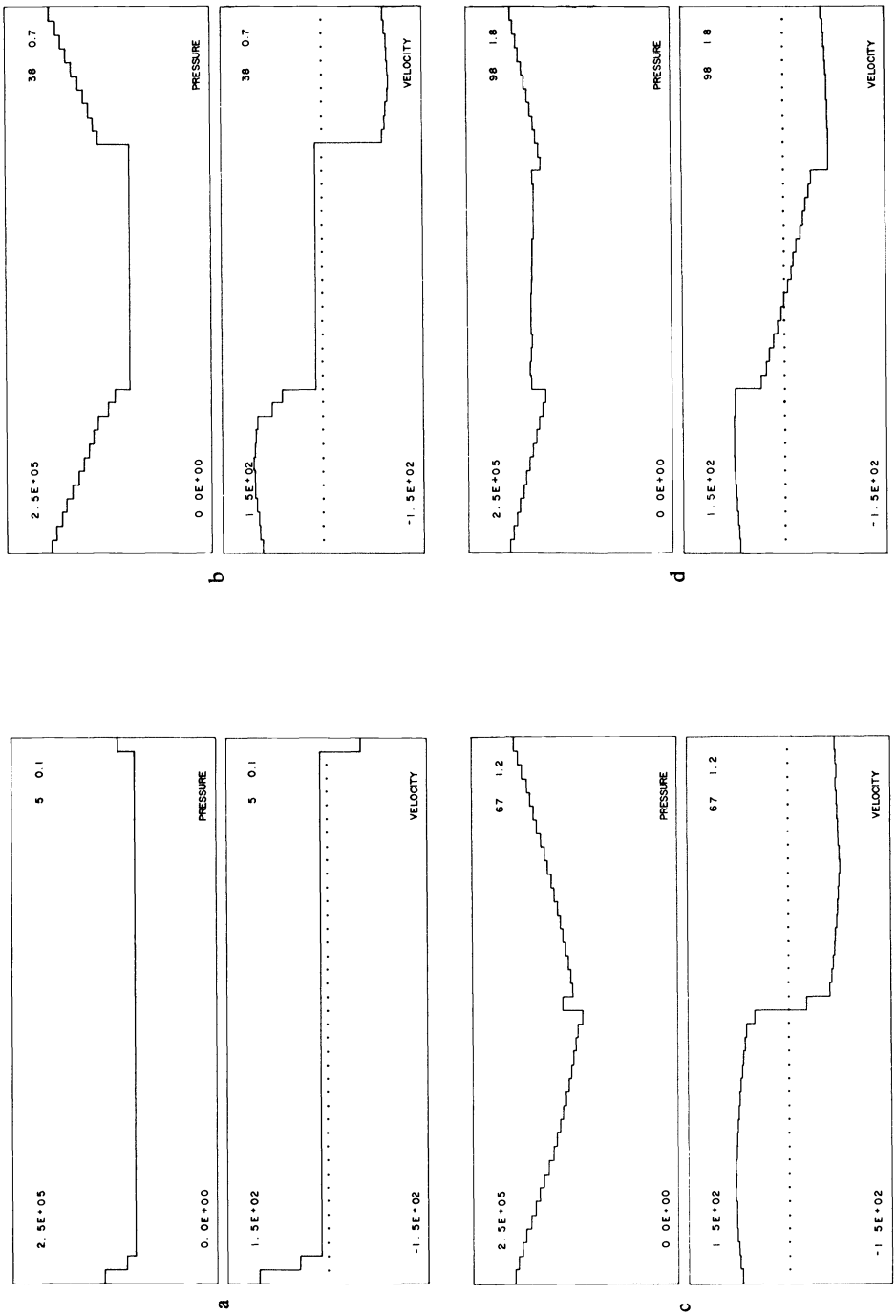


FIG. 6. *Solution using the uniform sampling method with Moody friction (40 mesh blocks). A comparison with Fig. 2 shows that the shocks are strongly attenuated.*

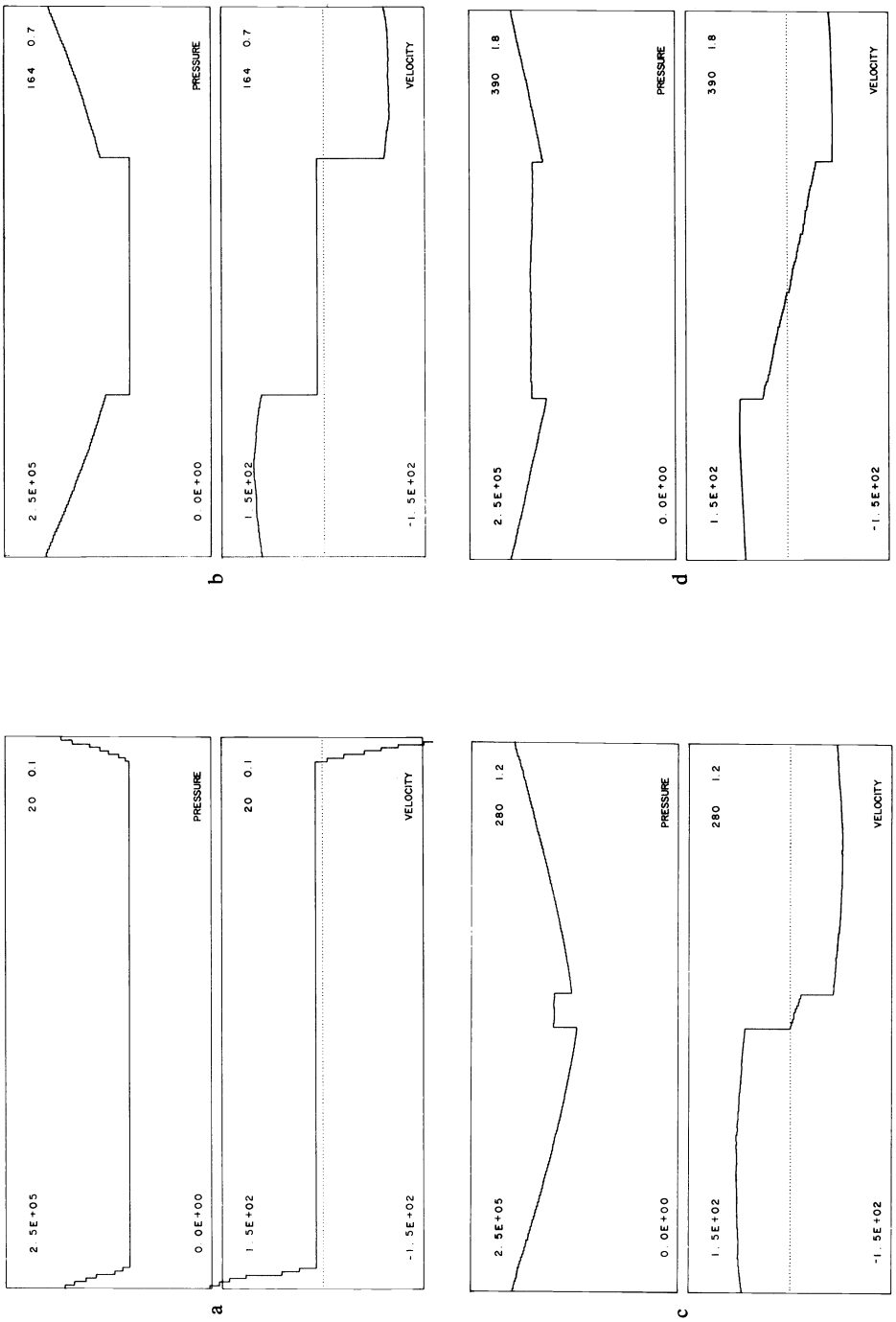


FIG. 7. *Solution using the uniform sampling method with Moody friction (160 mesh blocks). A comparison with Fig. 6 indicates that the shocks, though attenuated, are actually present, even for smooth initial data.*

REFERENCES

[1] A. J. CHORIN, *Random choice solutions of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–553.

[2] T. DUPONT, *Galerkin methods for modeling gas pipelines*, in Constructive and Computational Methods for Differential and Integral Equations, Lecture Notes in Mathematics 430, Springer-Verlag, Heidelberg, 1974, pp. 112–130.

[3] ———, private communication.

[4] J. GLIMM, *Solutions in the large for nonlinear hyperbolic systems of equations*, Comm. Pure Appl. Math., 18 (1965), pp. 697–715.

[5] M. LUSKIN, *An approximation procedure for nonsymmetric, nonlinear hyperbolic systems with integral boundary conditions*, SIAM J. Numer. Anal., 16 (1979), pp. 145–164.

[6] ———, *A finite element method for first order systems*, Math. Comput., 35 (1980), pp. 1093–1112.

[7] M. LUSKIN AND B. TEMPLE, *The existence of global weak solutions of the nonlinear waterhammer problem*, Courant Mathematics and Computing Laboratory Report DOE/ER/03077-174, Courant Institute, New York, New York, 1981.

[8] D. MARCHESIN, P. J. PAES-LEME AND R. SAMPAIO, *Transients in pipes*, Proc. VI Congresso Brasilerio de Engenharia Mecânica–COBEM-PUC/RJ (1981), pp. 341–353.

[9] H. H. RACHFORD, JR. AND E. L. RAMSEY, *Application of variational methods to transient flow in complex liquid transmission systems*, Soc. Petroleum Engineers, Trans., (1977), pp. 151–166.

[10] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems* 2nd ed., Wiley-Interscience, New York, 1967.

[11] G. A. SOD, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.

[12] ———, *A numerical study of a converging cylindrical shock*, J. Fluid Mech., 83 (1977), pp. 785–794.

[13] ———, *A hybrid random choice method with application to internal combustion engines*, Proc. Sixth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, 90, Springer-Verlag, New York, 1979.

[14] ———, *A numerical method for slightly viscous axisymmetric flows with application to internal combustion engines*, J. Comp. Phys., to appear.

[15] V. STREETER, *Fluid Mechanics*, McGraw-Hill, New York, 1971.

[16] A. P. SZILAS, *Production and transport of oil and gas*, Developments in Petroleum, 3, Elsevier, New York, 1975.

[17] E. B. WYLIE AND V. L. STREETER, *Fluid Transients*, McGraw-Hill, New York, 1978.

[18] W. ZIELKE, *Frequency-dependent friction in transient pipe flow*, J. Basic Engng., Trans. ASME, paper No. 67-WA/FE-15, 1967.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

FINITE DIFFERENCE CALCULATIONS OF BUOYANT CONVECTION IN AN ENCLOSURE, I. THE BASIC ALGORITHM*

HOWARD R. BAUM†, RONALD G. REHM‡,
P. DARCY BARNETT‡ AND DANIEL M. CORLEY†

Abstract. A novel mathematical model of buoyant convection in an enclosure, developed earlier, is solved by finite difference techniques in the two-dimensional case. This model has been developed as a principal analytical tool for the prediction of the movement of smoke and hot gases in fires. Effects of large density variations caused by substantial heating are retained while acoustic (high-frequency) waves, which are unimportant to buoyant convection, are analytically filtered out. No viscous or thermal conduction effects are included in the model. These two characteristics (filtering and lack of dissipative effects) distinguish the model from all others describing buoyant convection. The mathematical model consists of a mixed hyperbolic and elliptic set of nonlinear partial differential equations: the problem is a mixed initial, boundary value one. An explicit time-marching algorithm, second-order accurate in both space and time, is used to solve the equations. The computational procedure uses a software package for solving a nonseparable elliptic equation developed especially for this problem. The finite difference solutions have been carefully compared with analytical solutions obtained in special cases to determine the stability and accuracy of the numerical solutions. The computer model has been used to compute the buoyant convection produced in an enclosure by a spatially distributed heat source simulating a fire. The computed results show qualitative agreement with experimentally observed buoyant convection in enclosure fires.

Key words. buoyant convection, computations—finite difference, Euler equations, finite difference equations, fire-enclosures, fluid flow

1. Introduction. This paper presents the first results for a finite-difference integration of an approximate set of equations describing buoyant convection in an enclosure. The work represents a continuation of the research reported in [16], where the set of approximate equations was derived. The primary application of interest to the authors is the movement of smoke and hot gases caused by a fire in a room.

The research presented here is distinguished from previous numerical computations of buoyant convection in three respects. First, in this model the fluid is taken to be an inviscid, non-heat-conducting perfect gas, and the spatial and temporal magnitude and variation of the heat source, which simulates a fire and drives the flow, are taken as known. These approximations are justified because under conditions characteristic of even a small room fire, the Grashof numbers (representing the ratio of the inertial to viscous forces for natural convection) are large enough for molecular transport phenomena to be important only in wall boundary layers and in the highly convoluted flame sheets which constitute the region of intense heat addition. The study of the detailed flame structure of real fires is an extraordinarily complicated subject in its own right, and is bypassed here by specifying the heat source. Wall boundary layers represent a local refinement to be considered separately at a later date. Batchelor [3] gives a brief but relevant discussion of the applicability of the inviscid equations in the context of atmospheric motions.

It should be noted that such simplifications do not preclude a description of turbulence; but no turbulence model is explicitly included in this study. We note, however, that any turbulence model appended to the present equations must be of the "sub grid" variety, since no spatial or temporal averaging is implied in the equations derived in [16].

* Received by the editors January 15, 1982.
† Center for Fire Research, National Bureau of Standards, Washington, DC 20234.
‡ Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234.

The omission of any turbulence model is based on the observation, quantified by McCaffrey [12], that most of the energy containing fluctuations in buoyant plumes induced by laboratory scale diffusion flames are of low frequency and large spatial extent. Such fluctuations, with frequencies typically in the range 2–5 Hz and length scales comparable to the local plume width, are directly resolved by the computational procedure. A knowledge of the behavior of higher frequencies and smaller length scales is of course crucial to an understanding of combustion related phenomena. However, as noted above, such questions have been bypassed in the present study.

Simple models of smoke and hot gas transport which neglect molecular transport phenomena have been reasonably successful in predicting global properties of flow fields [15]. The present work is intended as a first step towards more detailed studies along these lines.

Second, the approximate set of equations integrated in this paper are characterized by the fact that large density variations due to temperature changes are admitted, but compressibility effects are suppressed. Such a fluid has been termed thermally expandable in other contexts [14]. In the fire setting, allowance for density variations due to temperature increases during combustion is essential. It is common for temperature in a flame to exceed 1000°C, implying that the density decreases locally to less than one-quarter its ambient value in the nearly constant-pressure process. In [16], a set of equations of motion was derived formally which permit description of large density variations in a flow while ignoring acoustic oscillations arising because of the elastic properties of the fluid. Such model equations include the important features of buoyant flows without requiring excessive computer time necessary to determine high-frequency sound waves when numerically integrated. In this sense the equations "filter out" the sound waves while describing the lower frequency, organized motions due to buoyant effects such as internal waves.

Finally, in previous efforts to compute flow fields, produced by buoyancy or by any other mechanism, there are few reported detailed checks on the quality of the numerical solutions to the finite difference equations. By contrast, in [4] and [17] detailed comparisons are made with analytical/numerical solutions obtained to the general difference equations in simple, special cases. Through these comparisons, confidence in both the algorithm and its implementation as a computer code was gained. Such tests showed that the algorithm is stable, and for the example cases performed, the error made in solving the difference equations at any time step, even in the nonlinear case, was over two orders of magnitude less than the discretization errors made in approximating the partial differential equations by finite difference equations. For buoyant flows of the type considered in this paper, it is especially important to have confidence in both the stability and the accuracy of the algorithm so that the real physical instability represented by the fluid motion can be distinguished from any computational instabilities.

In § 2 the equations derived in [16] are recast into the form in which they are solved, and the finite difference approximations to the equations are presented. In § 3 solutions determined by this model, for the buoyant flow produced by a heat source in a rectangular enclosure, are presented and discussed.

2. Formulation.

2.1. Continuous problem. In [16] the authors derived a set of nonlinear equations describing nondissipative, buoyancy driven flows of a perfect gas. The flows were assumed to be generated by a localized heat source in which the heat is added slowly so that the time scale associated with the heat-source growth and resultant fluid motion

is long compared with the transit time of an acoustic signal across the spatial extent of the source. Flows induced by a room fire generally satisfy this assumption. Properties of the equations were discussed in [16]. In this section these equations are rewritten in a form appropriate for numerical integration by finite difference techniques, and the boundary conditions for the equations are presented.

As in [16], we consider an inviscid, non-heat-conducting perfect gas. The magnitude and the spatial variation of the heat source (representing the exothermic reaction in a fire) are taken as known; justification for such a model is given in [16]. The fluid and the fire source are assumed confined in a closed rectangular room with the center of the source along the floor. In contrast to [16], we consider only a completely enclosed room (no leaks), and when difference equations are introduced, we confine attention to the two-dimensional evolution of the flow.

The continuity, momentum, energy, and state equations are given respectively in [16] as:

$$
\begin{aligned}
&\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x_i}(\rho u_i)=0,\\
&\rho\left(\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}\right)+\frac{\partial(p-p_0(t))}{\partial x_i}-\rho k_i g=0,\\
&\rho c_p\left(\frac{\partial T}{\partial t}+u_j\frac{\partial T}{\partial x_j}\right)-\frac{dp_0}{dt}=Q(x_i,t),\\
&p_0(t)=\rho RT.
\end{aligned}
\tag{1}
$$

Here ρ is density, u_i the velocity in the ith coordinate direction ($i=1, 2, 3$), p is the pressure excess above the mean pressure $p_0(t)$ in the room, T the temperature, c_p the constant-pressure specific heat, R the gas constant, $k_i g$ is the gravitational acceleration (of magnitude g) and $Q(x_i, t)$ the specified volumetric heat source. The spatially uniform mean pressure $p_0(t)$ depends only upon time and increases because of the heating within the room. It is determined in a completely enclosed room by the equation

$$
\frac{dp_0}{dt}=\frac{\gamma-1}{V}\int_V Q(x_i,t)\,dV, \tag{2}
$$

where γ is the ratio of specific heats, V is the volume of the room and the integration is performed over this entire volume. Equation (2) is a thermodynamic statement that the mean pressure rise as a function of time is determined by the total heat added to the room. (Heat can only be added or removed volumetrically and not through the walls because thermal conduction and radiative transport have been ignored in this model.) It will also turn out to be a mathematical consistency condition required if a solution for the pressure field is to exist.

Equations (1) and (2) are the approximate set of nonlinear equations solved by finite difference techniques in two spatial dimensions in this paper. The equations admit buoyant or internal-wave motions while "filtering out" high-frequency, acoustic waves. They reduce to the Boussinesq equations when heating is mild, total density variations are small, and variations in the mean background pressure can be neglected (as would be the case if the room considered here were open or if the mean pressure variation were comparable to the spatial pressure perturbation). To recast the equations into a form more suitable for numerical computation, we take the substantial derivative of the equation of state and use this with the energy equation to eliminate the temperature. The resulting equation describes the evolution of the density under

heating:

$$\frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} = -\rho \frac{\partial u_i}{\partial x_i} = -\rho D(x_j, t), \tag{3}$$

where

$$D(x_j, t) = \frac{1}{\gamma p_0(t)} \left[(\gamma - 1) Q(x_j, t) - \frac{dp_0}{dt} \right]. \tag{4}$$

Equation (3) and the continuity equation identify $D(x_i, t)$ as the divergence

$$\frac{\partial u_i}{\partial x_i} = D(x_j, t). \tag{5}$$

Finally, as in [16], the equation for the spatially variable portion of the pressure is obtained by dividing the momentum equations by density and taking the divergence of these equations. The resulting equation is

$$\frac{\partial}{\partial x_i} \left(\frac{1}{\rho} \frac{\partial p}{\partial x_i} \right) = -\left[\frac{\partial}{\partial x_i} \left(u_j \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial D(x_j, t)}{\partial t} \right]. \tag{6}$$

Equation (6) is the generalization for a "thermally expandable" fluid, which we consider here, of the well-known incompressibility condition in Boussinesq fluids. When the density is constant, then $D(\mathbf{x}, t) = 0$, and (6) reduces to Poisson's equation. The boundary conditions on these equations are that velocity normal to any (impermeable) wall vanish,

$$u_i n_i = 0, \tag{7}$$

where n_i are the normal components of a vector describing the boundary walls. From (1) and these conditions, the appropriate boundary conditions on the pressure equation are obtained:

$$n_i \frac{\partial p}{\partial x_i} = \rho g n_i k_i. \tag{8}$$

An important observation must be made concerning (6) and the Neumann boundary conditions (8). When (6) is integrated over the total volume of the room, both the left-hand side of the equation and the first term on the right are divergence forms and can be converted into integrals over the boundaries of the room. Application of boundary conditions (7) and (8) show that each of the terms is zero; therefore, the integral over the volume of $\partial D/\partial t$ must also be zero. The requirement that the integral of (4) for D over the volume be zero produces (2), the condition for the spatially uniform background pressure. Therefore, the elliptic equation (6) for the pressure, with the Neumann boundary conditions, is seen to produce a condition which must be satisfied for a solution to the equation to exist. This condition (2) determines the time evolution of the spatially uniform background pressure and demonstrates that the total pressure can be consistently separated into a spatially uniform background pressure and an inhomogeneous time-dependent overpressure. In the next section describing the difference equations, exactly analogous considerations are found to apply to the linear algebraic equations approximating (6) and the boundary conditions (8).

For selecting a difference scheme, the second of equations (1), the momentum equations, are rewritten in vector invariant form, noting that u_i are components of

the velocity vector field $\mathbf{u}$

$$\frac{\partial \mathbf{u}}{\partial t}+\frac{1}{2}\nabla(q^2)-\mathbf{u}\times\boldsymbol{\omega}=-\frac{1}{\rho}\nabla p+\mathbf{k}g, \tag{9}$$

where $q^2=\mathbf{u}\cdot\mathbf{u}$ and $\boldsymbol{\omega}=\nabla\times\mathbf{u}$ is the vorticity. The curl of (9) yields the vorticity-transport equation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t}-\nabla\times(\mathbf{u}\times\boldsymbol{\omega})=-\nabla\left(\frac{1}{\rho}\right)\times\nabla p. \tag{10}$$

Since $\mathbf{u}$ is a vector field, it is necessary to calculate correctly both its divergence and its curl. Equation (5) specifies the divergence of $\mathbf{u}$, and the equation for the pressure, (6), assures that the divergence is properly determined at each time. Equation (10) is the equation for the evolution of the curl of the velocity, the vorticity. The difference scheme selected must assure that (10) is satisfied in difference form.

The complete set of recast nonlinear equations are gathered and rewritten below:

$$\frac{\partial \rho}{\partial t}+u_i\frac{\partial \rho}{\partial x_i}=-\rho D(x_j, t), \tag{11a}$$

$$\frac{\partial u_i}{\partial t}+\frac{1}{2}\frac{\partial}{\partial x_i}(u_j u_j)-\varepsilon_{ijl}u_j\omega_l=-\frac{1}{\rho}\frac{\partial p}{\partial x_i}+k_i g, \tag{11b}$$

$$\frac{\partial}{\partial x_i}\left(\frac{1}{\rho}\frac{\partial p}{\partial x_i}\right)=-\left[\frac{\partial}{\partial x_i}\left(u_j\frac{\partial u_i}{\partial x_j}\right)+\frac{\partial D(x_j, t)}{\partial t}\right], \tag{11c}$$

where

$$\frac{dp_0}{dt}=\frac{\gamma-1}{V}\int_V Q(x_i, t)\,dV, \tag{11d}$$

$$D(x_i, t)=\frac{1}{\gamma p_0(t)}\left[(\gamma-1)Q(x_i, t)-\frac{dp_0}{dt}\right], \tag{11e}$$

ε_{ijl} is the permutation tensor and $\omega_i=\varepsilon_{ijl}\,\partial u_l/\partial x_j$ are the components of the vorticity vector.

Boundary conditions are

$$u_i n_i=0, \tag{12a}$$

$$n_i\frac{\partial p}{\partial x_i}=\rho g k_i n_i. \tag{12b}$$

For numerical integration, (11) and boundary conditions (12) are altered in two additional ways. First, we calculate the density and pressure differences from their initial values, which may be functions of the vertical coordinate. This is done to eliminate accuracy problems, since the thermally induced density and pressure differences can be very small during the early portion of the heating process. Second, the equations are made dimensionless. The nondimensionalization is done both for convenience and to ensure proper scaling. All dependent quantities are made to be of order unity in magnitude for purposes of the computation. The remaining dimensionless parameters characterize the strength and location of the heat source as well as the room geometry.

In Fig. 1 a schematic diagram of a fire evolving in a room and a set of coordinate axes are shown. It is assumed that initially the enclosure is filled with quiescent,

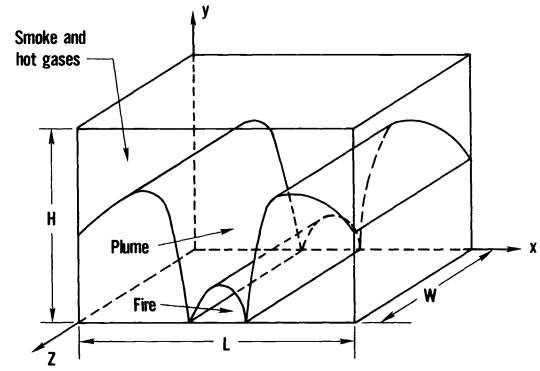


FIG. 1. *Schematic diagram of an enclosure with a two-dimensional heat source, plume and region of smoke and hot gases.*

stratified fluid of density $\rho_0(y)$, where we denote $x_1 = x$, $x_2 = y$ and $x_3 = z$. We define a density difference from ambient and a pressure difference as follows:

$$\tilde{\rho}(x, y, z, t) = \rho(x, y, z, t) - \rho_0(y), \tag{13a}$$

$$\tilde{p}(x, y, z, t) = p(x, y, z, t) - p_0(t) + g\int_0^y \rho_0(y')\,dy'. \tag{13b}$$

These differences $\tilde{\rho}$ and $\tilde{p}$ need not be small compared with $\rho_0(y)$ and $p_0(t)$ respectively. Then (11a)–(11c) become

$$\frac{\partial\tilde{\rho}}{\partial t} + u_i\frac{\partial\tilde{\rho}}{\partial x_i} + v\frac{d\rho_0}{dy} = -(\rho_0(y)+\tilde{\rho})D(x_j, t), \tag{14a}$$

$$\frac{\partial u_i}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x_i}(u_j u_j) - \varepsilon_{ijl}u_j\omega_l = -\frac{1}{\rho_0+\tilde{\rho}}\left[\frac{\partial p}{\partial x_i} - k_i g\tilde{\rho}\right], \tag{14b}$$

$$\frac{\partial}{\partial x_i}\left(\frac{1}{\rho_0+\tilde{\rho}}\frac{\partial\tilde{p}}{\partial x_i}\right) = -\left[\frac{\partial}{\partial y}\left(\frac{g\tilde{\rho}}{\rho_0+\tilde{\rho}}\right) + \frac{\partial D}{\partial t} + \frac{1}{2}\frac{\partial^2}{\partial x_i^2}(q^2) - \frac{\partial}{\partial x_i}\varepsilon_{ijl}u_j\omega_l\right], \tag{14c}$$

where $v = u_2$ and $q^2 = u_i u_i$, and the boundary condition (12b) becomes

$$n_i\frac{\partial\tilde{p}}{\partial x_i} = \tilde{\rho}gk_i n_i. \tag{15}$$

Finally, we form dimensionless equations using the density $\rho_{00} \equiv \rho_0(0)$, the height of the room H as the length scale and the free fall time $(H/g)^{1/2}$ as the time scale. Then, denoting dimensionless quantities with a hat,

$$\begin{aligned} &\hat{\rho} = \frac{\tilde{\rho}}{\rho_{00}}, \quad \hat{p} = \frac{\tilde{p}}{\rho_{00}gH}, \quad \hat{\rho}_0 = \frac{\rho_0}{\rho_{00}}, \\ &\hat{u}_i = \frac{u_i}{\sqrt{gH}}, \quad \hat{x}_i = \frac{x_i}{H}, \quad \hat{t} = \frac{t}{\sqrt{(H/g)}}. \end{aligned} \tag{16}$$

Equations (13) remain exactly the same in dimensionless form with g set equal to one. Subsequently, in this paper all quantities will be understood to be dimensionless, and the hat notation will be dropped. For the dimensionless coordinates, we note that $0 \leqq x \leqq 1/AR$, $0 \leqq y \leqq 1$ and $0 \leqq z \leqq 1/BR$ where $AR \equiv H/L$ and $BR \equiv H/W$. Also,

in the remainder of this paper, the problem will be specialized to two spatial dimensions so that all quantities will be assumed independent of z. This nondimensionalization, while simplifying the form of the equations, does not relate the scale of the induced motion to that of the source. An alternative scheme which does have this feature is derived in the Appendix. The resulting equations are somewhat less convenient for numerical computation, except in the Boussinesq limit when density differences are small. Hence, the variables as defined by (16) will be used throughout the remainder of the paper.

2.2. Finite difference equations.

2.2.1. Basis for the selection of the difference equations. In this section the finite difference equations and the boundary relations for the solution algorithm are presented. It is important first, however, to state the requirements which we used in selecting the scheme. These requirements do not necessarily specify a unique scheme, but restrict greatly the selection. As noted in the previous subsection, the momentum conservation equations provide relations for determining the velocity field, which is a vector field, as a function of time. Forming the difference equations is one criterion. This choice is made to assure that, when the discrete analogues of the divergence and curl are applied to the difference equations, suitable discretized equations for the pressure and the vorticity transport are obtained. Thus we can be certain that the divergence and the curl are calculated correctly in discrete form, the divergence producing the equation for the pressure and the curl producing an evolution equation for vorticity.

A second criterion imposed is that the difference equations provide second-order-accurate approximations to the partial differential equations. This condition is imposed because second-order accuracy is necessary to obtain reasonable spatial and temporal resolution for the hundreds of time steps required to calculate the complete evolution of the flow field in a room fire.

A well-understood buoyant flow field is that produced by internal waves in an ambient stratified environment. Internal waves have for a long time been analyzed and calculated [10], [21]; they are determined as solutions to the linearized partial differential equations of buoyant flow. In addition, internal waves can be expected to arise naturally in a room fire setting when the driving fire has heated and stratified the room air; after the fire has extinguished itself because of vitiated air, for example, pure internal wave modes will exist. An additional criterion which was imposed on the selection of the difference scheme was that it accurately reproduce internal-wave modes in an enclosure: if the difference scheme does not reproduce this rather simple linear flow field, it is unlikely to reproduce more complicated flows. In [4] the authors examined internal waves and some second-order linear difference equations that reproduce these waves. This analysis determined the difference scheme for all but the nonlinear convective terms in the momentum equations.

In an important paper for numerical weather forecasting [1], Arakawa discusses design of computational schemes for long-term numerical integration of the equations of fluid motion in the case of two-dimensional incompressible flow. The major thrust of his paper is a derivation of second-order accurate spatial difference schemes which eliminate the nonlinear computational instability first noted by Phillips [13]. Arakawa emphasizes that for two-dimensional, incompressible flow, the discrete approximations to quadratic forms of dependent variables, such as the velocity squared or the vorticity squared (or both), must be conserved when the continuous variables are, and he uses this constraint to select three difference approximations which are acceptable. Another

criterion imposed in the selection of our difference scheme is that it reduce to one of Arakawa's acceptable, or stable, schemes with respect to the nonlinear computational instability when the flow is incompressible. The scheme chosen is denoted J_3 by Arakawa [2] and conserves energy in the incompressible case. It can be obtained by differencing the vector invariant form of the momentum equations.

Finally, we wanted the difference scheme to be easily generalized to three-dimensional flow configurations. For a model of buoyant flow driven by a fire to be successful, it must be able to calculate three-dimensional flows. The scheme selected and presented below satisfies all of the criteria stated above.

2.2.2. The equations. The two-dimensional rectangular enclosure in dimensionless variables is covered with a uniform spatial grid for the finite difference scheme. There are I mesh cells in the x-direction and J mesh cells in the y-direction. Along the edges of this basic mesh, the two components of the vector velocity (u, v) and single surviving component of the vector vorticity $\omega = \partial v/\partial x - \partial u/\partial y$ are defined. At the center points of the basic grid cells, the scalar quantities such as density ρ and pressure p are defined. In Fig. 2 a typical mesh cell is shown, illustrating where all of the dependent variables in the finite difference scheme are defined relative to the cell.

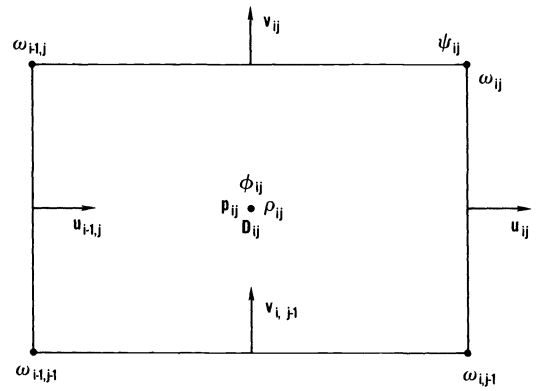


FIG. 2. *A typical mesh cell, with center located at* $x = (i - \frac{1}{2})\,\delta x$ *and* $y = (j - \frac{1}{2})\,\delta y$, *illustrating where all dependent variables for the finite difference scheme are defined.*

The following discretely evaluated functions will denote approximations to the corresponding solutions to (9) and (10):

$$u_{ij}^n \cong u(i\,\delta x, (j - \tfrac{1}{2})\,\delta y, n\,\delta t), \tag{17a}$$

$$v_{ij}^n \cong ((i - \tfrac{1}{2})\,\delta x, j\,\delta y, n\,\delta t), \tag{17b}$$

$$\rho_{ij}^n \cong \rho((i - \tfrac{1}{2})\,\delta x, (j - \tfrac{1}{2})\,\delta y, n\,\delta t), \tag{17c}$$

$$p_{ij}^n \cong p((i - \tfrac{1}{2})\,\delta x, (j - \tfrac{1}{2})\,\delta y, n\,\delta t), \tag{17d}$$

$$D_{ij}^n = D((i - \tfrac{1}{2})\,\delta x, (j - \tfrac{1}{2})\,\delta y, n\,\delta t), \tag{17e}$$

$$\omega_{ij}^n \cong \omega(i\,\delta x, j\,\delta y, n\,\delta t), \tag{17f}$$

where $\delta x = 1/(I \cdot AR)$ and $\delta y = 1/J$ are the mesh cell sizes in the x- and y-directions respectively, and where δt is the time-step size. Such a staggered grid is commonly used for multidimensional finite difference integrations [8].

With this notation, the following set of finite difference equations was used to approximate (11) and boundary conditions (12).

For (14a), $1 \leqq i \leqq I$, $1 \leqq j \leqq J$ and $n \geqq 2$,

$$\tilde{\rho}_{ij}^{n+1} = \frac{1}{1+(\frac{1}{2})D_{ij}^{n}\,\delta t}\{\tilde{\rho}_{ij}^{n-1}(1-(\tfrac{1}{2})D_{ij}^{n}\,\delta t) - 2\delta t(F_{\rho x ij}^{n} + F_{\rho y ij}^{n} + (\tfrac{1}{2})D_{ij}^{n}\rho_0(j))\}, \tag{18}$$

where

$$\tilde{\rho}_{ij}^{n} = \rho_{ij}^{n} - \rho_0(j) = \text{the density difference from the initial density,}$$

$$\rho_0(j) = \exp[-(j-\tfrac{1}{2})\,\delta y / Y_s] = \text{the prescribed ambient initial stratification,} \tag{19}$$

$$Y_s = \text{the stratification length scale.}$$

The flux terms $F_{\rho x_{ij}}^{n}$ and $F_{\rho y_{ij}}^{n}$ for $1 \leqq i \leqq I$, $1 \leqq j \leqq J$ are given by

$$\begin{aligned} F_{\rho x_{ij}}^{n} &= \left(\frac{2\rho_0(j)+\tilde{\rho}_{i+1,j}^{n}+\tilde{\rho}_{i-1,j}^{n}}{4}\right)\left(\frac{u_{ij}^{n}-u_{i-1,j}^{n}}{\delta x}\right) \\ &\quad + \left(\frac{\tilde{\rho}_{i+1,j}^{n}-\tilde{\rho}_{i-1,j}^{n}}{2\delta x}\right)\left(\frac{u_{ij}^{n}+u_{i-1,j}^{n}}{2}\right), \end{aligned} \tag{20a}$$

$$\begin{aligned} F_{\rho y_{ij}}^{n} &= \left(\frac{\rho_0(j+1)+\rho_0(j-1)+\tilde{\rho}_{i,j+1}^{n}+\tilde{\rho}_{i,j-1}^{n}}{4}\right)\left(\frac{v_{ij}^{n}-v_{i,j-1}^{n}}{\delta y}\right) \\ &\quad + \left(\frac{\rho_0(j+1)-\rho_0(j-1)+\tilde{\rho}_{i,j+1}^{n}-\tilde{\rho}_{i,j-1}^{n}}{2\,\delta y}\right)\left(\frac{v_{ij}^{n}+v_{i,j-1}^{n}}{2}\right). \end{aligned} \tag{20b}$$

Equation (18) employs a modification of the second-order accurate central difference (leap frog) temporal discretization. The modification eliminates an instability that would arise if the leap frog scheme had been applied. It affects the undifferentiated term $\tilde{\rho}D(x, y, t)$ in (14) that is well known to lead to a computational instability for ordinary differential equations when leap frog differencing is used [9].

For (14b),

$$u_{ij}^{n+1} = u_{ij}^{n-1} - 2\,\delta t\left\{F_{xij}^{n} + \frac{2}{\delta x}\left(\frac{\tilde{p}_{i+1,j}^{n}-\tilde{p}_{ij}^{n}}{2\rho_0(j)+\tilde{\rho}_{i+1,j}^{n}+\tilde{\rho}_{ij}^{n}}\right)\right\}, \tag{21a}$$

and for $1 \leqq i \leqq I-1$, $1 \leqq j \leqq J$,

$$v_{ij}^{n+1} = v_{ij}^{n-1} - 2\,\delta t\left\{F_{yij}^{n} + \frac{2}{\delta y}\,\frac{\tilde{p}_{i,j+1}^{n}-\tilde{p}_{ij}^{n}+(\tilde{\rho}_{i,j+1}^{n}+\tilde{\rho}_{ij}^{n})\frac{\delta y}{2}}{\rho_0(j+1)+\rho_0(j)+\tilde{\rho}_{i,j+1}^{n}+\tilde{\rho}_{ij}^{n}}\right\} \tag{21b}$$

for $1 \leqq i \leqq I$, $1 \leqq j \leqq J-1$.

The fluxes F_{xij}^{n} and F_{yij}^{n} are defined as follows: for $1 \leqq i \leqq I-1$, $1 \leqq j \leqq J$,

$$F_{xij}^{n} = \frac{1}{2\,\delta x}[(q_{i+1,j}^{n})^2 - (q_{ij}^{n})^2] - \tfrac{1}{2}(v_{\omega_{ij}}^{n}\omega_{ij}^{n} + v_{\omega_{i,j-1}}^{n}\omega_{i,j-1}^{n}), \tag{22a}$$

and for $1 \leqq j \leqq J-1$, $1 \leqq i \leqq I$

$$F_{yij}^{n} = \frac{1}{2\,\delta y}[(q_{i,j+1}^{n})^2 - (q_{ij}^{n})^2] + \tfrac{1}{2}(u_{\omega_{ij}}^{n}\omega_{ij}^{n} + u_{\omega_{i-1,j}}^{n}\omega_{i-1,j}^{n}), \tag{22b}$$

and where

$$\omega_{ij}^n = \frac{v_{i+1,j}^n - v_{ij}^n}{\delta x} - \frac{u_{i,j+1}^n - u_{ij}^n}{\delta y},$$

(22c)
$$v_{\omega_{ij}}^n = \tfrac{1}{2}(v_{ij}^n + v_{i+1,j}^n), \qquad u_{\omega_{ij}}^n = \tfrac{1}{2}(u_{i,j+1}^n + u_{ij}^n),$$
$$(q_{ij}^n)^2 = \left(\frac{u_{ij}^n + u_{i-1,j}^n}{2}\right)^2 + \left(\frac{v_{ij}^n + v_{i,j-1}^n}{2}\right)^2.$$

Note that boundary conditions (12a) on the normal velocities imply that $u_{0,j} = u_{I,j} = 0$ for $1 \leqq j \leqq J$ and $v_{i,0} = v_{i,J} = 0$ for $1 \leqq i \leqq I$. These boundary conditions are applied formally in the expressions for the fluxes

(23)
$$F_{\rho xij}^n, \quad F_{\rho yij}^n, \quad F_{xij}^n \quad \text{and} \quad F_{yij}^n$$

in mesh cells adjacent to boundaries.

The finite difference analogue to (14c) is, for $1 \leqq i \leqq I$ and $1 \leqq j \leqq J$,

$$\begin{aligned}
&\frac{2}{\delta x^2}\left[\frac{\tilde{p}_{i+1,j}^n - \tilde{p}_{ij}^n}{2\rho_0(j) + \tilde{\rho}_{i+1,j}^n + \tilde{\rho}_{ij}^n} - \frac{\tilde{p}_{ij}^n - \tilde{p}_{i-1,j}^n}{2\rho_0(j) + \tilde{\rho}_{ij}^n + \tilde{\rho}_{i-1,j}^n}\right] \\
&\quad + \frac{2}{\delta y^2}\left[\frac{\tilde{p}_{i,j+1}^n - \tilde{p}_{ij}^n}{\rho_0(j+1) + \rho_0(j) + \tilde{\rho}_{i,j+1}^n + \tilde{\rho}_{ij}^n} - \frac{\tilde{p}_{ij}^n - \tilde{p}_{i,j-1}^n}{\rho_0(j) + \rho_0(j-1) + \tilde{\rho}_{ij}^n + \tilde{\rho}_{i,j-1}^n}\right] \\
&\quad = -\frac{D_{ij}^{n+1} - D_{ij}^{n-1}}{2\,\delta t} + \frac{F_{x,i-1,j}^n - F_{x,ij}^n}{\delta x} + \frac{F_{yi,j-1}^n - F_{yij}^n}{\delta y} \\
&\quad\quad - \frac{1}{\delta y}\left[\frac{\tilde{\rho}_{i,j+1}^n + \tilde{\rho}_{ij}^n}{\rho_0(j+1) + \rho_0(j) + \tilde{\rho}_{i,j+1}^n + \tilde{\rho}_{ij}^n} - \frac{\tilde{\rho}_{ij}^n + \tilde{\rho}_{i,j-1}^n}{\rho_0(j) + \rho_0(j-1) + \tilde{\rho}_{ij}^n + \tilde{\rho}_{i,j-1}^n}\right].
\end{aligned} \tag{24}$$

This difference equation for the pressure arises formally from applying the finite difference analogue of the divergence operator to (21) and noting that the finite difference divergence of the velocity field satisfies the equation

(25)
$$\frac{u_{ij}^n - u_{i-1,j}^n}{\delta x} + \frac{v_{ij}^n - v_{i,j-1}^n}{\delta y} = D_{ij}^n.$$

(This equation is the difference approximation to (5).)

The boundary conditions (15) in discrete form become

(26a)
$$\tilde{p}_{0,j}^n = \tilde{p}_{1,j}^n, \qquad \tilde{p}_{I,j}^n = \tilde{p}_{I+1,j}^n \quad \text{for } 1 \leqq j \leqq J;$$

(26b)
$$\tilde{p}_{i,1}^n - \tilde{p}_{i,0}^n = \tfrac{1}{2}\delta y(\tilde{\rho}_{i,1}^n + \tilde{\rho}_{i,0}^n), \qquad \tilde{p}_{i,J+1}^n - \tilde{p}_{i,J}^n = -\tfrac{1}{2}\delta y(\tilde{\rho}_{i,J+1}^n + \tilde{\rho}_{i,J}^n) \quad \text{for } 1 \leqq i \leqq I.$$

Although $\tilde{\rho}_{i,0}^n$ and $\tilde{\rho}_{i,J+1}^n$ appear in the boundary conditions (26b), they also appear in (24) with $j = 1, J$ in the same combination. As a result, $\tilde{\rho}_{i,0}^n$ and $\tilde{\rho}_{i,J+1}^n$ never need to be specified to obtain a solution to (24) and the boundary conditions (26). Equation (24) and boundary conditions (26) constitute a singular linear algebraic system of equations. When (24), with boundary conditions (26) incorporated, is summed, the left-hand side sums to zero, demonstrating that all of the equations are not linearly independent. Also, the last three terms on the right-hand side sum to zero, producing the requirement that the double sum over $(D_{ij}^{n+1} - D_{ij}^{n-1})/2\,\delta t$ must vanish.

The vanishing of the sum of the left-hand sides of (24) and (26) is the discrete analogue of the fact that the left-hand side of the integral of (6) over the room volume vanishes when the boundary conditions $\mathbf{u} \cdot \mathbf{n} = 0$ are applied. The vanishing of the sum of the right-hand sides of (24) and (26) is the corresponding discrete analogue for the

requirement on the integral of the right-hand side of (6). This requirement, that $\sum\sum D_{ij}$ must vanish at each time level, is the discrete analogue of (11d).

Examination of (28) for D_{ij}^n shows that it has been chosen so that its double sum over all mesh points vanishes, and that the condition which must be satisfied to allow this choice produces (29) for the mean pressure. Therefore, the singular linear algebraic system is seen to be consistent and thus to allow a solution. The solution is made unique by specifying that the double sum over all mesh points of $\tilde{p}_{ij}^n$ is zero. This is tantamount to specifying that P_0 is literally the mean pressure in the room, with $\tilde{p}_{ij}^n$ the perturbation about the mean. Details of the algorithm used to solve (24) and (26) for $\tilde{p}_{ij}^n$ are presented in [11].

The heat source has been chosen to have the form

$$Q_{ij}^n = \hat{Q}_{ij} f^n, \tag{27a}$$

$$\hat{Q}_{ij} = \left(\frac{\beta}{\pi}\right)^{1/2} \lambda \exp\left[-\beta(x_i - x_c)^2 - \lambda y_j\right], \tag{27b}$$

$$x_i = (i - \tfrac{1}{2})\,\delta x, \qquad y_j = (j - \tfrac{1}{2})\,\delta y, \tag{27c}$$

$$f^n = Q_0 \tanh A t^n, \tag{27d}$$

$$t^0 = 0, \qquad t^n = \sum_{n'=0}^{n-1} \delta t^{n'}. \tag{27e}$$

Hence, the discrete divergence of the velocity field becomes

$$D_{ij}^n = \frac{1}{\gamma p_0^n}\left[(\gamma - 1)\hat{Q}_{ij} - K\right] f^n, \tag{28a}$$

where

$$K = \frac{\gamma - 1}{IJ} \sum_{i=1}^{I} \sum_{j=1}^{J} \hat{Q}_{ij}, \tag{28b}$$

and the mean background pressure is found from the difference equation

$$p_0^{n+1} = p_0^{n-1} + K f^n 2\,\delta t^n, \tag{29}$$

with $p_0^0 = p_0^1 = 1$ since $f^0 = 0$.

Since difference equations (18) and (21) are three-level, second-order schemes (leap frog) in time, a starting procedure is needed. The following first-order, explicit scheme was used to start the computation and to restart when a change in the time step had been made:

$$u_{ij}^{n+1} = u_{ij}^n - \delta t\left\{F_{xij}^n + \frac{2}{\delta x}\left(\frac{\tilde{p}_{i+1,j}^n - \tilde{p}_{ij}^n}{2\rho_0(j) + \tilde{\rho}_{i+1,j}^n + \tilde{\rho}_{ij}^n}\right)\right\}, \tag{30a}$$

$$v_{ij}^{n+1} = v_{ij}^n - \delta t\left\{F_{yij}^n + \frac{2}{\delta y}\left[\frac{\tilde{p}_{i,j+1}^n - \tilde{p}_{ij}^n + (\tilde{\rho}_{i,j+1}^n + \tilde{\rho}_{ij}^n)\dfrac{\delta y}{2}}{\rho_0(j+1) + \rho_0(j) + \tilde{\rho}_{i,j+1}^n + \tilde{\rho}_{ij}^n}\right]\right\}, \tag{30b}$$

$$\tilde{\rho}_{ij}^{n+1} = \tilde{\rho}_{ij}^n - \delta t\{\tfrac{1}{2}[\rho_0(j) + \tilde{\rho}_{ij}^n] D_{ij}^n + F_{\rho_{xij}}^n + F_{\rho_{yij}}^n\}, \tag{31a}$$

$$p_0^{n+1} = p_0^n + K f^n\,\delta t. \tag{31b}$$

When starting, (31a, b) are used to obtain p_0^{n+1} and $\tilde{\rho}_{ij}^{n+1}$. Then $\tilde{p}_{ij}^n$ is obtained from (24) with $(D_{ij}^{n+1} - D_{ij}^{n-1})/2\,\delta t$ replaced by $(D_{ij}^{n+1} - D_{ij}^n)/\delta t$. With this solution for $\tilde{p}_{ij}^n$,

(30a, b) are used to obtain u_{ij}^{n+1} and v_{ij}^{n+1}. The starting procedure is completed (and two levels of all dependent variables have been obtained) by solving (24) with n replaced by $n+1$ throughout. Subsequent time steps are taken in a straightforward fashion with the density and velocity components being advanced through (18) and (21), and the pressure being updated through (24).

The linear stability of the algorithm is the only other consideration for discussion. A linear stability analysis of (18) for the density shows that the time step δt must satisfy the following condition for stability:

$$\delta t^n \leqq \max_{\substack{1\leqq i\leqq I \\ 1\leqq j\leqq J}} \left\{ (D_{ij}^n)^2 + \left[\frac{|U_{ij}^n|}{\delta x} + \frac{|V_{ij}^n|}{\delta y} \right]^2 \right\}^{-1/2}, \tag{32}$$

where

$$U_{ij}^n \equiv \tfrac{1}{2}(u_{ij}^n + u_{i-1,j}^n), \qquad V_{ij}^n = \tfrac{1}{2}(v_{ij}^n + v_{i,j-1}^n).$$

When the stability condition (32) is not satisfied by a time step, the time step δt^n is halved. Then the time-marching algorithm is restarted using the last time-level values as initial conditions. A first-order time step is taken and then leap frog is resumed.

When the finite difference analogue of the curl (see (22)) is applied to the difference equations for the velocity components (21), a discretized form of the equation for the vorticity transport (10) is formed. A linear stability analysis of the difference equation yields exactly the same form for the stability criterion as that found above for the density equation. Reference to Fig. 2b shows that the density and vorticity are evaluated at different points in the mesh, however, and therefore the divergence D and the velocity components U and V are to be evaluated at different points than those used in (32). To account for the difference in the stability criterion implied by the different mesh location points, in all calculations performed using the algorithm described above, the time step was chosen to be less than or equal to 0.8, the maximum value found for the right-hand side of (32).

3. Example calculations. The algorithm described in § 2 has been used to compute solutions to the buoyant-flow equations for a heat source centered along the floor in a square enclosure. Other calculations will be reported in a companion paper.

In Fig. 3 contours of constant temperature (isotherms) are shown at dimensionless time 2.0 for a volumetric heat source centered along the floor in a square room. The rate of heat added per unit volume is largest along the floor at the center of the room and decreases in a Gaussian fashion with horizontal distance from the center and exponentially with height above the floor: the dependence of the heat source upon position in the room is given by (27) with $x_c = 0.5$. The heat source is "turned on" slowly according to (27a) and (27d) asymptoting to full strength around $t = 10.0$. At this early time the problem is still linear; the flow velocities are sufficiently small that convection is unimportant, and the temperature increase in the fluid is directly proportional to the volumetric rate of heat added. Therefore, the isotherms are also contours along which the volumetric heat-addition rate is constant. (These contours can be seen to be parabolas by examination of (27b), which describes the spatial dependence of the heat source selected for these computations.) These computations were performed on a spatial mesh of $I = J = 31$; the tick marks along the boundary of the enclosure show the mesh cell spacing.

At time 11.5 (Fig. 4), a buoyant thermal has developed, giving the appearance of a mushroom cloud. The temperature has increased and the density has decreased

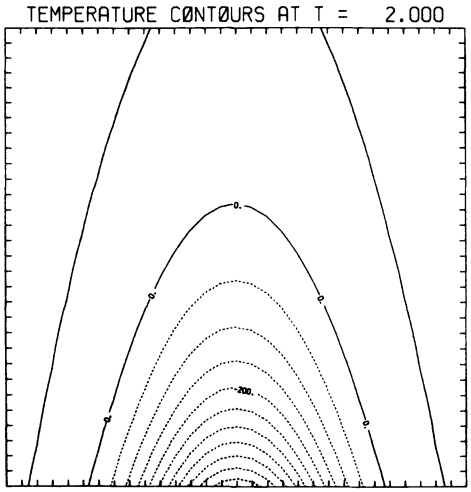


FIG. 3. *Contours of constant temperature at dimensionless time* $t = 2.0$ *in a square enclosure using a* 31×31 *mesh. At this early time convection is unimportant, and isotherms reflect contours of constant volumetric heat addition.*

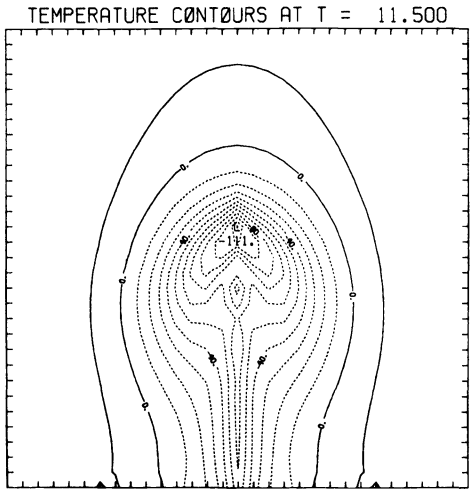


FIG. 4. *Contours of constant temperature at dimensionless time* $t = 11.5$ *in a square enclosure using a* 31×31 *mesh.*

where heating has occurred. The heated fluid has become lighter than its surroundings and begins to rise due to buoyancy. By continuity, surrounding fluid begins to be drawn into the region of the heat source, and the isotherms, therefore, appear to be pinched off at the bottom (near the center of the heat source). Two vortices of equal and opposite strength located on the two halves of the heat source have developed and have begun to rise with the fluid, being convected out of the region of primary heating. The buoyant thermal intensifies in strength as shown in the plot at time 13.5 (Fig. 5), until the thermal hits the ceiling, as shown in Fig. 6, time 15.5, and begins to spread. Inside the plume a distinctly periodic structure has begun to develop, as can be seen vividly in Fig. 5; here, progressing up the plume along its centerline, one finds a local low first, then a periodic sequence of local highs and lows.

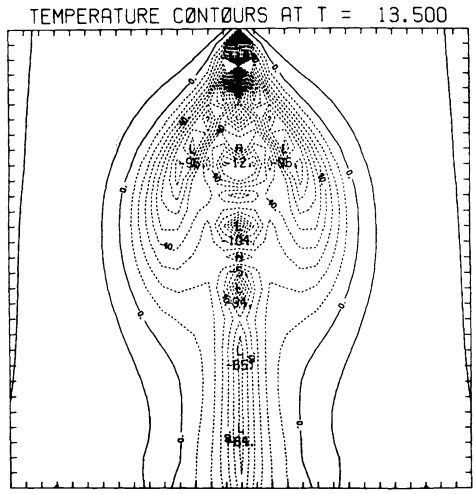


FIG. 5. *Contours of constant temperature at dimensionless time* $t = 13.5$ *in a square enclosure using a* 31×31 *mesh.*

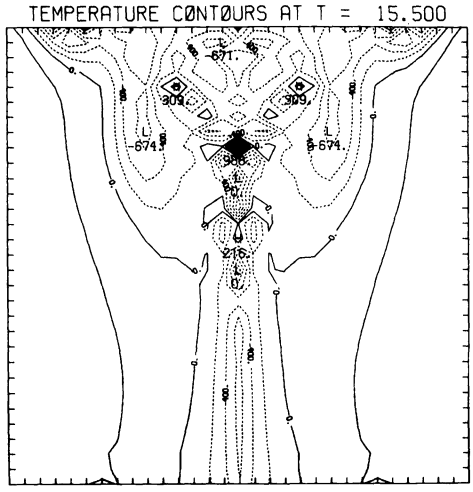


FIG. 6. *Contours of constant temperature at dimensionless time* $t = 15.5$ *in a square enclosure using a* 31×31 *mesh.*

The heated gases spread along the ceiling and fill the room from the top down, as shown in Figs. 7 and 8. This physical behavior is exactly what is observed in room-fire tests and in other experimental observations of heating in enclosures. The symmetry about the centerline of the room displayed in these computations is some measure of the accuracy with which they were performed: the heat source is placed symmetrically, but the computations were performed as if no symmetry existed. To assess the resolution of the computed results shown in Figs. 3–8, the flow field was computed again using a larger number of mesh points: $I = 63$, $J = 64$. Selected plots from this larger computation are shown in Figs. 9–11.

These plots demonstrate that the large-scale features determined by the 31×31 computation are correct and agree with those determined from the larger computation to within about ten percent. The results from the larger computation are characterized

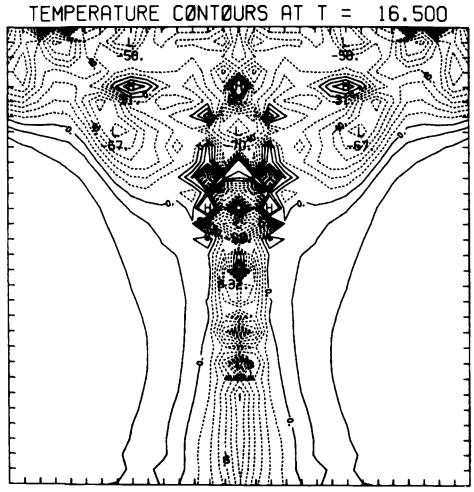


FIG. 7. *Contours of constant temperature at dimensionless time* $t = 16.5$ *in a square enclosure using a* 31×31 *mesh.*

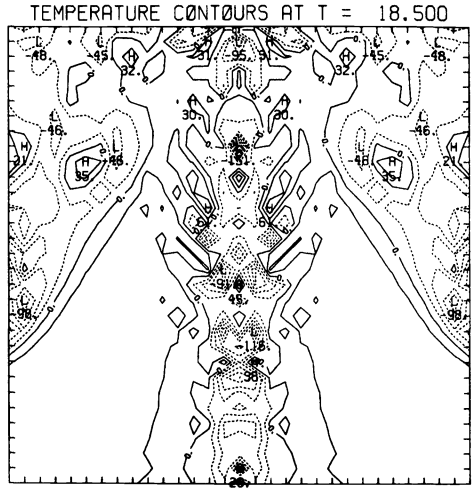


FIG. 8. *Contours of constant temperature at dimensionless time* $t = 18.5$ *in a square enclosure using a* 31×31 *mesh.*

by smoother isotherms and more detailed structures because of the greater resolution. We note that the time required for the buoyant thermal to reach the ceiling is about ten percent longer in the 63×64 results than in the 31×31 results, again apparently because of the greater spatial (and temporal) resolution of the larger computation.

Detail on a length scale of the order of one or two mesh cells must be disregarded because the computations cannot resolve such detail. On the other hand, features with a larger scale can be interpreted. The spatially periodic behavior in the plume noted above is a feature which requires some discussion. The starting thermal and the plume induced by a heat source in an enclosure are a result of physical instability of the flow field. In addition, in the introduction, we discussed the fact that this fluid model was one in which viscosity has been ignored, and therefore it could be considered to result from the Navier–Stokes equations in the limit of very large Grashof number (roughly Reynolds number squared).

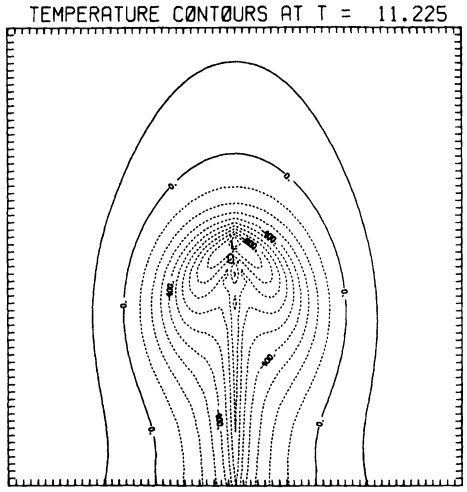


FIG. 9. *Contours of constant temperature at dimensionless time* $t = 11.225$ *in a square enclosure using a* 63×64 *mesh.*

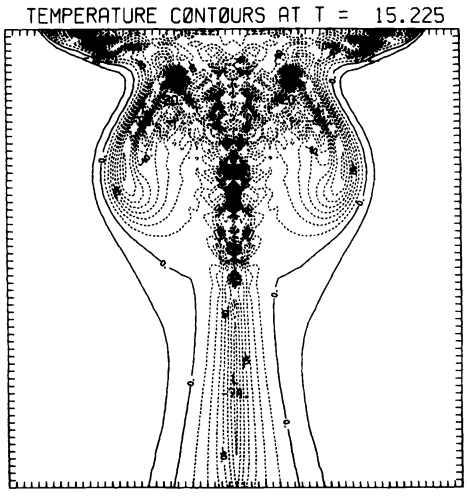


FIG. 10. *Contours of constant temperature at dimensionless time* $t = 15.225$ *in a square enclosure using a* 63×64 *mesh.*

Over the last several years, starting with the pioneering work of Brown and Roshko [5], there has been a reexamination of the meaning of turbulence in shear flows. There had been a growing realization that turbulence is not satisfactorily described in terms of velocity correlations and their corresponding spectra only. Rather, there are distinct coherent vortex structures in shear flows, and Brown and Roshko vividly demonstrated the existence of these coherent structures in turbulent shear flows using shadowgraphs to visualize the flow field. In particular, among many other interesting features, Brown and Roshko found that large scale coherent structures of the same type existed in a shear layer independent of the value of the Reynolds number provided only that the Reynolds number is large enough to have turbulence. Subsequent studies in other flows (see Roshko [19] for some of the references) have shown that organized structures exist in these flows also. In addition, recent theoretical

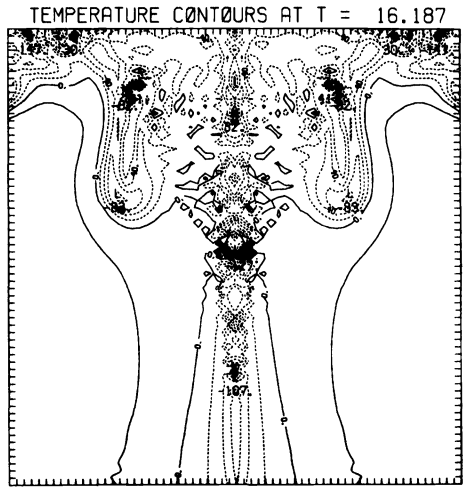


FIG. 11. *Contours of constant temperature at dimensionless time* $t = 16.187$ *in a square enclosure using a* 63×64 *mesh.*

studies have shown that many of the features of the large-scale coherent structures observed in high Reynolds number flows can be described by vortex structures satisfying Euler's equations.

The spatially periodic structures calculated in the starting thermal and the plume have been found to be vortices of alternating sign produced in the heating region and convected out by buoyancy. These vortices increase in strength with height above the floor and occur in antisymmetric pairs with respect to the centerline of the room. We interpret these structures as analogous to the large scale coherent structures observed in turbulent shear flows. In addition, because these vortices are convected with the buoyant flow, the spatial periodicity is translated into a temporal frequency: at any point within the plume, each of the dependent variables oscillates with a frequency related to the rate at which vortices are generated and convected away. Experiments, both at the National Bureau of Standards and elsewhere [22], have demonstrated qualitatively the same feature: namely, buoyant "puffs" or regular upwellings, followed by short quiescent periods, produced at a frequency determined by the experimental arrangement. The frequency of these puffs is also found to agree with the frequency predicted by these calculations.

Appendix. Alternate nondimensional variables. Consider the dimensional system of (1) and (4) written in the form:

$$\frac{\partial \rho}{\partial t} + u_i \frac{\partial \rho}{\partial x_i} + \rho\left(\frac{\gamma - 1}{1}\right)\frac{1}{p_0}\left[Q - \frac{1}{V}\int Q\, dV\right] = 0,$$

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j}\right) + \frac{\partial \tilde{p}}{\partial x_i} + [\rho - \rho_0(y)]g k_i = 0, \tag{A1}$$

$$\frac{\partial u_i}{\partial x_i} = \frac{\gamma - 1}{\gamma}\frac{1}{p_0}\left[Q - \frac{1}{V}\int Q\, dV\right].$$

Here ρ and u_i are the density and velocity components as defined in the text. The quantities $\rho_0(y)$ and $\tilde{p}$ are respectively the initial density stratification in the vertical (y) direction and the difference between the local pressure and the hydrostatic pressure

at the height in question. This pressure difference which affects the fluid motion, is the quantity $\tilde{p}$ defined in (13b) of the text. The heat source Q is prescribed in the form

$$Q = \frac{Q_0 f(t)}{l_x l_y l_z} \frac{1}{\sqrt{\pi}} e^{-(x-x_c)^2/l_x^2 - y/l_y}. \tag{A2}$$

Note the slight difference in notation from (27b) of the text.

We now seek to introduce nondimensional variables, denoted with an asterisk (*), that are close to those defined in the text but which reflect the strength of the heat source Q_0. To this end, we define the following quantities:

$$\begin{aligned} &x_i = Hx_i^*, \qquad u_i = Uu_i^*(x_b^*, t^*) \\ &t = \frac{H}{U} t^*, \qquad P_0 = P_a P_0^*(t^*), \\ &\tilde{P} = \rho_0(0) U^2 p^*(x_k^*, t^*), \\ &\rho = \rho_0(0)\{1 + \beta\rho^*(x_k^*, t)\}, \\ &\rho_0(y) = \rho_0(0)\{1 + \beta\rho_0^*(y^*)\}. \end{aligned} \tag{A3}$$

Here, P_a is the undisturbed ambient pressure, $\rho_0(0)$ the ambient density at the floor, and H the height of the enclosure. The reference velocity U and the dimensionless density ratio β are as yet undefined. These two quantities are now determined by requiring that in the Boussinesq limit, when the density ratio $\beta \to 0$, all nongeometric parameters disappear from the problem. This leads to the following equations for U and β:

$$\frac{U^2}{gH} = \beta, \qquad \frac{\beta U}{H} = \frac{Q_0}{H^2 l_z P_a}. \tag{A4}$$

This yields a velocity scale which differs from that employed in the text by a factor $\sqrt{\beta}$, and a dimensionless density ratio given by

$$\beta = \frac{1}{gH}\left\{\frac{Q_0}{l_z}\frac{g}{P_a}\right\}^{2/3}. \tag{A5}$$

Finally, the equations of motion (A1) become

$$\begin{aligned} &\frac{\partial \rho^*}{\partial t} + u_i^* \frac{\partial \rho^*}{\partial x_i^*} + (1 + \beta\rho^*) D^* = 0, \\ &(1 + \beta\rho^*)\left(\frac{\partial u_i^*}{\partial t^*} + u_k^* \frac{\partial u_i^*}{\partial x_k^*}\right) + \frac{\partial p^*}{\partial x_i^*} + k_i(\rho^* - \rho_0^*) = 0, \\ &\frac{\partial u_i^*}{\partial x_i^*} = \beta D^*, \\ &D^* = \frac{\gamma - 1}{\gamma} \frac{1}{p_0^*}\left[Q^* - \frac{H^2 l_z}{V} f(t^*)\right], \\ &Q^* = \frac{f(t^*)}{\sqrt{\pi}} \frac{H^2}{l_x l_y} \exp\left[-\left(\frac{H}{l_x}\right)^2 (x^* - x_c^*)^2 - \left(\frac{H}{l_y}\right) y^*\right]. \end{aligned} \tag{A6}$$

Note that when β is $O(1)$; i.e., when there are significant density variations, the nondimensionalization is for all practical purposes the same as that used in the text. The most significant feature of this derivation is (A5), which determines β and hence the conditions under which a non-Boussinesq model is necessary.

Acknowledgments. The authors wish to acknowledge useful discussions with J. Oliger and M. Ciment. Also, M. Knapp-Cordes has been helpful with many computational aspects of the work.

REFERENCES

[1] A. ARAKAWA, *Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part* I, J. Comp. Phys., 1 (1966), pp. 119–143.

[2] A. ARAKAWA AND V. R. LAMB, *Computational design of the basic dynamical processes of the UCLA general circulation model*, in General Circulation Models of the Atmosphere, J. Chang, ed., Methods in Computational Physics 17, Academic Press, New York, 1977, pp. 173–265.

[3] G. K. BATCHELOR, *The conditions for dynamical similarity of motions of a frictionless perfect-gas atmosphere*, Quart. J. Roy. Meteor. Soc., 79 (1953), pp. 224–235.

[4] H. R. BAUM AND R. G. REHM, *Finite difference solutions for internal waves in enclosures*, Report, National Bureau of Standards, Washington, DC, in preparation.

[5] G. L. BROWN AND A. ROSHKO, *On density effects and large structure in turbulent mixing layers*, J. Fluid Mech., 64 (1974), pp. 775–816.

[6] P. CONCUS AND G. GOLUB, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equation*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[7] P. CONCUS, G. COLUB AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[8] F. H. HARLOW AND A. A. AMSDEN, *Fluid dynamics, a LASL monograph*, Report LA4700, Los Alamos Scientific Laboratory, Los Alamos, NM, 1971.

[9] H. KREISS AND J. OLIGER, *Methods for the approximate solution of time dependent problems*, GARP Publication Series 10, Global Atmospheric Research Programme, February 1973.

[10a] H. LAMB, *Hydrodynamics*, 6th ed., Dover, New York, 1932, pp. 378–380.

[10b] L. PRANDTL, *Essentials of Fluid Dynamics*, Blackie and Son, London, 1952, pp. 374–377.

[11] J. LEWIS AND R. G. REHM, *The numerical solution of a nonseparable elliptic partial differential equation by preconditioned conjugate gradients*, NBS J. Res., 85 (1980), pp. 367–390.

[12] B. MCCAFFREY, *Purely buoyant diffusion flames: some experimental results*, Report NBSIR 79-1910, National Bureau of Standards, Washington, DC, 1979.

[13] N. A. PHILLIPS, *An Example of Non-Linear Computational Instability, The Atmosphere and Sea in Motion*, Rockefeller Inst. Press in association with Oxford University Press, New York, 1959, pp. 501–504.

[14] T. A. PORSCHING, *A finite difference method for thermally expandable fluid transients*, Nuclear Sci. and Engrg., 64 (1977), pp. 177–186.

[15] J. QUINTIERE, *Growth of fire in building compartments*, in Fire Standards and Safety, A. F. Robertson, ed., ASTM STP 614, American Society for Testing and Materials, New York, 1977, pp. 131–167.

[16] R. G. REHM AND H. R. BAUM, *The equations of motion for thermally driven, buoyant flows*, NBS J. Res., 83 (1978), pp. 297–308.

[17] R. G. REHM, H. R. BAUM, D. M. CORLEY AND P. D. BARNETT, *Finite difference calculations of buoyant convection in an enclosure, part* II: *Verification of the basic algorithm*, Report, National Bureau of Standards, Washington, DC, in preparation.

[18] P. J. ROACHE, *Computational Fluid Dynamics*, Hermosa Publishers, Albuquerque, NM, 1976.

[19] A. ROSHKO, *Structure of turbulent shear flows: A new look*, AIAA J., 14 (1976), pp. 1349–1357.

[20] P. SWARZTRAUBER AND R. SWEET, *Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations*, NCAR-TN/IA-109, 1975.

[21] J. S. TURNER, *Buoyancy Effects in Fluids*, Cambridge Univ. Press, 1973, Chapt. 2; also G. B. Whitham, *Linear and Nonlinear Waves*, John Wiley, New York, 1974, pp. 421–423.

[22] E. ZUKOWSKI, T. KUBOTA AND B. CETEGEN, *Entrainment in fire plumes*, Fire Safety J., 3 (1980/81), pp. 107–121.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 1, March 1983

0196-5204/83/0401-0010 $01.25/0

AUTOMATIC SELECTION OF METHODS FOR SOLVING STIFF AND NONSTIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS*

LINDA PETZOLD†

Abstract. This paper describes a scheme for automatically determining whether a problem can be solved more efficiently using a class of methods suited for nonstiff problems or a class of methods designed for stiff problems. The technique uses information that is available at the end of each step in the integration for making the decision between the two types of methods. If a problem changes character in the interval of integration, the solver automatically switches to the class of methods which is likely to be most efficient for that part of the problem. Test results, using a modified version of the LSODE package, indicate that many problems can be solved more efficiently using this scheme than with a single class of methods, and that the overhead of choosing the most efficient methods is relatively small.

Key words. stiff, nonstiff, ordinary differential equations, initial value problems

1. Introduction. This paper describes a scheme for automatically determining whether an initial value problem $dy/dt = f(y, t)$, $y(t_0) = y_0$, can be solved more efficiently using a class of methods suited for nonstiff problems or a class of methods designed for stiff problems. The decision is based on information which is available at the end of each step of the integration, so that if a problem changes character (i.e., from nonstiff to stiff or vice versa) in the interval of integration, the solver automatically switches to the class of methods which is likely to be most efficient for that part of the problem.

This scheme is useful in several different situations. The user of an ODE solver may not know whether his problem is stiff, or the solver may be called by another code (a package for solving partial differential equations or boundary value problems, for example) where the character of the problem is not known in advance. With the technique described here, the most effective family of methods is chosen automatically. Moreover, many "stiff" problems are often nonstiff in the initial phase, or transient. Integrating through the transient with stiff methods (by a "stiff method," we mean a method designed for stiff problems) is very expensive, whereas nonstiff methods are much better suited for this purpose. As the problem becomes stiff, the code can eventually switch to the stiff methods. In general, problems may be stiff in some intervals and nonstiff in others. This scheme selects the methods that are most efficient for each interval.

Several techniques have been reported in the literature (Shampine [5], [6], [7]) for detecting stiffness. Our objective here is somewhat different, because in addition to detecting stiffness we actually expect the code to shift to the methods which are most appropriate for the problem. Some of the ideas in these papers (especially Shampine [7]) have influenced the approach that was taken here. Shampine [8] outlines a scheme for automatically altering the solution algorithm based on stiffness of the problem for codes based on implicit A-stable formulas. Our scheme is somewhat more general than his in that we automatically select a method from a class of methods where all of the members are not necessarily A-stable.

The basic principles underlying the switching technique are quite simple and are explained in the next section. Some of the difficulties involved in implementing this scheme are described in § 3, along with the approaches we have taken to resolve these problems.

* Received by the editors December 3, 1980, and in revised form September 1, 1981.
† Applied Mathematics Division, Sandia National Laboratories, Livermore, California 94550.

This scheme has been implemented using Adams methods (orders 1–12 with functional iteration to solve the corrector equation) as the family of nonstiff methods, and backward differentiation formulas (BDF) (orders 1–5 with modified Newton iteration) as the family of stiff methods. These seem to be good choices because general purpose codes based on these methods are among the most effective codes available for solving nonstiff and stiff problems, respectively [1], [2], [4]. We have modified the code LSODE, an updated version of the GEAR package written by A. C. Hindmarsh [3], to automatically switch between Adams methods and BDF whenever it is appropriate. In order to illustrate the basic operation of the scheme, some care has been taken to keep the changes to the code to a minimum and to avoid exploiting any properties that are specific to this particular implementation (for example, error estimates based on the Nordsieck vector). It is clear, however, that by exploiting some of the features specific to a particular code, somewhat greater reliability and efficiency of this scheme could be achieved. In addition to a block of code which actually makes the decision and implements the change in method families, several other changes to the code were made so that reliable information about the problem could be obtained at each step. These changes occur in the stepsize and order control logic and in the corrector iteration in the Adams part of the code. In § 4 we describe the results of applying this code to some test problems.

The overhead of making the switching decisions is small. For a truly nonstiff problem, where the method family will never be changed (we always start out using nonstiff methods because they are much cheaper per step and many stiff problems are really nonstiff in the beginning of the interval), this code runs nearly as fast as the unmodified LSODE, using the Adams option with functional iteration. For problems which are stiff in some regions of the interval of integration and nonstiff in others, this code can be much faster than using either Adams methods or BDF over the entire interval. Because the cheaper nonstiff methods are used during the transient of a stiff problem, significantly fewer Jacobian evaluations are required for many stiff problems.

2. Basic strategy. As the integration proceeds our objective is to choose the family of methods which will solve a given problem most efficiently. This decision is made by comparing the method that is currently being used to the method that would be used if the code switched to the other family of methods. To compare the methods, we consider the stepsize that each method could use on the next step, and the cost per step of each method. Since one step of a nonstiff method is typically much cheaper than one step of a stiff method, we should favor using the nonstiff method as long as the stepsizes it uses are not very much smaller than the stepsizes that would be used by the stiff method.

What controls the stepsize for each method? For the nonstiff method several considerations affect the stepsize. First, we must choose a stepsize so that the formula is accurate over the next step (so that a norm of the local truncation error is less than some constant ε). If the code uses functional iteration to solve the corrector equation, the stepsize must be small enough so that the iteration will converge rapidly. Finally, the stepsize must be small enough so that the method will be stable. For the stiff method, the stepsize is chosen so that the formula is accurate over the next step. We will assume that stability and convergence of Newton's method do not restrict the stepsize that could be used by the stiff method. Obviously, this assumption may not always be correct. However, while the stiff method we are currently using may not be stable for the stepsize we would like to use on the next step, another method in

the family of stiff methods probably would be. Hopefully, the order control mechanism would find that method. Skelboe [9] describes an order control strategy to accomplish this; we will not take up this question here. If the stepsize is being controlled by convergence of Newton's method—due, for example, to a very poor approximation to the Jacobian matrix—these assumptions will cause the code to stay with the stiff methods even if they are doing very poorly. We see no good way to avoid this problem.

The first task is to estimate the stepsize that each method could use to achieve the requested accuracy. Let N be the nonstiff method and S the stiff method. If N and S are both linear multistep methods of order q, for example, then we can require the norm of the principal part of the local truncation error to be less than ε. Suppose we are currently using method N with step size h_{CURRENT}, that N is stable for this problem with this stepsize, and that $\|LTE_N\|$ is our estimate for the local truncation error. Then, it is well known that h_N and h_S (the stepsizes that the nonstiff and stiff methods could use on the next step) should satisfy

$$h_N \leqq \left(\frac{\varepsilon}{\|LTE_N\|}\right)^{1/(q+1)} h_{\text{CURRENT}} \tag{1}$$

and

$$h_S \leqq \left(\frac{\varepsilon (C_N/C_S)}{\|LTE_N\|}\right)^{1/(q+1)} h_{\text{CURRENT}}. \tag{2}$$

C_N and C_S are constants depending on the methods N and S.

The stepsize of N may also be affected by considerations such as stability and convergence of functional iteration, so we must find out what effects, if any, these conditions will have. To accomplish this, we need an estimate for $\|\delta f/\delta y\|$ or an estimate of the spectral radius $\rho(\delta f/\delta y)$.

When the stiff method is used, a Jacobian matrix is available and $\|\delta f/\delta y\|$ can be computed directly. The norm used is the matrix norm which is consistent with the vector norm that is used in the code—a weighted norm where the weights depend upon error tolerances. The norm can be computed cheaply (relative to the cost of a matrix factorization) whenever a new Jacobian matrix is formed. Thus, the norm which is available at any given time may correspond to a time several steps back, but it is not likely to be too severely in error because the stiff method reevaluates the Jacobian whenever it changes significantly.

If we are using the nonstiff method, a lower bound for $\|\delta f/\delta y\|$ can be obtained very cheaply during the corrector iteration, using the ideas of Shampine [7]. The bound is formed concurrently with our estimate of the rate of convergence of the iteration. The basic idea is that if the iteration is written as

$$y^{(k+1)} = h\gamma f(y^{(k)}) + \psi,$$

then

$$\frac{\|y^{(k+1)} - y^{(k)}\|}{\|y^{(k)} - y^{(k-1)}\|} \leqq h\gamma \left\|\frac{\delta f}{\delta y}\right\|.$$

The maximum of these ratios,

$$\frac{1}{h\gamma} \frac{\|y^{(k+1)} - y^{(k)}\|}{\|y^{(k)} - y^{(k-1)}\|}$$

obtained over the current step, is a lower bound for $\|\delta f/\delta y\|$. These bounds tend to be quite good (near the spectral radius of $\delta f/\delta y$), although they fluctuate when the dominant eigenvalues of $\delta f/\delta y$ are complex. Unless the difference between two iterates is so small that our estimate would be polluted by roundoff error, we force the nonstiff part of the code to take at least two corrector iterations, in order to generate a lower bound for $\|\delta f/\delta y\|$ on each step.

In any case, suppose a lower bound K for $\|\delta f/\delta y\|$ has been generated or $\|\delta f/\delta y\|$ has been computed directly. Then h_N must be small enough so that functional iteration will converge at a sufficiently rapid rate r, for example $r \leqq \frac{1}{2}$. Thus h_N must satisfy $h_N \gamma \|\delta f/\delta y\| \leqq \frac{1}{2}$, so we will require

$$h_N \leqq \frac{1}{2\gamma K}. \tag{3}$$

Stability also constrains the stepsize for the nonstiff method. If r_q is the radius of the largest half disc contained in the stability region of method N, we must have

$$h_N \rho\left(\frac{\delta f}{\delta y}\right) \leqq r_q,$$

or the computation can become unstable. Thus, we require h_N to satisfy

$$h_N \leqq \frac{r_1}{2K} \tag{4}$$

where K is our lower bound for $\|\delta f/\delta y\|$ and the factor $\frac{1}{2}$ is included so that we can be reasonably sure that h_N would lead to a stable computation (since K is only a lower bound). We actually require h_N to satisfy (4) when computing with the nonstiff method, not just for deciding whether to use the stiff or nonstiff methods. The reasons for this will be discussed in the next section.

Once these estimates have been generated, it is a simple matter to decide whether to use method N or method S. The stepsize that N could use on the next step is the largest h_N that satisfies conditions (1), (3) and (4). The stepsize that S could use is the largest h_S that satisfies (2). Supposing that N is cheaper per step than S, so that we would be willing to take as many as M_+ steps with N for each step that would have to be taken with S, then we will shift to method S (if we are currently using N) if

$$h_S \geqq M_+ h_N. \tag{5}$$

It is important to guard against changing families of methods too frequently, for it might happen that the computation is no longer stable. To avoid this, we stay with method S possibly a little bit longer than what is really optimal; that is, shift from S to N if

$$h_N \geqq M_- h_S. \tag{6}$$

In our code, we have taken $M_- = 1$.

Another factor that is working to prevent the code from shifting back and forth very often is that the constant K which is computed in N is a lower bound for $\|\delta f/\delta y\|$, while in S, $\|\delta f/\delta y\|$ is computed directly. This has a conservative effect in switching from S to N. As a final precaution, we force the code to wait 20 steps after a change in method families before considering a change again. This provides time for the error estimates to settle down after the switch before trying to use them to make another

such decision. The constants M_+ and M_- can be chosen to be functions of the dimension of the system of equations.

3. Implementation considerations. In any practical implementation of a scheme such as the one described above, it is important that the information upon which the code is basing its decisions be reliable, and not misleading. The code should recognize situations where it is not possible to obtain reliable information, and take some appropriate action. In this section we will describe some of the problems involved in ensuring that the information which our scheme requires, namely the local truncation error and a reasonable lower bound for $\|\delta f/\delta y\|$, are reliable, and in recognizing those situations where the information obtained may be misleading.

Several problems with estimating the local truncation error must be dealt with. The first is a problem involving instability. Normally, when the stepsize for the nonstiff method is limited by stability, it tends to oscillate about the largest stable value. (This is explained in detail in Shampine and Gordon [5].) When the stepsize is inside the region of absolute stability of the method, errors are damped. When it is outside the stability region, there is an error growth which causes the error estimates to increase. This eventually brings the stepsize back into the stability region (because the code adjusts the stepsize to keep the error estimate less than ε). In this way, most codes do in some sense detect instability and handle it automatically. When this happens, however, the code has no way of distinguishing whether the error estimates actually reflect the smoothness of the solution or are polluted by terms arising from instability. Thus, instability can make a problem appear to the code to have a solution that is much less smooth than it really is. This is obviously a very undesirable situation for our scheme, which is asking the question after every step, "How smooth is the solution relative to the size of the largest eigenvalues of the problem?"

There would seem to be several ways around this apparent dilemma. The simplest solution is to switch to the stiff methods at the first sign that the stepsize is being limited by stability. Unfortunately, this causes the code to use the stiff methods for many problems that are only marginally stiff and could be solved much more efficiently using the nonstiff methods.

Assuming then that our objective is to use the nonstiff methods as long as the stepsize is not being limited *very* severely by stability, what can we do to ensure that the estimates are not misleading? Because we intend to use the nonstiff method for some time while the problem is stiff, some rather subtle difficulties can occur. For example, even if we are using an A-stable corrector with functional iteration for solving a problem which is becoming stiff, the error estimates may become polluted by terms arising from instability, unless we are very careful about deciding when the corrector iteration has converged. This is because the stability that is of interest here is not the stability of the implicit method by itself (with the corrector solved exactly), but the stability of the method with only a finite, but not necessarily constant, number of corrector iterations.

For example, to illustrate the problems with stability, suppose the trapezoidal method, with functional iteration and automatic stepsize control, is used for solving a stiff problem, and that the corrector iteration is terminated when the norm of the difference between two iterates is less than δ (δ is some constant related to the error tolerance ε, like $\delta = \varepsilon/10$). As the problem becomes more and more stiff, the corrector iteration eventually fails to converge, and the stepsize is reduced. This may happen several times, until finally the stepsize is small enough that the iteration may converge, *not* because the iteration is contracting, but because the difference between the prediction and the first correction is less than δ. There are two ways to interpret this

behavior. If the stepsize is limited by convergence of the iteration, then the error estimate must be less than ε—possibly much less—if it is to be an accurate indication of the smoothness of the solution. However, since the error estimate is based on the difference between the predictor and the corrector, and this difference may be accurate only up to the error δ incurred from terminating the corrector iteration early, there is a limitation on how small an error estimate the code can resolve. Another way of seeing this is the following: As long as the difference between the prediction and the first correction is less than δ, the algorithm we are using is in some sense not the trapezoidal method. It is, instead, a prediction followed by one corrector iteration based on the trapezoidal method. This method is not A-stable. Thus, errors are amplified and the error estimate is misleading. These problems can be avoided if 1) the code is forced to take two corrector iterations per step to estimate a rate of convergence, and 2) the step is rejected if the rate of convergence is not rapid enough, *even though the difference between two successive iterates may be quite small.* This is an implicit limitation on the stepsize of the form $h\|\delta f/\delta y\| \leqq C$, for C some constant, because steps are rejected during the corrector iteration that do not satisfy this criterion. A problem with this strategy is that when convergence of the iteration is limiting the stepsize, there are likely to be many rejections of this type, and this is costly.

The solution that we have adopted for these problems of polluted error estimates is to explicitly limit the step size so as to try to ensure stability. Zlatev and Thomsen [11] employ a strategy of this type to avoid repeated step failures, although their code uses a user-supplied estimate of the magnitude of the largest eigenvalue of the Jacobian. The form that this takes in our modification to LSODE is that when a new stepsize and/or order is selected in the nonstiff part of the code, the stepsize that could be used in the next step for each order is computed as the minimum of the stepsize required for accuracy and the stepsize needed to ensure stability. (We require $hK \leqq r_q/2$, where K is our lower bound for $\|\delta f/\delta y\|$, and r_q is the radius of the largest disc contained in the stability region of the Adams PECE method of order q. There is no good reason for using the PECE stability region here—in fact, it is probably more reasonable to use the intersection of Adams–Moulton stability regions with regions where a rapid rate of convergence of the iteration could be obtained. The code is not very sensitive to these numbers, so long as they are the correct order of magnitude, and they decrease monotonically with the order from order 2 or 3 upwards.) Since this code chooses the order which can use the largest stepsize, an effect of this explicit limitation on the stepsize is that when stability is limiting the stepsize, the order is automatically lowered (unless it is already at second order) because the restrictions are less severe for lower order methods.

Sometimes the error estimate cannot be trusted because it may be polluted by roundoff error. This occurs frequently in three different situations: 1) At the very beginning of the computation, the stepsize is often much smaller than what is needed for accuracy, and it takes some time for the code to increase it. During this time, error estimates are quite small, and may be indistinguishable from zero by roundoff. 2) After passing over a discontinuity, the code may be taking very small steps, while the problem after the discontinuity is very smooth. The same problem as in 1) then occurs. 3) With the limitation on the stepsize to ensure stability, as the problem becomes more and more stiff the error estimates become smaller and smaller. If the tolerance is tight, and/or if the constant M_+ in (5) is relatively large, then the error estimates can be driven down to a level which is sensitive to roundoff. The code should switch to the stiff methods in situation (3), but not in (1) or (2).

The problem of detecting when the error estimate may be polluted by roundoff

does not appear to be trivial. We have taken a simple approach to this problem and some further work is necessary on this topic. We say that the estimate is indistinguishable from zero if the norm of the difference between the predictor and the corrector is less than one hundred times the norm of the predictor times the unit roundoff error of the machine. When this condition is detected, then if the last stepsize chosen had to be reduced to ensure stability, we switch from the nonstiff to the stiff methods.

In order to guard against switching back from the stiff methods immediately after passing this test, a similar test in the reverse direction is necessary. We switch from the stiff to the nonstiff methods if $H_N/H_S \geqq M_-$, and the estimated predictor-corrector difference, for the stepsize that the nonstiff method would use, is not so small as to be indistinguishable from zero as measured by the test described above.

The lower bound for $\|\delta f/\delta y\|$ can also be misleading because of roundoff. This has been noted by Shampine [7], and we use a test similar to his to describe whether to form the bound. If $\|y(k+1)-y(k)\| \leqq 100 \cdot u \cdot \|y(0)\|$, the bound is not formed and the corrector is considered to have converged (the iteration is terminated). Since a recent lower bound is important for our scheme, then if the bound has not been generated recently and the last stepsize chosen had to be reduced to ensure stability, we switch from the nonstiff to the stiff methods.

If the error tolerance ε is so small that the norm of the difference between the predictor and the corrector is forced to be smaller than one hundred times the norm of the predictor times the unit roundoff error of the machine, then the error estimates are indistinguishable from zero, and no lower bounds for $\|\delta f/\delta y\|$ are formed. In this situation it is impossible to tell whether the problem is stiff. To avoid these problems, we double ε if $\varepsilon \leqq 100 \cdot u \cdot \|y\|$ at the start of any step.

Another difficulty with lower bounds for $\|\delta f/\delta y\|$ occurs in problems whose dominant eigenvalues have large imaginary parts. For these problems, the lower bounds can fluctuate between values much smaller than the spectral radius of the Jacobian, and much larger. In response to this problem, and in the interests of being conservative, we use the maximum of all lower bounds generated since the last time a change in stepsize or order was considered (at most, $q+2$ steps). If a lower bound has not been generated during that time, the code may decide to switch to the stiff methods. If the switch is not made, the most recent nonzero maximum is used. These fluctuating estimates can cause problems for the stepsize and order selection mechanisms in the code. For instance, the stepsize may be restricted and the order lowered based on an unusually large estimate, when possibly the next time such a change is considered the estimate is much smaller. Since the earlier computation was probably very stable (because of the large estimate restricting the stepsize), the response of the code to the smaller estimate is likely to be to increase the stepsize and raise the order (this is because, without the effects of instability on the error estimates, high order differences of the solution tend to be smaller than low order differences). K. Stewart [10] has suggested using averages of the lower bounds; this looks like a very good idea, but in the case of wildly fluctuating estimates there still seem to be problems. Either an average that heavily weights large values, or the maximum over a large number of past steps would also help to minimize this difficulty. Order selection algorithms now used in nonstiff codes do not appear to be adequate when the stepsize is restricted to ensure stability and/or convergence of functional iteration. Different order selection algorithms should probably be used in this situation.

4. Practical experience. In this section we will first complete the description of the modifications that were made to LSODE, and then describe the results of using the modified code to solve some problems.

A change from one family of methods to the other is considered after every successful step, unless it has been less than twenty steps since the last switch. The test is skipped if the current method is an Adams method of order greater than five. The reasoning behind this is that the code is not likely to be using such a high order method for solving a stiff problem. If the problem is stiff, then stability will be restricting the stepsize, and the order should be lowered rapidly so that soon the test will be made (there is some danger in this logic, but there have been no problems with this in practice).

The method in the other family that we consider switching to is the one which is of the same order as the method that we are currently using. This is in some ways an arbitrary decision, because information is always available to consider any method with order less than or equal to the current order. Most of the time when the switch is made from Adams to BDF, the order is already quite low, so there is not much choice about which BDF to use. When switching from BDF to Adams there should not be any stability problems with the new method, because the stepsize is restricted to ensure stability. There is a potential source of problems when changing from Adams to BDF, however. The stepsize will generally be increased substantially in this direction, and it is possible that for the stepsize chosen, the BDF may be unstable (especially if the problem has eigenvalues near the imaginary axis). If this happens, it is possible that the code could be led into diagnosing that the problem is less smooth than it really is, and would then switch back to the nonstiff methods. This problem has not been encountered in practice.

The actual switch in method families has been implemented in the most obvious way using the Nordsieck data representation. New method coefficients are calculated, and the old Nordsieck vector is used as if that family of methods had been used all along. Thus, in the first step of the "BDF" after using Adams methods, a true BDF is not really being used, because that would require the polynomial represented by the Nordsieck vector to interpolate past values of the solution, whereas the Adams methods use a polynomial whose derivatives agree with the derivatives of the solution at past times. If the switch is done often, this could cause stability problems, but there is no problem if it is done only a few times. There are several devices in the code which have been described earlier designed to prevent it from thrashing between families of methods. Thrashing has not presented a serious problem in our experience.

The norms used in the code were all changed to weighted l_1 norms, and $\|\delta f/\delta y\|$ is computed in the stiff part of the code with the norm which is consistent with the weighted vector l_1 norm. (The weights are the same as the ones used in the unmodified LSODE, and depend on the error tolerances.) The constant M_+ was taken to be five in the tests described below, and $M_- = 5/M_+ = 1$. The test problems are relatively small (all have dimension less than or equal to 51 and most are much smaller than that) so for these problems our algorithm is conservative about diagnosing some problems as stiff. With these parameters and the range of tolerances used in the tests, the code occasionally runs into the roundoff limitations described earlier.

The modified code was tested on the nonstiff DETEST problems [4] and on the stiff DETEST problems [1], [2]. In addition, we solved van der Pol's equation,

$$\begin{aligned} y_1' &= y_2, & y_1(0) &= 2.0, \\ y_2' &= \eta(1-y_1^2)y_2 - y_1, & y_2(0) &= 0.0, \end{aligned} \tag{7}$$

with $\eta = 100.0$ on the interval $[0, 1000]$. This problem was chosen because it alternates between being stiff and nonstiff several times during the interval of integration, so that it is a good illustration of the code's ability to switch back and forth between the

two families of methods. All computations were done in single precision on a CDC 6600 with pure absolute error tolerances. The initial stepsize was determined automatically by the code (using the algorithm in LSODE) in all cases. An initial stepsize of 1.0E-12 was used for all problems. Detailed results for the DETEST problems along with results for unmodified LSODE for the same problems are available from the author. Both codes achieved comparable accuracies for the test problems. Conclusions based on these tests are summarized below.

Very few of the problems of nonstiff DETEST were diagnosed as stiff. The times that this happened the problems were solved in a comparable amount of time or faster (in terms of function evaluations, steps, and execution time) than unmodified LSODE using functional iteration.

On most of the nonstiff problems, the modified code used fewer steps, but more function evaluations, than LSODE with $MF = 10$ (Adams methods with functional iteration). This is mainly a consequence of forcing two corrector iterations per step. The limitations on the stepsize to ensure stability do not appear to seriously limit the efficiency of the code on these problems. The modified code was slightly less efficient than LSODE on the sum total over all of the nonstiff problems. This is to be expected, as there is some price to be paid for making the tests to diagnose stiffness for problems that are not stiff. A summary of results for the nonstiff DETEST problems is shown in Table 1. We have also included in this table the results of using LSODE with $MF = 22$ (BDF with Newton's method using finite-difference Jacobian), as an example of how a code which might be used if the problems were suspected to be stiff would perform on the test problems.

TABLE 1
Nonstiff test problems, summary.

Code	EPS	Exec. time	FCN calls	No. of steps
Switching*	10^{-3}	5.265	7,891	3,234
	10^{-6}	12.041	17,189	7,681
	10^{-9}	24.589	30,987	14,819
	Overall	41.894	56,067	25,734
LSODE ($MF = 10$)†	10^{-3}	4.334	5,412	3,557
	10^{-6}	10.417	11,081	8,948
	10^{-9}	24.243	22,581	19,595
	Overall	38.994	39,074	32,100
LSODE ($MF = 22$)‡	10^{-3}	10.263	8,503	3,909
	10^{-6}	25.488	19,237	10,454
	10^{-9}	61.403	43,926	28,595
	Overall	97.154	71,666	42,958

* The stiff methods in the code use modified Newton iteration with finite-difference generated Jacobian matrices.

† The option $MF = 10$ uses Adams methods with functional iteration.

‡ The option $MF = 22$ uses BDF and modified Newton iteration with finite-difference generated Jacobian matrices.

On the stiff problems, our experience indicates that the tests work very well at loose and moderate tolerances. With EPS $= 10^{-3}$, the code switched to the stiff methods at a reasonable time for every problem, and with EPS $= 10^{-6}$ the results were very

good except for problem E4. At tighter tolerances, there were minor difficulties mainly with B5 and E4. Both of these problems have eigenvalues with relatively large imaginary parts, especially B5 which has eigenvalues $-10 \pm 100i$. So far as we have been able to tell, the difficulties with these problems appear to be related to fluctuating lower bounds for $\|\delta f/\delta y\|$. The code switched back and forth between the two families of methods once on problems E2, EPS $= 10^{-3}$, and F4, EPS $= 10^{-3}$. This is the correct action for problem E2, which is van der Pol's equation (7) with $\eta = 5$. Problem F4 is the Field–Noyes chemical oscillator [2], and the response of the code to this problem appears to be incorrect, although this does not seriously degrade the efficiency of the code (over using LSODE with $MF = 22$).

A better test for deciding when the error estimate is polluted by roundoff would probably increase the reliability of this scheme at tight tolerances, although it performs well already for most problems, and we do not expect many users to ask for stringent tolerances when solving stiff problems. For practically all stiff problems, this new switching technique uses many fewer Jacobian evaluations, and it is definitely more efficient than LSODE ($MF = 22$) at loose tolerances. Since most stiff problems we would expect to see in a practical situation would be somewhat larger than the test problems (all have dimension less than or equal to ten) and would be solved with loose to moderate tolerances, this technique is useful. At stringent error tolerances, our scheme uses more function evaluations than LSODE ($MF = 22$), probably due again to forcing two corrector iterations per step in the nonstiff part of the code (during the transient). A summary of results for the stiff DETEST problems is given in Table 2, and detailed results are available from the author.

TABLE 2
Stiff test problems, summary.

Code	EPS	Exec. time	FCN calls	JAC calls	No. of steps
Switching*	10^{-3}	8.244	8,331	488	3,821
	10^{-6}	21.606	20,676	707	9,984
	10^{-9}	54.213	50,763	1,894	22,751
	Overall	84.063	79,770	3,089	36,556
LSODE ($MF = 22$)†	10^{-3}	14.488	10,143	753	5,368
	10^{-6}	27.414	19,227	1,285	11,136
	10^{-9}	61.564	43,129	2,700	27,103
	Overall	103.466	72,499	4,738	43,607

* The stiff methods in the code use modified Newton iteration with finite-difference generated Jacobian matrices.

† The option $MF = 22$ uses BDF and modified Newton iteration with finite-difference generated Jacobian matrices.

To illustrate how the code performs on a problem which repeatedly changes character during the interval of integration, we have included the results of applying the code to van der Pol's equation (7) with $\eta = 100.0$ on the interval [0, 1000]. A plot of the first component of the solution to this problem is shown in Fig. 1. During the times when the solution is changing rapidly, the problem is nonstiff, and when it is changing more slowly, it is stiff. Statistics on the performance of the modified code for this problem are shown in Tables 3a, b.

TABLE 3
van der Pol's equation test results

	Switched from BDF to Adams		Switched from Adams to BDF	
	Time	Step	Time	Step
EPS = 10^{-6}			.04095	34
	80.79	172	81.25	417
	162.21	555	162.59	680
	162.62	813	162.67	859
	243.64	981	244.09	1190
	325.05	1312	325.42	1423
	325.44	1519	325.50	1583
	406.46	1721	406.84	1846
	406.87	1979	406.92	2025
	487.89	2147	488.34	2356
	569.30	2478	569.75	2711
	650.72	2833	651.17	3042
	732.14	3164	732.59	3381
	813.54	3519	813.92	3644
	813.95	3777	814.00	3823
	894.97	3945	895.42	4154
	976.38	4276	976.76	4387
	976.78	4497	976.84	4546
EPS = 10^{-9}			.06328	73
	80.40	341	81.18	618
	81.19	840	81.28	953
	161.89	1214	162.70	1639
	243.32	1912	244.12	2332
	324.68	2584	325.53	3018
	406.14	3292	406.95	3719
	485.38	3917	485.47	3938
	487.51	4016	488.37	4433
	566.98	4655	569.79	5186
	650.38	5476	651.21	5913
	731.77	6171	732.63	6659
	813.23	6898	814.05	7369
	894.66	7641	895.46	8079
	976.07	7335	976.88	8766

Code	Error tolerance	No. of steps	FCN calls	JAC calls
Switching*	10^{-6}	4565	9311	372
	10^{-9}	8802	17465	456
LSODE† ($MF = 22$)	10^{-6}	5810	9124	707
	10^{-9}	15851	21617	1330

* The stiff methods in the code use modified Newton iteration with finite-difference generated Jacobian matrices.

† The option $MF = 22$ uses BDF and modified Newton iteration with finite-difference generated Jacobian matrices.

We are still making improvements to the switching code. Anyone desiring a copy of the code described in this paper (or possibly an updated version of this code) is encouraged to write to the author.

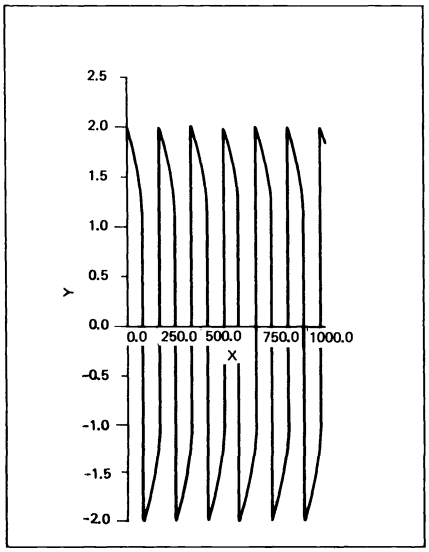


FIG. 1. *Van der Pol equation—first component.*

5. Summary. A scheme has been described for automatically determining whether a system of ordinary differential equations can be solved more efficiently using a class of methods suited for nonstiff problems or a class of methods designed for stiff problems. A code using this new technique is nearly as efficient for solving problems which are known in advance to be nonstiff (stiff) as codes designed for nonstiff (stiff) problems. Switching between families of methods is often more efficient than using a stiff method alone for problems which are nonstiff in some regions of the interval of integration and stiff in other regions. This scheme is useful for solving problems where the character of the problem is not known in advance, because the methods which are likely to be most efficient are selected automatically.

Acknowledgments. The author would like to thank L. F. Shampine for several stimulating discussions on this topic. Discussions with T. H. Jefferson, R. J. Kee, C. W. Gear, K. Stewart, and K. Jackson have also been very helpful in straightening out some of the ideas presented here.

REFERENCES

[1] W. H. ENRIGHT T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of ODE's*, BIT, 15 (1975), pp. 10–48.

[2] W. H. ENRIGHT AND T. E. HULL, *Comparing numerical methods for the solution of stiff systems of ODE's arising in chemistry*, in Numerical Methods of Differential Systems, Academic Press, New York, 1976, pp. 45–63.

[3] A. C. HINDMARSH, LSODE *and* LSODI, *two new initial value ordinary differential equation solvers*, ACM SIGNUM Newsletter, 15, 4, Dec. 1980.

[4] T. E. HULL, W. H. ENRIGHT, B. M. FELLEN AND A. E. SEDGWICK, *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal., 9 (1972), pp. 603–637.

[5] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations*, W. H. Freeman, San Francisco, 1975.

[6] L. F. SHAMPINE, *Stiffness and nonstiff differential equations solvers*, II: *Detecting stiffness with Runge–Kutta methods*, ACM Trans. Math. Software, 3 (1977), pp. 44–53.

[7] ———, *Lipschitz constants and robust ODE codes*, in Computational Methods in Nonlinear Mechanics, North-Holland, Amsterdam, 1980, pp. 427–449.

[8] ———, *Type-insensitive ODE codes based on implicit A-stable formulas*, SAND79-244, Sandia National Laboratories, Livermore, CA, 1979.

[9] S. SKELBOE, *The control of order and steplength for backward differentiation methods*, BIT 17 (1977), pp. 91–107.

[10] K. STEWART, Personal communication, Jet Propulsion Laboratory.

[11] Z. ZLATEV AND P. G. THOMSEN, *Automatic solution of differential equations based on the use of linear multistep methods*, ACM Trans. Math. Software, 5 (1979), pp. 401–414.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0001 $01.25/0

AN ITERATIVE METHOD FOR SOLVING INVERSE PROBLEMS OF A NONLINEAR WAVE EQUATION*

ROBERT P. HATCHER† AND Y. M. CHEN†

Abstract. The iterative numerical algorithm of the pulse-spectrum technique (PST) is extended to solve the inverse problem for a nonlinear wave equation arising from remote sensing of the ocean sound velocity profile by using finite-amplitude acoustic waves. Its applicability is demonstrated for the one-dimensional case. Numerical simulations are carried out to test the feasibility and to study the general characteristics of this technique without real measurement data. It is found that the PST does give reasonably good results, and hence it is a viable method.

Key words. iterative method, pulse-spectrum technique, inverse problems, nonlinear wave equation, numerical simulation

1. Introduction. The sound velocity profile of the ocean can be inferred numerically from a small number of experimental data obtained through remote sensing techniques on the boundary as opposed to in situ techniques in the interior. From the experimentalist's point of view, for the ease of performing a reliable experiment the measured physical quantity should be as fundamental as possible, e.g., the particle displacement, the particle velocity, the pressure, etc. Often this type of remote sensing problem can be formulated as an ill-posed inverse problem for a wave equation, where the solution is often not unique and does not depend continuously on the given data.

This type of inverse problem for the linear wave equation has been solved successfully using both numerical and analytical methods by many researchers in the past and the present, e.g. Chen and Tsien [1], [2], Cohen and Bleistein [3], [4], Hagin [5], etc. However, the linearized wave equation cannot accurately describe the correct physical phenomena and is strictly correct only for waves with infinitesimal amplitude. Hence one should consider this type of inverse problem for a nonlinear wave equation, which should at least contain a small nonlinear term so that it can describe the weak shock phenomenon. In this regard, to solve the inverse problems of a nonlinear wave equation, Nigul [6] and Nigul and Engelbrecht [7] have presented a perturbation method for solving an inverse problem of a nonlinear wave equation; however, it is limited to one-dimensional problems and is not suitable for numerical computation. In this paper we extend the so-called "pulse-spectrum technique" (PST)—an iterative computational algorithm—for determining the unknown velocity profile of a one-dimensional ocean model from the boundary measurements of the particle displacement and its space derivative (proportional to the pressure). It is more suitable for numerical computation and can be readily extended to solve three-dimensional inverse problems. The basic idea of PST is that data are measured in the time-domain with compact support (pulse-shaped functions), and the synthesis of the unknown coefficient is carried out numerically in the complex frequency domain by an iterative algorithm.

The PST was first introduced by Tsien and Chen [1] for solving an idealized velocity inverse problem for the linear wave equation. Then it was further developed to have the capability of handling noise and poorly distributed and inadequately measured data by Chen and Tsien [2]. Later it was used to solve an inverse problem in electromagnetic wave propagation by Tsien and Chen [8]. Recently, its versatility

* Received by the editors September 21, 1981, and in revised form May 10, 1982.

† Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, New York 11794.

has been demonstrated by Chen and Liu [9] in an application to solving inverse problems of a linear diffusion equation. Moreover, the discretized version of this iterative algorithm under idealized conditions has been proved to converge quadratically by Chen [10]. This is quite efficient from the numerical computation point of view.

The main purpose of the present paper is to extend the numerical algorithm of PST to solve inverse problems of a nonlinear wave equation and to demonstrate its applicability. For simplicity, only the formulation of the inverse problem of a one-dimensional nonlinear wave equation and its numerical algorithm are presented. The numerical algorithm is written in a modular form. The numerical method used in each module is chosen, not for its being the most efficient one in the one-dimensional case, but for its ability to be extended to the three-dimensional case in a straightforward manner. Then numerical simulations are carried out to test the feasibility and to study the intrinsic characteristics of this numerical algorithm without real measurement data. Finally, a comprehensive discussion of the numerical results and their implication in actual implementation of this computational algorithm are given.

2. Mathematical formulation of the problem. The problem set forth here from acoustical oceanography is the determination of the sound velocity profile, $c(x)$. This is achieved through remote sensing via acoustical pulses, which are produced near the surface and directed downwards through the stratified ocean and past a deep recording device near the bottom. The measurements of the pulses near the surface and near the bottom of the ocean (initially at rest), together with the governing equation for waves of finite amplitude, from nonlinear acoustics [11], make up the complete mathematical formulation of the inverse problem.

The equation of motion governing the particle displacement $\xi(x, t)$ in a nondissipative fluid undergoing longitudinal compression from acoustical disturbances is

$$\frac{\partial^2 \xi}{\partial t^2} = c^2(x)\left(1-\alpha \frac{\partial \xi}{\partial x}\right)\frac{\partial^2 \xi}{\partial x^2}, \qquad 0 \leqq x \leqq 1, \quad 0 \leqq t, \tag{1}$$

where $c(x)$ (>0) is the unknown sound velocity profile, and $\alpha(x)$ (>0) is the known nonlinearity parameter which is chosen to be small to avoid shock formation in the time interval of measurements. The nonlinearity parameter, α, is usually treated as a constant, but for the sake of generality it is considered as a function of x, $\alpha(x)$, throughout the treatment of this problem. The deep recording device is located at $x = 0$ and the other recording device, near the surface, is located at $x = 1$. In other words, x represents the height above the deep recording device.

The initial conditions, at the beginning of the remote sensing experiment, $t = 0$, are

$$\xi(x, 0) = d(x), \qquad \frac{\partial \xi}{\partial t}(x, 0) = f(x). \tag{2}$$

The common assumption is that the ocean is at rest, $d = f = 0$, but d and f are carried throughout this treatment in order to realize their effect in the solution process, as well as for the sake of generality.

Next, an acoustic pulse is produced and recorded at $x = 1$. It propagates downwards and past the deep recording device at $x = 0$ where it is measured again, i.e., the boundary conditions are

$$\xi(0, t) = g(t), \qquad \xi(1, t) = h(t). \tag{3}$$

If $c(x)$ were given, then the corresponding $\xi(x, t)$ would be determined by the above equation of motion, initial conditions and boundary conditions. However, since $c(x)$ is not known, $\xi(x, t)$ is not uniquely determined by the above constraints. Therefore, for the determination of $c(x)$ (cf. [1], [5]), an additional constraint is placed upon $\xi(x, t)$ by an extra set of data recorded at $x = 0$ as the pulse passes by; i.e.,

$$\frac{\partial \xi}{\partial x}(0, t) = p(t). \tag{4}$$

Thus, the inverse problem is to determine $c(x)$ from (1) given α and the data $d(x)$, $f(x)$, $g(t)$, $h(t)$ and $p(t)$. The experiment and hence the data recordings are terminated when the pulse has entirely passed by the deep recorder, i.e., out of the x domain at time $t = T$.

With regard to numerical use, any collection of data is essentially finite. In other words, the data are, at best, specified only along the edges of a space-time grid spanning the (x, t) domain. This discretized version of the problem is handled in § 4. Until then, the iterative solution of the problem will be treated in an analytic manner.

3. The iterative scheme. The following iterative algorithm is based upon successively improving upon an initial guess, $c_0(x)$, of the velocity profile and using the latest improved guess to figure out what the next correction should be. For convenience, it is $c^2(x)$ that will be determined, since c appears only in this form.

Let $c_n^2(x)$, $(n = 0, 1, 2, \cdots)$ be the nth iterative approximation of $c^2(x)$, where $c_0^2(x)$ is the initial guess, supposedly close to $c^2(x)$. Let $\xi_n(x, t)$ be the nth iterative approximation of $\xi(x, t)$, determined by (1), (2) and (3) with c_n^2 in place of c^2. Let

$$\delta c_n^2(x) = c_{n+1}^2(x) - c_n^2(x). \tag{5}$$

Given c_n^2 and thus ξ_n, the correction δc_n^2 will be determined as follows and then added to c_n^2 to get c_{n+1}^2, a better approximation of c^2. Similarly let

$$\delta \xi_n(x, t) = \xi_{n+1}(x, t) - \xi_n(x, t). \tag{6}$$

This is the corresponding correction that will occur in $\xi(x, t)$ when c_n^2 is replaced by c_{n+1}^2. These corrections will be assumed to be small.

Upon substitution of ξ_n and ξ_{n+1} $(=\xi_n + \delta\xi_n)$ into (1) with c_n^2 and c_{n+1}^2 $(=c_n^2 + \delta c_n^2)$, respectively, we get

$$\frac{\partial^2 \xi_n}{\partial t^2} = c_n^2 \left(1 - \alpha \frac{\partial \xi_n}{\partial x}\right) \frac{\partial^2 \xi_n}{\partial x^2}, \tag{7}$$

$$\frac{\partial^2}{\partial t^2}(\xi_n + \delta\xi_n) = (c_n^2 + \delta c_n^2)\left(1 - \alpha \frac{\partial}{\partial x}(\xi_n + \delta\xi_n)\right) \frac{\partial^2}{\partial x^2}(\xi_n + \delta\xi_n). \tag{8}$$

Subtracting (7) from (8), we obtain

$$\frac{\partial^2 \delta\xi_n}{\partial t^2} = c_n^2 \frac{\partial^2 \delta\xi}{\partial x^2} + \delta c_n^2 \left(1 - \alpha \frac{\partial \xi_n}{\partial x}\right) \frac{\partial^2 \xi_n}{\partial x^2} + c_n^2 \alpha O_1(\delta) + O_2(\delta^2), \tag{9}$$

where

$$O_1(\delta) = \frac{\partial \delta\xi_n}{\partial x} \frac{\partial^2 \xi_n}{\partial x^2} + \frac{\partial \xi_n}{\partial x} \frac{\partial^2 \delta\xi_n}{\partial x^2}$$

and $O_2(\delta)$ contains expressions of the second and third order in correction terms.

Since the boundary and initial conditions on ξ_n and ξ_{n+1} are identical, the conditions on $\delta\xi_n$ are homogeneous.

The last two terms of (9) will be neglected due to the supposedly small magnitudes of the correction terms and α. (In linear acoustics, $\alpha = 0$.)

$$\frac{\partial^2 \delta\xi_n}{\partial t^2} = c_n^2 \frac{\partial^2 \delta\xi_n}{\partial x^2} + \delta c_n^2 \left(1 - \alpha \frac{\partial \xi_n}{\partial x}\right) \frac{\partial^2 \xi_n}{\partial x^2}. \tag{10}$$

Using (7), this becomes

$$\frac{\partial^2 \delta\xi_n}{\partial t^2} - c_n^2 \frac{\partial^2 \delta\xi_n}{\partial x^2} = \delta c_n^2 c_n^{-2} \frac{\partial^2 \xi_n}{\partial t^2}. \tag{11}$$

This will serve as the basic relationship between δc_n^2 and $\delta\xi_n$, for c_n^2 and ξ_n have already been determined.

Next, according to PST, the time domain is replaced by a frequency spectrum. This is accomplished through the finite Fourier sine transform,

$$\tilde{\xi}_n(x, \omega) = \int_0^T \xi_n(x, t) \sin \omega t \, dt. \tag{12}$$

A finite transform is used not only because the data measurements were terminated at time T, but also because the infinite transforms may not converge.

It is reasonable to assume that all sound disturbances have left the x domain by the time that the acoustic pulse has passed entirely through it; i.e.,

$$\xi(x, t) = \frac{\partial \xi(x, t)}{\partial t} = 0, \qquad t \geqq T. \tag{13}$$

Thus, an infinite integral of ξ would present no problems. But since ξ_n is only an approximation of ξ, the pulse produced at $x = 1$ most likely will not quite match the boundary conditions at $x = 0$, e.g., $\partial\xi(0, t)/\partial x \neq p(t)$, and thus will not simply pass out of the x domain leaving the system at rest. Instead, a small residual wave is produced by the boundary conditions at $x = 0$ minus the effect of the acoustic pulse as it reaches that point. Since the equation governing these waves is designed for nondissipative fluids, the residual wave may reflect back and forth indefinitely without ever dying out. Hence, the infinite transform of ξ_n will not converge, in general.

Ideally, the determined correction δc_n^2 will be such that $c_{n+1}^2 = c^2$ and thus $\xi_{n+1} = \xi$. From the latter equality, (6) and (13), we get

$$\begin{aligned} &\delta\xi_n(x, T) = -\xi_n(x, T), \\ &\frac{\partial \delta\xi_n(x, T)}{\partial t} = -\frac{\partial \xi_n(x, T)}{\partial t}. \end{aligned} \tag{14}$$

Applying the finite Fourier sine transform to (11), keeping in mind the homogeneous conditions on $\delta\xi_n$, we get

$$\begin{aligned} \frac{\partial^2 \delta\tilde{\xi}_n}{\partial x^2} + \omega^2 c_n^{-2} \delta\tilde{\xi}_n &= c_n^{-2} \left\{ -\frac{\partial \xi_n(x, T)}{\partial t} \sin \omega T + \omega \xi_n(x, T) \cos \omega T \right\} \\ &+ \delta c_n^2 c_n^{-4} \left\{ -\frac{\partial \xi_n(x, T)}{\partial t} \sin \omega T + \omega [\xi_n(x, T) \cos \omega T - d(x)] + \omega^2 \tilde{\xi}_n \right\}. \end{aligned} \tag{15}$$

The boundary conditions accompanying this nonhomogeneous linear ordinary differential equation are

$$\delta\tilde{\xi}(0,\omega)=\int_0^T \delta\xi_n(0,t)\sin\omega t\,dt=0,$$
$$\delta\tilde{\xi}(1,\omega)=\int_0^T \delta\xi_n(1,t)\sin\omega t\,dt=0. \tag{16}$$

Now, everything on the right-hand side, RHS (x,ω), of (15) is known except δc_n^2. Hence, $\delta\tilde{\xi}_n$ can be expressed explicitly in terms of δc_n^2 by means of Green's function $G_n(\omega,x;y)$ of (15) and (16):

$$\int_0^1 \text{RHS}\,(y,\omega)G_n(\omega,x;y)\,dy=\delta\tilde{\xi}_n(x,\omega). \tag{17}$$

With $\delta\tilde{\xi}_n$ in this form, we can finally use the extra boundary condition, (4), that was supplied in order to compensate for the lack of knowledge about $c(x)$. Since $\xi_{n+1}=\xi$, ideally speaking, (4) may be written as

$$\frac{\partial\delta\tilde{\xi}_n(0,\omega)}{\partial x}=\frac{\partial\tilde{\xi}(0,\omega)}{\partial x}-\frac{\partial\tilde{\xi}_n(0,\omega)}{\partial x}=\tilde{p}(\omega)-\frac{\partial\tilde{\xi}_n(0,\omega)}{\partial x}. \tag{18}$$

This is incorporated into (17) after differentiation and setting x equal to 0:

$$\int_0^1 \text{RHS}\,(y,\omega)\frac{\partial G_n(\omega,0;y)}{\partial x}\,dy=\tilde{p}(\omega)-\frac{\partial\tilde{\xi}_n(0,\omega)}{\partial x}. \tag{19}$$

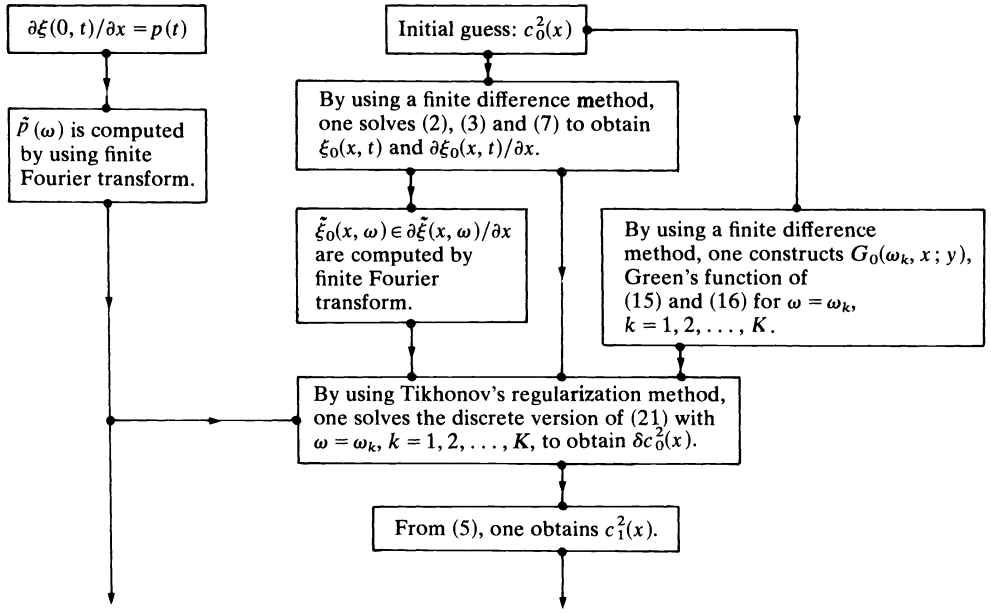
Finally, each expression in (19) is known except δc_n^2, which appears in one of the two terms of RHS. Transferring the other term to the right-hand side of (19), we obtain

$$\int_0^1 \delta c_n^2 c_n^{-4}\left\{-\frac{\partial\tilde{\xi}_n(y,t)}{\partial t}\sin\omega T+\omega[\tilde{\xi}_n(y,T)\cos\omega T-d(y)]+\omega^2\tilde{\xi}_n(y,\omega)\right\}\frac{\partial G_n(\omega,0;y)}{\partial x}\,dy$$
$$=\tilde{p}(\omega)-\frac{\partial\tilde{\xi}_n(0,\omega)}{\partial x}+\int_0^1 c_n^{-2}\left[\frac{\partial\tilde{\xi}_n(y,T)}{\partial t}\sin\omega T-\omega\tilde{\xi}_n(y,T)\cos\omega T\right]\frac{\partial G_n(\omega,0;y)}{\partial x}\,dy. \tag{20}$$

This can be written in the abbreviated form

$$\int_0^1 K_n(\omega,y)\,\delta c_n^2(y)\,dy=b_n(\omega). \tag{21}$$

Hence, δc_n^2 is the solution of a Fredholm integral equation of the first kind which is an ill-posed problem, in general. Thus, special techniques must be employed in order to insure inverse continuity in the solution. Here we prefer to use Tikhonov's regularization method [12], not for its efficiency in solving the one-dimensional Fredholm integral equation of the first kind, but for its ability to be extended to solve the three-dimensional case in a straightforward manner. The essence of the first cycle of iteration is given in the accompanying diagram and the procedure for later cycles is exactly the same.



4. Discretization.

(a) Nonlinear wave equation. To perform the iterative algorithm numerically, the x and t intervals, $[0, 1]$ and $[0, T]$, are partitioned into M and N subdivisions, respectively, thereby producing a uniform mesh with $\Delta x = 1/M$ and $\Delta t = T/N$. All functions are replaced by their discrete counterparts; e.g., $u_{ij} \equiv \xi_n(i\Delta x, j\Delta t)$, $c_i^2 \equiv c_n^2(i\Delta x)$, and $a_i \equiv a(i\Delta x)$, where $n = 0, 1, \cdots$ denotes the iteration, $i = 1, \cdots, M-1$, and $j = 1, \cdots, N$. The initial and boundary values of ξ (not included in u_{ij}), are given as data: $d_i \equiv \xi(i\Delta x, 0)$, $f_i \equiv \partial\xi(i\Delta x, 0)/\partial t$, $g_j \equiv \xi(0, j\Delta t)$, and $h_j \equiv \xi(1, j\Delta t)$, where $i = 0, \cdots, M$ and $j = 0, \cdots, N$. The extra information supplied, in order to determine the true velocity profile $c(x)$, is also given as data; i.e., $p_j \equiv \partial\xi(0, j\Delta t)/\partial x$, where $j = 0, \cdots, N$. The nonlinearity parameter and the initial guess of $c^2(x)$ must also be supplied, as a_i and c_i^2, respectively. Any additional data are indicated where necessary.

To aid the simulation of the nonlinear wave equation (7), let

$$
\begin{aligned}
w_{ij} &\equiv \frac{\partial\xi_n[(i-1/2)\Delta x, j\Delta t]}{\partial x}, \qquad i = 1, \cdots, M, \quad j = 0, \cdots, N-1, \\
v_{ij} &\equiv \frac{\partial\xi_n[i\Delta x, (j-1/2)\Delta t]}{\partial t}, \qquad i = 1, \cdots, M-1, \quad j = 1, \cdots, N.
\end{aligned} \tag{22}
$$

By utilizing finite difference approximations, u_{ij} is calculated by the following iterative scheme (starting with $j = 1$):

$$
\begin{aligned}
&w_{i0} = (d_i - d_{i+1})/\Delta x, \\
&v_{i1} = f_i + \Delta t 2^{-1} c_i^2 [1 - a_i(w_{i+1,0} + w_{i,0})/2](w_{i+1,0} - w_{i,0})/\Delta x, \\
&u_{ij} = u_{i,j-1} + \Delta t v_{ij} \qquad (u_{i0} \equiv d_i), \\
&w_{1j} = (u_{1j} - g_j)/\Delta x, \qquad w_{Mj} = (h_j - u_{M-1,j})/\Delta x, \\
&w_{ij} = w_{i,j-1} + \Delta t(v_{ij} - v_{i-1,j})/\Delta x \qquad (i \neq 1, M), \\
&v_{i,j+1} = v_{ij} + \Delta t c_i^2 [1 - a_i(w_{i+1,j} + w_{ij})/2](w_{i+1,j} - w_{ij})/\Delta x.
\end{aligned} \tag{23}
$$

For the above order of calculations it is only necessary to store the latest values of v_{ij} and w_{ij}, with respect to j. For later use, the quantities $\Delta u'_{0j} \equiv \partial[\xi(0, j\Delta t) - \xi_n(0, j\Delta t)]/\partial x, j = 0, \cdots, N$, and $\dot{u}_{iT} \equiv \partial \dot{\xi}_n(i\Delta x, T)/\partial t$, $i = 1, \cdots, M-1$, are calculated as follows:

$$\begin{aligned} &\Delta u'_{0j} = p_j - w_{1j}, \\ &\dot{u}_{iT} = v_{iN} + 2^{-1}\Delta t c_i^2[1 - a_i(u_{i+1,N} - u_{i-1,N})/2\Delta x](u_{i+1,N} - 2u_{iN} + u_{i-1,N})/\Delta x^2, \end{aligned} \tag{24}$$

where u_{0N} and u_{MN} denote the boundary values g_n and h_N, respectively.

In order to avoid instability in the above scheme, the (x, t) mesh must be capable of transmitting information at least as fast as the characteristics of the nonlinear wave equation, i.e.,

$$\frac{\Delta x}{\Delta t} > c_n(x)\left[1 - a(x)\frac{\partial \xi_n}{\partial x}\right]^{1/2}, \tag{25}$$

which implies that

$$\frac{\partial \xi_n}{\partial x} > a(x)^{-1}\left[1 - \left(\frac{\Delta x}{\Delta t} c_n(t)\right)^2\right]. \tag{26}$$

Once $\xi_n(x, t)$ has been approximated by the above scheme, (26) can be utilized as a check of its validity. However, for waves of finite but moderate amplitude, $|a\partial\xi/\partial x|$ is no longer negligible, compared to 1, but can still be assumed to be less than 1. Hence, a safe choice for the (x, t) mesh would be one such that

$$\left(\frac{\Delta x}{\Delta t}\right)^2 \geqq 2 \max c_i^2. \tag{27}$$

When an isolated wave pulse is introduced through one of the boundaries, the above scheme tends to slightly "overshoot" the trailing end of the wave, where it should end abruptly. This produces a small oscillating tail on the end of the traveling wave pulse (numerically induced dispersion). In order to prevent these extra oscillations, artificial viscosity can be introduced into the nonlinear wave equation by the inclusion of an additional term, producing the equation

$$\frac{\partial^2 \xi}{\partial t^2} = c^2(x)\left[1 - a(x)\frac{\partial \xi}{\partial x}\right]\frac{\partial^2 \xi}{\partial x^2} + \gamma(x)\frac{\partial^3 \xi}{\partial t\, \partial x^2}, \tag{28}$$

where $\gamma(x)$ is a small parameter controlling the amount of viscosity. This alteration is effected in the above scheme by replacing the last equation of (23) with

$$\begin{aligned} v_{i,j+1} = v_{ij} &+ \Delta t c_i^2[1 - a_i(w_{i+1,j} + w_{ij})/2](w_{i+1,j} - w_{ij})/\Delta x \\ &+ \Delta t \gamma_i(v_{i+1,j} - 2v_{ij} + v_{i-1,j})/\Delta x^2, \end{aligned} \tag{29}$$

where $\gamma_i \equiv \gamma(i\Delta x)$, $i = 1, \cdots, M-1$.

(b) Finite sine transform. Once $\xi_n(x, t)$ has been approximated by the previous scheme, its finite sine transform (12) is calculated under the assumption that ξ_n is a linear function with respect to t within each interval of the (x, t) mesh, i.e.,

$$\xi_n(i\Delta x, t) = \sum_{j=0}^{N} u_{ij} L_j(t), \qquad 0 \leqq t \leqq T, \tag{30}$$

where $u_{i0} \equiv d_i$ and

$$L_j(t) \equiv \begin{cases} 1-|j-t/\Delta t|, & j-1 \leqq t/\Delta t \leqq j+1, \\ 0, & \text{elsewhere.} \end{cases} \tag{31}$$

The finite sine transform, $\tilde{u}_i(\omega) \equiv \tilde{\xi}_n(i\Delta x, \omega)$, for the above piecewise linear form of ξ_n reduces to

$$\tilde{u}_i(\omega) = \sum_{j=0}^{N} u_{ij} S_j(\omega) \tag{32}$$

where

$$S_j(\omega) = \int_0^T L_j(t) \sin \omega t \, dt. \tag{33}$$

Through integration by parts, it follows that

$$\begin{aligned} &S_0(\omega) = \omega^{-1}(1-\sin \omega t/\omega t), \\ &S_j(\omega) = 2\omega^{-1}\Delta t^{-1}(1-\cos \omega t) \sin \omega j \Delta t, \qquad j = 1, \cdots, N-1, \\ &S_N(\omega) = \omega^{-2}\Delta t^{-1}[\sin \omega T - \sin \omega(T-\Delta t)] - \omega^{-1} \cos \omega T. \end{aligned} \tag{34}$$

Similarly, assuming that $\partial \delta \xi_n(0, t)/\partial x$ has an equivalent piecewise linear form, $\Delta \tilde{u}_0'(\omega) \equiv \partial \delta \tilde{\xi}_n(0, \omega)/\partial x$ is approximated by

$$\Delta u_0'(\omega) = \sum_{j=0}^{N} \Delta u_{0j}' S_j(\omega). \tag{35}$$

Due to the design of the scheme, (32) and (35) provide the exact finite sine transform of any continuous function that, for fixed x, is linearly dependent upon t within each subdivision of the (x, t) mesh. For numerical use, this scheme is performed for the set of frequencies $\{\omega_k\}$, $k = 1, \cdots, K$. The corresponding approximations of the finite sine transforms are denoted by $\tilde{u}_{ki} \equiv \tilde{u}_i(\omega_k)$ and $\Delta \tilde{u}_{0k}' \equiv \Delta \tilde{u}_0'(\omega_k)$.

(c) Green's function. Next, the Green's function $G_n(\omega, x; y)$ of (15) and (16) is found in order to obtain $\partial G_n(\omega, 0; y)/\partial x$ for (20). Consider the formulation

$$G_n(\omega, x; y) = \begin{cases} k_0(\omega, y) g(\omega, x), & 0 \leqq x \leqq y, \\ k_1(\omega, y) h(\omega, x), & y \leqq x \leqq 1, \end{cases} \tag{36}$$

where

$$\begin{aligned} &\frac{\partial^2 g}{\partial x^2} + \omega^2 c_n^{-2}(x) g = 0, \qquad 0 \leqq x \leqq 1, \\ &g(\omega, 0) = 0, \qquad \frac{\partial g(\omega, 0)}{\partial x} = 1, \\ &\frac{\partial^2 h}{\partial x^2} + \omega^2 c_n^{-2}(x) h = 0, \qquad 0 \leqq x \leqq 1, \\ &h(\omega, 1) = 0, \qquad \frac{\partial h(\omega, 1)}{\partial x} = 1. \end{aligned} \tag{37}$$

From the required conditions for G_n, it easily follows that

$$
\begin{aligned}
k_0(\omega, y) &= h(\omega, y)/[g(\omega, y)\partial h(\omega, y)/\partial x - h(\omega, y)\partial g(\omega, y)/\partial x], \\
k_1(\omega, y) &= g(\omega, y)/[g(\omega, y)\partial h(\omega, y)/\partial x - h(\omega, y)\partial g(\omega, y)/\partial x],
\end{aligned} \tag{38}
$$

and hence

$$
\frac{\partial G_n(\omega, 0; y)}{\partial x} = k_0(\omega, y). \tag{39}
$$

The simplest numerical scheme for calculating $g_{ki} \equiv g(\omega_k, i\Delta x)$ and $h_{ki} \equiv h(\omega_k, i\Delta x)$, $i = 1, \cdots, M-1$, would be to let $g'_{ki} \equiv \partial g(\omega_k, (i-\frac{1}{2})\Delta x)/\partial x$ and $h'_{ki} \equiv \partial h(\omega_k, (i+\frac{1}{2})\Delta x)/\partial x$, $i = 1, \cdots, M-1$, and discretize (37) by

$$
\begin{aligned}
g_{k,i+1} &= g_{ki} + \Delta x g'_{k,i+1}, \\
g'_{k,i+1} &= g'_{ki} - \Delta x \omega_k^2 c_i^{-2} g_{ki}, \\
h_{k,i-1} &= h_{ki} - \Delta x h'_{k,i-1}, \\
h'_{k,i-1} &= h'_{ki} + \Delta x \omega_k^2 c_i^{-2} h_{ki}, \\
g_{k0} &= h_{kM} = 0, \qquad g'_{k1} = h'_{k,M-1} = 1.
\end{aligned} \tag{40}
$$

A similar method was used by Pruess [13] to compute the eigenvalues of Sturm–Liouville problems. The accuracy of this scheme is limited by the fact that Δx cannot be chosen arbitrarily small, since Δx has already been set by the compactness of the x grid on which $c_i^2 \equiv c_n^2(i\Delta x)$ was given. But if $c_n^2(x)$ is assumed to behave linearly between neighboring nodes of the grid, then the accuracy can be improved by inserting enough, $(P-1)$, extra nodes between each neighboring pair of nodes of the grid and using linear interpolation to evaluate c_n^2 at each of these extra nodes. Then the numerical scheme (40) can be applied to this new grid (containing MP subdivisions), with Δx replaced by $\Delta x/P$ (and minor subscripting adjustments).

Once $g(\omega, x)$ and $h(\omega, x)$ have been approximated by this finite difference scheme, $G'_{n0ki} \equiv \partial G_n(\omega_k, 0; i\Delta x)/\partial x$ is calculated from (39), i.e.,

$$
G'_{n0ki} = h_{ki}/(g_{ki}h'_{ki} - h_{ki}g'_{ki}). \tag{41}
$$

The choice of frequencies, ω_k, should be such as to avoid those frequencies where the Green's function is singular or near singular. When such frequencies are used, $G_n(\omega, x; y)$ becomes arbitrarily large, and hence the errors in approximating (20) are magnified.

(d) Integral equation. The integrals of (20) are calculated below by the simple "rectangular rule," summing over the M subdivisions of the x grid. Actually, only $M-1$ subdivisions are summed over because the last addendum vanished since $\lim_{y\to 1} \partial G_n(\omega, 0; y)/\partial x = 0$, as can be seen from (38) and (39). The integral equation (20) is then approximated by

$$
\begin{aligned}
&\sum_{i=1}^{M-1} \Delta c_i^2 c_i^{-4}[-\dot{u}_{iT}\sin\omega_k T + \omega_k(u_{iN}\cos\omega_k T - d_i) + \omega_k^2 \tilde{u}_{ki}]G'_{n0ki}\Delta x \\
&\qquad = \Delta u'_{0k} + \sum_{i=1}^{M-1} c_i^{-2}(\dot{u}_{iT}\sin\omega_k T - \omega_k u_{iN}\cos\omega_k T)G'_{n0ki}\Delta x,
\end{aligned} \tag{42}
$$

where $\Delta c_i^2 \equiv \delta c_n^2(i\Delta x)$, $i = 1, \cdots, M-1$, is the unknown to be determined, and $k = 1, \cdots, K$.

In abbreviated form, (42) can be expressed symbolically by the linear system

$$Ae = b, \tag{43}$$

where $e_i = \Delta c_i^2$, $i = 1, \cdots, M-1$, b_k equals the right-hand side of (42), $k = 1, \cdots, K$, and the K by $M-1$ matrix A is composed of the coefficients A_{ki} of Δc_i^2 on the left-hand side of (42).

Since (43) represents an ill-posed problem, the determination of e can now be handled by Tikhonov's regularization method [12] along with the error principle [14]. The regularized solution of (43) satisfies

$$(A^T A + \beta I)u = A^T b, \tag{44}$$

where $u_i = \Delta c_{\beta i}^2 \equiv \delta c_{n\beta}^2(i\Delta x)$, $i = 1, \cdots, M-1$, and the matrix A has the dimensions of $m' = K$ by $n' = M-1$.

The desired regularization parameter β can be defined by setting, in advance, the error level of u, e.g.,

$$\rho(\beta) \equiv \|Au - b\|_{m'} = r\|b\|_{m'}, \qquad 0 < r < 1, \tag{45}$$

where r is chosen in accordance with the level of errors within A and b. Given an initial guess $\beta_0 > 0$, a sequence of improved estimates β_i, converging to β, is produced by the modified Newton method,

$$\beta_{i+1} = \left[\beta_i^{-1} + \frac{\rho^3(\beta_i)}{\beta_i^3} \frac{\rho^{-1}(\beta_i) - r^{-1}\|b\|_{m'}^{-1}}{(u_{R\beta i}, u'_{R\beta_i})_{n'}}\right]^{-1}. \tag{46}$$

Furthermore, A is decomposed into QDR; Q and R are unitary matrices, and D is a bidiagonal matrix with nonzero elements on only the main diagonal and super-diagonal; $u_{R\beta}$ and $u'_{R\beta_i}$ satisfy $(D^T D + \beta_i I)u_{R\beta_i} = D^T Q^T b$ and $(D^T D + \beta_i I)u'_{R\beta_i} = -u_{R\beta_i}$, respectively.

5. Numerical examples. Throughout the iterative solution of the remote sensing problem, all quantities are considered to be dimensionless. For each example, the wave propagation was solved on an (x, t) grid of size $\Delta x = \Delta t = 0.025$ with $T = 3$ (i.e. $M = 40$ and $N = 120$). The initial conditions are homogeneous since the medium is initially at rest. The acoustic pulse is introduced into the medium through the boundary condition at $x = 1$. The pulse was chosen with the intention of holding down the size of the neglected terms of (9) and the tail oscillations produced by the numerical simulation. The use of artificial viscosity is kept to a minimum. The pulse constructed for this purpose is

$$\xi(1, t) = \begin{cases} -\sin^2 \pi/(6.5t^2 + 1)^2, & 0 \leqq t \leqq 1, \\ 0, & 1 < t \leqq T. \end{cases} \tag{47}$$

This pulse propagates leftward and passes through the boundary at $x = 0$, producing the left boundary condition, $\xi(0, t)$. Thus, for the examples of this section, $\xi(0, t)$ and $\partial\xi(0, t)/\partial x$ were numerically constructed by simulating the propagation of the pulse, using the true velocity profile $\bar{c}(x)$, from $x = 1$, past $x = 0$, to a boundary far enough to the left of $x = 0$ with homogeneous boundary conditions.

The number K of frequencies for use in the finite sine transforms was chosen to be 39, thereby producing a 39 by 39 (K by $M-1$) linear system (42) for the determination of Δc^2. The utilized frequencies were confined to the interval $(0, 20)$ since larger frequencies produce larger "phase differences" (i.e., errors), in Green's

function. The introduction of extra "nodes" (up to 20 for each Δx) into the scheme improved accuracy notably, but smaller frequencies were still preferred.

For simplicity of analysis, the true solution of the following examples was set at $\bar{c}^2(x) = 0.5$. The basic differences among the examples are the initial guesses $c_0^2(x)$, the chosen frequencies ω_k, and r the proportion of error to be permitted by the error principle in the regularization method. The nonlinearity parameter was set at $a = 0.05$. Considering the given pulse, true solution, and sizes of Δx and Δt, the nonlinearity parameter was set nearly as large as possible without violating the wave simulation stability criterion (25).

The first two examples demonstrate the effect of different choices of the error level in the error principle. Figures 1 and 2 illustrate the results obtained from setting $r = 50\%$ and 75%, respectively. From the total number of iterations performed, it is apparent that Example 1 converged nearly twice as fast as Example 2. This may be expected since the latter example was not required to comply with the linear system (42) as closely as the former example, i.e., 25% more error was permitted.

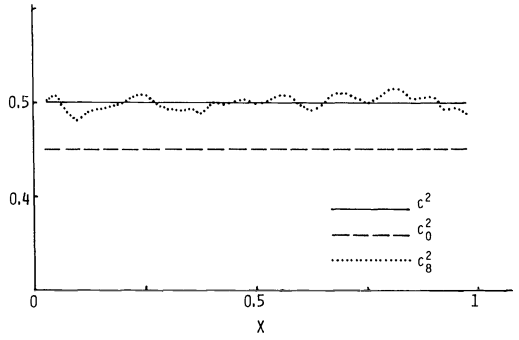


FIG. 1. *Numerical Example* 1.

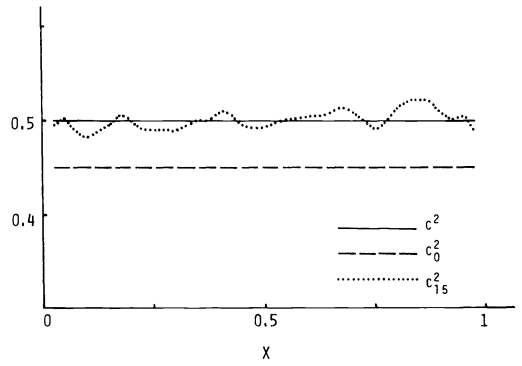


FIG. 2. *Numerical Example* 2.

However, in exchange for the greater error allowance, the regularized solutions of each iteration of Example 2 were smoother and less oscillatory. Thus, they could be repeatedly added to the latest c^2 approximation for many more iterations before the c^2 approximation became out of hand, e.g. too oscillatory. On the other hand, the iterative regularized solutions of Example 1 were of greater magnitude, producing faster convergence but at the risk of greater oscillations.

The oscillations of Example 1 were partially corrected for in the subsequent iterations. However, if r had been set much lower, e.g. 30%, the convergence might have appeared to be faster in the first few iterations, but the accompanying oscillations would have been too large to permit recovery in subsequent iterations. Hence, small values of r, i.e., close adherence to the linear system, is advisable when the initial estimate c_0^2 is close enough to the true solution $\bar{c}^2$, thereby limiting the possibility of unmanageable oscillations. But for initial estimates which are not very close to the true solution, a larger value of r is advisable, thereby producing slower convergence but of a more stable nature.

In Fig. 3, Example 3, a poorer initial estimate was chosen. Instead of choosing a single value of r for the whole algorithm, r was chosen individually for each iteration. A few values of r were tested for each iteration, from which one was chosen with the intention of achieving a favorable trade-off between the speed and stability of the overall convergence of the algorithm. The values chosen ranged from 30% to 60%, but there was no strong pattern to describe the choice of r aside from a slight decrease in r in the later iterations.

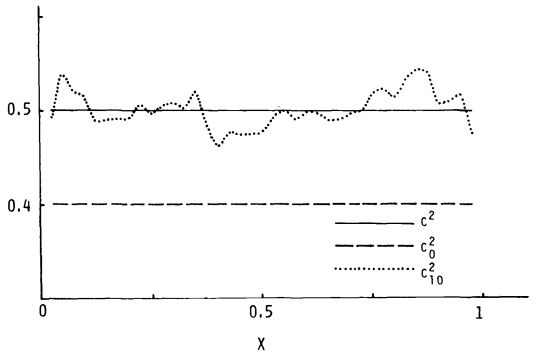


FIG. 3. *Numerical Example* 3.

In all of the examples presented so far, the initial estimate $c_0^2(x)$ was a constant. In the following examples, the algorithm's response to initial estimates of other forms is examined. In Figs. 4 and 5, the initial estimate is still linear but crosses over the true solution at the midpoint, $x = 0.5$, thus requiring a different amount of correction at every point of x. This initial estimate is tested with $r = 30\%$ and 50% in Examples 4 and 5, respectively. Similarly, the former example converged about twice as fast as

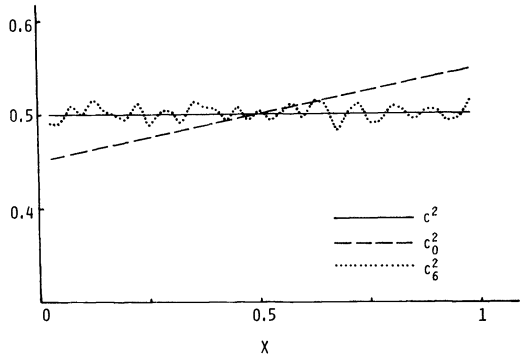


FIG. 4. *Numerical Example* 4.

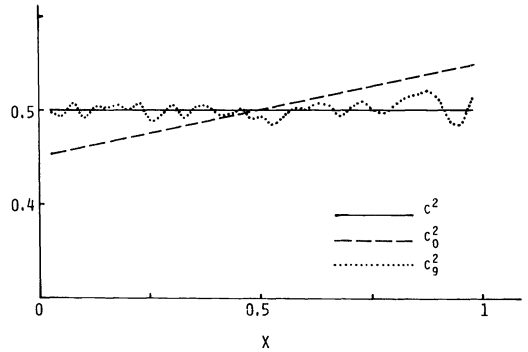


FIG. 5. *Numerical Example* 5.

the latter. Due to the closeness of the initial estimate, the greater oscillatory nature of the iterative regularized solutions of Example 4 could be "corrected" without getting out of control.

The next three examples are tests of the resolving power of the algorithm. In Figs. 6, 7, and 8, the initial estimates agree with the true solution everywhere except in a "wide," "medium," or "narrow" interval centered at $x = 0.5$. For each example, r was set equal to 50%. The algorithm performed quite well for these examples. The "narrow" Example 8 converged the fastest of the three. However, the "medium" Example 7 converged much closer after several more iterations.

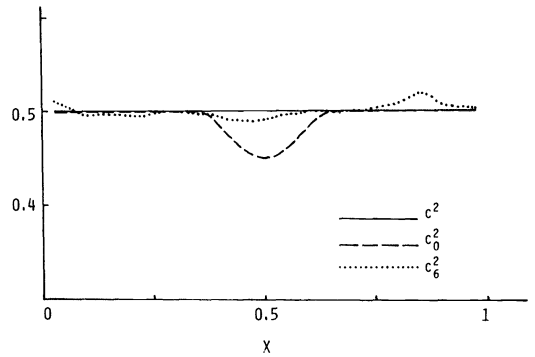


FIG. 6. *Numerical Example* 6.

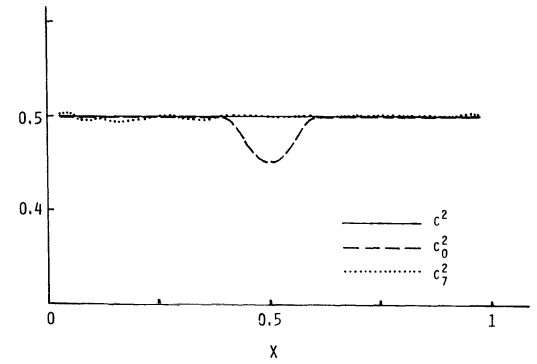


FIG. 7. *Numerical Example* 7.

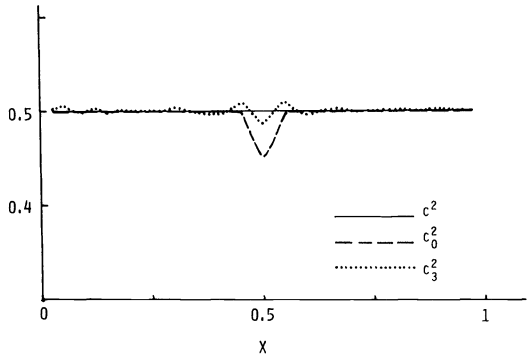


FIG. 8. *Numerical Example* 8.

6. Discussion. The testing and analysis of many nonlinear numerical examples has revealed those schemes of the algorithm which have a dominating effect upon its effectiveness. The finite difference scheme for solving the nonlinear wave equation is crucial in the preparation of the information, particularly $\partial\delta\xi_n(0, t)/\partial x$, which will determine δc_n^2. Its effectiveness diminishes in the late iterations as c_n^2 approaches $\bar{c}^2$, for then $\delta\xi_n$ tends to vanish, allowing numerical errors to obscure the essential information. This is especially true for the derivative $\partial\delta\xi_n(0, t)/\partial x$, which in addition is very sensitive to the undesired tail oscillations trailing the wave pulse.

The accuracy of the numerical algorithm can be improved greatly if more efforts are made in computing each individual step of the numerical algorithm and if a larger number of ω_k's is used and their values are properly chosen according to either the well-conditioned matrix criterion [15] or the minimum error criterion [16] in solving the Fredholm integral equation of the first kind. Furthermore, another restriction upon the choice of frequencies is the avoidance of neighborhoods of the singularities (with respect to ω) of Green's functions. They should be avoided, since in the neighborhood of singularities, numerical errors are most likely to be disproportionately large.

The numerical examples illustrate the effects of various values of r in the iterative algorithm under different conditions. When smaller values of r (i.e., closer adherence to the linear system) are permissible, their corresponding c^2 approximations improve faster in the early iterations than those with larger values of r. However, for smaller values, the regularized solutions Δc^2 have larger oscillations. Thus, the oscillations can accumulate faster in the c^2 approximation, thereby making it useless in relatively few iterations. Hence, smaller values of r are preferable when not many iterations are expected, e.g. when the initial guess $c_0^2(x)$ is close to the true solution $\bar{c}^2(x)$. But for initial guesses that are not so close to the true solution, it is advisable to use the more stable approximations arising from the smoother regularized solutions corresponding to larger values of r.

Aside from the above problem of accumulating oscillations, the regularization method has performed well along with the iterative algorithm. For example, the algorithm has performed successfully for both very close and very distant initial guesses. Also, the algorithm possesses good "resolving power," as illustrated in Figs. 6, 7 and 8. The results here can be further improved if $\delta c_n^2 = 0$ is imposed at least on one of the boundaries, as it was in the case of inverse problems of the linear wave equation [1], [2], [8].

REFERENCES

[1] D. S. TSIEN AND Y. M. CHEN, *A numerical method for nonlinear inverse problems in fluid dynamics*, in Computational Methods in Nonlinear Mechanics, Proc. International Conference on Computational Methods in Nonlinear Mechanics, Univ. of Texas at Austin, Austin, TX, 1974, 935–943.

[2] Y. M. CHEN AND D. S. TSIEN, *A numerical algorithm for remote sensing of density profiles of a simple ocean model by acoustic pulses*, J. Comput. Phys., 25 (1977), pp. 366–385.

[3] J. K. COHEN AND N. BLEISTEIN, *An inverse method for determining small variations in propagation speed*, SIAM J. Appl. Math., 32 (1977), pp. 784–799.

[4] ———, *Velocity inversion procedure for acoustic wave*, Geophysics, 44 (1979), pp. 1077–1087.

[5] F. HAGIN, *A stable approach to solving one-dimensional inverse problems*, SIAM J. Appl. Math., 40 (1981), pp. 439–453.

[6] U. NIGUL, *Influence of nonlinear effects on one-dimensional echo signals from elastic targets*, Soviet Phys. Acoust., 21 (1975), pp. 93–95.

[7] U. NIGUL AND J. ENGELBRECHT, *On dissipative, diffractional, and nonlinear effects in acousto-diagonostics of layered media*, in Modern Problems in Elastic Wave Propagation, J. Miklowitz and J. D. Achenbach, eds., Wiley-Interscience, New York, 1978, pp. 265–282.

[8] D. S. TSIEN AND Y. M. CHEN, *A pulse-spectrum technique for remote sensing of stratified media*, Radio Sci., 13 (1978), pp. 775–783.

[9] Y. M. CHEN AND J. Q. LIU, *A numerical algorithm for remote sensing of thermal conductivity*, J. Comput. Phys., 43 (1981), pp. 315–326.

[10] Y. M. CHEN, *Numerical methods for solving a class of inverse matrix problems in active remote sensing*, in Proc. International Symposium on Ill-Posed Problems: Theory and Practice, University of Delaware, Newark, DE, Oct. 1979.

[11] R. T. BEYER, *Nonlinear acoustics*, Amer. J. Phys., 41 (1973), pp. 1060–1067.

[12] A. N. TIKHONOV AND V. Y. ARSENIN, *Solutions of Ill-Posed Problems*, John Wiley, New York, 1977.

[13] S. A. PRUESS, *Higher-order approximation to Sturm–Liouville eigenvalues*, Numer. Math., 24 (1975), pp. 241–247.

[14] V. A. MOROZOV, *The error principle in the solution of operational equations by the regularization method*, U.S.S.R. Computational Math. and Math. Phys., 8 (1968), pp. 63–87.

[15] F. HAGIN, *On the construction of well-conditioned systems for Fredholm I problems by mesh adapting*, J. Comput. Phys., 36 (1980), pp. 154–169.

[16] E. TSIMIS, *On the inverse problem by means of the integral equation of the first kind*, Ph.D. Thesis, Dept. of Applied Math. and Statistics, State Univ. of New York at Stony Brook, Stony Brook, NY, 1977.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0002 $01.25/0

PITFALLS IN THE NUMERICAL SOLUTION OF LINEAR ILL-POSED PROBLEMS*

J. M. VARAH†

Abstract. Very special computational difficulties arise when we attempt to solve linear systems arising from integral equations of the first kind. We examine here existence and uniqueness questions associated with so-called *reasonable* solutions for such problems, and present results using the best-known methods on inverse Laplace transform problems. We also discuss the choice of free parameters occurring in these methods from the same point of view.

Key words. ill posed problems, integral equations of the first kind, inverse Laplace transform, singular value decomposition

1. Introduction. We are concerned with the numerical solution of the linear $n \times n$ system

$$Kf = g \tag{1.1}$$

where K is inherently ill-conditioned because of the source of the system. We assume for definiteness that the source is an integral equation of the first kind in one dimension:

$$\int \tilde{K}(s, t)\tilde{f}(t)\, dt = \tilde{g}(s). \tag{1.2}$$

We shall only consider the case when $\tilde{K}$ is a *smooth* kernel; then clearly the mapping $\tilde{f} \to \tilde{g}$ by (1.2) "smooths out" functions, and even takes noncontinuous $\tilde{f}$ into smooth $\tilde{g}$, so we can't hope to solve (1.2) for arbitrary $\tilde{g}$. In § 2 we examine this problem in more detail, in an attempt to specify when the given discrete problem (1.1) has a reasonable solution.

Several methods have been developed for solving (1.1), and in § 3 we consider four of these: the truncated singular value decomposition method, the regularization method, the modified regularization method, and a function expansion method. We refer to Varah (1979) and Björck and Elden (1979) for more details and references. Each of these methods involves a free parameter, and for an appropriate choice of this parameter each method is relatively stable with respect to perturbations in the data, so that each is a reasonable computational method. As we shall see, however, the solutions obtained may be very different from one another, so that it is extremely difficult to say which is "the best" numerical solution. In § 4, we examine several numerical examples illustrating this behavior, all involving the inverse Laplace transform operator in (1.2).

Finally in § 5, we discuss the choice of free parameters in the methods. We find that although reasonable choices can be made for the expansion methods, this does not appear to be the case for regularization methods.

2. Existence of reasonable solutions. For the continuous problem (1.2), one way of specifying the existence of a reasonable solution is the Picard condition: for L_2 kernels $\tilde{K}$ with $\iint \tilde{K}(s, t)^2\, ds\, dt < \infty$, there are orthogonal functions $\{\phi_i(s)\}$, $\{\psi_i(t)\}$, and

* Received by the editors September 25, 1981, and in revised form March 22, 1982. This material was first presented at the Gatlinburg VIII meeting in Oxford, England in July 1981.

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

corresponding scalars $\lambda_i \to 0$ so that

$$\int \tilde{K}(s, t)\phi_i(s)\, ds = \lambda_i \psi_i(t), \qquad \int \tilde{K}(s, t)\psi_i(t)\, dt = \lambda_i \phi_i(s).$$

Then if $\tilde{g}(s) = \sum \tilde{\beta}_i \phi_i(s)$, then $\tilde{f}(t) = \sum (\tilde{\beta}_i/\lambda_i)\psi_i(t)$; however, $\tilde{f} \in L_2$ only if $\sum (\tilde{\beta}_i/\lambda_i)^2 < \infty$, which is the Picard condition. This obviously restricts the class of data function $\tilde{g}$ for the problem.

We should add at this point that the rate at which the $\lambda_i \to 0$ depends directly on the smoothness of the operator $\tilde{K}$: for example, when $\tilde{K}$ is the (nonsmooth) Green's function

$$\tilde{K}(s, t) = \begin{cases} s(1-t), & s < t, \\ t(1-s), & s \geqq t, \end{cases}$$

then $\lambda_n = \pi^2/n^2$. And, when $\tilde{K}$ is the (smooth) harmonic continuation operator

$$\tilde{K}(s, t) = \frac{1}{2\pi} \frac{1-\rho^2}{1-2\rho \cos (s-t)\rho^2}$$

on $(0, 2\pi)$, then $\lambda_n = \rho^n (\rho < 1)$. Thus, smooth kernels lead to much more ill-conditioned problems.

Now consider the discrete problem (1.1), which can be derived from (1.2) by applying a specific quadrature rule in t, say

$$\sum_1^n w_j \tilde{K}(s, t_j)\tilde{f}(t_j) = \tilde{g}(s),$$

which, applied at n sample points s_i, gives the linear system $Kf = g$. The discrete analogue of the above expansions for the continuous problem is of course the singular value decomposition (SVD)

$$K = UDV^T,$$

where U and V are orthogonal, and $D = \operatorname{diag}(\sigma_i)$ with $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq 0$. However, to derive some reasonable discrete version of the Picard condition is not so straightforward—we need first of all to define what is meant by a reasonable solution to (1.1).

To avoid scaling problems, we assume the discrete problem has been scaled so that $\|K\| = O(1)$ and $\|g\| = O(1)$; we use l_2 norms throughout the discussion (so that $\|K\| = \sigma_1$ and $\|g\|^2 = \sum_i g_i^2$). Moreover, the $O(1)$ notation means "about the same size as", and does not refer to any kind of asymptotic behavior.

DEFINITION. The vector f is a *reasonable solution* to (1.1) with noise level ε if $\|f\| = O(1)$ and $\|Kf - g\| = O(\varepsilon)$.

The question of whether a given discrete problem (1.1) has a reasonable solution can be decided by means of the SVD of K: if $K = UDV^T$ and we define y and β by $V^T f = y$, $U^T g = \beta$, then

$$\|Kf - g\|_2^2 = \sum_1^n (\sigma_i y_i - \beta_i)^2 \quad \text{and} \quad \|f\|_2^2 = \sum_1^n y_i^2.$$

Thus it is clear that for f to be a reasonable solution, the given data g must be such that

$$\beta_i = O(\sigma_i), \qquad i \leqq i_0, \tag{2.1}$$

with i_0 chosen so that $|\beta_i| < \varepsilon$ for $i > i_0$.

Notice that this is a condition on the discrete data g, and is in some sense a discrete analogue of the Picard condition. Typically, in practice, if the data have noise

level ε, the $|\beta_i|$ decrease to $O(\varepsilon)$ and remain at about this level for $i_0 < i \leqq n$, whereas the σ_i decrease to zero. Thus for there to be a reasonable solution, the $|\beta_i|$ must decrease as fast as the σ_i, down to the noise level.

Also, this condition (2.1) provides a basis for *existence* of a reasonable solution f, but says nothing about uniqueness. Indeed, it is clear again from the SVD expansion that there will be many reasonable solutions; for $i > i_0$ (assuming $\sigma_i < \varepsilon$ for $i > i_0$ as well) the y_i are essentially arbitrary. We will return to this point later when comparing the solutions obtained by various methods. Here we wish to emphasize that all such solutions are equally valid from a strictly computational point of view; in order to obtain unique solutions, we must further restrict the problem using other means (for example by restricting the class of solutions allowed or specifying a particular expansion for the solution).

We also feel this approach (of reasonable solutions) is more appropriate than that of conditioning given in Varah (1979), because it does not depend on the method used to solve the problem.

One may well ask if the concept of "reasonable solution" introduced here is too broad, leading as it does to nonunique solutions. We feel it is at least a *starting point* for discussion of methods and solutions of discrete ill-posed problems. For particular problems, it can be further restricted using the properties appropriate to that problem, and in so doing may lead to unique solutions. For example, with inverse Laplace transform problems (see the examples following), it may be appropriate to specify the asymptotic nature of the solution.

As well, although we have used L_2 and l_2 norms because of the connection between the L_2 theory for the continuous problem and the singular value decomposition for the discrete problem, other norms may be more appropriate. Indeed, the inverse Laplace transform which we use as the basis of our numerical examples is such a case, since the kernel is not L_2. This appears to be of little consequence, however, as far as the discrete methods and solutions are concerned.

3. The numerical methods.

(a) SVD. This is the truncated SVD method, which produces f as $f = \sum_1^k y_i v^{(i)}$, where $y_i = \beta_i/\sigma_i$ and k is the free parameter. If k is chosen as in § 2 with noise level ε known, and if the problem has a reasonable solution, then it is easy to see that this method always produces such a reasonable solution: $\|Kf - g\|_2 = O(\varepsilon)$ and $\|f\|_2 = O(1)$. Since $y_i = 0$ for $i > i_0$, the SVD solution can be said to provide the *minimal* reasonable solution possible.

Moreover, even if the noise level ε is not known, one can examine the sequences $\{\sigma_i\}$, $\{\beta_1\}$ and take k, the cutoff point, where the σ_i become smaller than the $|\beta_i|$. This gives a reasonable solution for ε chosen accordingly.

(b) LS. This is the original regularization method, which produces f as

$$f_\alpha = \min_f (\|Kf - g\|_2^2 + \alpha^2 \|f\|_2^2)$$

or equivalently as the solution of the normal equations

$$(K^T K + \alpha^2 I) f_\alpha = K^T g.$$

In examining the nature of the solution f_α, it is most convenient to again use the SVD expansion; then

$$f_\alpha = \sum_1^n \left(\frac{\beta_i}{\sigma_i + \alpha^2/\sigma_i} \right) v^{(i)}. \tag{3.1}$$

At first sight, it appears that this is rather different from the SVD solution in (a), since all vectors $v^{(i)}$ contribute. However for properly chosen values of the free parameter α, most of these are damped, and we are left with a solution very much like the SVD. To see this, consider the same situation as before, with $\sigma_i \to 0$, $|\beta_i| \to \varepsilon$ and $\beta_i = O(\varepsilon)$ for $i > i_0$. First, if $\alpha \ll \varepsilon$ and we have some $\sigma_i = \alpha$, then the ith term in (3.1) is $(\beta_i/2\alpha)v^{(i)}$ and $|\beta_i/2\alpha| \gg 1$ so we cannot have a reasonable solution—so we must choose $\alpha \geqq \varepsilon$. Moreover if α is chosen roughly equal to ε, then for $i \leqq i_0$, the ith coefficient $(\beta_i/\sigma_i)/(1+\alpha^2/\sigma_i^2) \cong \beta_i/\sigma_i$ since $\sigma_i \geqq \varepsilon$ and for $i > i_0$, $(\beta_i/\sigma_i)/(1+\alpha^2/\sigma_i^2) \cong 0$ since $\sigma_i \to 0$.

Thus the LS solution, with appropriate α, is close to the SVD solution, and this is certainly borne out by numerical experience. The most important connection is that both give a solution in terms of the expansion of singular vectors $\{v^{(i)}\}$ of K.

(c) MLS. This is the modified regularization method, which produces f as

$$f_\alpha = \min_f (\|Kf - g\|_2^2 + \alpha^2 \|Lf\|_2^2)$$

or equivalently as the solution of the normal equations

$$(K^T K + \alpha^2 L^T L) f_\alpha = K^T g.$$

Here L is usually some discrete approximation to a derivative operator, and we in fact use simple forward differences to get L in our numerical examples. However, other choices of L may be appropriate in other cases: for example a discrete Sobolev norm using a linear combination of differences of order 0, 1, and 2. Indeed, the relation between the choice of L and the discrete solution generated is not well understood and needs to be investigated further.

Here the appropriate expansion is not the SVD of K, but the generalized SVD of the matrix pair (K, L) as in van Loan (1976):

$$K = UD_a X^{-1}, \qquad L = VD_b X^{-1},$$

with U and V orthogonal, D_a and D_b diagonal, and X the matrix which simultaneously diagonalizes the symmetric pair $(K^T K, L^T L)$. Using this decomposition, the solution f_α can be written as an expansion in the vectors $x^{(i)}$, the columns of X:

$$f_\alpha = \sum_1^n \left[\frac{\beta_i}{a_i + \alpha^2 b_i^2/a_i}\right] x^{(i)}$$

where again $\beta = U^T g$ and $\{a_i\}$, $\{b_i\}$ are the elements of D_a, D_b, ordered so that the $\{a_i\}$ (which are the singular values of KX) are decreasing. Notice that this has the same form as the LS solution, except that the $\{x^{(i)}\}$ have replaced the $\{v^{(i)}\}$. We should add that although the above expansion gives a nice way of representing the MLS solution, it may be easier computationally to use the regular SVD method to solve the system.

(d) KX. This is the truncated expansion method, using any set of vectors $\{x_k^{(i)}\}_1^k$—one asks for the best solution of $Kf = g$ using vectors of the form $f = \sum_1^k c_i x^{(i)}$. Thus we find c as $\min_c \|KXC - g\|_2^2$, solving using the QR method or normal equations on KX. In particular, using the $x^{(i)}$ as above in (c) would give $c_i = \beta_i/a_i$, in which case this method would bear the same relation to MLS as SVD does to LS. Indeed, we mention this method only for completeness, and do not include any numerical results; it can be useful if we can restrict the form of the solution f to such an expansion—again

this is adding more constraints to the problem than in the other methods, and may result in there being a "unique" solution.

4. Numerical examples. All our examples stem from discretizations of the inverse Laplace transform:

$$\int_0^\infty e^{-st} f(t)\, dt = g(s).$$

This problem illustrates nicely all of the pitfalls associated with ill-posed problems with smooth kernels, when different data $g(s)$ are used. The integral is discretized using n-point Gauss–Laguerre quadrature, so approximations are generated for $f(t)$ at the Gauss–Laguerre abscissae $\{t_j\}_1^n$. The sample points $\{s_i\}$ can vary, but normally n points equally spaced in $(0, n]$ gave the best results. Although all our examples come from known f and g, so that we can measure the "error" in f, this is rather artificial: in practical cases the data g will only be known at specific points $\{s_i\}$, and we can only measure the error in the residual sense ($\|Kf - g\|$). We should add that results may be very different with nonsmooth kernels, and that more care may be necessary in discretizing the problem.

For this problem, the singular vectors $v^{(i)}$ are asymptotic to zero (i.e., $v_j^{(i)} \to 0$ as $j \to n$), whereas the X-vectors $x^{(i)}$ are asymptotic to 1.0. As well, both sets of vectors satisfy the oscillation property: $v^{(i)}$ and $x^{(i)}$ each change sign $(i-1)$ times.

Example 1.

$$g(s) = \frac{1}{s+0.5}, \qquad f(t) = e^{-t/2}.$$

Here we used $n = 10$ points with $\{s_i\}_1^n$ equally spaced in $[1, 10]$. For SVD, it is instructive to see the $\{\sigma_i\}$ and $\{\beta_i\}$ explicitly:

σ_i	.74	.28	.055	$.29\times10^{-2}$	$.17\times10^{-4}$	$.57\times10^{-8}$
β_i	−.92	−.018	−.014	$.61\times10^{-2}$	$-.16\times10^{-2}$	$.57\times10^{-2}$

σ_i	$.51\times10^{-13}$	$.52\times10^{-20}$	$.36\times10^{-25}$	$.99\times10^{-29}$
β_i	$.47\times10^{-2}$	$-.34\times10^{-2}$	$.19\times10^{-2}$	$-.12\times10^{-2}$

Notice that although no noise level ε is known, the $|\beta_i|$ decrease to about 10^{-2} while the σ_i decrease to zero. Clearly we should take $k = 3$; this gives a residual of .012 and a maximum error of .1. For the LS and MLS methods, it is not so clear how to choose α. Indeed it appears that for a large range of α, we can obtain reasonable solutions, even though they may be very different. This is particularly true of MLS, where the expansion vectors $x^{(i)}$ are asymptotic to 1, not 0, and the asymptotic nature of the solution to the inverse Laplace transform is not well determined by the data $g(s_i)$. We give the basic results in Table 1 and the graphical results in Fig. 1 (for SVD and LS) and Fig. 2 (for MLS). Notice that the discrete solutions generated have been converted to continuous curves for the graphs. This has been done by cubic spline interpolation, choosing the knots interactively so that the curves represent the solution appropriately.

TABLE 1

	LS		MLS	
α	$\|Kf-g\|$	$\max\|f_i-f(t_i)\|$	$\|Kf-g\|$	$\max\|f_i-f(t_i)\|$
.1	.023	.083	.017	.40
.01	.012	.17	.012	.22
.001	.010	1.7	.012	1.6

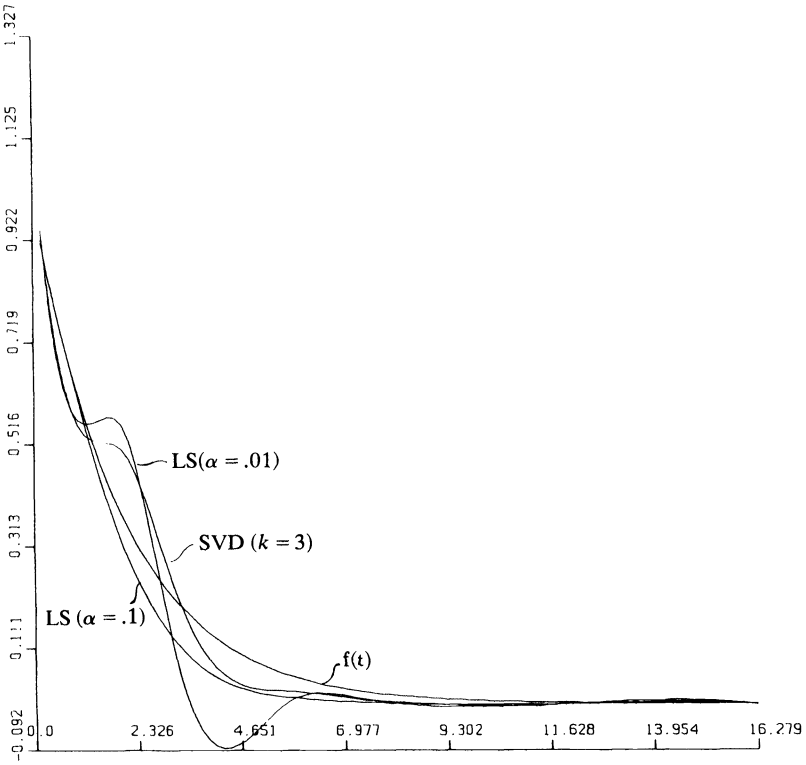


FIG. 1. SVD *and* LS *solutions for Example* 1.

Example 2.

$$g(s)=\frac{1}{s}-\frac{1}{s+0.5}, \qquad f(t)=1-e^{-t/2}.$$

Again we used 10 points with $\{s_i\}$ equally spaced in [1, 10]. Of course the $\{\sigma_i\}$ are as in Example 1, and the $\{\beta_i\}$ are as follows:

β_i	$-.31$	$-.16$	$-.034$	$-.32\times10^{-2}$	$.81\times10^{-3}$	$-.57\times10^{-3}$
β_i		$-.34\times10^{-3}$	$.17\times10^{-3}$	$-.59\times10^{-4}$	$.49\times10^{-4}$	

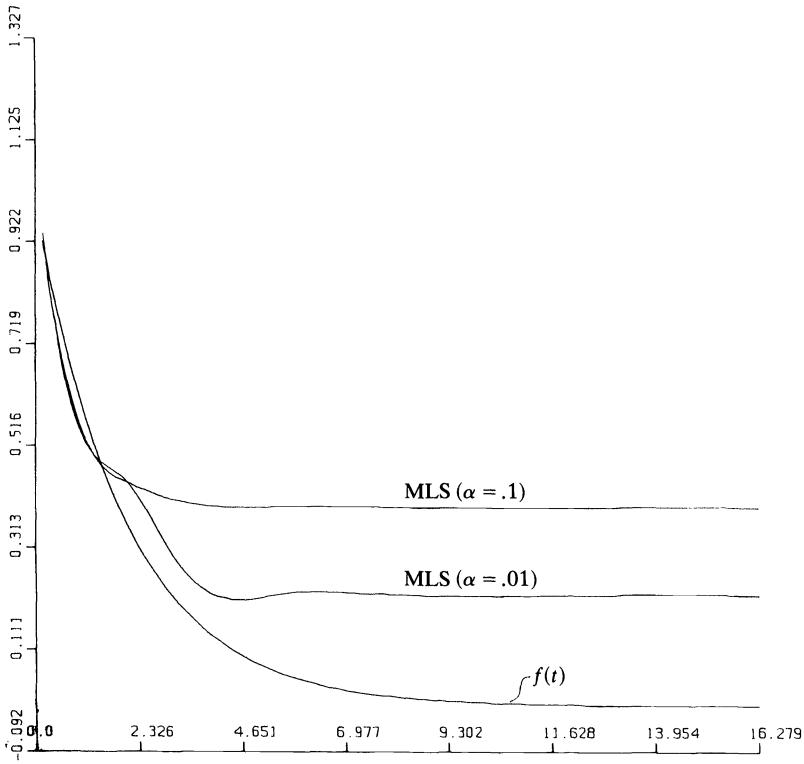


FIG. 2. MLS *solutions for Example* 1.

In this case the $\{\beta_i\}$ decay more rapidly to about 10^{-4}, so that it is appropriate to take $k = 4$ terms (or possibly $k = 3$); this gives a residual of .0011, and an error of 1.0 because the solution $f(t)$ is asymptotic to 1.0 (not 0 as in Example 1). However, the SVD solution is still reasonable from the point of view of our definition, and it can only be seen as incorrect if more information is supplied about the problem.

For LS and MLS, the same comments apply as in Example 1. Here, although the MLS solution may look better (for some α) because of its asymptotic nature, there is no way to guarantee this for a given practical problem, again unless more is specified about the problem. We take the view that *all* solutions given here are reasonable. We again give LS and MLS results in Table 2 and the graphical solutions in Figs. 3 and 4.

TABLE 2

	LS		MLS	
α	$\|Kf-g\|$	$\max \|f_i - f(t_i)\|$	$\|Kf-g\|$	$\max \|f_i - f(t_i)\|$
.1	.032	1.0	.013	.39
.01	.0033	1.0	.0017	.24
.001	.0011	1.0	.0011	.22

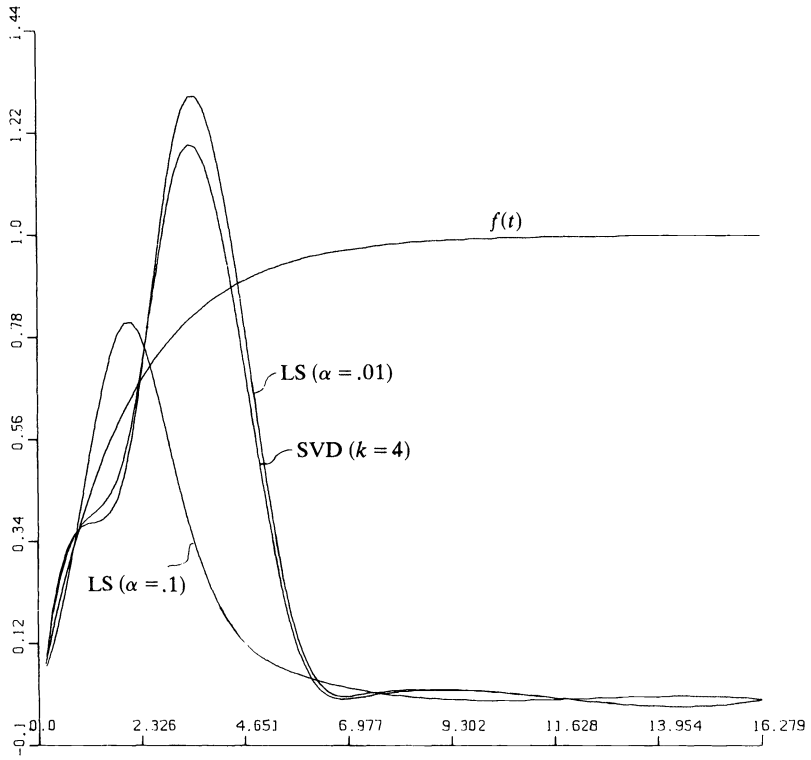


FIG. 3. SVD *and* LS *solutions for Example* 2.

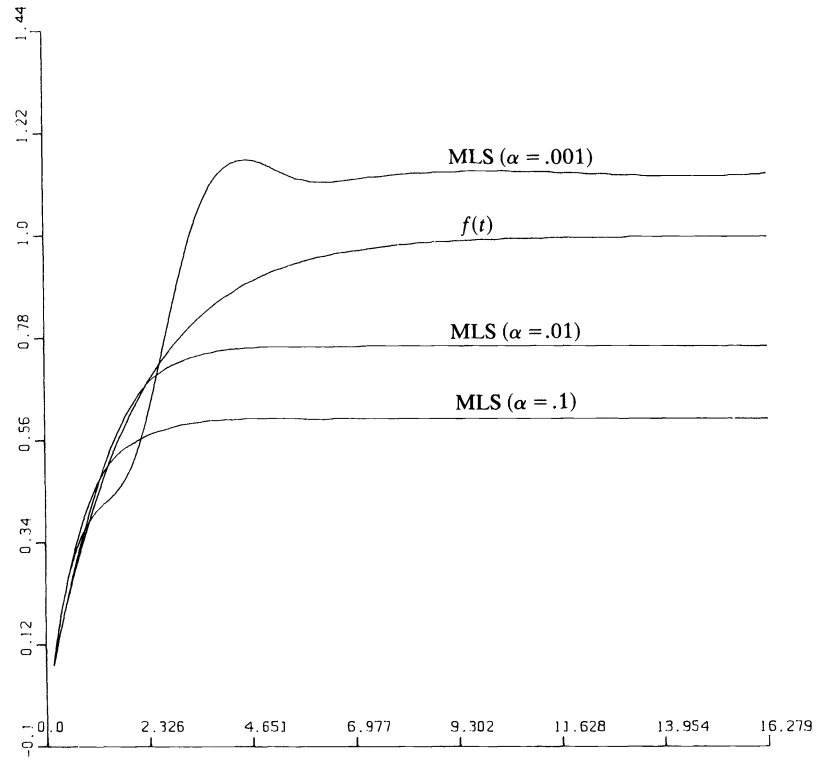


FIG. 4. MLS *solutions for Example* 2.

Example 3.

$$g(s)=\frac{2}{(s+0.5)^3}, \qquad f(t)=t^2e^{-t/2}.$$

Here we used 20 points with the $\{s_i\}$ equally spaced in $[1, 10]$. Truncating the SVD expansion at $k=7$ terms gave a residual of $.25\times10^{-4}$ and an error of 0.17. The LS and MLS solutions (for various α) are given in Table 3, with plots in Figs. 5 and 6.

TABLE 3

	LS		MLS	
α	$\|Kf-g\|$	$\max\|f_i-f(t_i)\|$	$\|Kf-g\|$	$\max\|f_i-f(t_i)\|$
10^{-2}	$.60\times10^{-2}$	.54	$.27\times10^{-2}$	1.8
10^{-3}	$.28\times10^{-3}$	.30	$.27\times10^{-3}$	1.3
10^{-4}	$.35\times10^{-4}$	.17	$.32\times10^{-4}$	1.0
10^{-5}	$.21\times10^{-4}$	.71	$.24\times10^{-4}$	.29

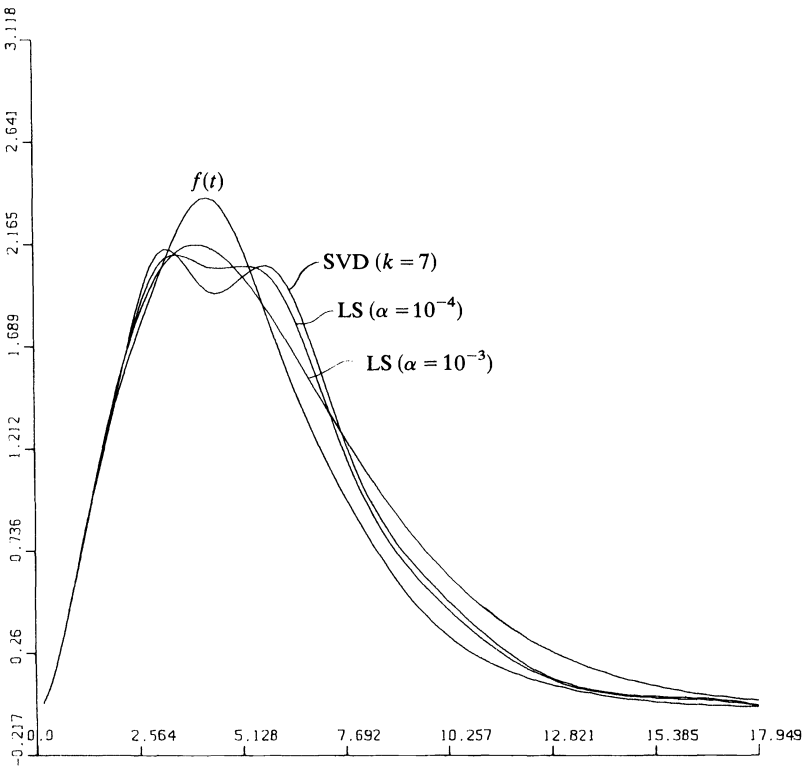


FIG. 5. SVD *and* LS *solutions for Example* 3.

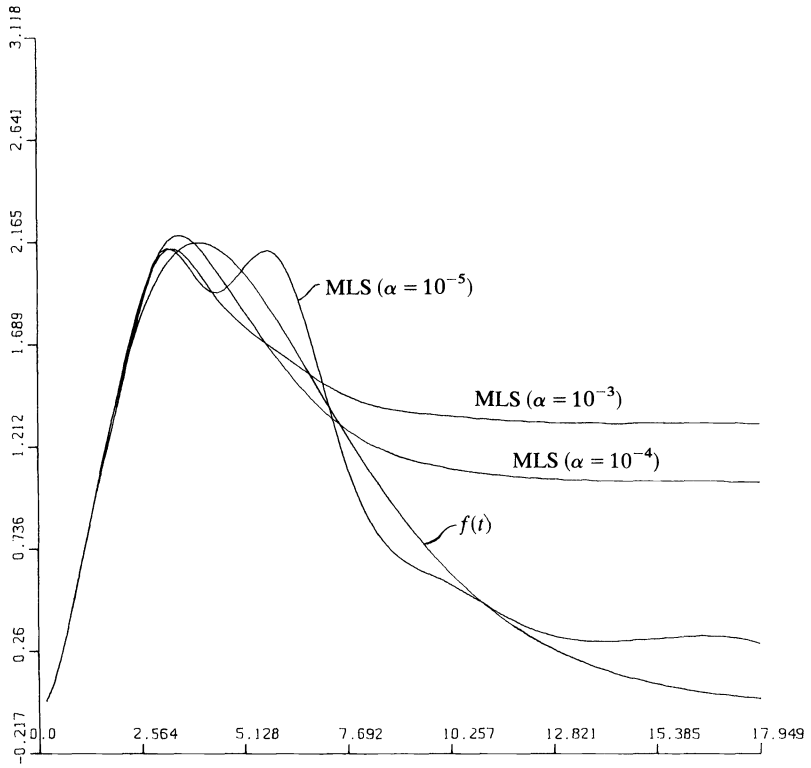


FIG. 6. MLS *solutions for Example* 3.

Example 4.

$$g(s) = \frac{e^{-2s}}{s}, \qquad f(t) = \begin{cases} 0, & t \leqq 2, \\ 1, & t > 2. \end{cases}$$

Here is one example of a discontinuous transform. The best SVD solution (with $k = 4$) had a residual of $.79 \times 10^{-4}$ and an error of 1.0. The LS and MLS solutions are given below in Table 4, and graphically in Figs. 7 and 8.

Of course, the SVD and LS solutions are asymptotic to zero, and again the MLS solutions can be asymptotic to almost anything, depending upon the value of α. We feel that the most important point here is that all the solutions given here are reasonable, and can only be specified more precisely if more constraints are put on the problem.

TABLE 4

	LS		MLS	
α	$\|Kf-g\|$	$\max \|f_i - f(t_i)\|$	$\|Kf-g\|$	$\max \|f_i - f(t_i)\|$
.1	.03	1.0	.021	.65
.01	$.38 \times 10^{-2}$	1.0	$.23 \times 10^{-2}$	.45
.001	$.42 \times 10^{-3}$	1.0	$.33 \times 10^{-3}$	.34
.0001	$.77 \times 10^{-4}$	1.0	$.99 \times 10^{-4}$	.31

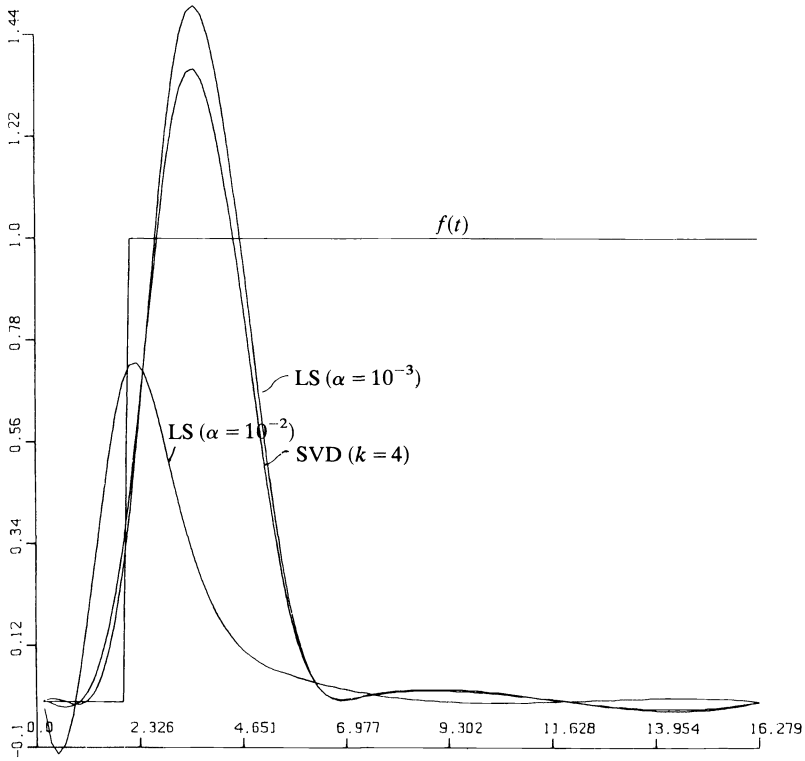


FIG. 7. SVD *and* LS *solutions for Example* 4.

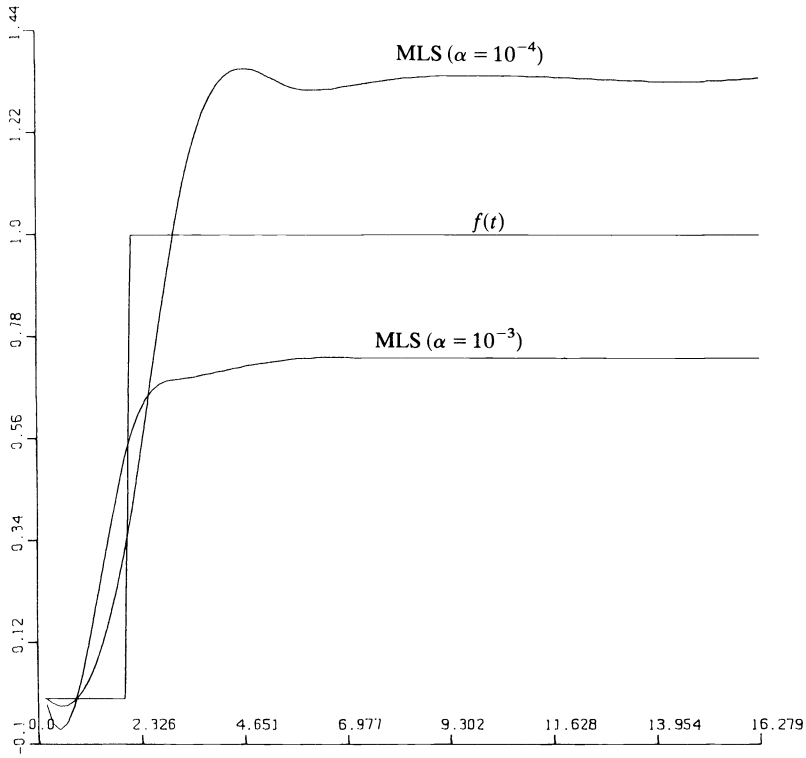


FIG. 8. MLS *solutions for Example* 4.

5. Choice of the free parameter. As we indicated earlier, the choice of k in the SVD method is fairly clear from the decay of the $\{\beta_i\}$ and $\{\sigma_i\}$. Unfortunately, this does not appear to be the case for choosing α in the LS and MLS methods. Various strategies have been put forward for specifiing α, notably the technique of generalized cross-validation (GCV), which has been successfully used for problems with nonsmooth kernels (see Golub, et al. (1979)). Here α is chosen to minimize the function

$$V(\alpha) = \frac{1/n\|g - Kf_\alpha\|_2^2}{(1/n \text{ tr } (I - A(\alpha)))^2}$$

where $A(\alpha) = K(K^TK + \alpha^2 L^TL)^{-1}K^T$. Except for the $1/n$ factors, the numerator is just the square of the residual, and when $L = I$ (i.e., LS method) the denominator can be expressed as $(\sum_1^n \alpha^2/(\alpha^2 + \sigma_i^2))$. In this case, notice that when some σ_i are very small (as for our smooth kernel), the corresponding terms in this sum will be very close to 1.0 for large ranges of α; as well we have found that the residual is also nearly constant for large ranges of α, so that $V(\alpha)$ will be *very flat* and it will be very difficult to minimize. As an example of this, we plotted $V(\alpha)$ on a log-log scale for $0 \geqq \log \alpha \geqq -20$ for Example 1 above, and present the results in Fig. 9. Similar results were obtained for the other examples.

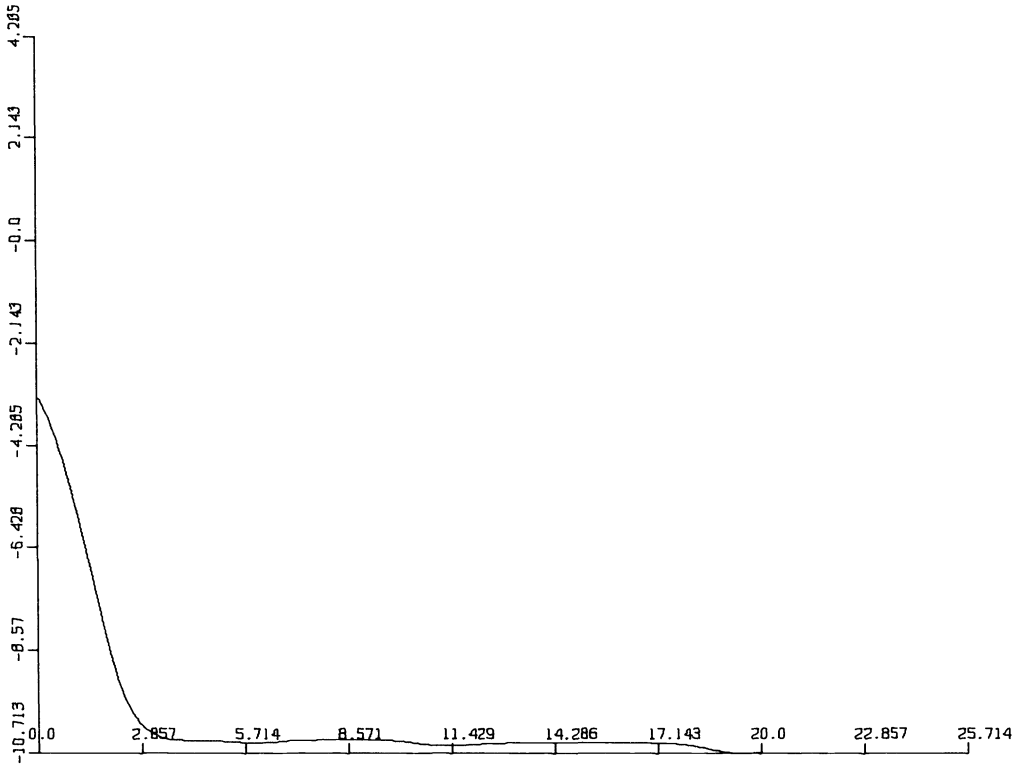


FIG. 9. *Log-log plot of* GCV *function for Example* 1.

Thus it seems impossible to choose α using this technique. However, we feel this is not due to any intrinsic fault with the technique, but because *reasonable* solutions in our sense can be obtained over a wide range of the parameter α, even though they may look very different.

REFERENCES

A. BJÖRCK AND L. ELDEN (1979), *Methods in numerical algebra for ill-posed problems*, in International Symposium on Ill-Posed Problems: Theory and Practice, University of Delaware, Newark, DE.

G. GOLUB, M. HEATH AND G. WAHBA (1979), *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21, pp. 215–223.

C. VAN LOAN (1976), *Generalizing the singular value decomposition*, SIAM J. Numer. Anal., 13, pp. 76–83.

J. VARAH (1979), *A practical examination of some numerical methods for linear discrete ill-posed problems*, SIAM Rev., 21, pp. 100–111.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0003 $01.25/0

A COMPARISON OF SOME METHODS FOR SOLVING SPARSE LINEAR LEAST-SQUARES PROBLEMS*

ALAN GEORGE†, MICHAEL T. HEATH‡ AND ESMOND NG†

Abstract. The method of normal equations, the Peters–Wilkinson algorithm and an algorithm based on Givens rotations for solving large sparse linear least squares problems are discussed and compared. Numerical experiments show that the method of normal equations should be considered when the observation matrix is sparse and well conditioned. For ill-conditioned problems, the algorithm based on Givens rotations is preferable.

Key words. sparse linear least-squares problems, Peters–Wilkinson method, Givens rotations, normal equations

1. Introduction. Let A be an $m \times n$ sparse matrix with $m \geqq n$, and consider the system of linear equations

$$Ax = b, \tag{1.1}$$

where b and x are vectors of length m and n respectively. In general, there may not exist a solution x such that (1.1) is exactly satisfied. Thus, (1.1) is usually solved in the least-squares sense; that is, the solution x is chosen to minimize the Euclidean norm of the residual vector

$$r = Ax - b. \tag{1.2}$$

Throughout this paper, we will assume that the columns of A are linearly independent. Under this assumption, it is easy to show that the solution x is unique and satisfies the symmetric positive definite $n \times n$ system of linear equations

$$A^T Ax = A^T b, \tag{1.3}$$

which is referred to as the system of normal equations.

When $A^T A$ is sparse, the normal equations in (1.3) can be solved efficiently since the problem of solving large sparse symmetric positive definite systems is already well understood [7].

However, it is well known that computing $A^T A$ explicitly may not be desirable since the condition number of $A^T A$ is the square of that of A. Thus the matrix $A^T A$ may be quite ill-conditioned if A is poorly conditioned, and the solution x will be sensitive to perturbations in the data. Moreover, severe numerical cancellation and roundoff may occur in computing $A^T A$ [10].

Several numerically stable algorithms have been proposed for solving (1.1) without computing $A^T A$ explicitly. In this paper, we will compare two such algorithms. In addition to being numerically stable, the two algorithms also attempt to exploit sparsity in A.

The first algorithm was originally proposed by Peters and Wilkinson for solving (1.1) without considering the sparsity of A [13]. Recently, Björck and Duff have

* Received by the editors October 6, 1981, and in revised form May 25, 1982. This research was sponsored jointly by the National Geodetic Survey of the National Ocean Survey, NOAA, U.S. Department of Commerce, under Interagency Agreement 40-1108-80, 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 Canadian Natural Sciences and Engineering Research Council.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

‡ Mathematics and Statistics Research Department, Computer Sciences Division, Union Carbide Corporation, Oak Ridge, Tennessee 37830.

advocated its use for sparse A [2]. The algorithm is based essentially on Gaussian elimination with complete pivoting. The second algorithm is due to George and Heath [6], and is based on the use of Givens rotations. As a basis for comparison, we have also included results for a conventional normal-equations implementation.

An additional algorithm for least-squares problems, the augmented matrix or sparse tableau method, has been proposed in the sparse case by Hachtel [8]. This algorithm was included in the earlier comparisons of Duff and Reid [3], but in our tests it required considerably more storage than competing methods (4 to 8 times as much) and has therefore not been included in our final tabulations.

An outline of the remainder of this paper is as follows. In §§ 2 and 3, we review briefly the two algorithms. Then some numerical experiments are provided in § 4 and some concluding remarks appear in § 5.

2. The Peters–Wilkinson (P–W) algorithm. The first step of this algorithm is the computation of an LU-decomposition of A using both row and column interchanges. Thus, we have

$$PAQ = LU, \tag{2.1}$$

where P and Q are respectively $m \times m$ and $n \times n$ permutation matrices, L is an $m \times n$ unit lower trapezoidal matrix and U is an $n \times n$ upper triangular matrix. When A is sparse, the matrices P and Q are chosen to maintain numerical stability and preserve sparsity simultaneously. Then (1.1) can be written as

$$PAQQ^T x = Pb \quad \text{or} \quad LUQ^T x = Pb. \tag{2.2}$$

If $y = Q^T x$ and $d = Pb$, then (2.2) becomes

$$LUy = d. \tag{2.3}$$

When $m = n$, the matrix L is unit lower triangular, and the solution x can be obtained by solving two triangular systems

$$Lz = d \quad \text{and} \quad Uy = z$$

and then computing

$$x = Qy.$$

However, if $m > n$, then z is the least-squares solution to the overdetermined system

$$Lz = d. \tag{2.4}$$

Apparently, nothing has been gained so far. However, experience has shown that by using an appropriate pivoting strategy, the condition of L can be controlled and any ill-conditioning can be isolated in U [13]. Thus (2.4) can safely be solved via the normal equations; that is, we can solve

$$L^T Lz = L^T d. \tag{2.5}$$

Then the solution x can be obtained by solving

$$Uy = z \quad \text{and} \quad x = Qy.$$

This algorithm is an attractive candidate for solving (1.1) when A is sparse, because there already exist efficient sparsity-exploiting algorithms for computing the LU-decomposition of A [4] and for solving (2.5) [5].

3. An algorithm based on Givens rotations. The basic step in this algorithm is to determine an $m \times m$ orthogonal matrix Q which reduces A to an upper trapezoidal matrix

$$QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{3.1}$$

where R is an $n \times n$ upper triangular matrix. Then (1.1) can be written as

$$QAx = Qb, \tag{3.2}$$

and the least squares solution is unchanged due to the orthogonal invariance of the Euclidean norm. Let

$$Qb = \begin{pmatrix} c \\ d \end{pmatrix},$$

where c and d are vectors of length n and $(m-n)$ respectively. Then (3.2) becomes

$$\begin{pmatrix} R \\ 0 \end{pmatrix} x = \begin{pmatrix} c \\ d \end{pmatrix},$$

and the solution x can be obtained by solving a triangular system

$$Rx = c. \tag{3.3}$$

Note that if $m = n$, the vector d will be null.

Even though it is well known that application of orthogonal transformations is numerically stable, this approach has not been popular for solving (1.1) when A is sparse. Apparently it has been assumed that orthogonal transformations will cause unacceptable fill-in during the reduction of A.

Recently, George and Heath have proposed a new efficient way to compute R based on Givens rotations [6]. They observe that the upper triangular matrix R is mathematically equivalent to the upper triangular Cholesky factor of the matrix A^TA. Furthermore, any column permutation on the columns of A induces a symmetric permutation on the rows and columns of A^TA, and vice versa. Thus one can choose a column permutation P for A so that the Cholesky factor of P^TA^TAP suffers low fill-in. Note that since no row or column interchanges are necessary during the Cholesky decomposition [14], the pivotal sequence is known once the permutation matrix P (or ordering) has been determined. Thus the positions of the nonzeros, and the storage requirement for the Cholesky factor (hence R) can be determined before the actual numerical decomposition begins. These two steps can be done without actually carrying out the transformation on A; only the nonzero structure of A is required. The reader is referred to [6] for more details.

After the data structure for R has been determined and set up, the rows of A can then be rotated one by one into R using Givens rotations. Thus, this algorithm has the advantage that the storage requirement can be determined and fixed before any numerical computation is carried out. The orthogonal matrix Q is usually dense and therefore is not stored, but rather the Givens rotations are simply discarded as they are used. If it is anticipated that additional problems will be solved with the same matrix A but different right-hand sides b, then the rotations could be written onto an external file, or the system $R^TRx = A^Tb$ could be solved using the already computed R. Assuming Q is not stored, the Givens algorithm requires no more storage than the normal equations.

Although the sparsity of A^TA, and hence R, is not affected by the order in which the rows are processed, the number of arithmetic operations needed to carry out the numerical factorization is dependent on the row ordering. A heuristic row ordering strategy such as that proposed in [6] can substantially reduce the numerical factorization time for some classes of problems. However, this gain is at least partially offset by the time required to compute the row ordering, and also, depending on implementation, increases in storage or I/O traffic. For these reasons, in our tests the original row ordering was used for all problems.

4. Numerical experiments. Two sparse matrix packages were used to implement the P–W and Givens algorithms. The first package was MA28 from Harwell [4], which was designed to solve general sparse systems of equations. It was used to compute the LU-decomposition of A in the P–W algorithm (2.1). The pivoting strategy, due to Markowitz [11], attempts to maintain numerical stability and preserve sparsity at the same time. This involves the use of a so-called threshold pivoting technique; during the decomposition, an element of the partially reduced matrix may be considered as a pivot if its magnitude is larger than the product of a user-specified threshold parameter and the absolute value of the element with the largest magnitude in that row. Unfortunately, such a pivoting strategy controls the condition of U rather than that of L. Björck and Duff [2] suggest instead a complete threshold pivoting strategy. In order to avoid modifying MA28, however, and in view of the greater expense of a complete pivoting strategy, we followed the simple expedient of applying MA28 to A^T, which leads to a satisfactorily conditioned L. In order to show how much difference this makes, we also present the results obtained when MA28 is applied to A. The threshold parameter we used in the experiments was 0.1.

The second package, SPARSPAK, developed at the University of Waterloo [5] and designed to solve sparse symmetric positive definite systems, was used to determine the nonzero structure of the Cholesky factor R in the Givens algorithm and to solve the normal equations in (2.5) in the P–W algorithm. In both cases, the orderings (i.e. the column and row permutations) used were provided by a minimum-degree algorithm [7], which is a symmetric version of the Markowitz pivoting strategy.

The experiments were performed on an IBM 4341 and all times reported are in seconds. The storage requirements reported were provided either explicitly or implicitly through some variables appearing in the internal labelled common blocks used by the two packages. The programs were written in FORTRAN and compiled using the IBM Extended Optimizing Compiler and single precision floating point arithmetic. Single precision was used in order to emphasize any numerical differences among the algorithms.

Our test set consists of ten problems, some of which are real and some of which were artificially generated. We believe all of them are representative of problems arising in finite element applications and in surveying. Table 1 contains some characteristics of each of the ten problems, and some more details about the artificially generated problems follow.

One type of artificially generated problem involves a network, typical of those arising in geodetic adjustment applications [9]. Each network may be regarded as being composed of q^2 "junction boxes", connected to their neighbors by chains of length l. An example is shown in Fig. 1 with $q = 3$ and $l = 2$. In our test problems, $l = 5$. There are two variables associated with each vertex in the network. There are μ observations involving four variables associated with each pair of vertices joined by an edge, and there are μ additional observations involving six variables associated with each triangle in the network. In our experiments $\mu = 1$ or 2.

TABLE 1
Characteristics of the test problems.

Problem number	Number of rows	Number of columns	Number of nonzeros	Remarks
1	313	176	1557	Sudan survey data.
2	1033	320	4732	Least-squares problem in the analysis of gravity-meter observations provided by M. A. Saunders (well-conditioned) [12].
3	1033	320	4719	Problem similar in structure to Problem 2 but ill-conditioned.
4	1850	712	8755	Similar to Problem 2, but larger in size.
5	1850	712	8636	Problem similar in structure to Problem 4, but ill-conditioned.
6	784	225	3136	Artificial—15×15 grid problem.
7	1444	400	5776	Artificial—20×20 grid problem.
8	1512	402	7152	Artificial—3×3 network, $\mu = 2$.
9	900	269	4208	Artificial—4×4 network, $\mu = 1$.
10	900	269	4208	A geodetic network problem provided by the U.S. National Geodetic Survey (ill-conditioned).

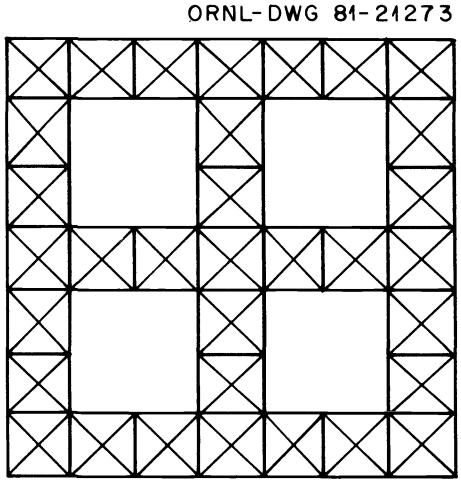


FIG. 1. *A* 3×3 *network with* $l = 2$.

The second class of artificially generated problems consists of some simulated least-squares problems on $q \times q$ square grids. These problems are typical of those arising in the natural factor formulation of finite element methods [1]. Each grid consists of $(q-1)^2$ small squares. An example is shown in Fig. 2 with $q = 3$. Associated with each of the q^2 grid points is a variable, and associated with each small square are four observations involving the four variables at the corners of the square.

The numerical values in the observation matrices and the right-hand sides of the artificial problems were generated using a random number generator.

For completeness, we include a summary of the actual computational phases for the methods under study, along with some remarks about the implementation when it has implications with regard to the reporting of storage requirements. As a basis for comparison, our numerical experiments include the solution of the test problems using the conventional normal equations approach (1.3), using SPARSPAK.

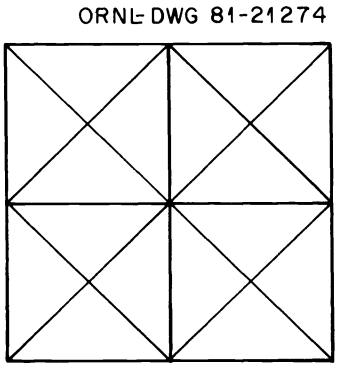


FIG. 2. *A* 3×3 *grid.*

Method of normal equations. There are four distinct phases in the method of normal equations:

(1) Ordering phase. This determines a symmetric permutation P for the matrix A^TA via a minimum degree algorithm.
(2) Storage allocation phase. After the permutation (or ordering) is determined, the data structure required to store the nonzeros is determined and set up.
(3) Numerical computation phase. A^TA and A^Tb are computed numerically. It is assumed that the rows of A reside on an external file. Thus the time required for this phase includes some I/O overhead.
(4) Solution phase. Factor the matrix P^TA^TAP and solve for the solution x.

The algorithm based on Givens rotations. There are three phases in the Givens algorithm, and they are similar to those in the method of normal equations:

(1) Ordering phase. This determines a symmetric permutation P for the matrix A^TA via a minimum degree algorithm.
(2) Storage allocation phase. After the permutation P is determined, the data structure for the nonzeros is determined and set up.
(3) Solution phase. The rows of AP are rotated into R using Givens rotations (which are also applied to the right-hand side b), and the solution x is computed.

Note that in phases 1 and 2 of both this scheme and the normal equations scheme, only the nonzero structure of A is required. The numerical values of A are used in the last phase (last two phases for normal equations).

The Peters–Wilkinson algorithm. The P–W algorithm is more complicated than the method of normal equations and the Givens algorithm in terms of implementation. There are six phases:

(1) Initial LU-decomposition. This computes an LU-decomposition of A (or A^T) as in (2.1).
(2) Ordering phase. This determines a symmetric permutation S for the matrix L^TL using a minimum degree algorithm.
(3) Storage allocation phase. After the symmetric permutation is determined, the data structure for storing the nonzeros in S^TL^TLS is determined and set up.
(4) Numerical computation phase. The matrix S^TL^TLS and vector S^TL^Td are computed numerically.
(5) Solution phase. The matrix S^TL^TLS is factored and the solution to the normal equations in (2.5) is computed.

(6) Back substitution phase. The solution to the original least-squares problem is finally computed using the upper triangular matrix obtained in the first phase.

Note that the numerical values of A are required only in the first phase. In phases 2 and 3, only the nonzero structure of L is needed. Thus the storage required for the initial LU-decomposition can be released by writing it onto an external file and reading it back when it is needed in phases 4 and 6. This means that the storage required in the solution phase need only include the storage required for L^TL; it need not include any storage required for the upper triangular matrix U which is obtained in the initial LU-decomposition. The matrix U remains on external storage, and is read back in the back substitution phase. Thus, the time reported in Table 2 includes some I/O overhead.

The package MA28 has the property that its execution time varies with the amount of storage provided to it. For any given problem, it requires a certain minimum amount s, which it reports, but it also may execute substantially faster if more than s is provided. This presents us with a reporting problem that is not present with the other methods under study. In order to be scrupulously fair to the P–W scheme, we have reported s under "max. store", but in order to obtain execution times we provided enough storage so that providing any more made no appreciable difference to execution time. The extent to which this might favor the P–W results is illustrated in Table 2, which contains results for Problem 1 of our test set. Much more sensitive examples can be found.

TABLE 2
Execution time versus storage for MA28 *applied to Problem* 1.

Storage	Problem time	Total time
$1.0s$	4.61	5.68
$1.1s$	3.74	4.85
$1.2s$	3.54	4.62
$1.5s$	3.62	4.75

The "max. store" reported in Tables 3–6 is the maximum number of storage locations required for any phase in each method. It includes the space required to store the numerical values, the permutations, and pointers for accessing the numerical values. In our implementation, both floating point numbers and integers occupy one storage location. In the normal-equations and Givens schemes, the factor-solve phase requires the most storage, while in the P–W scheme it is the initial LU-factorization which requires the most storage.

The "problem time" reported in Tables 2–6 is the total time in seconds required to carry out the major steps in each method. The "total time" includes additional overhead time such as disk I/O (but excludes time required for computing the relative error and residual). It should be noted that this overhead time is larger for P–W using A^T than for P–W using A. The difference is due to the time required in the former case to rearrange the data structure from column to row orientation.

The relative error reported in Tables 3–6 was determined by comparing the single precision results with a double precision solution computed using Givens rotations. The relative error is computed as $\|x - x^*\|_\infty / \|x^*\|_\infty$, where x^* is the double precision solution. The residual reported is $\|b - Ax\|_2$.

TABLE 3
Method of normal equations.

Problem	Max. store	Problem time	Total time	Rel. error	Residual
1	6821	2.46	2.46	2.02E−5	3.62E+0
2	10189	14.08	14.08	3.86E−3	8.51E−1
3	...	...	...	...	...
4	23233	25.85	25.85	1.95E−3	1.43E+0
5	23105	25.23	25.23	2.09E−1	4.02E+0
6	5753	4.25	4.25	1.53E−5	8.27E+0
7	11706	8.17	8.17	2.47E−5	1.11E+1
8	12103	8.67	8.67	1.17E−5	1.22E+1
9	23729	10.08	10.08	3.76E−5	1.01E+1
10	...	...	...	...	...

· · · method failed due to zero or negative pivot in Cholesky decomposition phase.

TABLE 4
Givens transformations.

Problem	Max. store	Problem time	Total time	Rel. error	Residual
1	6821	2.86	3.49	1.71E−5	3.62E+0
2	10189	15.53	17.51	1.00E−4	7.53E−1
3	10181	15.66	17.62	7.63E−3	7.53E−1
4	23233	40.32	43.91	9.06E−5	1.28E+0
5	23105	39.88	43.36	2.48E−3	1.29E+0
6	5753	9.06	10.50	8.04E−5	8.27E+0
7	11706	31.71	34.30	6.45E−5	1.11E+1
8	12103	13.48	16.57	2.98E−5	1.22E+1
9	23729	13.50	16.39	1.53E−5	1.01E+1
10	12662	26.04	27.75	3.50E−4	2.35E+1

TABLE 5
Peters–Wilkinson method (using the LU-decomposition of A).

Problem	Max. store	Problem time	Total time	Rel. error	Residual
1	9594	3.55	4.63	3.33E−4	3.62E+0
2	34519	31.58	35.01	1.63E−2	3.08E+0
3	...	...	...	...	...
4	66329	99.59	105.96	1.82E−2	7.35E+0
5	65932	97.63	104.05	2.15E−1	6.70E+0
6	20404	5.20	7.66	2.99E−5	8.27E+0
7	37564	9.44	13.97	3.62E−5	1.11E+1
8	42786	19.13	23.95	2.70E−4	1.22E+1
9	45228	30.14	35.25	5.48E−4	1.01E+1
10	...	...	...	...	...

TABLE 6
Peters–Wilkinson method (using the LU-decomposition of A^T).

Problem	Max. store	Problem time	Total time	Rel. error	Residual
1	9669	3.59	5.54	3.38E−4	3.62E+0
2	37564	50.59	57.34	2.67E−3	8.04E−1
3	35939	41.31	47.66	9.74E−2	1.19E+0
4	73011	190.08	203.27	3.39E−4	1.34E+0
5	74950	217.58	231.16	6.63E−2	1.70E+0
6	20424	4.98	8.88	3.78E−5	8.27E+0
7	37584	9.18	16.08	5.77E−5	1.11E+1
8	42836	16.59	24.22	1.04E−4	1.22E+1
9	45253	23.06	32.43	5.97E−3	1.01E+1
10	31523	42.96	48.34	1.57E−3	2.35E+1

5. Concluding remarks. Following are some observations about the experiments, some of which we include for interest even though we did not regard them as important enough to illustrate via tables.

(1) The storage requirements for the normal equations and the Givens algorithm are the same. This is not surprising since the basic steps in these two algorithms are the same. Recall that the Givens rotations are not saved.

(2) The storage requirements for the method of normal equations and the Givens algorithm are better than the P–W scheme. The minimal storage required to execute successfully the P–W algorithm can be 2 to 3 times that required for the other two methods.

(3) The method of normal equations executes faster than the other two methods.

(4) Whether the P–W algorithm or the Givens algorithm is faster seems to be problem-dependent.

(5) In the P–W algorithm, the most expensive phases are the computation of the LU-decomposition of A and the formation of L^TL. The storage requirement and the execution time are large in the initial LU-decomposition phase. On the other hand, the most expensive phase in the Givens algorithm is the orthogonal reduction phase.

(6) The normal-equations and P–W methods are comparable in accuracy to the Givens algorithm for the well-conditioned problems. Both fail or give poor results on the three ill-conditioned problems. In order for the P–W algorithm to produce acceptable results for ill-conditioned problems, a pivoting strategy which yields a well-conditioned L must be used.

(7) It was interesting to note that the storage required to determine the nonzero structure and store the nonzeros of the Cholesky factors of the normal equations in (1.3) and (2.5) are very close. This suggests that A^TA and L^TL may have similar structures.

The above observations suggest that for well-conditioned problems the method of normal equations should be seriously considered because of its small storage requirement and execution time. However, since there exist problems for which the normal equations will fail due to numerical difficulties, the Givens algorithm should be used in those situations if the available storage is restricted and time is not a major issue.

Note that it is assumed in the method of normal equations and the Givens algorithm that if the matrix A is sparse, then the matrix product A^TA is also sparse. If it is not the case, these two methods are likely to be inefficient. However, in most of the latter cases, this phenomenon is due to the presence of a few relatively dense rows in A. In [6], George and Heath have proposed a way of circumventing this problem. Those dense rows are temporarily discarded, yielding a problem for which A^TA is sparse. After the solution x to the modified least squares problem is determined, the solution to the original problem can be obtained by modifying x using those discarded rows (see [6, § 5] for details). The presence of a few dense rows causes similar difficulties for the P–W algorithm, and an updating method for that case is given by Björck and Duff [2].

REFERENCES

[1] J. H. ARGYRIS AND O. E. BRÖNLUND, *The natural factor formulation of the stiffness matrix displacement method*, Computer Methods Appl. Mech. Engrg., 5 (1975), pp. 97–119.

[2] A. BJÖRCK AND I. S. DUFF, *A direct method for the solution of sparse linear least squares problems*, Linear Algebra Appl., 34 (1980), pp. 43–67.

[3] I. S. DUFF AND J. K. REID, *A comparison of some methods for the solution of sparse overdetermined systems of linear equations*, J. Inst. Math. Appl., 17 (1976), pp. 267–280.
[4] I. S. DUFF, *MA28—A set of FORTRAN subroutines for sparse unsymmetric linear equations*, Rep. R-8730, AERE, Harwell, England, July 1977.
[5] A. GEORGE, J. W-H. LIU AND E. NG, *User's Guide for SPARSPAK: Waterloo sparse linear equations package*, Res. Rep. CS-78-30 (revised), Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1980.
[6] A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69–83.
[7] A. GEORGE AND J. W-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[8] G. D. HACHTEL, *Extended application of the sparse tableau approach—Finite elements and least squares*, Tech. Rep., Electrical Science and Engineering Dept., University of California–Los Angeles, 1974.
[9] G. B. KOLATA, *Geodesy: Dealing with an enormous computer task*, Science, 200 (1978), pp. 421–422.
[10] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
[11] H. M. MARKOWITZ, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.
[12] C. C. PAIGE AND M. A. SAUNDERS, *A bidiagonalization algorithm for sparse linear equations and least-squares problems*, Tech. Rep. SOL 78-19, Dept. Operations Research, Stanford University, Stanford, CA, 1978.
[13] G. PETERS AND J. H. WILKINSON, *The least squares problem and pseudo-inverses*, Comput. J., 13 (1970), pp. 309–316.
[14] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0004 $01.25/0

AN INITIALIZATION PROGRAM FOR SEPARABLY STIFF SYSTEMS*

DAVID S. WATKINS†

Abstract. Initialization of a separably stiff system of ordinary differential equations consists of readjusting certain initial values, while holding the others fixed, in such a way that the solution of the system does not have an initial transient. This paper discusses the features of a FORTRAN program which solves the initialization problem. The algorithm used is simpler and faster than those used previously. Numerical results are given for a simple test problem and more complex problems which arise in the physics of the upper ionosphere and the solar wind.

Key words. ODE, stiff, initialization, invariant subspace

1. Introduction. A stiff system [15] of ordinary differential equations is said to be *separably stiff* if the eigenvalues of the Jacobian matrix can be partitioned into two well-separated sets, one consisting of large or *stiff* eigenvalues, the other consisting of (comparatively) small eigenvalues. In a stiff system all large eigenvalues have large negative real part. Throughout this paper the adjective *large*, when referring to an eigenvalue, will mean "having large negative real part". The *initialization problem* is to readjust some of the initial values, while holding the others fixed, in such a way that the solution does not have an initial transient. In [22] the author has briefly described a class of ionospheric physics problems in which the initialization problem arises. In general, initialization is important in problems for which some of the initial values are not known with confidence, but where it is known on physical grounds that the correct solution does not have an initial transient.

In this paper we discuss the features of a FORTRAN program, written by the author, which will solve a large class of initialization problems. The program embodies a number of improvements in speed, reliability and generality compared with the procedures described in the author's previous papers [22], [23]. The program is available from the author.

Previous literature on the initialization problem includes [22], [23], [7], and [1]. Kreiss [7] considered the initialization problem for systems having purely imaginary eigenvalues for which the solutions oscillate on two different time scales. The problem was to suppress the rapidly oscillating components associated with large, imaginary eigenvalues. The program discussed in this paper could be modified to handle problems of this type. Alfeld [1] showed how to obtain a transient-free solution but did not consider holding some of the initial values fixed.

2. The linear case. We will first show how to initialize the linear initial value problem

$$y' = b + cx + Ay, \qquad y(x_0) = w. \tag{1}$$

Suppose A has k stiff eigenvalues and $n-k$ small eigenvalues. In typical applications $k \ll n$, although this is not necessary. There exists a (nonunique) orthogonal transformation Q such that

$$Q^T A Q = B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix} \tag{2}$$

* Received by the editors November 16, 1981, and in revised form March 29, 1982. This work was supported in part by the National Science Foundation under grant MCS-8102382.

† Department of Pure and Applied Mathematics, Washington State University, Pullman, Washington 99164.

where B_{22} is $k \times k$ and has stiff eigenvalues, and B_{11} has small eigenvalues. Section 3 explains how this transformation is carried out in practice. For now it suffices to note that the last k columns of Q must be an orthonormal basis for the *dominant space* of A^T, the invariant subspace of A^T associated with the stiff eigenvalues. Letting $y = Q\hat{y}$, $w = Q\hat{w}$, $b = Q\hat{b}$, etc., (1) can be transformed to

$$\hat{y}' = \hat{b} + \hat{c}x + B\hat{y}, \qquad \hat{y}(x_0) = \hat{w}, \tag{3}$$

or, in partitioned form,

$$\hat{y}_1' = \hat{b}_1 + \hat{c}_1 x + B_{11}\hat{y}_1 + B_{12}\hat{y}_2, \qquad \hat{y}_1(x_0) = \hat{w}_1, \tag{4a}$$

$$\hat{y}_2' = \hat{b}_2 + \hat{c}_2 x + \quad B_{22}\hat{y}_2, \qquad \hat{y}_2(x_0) = \hat{w}_2. \tag{4b}$$

The general solution of (4b) is

$$\hat{y}_2 = \exp(xB_{22})v - B_{22}^{-1}(B_{22}^{-1}\hat{c}_2 + \hat{b}_2 + \hat{c}_2 x),$$

where v is an arbitrary constant vector. Since all eigenvalues of B_{22} are stiff, $\hat{y}_2$ will have an initial transient if (and only if) $v \neq 0$. Thus, the unique value of $\hat{w}_2$ which gives a transient-free solution is that for which $v = 0$, namely

$$\hat{w}_2 = -B_{22}^{-1}(B_{22}^{-1}\hat{c}_2 + \hat{b}_2 + \hat{c}_2 x_0). \tag{5}$$

If this value of $\hat{w}_2$ is chosen, then (4a) will be transient-free, regardless of the choice of $\hat{w}_1$, since B_{11} has no stiff eigenvalues. Thus there is an $(n-k)$-dimensional manifold of initial conditions $w = Q\hat{w}$ for which the solution is transient-free, namely the set of w for which $\hat{w}_2$ satisfies (5). To specify a unique transient-free solution, $n-k$ conditions must be placed on w. There are numerous ways one might do this, depending on the requirements of the particular problem. The needs of our problems dictated that $n-k$ components of w be held fixed. Thus our code will hold any $n-k$ components of w fixed and adjust the others.

To see how this can be done, consider first the homogeneous case

$$y' = Ay, \qquad y(x_0) = z. \tag{6}$$

The choices of z which lead to a transient-free solution are those which lie in the *codominant space* of A, the invariant subspace of A associated with the small eigenvalues. It is easy to show that this space is the orthogonal complement of the dominant space of A^T. Recalling that the latter is spanned by the last k columns of Q, and writing $Q = [Q_1 Q_2]$, the transient-free choices of z are just those for which $Q_2^T z = 0$.

For notational convenience suppose that the components of z which are to be held fixed are the first $n-k$. Let $z = [z_1^T z_2^T]^T$, where z_1 consists of the components which are to be held fixed. Making the further partition $Q_2 = [Q_{12}^T Q_{22}^T]^T$, the condition $Q_2^T z = 0$ can be rewritten as

$$Q_{22}^T z_2 = -Q_{12}^T z_1. \tag{7}$$

For any choice of z_1, z_2 can be determined uniquely from (7), provided that Q_{22} is nonsingular.

The nonhomogeneous case can be reduced to the homogeneous case by subtraction of an arbitrary transient-free solution from the nonhomogeneous problem. Specifically, let $w = [w_1^T w_2^T]^T$, where w_1 is to be held fixed and w_2 is to be adjusted. Let $\hat{w}_2$ be given by (5), and let $s = Q_2\hat{w}_2$. Then $y(x_0) = s$ gives a transient-free solution to the nonhomogeneous problem (the one given by $w_1 = 0$). Let $z = w - s$. The components of z_1 are determined by $z_1 = w_1 - s_1$, where s has been partitioned

conformally with z and w. For this fixed z_1, z_2 may be adjusted so that the homogeneous problem (6) has a transient-free solution. Then w_2 $(=z_2+s_2)$ gives a transient-free solution of the nonhomogeneous problem. To get an explicit formula for w_2, note that $w=z+s$, whence $Q_2^Tw=Q_2^Tz+Q_2^Ts=0+\hat{w}_2$. Thus

$$Q_{22}^Tw_2=-Q_{12}^Tw_1+\hat{w}_2, \tag{8}$$

and we can solve uniquely for w_2 if Q_{22} is nonsingular. The steps in the calculation of w_2 can be summarized as follows:

1. Determine an orthonormal basis of the dominant space of A^T. (Let Q_2 denote the $n\times k$ matrix whose columns are the basis.)
2. Compute $B_{22}=Q_2^TAQ_2$.
3. Compute $\hat{w}_2$ using (5).
4. Solve for w_2 using (8).

Step 4 is different from the corresponding step in the procedures outlined in [22] and [23]. The new approach, which may be regarded as the dual of the older procedures, has a significant advantage. The earlier procedures require the full transforming matrix Q, whereas the present one uses only the last k columns Q_2. If $k\ll n$, this allows substantial savings of space and time which will certainly prove decisive for large problems.

For future reference we combine (8) and (5) into a single equation:

$$w_2=-(Q_{22}^T)^{-1}(Q_{12}^Tw_1+B_{22}^{-1}(B_{22}^{-1}Q_2^Tc+Q_2^Tb+Q_2^Tcx_0)). \tag{9}$$

The main cost of the present procedure is that of calculating the dominant space. This will be discussed in the following section.

The procedure requires the solution of three systems of linear equations involving two $k\times k$ coefficient matrices in Steps 3 and 4. Our program uses standard LINPACK routines [3] to solve these systems.

3. Calculation of the dominant space. The program has two ways of calculating the dominant space of A^T: simultaneous iteration [24], which is fast if $k\ll n$, and the QR algorithm [24], [17], which is slow but robust. The simultaneous iteration option uses the basic orthogonal variant (Wilkinson [24, p. 607]): Starting with k orthonormal vectors, $V=[v_1\cdots v_k]$, the algorithm calculates $W=A^TV$, then performs a QR decomposition to orthonormalize the columns of W. These orthonormalized columns are taken as a new V, and the process is repeated. The span of the columns of V converges to the dominant space of A^T. The rate of (linear) convergence is $|\lambda_{k+1}/\lambda_k|$, where λ_k and λ_{k+1} are the kth and $(k+1)$st largest eigenvalues of A. Therefore convergence is fast if the k stiff eigenvalues are well separated from the small ones. More sophisticated versions of simultaneous iteration have been developed [21], [9], [18]. We have not used any of these algorithms because they do not improve the accuracy of the invariant subspace. They do improve the accuracy of individual eigenvalues and eigenvectors, but these are not needed by the algorithm.

If the simultaneous iteration option is used, the user must specify k in advance. However, the routine does have a limited ability to adjust the value of k in the event that convergence is too slow. Estimates of $|\lambda_1|$ and $|\lambda_{i+1}/\lambda_i|$, $i=1,\cdots,k$, can easily be obtained from simultaneous iteration with k vectors. If $|\lambda_{k+1}/\lambda_k|$ is unsatisfactory and $|\lambda_k/\lambda_{k-1}|$ or $|\lambda_{k-1}/\lambda_{k-2}|$ is substantially better, then k is reduced by one or two. Otherwise, if $|\lambda_{k+1}|$ is judged to be large, k is increased by 1. (The question of what constitutes a large eigenvalue is addressed below.) The user must specify minimum and maximum permissible values of k. The program will not make a change in k

which violates these bounds. If k is increased, the number of initial values to be adjusted must also be increased. A *confidence hierarchy*, supplied by the user, is used to decide which values are to be adjusted. Those k values in which the user has least confidence are adjusted.

Simultaneous iteration cannot distinguish between large negative and large positive eigenvalues. The presence of an eigenvalue with a large positive real part would indicate that the system is unstable rather than stiff. Applying the initialization procedure to such a system would have an unintended effect—rapidly growing components of the solution would be suppressed, whereas the intention is to suppress rapidly decaying components. Therefore, after the calculation of B_{22}, the code performs a crude test to detect positive eigenvalues. Since the transformed matrix tends toward (block) triangular form, the main diagonal entries of B_{22} should roughly approximate the stiff eigenvalues. In order to take into account complex eigenvalues, the program actually examines the 2×2 blocks along the main diagonal. If any of the eigenvalues of these blocks has a positive real part, a warning message is produced, stating that the system is unstable, not stiff. While this test is not foolproof, it should detect large positive eigenvalues in the vast majority of cases.

If the *QR* algorithm option is used, the user does not specify k in advance. The *QR* algorithm determines how many stiff eigenvalues there are and whether or not the system is separably stiff, based on gaps in the spectrum and the length of the interval of integration specified by the user. Two positive numbers $m < M$ are used to specify whether the system is separably stiff. Any eigenvalue satisfying $\max(|\mathrm{Re}\,(\lambda)|, |\mathrm{Im}\,(\lambda)|) < m$ is considered small, while an eigenvalue with $\mathrm{Re}\,(\lambda) < -M$ is considered stiff. Values which seem to work well are $m = 10/L$ and $M = 100/L$, where L is the stated length of the interval of integration. If the matrix fails this first test but has stiff eigenvalues, a second test is performed. The intermediate and stiff eigenvalues are searched for gaps such that $-\mathrm{Re}\,(\lambda)/|\mathrm{Re}\,(\mu)| > M/m$ for all λ on the "stiff" side of the gap and all μ on the "small" side. If such gaps exist, the system is considered separably stiff and the eigenvalues are partitioned at the first such gap from the origin. Once the value of k has been determined, the confidence hierarchy is used to decide which initial values will be adjusted.

The *QR* routine differs from standard *QR* programs in several details. As usual, the matrix is initially reduced to upper Hessenberg form, but this is preceded by a symmetric row and column interchange which moves the most negative main diagonal element to the bottom right corner of the matrix. That entry then remains undisturbed as the Hessenberg reduction is done by rows from bottom to top. Since the shifts for the *QR* algorithm are taken to be eigenvalues of the trailing 2×2 principal submatrix, the large negative element in the bottom corner makes it likely that large negative shifts will be chosen, and that, consequently, the stiff eigenvalues will emerge first. As each eigenvalue emerges, the 1-norm of the remaining deflated matrix is calculated. If that norm is less than m, the *QR* algorithm is terminated, as all remaining eigenvalues are small. The exact shifting strategy is to take a double implicit *QR* step if the eigenvalues of the trailing 2×2 matrix are complex and a single implicit step if they are real. In the latter case the shift is taken to be that eigenvalue which is closer to the (n, n) entry of the matrix. The single steps are clearly superior in some situations, for example, if $k = 1$. An eigenvalue swapping facility in the spirit of Stewart [19] is built in for use in the unlikely event that the large and small eigenvalues are intermixed.

The costs of the *QR* option are as follows. The initial Hessenberg reduction costs $\frac{5}{3}n^3$ multiplications, plus the cost ($\frac{2}{3}n^3$) of updating the transforming matrix. The *QR* steps are implemented in fast, scaled plane rotations [4], [5], [8], so the cost of each

single QR step is only $2n^2$. Updating the transforming matrix costs another $2n^2$. Still, even if only one step were needed for each eigenvalue, the total cost of the QR portion of the routine would be $\frac{5}{3}n^3$. A more realistic figure is three times that. Of course much is saved if $k \ll n$ and the routine is able to stop early. Unfortunately, stopping early does not decrease the cost of the Hessenberg reduction.

If k is known and $k \ll n$, the simultaneous iteration option is economical. In this case the costs of orthonormalization and testing for convergence are negligible. The cost of each power iteration is n^2k multiplications. The number of iterations is reasonable if the small eigenvalues are much smaller than the stiff ones. The cost of calculating B_{22} is $nk(n-k)$ multiplications. (B_{22} is produced automatically by the QR option.)

4. Sensitivity of the initialization problem. In § 2 we saw that w_2 can be calculated from (8) provided that Q_{22} is nonsingular. If Q_{22} is singular, the initialization problem does not have a unique solution: For most choices of w_1 there is no choice of w_2 for which the solution has no transient, while for the remaining choices of w_1 there are infinitely many w_2 for which the solution is transient-free. It is extremely unlikely that Q_{22} will be exactly singular. What is not quite so unlikely is that Q_{22} may be ill-conditioned, in which case the computation will be inaccurate.

It is clear from (9) that the condition number of Q_{22}, $\kappa(Q_{22})$, is a condition number for the initialization problem. (The likelihood that B_{22} is ill-conditioned is small because B_{22} has no small eigenvalues.) Actually we prefer to use $\|Q_{22}^{-1}\|$ instead of $\kappa(Q_{22})$, as the former also gives an indication of the rate of convergence in the nonlinear case (discussed below). In the spectral norm it is always true that $\|Q_{22}^{-1}\| \geqq \kappa(Q_{22})$, because $\|Q_{22}\| \leqq 1$.

The program includes an option to calculate $\|Q_{22}^{-1}\|$. When $k = 1$ the estimate is free, since Q_{22} is then 1×1. If $k > 1$ the LINPACK routine DGECO supplies, at small cost, an estimate of $\kappa(Q_{22})$ [2]. We have modified the routine slightly so that it returns an estimate of $\|Q_{22}^{-1}\|$ instead of $\kappa(Q_{22})$.

In [23] we showed that $\|Q_{11}^{-1}\|$ is a condition number for the initialization problem, where

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}.$$

By [20, Thm. A.1, p. 657], Q_{11} and Q_{22} have the same smallest singular value. Therefore, in the spectral norm, $\|Q_{22}^{-1}\| = \|Q_{11}^{-1}\|$.

5. The nonlinear case. The nonlinear initial value problem

$$y' = f(x, y), \qquad y(x_0) = w \tag{10}$$

can be initialized by iteration on a sequence of linear problems. Let $w^{(0)} = [w_1^T w_2^{(0)T}]^T$ be a starting guess, and linearize the problem about $w^{(0)}$:

$$\begin{aligned} y' = f(x_0, w^{(0)}) &+ \frac{\partial f}{\partial x}(x_0, w^{(0)})(x - x_0) \\ &+ \frac{\partial f}{\partial y}(x_0, w^{(0)})(y - w^{(0)}), \qquad y(x_0) = w^{(0)}. \end{aligned} \tag{11}$$

This system has the form $y' = b + cx + Ay$, where $A = \partial f/\partial y(x_0, w^{(0)})$, $c = \partial f/\partial x(x_0, w^{(0)})$, and $b = f(x_0, w^{(0)}) - cx_0 - Aw^{(0)}$. It can be initialized by the method of § 3 to produce $w^{(1)} = [w_1^T w_2^{(1)T}]^T$ such that the solution to the linearized problem does not have an

initial transient. Now the problem can be relinearized about $w^{(1)}$, the process can be repeated to get $w^{(2)}$, and so on. For the range of problems which we have considered, the sequence $(w^{(m)})$ nearly always converges to a w for which the solution to (10) is transient-free. In [22] and [23] we discussed the convergence of the method. We found that convergence is aided by stiffness and lack of severe nonlinearity and impeded by severe nonlinearity and ill-conditioning of the initialization problem. These conclusions need not be justified anew here, but it is at least worth mentioning that (9) is better suited for the analysis of convergence than any of the equations in [22] or [23].

At each step the program has to compute a new Jacobian, then find the associated dominant invariant subspace. If the simultaneous iteration method is used, the final vectors from the previous step are used as starting vectors for the power iterations in the current step. Since the Jacobian changes little as convergence approaches, fewer iterations are required in the later steps. The user has the option of specifying that the *QR* method be used on the first step and that simultaneous iteration be used on subsequent steps.

People who work with stiff systems are conditioned to the idea of working with an old Jacobian, not updating it except when absolutely necessary. This conditioning, together with the resemblance of the procedure to Newton's method, suggests that the iterations could be done without updating the Jacobian on each step. The savings would be large. Unfortunately, the procedure is not Newton's method—the Jacobian is not used in the same manner here. If the Jacobian is not updated regularly, the outcome will be worse than convergence failure—the iterations will converge to the solution of the wrong problem. Therefore, in order to get an accurate solution, the Jacobian must be kept up-to-date. While our program updates the Jacobian on every iteration, it might be enough to update it every second or third iteration. Then the iterations would be less expensive on the average, but more iterations would be required. We have not studied this question systematically.

While the Jacobian must be kept fresh, we have found that the program works well with Jacobians evaluated by numerical differentiation, in spite of the fact that numerical Jacobians are notoriously inaccurate. The good performance of the algorithm under these conditions is probably due to the fact that the dominant space, which is what matters, is generally insensitive to perturbations of the Jacobian. By [16] an invariant subspace S of A will tend to be stable whenever the spectrum of $A|_S$ is well separated from the rest of the spectrum of A. This condition is certainly satisfied in the case of a separably stiff system.

The iterations evidently converge linearly. This suggests that some relaxation technique or Aitken acceleration could profitably be used to accelerate convergence. This is another area which warrants study.

6. Numerical results. The program has been tested on numerous small linear problems of the form $y' = b + cx + Ay$, with various numbers of stiff eigenvalues, with the expected results. In all cases the program found the transient-free solution in one iteration. The Jacobian matrices for these problems were generated by applying random orthogonal similarity transformations to (block triangular) matrices with known eigenvalues. A user with problems of the special form $y' = b + cx + Ay$ has the option of having the program halt after one iteration without testing for convergence.

Example 1. Consider the simple nonlinear problem

$$y_1' = k_3 y_3 y_4 - k_1 y_1 y_2, \qquad y_1(0) = 1,$$

$$y_2' = k_4 y_4 y_1 - k_2 y_2 y_3, \qquad y_2(0) = 1,$$

$$y_3' = k_1 y_1 y_2 - k_3 y_3 y_4, \qquad y_3(0) = 1,$$

$$y_4' = k_2 y_2 y_3 - k_4 y_4 y_1, \qquad y_4(0) = 1,$$

where $k_1 = 1.0$, $k_2 = 10^4$, $k_3 = 2.0$, $k_4 = 10^5$, and the interval of integration is [0, 1]. Use of the *QR* option reveals that the system has one stiff eigenvalue $\lambda_1 = -1.100 \times 10^5$. On deflation the remaining matrix has norm less than 10 ($= m$), so the remaining eigenvalues are not calculated. If y_1, y_3, and y_4 are held fixed while y_2 is adjusted, the successive iterates are as shown in Table 1. The *QR* and simultaneous iteration options give identical results, as expected. The eigenvalues on the final iteration do not differ much from those on the first iteration. They are $\lambda_1 = -1.100 \times 10^5$, $\lambda_2 = -17.46$, $\lambda_3 \cong 10^{-11}$, $\lambda_4 \cong 10^{-11}$. The condition number is 17.4.

TABLE 1
Example 1.

Iteration	y_2
0	1.0
1	10.00080012366
2	10.00145509914
3	10.00145514681
4	10.00145514681

Example 2. This is a system of five differential equations describing steady state density, velocity, temperature, temperature anisotropy, and heat flow of electrons in the solar wind, as a function of distance from the sun. The system is the object of a study currently being conducted by R. W. Schunk and the author. Unfortunately, the equations are too complicated to be listed here. The reader is referred to [10], [11], [12], [13], and [14] for more information about this example and Example 3.

The system has one stiff eigenvalue. The initial temperature anisotropy is not known with confidence, so it is allowed to vary. The successive iterates are as shown in Table 2. On the final iteration the eigenvalues are -1.706×10^{-10}, -4.044×10^{-12}, -3.432×10^{-12}, 5.641×10^{-14}, and 1.238×10^{-12}. The units of the eigenvalues are cm^{-1}. The length of the interval of integration is about 10^{13} cm. Therefore the stiffest eigenvalue gives rise to a transient whose length (roughly 10^{10} cm) is approximately one-thousandth that of the interval of integration. The condition number for the problem is 180. In this and the next example the Jacobians were evaluated numerically.

Example 3. This is a system of eight equations describing density, velocity, temperature, temperature anisotropy, and heat flow of electrons and protons in the

TABLE 2
Example 2. *Solar wind problem.*

Iteration	Temperature anisotropy
0	1.0
1	100306.4158
2	94697.6769
3	94774.2620
4	94773.0612
5	94773.0800
6	94773.0797

topside ionosphere [12]. There are eight equations, not ten, because electron and proton densities are equal, as are the velocities. The electron and proton temperature anisotropies are allowed to vary. The results are given in Table 3. The eigenvalues are $\lambda_1 = -6.067 \times 10^{-3}$, $\lambda_2 = -5.001 \times 10^{-5}$, $\lambda_3 = -1.612 \times 10^{-6}$, $\lambda_4 = -7.164 \times 10^{-7}$, $|\lambda_i| < 2 \times 10^{-8}$, $i = 5, \cdots, 8$, so it would be legitimate to take either $k = 1$ or $k = 2$. The length of the interval of integration is 10^9 cm. The condition number estimate is 9×10^5, which is high but not intolerable on the sixteen-digit computer (Amdahl 470 V/8, double precision) which we used.

TABLE 3
Example 3. *Topside ionosphere problem.*

Iteration	Temperature anisotropy	
	Electrons	Protons
0	−700	−1300
1	−697.30957084	−1282.4621540
2	−697.31041331	−1282.4027314
3	−697.31041310	−1282.4027607
4	−697.31041310	−1282.4027614

It is important that the transient-free solution be determined very accurately, simply because the code will be used as a module in much larger programs which depend on accurate output from their subroutines. Another reason for insisting on an accurate solution can be illustrated by comparing the stepsize which can be taken by an automatic ODE solver on a carefully initialized problem against that which can be taken if the problem has not been initialized accurately. We integrated the system of Example 3 twice, once using the accurate temperature anisotropy values $-697.310\cdots$ and $-1282.402\cdots$ and one using the inaccurate values -700.0 and -1300.0. Note that the inaccurate values are off by only about one percent. We used the ODE solver LSODE [6], which automatically chooses the (initial and subsequent) stepsizes. With the accurate initial values the initial stepsize was 3×10^6 cm, whereas with the inaccurate values the stepsize was 70 cm. Only after about 25 steps did the stepsize reach 10^6 cm. Thus even a slight inaccuracy in the initial values leads to a significant (and potentially costly) transient.

7. Conclusions. We have described an initialization program which calculates the dominant space of a stiff system in order to determine initial conditions which give rise to a transient-free solution. The program uses either simultaneous iteration or the *QR* algorithm to calculate the dominant space. The strength of the *QR* option is that it automatically decides whether or not the problem is separably stiff and determines the number of stiff eigenvalues. On the other hand, the simultaneous iteration option is faster if the dimension of the dominant space is small. If the stiff system is linear with constant coefficients, the initialization is done by a direct procedure, whereas for all other types of problems iteration must be employed. The code optionally estimates a condition number which provides a measure of the sensitivity of the initialization problem. We have given some sample numerical results illustrating the performance of the algorithm. The sample problems include systems which model phenomena in the topside ionosphere and the solar wind.

REFERENCES

[1] P. ALFELD, *A method of skipping the transient phase in the solution of separably stiff initial value problems*, Math. Comp., 35 (1980), pp. 1173–1176.

[2] 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.

[3] J. J. DONGARRA et al., *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[4] W. M. GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329–336.

[5] S. HAMMARLING, *A note on modifications to the Givens plane rotation*, J. Inst. Math. Appl., 13 (1974), pp. 215–218.

[6] A. C. HINDMARSH, *LSODE and LSODI, two new initial value ordinary differential equation solvers*, SIGNUM Newsletter, 15 (1980), pp. 10–11.

[7] H.-O. KREISS, *Problems with different time scales for ordinary differential equations*, SIAM J. Numer. Anal., 16 (1979), pp. 980–998.

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

[9] H. RUTISHAUSER, *Simultaneous iteration method for symmetric matrices*, Numer. Math., 16 (1970), pp. 205–223. = Contribution II/9 of Wilkinson and Reinsch [25].

[10] R. W. SCHUNK, *Transport equations for aeronomy*, Planet. Space Sci., 23 (1975), pp. 437–485.

[11] ———, *Mathematical structure of transport equations for multispecies flows*, Rev. Geophys. Space Phys., 15 (1977), pp. 429–445.

[12] R. W. SCHUNK AND D. S. WATKINS, *Comparison of solutions to the 13-moment and standard transport equations for low speed thermal proton flows*, Planet. Space Sci., 27 (1979), pp. 433–444.

[13] ———, *Electron temperature anisotropy in the polar wind*, J. Geophys. Res., 86 (1981), pp. 91–102.

[14] ———, *Proton temperature anisotropy in the polar wind*, J. Geophys. Res., 87 (1982), pp. 171–180.

[15] L. F. SHAMPINE AND C. W. GEAR, *A user's view of solving stiff ordinary differential equations*, SIAM Rev., 21 (1979), pp. 1–17.

[16] G. W. STEWART, *Error bounds for approximate invariant subspaces of closed linear operators*, SIAM J. Numer. Anal., 8 (1971), pp. 796–808.

[17] ———, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[18] ———, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numer. Math., 25 (1976), pp. 123–136.

[19] ———, *HQR3 and EXCHNG: FORTRAN subroutines for calculating and ordering the eigenvalues of a real, upper Hessenberg matrix*, ACM Trans. Math. Software, 2 (1976), pp. 275–280.

[20] ———, *On the perturbation of pseudo-inverses, projections, and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634–662.

[21] W. J. STEWART AND A. JENNINGS, *A simultaneous iteration algorithm for real matrices*, ACM Trans. Math. Software, 7 (1981), pp. 184–198.

[22] D. S. WATKINS, *Determining initial values for stiff systems of ordinary differential equations*, SIAM J. Numer. Anal., 18 (1981), pp. 13–20.

[23] ———, *Efficient initialization of stiff systems with one unknown initial condition*, SIAM J. Numer. Anal., 18 (1981), pp. 794–800.

[24] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, London and New York, 1965.

[25] J. H. WILKINSON AND C. REINSCH, *Handbook for Automatic Computation, Vol. II, Linear Algebra*, Springer-Verlag, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0005 $01.25/0

A LANCZOS ALGORITHM FOR COMPUTING SINGULAR VALUES AND VECTORS OF LARGE MATRICES*

JANE CULLUM†, RALPH A. WILLOUGHBY† AND MARK LAKE†

Abstract. Any real matrix A has associated with it the real symmetric matrix

$$B \equiv \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$$

whose positive eigenvalues are the nonzero singular values of A. Using B and our Lanczos algorithms for computing eigenvalues and eigenvectors of very large real symmetric matrices, we obtain an algorithm for computing singular values and singular vectors of large sparse real matrices. This algorithm provides a means for computing the largest and the smallest or even all of the distinct singular values of many matrices.

Key words. singular value decomposition, Lanczos bidiagonalization algorithm, large matrix

1. Introduction. The singular value decomposition of a real $l \times n$ matrix A is the natural generalization of the corresponding eigenvector decomposition one obtains for real symmetric matrices. The singular value decomposition of A has the form

$$A = X \Sigma Y^T \tag{1.1}$$

where X and Y are orthogonal matrices, X is $l \times l$, Y is $n \times n$, and Σ is an $l \times n$ rectangular diagonal matrix with nonnegative diagonal entries. The columns of X and of Y are called respectively the left and the right singular vectors of A. See e.g. Stewart (1973, Chap. 6). Without loss of generality in the discussion we assume that $l \geqq n$. In this case

$$\Sigma = \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} \tag{1.2}$$

where Σ_1 is an $n \times n$ diagonal matrix with entries $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ called the singular values of A. We have the following well-known lemma.

LEMMA 1.1. *Let $A = X \Sigma Y^T$ be a singular value decomposition of the $l \times n$ matrix A with $l \geqq n$. Let $X = (X_1, X_2)$ where X_1 is $l \times n$ and X_2 is $l \times (l-n)$. Then*

$$AY = X_1 \Sigma_1, \quad A^T X_1 = Y \Sigma_1 \quad \text{and} \quad A^T X_2 = 0. \tag{1.3}$$

If A has rank $r < n$, let $Y = (Y_1, Y_2)$ where Y_1 is $n \times r$ and Y_2 is $n \times (n-r)$. Then $AY_2 = 0$. In fact, singular vectors $x \in X_1$ and $y \in Y$ corresponding to any singular value σ_j, including any zero singular values, are characterized by the conditions:

$$Ay = \sigma_j x, \quad A^T x = \sigma_j y, \quad x \neq 0, \quad y \neq 0. \tag{1.4}$$

The remaining left singular vectors X_2 can be any orthonormal set of $(l-n)$ vectors that is orthogonal to X_1.

Proof. The proof follows directly from (1.1) and (1.2) and then (1.3). □

Singular values and vectors are useful tools in many applications, providing, for example, measures of the sensitivity of solutions of given problems to errors in the information provided (see e.g. Lawson and Hanson (1974)), ways of stabilizing computational methods for ill-posed problems (see e.g. Hanson and Phillips (1975)), and data reduction schemes useful for example in pattern recognition (see e.g. Andrews and Hunt (1977)).

* Received by the editors May 7, 1980, and in revised form May 21, 1982.
† IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 10598.

Reliable programs for computing singular values and vectors of matrices of moderate size are available in the LINPACK (1979) and EISPACK (1976) Fortran libraries. These procedures are based upon the algorithm given in Golub and Kahan (1965) which explicitly transforms the given matrix A into a bidiagonal matrix. The amount of computer storage required is proportional to the product ln of the dimensions of A. The corresponding arithmetic operations count is proportional to $[\max(l, n)][\min(l, n)]^2$. In addition, these programs compute all of the singular values. Thus they are not very practical for large matrices.

The Lanczos singular value procedure which we describe below can be very efficient for large matrices and can be used to compute a few or many of the singular values in any portion of the spectrum. For this to be true it must be possible for the matrix-vector products $A^T u$ and Av to be computed with computer storage requirements and arithmetic operation counts which are linear in $l = \max(l, n)$. This is true for example for sparse A. For the computation of q distinct singular values of A, the computer storage required by the algorithm is then linear in l, and the total operation count is linear in ql where $q \leqq n = \min(l, n)$. In addition to the above, the efficiency and in fact the practicality of this algorithm depend upon the distribution of the singular values, that is, upon how clustered they are, and upon which subsets of the singular values are to be computed. Generally, the denser or more clustered the desired singular values of A are, then the more difficult it is to compute them. Examples are provided in § 4.

We use an idea from Lanczos (1961, Chap. 3). This idea has also been used elsewhere; in particular, Golub and Kahan (1965) and Paige (1974) are relevant for the discussion here. Associated with any real $l \times n$ matrix A is the symmetric $(l+n)\times(l+n)$ matrix

$$B = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}. \tag{1.5}$$

It is straightforward to demonstrate that the distinct eigenvalues of B are 0 and $\pm\bar{\sigma}_k$, $1 \leqq k \leqq s \leqq r$, where r is the rank of A and the $\bar{\sigma}_j$ with $\bar{\sigma}_1 > \bar{\sigma}_2 > \cdots \bar{\sigma}_s > 0$ are the distinct nonzero singular values of A. The multiplicity of the 0 eigenvalue of B is $l+n-2r$. In fact, we have the following lemma, which states that the singular values and corresponding singular vectors of A form an eigenvalue eigenvector decomposition of B. The proof of Lemma 1.2 follows directly from the definitions.

LEMMA 1.2. *Let A be a real $l \times n$ matrix with $l \geqq n$ and B be defined by* (1.5). *Let $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ be the singular values of A corresponding to the singular value decomposition $A = X\Sigma Y^T$. Define X_1, X_2 and Σ_1 as in Lemma* 1.1 *and let Z be the following $(l+n)\times(l+n)$ matrix*:

$$Z = \frac{1}{\sqrt{2}} \begin{bmatrix} X_1 & X_1 & \sqrt{2}X_2 \\ Y & -Y & 0 \end{bmatrix}. \tag{1.6}$$

Then (i) *Z is orthogonal and if*

$$\Lambda = \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & -\Sigma_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{then } BZ = Z\Lambda. \tag{1.7}$$

That is, Z is an eigenvector basis for B, and the eigenvalues of B are just $\pm\sigma_j$, $j = 1, \cdots, n$, together with $l-n$ zero eigenvalues corresponding to $l \neq n$.

(ii) *Zero eigenvalues of B that correspond to zero singular values of A are characterized in B by eigenvectors $(x, y)^t$ with $x \neq 0$ and $y \neq 0$.*

In the above we have used $(x, y)^t$ as an abbreviation for the awkward notation $(x^T, y^T)^T$. We have the following converse of Lemma 1.2, which allows us to obtain suitable sets of singular vectors of A from the computed eigenvectors of B. Observe from (1.3) that the vectors X_2 perform no real function in the singular value decomposition of A; therefore we will not compute these singular vectors. We only compute right singular vectors Y and a suitable set of left singular vectors for X_1.

LEMMA 1.3. *Let A be a real $l \times n$ matrix with $l \geqq n$, and let B be defined by* (1.5).

(i) *For any positive eigenvalue λ_i of B and corresponding eigenvector $(x_i, y_i)^t$ of norm $\sqrt{2}$, λ_i is a singular value of A, and x_i and y_i are respectively left and right singular vectors of A corresponding to λ_i.*

(ii) *If 0 is an eigenvalue of B and there is a corresponding eigenvector of B, $(x_i, y_i)^t$ with $x_i \neq 0$ and $y_i \neq 0$, then 0 is a singular value of A; otherwise, A has full rank. Appropriate left and right singular vectors of A can be obtained by computing orthogonal eigenvectors $(x_i, y_i)^t$, $i = 1, 2, \cdots, n - r$, of B with $x_i \neq 0$ and $y_i \neq 0$ and then orthogonalizing the x_i and the y_i, respectively.*

Proof. Part (i) is a direct consequence of (1.3). Part (ii) follows from Lemma 1.2 and the fact that these y_i and x_i must lie respectively in the null spaces of A and of A^T. □

Thus, we can generate a singular value decomposition of a given $l \times n$ matrix A by computing the eigenvectors of the related symmetric matrix B given in (1.5). Golub and Kahan (1965) pointed out that the Lanczos recursions together with the matrix B in (1.5) could be used to obtain a Lanczos bidiagonalization algorithm for computing a singular value decomposition of a matrix. However, they chose to use Householder transformations to reduce the given matrix A to bidiagonal form because of their inherent stability. Paige (1974), in the context of developing a Lanczos procedure for solving the linear least squares problem for large sparse matrices, elaborated upon the Lanczos bidiagonalization algorithm proposed by Golub and Kahan (1965).

Recently, Golub, Luk and Overton (1981) have applied an iterative block Lanczos version of the Lanczos bidiagonalization to the matrix B to compute a few of the largest singular values of a real matrix A and corresponding singular vectors. Block Lanczos procedures are procedures for maximizing sums and differences of Rayleigh quotients of B (see Cullum (1978), Cullum and Donath (1974)), and are therefore very suitable for computing a few of the largest singular values (and corresponding vectors) since these appear on the extremes of the spectrum of B.

In some applications, more than just a few singular vectors may be required, or singular vectors may be needed from some other portion of the spectrum than the large end. In some data reduction schemes a few of the singular vectors corresponding to the largest singular values may suffice. However, in other problems one may need more vectors to reproduce a reasonable facsimile of the given pattern. In stabilization schemes for solving ill-posed problems, the singular values of primary interest are those at the lower end of the spectrum. In replacing A by the symmetric matrix B, the smallest singular values of A become the middle of the spectrum of B and thus are as far from the extremes of the spectrum as possible. It is very difficult, if not impossible, for a general block Lanczos procedure that starts from a set of randomly generated vectors to compute these smallest singular values. The algorithm that we present can be used to compute either a few or many distinct singular values at the ends of or on interior portions of the spectrum.

In § 2 we briefly summarize the Lanczos eigenvalue/eigenvector procedures for real symmetric matrices that are described in Cullum and Willoughby (1979), (1980a), (1980b), (1981). In these references it was demonstrated that for many symmetric matrices a single-vector Lanczos procedure with no reorthogonalization could be used, not only to reliably compute the distinct, extreme eigenvalues, but to compute the distinct eigenvalues in any portion of the spectrum. The amount of computation required is a function of the denseness of the desired eigenvalues, of their relative locations in the specrum of the given matrix (i.e., extreme or interior) and of the gap stiffness, the ratio of the largest gap between two neighboring distinct eigenvalues to the smallest such gap. The gap stiffness is a measure of the maximum variation in the spacing between successive eigenvalues of B. If this stiffness is "not large"; then the Lanczos tridiagonalization procedure can readily compute all of the eigenvalues of the given matrix.

In § 3 we apply these Lanczos procedures to the symmetric matrix B in (1.5), incorporating savings in operation counts that are outlined in Golub and Kahan (1965). We summarize the special properties of the resulting symmetric tridiagonal matrices that are generated and consider the question of suppressing the extra zero eigenvalues of B which occur whenever $l \neq n$.

In § 4 we apply the resulting Lanczos singular value procedure to several test problems and illustrate both its performance and the differences between our proposed algorithm and the one proposed in Paige (1974). Examples illustrate that our procedure provides the user with a look-ahead capability and superior resolution power, and eliminates much of the ambiguity that could arise in picking a convergence tolerance. Corresponding singular vectors are computed. Estimates of the errors of all of the computed quantities are given. Fortran code implementing the algorithm described here is given in Cullum, Willoughby and Lake (1982).

2. Lanczos tridiagonalization without reorthogonalization. We briefly summarize the eigenvalue and eigenvector algorithms for real symmetric matrices described in Cullum and Willoughby (1979), (1980a), (1980b), (1981). Lanczos tridiagonalization transforms a real symmetric $N \times N$ matrix B into a family of real symmetric tridiagonal matrices T_m, $m = 1, 2, \cdots$. We select a unit N-vector $\tilde{v}_1$ randomly. There is an implicit assumption in these Lanczos procedures that the starting vector has a nontrivial projection on some eigenvector of B corresponding to each eigenvalue that we want to compute. We than use the following Lanczos recursion for $i = 1, \cdots, m$ with $\tilde{v}_0 = 0$, $\beta_1 = 0$ and

$$\beta_{i+1}\tilde{v}_{i+1} = B\tilde{v}_i - \alpha_i\tilde{v}_i - \beta_i\tilde{v}_{i-1}, \tag{2.1}$$

with the coefficients recommended by Paige (1972):

$$\alpha_i = \tilde{v}_i^T(B\tilde{v}_i - \beta_i\tilde{v}_{i-1}) \quad \text{and} \quad |\beta_{i+1}| = \|B\tilde{v}_i - \alpha_i\tilde{v}_i - \beta_i\tilde{v}_{i-1}\| \tag{2.2}$$

to generate symmetric tridiagonal matrices T_m of order m with

$$T_m(i, i) = \alpha_i \quad \text{and} \quad T_m(i, i+1) = T_m(i+1, i) = \beta_{i+1}. \tag{2.3}$$

This generation requires only the matrix-vector products $B\tilde{v}_i$, and only the two most recently generated Lanczos vectors must be kept in fast storage. The subsequent eigenvector computations use the Lanczos vectors generated in the Lanczos eigenvalue computations. The user can either store these vectors off-line as they are generated, to be recalled for the eigenvector computations, or regenerate these vectors for the eigenvector computations.

For each m we define the related symmetric tridiagonal matrix of order $m-1$,

$$\hat{T}_2(i,i)=\alpha_{i+1} \quad \text{and} \quad \hat{T}_2(i,i+1)=\hat{T}_2(i+1,i)=\beta_{i+2}, \tag{2.4}$$

obtained from a given T_m by deleting the first row and column. This definition should include the parameter m, but for simplicity we have left it out. The eigenvalue computation proceeds as follows.

LANCZOS EIGENVALUE PROCEDURE FOR SYMMETRIC MATRICES.

Step 0. Set $\tilde{v}_0=0$ and $\beta_1=0$ and generate a random unit vector $\tilde{v}_1$ of order N.

Step 1. For some suitable M generate for $j\leqq M$, the scalars α_j and β_{j+1} using (2.1)–(2.2). Store the α_j, $1\leqq j\leqq M$, the β_j, $2\leqq j\leqq M+1$, and the last two Lanczos vectors $\tilde{v}_M$ and $\tilde{v}_{M+1}$ off-line.

Step 2. For some $m\leqq M$, compute the eigenvalues of T_m in the intervals of interest where convergence has not yet been established. Determine the numerical multiplicities of each of these eigenvalues. Accept numerically multiple eigenvalues as converged eigenvalues of B. Test each simple computed eigenvalue of T_m to determine whether or not it is pathologically close to an eigenvalue of the corresponding $\hat{T}_2$ matrix. If so, reject that eigenvalue as spurious. Otherwise, accept it as an approximate eigenvalue of B.

Step 3. Compute error estimates for the simple "good" eigenvalues obtained in Step 2. If convergence is indicated, terminate. Otherwise increment m, if necessary enlarge T_M, and go to Step 2.

If Sturm sequencing is used to calculate the eigenvalues of T_m, then the multiplicity test and the test for rejection in Step 2 are done simultaneously, so there is no extra cost incurred in doing the rejection test.

LANCZOS EIGENVECTOR PROCEDURE FOR SYMMETRIC MATRICES AND CONVERGED EIGENVALUES.

Step 0. The matrices T_m, $m\leqq M$, generated in the Lanczos eigenvalue computations are read from storage.

Step 1. Using Sturm sequencing (see for example Jennings (1977)) determine, for each eigenvalue μ being considered, the first value $m1(\mu)$ of m for which μ appears as an eigenvalue of T_m to within a given accuracy, typically $\varepsilon(\mu)\equiv\max(10^{-10}, 10^{-10}|\mu|)$. Also if possible compute the first value $m2(\mu)$ of m for which μ appears as a double eigenvalue of T_m to within $\varepsilon(\mu)$.

Step 2. For each μ, define $ma(\mu)\equiv m1(\mu)+\frac{3}{8}(m2(\mu)-m1(\mu))$ unless $m2(\mu)>M$, in which case set $ma(\mu)=M$. (See comments below.) For each μ compute a unit eigenvector w of $T_{ma(\mu)}$ and use it to compute the following error estimate for the accuracy of the corresponding Ritz vector.

$$\varepsilon_\mu^e=|\beta_{m+1}w(m)|. \tag{2.5}$$

In (2.5) $m=ma(\mu)$ and $w(m)$ is the mth component of the corresponding eigenvector w. If this error estimate is small go to the next μ. Otherwise, use Sturm sequencing to determine if there is a spurious eigenvalue close to μ. If not go to the next μ. If there is, then change $ma(\mu)$ and repeat the tridiagonal eigenvector computation for that μ. Note that the associated expensive Ritz computation is not performed until Step 3.

Step 3. For each relevant eigenvalue μ and corresponding T_m eigenvector w, we calculate the corresponding normalized Ritz vector

$$z \equiv \tilde{V}w/\|\tilde{V}w\| \tag{2.6}$$

where $\tilde{V} \equiv \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_m\}$ and $m = ma(\mu)$ is a function of μ. A single pass through the Lanczos vectors $\tilde{v}_k$ is all that is required to generate a whole set of Ritz vectors. Note that the number of Lanczos vectors actually used varies with μ.

Step 4. Compute the corresponding error estimates for each Ritz vector z considered as an eigenvector of B:

$$\varepsilon_\mu \equiv \|Bz - \mu z\|. \tag{2.7}$$

Details of the eigenvalue and eigenvector computations and Fortran code are given in Cullum and Willoughby (1980a), (1980b), (1981), (1982).

In Cullum and Willoughby (1980a) we saw empirically that for a given eigenvalue μ almost any choice of $ma(\mu)$ within a very broad centralized range between $m1(\mu)$ and $m2(\mu)$ is suitable for the eigenvector computation. When $m2(\mu)$ is not known and we set $ma(\mu) = M$, it is of course possible that $ma(\mu)$ is not large enough. This can be checked by looking at the error estimate obtained from (2.5). This quantity gives an excellent a priori estimate of the accuracy of the corresponding Ritz vector considered as an eigenvector of B. If for some μ this estimate is poor, we can, as indicated in Step 2, modify the order $ma(\mu)$ even to the extent of enlarging T_M and repeating Step 2.

Step 3 indicates that all the Ritz vectors are computed with one pass through the Lanczos vectors. Whether or not this is feasible of course depends upon the storage available, since for each computed μ both the computed eigenvector of $T_{ma(\mu)}$ and the corresponding Ritz vector are kept in main storage for this computation. The point is that an entire set of eigenvectors can be computed simultaneously.

The eigenvalue/eigenvector computations occur in two stages. The eigenvalues are computed first, and then for a selected subset of the eigenvalues, corresponding eigenvectors are computed. In order for the eigenvector procedure to work properly each eigenvalue for which an eigenvector is to be computed must have converged as an eigenvalue of the tridiagonal matrices T_m, $m = 1, 2, \cdots$. It is not sufficient for the error estimates in (2.5) to be small, because this could be due to a small β_{j+1}. The last component $w(m)$ of the associated eigenvector of T_m must be small. This of course occurs when the error estimate is small as long as the corresponding β_{m+1} is not abnormally small.

The amount of computation required to compute the desired eigenvalues and eigenvectors of a given matrix B depends directly upon the denseness of the desired eigenvalues, on the relative locations of these eigenvalues in the spectrum of B and on the gap stiffness. Given an eigenvalue on the extreme of the spectrum and an eigenvalue in the interior of the spectrum but with the same eigenvalue gaps, the extreme eigenvalue will converge first. However, an interior eigenvalue with a larger gap than a particular extreme eigenvalue may converge sooner than that extreme one does. Since there is no reorthogonalization, eigenvalues that have converged by a given m may begin to replicate as m is increased. The degree of replication that occurs depends upon the gap ratios: the larger the ratio, the more the replication. The more replication there is the more difficult it is to compute small dense eigenvalues.

Examples of the convergence achievable for matrices with different eigenvalue distributions are given in Cullum and Willoughby (1979), (1980a), (1980b), (1981).

3. Singular value and singular vector computations. We obtain a Lanczos singular value algorithm for any real $l \times n$ matrix A by applying the Lanczos procedures described in the preceding section to the corresponding matrix B in (1.5) with a particular choice of starting vector suggested by Golub and Kahan (1965). First, in Lemma 3.1 we summarize the recursions that result from applying the Lanczos recursions given in (2.1)–(2.2) to the matrix B in (1.5).

LEMMA 3.1. *Let A be a real $l \times n$ matrix and let B be the associated symmetric matrix defined in* (1.5). *Apply the Lanczos tridiagonalization recursion specified in* (2.1) *and* (2.2) *to B with the unit starting vector $(v, 0)^t$ where v is $l \times 1$ or $(0, u)^t$ where u is $n \times 1$. Then in either case the diagonal entries of the symmetric tridiagonal matrices generated are all identically zero, the Lanczos vectors generated alternate between the two forms $(v, 0)^t$ and $(0, u)^t$ and the Lanczos recursions in* (2.1) *and* (2.2) *reduce to the following*:

(i) *For a starting vector of the form $(v, 0)^t$, define $v_1 \equiv v$, $u_0 \equiv 0$ and $\beta_1 \equiv 0$. For $i = 1, \cdots, m/2$ we obtain the Lanczos recursions*

$$\begin{aligned} \beta_{2i} u_i &= A^T v_i - \beta_{2i-1} u_{i-1}, \\ \beta_{2i+1} v_{i+1} &= A u_i - \beta_{2i} v_i. \end{aligned} \tag{3.1}$$

(ii) *For a starting vector of the form $(0, u)^t$ define $u_1 \equiv u$, $v_0 \equiv 0$ and $\beta_1 \equiv 0$. For $i = 1, \cdots, m/2$ we obtain the Lanczos recursions*

$$\begin{aligned} \beta_{2i} v_i &= A u_i - \beta_{2i-1} v_{i-1}, \\ \beta_{2i+1} u_{i+1} &= A^T v_i - \beta_{2i} u_i, \end{aligned} \tag{3.2}$$

where in each case in (3.1) *and* (3.2) *the β_j are chosen to normalize the u_i and v_i vectors.*

Proof. The proof consists of directly writing down the Lanczos recursions for B and then observing that they reduce to either (3.1) or (3.2) above. □

Lemma 3.1 tells us that as long as the starting vector is of the form $(v, 0)^t$ or $(0, u)^t$, then successive Lanczos vectors are of the form $(0, u_i)^t$ or $(v_i, 0)^t$ or vice versa. Thus the total storage required for two successive Lanczos vectors is only $l + n$. The Lanczos symmetric tridiagonal matrices T_m generated are for $i = 1, \cdots, m/2$,

$$\begin{aligned} T_m(2i-1, 2i-1) &= 0, \qquad T_m(2i-1, 2i) = \beta_{2i}, \\ T_m(2i, 2i) &= 0, \qquad T_m(2i, 2i+1) = \beta_{2i+1}. \end{aligned} \tag{3.3}$$

Since each diagonal entry of the tridiagonal matrices generated is 0, only one long vector array is needed to store these matrices. At each iteration either Au or $A^T v$ is computed, not both. Thus, although the order of B may be as much as twice that of A, the computational requirements per iteration of the Lanczos recursion do not increase proportionally when we replace A by B in (1.5). However, we should repeat that B has twice as many distinct nonzero eigenvalues as A has distinct nonzero singular values, since each singular value of A (including also any zero singular values) appears as a $\pm$ pair of eigenvalues of B. Moreover, the $\pm$ pairs of the small singular values of A become the center of the eigenvalue spectrum of B. Thus the small singular values are the most difficult singular values for our Lanczos procedure to compute if these values are small in magnitude or very densely packed. Therefore, even though the cost per iteration may not be more than working directly with a symmetric matrix whose order is $(l+n)/2$, the number of iterations required to get

the desired eigenvalues of B may be considerably more than what would be required with a symmetric matrix with an eigenvalue distribution that matched the distribution of the singular values of A. In § 4 we present one such specific comparison.

An obvious question to consider is whether or not it matters which choice of starting vector $(v, 0)^t$ or $(0, u)^t$ that we use and whether or not we will lose any part of the desired eigenspace of B if one of these special forms is used. A related question for the case $l \neq n$ is what effect does one of these choices have on one's ability to compute the zero singular values of A (if there are any) without being confused by zero eigenvalues of B arising when $l \neq n$. The following lemma provides an answer to these questions.

LEMMA 3.2. *Let A be a real $l \times n$ matrix. Let B be the associated symmetric matrix given in* (1.5).

(i) *If $l > n$ then the eigenvectors of B corresponding to zero eigenvalues of B resulting from $l \neq n$ have no projections on vectors of the form $(0, u)^t$ where u is $n \times 1$. If $l < n$ then this statement holds for vectors of the form $(v, 0)^t$ where v is $l \times 1$.*

(ii) *If $l > n$ and u is generated randomly, then in exact arithmetic the expectation that u will have a projection on each desired right singular vector of A is the same as it would be for computing the distinct eigenvalues of the symmetric matrix A^TA. Similarly, for $l < n$, the expectation that v will have a projection on each of the desired left singular vectors is the same as it would be for computing the distinct eigenvalues of the symmetric matrix AA^T.*

Proof. The proof of (i) is immediate from Lemma 1.2 and its analogue for the case $l < n$. Lemma 1.2 states that for $l > n$ the eigenvectors of B corresponding to the extraneous 0 eigenvalues due to $l \neq n$ are of the form $(x, 0)^t$ where x is of length l. Similarly when $l < n$, these eigenvectors are of the form $(0, y)^t$. To prove part (ii) we show that in exact arithmetic the Lanczos vectors u_i generated using (3.2) are the Lanczos vectors obtained by applying the symmetric Lanczos recursions to A^TA. Applying (3.2) and rearranging we obtain the following recursion:

$$\beta_{2i}\beta_{2i+1}u_{i+1} = A^TAu_i - (\beta_{2i}^2 + \beta_{2i-1}^2)u_i - \beta_{2i-1}\beta_{2i-2}u_{i-1}$$

where $\beta_1 \equiv 0$. Using the fact that in exact arithmetic the vectors u_i are orthonormal, it is straightforward to show that the coefficients in the above recursion are precisely the corresponding α and β coefficients for A^TA defined by (2.2). □

Restating Lemma 3.2 we have that if A has more rows than columns, then the appropriate vector is of the form $(0, u)^t$ where u has the same dimension as the number of columns, and if A has more columns than rows, then the appropriate starting vector is of the form $(v, 0)^t$ where again v has the same dimension as the number of columns. These choices minimize our chances of introducing any of the eigenvectors of B associated with any zero eigenvalues of B arising from $l \neq n$.

Lemma 3.2 raises the obvious question of computing the singular values of A by computing the eigenvalues of the smaller of the two corresponding symmetric matrices A^TA or AA^T, instead of working with A directly. This is not a good idea for several reasons. The nonzero eigenvalues of either of these matrices are the squares of the nonzero singular values of A. So, in particular, very small singular values may appear numerically as zero eigenvalues in A^TA or AA^T, and very large singular values become even larger. The squares of any such small singular values are also closer together than the singular values themselves are, and the squares of any large eigenvalues are further apart. Thus, the gap stiffness of the eigenvalues of A^TA may be much worse than that of the singular values themselves. However, the smallest and the largest singular values of A become the extremes of the spectrum of either A^TA or AA^T,

and in some cases that may yield a significant advantage in terms of how our Lanczos algorithm converges. In fact, if A is a well-conditioned matrix in the sense that there are no singular values whose squares are too near "zero", then it may be possible to compute good approximate singular values and vectors by computing the eigenvalues and eigenvectors of either one of the above matrices. However, even in this case there can be degradation of the accuracy over what is achievable if A is used directly. Cullum and Willoughby (1980c) give some examples illustrating such differences. Moreover, there is no a priori test to determine whether or not a given matrix is well-conditioned in the above sense, so it is not possible to recommend this approach.

The following lemma tells us that the Lanczos tridiagonal matrices T_m generated using the Lanczos recursion in (3.1) or (3.2) retain some of the properties of the original B matrix.

LEMMA 3.3. (i) *For even orders $m = 2j$, $j = 1, 2, \cdots$, the eigenvalues of T_m occur in $\pm$ pairs.*

(ii) *For odd orders $m = 2j-1$, $j = 1, 2, \cdots$, the eigenvalues of T_m occur in $\pm$ pairs together with an extraneous 0 eigenvalue.*

Proof. For any γ let $d_i(\gamma) =$ determinant $(T_i - \gamma I)$. We prove by induction that for any γ and for any j

$$d_{2j}(\gamma) = d_{2j}(-\gamma) \quad \text{and} \quad d_{2j-1}(\gamma) = -d_{2j-1}(-\gamma).$$

That is, for any order m if γ is an eigenvalue of T_m then $-\gamma$ is also an eigenvalue of T_m. In addition, for odd orders m, $d_m(0) = -d_m(0)$, so therefore 0 must be an eigenvalue of such T_m. Set $d_0(\gamma) \equiv 1$. Clearly, $d_1(\gamma) = -\gamma = -d_1(-\gamma)$ and $d_2(\gamma) = \gamma^2 - \beta_2^2 = d_2(-\gamma)$. For symmetric tridiagonal matrices with 0 diagonals we have the determinant recursion $d_i(\gamma) = -\gamma d_{i-1}(\gamma) - \beta_i^2 d_{i-2}(\gamma)$. Now assume (i) and (ii) are true for $j = k$. Then using this determinant recursion we get that $d_{2k+1}(-\gamma) = \gamma d_{2k}(-\gamma) - \beta_{2k+1}^2 d_{2k-1}(-\gamma) = \gamma d_{2k}(\gamma) + \beta_{2k+1}^2 d_{2k-1}(\gamma) = -d_{2k+1}(\gamma)$. Similarly, we get $d_{2k+2}(-\gamma) = d_{2k+2}(\gamma)$. □

By direct substitution it is easy to show, for any order m and any positive eigenvalue μ of T_m with eigenvector $w = (w_1, w_2, \cdots, w_m)^t$, that the vector $\bar{w} = (w_1, -w_2, \cdots, (-1)^{m-1} w_m)^t$ is an eigenvector of T_m for the corresponding eigenvalue $-\mu$. Theoretically, we know that the matrices T_m cannot have multiple eigenvalues unless one of the β_j vanish. In practice, however, numerically multiple eigenvalues are obtained. If A does not have full rank, then for large enough m, T_m should have some very small eigenvalues corresponding to the zero singular values of A. These, however will occur in $\pm$ pairs. For odd order T_m, as Lemma 3.3 demonstrated, there is an extraneous 0 eigenvalue. By direct substitution it is easy to demonstrate that the even-numbered components of any eigenvector corresponding to such a 0 eigenvalue must all vanish. Lemma 3.3 tells us that we need only compute the nonnegative eigenvalues of each T_m, and that to avoid confusion with an extraneous zero eigenvalue due only to the peculiar form of T_m, we should use only even ordered T_m.

Before proceeding we make one more observation related to part (ii) of Lemma 3.2 and to our comments about the matrix $A^T A$. Consider for example (3.2). We can write the first equation in matrix form as

$$AU = VJ \tag{3.4}$$

where $U = \{u_1, \cdots, u_i\}$, $V = \{v_1, \cdots, v_i\}$ and J is the $i \times i$ bidiagonal matrix with $J(j, j) = \beta_{2j}$ and $J(j, j+1) = \beta_{2j+1}$. The nonzero singular values of J are the same as the positive eigenvalues of

$$\bar{J} \equiv \begin{pmatrix} 0 & J \\ J^T & 0 \end{pmatrix}.$$

But there is a simple permutation of $\bar{J}$ that maps $\bar{J}$ into T_{2i}. Thus, since singular values are preserved under permutations, we could compute the singular values of A by computing the eigenvalues of the smaller symmetric tridiagonal matrix J^TJ where $J^TJ(j,j)=\beta_{2j}^2+\beta_{2j-1}^2$ and $J^TJ(j,j+1)=\beta_{2j}\beta_{2j+1}$. (Note that theoretically this is just the matrix obtained earlier when we considered A^TA in the proof of Lemma 3.3.)

This raises the possibility of computing the singular values of A by computing only with the smaller but nonsymmetric bidiagonal matrix J. The singular value code in LINPACK uses a modified QR algorithm that works directly with J. Golub, Luk and Overton (1981) also use the J matrices instead of the corresponding T-matrices. In our implementation we work directly with the symmetric tridiagonal matrices T_m. These matrices are needed for our classification test (see Step 2 of our algorithm in § 4), and are needed for computing the error estimates for the computed singular values and vectors.

In order to study the behavior of the proposed algorithm systematically, it is desirable to have a family of test matrices whose singular values and vectors are known and such that various types of singular value distributions can be considered. Lemma 3.4 tells us that the family of rectangular diagonal matrices given in (1.2) is a suitable family since these matrices are equivalent, in the sense specified in this lemma, to any other family of matrices except for the way in which the roundoff error incurred in evaluating Au or A^Tv propagates. This proof is straightforward.

LEMMA 3.4. *Given any real $l\times n$ matrix A with $l\geqq n$, in eact arithmetic there is a member Σ of the family of rectangular diagonal matrices in* (1.2) *which has the same singular values as A and generates the same Lanczos tridiagonal matrices T_m, $m=1,2,\cdots$, as A does when the recursions in* (3.2) *are applied to it using a starting vector $(0,u)^t$ for the rectangular matrix Σ, and using the corresponding starting vector $(0,Yu)^t$ for the matrix A. A similar result holds when $l\leqq n$ with $\Sigma=[\Sigma_1\ 0]$,*

Lemma 3.4 is valid only in exact arithmetic, which of course we do not have. However, it does indicate that we should be able to obtain some understanding of how our Lanczos singular value procedure functions by running numerical tests on members chosen from this very special family of matrices. Numerical results are discussed in the next section.

4. Numerical results. The following Lanczos singular value procedure which is based upon Lemmas 1.2–1.3 and 3.1–3.3 was programmed in Fortran and exercised on an IBM 3033. The Fortran code is available in Cullum, Willoughby and Lake (1982). We outline this procedure for an $l\times n$ real, rectangular matrix A with $l\geqq n$. The analogue for the case $l\leqq n$ is easily obtained and the computer program directly handles either case.

LANCZOS SINGULAR VALUE/VECTOR PROCEDURES ($l\geqq n$).

Step 0. Set $v_0=0$ and $\beta_i=0$ and generate a random unit vector u_1 of order n.

Step 1. For some suitable M, generate for $1\leqq i\leqq M$ the scalars β_{2i} and β_{2i+1} and unit vectors v_i, u_{i+1} such that

$$\beta_{2i}v_i=Au_i-\beta_{2i-1}v_{i-1},$$
$$\beta_{2i+1}u_{i+1}=A^Tv_i-\beta_{2i}u_i.$$

Store the scalars β_j, $1\leqq j\leqq 2M+1$, and the last two Lanczos vectors v_M and u_{M+1}.

Step 2. For some even $m\leqq 2M$, compute the nonnegative eigenvalues of T_m in the intervals of interest on those subintervals where convergence has not

yet been established. T_m is the symmetric tridiagonal matrix of order m given in (3.3). Determine the numerical multiplicities of each of these eigenvalues. Accept each numerically multiple eigenvalue as a converged singular value of A. Test each simple computed eigenvalue μ of T_m to determine whether or not it is also an eigenvalue of the corresponding $\hat{T}_2$ (the matrix obtained by deleting the first row and column of T_m). If it is, reject that eigenvalue as "spurious"; otherwise, accept it as good and label it as an approximation to a singular value of A. (Note that if the Sturm sequencing version of the symmetric tridiagonal eigenvalue subroutine is being used then the test for numerical multiplicity and the $\hat{T}_2$ test for spurious eigenvalues are done simultaneously, so no extra cost is incurred.)

Step 3. Using inverse iteration on $(T_m - \mu I)$ compute an error estimate (cf. (2.5)) for each μ which is an "isolated" (considered as an eigenvalue of T_m) approximate singular value of A.

Step 4. Test for convergence of the computed approximate singular values using the error estimates. If convergence of all the desired singular values is observed, proceed to the singular vector computations. Otherwise increment m, return to Step 2 and generate additional β_j. Depending upon the original choice of M, this may mean using $m > 2M$, so that additional β must be generated. If so go to Step 1 instead of Step 2.

Step 5. Compute left and right singular vectors for each of a user-selected subset of the computed converged singular values by computing the corresponding eigenvectors of B using the Lanczos eigenvector procedure described in § 2 appropriately modified to incorporate the special structure of the tridiagonal matrices T_m and of the Lanczos vectors.

An important and unique feature of our algorithm is the determination (see Step 2) of the "good" eigenvalues of T_m on the intervals of interest. In exact arithmetic, for $l \geqq n$, each tridiagonal matrix T_m (for small enough m) is the orthogonal projection of the corresponding B matrix (1.5) onto the subspace spanned by the Lanczos vectors

$$\{(0, u_1)^t, (v_1, 0)^t, (0, u_2)^t, (v_2, 0)^t, \cdots, (v_{m/2}, 0)^t\}.$$

Therefore, in exact arithmetic every nonzero eigenvalue of T_m can be considered as an approximation to an eigenvalue of B and thus to a singular value of A. However, in practice, due to roundoff errors, the Lanczos vectors generated are not globally orthogonal, so that the matrices T_m are not orthogonal projections of B, and in fact some of the eigenvalues of a given T_m may be extraneous. Our selection as stated in Step 2 is based upon the identification and subsequent rejection of these "spurious" eigenvalues of T_m. This selection is made prior to and independent of the computation of the error estimates in Step 3. This list of "good" singular values at a given value of m provides a look-ahead capability. It identifies which of the eigenvalues of T_m can be expected to persist as we enlarge m and gives estimates for the singular values of A on the intervals of interest long before convergence has occurred.

In Step 3 the error estimates ε_μ in (2.5) are calculated for a selected set of the isolated "good" eigenvalues of T_m; these estimates are then used in Step 4 with a weak convergence tolerance to determine whether convergence has occurred. An eigenvalue of T_m is considered to be isolated if it is a simple eigenvalue of T_m and the minimal gap between it and the eigenvalues of T_m closest to it is larger than a tolerance which depends upon the eigenvalue in question. See Cullum, Willoughby and Lake (1982) for details.

An alternative approach for determining which simple eigenvalues of T_m represent singular values of A has been used elsewhere (cf. Paige (1974)) and is based upon computing the error estimates ε_μ defined in (2.5) for every simple eigenvalue of T_m and then testing these estimates relative to a user specified tolerance, ε_{tol}. Before proceeding we give examples that illustrate possible difficulties that can result from using this latter classification scheme, and that illustrate the look-ahead capability of our procedure and its superior resolution power.

Table 4.1 illustrates the sensitivity of the eigenvalue selection process to the user-selected tolerance ε_{tol}. It also illustrates that, given only a list of eigenvalues of T_m together with their corresponding ε_μ error estimates, one cannot necessarily make statements about the actual overall distribution of the singular values of A. The A matrix for Table 4.1 is only of order 20, and one would expect the situation to be much more complicated if we were working with a larger matrix. Following Lemma 3.4 we use rectangular diagonal matrices as test problems, and we will treat each example as though it were unsymmetric.

Example 4.1. Let A_1 be the rectangular 20×20 diagonal matrix with the diagonal entries

0.1,	1.7,
0.2,	1.8,
0.3,	1.9,
1.0,	6.9,
1.000001,	6.9001,
1.2,	9.9001,
1.3,	9.900101,
1.4,	9.900111,
1.5,	10,
1.6,	100.

We applied our Lanczos singular value procedure to A_1, letting $m = 40$, 60, and 70. Table 4.1 lists corresponding computed eigenvalues of T_m, the accuracy of those eigenvalues which are approximate singular values of A_1, and the computed error estimates. In each table the column labeled "Our class" contains our classification of the corresponding eigenvalue. If the eigenvalue is "good", then this classification is just the multiplicity of that eigenvalue as an eigenvalue of the corresponding T_m. if the eigenvalue is "spurious", then this classification is 0. The column labeled "Computed amingap" contains the minimal computed gap between the corresponding computed good eigenvalue (singular value) and the good eigenvalues (singular values) that are closest to it. An * means that the estimate of the error in that singular value indicates 12 or more digits of accuracy. In the tables the eigenvalues are numbered according to their position as an eigenvalue of T_m. Observe that the order N of B_1 in (1.5) corresponding to A_1 is 40 and that B_1 has 40 nonzero eigenvalues. Hence the orders of the tridiagonal matrices being considered are respectively $m = N$, $m = 1.5N$ and $m = 1.75N$, orders that are quite reasonable.

From Table 4.1a we see that at $m = 40$ our algorithm labels all of the 17 computed positive eigenvalues of T_m as "good", giving us an overall picture of the singular value spectrum of A. From Table 4.1a we also see that a classification procedure which was based purely on the size of the error estimates and used any convergence tolerance of between 7 and 10 digits would identify at most the five singular values 6.9, 6.9001, 9.900111, 10 and 100 as converged and would provide no overall estimate of the rest of the singular value spectrum. Note that the computed error estimates for the singular

TABLE 4.1a
Example 4.1. *Identification of computed eigenvalues at* $m = 40$.

No. in T_m	Computed eigenvalue	Our class	Computed amingap	Error estimate	True error
1	.1530	1	.138	9×10^{-2}	5×10^{-2}
2	.2914	1	.138	5×10^{-2}	9×10^{-3}
3	.9987	1	.09	6×10^{-2}	1×10^{-3}
4	1.089	1	.09	4×10^{-1}	1×10^{-2}
5	1.281	1	.1	1×10^{-1}	2×10^{-2}
6	1.384	1	.1	2×10^{-1}	2×10^{-2}
7	1.580	1	.064	2×10^{-1}	2×10^{-2}
8	1.644	1	.064	2×10^{-1}	6×10^{-2}
9	1.800	1	.1	1×10^{-2}	2×10^{-4}
10	1.900	1	.1	7×10^{-4}	4×10^{-7}
11	6.900	1	10^{-4}	8×10^{-11}	*
12	6.9001	1	10^{-4}	2×10^{-10}	*
13	9.9001	1	10^{-6}	2×10^{-6}	*
14	9.900101	1	10^{-6}	1×10^{-6}	*
15	9.900111	1	10^{-5}	5×10^{-7}	*
16	10.000	1	.1	*	*
17	100.00	4	90.	*	*

values near 9.9 do not reflect the fact that these singular values have actually converged because these values are very close together.

At $m = 60$ all of the singular values of Example 4.1 have been computed accurately but the error estimates for the singular values 1.0 and 1.000001 do not clearly indicate their convergence. In Table 4.1b we list only those eigenvalues of T_m near 1 and near 9.9. All other error estimates clearly indicated convergence. Note the two spurious eigenvalues.

TABLE 4.1b
Example 4.1. *Identification of converged singular values,* $m = 60$.
(*All singular values have converged.*)

No. in T_m	Computed eigenvalue	Our class	Computed amingap	Error estimate	True error
3	.3000	1	.1	*	*
4	1.000	1	10^{-6}	3×10^{-8}	*
5	1.000001	1	10^{-6}	3×10^{-7}	*
6	1.200	1	.1	*	*
15	6.9001	2	10^{-4}	*	*
16	9.817	0	–	9×10^{-1}	–
17	9.9001	1	10^{-6}	*	*
18	9.900101	1	10^{-6}	*	*
19	9.900102	0	–	4×10^{-5}	–
20	9.900111	1	10^{-5}	*	*

From Table 4.1b we see that the singular values 1.000 and 1.000001 would only be identified by error estimate tolerances of 6 digits or less, but such a weak tolerance can lead to serious misclassification. We give an example of this type of misclassification in Table 4.1c. Since both singular values near 1.0 would be missed with a reasonable test based only on error estimates, one would have to enlarge m and recompute the

eigenvalues on this interval. The question one must then consider is: What other intervals should also be included when we enlarge m? If one uses just the error estimates as a guide in making this selection then one obtains two possible intervals of nonconvergence (.99, 1.01) and (9.9, 9.90015). Our algorithm, on the other hand, would know to consider only the former interval since the other eigenvalue with the small error estimates has been labeled spurious. If in fact we do take both of these intervals and enlarge to T_{70}, we get the quantities given in Table 4.1c. Here we have listed only those portions of the spectrum of T_{70} in the intervals (.99, 1.01) and (9.9, 9.90015).

TABLE 4.1c

Example 4.1. *Identification of computed eigenvalues at* $m = 70$ *on the subintervals* (.99, 1.01) *and* (9.9, 9.90015).

No. in T_m	Computed eigenvalue	Our class	Computed amingap	Error estimate	True error
4	1.000	1	10^{-6}	10^{-10}	*
5	1.000001	1	10^{-6}	10^{-9}	*
19	9.9001002[a]	1	10^{-6}	*	*
20	9.9001007	0	–	7×10^{-7}	–
21	9.900101[a]	1	10^{-6}	*	*
22	9.9001110	0	–	2×10^{-6}	–
23	9.9001112[a]	1	10^{-5}	*	*

[a] These values agree with the machine representations of these eigenvalues to more than 12 decimal digits.

From Table 4.1c we see that a weak identification test based only on the error estimates such as that needed to identify the singular values 1.0 and 1.000001 in Table 4.1b can lead to the mislabeling of spurious eigenvalues as good. With an error tolerance of 10 digits, eigenvalues 1.0 and 1.000001 are not accepted; with a tolerance of 9 digits, 1.0 is accepted, and with a tolerance of 8 digits both 1.0 and 1.000001 are accepted. However, we also note that with an error tolerance of 7 digits we would get a total of 21 eigenvalues accepted because eigenvalue $\mu = 9.9001007$ would be mislabeled as converged, when in fact it is extraneous. If we drop the tolerance to 6 digits 22 eigenvalues are accepted.

Table 4.1 clearly illustrates the sensitivity of an acceptance test which is based purely on the size of the error estimates to the choice of that tolerance. If the tolerance is tight, not much of the spectrum is seen initially; if it is loose then one may incorrectly label extraneous eigenvalues as converged singular values, thereby reducing the resolving power of the algorithm. Thus, if one wanted to have an algorithm based purely on the error estimates, then the only safe choice for ε_{tol} would seem to be a strong convergence tolerance of 9 or 10 digits, which as we have seen limits one's ability to obtain an overall picture of the spectrum at an early stage in the computations. However, we can easily demonstrate that a tight convergence tolerance can also lead to problems with such an approach.

In Cullum, Willoughby and Lake (1981) we gave a small 20×20 example where with a tight error tolerance the number of eigenvalues identified as good actually decreased when the order m of the Lanczos tridiagonal matrices T_m was increased. For this example at $m = 80$ with a convergence tolerance of anywhere from 6 to 10 digit accuracy, 18 eigenvalues were labeled as "good". At $m = 90$, however, with a

10 digit tolerance only 16 eigenvalues were "good". If the convergence tolerance was relaxed to 9 digits, then 17 eigenvalues were "good", and if the tolerance was relaxed further to 8 digits or less, then 18 eigenvalues were "good". Similar behavior occurred at other values of m. This happens because the computed error estimates can vary due to the clustering of eigenvalues of T_m. This clustering can be due either to the appearance of extraneous eigenvalues near the ones of interest or to the presence of genuine clusters of singular values in A. In either case the sizes of the error estimates can fluctuate as m is varied.

One other passing comment with respect to our classification procedure versus one based purely on the size of the computed error estimates: We only have to compute error estimates on those eigenvalues of T_m that have been classified as "good", whereas a classification procedure based solely on the size of the error estimates requires the computation of estimates for each (nonmultiple) computed eigenvalue of T_m.

It is interesting to note the difficulties that one encounters with an odd order T_m that are caused by the extraneous 0 eigenvalue. Several examples are given in Cullum, Willoughby and Lake (1981). For odd m, problems can occur with either classification scheme. Initially our algorithm will label the extraneous 0 eigenvalue of any odd order T_m as "good" because the corresponding $\hat{T}_2$ matrix is of even order and does not have a 0 eigenvalue. Eventually, however, as m is increased our algorithm will discover that the 0 is not related to A and eventually will relabel "0" as spurious. What happens in practice is that as the order m is increased a pair of the eigenvalues of $\hat{T}_2$ coalesces to 0. On the other hand, the error estimate approach successfully labels "0" as not converged for any m; however, it cannot identify very small singular values (until they become double) unless the convergence tolerance is weakened.

Now we apply our algorithm to a fairly large $l = n = 411$ test matrix, Example 4.2, which was generated in a power system study. The matrix A_2 is a symmetric negative definite matrix, and in practice we would use the program in Cullum and Willoughby (1980b) to directly compute the eigenvalues and eigenvectors. We can, however, for test purposes treat A_2 as an unsymmetric matrix, computing the eigenvalues as singular values and the eigenvectors as singular vectors, and then compare the cost of those computations with the cost of computing the eigenvalues and eigenvectors directly. This gives us one estimate of the cost of transforming a given matrix into the B-format in (1.5). A_2 has eigenvalues that range from -2.365 to -1.46×10^{-8}. The gaps between successive eigenvalues vary from .746 at $\lambda_1 = -2.365$ to 1.04×10^{-5} at $\lambda_{58} = -.65344$. Most of the gaps vary between 2×10^{-3} and 10^{-4}.

In the given application the small eigenvalues (singular values) and corresponding eigenvectors (singular vectors) are of interest. However, for completeness we consider the convergence of both ends of the singular value (eigenvalue) spectrum. First consider the convergence of the 6 largest singular values as m is increased. The observed convergence is summarized in Table 4.2a. The true error is not tabulated in any of these tables because the eigenvalues of A_2 are not known analytically.

Table 4.2a illustrates that the largest singular values, those in the interval (1.3, 2.5), can be computed easily. By $m = 60 = .073N$, where $N = 822$ is the order of B_2, all six have been computed accurately. In fact many of the large singular values can be computed easily. For example, by $m = 80 = .097N$ the 13 largest singular values are obtained.

Now consider the convergence of the six smallest singular values, those in the interval (0, .02) (see Table 4.2b). These values are in the absolute center of the spectrum of B_2 which has 822 distinct eigenvalues, twice as many as A_2, since each

nonzero eigenvalue of A_2 appears in B_2 as a $\pm$ pair of eigenvalues. Convergence of the smallest singular values is achieved by $m = 2N = 1644$ which is quite reasonable relative to the size of B_2.

TABLE 4.2a
Example 4.2. *Convergence of largest singular values. Interval* (1.3, 2.5).

Order of T_m	Computed singular value	Computed amingap	Error estimate
20	2.36524	.76	4×10^{-7}
	1.60550	.04	5×10^{-2}
	1.56259	.04	4×10^{-2}
40	2.36524	.746	1×10^{-13}
	1.61887	.045	2×10^{-5}
	1.57375	.037	5×10^{-5}
	1.53648	.037	2×10^{-4}
	1.44614	.02	2×10^{-3}
	1.42350	.02	1×10^{-3}
60	1.61887		*
	1.57375		*
	1.53648		8×10^{-12}
	1.44615		6×10^{-10}
	1.42350		4×10^{-10}

TABLE 4.2b
Example 4.2. *Convergence of smallest singular values on the interval* (0.0, .02).

Order T_m	Computed singular value	Computed amingap	Error estimate
$412 = .5N$	$.2236\times10^{-3}$		4×10^{-2}
	$.1295\times10^{-1}$		2×10^{-2}
$822 = N$	$.4159\times10^{-4}$		6×10^{-2}
	$.33\times10^{-2}$		1×10^{-2}
	$.129\times10^{-1}$		5×10^{-3}
	$.143\times10^{-1}$		5×10^{-2}
$1234 = 1.5N$	$.233\times10^{-6}$	.0027	2×10^{-1}
	$.27129\times10^{-2}$	.0007	9×10^{-4}
	$.34476\times10^{-2}$	.0007	4×10^{-4}
	$.12839\times10^{-1}$	.0003	6×10^{-5}
	$.13140\times10^{-1}$	.0003	1×10^{-4}
$1644 = 2N$	$.1457\times10^{-7}$	.0027	2×10^{-6}
	$.27128\times10^{-2}$	.0007	2×10^{-9}
	$.34467\times10^{-2}$	1.1×10^{-5}	6×10^{-8}
	$.34579\times10^{-2}$	1.1×10^{-5}	2×10^{-7}
	$.12838\times10^{-1}$	.0003	7×10^{-12}
	$.13140\times10^{-1}$	.0003	1×10^{-11}
$1850 = 2.25N$	$.1457\times10^{-7}$		3×10^{-10}
	$.27128\times10^{-2}$		*
	$.34467\times10^{-2}$		8×10^{-12}
	$.34579\times10^{-2}$		3×10^{-11}
	$.12838\times10^{-1}$		*
	$.13140\times10^{-1}$		*

Tables 4.2c and 4.2d allow us to compare the observed convergence of the singular values of A_2 obtained using the Lanczos singular value procedures with the corresponding observed convergence of the eigenvalues of A_2 obtained when we apply our symmetric Lanczos procedure directly to A_2. Such a comparison gives us some understanding of the penalties incurred in working with a nonsymmetric matrix versus a symmetric matrix of the same order and whose eigenvalue distribution matches the singular value distribution of the nonsymmetric matrix.

TABLE 4.2c
Example 4.2. *Convergence of eigenvalues largest in magnitude. Interval* $(-2.5, -1.3)$.

Order T_m	Computed good eigenvalues	Computed amingap	Error estimate
18=.044N	−2.36524	.764	3×10^{-9}
	−1.61827	.046	2×10^{-2}
	−1.57266	.043	2×10^{-2}
	−1.52928	.043	5×10^{-2}
	−1.4252	.104	2×10^{-2}
	−1.240	.127	2×10^{-1}
30=.073N	−2.36524	.764	2×10^{-12}
	−1.61887	.045	2×10^{-6}
	−1.57375	.037	5×10^{-6}
	−1.53648	.037	2×10^{-5}
	−1.44615	.023	3×10^{-4}
42=.102N	−2.36525		*
	−1.61887		*
	−1.57375		*
	−1.53648		3×10^{-10}
	−1.44615		2×10^{-10}
	−1.42350		8×10^{-8}

TABLE 4.2d
Example 4.2. *Convergence of eigenvalues smallest in magnitude on the interval* $(-0.013, 0.0)$.

Order T_m	Computed "good" eigenvalues	Computed amingap	Error estimate
104=.25N	$-.14576\times10^{-7}$	.0027	2×10^{-8}
	$-.27128\times10^{-2}$	.0007	2×10^{-5}
	$-.344762\times10^{-2}$	.0007	3×10^{-5}
	$-.48854\times10^{-2}$	.0014	2×10^{-2}
	$-.12839\times10^{-1}$	.0003	4×10^{-4}
	$-.13144\times10^{-1}$	.0003	8×10^{-4}
206=.50N	$-.14576\times10^{-7}$	.0027	*
	$-.27128\times10^{-2}$	.0007	*
	$-.34467\times10^{-2}$	1.1×10^{-5}	*
	$-.34579\times10^{-2}$	1.1×10^{-5}	*
	$-.12839\times10^{-1}$	.0003	*
	$-.1314\times10^{-1}$	.0003	*

We observe from Tables 4.2a and 4.2c that the large singular values converged even better than might be expected. Less than 50% more work was required to compute these quantities than is required when they are considered as eigenvalues

of the symmetric A_2. This is understandable since B_2 has twice as many eigenvalues as A_2 and the relative positions of these eigenvalues in the spectrum of B_2 are the same as in A_2, i.e., they are still extreme and with the same local gap structure. The convergence of the smallest singular values is, however, very different from the convergence of these values when they are generated as the smallest eigenvalues of A_2. These values appear at the exact center of the spectrum of B_2, so we get a pair of tiny eigenvalues $\cong \pm 10^{-8}$ instead of an isolated tiny eigenvalue on the extreme of the spectrum of A_2. Thus, we see from Table 4.2d that $.145 \times 10^{-7}$ appeared as an eigenvalue of T_m for A_2 as early as $m = 104$, whereas when it is considered as an eigenvalue of B_2 it doesn't appear accurately as an eigenvalue of a corresponding T_m until somewhere between $m = 1234$ and $m = 1644$. This clearly illustrates the effect of the transformation from A_2 to B_2 on any very small singular values. However, if we use the more reasonable comparison of the size of B_2 to the values of m used: $m = 2N = 1644$ and $m = 2.25N = 1850$: then the observed convergence is very reasonable since B_2 has $N = 822$ nonzero eigenvalues.

We note for this particular example that by $m = 1850 = 2.25N$ we actually have *all* of the singular values of A_2 accurately. Thus, a user could compute any portion of the spectrum that is of interest, not just a few of the smallest or largest singular values.

To demonstrate the corresponding Lanczos singular vector computations, we computed 12 singular vectors, six for the six smallest singular values and six for the six largest singular values. The orthogonality factors for each subsequent pair of singular vectors were computed and all were less than 2×10^{-9}. Furthermore, for all of the singular vectors, the error estimates $\|B_2 z - \sigma z\|$ were all less than 3×10^{-10}. The corresponding quantities

$$\|B_2 z - \sigma z\|/\text{minimal gap}$$

were all less than 10^{-6}. The corresponding error estimates for computing the eigenvectors of A_2 by treating A_2 directly as a symmetric matrix $\|A_2 x - \lambda x\|$, were all less than 2×10^{-10}, and the corresponding quantities

$$\|A_2 x - \lambda x\|/\text{minimal gap}$$

were all less than 5×10^{-6}. Note that since A_2 is symmetric the left and the right singular vectors should be identical, and in fact equal to the corresponding eigenvectors of A_2. We used this fact to compare the Lanczos generated eigenvectors of A_2 with the corresponding computed singular vectors. In all cases except for the smallest singular value σ_1 the norm of the difference between the eigenvectors and the singular vectors was less than 4×10^{-9}. For σ_1 this quantity was 1.4×10^{-6}.

The cost of this Lanczos singular value procedure is directly proportional to the degree of difficulty in resolving the desired singular values. Therefore, this procedure is not always practical. However, there are many matrices for which it can provide accurate singular values and corresponding singular vectors with a nontrivial but still reasonable amount of work. We emphasize the very important fact that this singular value Lanczos algorithm requires a minimal amount of storage, less than $5\frac{1}{2}$ vectors of length m plus whatever is required to generate the products Au and $A^T v$. For a matrix of order $N = 411$ such as A_2 with $m = 2.25N$, this is $5.5 \times m \times 8 = 41K$ bytes of storage plus whatever is needed to generate the matrix-vector products versus storage requirements of the order of N^2, $1.35M$ bytes for the LINPACK singular value subroutine. Obviously, for a given amount of computer storage, one can use this singular procedure on much larger matrices.

REFERENCES

[1] H. C. ANDREWS AND B. R. HUNT (1977), *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ.

[2] J. CULLUM AND W. E. DONATH (1974), *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace for large, sparse symmetric matrices* in Proc. 1974 IEEE Conference on Decision and Control, Phoenix, AZ, pp. 505–509.

[3] J. CULLUM (1978), *The simultaneous computation of a few of the algebraically largest and smallest eigenvalues of a large sparse symmetric matrix*, BIT, 18, pp. 265–275.

[4] J. CULLUM AND R. A. WILLOUGHBY (1979), *Lanczos and the computation in specified intervals of the spectrum of large, sparse real symmetric matrices* in Sparse Matrix Proceedings 1978, I. Duff and G. W. Stewart, eds., Society for Industrial and Applied Mathematics, Philadelphia, pp. 220–255.

[5] ——— (1980a), *Computing eigenvectors (and eigenvalues) of large, symmetric matrices using Lanczos tridiagonalization*, in Numerical Analysis Proceedings, Dundee, 1979, G. Alistar Watson, ed., Lecture Notes in Mathematics, 773, Springer-Verlag, Berlin, pp. 46–63.

[6] ——— (1980b), *Computing eigenvalues of large symmetric matrices—An implementation of a Lanczos algorithm without reorthogonalization*, Computer Programs, Research Report RC 8298, IBM, Yorktown Heights, NY.

[7] ——— (1980c), *Computing singular values and corresponding singular vectors of large matrices by Lanczos tridiagonalization*, Research Report RC 8200, IBM, Yorktown Heights, NY.

[8] ——— (1981), *Computing eigenvalues of very large symmetric matrices—An implementation of a Lanczos algorithm with no reorthogonalization*, J. Comput. Phys., 44, pp. 329–358.

[9] ——— (1982), *A Lanczos algorithm for computing eigenvectors of very large symmetric matrices*, Computer Programs, Research Report, IBM, Yorktown Heights, NY, to appear.

[10] J. CULLUM, R. A. WILLOUGHBY AND M. LAKE (1981), *A Lanczos algorithm for computing singular values and vectors of large matrices*, Research Report RC 9166, IBM, Yorktown Heights, NY.

[11] ——— (1982), *A Lanczos algorithm for computing singular values and vectors of large matrices*, Computer Programs, Research Report, IBM, Yorktown Heights, NY, to appear.

[12] B. T. SMITH, J. M. BOYLE, B. S. GARROW, Y. IKEBA, V. C. KLEMA, AND C. B. MOLER, *EISPACK Guide* (1976), *Matrix Eigensystem Routines*, Lecture Notes in Computer Science, 16, 2nd ed., Springer-Verlag, New York.

[13] G. H. GOLUB AND W. KAHAN (1965), *Calculating the singular values and pseudoinverse of a matrix*, SIAM J. Numer. Anal., 2, pp. 205–224.

[14] G. H. GOLUB, F. T. LUK AND M. L. OVERTON (1981), *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7, pp. 149–169.

[15] R. J. HANSON AND J. L. PHILLIPS (1975), *An adaptive numerical method for solving linear Fredholm integral equations of the first kind*, Numer. Math., 2, pp. 291–307.

[16] A. JENNINGS (1977), *Matrix Computation for Engineers and Scientists*, John Wiley, New York.

[17] C. LANCZOS (1961), *Linear Differential Operators*, Van Nostrand, London.

[18] C. LAWSON AND R. HANSON (1974), *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ.

[19] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH AND G. W. STEWART (1979), *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

[20] C. C. PAIGE (1972), *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Appl., 10, pp. 373–381.

[21] ——— (1974), *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11, pp. 197–209.

[22] W. STEWART (1973), *Introduction to Matrix Computations*, Academic Press, New York.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0006 $01.25/0

THE SINGULAR VALUE DECOMPOSITION IN PRODUCT FORM*

J. J. M. CUPPEN†

Abstract. A gain of about 50% in the CP-time required for the calculation of the singular value decomposition of a general matrix can be achieved by not forming the orthogonal factors explicitly, but storing the Householder reflections and Jacobi rotations that compose them. An efficient method for storing the Jacobi rotations is given. The storage required for the resulting decomposition is, for general matrices, about $1\frac{1}{2}$ times what is usual, but it is not larger than usual for matrices arising in ill-posed problems. The storage required for the decomposition of a general matrix can be reduced to what is usual by giving up part of the gain in efficiency and applying an "ultimate shift" strategy.

Key words. singular value decomposition, product form, ultimate shifts, ill-posed problems

AMS-MOS classification. 65F15,20

Introduction. The singular value decomposition (SVD) of an $m \times n$ matrix A is given by

$$A = U\Sigma V^T, \tag{1}$$

where U and V are square orthogonal matrices and Σ is an $m \times n$ diagonal matrix (i.e., a matrix having nonzero elements only in the diagonal positions (i, i) for $i = 1, 2, \cdots, \min(m, n)$). If $m > n$, we may also write

$$A = (U_1|U_2)\begin{pmatrix}\Sigma_1\\ 0\end{pmatrix} V^T = U_1\Sigma_1 V^T, \tag{2}$$

where Σ_1 is a square diagonal matrix and U_1 is an $m \times n$ column-orthogonal matrix.

The high quality algorithms presently available for the calculation of the SVD (Golub and Reinsch [1970], Lawson and Hanson [1974] (IMSL) and Dongarra et al. [1979] (LINPACK)) first transform A to bidiagonal form using Householder transformations:

$$A = U^{(1)}BV^{(1)T}, \tag{3}$$

where

$$\begin{aligned} U^{(1)} &= (I - 2u_1u_1^T)\cdots(I - 2u_nu_n^T),\\ V^{(1)} &= (I - 2v_1v_1^T)\cdots(I - 2v_{n-2}v_{n-2}^T), \end{aligned} \tag{4}$$

and then apply a QR-iteration to B during which B converges to diagonal form:

$$B^{(1)} = B, \qquad B^{(i)} = L_{l_i}^{(i)}\cdots L_1^{(i)}B^{(i-1)}R_1^{(i)}\cdots R_{r_i}^{(i)}, \quad i = 2, \cdots, k. \tag{5}$$

Here $L_j^{(i)}$ and $R_j^{(i)}$ are Jacobi rotations which have the form

$$I + (p|q)\begin{pmatrix} c-1 & s\\ -s & c-1\end{pmatrix}(p|q)^T, \qquad c^2 + s^2 = 1,$$

* Received by the editors July 9, 1981, and in revised form May 5, 1982.

† Department of Mathematics, University of Amsterdam, the Netherlands. Present address: Philips Medical Systems, MDP-QA2, NL5600MD Eindhoven, the Netherlands.

where p and q are unit basis vectors. If

$$\Sigma = B^{(k)},$$

$$U = \prod_{i=2}^{k} \left(\prod_{j=1}^{l_i} L_j^{(i)} \right) U^{(1)}, \tag{6}$$

$$V = \prod_{i=2}^{k} \left(\prod_{j=1}^{r_i} R_j^{(i)} \right) V^{(1)},$$

then (1) holds.

The proposed modification. In the algorithms mentioned above V is formed by multiplying the Householder reflections $(I - 2v_i v_i^T)$ after the bidiagonalization is completed and then applying the rotations $R_j^{(i)}$ to the product as they emerge during the QR-iteration. U is formed either the same way (except that the calculations may be restricted to $U_1^{(1)}$ and U_1) or an option is provided (cf. Lawson and Hanson [1974]) to premultiply a given vector or matrix by the transformations composing U^T.

The proposed modification is to avoid the calculations involved in forming U and V explicitly by storing the transformations which compose them. The multiplication of a vector by U, U^T, V or V^T can then be performed by applying the stored transformations in their correct order.

For the Householder transformations, determined by the vectors u_i and v_i, this approach is standard. The vectors u_i and v_i can be stored in the array originally containing the matrix A ($m \times n$ places). With respect to the Jacobi rotations some difficulties have to be solved.

Firstly, the number of locations required for storing one rotation shall be considered. In the next section it is shown that storing one real per rotation is sufficient, besides one index vector of length $n-1$ describing the convergence history of the QR-iteration.

Secondly, it is not known in advance how many rotations will be generated during the decomposition. Denoting the number of rotations by rn^2 it shall be shown that r depends on the type of matrix, the required accuracy, and the shift strategy used in the QR-iteration. In general r is about 2, but for a special class of problems, or if an ultimate shift strategy is applied, r is about 1 or smaller.

The amount of storage required for the complete decomposition thus amounts to $mn + rn^2 + n - 1$ locations, compared to $mn + n^2$ locations required for the usual approach.

Storing the Jacobi rotations. The QR-iteration that diagonalizes a bidiagonal matrix B (described in more detail by Dongarra et al. [1979]) operates in a sequence of sweeps (cf. (5)); sweep i transforms a bidiagonal matrix $B^{(i-1)}$, say, into a bidiagonal matrix $B^{(i)}$. It consists of a sequence of Jacobi rotations of the form

$$J = I + (e_{j_1} | e_{j_2}) \begin{pmatrix} c-1 & s \\ -s & c-1 \end{pmatrix} (e_{j_1} | e_{j_2})^T, \qquad c^2 + s^2 = 1. \tag{7}$$

Normally, $j_2 = j_1 + 1$ and j_1 and j_2 pass through the index range spanning the trailing diagonal submatrix of $B^{(i-1)}$ of dimension two or larger, separated from the rest of $B^{(i-1)}$ by off-diagonal elements that have become negligible in earlier sweeps. The only exception is that if diagonal element j becomes negligible during a QR-sweep, the sweep is stopped at that point and a special deflation sweep (cf. Dongarra et al.

[1979]) is applied which immediately transforms to zero the jth and $(j+1)$st off-diagonal elements. In the present approach, deflation is not applied to a negligible last diagonal element of the current submatrix.[1]

The Jacobi rotation (7) is defined by the indices j_1 and j_2, the sine s and the cosine c. Stewart [1976] has remarked that c and s can be stably recomputed from the single number z defined by $z=1$ if $c=0$, $z=s$ if $|s|<|c|$ and $z=1/c$ otherwise.

If the numbers z characterizing the consecutive Jacobi rotations are stored in a linear array (or on a file that can be read both forward and backward), then the complete transformation is determined once the index ranges of the QR- and deflation sweeps are known. This is achieved by storing, in an index array c of length $n-1$, the number of the sweep in which each off-diagonal element converges:

i) if off-diagonal element j is initially negligible then $c(j)=0$;

ii) if deflation is applied to an initially negligible diagonal element j $(j<n)$ then $c(j)=-1$;

iii) if off-diagonal element j has become negligible after initial deflation then $c(j)=1$;

iv) if deflation is applied to a negligible diagonal element j during QR-sweep i then $c(j)=-i$;

v) if off-diagonal element j has become negligible in sweep i then $c(j)=i$.

From the information stored in c it is not difficult to calculate, both in forward and backward directions, the index ranges traversed by the consecutive QR and deflation sweeps and thereby to determine the index pair (j_1, j_2) belonging to each stored rotation. An example of a convergence pattern as it occurred in an experiment is given in Table 1 below. This example, arising from a matrix which belongs to an ill-posed problem, shows some off-diagonal elements which were initially negligible (after Householder reduction), and much early convergence in the middle of the matrix. This seems to be typical for matrices arising in ill-posed problems.

TABLE 1
Convergence pattern electrocardiography problem. $m=283$, $n=50$, *precision* 10^{-6}

i	$c(i), \cdots, c(i+9)$									
1	37	36	12	35	34	32	14	31	31	30
11	29	26	28	27	26	25	24	23	23	22
21	18	21	20	19	18	17	17	16	14	14
31	15	14	13	12	8	11	10	9	9	8
41	6	7	6	5	4	3	2	0	0	

Performance. The proposed modification must be judged by the amount of work involved and the amount of storage required. The accuracy of the calculated decomposition is not affected since the numerical properties of the algorithm are not changed.

In Table 2 an outline is given of the leading terms of i) the number of storage locations required, ii) the floating point multiplication count for the decomposition and iii) the floating point multiplication count for solving a single least squares problem once the decomposition has been calculated. The first column applies to the usual approach (cf. Dongarra et al. [1979]), the second column to the modified approach. The number r is the average number of QR-sweeps per singular value. For general problems r is around 2.

[1] There would be no room in the index vector containing the convergence pattern to store this; moreover, the next regular QR-sweep will perform the required deflation.

TABLE 2

	Standard SVD	Proposed modification
i) storage locations	$mn+n^2$	$mn+rn^2$
ii) decomposition: # mults	$3mn^2-\frac{1}{3}n^3+2r(m+n)n^2$	$2mn^2-\frac{2}{3}n^3$
iii) solstep: # mults (sqrts)	$mn+n^2$	$2mn+6rn^2$ mults $+rn^2$ sqrts

An experiment was performed for a series of randomly generated matrices and for a series of matrices arising from the (ill-posed) "inverse problem of electrocardiography" (cf. Cuppen [1981b]). Tables 3 and 4 display the CP-times required by Linpack's SSVDC (Dongarra et al. [1979]) and the same routine modified as described. The average number r of QR-sweeps per singular value, more precisely defined as the number of Jacobi rotations divided by n^2, is also given for each matrix treated. The experiments were performed on a CDC Cyber 170-750, machine precision $\frac{1}{2}10^{-14}$, compiler FTN5, opt = 3.

TABLE 3
Ill-posed problem

		SSVDC req. acc. $\frac{1}{2}10^{-14}$	Modified req. acc. $\frac{1}{2}10^{-14}$		Modified req. acc. 10^{-6}	
m	n	CP-time	CP-time	r	CP-time	r
25	25	.15	.07	.85	.05	.32
50	50	.74	.30	.75	.24	.21
100	100	4.26	1.56	.64	1.31	.16
283	25	.61	.27	1.24	.24	.66
283	50	2.37	1.02	1.08	.93	.40
283	100	9.14	3.98	.85	3.44	.19

TABLE 4
Random matrix

		SSVDC req. acc. $\frac{1}{2}10^{-14}$	Modified req. acc. $\frac{1}{2}10^{-14}$		Modified req. acc. 10^{-6}	
m	n	CP-time	CP-time	r	CP-time	r
25	25	.20	.09	1.96	.08	1.48
50	50	.98	.42	1.88	.36	1.44
100	100	5.97	2.03	1.85	1.85	1.44
283	25	.75	.29	2.31	.28	1.77
283	50	2.98	1.11	2.21	1.05	1.63
283	100	12.21	4.15	2.12	4.01	1.61

The great difference between Tables 3 and 4 is caused by the phenomenon that matrices arising from ill-posed problems have a cluster of singular values at zero. This causes the matrices to split repeatedly during the QR-iteration (intermediate off-diagonal elements becoming negligible, cf. Table 1). For randomly generated matrices intermediate splitting was not observed.

For ill posed problems r is relatively small and the time spent in the QR-iteration is insignificant for both the standard and the modified approach. Consequently the

modification gives a gain in the CP-time of 50% or more, which agrees with the operation counts given in Table 2. The storage used is, due to the small r, about the same for the two approaches (equal if $r = 1$). For randomly generated matrices r is about 2 as expected. The gain in CP-time when using the modified approach varied between 50% and 65% for the very largest matrix used. Up to $1\frac{1}{2}$ times as much storage as usual was required.

The amount of work necessary for solving, using an already calculated decomposition, one or more least squares problems can be compared for the two approaches by means of the operation counts given in Table 1, for example counting one square root as six multiplications. For three representative values of r, i.e., $r \approx 2$ for general matrices, $r \approx 1$ for matrices arising from ill-posed problems and $r \approx .2$ for the same matrices in case a lower precision is required, Table 5 gives the ratio by which such a "solve step" (as opposed to calculating the matrix decomposition) is more expensive with the modified approach in columns 1 and 2 (for $m \approx n$ and $m \gg n$ respectively). In columns 3 and 4 the number of right-hand sides (to the same matrix) is given for which the joint loss in efficiency of the solve steps approximately counterbalances the gain that is achieved by calculating the decomposition with the modified approach.

TABLE 5

	Solve step modified/standard		Efficiency cross-over point	
	$m \approx n$	$m \gg n$	$m \approx n$	$m \gg n$
$r \approx 2$	13	2	$\frac{1}{3}n$	$5n$
$r \approx 1$	7	2	$\frac{1}{2}n$	$3n$
$r \approx .2$	2.2	2	$\frac{7}{8}n$	$\frac{7}{5}n$

Under certain circumstances, the larger amount of storage that is required for the modified approach, when applied to general matrices, might be prohibitive. It is then possible to write the vectors and rotations on files as soon as they are generated, but it is also possible to reduce the number of rotations required by using a more precise, but also more expensive, shift strategy in the QR-iteration. This shift strategy is described in the next section.

The ultimate shifts. It was observed by Parlett ([1980, p. 164]) that for the symmetric tridiagonal eigenproblem the use of the "ultimate shifts" strategy may decrease the number r of required QR-sweeps per eigenvalue to 1. Applied to the standard algorithms for calculating the singular value decomposition this would save a substantial part of the work required for updating U and V during the QR-phase. It means that one first calculates one or more of the singular values of B by the usual methods without accumulation of the transformations and then, using these found values to determine the shifts, performs a second QR-iteration on the original B with accumulation. This second iteration converges, at least in exact arithmetic, in one sweep for each singular value. In the usual algorithms this would save up to 3/8 of the work required as can be seen from Table 2, since r would be 1 instead of 2.

However, for the obvious implementations of this method, a simple experiment shows that, in the presence of round-off errors, the second QR-iteration does very often not converge in one sweep per singular value. To be more precise, for an unreduced bidiagonal matrix one sweep of QR, with an exact singular value determin-

ing the shift, and performed in exact arithmetic, transforms the last off-diagonal element into zero. In practice it most often occurs that even if one uses the machine number closest to some exact singular value for the shift, the last off-diagonal element is transformed into a small but nonnegligible quantity (say 10^3 or 10^4 times machine precision for matrices of order 20). This phenomenon can be understood by considering the approximate singular value which is used, as an exact singular value of a bidiagonal matrix with a perturbed last diagonal element. It then turns out (cf. Bunch et al. [1978] and Cuppen [1981a]) that a small perturbation of the singular value may correspond to a large perturbation of the diagonal element. Because only the last rotation in a *QR*-sweep is influenced by this diagonal element, it can then easily be seen that a large change in the value of the last off-diagonal element results (which would otherwise become zero). This does not contradict the stability of the *QR*-iteration since this large change is not an error in the sense of backward error analysis. Within the machine precision the resulting matrix is orthogonally similar (or is equivalent for the present transformations) to the original one, it only has not yet converged as one would have expected.

The conclusion which can be drawn is that the precision of the ultimate shifts and the precision of the arithmetic with which the *QR*-transforms are carried through on one hand as compared to the requested accuracy of the decomposition (the relative tolerance to judge convergence) on the other hand, are crucial. If the requested accuracy is distinctly less strict than the precisions mentioned, then the ultimate shift strategy may be expected to give convergence in one sweep mostly (cf. Tables 6 and 7).

However, the accuracy of an ultimate shift based on a precalculated singular value is also influenced by the cumulative effect of rounding errors in the preceding *QR*-sweeps. Therefore when an initially calculated singular value is to be used for an ultimate shift it is best first updated in an extra *QR*-iteration on a copy of the matrix as it then stands.

In many practical situations the accuracy requested will be less than the machine precision, for example, if the matrix to be treated is known only in a low precision. However, if full precision is required we may either allow for extra iterations to be made which will cost extra space for storing the rotations, or use extended precision arithmetic for the calculation of ultimate shifts and the *QR*-sweep (not for the stored

TABLE 6
Ill-posed problem

			Modified method					
			single prec. ult. shift				ext. prec. u.s.	
		SSVDC req. acc. $\frac{1}{2}10^{-14}$	req. acc. $\frac{1}{2}10^{-14}$		req. acc. 10^{-6}		req. acc. $\frac{1}{2}10^{-14}$	
m	n	CP-time	CP-time	r	CP-time	r	CP-time	r
25	25	.15	.15	.86	.10	.31	.19	.74
50	50	.74	.66	.75	.47	.20	.88	.64
100	100	4.26	3.38	.65	2.20	.16	5.16	.55
283	25	.61	.35	1.24	.33	.64	.40	.92
283	50	2.37	1.46	1.08	1.41	.40	1.76	.99
283	100	9.14	6.97	.85	4.90	.20	6.45	.76

rotations!). This will introduce overhead costs, but the computations involved are of the order of n^2 in total as compared to n^3 for the complete SVD.

The results of a series of experiments with the matrices also used above are given in Tables 6 and 7. These show that ultimate shifts indeed reduce the amount of storage required, but that they involve a great loss in efficiency. Ultimate shifts should only be used in case of large matrices which are not of the type as given by the examples arising from the ill-posed problem.

TABLE 7
Random problem

			Modified method					
			single prec. ult. shift				ext. prec. u.s.	
		SSVDC req. acc. $\frac{1}{2}10^{-14}$	req. acc. $\frac{1}{2}10^{-14}$		req. acc. 10^{-6}		req. acc. $\frac{1}{2}10^{-14}$	
m	n	CP-time	CP-time	r	CP-time	r	CP-time	r
25	25	.20	.17	1.31	.16	.96	.25	.96
50	50	.98	.70	1.33	.68	.98	.98	.98
100	100	5.97	3.17	1.31	3.12	.99	4.31	.99
283	25	.75	.37	1.40	.36	.96	.43	.96
283	50	2.98	1.44	1.44	1.40	.98	1.72	.98
283	100	12.21	5.51	1.45	5.33	.99	6.55	.99

Discussion. The numerical experiments show that the proposed modification can give a substantial gain in CP-time when computing the singular value decomposition.

The method can very well work with files if storage in core is so restrictive or expensive that the usual approach cannot be used.

The approach is also applicable to the eigenvalue decomposition of symmetric matrices.

REFERENCES

J. R. BUNCH, C. P. NIELSEN AND D. C. SORENSEN, *Rank one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

J. J. M. CUPPEN [1981a], *A divide and conquer method for the symmetric eigenproblem*, Numer. Math., 31 (1981), pp. 177–195.

———, [1981b], *Calculating the isochrones of ventricular depolarization*, submitted for publication.

J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

———, *The economical storage of plane rotations*, Numer. Math., 25 (1976), pp. 137–138.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0007 $01.25/0

ORTHOGONAL ROTATION TO A PARTIALLY SPECIFIED TARGET*

FRANKLIN T. LUK†

Abstract. This paper addresses the problem of finding an orthogonal transformation of an arbitrary factor solution that will lead to a least squares fit of a partially specified target matrix. An iterative computing procedure is presented.

Key words. orthogonal rotations, factor analysis, least squares, partially specified target, Procrustes problem

1. Introduction. We consider the problem of orthogonally rotating a given factor matrix to a least squares fit to a partially prescribed factor pattern. The special case where all the target elements are zero has been solved by Lawley and Maxwell [5], albeit not in the least squares sense. The general case remains an *open* problem and Browne [2] presents perhaps the only attempt at computing an *approximate* solution. He constructs an orthogonal transformation as a sequence of plane rotations. The rotations are determined by Newton's method which requires very good initial guesses. In this paper we shall show that these planar rotations are expressible as solutions to linear least squares problems with an equality constraint. Effective numerical techniques are therefore applicable, and a procedure based on the singular value decomposition (cf. Golub [4]) will be presented. We shall also give an example for which Browne's method fails to compute a least squares fit to the target.

2. The algorithm. Our problem is frequently referred to as a Procrustes problem, a term borrowed from Greek mythology in which a highwayman named Procrustes is supposed to have made all his victims fit his bed, cruelly stretching the short ones and lopping off the tall ones. Mathematically, our problem is to determine an $m \times m$ orthogonal matrix Q that will transform a given $n \times m$ factor matrix A to best fit a partially prescribed $n \times m$ target factor pattern B in the least squares sense. Let $I(j)$ represent the set of row indices corresponding to specified elements of column j of B. If $F = AQ$, the criterion to be minimized is

$$\tau \equiv \sum_{j=1}^{m} \left\{ \sum_{i \in I(j)} (f_{ij} - b_{ij})^2 \right\}, \tag{2.1}$$

where $F = (f_{ij})$ and $B = (b_{ij})$.

Following Browne [2] we shall approximate Q by a product of plane rotations. Designate by A^c the current rotated factor matrix at an intermediate stage, and let $A^c = (a_{ij}^c) = (a_1^c, a_2^c, \cdots, a_m^c)$. We now consider a rotation over the plane defined by the ith and jth factors:

$$(a_i^c, a_j^c) \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{2.2}$$

Suppose that $S(i) = |I(i)|$ and $S(j) = |I(j)|$. Let $i(1), i(2), \cdots, i(S(i))$ and $j(1), j(2), \cdots, j(S(j))$ denote the row indices of the specified elements of columns i

* Received by the editors September 28, 1981, and in revised form June 18, 1982. This research was supported in part by the U.S. Army Research Office under grant DAAG 29-79-C0124.

† Department of Computer Science, Cornell University, Ithaca, New York 14853.

and j, respectively, of the target B. Our goal is to determine a rotational angle θ to minimize the euclidean length of the vector

$$\begin{bmatrix} a^c_{i(1),i} & a^c_{i(1),j} \\ a^c_{i(2),i} & a^c_{i(2),j} \\ \vdots & \vdots \\ a^c_{i(S(i)),i} & a^c_{i(S(i)),j} \\ a^c_{j(1),j} & -a^c_{j(1),i} \\ a^c_{j(2),j} & -a^c_{j(2),i} \\ \vdots & \vdots \\ a^c_{j(S(j)),j} & -a^c_{j(S(j)),i} \end{bmatrix} \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} - \begin{bmatrix} b_{i(1),i} \\ b_{i(2),i} \\ \vdots \\ b_{i(S(i)),i} \\ b_{j(1),j} \\ b_{j(2),j} \\ \vdots \\ b_{j(S(j)),j} \end{bmatrix}. \tag{2.3}$$

In other words, we want to solve the constrained least squares problem:

$$\|C\mathbf{x}-\mathbf{d}\|_2 = \text{minimum}, \tag{2.4}$$

subject to

$$\|\mathbf{x}\|_2 = 1. \tag{2.5}$$

Effective algorithms for solving this problem are well known. We shall describe one based on the singular value decomposition (cf. Golub [4]).

The singular value decomposition of an $n \times 2$ matrix C is readily computable. First, two Householder transformations, say H_1 and H_2, can be applied to reduce the matrix to upper triangular form, i.e.,

$$H_2H_1C = \begin{bmatrix} p & q \\ 0 & r \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \equiv \begin{bmatrix} R \\ \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.6}$$

We are interested in orthogonalizing the two rows of R. As this problem is trivial if $pqr = 0$, we assume that all three elements are nonzero. As in Luk [6] we construct an orthogonal rotation Q such that Q^TR has orthogonal rows. Let us define

$$\alpha \equiv 2qr, \quad \beta \equiv p^2+q^2-r^2, \quad \gamma \equiv (\alpha^2+\beta^2)^{1/2}. \tag{2.7}$$

If $\beta > 0$, we compute

$$c = \left(\frac{\gamma+\beta}{2\gamma}\right)^{1/2}, \qquad s = \frac{\alpha}{2\gamma c}, \tag{2.8}$$

else we compute

$$s = \left(\frac{\gamma-\beta}{2\gamma}\right)^{1/2}, \qquad c = \frac{\alpha}{2\gamma s}. \tag{2.9}$$

Defining

$$Q \equiv \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \tag{2.10}$$

we have

$$Q^TR = \begin{pmatrix} cp & cq+sr \\ -sp & -sq+cr \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}\begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix} \equiv \Sigma V^T, \tag{2.11}$$

where $\sigma_1 \geqq \sigma_2 > 0$ follows from the condition $pqr \neq 0$, and

$$\sigma_1 = [c^2p^2 + (cq + sr)^2]^{1/2}, \qquad \sigma_2 = [s^2p^2 + (-sq + cr)^2]^{1/2},$$

$$v_{11} = v_{22} = \frac{cp}{\sigma_1}, \qquad v_{12} = -v_{21} = -\frac{sp}{\sigma_2}.$$

The singular value decomposition of C is then given by

$$C = U\Sigma V^T, \tag{2.12}$$

where

$$U = H_1 H_2 \begin{pmatrix} Q \\ 0 \end{pmatrix}.$$

We shall use the notations

$$U = (\mathbf{u}_1, \mathbf{u}_2) \tag{2.13}$$

and

$$V = (\mathbf{v}_1, \mathbf{v}_2). \tag{2.14}$$

Let us now consider the constrained least squares problem (2.4) and (2.5). For a zero matrix C, we choose the solution $\mathbf{x} = (1, 0)^T$, i.e. the null rotation. If $C \neq 0$ but $\mathbf{d} = \mathbf{0}$, then the solution is $\mathbf{x} = \mathbf{v}_2$ (we let $\mathbf{v}_2 = (1, 0)^T$ if $\sigma_1 = \sigma_2$). Henceforth we shall assume also that $\mathbf{d} \neq \mathbf{0}$. If C is of rank one, then

$$C^+\mathbf{d} = (\mathbf{u}_1^T\mathbf{d}/\sigma_1)\mathbf{v}_1 \equiv \xi\mathbf{v}_1. \tag{2.15}$$

For $|\xi| < 1$, we let

$$\mathbf{x} = \xi\mathbf{v}_1 + (1 - \xi^2)^{1/2}\mathbf{v}_2, \tag{2.16}$$

else we let

$$\mathbf{x} = \text{sign}\,(\xi)\mathbf{v}_1. \tag{2.17}$$

If C is of full rank, we consider the matrix equation

$$(C^TC + \lambda I)\mathbf{x} = C^T\mathbf{d}, \tag{2.18}$$

where λ is a Lagrange multiplier. The proper value for λ is a solution to the nonlinear equation

$$\det[(\Sigma^2 + \lambda I)^2 - \mathbf{g}\mathbf{g}^T] = 0, \tag{2.19}$$

where

$$\mathbf{g} = \Sigma U^T\mathbf{d} \equiv \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}. \tag{2.20}$$

Although (2.19) is quartic, it can be shown (cf. Forsythe and Golub [3]) that we need the largest real root λ_1. Furthermore, if

$$\|C^+\mathbf{d}\|_2 > 1,$$

then λ_1 is the unique root in the open interval $(0, \|\mathbf{g}\|_2)$; otherwise, except for a singular case, λ_1 is the unique root in the semiopen interval $(-\sigma_2^2, 0]$. There are many methods with assured convergence for this zero-finding problem. For example, we may use the

method in Brent [1, Chap. 4] to compute λ_1. The solution vector is then given by

$$\mathbf{x} = V(\Sigma^2 + \lambda_1 I)^{-1}\mathbf{g}. \tag{2.21}$$

The singularity occurs when $\sigma_1 > \sigma_2$, $g_2 = 0$ and $|g_1| \leqq (\sigma_1^2 - \sigma_2^2)$. We then have $\lambda_1 = -\sigma_2^2$ and

$$\mathbf{x} = \left(\frac{g_1}{\sigma_1^2 - \sigma_2^2}\right)\mathbf{v}_1 + \eta\mathbf{v}_2, \tag{2.22}$$

where η is chosen such that $\|\mathbf{x}\|_2 = 1$.

Browne [2] proposes that the ith and jth columns of A^c be considered for reflection before the rotation. For example, the ith column would be reflected ($a_{ki}^c := -a_{ki}^c$, $k = 1, \cdots, n$) if

$$\sum_{k \in I(i)} a_{ki}^c b_{ki} < 0. \tag{2.23}$$

The additional work for (2.23) is minimal, for the result can be used in the computation of the vector $\mathbf{g}$ in (2.20). The value of reflection is apparent from this simple example:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ x & x \end{pmatrix}.$$

This procedure of reflections and plane rotations is applied to the factor matrix in a systematic order. Transformations are made on factors 1 with 2, 1 with 3, $\cdots$, 1 with m, 2 with 3, $\cdots$, $(m-1)$ with m to constitute a cycle. The cycle is repeated until either there is one consisting of all null transformations or the criterion τ of (2.1) fails to decrease.

3. Discussion. In [2] Browne also offers some guidelines on the reordering of columns of A^c. His procedure is expensive and may not be worthwhile. Consider this example in [2]: the given factor matrix is

$$A = \begin{pmatrix} .664 & .322 & -.075 \\ .688 & .248 & .192 \\ .492 & .304 & .224 \\ .837 & -.291 & .037 \\ .705 & -.314 & .155 \\ .820 & -.377 & -.104 \\ .661 & .397 & .077 \\ .457 & .294 & .488 \\ .765 & .428 & .009 \end{pmatrix},$$

and the partially prescribed target is

$$B = \begin{pmatrix} x & 0 & x \\ x & 0 & 0 \\ x & 0 & 0 \\ x & x & x \\ x & x & 0 \\ x & x & x \\ .7 & x & x \\ 0 & x & x \\ .7 & x & x \end{pmatrix}.$$

We have performed the transformations with no column reordering. A rotation is considered null if $|\sin\theta| \leqq .001$. After six cycles, our procedure produces the final rotated matrix

$$AQ = \begin{pmatrix} .611 & -.023 & -.420 \\ .734 & -.055 & -.173 \\ .607 & .086 & -.093 \\ .608 & -.622 & -.176 \\ .549 & -.564 & -.017 \\ .499 & -.713 & -.261 \\ .705 & .069 & -.314 \\ .235 & -.024 & -.691 \\ .768 & .039 & -.422 \end{pmatrix}.$$

The criterion τ of (2.1) decreases as follows:

Cycle	Criterion
0	.583264
1	.182168
2	.113184
3	.110034
4	.109858
5	.109852
6	.109852

Our results thus compare favorably with those given in [2].

Finally, we present one example for which the plane rotations fail to minimize the criterion τ. Let

$$A = \begin{pmatrix} 3 & -2 & -2 \\ -2 & 3 & -2 \\ -2 & -2 & 3 \\ -2 & -2 & -2 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ x & x & x \end{pmatrix},$$

so that τ equals 36. Browne's procedure will stop because only null rotations are generated. However, the Householder matrix

$$Q = I - 2\mathbf{e}\mathbf{e}^T,$$

where $\mathbf{e}^T = (1, 1, 1)/\sqrt{3}$, will produce the transformed matrix

$$AQ = \begin{pmatrix} \frac{11}{3} & -\frac{4}{3} & -\frac{4}{3} \\ -\frac{4}{3} & \frac{11}{3} & -\frac{4}{3} \\ -\frac{4}{3} & -\frac{4}{3} & \frac{11}{3} \\ 2 & 2 & 2 \end{pmatrix},$$

for which τ equals 32.

Acknowledgment. The author is deeply grateful to C. F. Van Loan for many valuable suggestions, including the nonconvergence example.

REFERENCES

[1] R. P. BRENT, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[2] M. W. BROWNE, *Orthogonal rotation to a partially specified target*, British J. Math. Statist. Psychol., 25 (1972), pp. 115–120.

[3] G. E. FORSYTHE AND G. H. GOLUB, *On the stationary values of a second-degree polynomial on the unit sphere*, J. Soc. Indust. Appl. Math., 13 (1965), pp. 1050–1068.

[4] G. H. GOLUB, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[5] D. N. LAWLEY AND A. E. MAXWELL, *Factor transformation methods*, British J. Statist. Psychol., 17 (1964), pp. 97–103.

[6] F. T. LUK, *Computing the singular-value decomposition on the ILLIAC IV*, ACM Trans. Math. Software, 6 (1980), pp. 524–539.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0008 $1.25/0

A HYBRID ASYMPTOTIC-FINITE ELEMENT METHOD FOR STIFF TWO-POINT BOUNDARY VALUE PROBLEMS*

R. C. Y. CHIN† AND R. KRASNY†‡

Abstract. An accurate and efficient numerical method has been developed for a nonlinear stiff second order two-point boundary value problem. The scheme combines asymptotic methods with the usual solution techniques for two-point boundary value problems. A new modification of Newton's method or quasilinearization is used to reduce the nonlinear problem to a sequence of linear problems. The resultant linear problem is solved by patching local solutions at the knots or equivalently by projecting onto an affine subset constructed from asymptotic expansions. In this way, boundary layers are naturally incorporated into the approximation. An adaptive mesh is employed to achieve an error of $O(1/N^2)+O(\sqrt{\varepsilon})$. Here, N is the number of intervals and $\varepsilon \ll 1$ is the singular perturbation parameter. Numerical computations are presented.

Key words. projection method, quasilinearization, asymptotic expansions, boundary layers, singular perturbations, two-point boundary value problems, γ-elliptic splines

1. Introduction.

We treat the following stiff boundary value problem:

$$\begin{aligned} &\varepsilon y'' = f(x, y), \qquad a < x < b, \quad f(x, y) \in C^2[(a, b) \times R], \\ &f_y(x, y) > 0, \qquad 0 < \varepsilon \ll 1, \end{aligned} \tag{1.1}$$

with boundary conditions:

$$a_0 y(a) - a_1 y'(a) = \alpha, \tag{1.2}$$

$$b_0 y(b) + b_1 y'(b) = \beta. \tag{1.3}$$

Under the assumptions that $a_0 a_1 > 0$, $b_0 b_1 > 0$ and $|a_0| + |b_0| > 0$ Keller [26] and Bernfield and Lakshmikantham [8] have shown that the boundary value problem (BVP) has a unique solution. Using singular perturbation theory, O'Malley [31], [32] has proved that boundary layers occur at either or both boundaries depending on whether the solution of the reduced problem:

$$f(x, z(x)) = 0$$

satisfies the boundary conditions (1.2) and (1.3). In particular, Fife [18] and Bris [12] show that for a Dirichlet problem there are boundary layers at both boundaries.

This class of stiff or singular perturbation boundary value problems appears in many physical applications, for example, the confinement of a plasma column by radiation pressure (Troesch [42]), the theory of gas porous electrodes (Gidaspow et al. [22], Markin et al. [28]), the performance of catalytic pellets (Aris [3]) and in geophysical fluid dynamics (Carrier [14]).

Our method combines quasilinearization (Newton's method) in function space, asymptotic expansions and patching of local solutions. Quasilinearization is applied to reduce the nonlinear problem to a sequence of linear problems (Ascher, Christiansen and Russell [4], deBoor and Swartz [10], Russell and Shampine [40]). This is achieved in our scheme by interpolating $f(x, y)$ by a piecewise linear function of y for fixed x. This step differs from the usual quasilinearization in that the resulting BVP has

* Received by the editors January 4, 1981, and in revised form June 7, 1982. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

† University of California, Lawrence Livermore National Laboratory, Livermore, California 94550.
‡ Graduate student, Department of Mathematics, University of California, Berkeley, California 94720.

discontinuous coefficients. We use asymptotic methods to construct the local solutions in each subinterval. This should be compared with other asymptotic methods which also seek to capture the essential behavior of the differential equation (Flaherty and Mathon [19], Flaherty and O'Malley [20], deGroen and Hemker [23], deGroen [24], Hemker [25], Yarmish [45]). The local solutions are patched together by requiring that the computed solution and its derivative be continuous at the knots (Rose [38], Shampine [42]). This assures the computed solution is globally twice differentiable in an asymptotic sense.

This patching process for solving the linearized BVP is equivalent to a modified Galerkin method. This involves projection onto an affine subset constructed from the γ-elliptic splines of Schultz [41]. The computed solution, however, is based on γ_ε-splines which are asymptotic approximations of these. The γ_ε-splines become more accurate as $\varepsilon \to 0$ with no corresponding increase in calculational expense. This property is inherent in methods based on asymptotic expansions and is absent in finite difference and finite element methods based on polynomials (Abrahamsson [1], Berger et al., [6], Kellogg and Tsan [27], Osher [34]).

Another novel feature of our method is the mesh selection strategy. The interpolation error is proportional to $f_{yy}(\Delta y)^2$ which leads to a criterion for how the y-mesh should be chosen. This contrasts with the usual methods which specify the x-mesh. We consider both a uniform y-mesh and a y-mesh resulting from equally distributing the interpolation error (deBoor [9]). In either case, an iteration is introduced to construct the numerical solution and the grid structure. The resulting error depends only on the number of intervals $O(1/N^2)$ rather than their size. This facet of the method appears to be new.

Our study has been influenced by Pruess' method of solving linear boundary value problems by approximating the coefficients [36]. The essence of Pruess' method lies in the ability of the approximation to capture the essential behavior of the solution. This is because the coefficients of the DE can be well approximated locally by polynomials while the solution cannot. In this light, our method may be viewed as solving nonlinear boundary value problems by approximating terms of the differential equation.

A detailed numerical analysis of the error and convergence properties will be presented in a forthcoming paper. However, some of the preliminary error analysis is presented to motivate the adaptive mesh selection. In § 2, we describe the numerical algorithm. In § 3, we discuss the error introduced by quasilinearization. In § 4, we present the adaptive mesh technique and the solution of the algebraic system. In § 5, we describe the construction of the γ-elliptic splines using asymptotics. In § 6, computations are presented. Finally, conclusions and generalizations are presented in § 7.

2. Derivation of the numerical method. We begin by replacing $f(x, y)$ by $I_f(x, y)$, its piecewise linear interpolant with respect to a y-mesh $(y_i,\ i = 0, 1, \cdots, N)$ for fixed x:

$$\begin{aligned} I_f(x, y) &= f(x, y_{i-1}) + (y - y_{i-1})[y_{i-1}, y_i]f, \qquad y_{i-1} < y < y_i, \\ &= g_{0_i}(x; y_{i-1}, y_i) + g_{1_i}(x; y_{i-1}, y_i)y, \qquad i = 1, \cdots, N, \end{aligned}$$

where

$$[y_{i-1}, y_i]f = (f(x, y_i) - f(x, y_{i-1}))/(y_i - y_{i-1}).$$

We will solve the linear BVP $\varepsilon y'' = I_f(x, y)$ together with boundary conditions (1.2)–(1.3) by a "patching" or multiple shooting process. With respect to an x-mesh,

$a = x_0 < x_1 < \cdots < x_N = b$, define the following elliptic differential operators, $E_i y = -\varepsilon y'' + g_{1_i} y$, for $x_{i-1} < x < x_i$. On each interval we consider the linear BVP:

$$E_i y = -g_{0_i} \tag{2.1}$$

with boundary conditions given by

$$\begin{aligned} &a_0 y(a) - a_1 y'(a) = \alpha, \qquad y(x_1) = y_1, \quad i = 1, \\ &y(x_{i-1}) = y_{i-1}, \qquad y(x_i) = y_i \quad \text{for } i = 2, \cdots, N-1, \\ &b_0 y(b) + b_1 y'(b) = \beta, \qquad y(x_{N-1}) = y_{N-1} \quad \text{for } i = N. \end{aligned} \tag{2.2}$$

It will sometimes be convenient to write these equations collectively as

$$Ey = -\varepsilon y'' + g_1 y = -g_0. \tag{2.3}$$

Let $u_i(x)$, $v_i(x)$ be the basis for the null space for E_i normalized so that

$$u_i(x_{i-1}) = v_i(x_i) = 0 \quad \text{and} \quad u_i(x_i) = v_i(x_{i-1}) = 1.$$

Let y_{p_i} be the particular solution of the nonhomogeneous BVP

$$\begin{aligned} &E_i y_{p_i} = -g_{0_i}, \\ &y_{p_i}(x_{i-1}) = y_{p_i}(x_i) = 0, \qquad x_{i-1} < x < x_i. \end{aligned} \tag{2.4}$$

Then the solution of (2.1) is given by:

$$y(x; y_{i-1}, y_i) = y_i u_i(x) + y_{i-1} v_i(x) + y_{p_i}(x), \qquad x_{i-1} < x < x_i. \tag{2.5}$$

There are $2N$ degrees of freedom from $\{y_i\}$ $i = 0, 1, \cdots, N$ and $\{x\}_k$ $k = 1, 2, \cdots, N-1$. By requiring that the solution (2.5) satisfies the boundary conditions (2.2) and is continuously differentiable at the knots x_i, $i = 1, \cdots, N-1$, we obtain a nonlinear algebraic system relating the x_i and y_i. With the remaining $N-1$ degrees of freedom we may prescribe $N-1$ additional relations to uniquely determine the $2N$ quantities. This prescription *should* achieve the goal of improving efficiency and accuracy and will be discussed later.

The condition that $y \in C^1[a, b]$ is equivalent to equality of the derivatives from adjacent intervals at the knots:

$$y'(x_i^-; y_{i-1}, y_i) = y'(x_i^+; y_i, y_{i+1}).$$

Differentiating (2.5) we obtain:

$$\begin{aligned} &v_i'(x_i) y_{i-1} + [u_i'(x_i) - v_{i+1}'(x_i)] y_i - u_{i+1}'(x_i) y_{i+1} \\ &\quad = y_{p_{i+1}}'(x_i) - y_{p_i}'(x_i), \qquad i = 1, 2, \cdots, N-1. \end{aligned} \tag{2.6}$$

At the boundary nodes $x_0 = a$ and $x_N = b$ we require that the boundary conditions (1.2)–(1.3) be satisfied. This gives:

$$\begin{aligned} &[a_0 - a_1 v_1'(a)] y_0 - a_1 u_1'(a) y_1 = \alpha + a_1 y_{p_1}'(a), \\ &v_{N-1}'(b) y_{N-1} + [b_0 + b_1 u_N'(b)] = \beta - b_1 y_{p_N}'(b). \end{aligned} \tag{2.7}$$

Equations (2.6)–(2.7) can be written in the form:

$$A(x, y) y = h(x, y), \tag{2.8}$$

where $y = (y_0, y_1, \cdots, y_N)^T$, $x = (x_0, x_1, \cdots, x_N)^T$.

Here h is an $N+1$ component vector and A is a symmetric diagonally dominant tridiagonal matrix of dimension $N+1$. A has positive diagonal and negative off-diagonal elements and accordingly is a Stieltjes matrix. These properties follow from

$f_y > 0$ and from the construction of the basis functions u_i and v_i. Symmetry of A may be demonstrated by noting that the Wronskian of u, v is constant in each interval $x_{i-1} < x < x_i$. Moreover, $y \in C^2[a, b]$ since $I_f(x, y) \in C[a, b]$. It should also be noted that the linearized problem (2.2)–(2.3) is solved exactly by the above patching procedure.

The particular solution y_{p_i} may be expressed by the usual variation of parameters formula since the Green's function for the Dirichlet problem for (2.1) exists. Thus we have:

$$y_{p_i} = \frac{v_i}{c} \int_{x_{i-1}}^{x} u_i g_{0_i}\, d\xi + \frac{u_i}{c} \int_{x}^{x_i} v_i g_{0_i}\, d\xi \tag{2.9}$$

where $c = \varepsilon$ Wronskian (u_i, v_i).

We now describe how (2.8) can be obtained by a Galerkin procedure. Let $S(E, \{x_i\})$ be the space of γ-elliptic splines generated by the basis w_i:

$$w_0(x) = \begin{cases} v_1(x), & x \in (x_0, x_1), \\ 0 & \text{otherwise}, \end{cases}$$

$$w_i(x) = \begin{cases} u_i(x), & x \in (x_{i-1}, x_i), \\ v_i(x), & x \in (x_i, x_{i+1}), \\ 0 & \text{otherwise}, \end{cases} \quad i = 1, 2, \cdots, N-1,$$

$$w_N(x) = \begin{cases} u_N(x), & x \in (x_{N-1}, x_N), \\ 0 & \text{otherwise}. \end{cases}$$

Schultz [41] considered the problem of interpolating in spaces generated by such splines called γ-elliptic splines, where γ is the smallest eigenvalue of the operator E. Note that our splines depend on both the x and y meshes.

Consider the problem of finding an element y contained in the affine subset $[y_p] \oplus S(E, \{x_i\})$, which satisfies (2.1) along with the boundary conditions (2.2). Let

$$a(u, v) = \frac{1}{\varepsilon} \int_a^b (\varepsilon u'v' + g_1 uv)\, dx, \qquad (u, v) = \frac{1}{\varepsilon} \int_a^b uv\, dx.$$

A discretized weak form of this problem is: Find $y \in [y_p] \oplus S(E, \{x_i\})$ such that

$$a(y, w) = (g_0, w) + (-a_0 y(a) + \alpha) w(a)/a_1 - (b_0 y(b) + \beta) w(b)/b_1 \tag{2.10}$$

for all $w \in S(E, \{x_i\})$.

In case a_1 or $b_1 = 0$, we have Dirichlet boundary conditions and another appropriate weak form (Aubin [5]) holds. In any case equations (2.10) form a nonlinear system which is identical to (2.6)–(2.7). In verifying this, the following formulas are useful:

$$a(y, w_i) = y_{i+1} w_i'(x_{i+1}^-) + y_i (w_i'(x_i^-) - w_i'(x_i^+)) - y_{i-1} w_i'(x_{i-1}^+),$$

$$(g_0, w_i) = y_{p_{i+1}}'(x_i^+) - y_{p_i}'(x_i^-) \quad \text{for } i = 1, \cdots, N-1$$

with similar formulas for w_0, w_N.

Thus the matrix elements in (2.8) are recognized as simply $a(w_i, w_j)$, and it follows that the matrix is symmetric, positive definite and tridiagonal.

The usual Galerkin method projects y onto a subspace of $C^0[a, b]$. We call our procedure a modified Galerkin method because it projects y onto an affine subset, i.e., a translated linear subspace. In other words, we can view our modified Galerkin

method as applying the usual Galerkin method to $y - y_p$. It is striking that since $a(y_p, w_i) = 0$, both Galerkin and modified Galerkin give the same set of discretized equations. The difference is that projecting onto the affine subset solves the linearized problem (2.3) exactly.

The equivalence of the "patching" procedure and the modified Galerkin method is another example of the interdependence of projection methods for two-point BVP as discussed by Reddien [36].

3. Discussion of the error due to quasilinearization. In this section we briefly discuss the error in order to motivate the mesh selection procedure. A detailed numerical analysis of the method will be presented in a forthcoming paper. For present purposes, a few of the preliminary results will be discussed.

For definiteness, let y denote the exact solution and let it satisfy $\varepsilon y'' = f(x, y)$ together with boundary conditions (1.2)–(1.3). Our method makes two main approximations to the original nonlinear BVP in order to arrive at the computed solution. We list these approximations along with their defining properties:

1. Let z be the γ-elliptic spline solution and satisfy

$$\varepsilon z'' = I_f(x, z; z_i) \text{ together with boundary conditions,} \tag{3.1}$$

where $I_f(x, z; z_i) = g_0(x; z_i) + g_1(x, z_i)z$ is the linear interpolation of f with knots z_i.

2. Let z_ε be the actual computed solution, which is an asymptotic approximation to z as will be explained in § 5.

As seen in § 2, an equivalent formulation of (3.1) is

$$\begin{aligned} &A(x, z)z = h(x, z), \qquad x = \{x_i\}^T, \quad z = \{z_i\}^T, \\ &z(x) = z_{i-1}w_{i-1}(x) + z_i w_i(x) + z_{p_i}(x) \quad \text{on } [x_{i-1}, x_i], \end{aligned} \tag{3.2}$$

where w_i is the normalized γ-elliptic spline basis function at x_i and z_{p_i} is defined as in (2.4).

We will show that $|y - z| < c/N^2$. This is the error from the linear interpolation. It follows from asymptotic error bounds (Olver [29], [30]) that $z = z_\varepsilon(1 + \zeta)$, where $|\zeta| = O(\sqrt{\varepsilon})$ as $\varepsilon \to 0$.

Using these two bounds we conclude that the pointwise error in the method can be bounded by a sum of two terms:

$$|y - z_\varepsilon| < |y - z| + |z_\varepsilon||\zeta|;$$

thus we have $\sup |y - z_\varepsilon| < O(1/N^2) + O(\sqrt{\varepsilon})$. Two important special cases deserve to be singled out:

(1) For an autonomous equation $\varepsilon y'' = f(y)$, our method solves the linearized problem exactly. Thus, $z = z_\varepsilon$, and the second term in the error bound is not present. We expect our method to be competitive in this case even for ε not small.

(2) For a linear BVP the first term in the error bound is not present. In this case our method simply reduces to the Liouville–Green or WKB asymptotic solution (Olver [29]).

We will discuss the first term now and reserve a discussion of the second term until the asymptotics have been introduced.

Define $\phi = y - z$. Subtracting the two equations for y and z we get

$$\varepsilon\phi'' = f(x, y) - I_f(x, z; z_i).$$

The right-hand side can be written

$$f(x, y)-I_f(x, z; z_i)=f(x, y)-f(x, z)+f(x, z)-I_f(x, z; z_i)$$
$$=f_y(x, y^*)(y-z)+f(x, z)-I_f(x, z; z_i)$$

for some y^* between y and z. Thus ϕ satisfies the singular perturbation problem with homogeneous boundary conditions,

$$\varepsilon\phi''-f_y(x, y^*)\phi=f(x, z)-I_f(x, z; z_i).$$

The driving term on the right-hand side is simply the error due to the linear interpolation and can be bounded by a standard result of approximation theory. For a Dirichlet problem, we can use the maximum principle (Dorr, Parter and Shampine [16], Protter and Weinberger [35]) to obtain the following a posteriori error bound.

LEMMA 1. *Let*

$$D=[a, b]\times \textit{range of exact solution}, \quad m^2=\min f_y \quad \textit{on } D,$$
$$M=\max|f_{yy}(x, z(x))| \quad \text{on } [a, b], \qquad \Delta z=\max|z_i-z_{i-1}|,$$
$$\kappa(x)=1-\left\{\sinh\left[\frac{m(x-a)}{\sqrt{\varepsilon}}\right]+\sinh\left[\frac{m(b-x)}{\sqrt{\varepsilon}}\right]\right\}\Big/\sinh\left[\frac{m(b-a)}{\sqrt{\varepsilon}}\right].$$

Then

$$|\phi(x)|<(M/8m)\Delta z^2\kappa(x).$$

LEMMA 2.

$$\textit{If } f_{yy}(x, z(x))>0 \textit{ on } [a, b] \textit{ then } \phi>0.$$
$$\textit{If } f_{yy}(x, z(x))<0 \textit{ on } [a, b] \textit{ then } \phi<0.$$

The comparison function $\kappa(x)$ is the solution of $\varepsilon\kappa''-m^2\kappa=-m^2$ with homogeneous Dirichlet boundary condition. It is strictly less than 1 and has boundary layers at $x=a, b$. Thus we expect a smaller error closer to the boundary. With further assumptions on the boundary condition parameters a similar bound holds for the Robin problem. Here, the function $\kappa(x)$ must be modified to account for Robin boundary conditions (Friedman [21], Stakgold [43]).

The form of the error bound suggests the following mesh selection strategy. For a given number of mesh intervals N, uniformly spaced z_i points will produce an error bound of the form c/N^2. Another possibility is to choose the z_i points to equally distribute the interpolation error in the sense of deBoor [9], which also gives a c/N^2 bound. In either case the constant c depnds on f_y, f_{yy} but is independent of ε.

In constructing the equidistributed interpolation error mesh, it is necessary to apply an extension of de Boor's theorem [9] to the present case of interpolating $f(x, y)$ with respect to y for each x. This is a trivial extension since the piecewise linear approximation holds for any x. Thus the relation:

$$\int_{z_0}^{z_i(x)}|f_{yy}(x, u)|^{1/2}\,d\xi=\frac{i}{N}\int_{z_0}^{z_N}|f_{yy}(x, \xi)|^{1/2}\,d\xi, \quad i=1, 2, \cdots, N-1 \tag{3.3}$$

generates a family of curves with an equidistributed interpolation error. The curves are separated, and for any x, we have

$$z_0<z_1(x)<z_2(x)<\cdots<z_{N-1}(x)<z_N.$$

The choice of whether to use a uniformly spaced z-mesh or an equidistributed z-mesh must depend on $f(x, y)$. If f is quadratic in y, the two criteria agree. In de Boor [11] it is shown that in using the equidistributed knots, the optimal bound $O(1/N^2)$ is attained in some cases where the uniform knots are less than optimal. However, the uniform knots are easier to compute so we prefer them. This removes the remaining $N-1$ degrees of freedom discussed in § 2.

In either case the mesh is determined by the solution and must be computed dynamically. An iterative scheme is introduced in the next section to construct the numerical solution and grid structure.

4. Adaptive mesh technique and solution of the algebraic system. We will now discuss the solution of the nonlinear system of equations $A(x, z)z = h(x, z)$ subject to $N-1$ additional relations. In the error analysis of the previous section, we have seen that it is desirable to have uniformly spaced z_i points or to have equidistributed interpolation error. We will present an algorithm which solves this system along with the independent $N-1$ constraints that the z_i be equally spaced. We will also discuss how to constrain the z_i so that the interpolation error is equidistributed.

The first step is a fixed point iteration with a fixed x-mesh, say x^k. The solution z^k of the "inner" iteration

$$A(x^k, z^{k,m})z^{k,m+1} = h(x^k, z^{k,m})$$

(where m is an iteration counter) is an "acceptable" numerical solution of the boundary value problem. However, z^k does not necessarily have the desirable error properties, namely, $|y - z| = O(N^{-2})$. To achieve a uniformly spaced z_i, we update at the end of each "inner" iteration the value x_i^{k+1}. This entails the solution of a nonlinear equation:

$$z^k(x_i^{k+1}) = z_i^k,$$

where

$$z_i^k = i(\max z^k(x) - \min z^k(x))/N. \tag{4.1}$$

This procedure is called an "outer" iteration and amounts to an adaptive mesh technique. The $\{x_i^{k+1}\}$ is the new mesh. "Inner" and "outer" iterations are continued until convergence, i.e.,

$$|x_i^{k+1} - x_i^k| < \varepsilon_{\text{Tol}}$$

and (4.1) holds within ε_{Tol}, where $0 < \varepsilon_{\text{Tol}} \ll 1$. Similarly, for a solution with equidistributed interpolation error we replace (4.1) by (3.3).

It remains to discuss how we obtain the initial guesses for x, z based on N intervals. We do this by solving a sequence of problems based on an increasing number of intervals: $N = 1, 2, 4, 8, \cdots, 2^k, \cdots$. The initial guess for $N = 2^{k+1}$ intervals comes from the solution for $N = 2^k$ intervals. By saving the previous solutions, we can perform Richardson extrapolation conveniently. Note that if f is convex then the linear interpolations I_f approach f monotonically as N increases. By Lemma 2, we then have that the corresponding solutions z approach y monotonically.

5. Construction of the elliptic splines. In the nonlinear system $A(x, z)z = h(x, z)$, the matrix and right-hand side depend upon the derivatives of the spline basis functions w_i. In order to carry through the iteration scheme previously described, it is necessary to have a convenient and efficient way of computing these functions. The theory of asymptotics for linear ODE gives analytic representations (w'_{ε_i}) which, for $\varepsilon \to 0$ are asymptotic to the exact w_i'. The representations are in terms of exponential functions

of the g_{1_i} and can be efficiently computed. Due to the asymptotic property, they become more accurate as ε becomes smaller.

The following result is called the Liouville–Green or WKB approximation (Olver [29]). Let

$$b_i^{\pm}(x) = g_{1_i}^{-1/4}(x) \exp\left\{\pm \frac{1}{\sqrt{\varepsilon}} \int g_{1_i}^{1/2}(x)\, dx\right\}. \tag{5.1}$$

Then there exist error functions e^+, e^- such that $b^+(1+e^+)$, $b^-(1+e^-)$ are exact solutions of $E_i y = 0$. On $x_{i-1} < x < x_i$ the error functions satisfy:

$$|e_i^{\pm}(x)| \leqq \exp\left\{\frac{\sqrt{\varepsilon}}{2} V_{\pm}(F)\right\} - 1, \qquad F(x) = \int \frac{1}{g_{1_i}^{1/4}} \frac{d^2}{dx^2}\left(\frac{1}{g_{1_i}^{1/4}}\right) dx$$

where $V_{\pm}$ is the variation of $F(x)$ between x_{i-1} and x_i. Using these explicit $b_i^{\pm}$ we construct as before the approximations u_{ε_i}, v_{ε_i} and eventually get w_{ε_i} which are asymptotic approximations to the γ-elliptic splines w_i. These functions can be differentiated analytically and the results used to compute $A(x, z)$.

We introduce $S_\varepsilon(E, \{x_i\})$, the space of γ_ε-elliptic splines, generated by the set $\{w_{\varepsilon_i}; i = 0, 1, \cdots, N\}$. In general, $S_\varepsilon(E, \{x_i\})$ approximates $S(E, \{x_i\})$. In the important special case of an autonomous equation ($\varepsilon y'' = f(y)$), g_1 is independent of x and the integration in (5.1) can be carried out exactly. Then the WKB approximation reduces to the familiar exponential basis (Berger et al., [7], deGroen and Hemker [23], deGroen [24], Hemker [25]) for a constant coefficient equation and is an exact solution of $E_i y = 0$. For this case we therefore have $S_\varepsilon(E, \{x_i\}) \equiv S(E, \{x_i\})$. If the integral in (5.1) cannot be evaluated exactly, a quadrature can be used. Since for many interesting problems $g_1 \sim f_y(x, y)$ is a nice function of x this quadrature can be done cheaply.

5.1. Construction of y_p. The asymptotic representation of the particular solution y_{p_i} can be calculated by substituting the WKB representations of the local solutions into (2.9) and by performing integration by parts. This procedure yields a consistent expansion to all orders, but the amount of algebra to do these calculations can become prohibitive (Eckhaus [17]). However, the leading term is easily obtained:

$$y_{p_i} = -G_{p_i}(x) + G_{p_i}(x_{i-1}) v_{\varepsilon_i}(x) + G_{p_i}(x_i) u_{\varepsilon_i}(x) + O(\varepsilon),$$

where

$$G_{p_i}(x) = g_{0_i}/g_{1_i}.$$

Alternatively we may use the results of Flaherty and O'Malley [20], Olver [29], O'Malley [33] in which the particular solution is decomposed into a solution z_i of the reduced problem and boundary layer corrections L_i, R_i:

$$y_{p_i}(x) = L_i(x) + z_i(x) + R_i(x).$$

The solution z_i is expanded as an asymptotic series in ε and the coefficients are obtained recursively:

$$z_i = \sum_{n=0} \varepsilon^n z_{i_n}(x),$$

$$z_{i_0}(x) = -G_{p_i}(x),$$

$$z''_{i-1_n} = g_{0_i} + g_{1_i} z_{i_n}.$$

The boundary layer corrections L_i, R_i are given by

$$L_i(x) = -z_i(x_{i-1})v_{\varepsilon_i}(x), \qquad R_i(x) = -z_i(x_i)u_{\varepsilon_i}(x).$$

Note that $z_{i_0}(x)$ is the outer solution of (2.1). For the computations we have used only the leading term for y_p. This allows us to efficiently compute the right-hand side $h(x, y)$ of the algebraic system.

6. Numerical solutions. In this section, we present numerical solutions to a number of boundary value problems in the chemical and physical sciences.

Our first example is due to Carrier [14]. It arises in singular perturbation theory and geophysics:

$$\varepsilon y'' = [1 - 2b(1-x^2)y - y^2] \quad \text{with } y'(0) = 0, \quad y(1) = 0. \tag{6.1}$$

Note that when $b = 0$, if $f_y(1, 0) = 0$ and, thus, $x = 1$ and $y = 0$ is a turning point of the problem. Since the WKB approximation breaks down near a turning point our method is, in principle, ineffective. However, because the chords which interpolate f have nonzero slope, we can successfully compute the solution provided the number of mesh points is not large. For $b = 0$, (6.1) is autonomous and no such difficulties are encountered. Plotted in Fig. 6.1 is the graph of the solution for $\varepsilon = 10^{-6}$ and $b = 0$ with $N = 4$. For this problem, the equidistributed interpolation error mesh is identical to a uniform y-mesh since $f(x, y)$ is a quadratic function of y. In Table 6.1, the value $x(y = -.5)$ is given as a function of N, the number of intervals, using $N = 64$ solution

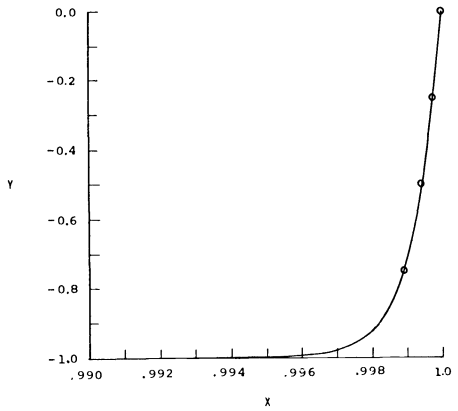


FIG. 6.1. *Solution of Carrier's problem with* $\varepsilon = 10^{-6}$, $b = 0$ *and* $N = 4$.

TABLE 6.1
Carrier's problem with $\varepsilon = 10^{-6}$

N	$x_N(y = 0.5)$	$\lvert x_N - x_{64}\rvert \times 10^5$	$e_N \times 10^5 \times N^2$	Extrapolated value
2	.99941385	2.29	9.16	
4	.99943121	.554	8.86	.999437
8	.99943538	.137	8.77	.99943677
16	.99943642	.033	8.43	.99943677
32	.99943668	.007	7.17	
64	.99943675	.0		

as a representation of the exact solution. The error in the location of $x(y=-.5)$ is tabulated in the second column. Column 3 shows that the error in locating $y=-.5$ is, indeed, $O(1/N^2)$. Finally, Richardson's extrapolation is applied to the results and is tabulated in column 5. For $b=1$, we plot the solution in Fig. 6.2 for $\varepsilon = 10^{-6}$ and $N=8$. Note that our method computes successfully the transition between the outer solution and the boundary layer with only a few grid points (see Fig. 6.3). This attests to the importance of using a local basis that captures the essential behavior of the solution.

The second example is the Troesch problem modeling the confinement of a plasma column by radiation pressure [44]:

$$y'' = \varphi \sinh \varphi y, \quad \varphi > 0, \qquad y(0)=0, \quad y(1)=1.$$

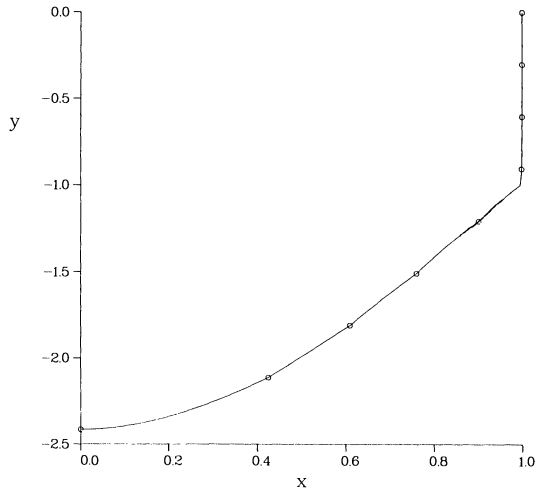


FIG. 6.2. *Solution of Carrier's problem with* $\varepsilon = 10^{-6}$, $b=1$ *and* $N=8$.

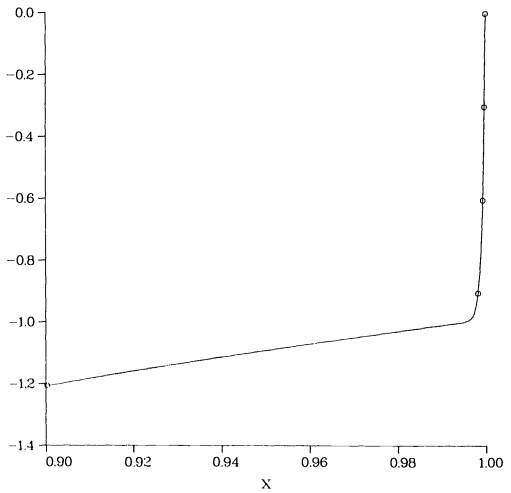


FIG. 6.3. *An enlarged graph of Fig.* 6.2. *for* $.90 \leqq x \leqq 1.00$.

This is a particularly difficult problem because of the strongly nonlinear right-hand side. The problem of computing $y'(0)$ and $y'(1)$ has been used as a benchmark for stiff boundary value problem algorithms (Roberts and Shipman [39], Troesch [44]). The derivatives $y'(0)$ and $y'(1)$ have the following asymptotic expansion (Anglesio and Troesch [2], Chin [15]):

$$y'(0) = y_0'(0)\left\{1+\frac{y_0'(0)^2}{4}\left[\varphi - 1 + \frac{\cosh(\varphi/2)}{\sinh^2(\varphi/2)}\right]\right\} + O(\varphi e^{-3\varphi}), \tag{6.2}$$

where

$$y_0'(0) = 8e^{-\varphi}\tanh(\varphi/4)$$

and

$$y'(1) = 2\sinh(\varphi/2)[1-\{y'(0)/2\sinh(\varphi/2)\}^2]^{1/2}. \tag{6.3}$$

We plot in Fig. 6.4 the relative error in the derivative $y'(0)$ between the numerical and the asymptotic solution (6.2) multiplied by N^2 as a function of N for $\varphi = 1$, 5, 10, 20. The N^2 normalization is suggested by the error analysis of § 3. The computations are done with an uniformly spaced y-mesh. Also plotted in Fig. 6.4 is a calculation for $\varphi = 5$ using a equidistributed interpolation error mesh. Figure 6.5 is a similar plot for the derivative $y'(1)$. It is seen from Figs. 6.4–6.5 that $y'(1)$ is calculated more accurately than $y'(0)$ as φ increases. This is because the mesh points collect in the boundary layer with increasing φ. Moreover, the equidistributed interpolation error

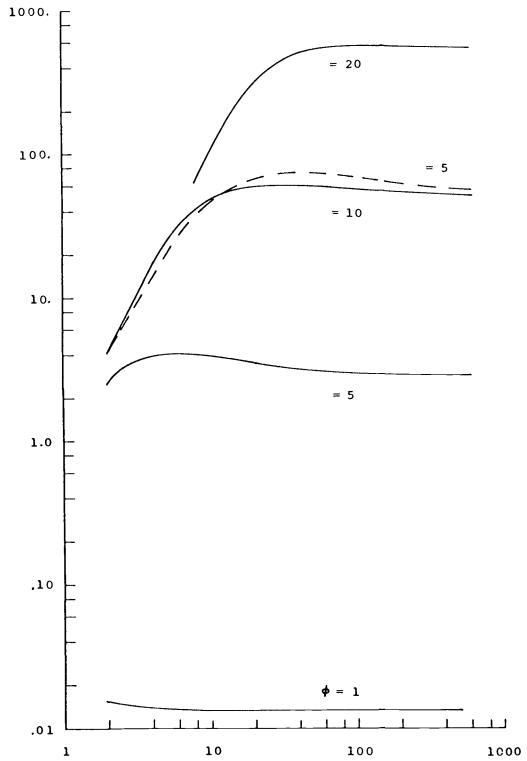


FIG. 6.4. *Solution of Troesch problem: Relative error of* $y'(0)$, $[z'(0)-y'(0)]N^2/y'(0)$ *as a function of the number of intervals, N. Equidistribution of interpolation error is used* (– –).

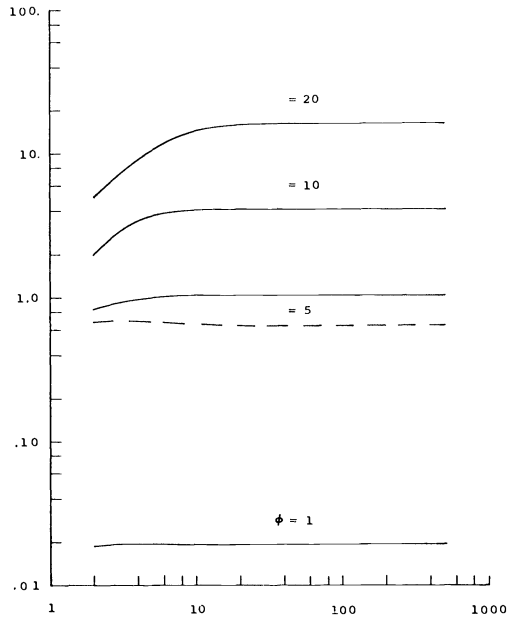


FIG. 6.5. *Solution of Troesch problem: Relative error of* $y'(1)$, $[z'(1)-y'(1)]N^2/y'(1)$ *as a function of the number of intervals, N. Equidistribution of interpolation error is used* (– –).

mesh tends to locate more points in the boundary layer than the uniformly spaced y-mesh. As a consequence, the equidistributed interpolation error mesh yields a more accurate value of $y'(1)$ while the uniformly spaced y-mesh gives a better approximation to $y'(0)$. Overall if both derivatives $y'(0)$ and $y'(1)$ are desired, the uniformly spaced y-mesh tends to perform better.

Our final example appears in the theory of diffusion and reaction in permeable catalysts (Aris [3]):

$$\frac{d^2c}{dx^2} = \varphi^2 \exp\left\{\gamma\left(1-\frac{1}{T}\right)\right\} f(c),$$
$$\frac{d^2T}{dx^2} = -\beta\varphi^2 \exp\left\{\gamma\left(1-\frac{1}{T}\right)\right\} f(c). \tag{6.4}$$

with boundary conditions

$$c'(0) = T'(0) = 0, \qquad c(1) = 1 - \frac{1}{Sh}\, c'(1), \tag{6.5}$$

and

$$T(1) = 1 - \frac{1}{Nu}\, T'(1).$$

Equations (6.4) may be combined with boundary conditions (6.5) to yield

$$T(x) = 1 + \beta\frac{Sh}{Nu} + \beta\left[1 - \frac{Sh}{Nu}\right] c(1) - \beta c(x). \tag{6.6}$$

Substituting (6.6) into (6.4), we obtain an equation for c. However, the boundary value $c(1)$ appears in the differential equation and, thus, complicates the solution

procedure. We solve (6.4a), (6.6) and boundary conditions (6.5) using the parameters given by Carey and Finlayson [13]:

$$\beta = 0.2, \quad Sh/Nu = 50, \quad Sh = 250, \quad \varphi = 14.44, \quad \gamma = 20 \quad \text{and} \quad f(c) = c.$$

The most important feature of the solution is the accurate computation of $dT/dx(1)$ (Carey and Finlayson [13]). In view of the results on the Troesch problem and of the added expense in applying the equidistributed interpolation error mesh, a uniformly spaced y-mesh is selected for this set of computations.

The solution for $N = 4$ is plotted in Fig. 6.6. The value $dT/dx(1)$ is tabulated in Table 6.2 as a function of N. It is clearly seen from Table 6.2 that four place accuracy in $dT/dx(1)$ is obtained with just two intervals.

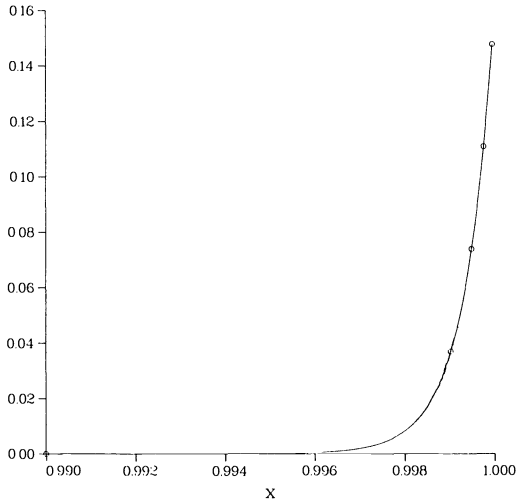


FIG. 6.6. *Solution of a diffusion-reaction equation with* $N = 4$.

TABLE 6.2
Solution of a diffusion-reaction equation

N	$dT/dx(1)$
1	−4.2586013
2	−4.2607540
4	−4.2612872
8	−4.2613201
16	−4.2614534
32	−4.2614617
64	−4.2614637

7. Conclusions and generalizations. It can be seen from the numerical examples just how important is the use of a basis that captures the essential behavior of the solution. The basis functions must reflect the rapid growth or decay of the solution. This property is inherent in asymptotic methods and in solving boundary value problems by approximating the coefficients.

Since the class of problems we have considered is rather restricted, we close with some suggestions about extensions of the method to a wider class. For problems with f depending on y and y', a multivariate interpolation with respect to y and y' on a triangular domain may be used to reduce the nonlinear problem to a collection of disjointed linear two-point boundary value problems. The same end can be achieved by combining a straightforward application of quasilinearization and approximating the coefficients of the linear differential equation. The subinterval problems are then solved by asymptotic methods. The local solutions are "patched" together to form the global solution. Here, the WKB method is invalid near turning points. This calls for a uniform asymptotic method in the neighborhood of a turning point.

For a system of differential equations, quasilinearization followed by approximating the coefficients of the linear DE is more suitable. The resulting linear boundary value problem in each subinterval, however, is by no means trivial to solve.

Acknowledgments. We thank Gerald Hedstrom for many stimulating discussions. We also thank the U.S. Department of Energy Office of Basic Energy Sciences, Applied Mathematics and Statistics Division. We thank J. E. Flaherty and G. J. Majda for suggesting improvements in the presentation of the manuscript.

REFERENCES

[1] L. R. Abrahamsson, H. B. Keller and H. O. Kreiss, *Difference approximations for singular perturbations of systems of ordinary differential equations*, Numer. Math., 22 (1974), pp. 367–391.

[2] J. D. Anglesio and B. A. Troesch, *High precision results for a two-point boundary value problem*, J. Comp. Appl. Math., 6 (1980), pp. 99–103.

[3] R. Aris, *The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts*, Vol. I, Clarendon Press, Oxford, 1975.

[4] U. Ascher, J. Christiansen and R. D. Russell, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659–679.

[5] J. P. Aubin, *Approximation of Elliptic Boundary-Value Problems*, Wiley–Interscience, New York, 1972, Chapt. 6.

[6] A. E. Berger, J. M. Solomon, M. Ciment, S. H. Leventhal and B. C. Weinberg, *Generalized OCI schemes for boundary layer problems*, Math. Comp., 35 (1980), pp. 659–731.

[7] A. E. Berger, J. M. Solomon and M. Ciment, *An analysis of uniformly accurate difference method for a singular perturbation problem*, Math. Comp., 37 (1981), pp. 79–94.

[8] S. R. Bernfeld and V. Lakshmikantham, *An Introduction to Nonlinear Boundary Value Problems*, Academic Press, New York, 1974.

[9] C. de Boor, *Good approximation by splines with variable knots*, in Spline Functions and Approximation Theory. A. Muir and A. Sharma, eds., Birkhauser Verlag, Basel, 1973, pp. 52–72.

[10] C. de Boor and B. Swartz, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 583–606.

[11] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[12] N. I. Bris, *On boundary problems for the equation* $\varepsilon y'' = f(x, y, y')$ *for small* ε, Dokl. Akad. Nauk. SSSR, 95 (1954), pp. 429–432. (In Russian.)

[13] G. F. Carey and B. A. Finlayson, *Orthogonal collocation on finite elements*, Chem. Engr. Sci., 30 (1975), pp. 587–596.

[14] G. F. Carrier, *Singular perturbations and geophysics*, SIAM Rev., 12 (1970), pp. 175–193.

[15] R. C. Y. Chin, *On the matched asymptotic solution of the Troesch problem*, J. Comp. Appl. Math., 7 (1981), 181–185.

[16] F. W. Dorr, S. V. Parter and L. F. Shampine, *Applications of the maximum principle to singular perturbation problems*, SIAM Rev., 15 (1973), pp. 43–88.

[17] W. Eckhaus, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979.

[18] P. C. Fife, *Semilinear elliptic boundary value problems with small parameters*, Arch. Rational Mech. Anal., 52 (1973), pp. 205–232, MR51 #10863.

[19] J. E. Flaherty and W. Mathon, *Collocation with polynomial and tension splines for singularly-perturbed boundary value problems*, this Journal, 1 (1980), pp. 260–289.

[20] J. E. FLAHERTY AND R. E. O'MALLEY, JR, *The numerical solution of boundary value problems for stiff differential equations*, Math. Comp., 31 (1977), pp. 66–93.
[21] B. FRIEDMAN, *Principles and Techniques of Applied Mathematics*, John Wiley, New York, 1956.
[22] D. GIDASPOW AND B. S. BAKER, *A model for discharge of storage batteries*, J. Electrochem. Soc., 120 (1973), pp. 1005–1010.
[23] P. P. N. DE GROEN AND P. W. HEMKER, *Error bounds for exponentially fitted Galerkin methods applied to stiff two-point boundary value problems*, in Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. H. Miller, eds., Academic Press, London, 1979.
[24] P. P. N. DE GROEN, *A finite element method with a large mesh-width for a stiff two-point boundary value problem*, J. Comp. Appl. Math., 7 (1981), pp. 3–15.
[25] P. W. HEMKER, *A numerical study of stiff two-point boundary problems*, Ph.D. dissertation, Mathematisch Centrum, Amsterdam, 1977.
[26] H. B. KELLER, *Numerical Methods for Two-Point Boundary Value Problems*, Ginn-Blaisdell, Waltham, MA, 1968.
[27] R. B. KELLOG AND A. TSAN, *Analysis of some difference approximations for a singular perturbation problem without turning points*, Math. Comp., 32 (1978), pp. 1025–1039.
[28] V. S. MARKIN, A. A. CHERNENKO, YU. A. CHIZMADZHEV AND YU. G. CHIRKOV, *Aspects of the theory of gas porous electrodes*, in Fuel Cells, Their Electrochemical Kinetics, V. S. Bagotskii and Yu. B. Vasil'ev, eds., Consultants Bureau, New York, 1966, pp. 21–33.
[29] F. W. J. OLVER, *Asymptotics and Special Functions*, Academic Press, New York, 1974.
[30] ———, *Asymptotic approximations and error bounds*, SIAM Rev., 22 (1980), pp. 188–203.
[31] R. E. O'MALLEY, JR, *On a boundary value problem for a nonlinear differential equation with a small parameter*, SIAM J. Appl. Math., 17 (1969), pp. 569–581.
[32] ———, *Introduction to Singular Perturbations*, Academic Press, New York, 1974.
[33] ———, *Boundary layer methods for ordinary differential equations with small coefficient multiplying the highest derivatives*, Proc. Sympos. on Constructive and Computational Methods for Differential and Integral Equations, Lecture Notes in Mathematics 430, Springer-Verlag, New York, 1974, pp. 363–389.
[34] S. OSHER, *Numerical solution of singular perturbation problems and hyperbolic systems of conservation law*, in Analytical and Numerical Approaches to Asymptotic Problems in Analysis, S. Axelsson, L. S. Frank, and A. Van Der Sluis, eds., North-Holland, Amsterdam, 1981, pp. 179–204.
[35] M. H. PROTTER AND H. F. WEINBERGER, *Maximum Principles in Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1967.
[36] S. A. PRUESS, *Solving linear boundary value problems by approximating the coefficients*, Math. Comp., 27 (1973), pp. 551–561.
[37] G. W. REDDIEN, *Projection methods for two-point boundary value problems*, SIAM Rev., 22 (1980), pp. 156–171.
[38] M. E. ROSE, *Finite difference schemes for differential equations*, Math. Comp., 18 (1964), pp. 179–195, MR32 #605.
[39] S. M. ROBERTS AND J. S. SHIPMAN, *On the closed form solution of Troesch's problem*, J. Comp. Phys., 21 (1976), pp. 291–304.
[40] R. D. RUSSELL AND L. F. SHAMPINE, *A collocation method for boundary value problems*, Numer. Math., 19 (1972), pp. 1–28.
[41] M. H. SCHULTZ, *Elliptic spline functions and the Rayleigh–Ritz–Galerkin method*, Math. Comp., 24 (1970), pp. 65–80.
[42] L. F. SHAMPINE, *Boundary value problems for ordinary differential equations*. II: *patch bases and monotone methods*, SIAM J. Numer. Anal., 6 (1969), pp. 414–431.
[43] I. STAKGOLD, *Boundary Value Problms of Mathematical Physics* Vol. I, Macmillan, New York, 1967.
[44] B. A. TROESCH, *A simple approach to a two-point boundary value problem*, J. Comp. Phys., 21 (1976), pp. 279–290.
[45] J. YARMISH, *Newton's method techniques for singular perturbations*, SIAM J. Math. Anal., 6 (1975), pp. 661–680.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0009 $01.25/0

MULTIGRID METHODS FOR DIFFERENTIAL EIGENPROBLEMS*

A. BRANDT†, S. McCORMICK‡ AND J. RUGE‡

Abstract. This paper develops an efficient multigrid algorithm for solving the eigenvalue problem associated with a linear differential operator. The algorithm is based on the full approximation scheme (FAS) and incorporates a Ritz projection process for simultaneous computation of several eigenvalues and their eigenvectors. Included are the results of some numerical experiments that illustrate its performance in various contexts.

Key words. multigrid, eigenvalue, nonlinear, partial differential equations, eigenvectors, eigenfunction

1. Introduction. The usual method for finding eigenvalues of a differential operator is to discretize the problem and solve the resulting matrix eigenvalue problem by some algebraic technique. If, as we assume here, the first several eigenvalues and corresponding eigenvectors are desired, then one may use a simultaneous or block version of such methods as inverse iteration, Rayleigh quotient iteration or Lanczos.

This approach of treating the discrete problem as a purely algebraic one can result in a loss of valuable information, especially concerning the smoothness of the eigenvectors. In general, the operator's eigenvectors corresponding to the desired smaller eigenvalues are very smooth, so that they are fairly well approximated on coarser grids. Certain multigrid processes (e.g., FMG described in § 3.2) take full advantage of this smoothness and are therefore very effective for solving such problems.

The experiments we have performed indicate that the first eigenvalue of a differential operator can be approximated to within truncation error with a little more work than is needed for solving the related boundary value problem by multigrid. When more than one eigenvalue is desired, the work needed per eigenvalue increases somewhat due primarily to the orthogonalization and Ritz steps used to prevent all of the emerging eigenvalue approximations from converging to the first eigenvalue. Nevertheless, as we note in § 7, the total work is $\frac{20}{3}q^2n + O(q^2n + q^3 \log n)$, where n is the number of fine grid points and q is the number of desired eigenvalues.

After introducing the notation and some basic multigrid ideas, the method for finding an approximation to the first eigenvalue of the operator is detailed and discussed. This method is then extended in § 6 to the computation of several eigenvalues.

Basically, the algorithm proposed in this paper uses the version of multigrid that treats the eigenvalue problem as a nonlinear problem on all grids. The problem is solved on successively finer grids, using the solution at each level as the initial guess for the next. To improve this initial guess, a multigrid cycle is then performed for each eigenvector, retaining nonlinearity on coarser grids and maintaining separation of the vectors by coarse-grid orthogonalization with respect to previous eigenvectors.

The Ritz projection is used to maintain a stable basis for the emerging invariant subspace approximation and results in accelerating the speed of convergence of the multigrid iteration to the true eigenvectors. It raises several questions in algorithm design, as we shall see.

* Received by the editors October 6, 1981. This work was supported by the National Science Foundation under grant MCS80-17056 and the Air Force Office of Scientific Research under grant AFOSR-F-33615-79-C-3223.

† Applied Mathematics Department, Weizmann Institute of Science, Rehovot, Israel.

‡ Department of Mathematics, Colorado State University, Fort Collins, Colorado 80523.

Several other algorithms incorporating multigrid methods have been proposed for the solution of differential eigenvalue problems. Hackbusch [5] developed a method for approximating the eigenvalues of an elliptic differential operator. The basic algorithm is similar to ours although the emphasis of his work is mainly theoretical. Strakhovskaya [9] proposes another method for approximating the first eigenvalue that is similar to Hackbusch's but that uses coarser grids to solve the residual equation obtained on the finest. Clemm and Guderley [4] use the theory of a two-level method for linear problems to develop a method for finding several eigenvalues. It is much different from the modern recursive multigrid approach taken here. McCormick [7] uses Newton's method (that is, Rayleigh quotient iteration) with multigrid as the inner-loop equation solver, together with Ritz projections to compute several eigenvalues and eigenvectors.

The method developed by Alcouffe, et al. [1] for solving linear problems has been extended to apply to eigenvalue problems. It is similar to our approach for computing the first eigenvalue, although their emphasis is in applying the method to their problem and ours is in a full development of the algorithm.

The essential features of our algorithm were developed by the authors in 1979. In fact, the present algorithm is an improvement of the one coded and distributed on magnetic tape at the 1979 multigrid workshop at Yale University.

2. Notation. Let L be a differential operator on a set of functions defined on a domain Ω in $\mathbb{R}^d$. Let $G^1, G^2, \cdots, G^m$ be a sequence of increasingly finer grids that extend over Ω. Assume that all grids are uniform and that each, except G^1, is a refinement of the previous grid made by halving the mesh size. Let L^k denote the finite difference approximation to L on grid G^k.

Since interaction between grids is necessary, we need a procedure for transferring functions from one grid to another. Let G^k and G^l be two different grids and let u^k be a function defined on G^k. Let I_k^l be a mapping from the functions on G^k to the functions on G^l such that:

a) if G^k is coarser than G^l, $I_k^l u^k$ is the function obtained on G^l by linear interpolation of u^k to G^l; and

b) if G^k is finer than G^l, the value of $I_k^l u^k$ at a given point of G^l is the weighted average of a small number of points neighboring the corresponding G^k point. (This includes the frequently used definition that the value of $I_k^l u^k$ at a given point of G^l is just the value of u^k at the corresponding point of G^k.)

For $k < l-1$ we assume that $I_l^k = I_{l-1}^k I_l^{l-1}$.

In the following, the inner product denoted by $\langle \cdot, \cdot \rangle$ is a discrete approximation to the continuous $L_2(\Omega)$ inner product which, in d dimensions, is given by

$$\langle u^k, v^k \rangle = \sum_{x \in G^k} h_k^d u^k(x) v^k(x).$$

Here, h_k is the mesh size of G^k and $u^k(x)$ and $v^k(x)$ represent the values of u^k and v^k at the grid point x.

3. Basic multigrid processes.

3.1. Multigrid cycle. The efficiency of multigrid methods results from the fact that, although relaxation is usually slow to converge, it is quick to reduce high-frequency error components. This allows the problem to be transferred to a coarser grid where the error can be resolved with much less work. (Not only is relaxation cheaper per sweep on coarser grids, but the solution process is also much more

effective.) The coarse grid equation can be solved by relaxation and appeal to still coarser grids. The coarsest grid used is chosen so that solution of the problem there is inexpensive compared to the work performed on the fine grid. The number of relaxation sweeps needed to smooth the error on each grid is generally small and can be predetermined by the usual mode analysis, for example. The process of using a predetermined number of sweeps per grid is called fixed cycle multigrid. Such a cycle is defined by the following steps (where G^l is the finest grid).

1. Set $k \leftarrow l$.
2. Relax ν_1 times on the G^k problem.
3. If $k = 1$, go to 4.
   Otherwise, set $k \leftarrow k - 1$, transfer the problem to a coarser grid and go to 2.
4. Relax ν_2 times on the G^k problem.
5. If $k = l$, stop.
   Otherwise, correct the G^{k+1} solution approximation using the solution of the G^k problem, set $k \leftarrow k + 1$ and go to 4.

(This actually illustrates the so-called V-cycle, where one multigrid cycle on the G^k problem involves exactly one multigrid cycle on G^{k-1}. The so-called W-cycle uses two G^{k-1} cycles before correcting the G^k solution.) Note that the problem on G^1, the coarsest grid, is solved by $\nu_1 + \nu_2$ relaxation sweeps.

3.2. Full multigrid (FMG) algorithm. The cycling scheme described above does not provide a procedure for determining a good initial guess for the problem on G^l. However, since the G^l problem is approximated by one on G^{l-1}, it is natural to determine a good starting vector by solving the G^{l-1} problem and interpolating the result to G^l. This suggests solving the problem first on G^1 (by some convenient method) and then solving the grid G^l problem by multigrid cycling, using the interpolated solution from level $l-1$ as the initial approximation ($l = 2, 3, \cdots, m$). Cubic interpolation may be used here for the initial approximation to the G^l cycle (cf. [3]). Algorithms utilizing this procedure are referred to by the term full multigrid (FMG).

Throughout this paper, l denotes the currently finest level, that is, the finest level reached thus far in the FMG procedure, and level k is the current level in the cycling process.

3.3. Full approximation scheme (FAS). For linear problems, the residual equation for the G^k problem may be transferred to G^{k-1}. The solution of the resulting coarse grid problem then yields an approximation to the error of the G^k solution approximation, so the G^{k-1} solution is interpolated to G^k and added to the solution there. However, for nonlinear problems, such as eigenproblems, instead of the actual error, it is more convenient to approximate the fine grid solution itself on coarse grids. This leads to the very powerful but subtle so-called full approximation scheme (FAS), the details of which can be found in [2]. The essence of FAS is the construction of the coarse grid problem so that its solution is a good approximation to the fine grid solution transferred to the coarse grid. This ensures that the difference between the fine grid approximation and the coarse grid solution is an approximation to the smooth components of the fine grid error so that it can be used as a correction to reduce these components. Note, at convergence, that the coarse grid solution must provide a zero correction; that is, the coarse grid solution is just the fine grid solution transferred to the coarse grid.

To be more specific, assume that FMG is applied to $LU = f$ and level l is the currently finest grid. The G^l version of the problem is written as $L^l U^l = f^l$. Assuming

we have an approximation, u^l, to the exact solution, U^l, and the error $U^l - u^l$ is smooth, then the problem is transferred to G^{l-1} as

$$L^{l-1}U^{l-1} = I_l^{l-1} f^l + \tau^{l-1},$$

where $\tau^{l-1} = L^{l-1} I_l^{l-1} u^l - I_l^{l-1} L^l u^l$. (Note that τ^{l-1} is approximately what is needed to obtain $I_l^{l-1} U^l$ as the coarse grid solution; that is, if $u^l = U^l$, then the level $l-1$ solution is exactly $I_l^{l-1} U^l$.) Transfer of the problem to grid G^k, $k \leqq l-1$, using the same idea yields the problem

$$L^k U^k = I_l^k f^l + \tau^k, \tag{3.1}$$

where

$$\tau^k = L^k I_{k+1}^k u^{k+1} + I_{k+1}^k (\tau^{k+1} - L^{k+1} u^{k+1}). \tag{3.2}$$

The notation we adopt here uses the usual somewhat simplified multigrid conventions. However, this simplicity introduces unfortunate ambiguities. Specifically, note that capital letters are used to denote exact solutions and small letters to denote current approximations to these solutions. Thus, U^k is the exact solution to the G^k problem, so it depends on τ^k, which in turn depends on $u^{k+1}, \cdots, u^l$, and l itself. Thus, τ^k and hence U^k do not represent the same functions throughout the solution process. This is an ambiguity that is necessary to avoid substantially more complex notation, but should cause no difficulty to the reader if this dependence of the coarse grid solution on the emerging fine grid solution is kept in mind.

Once a suitable approximation, u^k, to U^k is found, the approximation u^{k+1} can be corrected according to

$$u^{k+1} \leftarrow u^{k+1} + I_k^{k+1} (u^k - I_{k+1}^k u^{k+1}). \tag{3.3}$$

This is the so-called FAS interpolation step. (Note that I_{k+1}^k is used in two different terms in (3.2). In some instances, it is advisable to use different interpolation schemes for each of these terms, but it is essential that the interpolation used for u^{k+1} in (3.2) be identical to the interpolation used in (3.3).)

The processes mentioned above form the basis for the algorithms used in this paper.

4. The problem. The problem treated in this paper is to find approximations to the first few eigenvalues and associated eigenvectors of the differential operator L defined on functions with domain Ω. That is, we are looking for the smallest real numbers $\Lambda_1 \leqq \Lambda_2 \leqq \cdots \leqq \Lambda_q$ and functions $U_1, U_2, \cdots, U_q$ so that

$$\begin{aligned} &LU_i - \Lambda_i U_i = 0 \quad \text{on } \Omega, \\ &U_i = 0 \quad \text{on } \partial\Omega, \\ &\langle U_i, U_j \rangle = L_{ij}, \quad i, j = 1, 2, \cdots, q. \end{aligned} \tag{4.1}$$

Here we assume that L is an elliptic, self-adjoint differential operator on Ω. (Our work extends naturally to the generalized eigenproblem $LU - \Lambda MU = 0$ with appropriate assumptions on M. We restrict our attention to the case $M = I$ for simplicity.)

Several alternatives can be used in specifying the accuracy required of the eigenvalues and their eigenvectors. For example, a mesh size can be prespecified, determining the discrete operator whose first q eigenvalues approximate those of L. In this case, it is required to find the eigenvalues and eigenvectors to the level of truncation error, that is, to the level of accuracy determined by the extent to which

the discrete eigenvalues and eigenvectors approximate those of the continuous operator. Solving beyond this level does not necessarily improve the error between the approximations and the solutions of the continuous problem. Note in this case that the accuracy is not uniform; that is, the error is larger in the higher eigenvalues than the lower ones since higher eigenvalues generally have larger truncation errors.

Another way to define the accuracy requirements is to specify each eigenvalue error explicitly by giving a fixed tolerance for all eigenvalue approximations. This method is more intricate, since different fine grids will usually be required for different eigenvalue/eigenvector approximations.

The method presented in this paper is designed according to the first accuracy criterion, although the design and testing of an algorithm for use with the second is discussed briefly in § 6. We first examine the case where only one eigenvalue is desired. This avoids this accuracy question yet clarifies some of the basic processes involved in the full algorithm.

Following the notation introduced in § 3, the current G^k approximations to the ith G^l eigenvector and eigenvalue are denoted by u_i^k and λ_i^k, respectively, where G^l is the currently finest grid. Upper case is reserved for exact solutions so that U_i^l and Λ_i^l are the exact G^l solutions. When no ambiguity exists, superscripts and subscripts will be dropped.

5. The method for the first eigenvalue.

5.1. The basic method. The full algorithm is listed in the appendix. In this section we discuss the essentials for a version of this method as it applies to the computation of the first eigenvalue only.

The first step is to obtain the coarsest grid approximation to the first eigenvalue and its eigenvector. Some initial approximation u^1 on this grid is chosen at random, although whatever information is known about the first eigenvector may be used here to provide a better initial guess. λ is chosen as a suitable approximation to the first eigenvalue of the operator. On this coarse grid, relaxation is performed on the equation $L^1u^1-\lambda u^1=0$ with λ fixed, followed by an update of λ by computing the Rayleigh quotient

$$\lambda=\langle L^1u^1, u^1\rangle/\langle u^1, u^1\rangle.$$

This process is repeated until a fairly accurate solution emerges. The vector is then normalized. In the experiments reported in this paper, 15 iterations were generally sufficient, although this value depends on the mesh size of the coarsest grid and the separation of the first and second eigenvalues there (cf. Kahan [6]). In any case, these iterations cost very little since the coarsest grid has only a few points.

Once a solution has been obtained on a level $l-1$, u^{l-1} is cubically interpolated to level l, where it is then improved by one fixed multigrid cycle from level l to level 1 and back. On all but the coarsest grid of this cycle, relaxation is performed holding λ fixed. Two relaxation sweeps per grid ($\nu_1=\nu_2=2$) usually ensure sufficient smoothing and elimination of high frequencies introduced by coarse grid corrections. Thus, the cycle begins with a fixed number of level l relaxation sweeps on the approximate vector followed by FAS transfer of the problem to the next coarser grid. This is repeated until the coarsest grid is reached, where the nonlinear problem resulting there from FAS transfers is solved by repeated relaxation and normalization of the approximate vector and update of λ by (5.6). FAS interpolation (3.3) is then used to transfer the corrected solution back to the next finer grid, where a fixed number of relaxation sweeps is performed, followed by transfer of the approximation to a finer

grid until level l is again reached. Relaxation on u^l is then performed to complete the level l cycle is then increased by 1.

The process of cubic interpolation followed by one multigrid cycle on level l is then continued to yet finer grids until a solution is obtained on level m, the finest grid to be used.

The remainder of this section is devoted to a detailed discussion of certain aspects of this method.

5.2. The coarse grid problem. The coarse grid equation is derived in much the same way as described in § 3. Specifically, let the currently finest grid problem be written as

$$L^l U^l - \Lambda^l U^l = 0, \qquad \eta^l(U^l) = 1, \tag{5.1a,b}$$

where η^l is a normalization constraint functional which specifies the size of the solution. This constraint is necessary for uniqueness since a solution U^l to (5.1a) is specified only up to a multiplicative constant. The specific nature of η will be discussed below. If we have an approximation u^l to U^l, and the error has been smoothed by relaxation, then the G^{l-1} equations are

$$L^{l-1}U^{l-1} - \Lambda^{l-1}U^{l-1} = \tau^{l-1}, \qquad \eta^{l-1}(U^{l-1}) = \sigma^{l-1},$$

where $\tau^{l-1} = L^{l-1}I_l^{l-1}u^l - I_l^{l-1}L^l u^l$ and σ^{l-1} is some appropriate quantity. In general, the G^k problem is

$$L^k U^k - \Lambda^k U^k = \tau^k, \qquad \eta^k(U^k) = \sigma^k \tag{5.2a, b}$$

with $\tau^k = I_{k+1}^k \tau^{k+1} + L^k I_{k+1}^k u^{k+1} - I_{k+1}^k L^{k+1} u^{k+1}$, $\tau^l = 0$.

To see what σ^k should be, it is necessary to examine the goal of the coarse grid solution process. Since Λ^k is unknown, there are fewer equations than unknowns, hence (5.2a) can have an infinite number of solutions. Condition (5.2b) is needed in order to uniquely define the solution. One possible way to define σ^k, analogous to the definition of τ^k, is

$$\sigma^k = \sigma^{k+1} + [\eta^k(I_{k+1}^k u^{k+1}) - \eta^{k+1}(u^{k+1})], \qquad \sigma^l = 1. \tag{5.3}$$

This definition arises from the attempt to force the corrected level l solution to satisfy (5.1b). However, this is not actually necessary, since on the fine grid a solution of any reasonable size is acceptable. Thus, it is acceptable to use a constraint which seeks to maintain the size of the solution, and which, together with (5.2a), provides a zero correction at convergence. Such a constraint is given by

$$\sigma^k = \eta^k(I_{k+1}^k u^{k+1}). \tag{5.4}$$

This constraint is easier to enforce than (5.3) and appears to work well experimentally, so we adopt this definition for our algorithm.

The most obvious choice for the normalization functional is

$$\eta^i(u^i) = \langle u^i, u^i \rangle.$$

However, since the condition

$$\langle u^k, u^k \rangle = \sigma^k$$

when coupled with (5.2a) admits two related solutions, one of which is essentially the negative of the desired one, it is possible to converge to the wrong one on the coarsest grid. In addition to obtaining a bad eigenvalue approximation, this would cause the

corrected approximation to lose most of the eigenvector component that we really want. It is therefore better to use a linear normalization constraint to ensure that the extraneous solution is eliminated. To maintain the "sign" of the coarse grid solution, we can choose $\eta^k = \langle u^k, I_{k+1}^k u^{k+1} \rangle$ so that the normalization constraint then becomes

$$\langle u^k, I_{k+1}^k u^{k+1} \rangle = \langle I_{k+1}^k u^k, I_{k+1}^k u^{k+1} \rangle. \tag{5.5}$$

This normalization constraint, which we adopt here, ensures that the coarse grid solution is approximately in the same direction as the current fine grid approximation, and is compatible at convergence with (5.1a).

Relaxation on the finest few grids affects high-frequency error, which is generally small compared to the solution norm, so the size of the solution changes only slightly. However, on the coarser grids (i.e., for $k = 1$ or 2), low-frequency changes are made and the magnitude of the major component of the eigenvector approximation can change significantly. For this reason, we enforce the normalization constraint (5.5) on the coarse grids only.

For the eigenvalue calculation, note that τ^k naturally provides the correction needed to compute a Rayleigh quotient on the coarser grid. That is, λ can be updated by the relation

$$\lambda = \frac{\langle L^k u^k - \tau^k, u^k \rangle}{\langle u^k, u^k \rangle}. \tag{5.6}$$

5.3. Relaxation. Gauss–Seidel relaxation with a shift parameter for equation (5.2a) was used in the experiments reported in this paper. Let M^k be the lower triangular part of L^k (including the main diagonal), u^k and λ the current approximations to U^k and Λ^k and $\bar{u}^k$ the approximation after relaxation. For each constant shift μ, this relaxation scheme is given by

$$\bar{u}^k = u^k + (M^k - \mu I)^{-1}(\tau^k - (L^k - \lambda I)u^k). \tag{5.7}$$

A useful tool for analyzing (5.7) is the smoothing rate analysis exemplified in [2]. For simplicity, let L^k be the usual five-point approximation to $-\Delta$. If we describe the grid by $G^k = \{(\alpha h_k, \beta h_k) \in \Omega\colon \alpha, \beta \text{ integers}\}$, where Ω is the function domain for L^k, then a given Fourier error component before and after the relaxation sweep is denoted by $A_\theta e^{i(\theta_1\alpha+\theta_2\beta)}$ and $\bar{A}_\theta e^{i(\theta_1\alpha+\theta_2\beta)}$, respectively. Letting the corresponding component of the actual solution U^k be given by $B_\theta e^{i(\theta_1\alpha+\theta_2\beta)}$, then $\bar{A}_\theta$ is given as a function of A_θ and B_θ by the following equation corresponding to the relaxation scheme (5.7):

$$\begin{aligned}\bar{A}_\theta = &-A_\theta(e^{i\theta_1} + e^{i\theta_2} + h^2(\lambda - \mu))/(e^{-i\theta_1} + e^{-i\theta_2} - 4 + h^2\mu) \\ &+ h^2(\Lambda^k - \lambda)B_\theta/(e^{-i\theta_1} + e^{-i\theta_2} - 4 + h^2\mu).\end{aligned} \tag{5.8}$$

If we define

$$C_\theta = h^2(\Lambda^k - \lambda)B_\theta/(e^{i\theta_1} + e^{i\theta_2} + e^{-i\theta_1} + e^{-i\theta_2} - 4 + h^2\lambda),$$

then we obtain

$$\left|\frac{\bar{A}_\theta - C_\theta}{A_\theta - C_\theta}\right| = |e^{i\theta_1} + e^{i\theta_2} + h^2(\lambda - \mu)|/|e^{-i\theta_1} + e^{-i\theta_2} - 4 + h^2\mu|. \tag{5.9}$$

Note that if $h^2(\lambda - \mu)$ and $h^2\mu$ are small, as is generally the case on finer grids with a proper choice of μ, and if C_θ is small compared to A_θ, then the (θ_1, θ_2) component smoothing rate approaches that for the Poisson problem. On finer grids, $h^2(\Lambda - \lambda)$ is small, even when the eigenvalue approximation has not been updated, and for higher

frequencies, the denominator of C_θ is relatively large. In addition, B_θ is generally small since the solution is smooth and has very small high-frequency components. On coarser grids, however, h^2 can be of significant size, so that C_θ is no longer small compared to A_θ, causing a deterioration of the smoothing rate. To reduce $\Lambda - \lambda$ and thereby minimize the affect of C_θ, we update λ on the coarser grids where the Rayleigh quotient is inexpensive to compute.

The experiments reported in this paper used $\mu = 0$, that is, relaxation without a shift. Generally, this works well if the first eigenvalue is small compared to the spectral radius of L^k, which is always the case here. Relaxation with the shift $\mu = \lambda$ would actually yield slightly better smoothing rates in this case, and for some problems, such as for $-\Delta + cI$ with large c, a nonzero shift may be necessary. However, when a shift is used, care must be taken on coarser grids to avoid letting the denominator of (5.9) become too small since this could lead to substantial magnification of high frequencies. A robust algorithm should incorporate a full shift ($\mu = \lambda$) on finer grids and adjust the shift on coarser grids in order to ensure convergence there. This is especially true for the initial stages of obtaining coarse grid eigenvalue approximations and for computation of higher eigenvalues.

6. The full algorithm.

6.1. General discussion. There are several possibilities for extending the method described in § 5 to apply to the computation of the first q eigenvalues and eigenvectors of L. They differ mostly by the degree to which they handle the eigenvector computations simultaneously. For example, a fully simultaneous extension of the method of § 5 would update all eigenvector and eigenvalue approximations together, both in the fine grid iterations and coarse grid corrections. A fully sequential process would attempt to compute to convergence each eigenvector and eigenvalue in turn. Of course, both methods require additional processes to ensure numerical separateness of the approximations so that they do not all converge to the same eigenvectors. This usually amounts to some sort of orthogonalization process that reflects the orthogonality of the eigenvectors themselves.

One attribute of the full simulataneous method is its ability to use Ritz projections with all of its attendant advantages. For example, as with conventional techniques such as the power method (cf. [8]), the emphasis is placed on producing a good approximation to the subspace spanned by the first q eigenvectors of L. This has many subtle advantages, even more than with the conventional uses of Ritz, but the most direct is that convergence of a specific eigenvalue depends now on its separation not from its neighbor but from the $(q+1)$st eigenvalue of L. The major disadvantage with the fully simultaneous method (and one of the main advantages of the fully sequential one) is storage requirements. All vector approximations must be maintained on all levels of the multigrid cycles. The additional storage is up to $\frac{2}{3}$ of the storage needed (in any case) for storing all of the vectors on the finest grid (since on coarser grids both u^k and τ^k are stored).

The method we propose is intermediate to these two extremes. To retain the advantages of each, we carry all vectors simultaneously through the FMG process by maintaining approximations to all of the eigenvectors and by performing orthogonalization and Ritz projections on the currently finest grid, and we proceed sequentially within the cycling scheme by performing in turn a fixed multigrid cycle on each currently finest grid eigenvector approximation. Sequential use of the multigrid cycling process is apparently no less effective than, yet reduces the storage requirements of, the simultaneous approach. The steps of our algorithm are listed in the appendix and loosely described as follows.

6.2. Initial coarse grid approximations. The first step is to obtain coarse grid approximations to as many of the desired eigenvectors as possible. In general, the *correspondence* between eigenfunctions of the continuous operator and eigenvectors of a discrete approximation to that operator is not exact; that is, the eigenvector corresponding to the ith eigenvalue on some L^h may be a closer approximation to the jth eigenfunction of L with $i \neq j$. This may not necessarily cause problems if i and j are less than q. However, if a coarse grid eigenvector is computed which does not correspond to one of the desired eigenvectors on the finer grids, then it is unlikely that further finer grid work will achieve the desired accuracy. This is less likely to happen if the eigenvalues computed on a coarse grid are limited to a fixed part of the spectrum. Thus, on the coarsest grid G^1, we approximate only $c|G^1|$ eigenvalues and eigenvectors, where $|G^1|$ represents the number of interior G^1 grid points and $c<1$. $c=\frac{1}{4}$ is usually safe, although at times a higher value was used in the experiments reported in this paper since the correspondence for our problems was known to be exact.

The relaxation steps for approximating the eigenvalues and eigenvectors on the coarsest grid are the same as for one eigenvalue with the addition of a Gram–Schmidt orthogonalization step after each sweep. Thus, the ith eigenvector approximation is kept orthogonal to u_j^1, $j=1,\cdots,i-1$, and normalized only at convergence. Once all of the eigenvectors on G^1 have been suitably approximated (i.e., we have accepted $\min\{q, c|G^1|\}$ approximations), then a Ritz projection is performed (see § 6.3). The resulting vectors are then cubically interpolated to the next finer grid, G^2.

In general, once starting vectors are cubically interpolated to level l, one multigrid cycle as described in § 5, along with orthogonalization conditions given in § 6.3 below, is then performed on each u_i^l individually. The coarsest grid $j \geqq 1$ used in this cycle for u_i^l is one on which u_i first appeared (i.e., j is the smallest index so that $i \leqq \min\{q, c|G^j|\}$). If $q > c|G^{l-1}|$, then each of the vectors u_i^l, $c|G^{l-1}| < i \leqq \min\{q, c|G^l|\}$, are computed by the coarse grid process described above (i.e., by relaxation sweeps, each sweep being followed by orthogonalization with respect to $u_1^l,\cdots,u_{i-1}^l$). All the vectors are then cubically interpolated to G^{l+1} and the process is continued until $l=m$.

6.3. Ritz projection. As stated before, once the vectors $u_1^l,\cdots,u_q^l$ have been corrected by multigrid cycles on G^l, the subspace $X=\text{span}\{u_1^l,\cdots,u_q^l\}$ is a good approximation to the subspace spanned by the eigenvectors $U_1^l,\cdots,U_q^l$ that we seek. Ritz projection is a process which finds $\bar{u}_1^l,\cdots,\bar{u}_q^l$ in X and $\bar{\lambda}_1,\cdots,\bar{\lambda}_q$ so that the orthogonal projection of such $L^l\bar{u}_i^l-\bar{\lambda}_i^l\bar{u}_i^l$ onto X is zero. This ensures that any eigenvector of L^l contained in X will be found by Ritz projection. More generally, it will determine a basis for X that is closest to the U_i^l in some sense.

To determine the Ritz vectors we first perform a Gram–Schmidt orthonormalization on $u_1^l,\cdots,u_q^l$, resulting in vectors $\tilde{u}_1^l,\cdots,\tilde{u}_q^l$. Then $\mathcal{U}\mathcal{U}^T$ is an orthogonal projection operator onto X, where

$$\mathcal{U}=[\tilde{u}_1^l,\cdots,\tilde{u}_q^l].$$

Any vector in X can thus be written as $\mathcal{U}z$, where z is a q-vector. Letting $\bar{u}_i^l=\mathcal{U}z_i$, then Ritz projection attempts to find z_i, $\bar{\lambda}_i$, $i=1,2,\cdots,q$, so that

$$\mathcal{U}\mathcal{U}^T(L^l\mathcal{U}z_i-\bar{\lambda}_i\mathcal{U}z_i)=0$$

or, since $\mathcal{U}$ is full rank, so that

$$\mathcal{U}^TL^l\mathcal{U}z_i-\bar{\lambda}_iz_i=0.$$

Hence, the Ritz process requires the solution of a $q \times q$ symmetric eigenvalue problem. Since q is small relative to the size of the fine grid, then the work involved is small. Note that the resulting vectors $\bar{u}_i^l$ form an orthonormal set and that the eigenvalues produced are the Rayleigh quotients of these vectors. This follows clearly from the condition that $L^l\bar{u}_i^l - \lambda_i\bar{u}_i^l$ is orthogonal to X and hence $\bar{u}_i^l$.

Ritz projection could be performed on a coarse grid using FAS approximations to the entries in the Ritz matrix, $\mathcal{U}^T L^k L\mathcal{U}$, but it is better to wait until the coarse grid cycles have been completed on the currently finest grid. At that time, the q-dimensional eigenspace approximation has been corrected. Note that no more fine grid work is needed to perform Ritz on the fine grid than on the coarse grid since the FAS approximation to the Ritz matrix involves calculations on the finest grid. An added advantage of using Ritz on the fine grid after the cycles is that the results are generally more accurate.

6.4. Coarse grid corrections. The fine grid problem (4.1) we seek to solve is equivalent to finding the smallest $\Lambda_1^h \leqq \Lambda_2^h \leqq \cdots \leqq \Lambda_q^h$ and $U_1^h, U_2^h, \cdots, U_q^h$ so that for $i = 1, 2, \cdots, q$,

$$L^h U_i^h - \Lambda_i^h U_i^h = 0, \qquad U_i^h = 0 \quad \text{on } \partial\Omega,$$

$$\langle U_i^h, U_j^h \rangle = \delta_{ij}, \qquad j = 1, 2, \cdots, i.$$

Since the Ritz process requires only a numerically well-determined subspace for the eigenspace from which it computes the eigenvalue approximations, the aim of the multigrid cycle performed on the ith eigenvector approximation is to produce a vector which differs from U_i^h only in the directions of the vectors U_j^h, $j \leqq q$. Thus strict orthogonalization is unnecessary, and we can instead try to maintain the amount of separation that already exists between the vectors. We will refer to these separation constraints as orthonormalization conditions, although it should be understood that these only approximate true orthonormality. The multigrid cycle for the ith vector will tend to produce an approximation to the vector with minimum Rayleigh quotient which satisfies these separation constraints. That is, components of higher eigenvectors are eliminated from the approximation, and those of the previously computed eigenvector approximations remain unchanged. Relaxation changes low-frequency error components only very slightly, and since these components dominate the approximation, then vector norms and the amount of vector separation are approximately maintained during relaxation on the fine grid. Thus normalization and orthogonalization may be reserved for use on coarse grids only, where low-frequency changes will occur.

Orthogonalization is therefore performed only on the very coarsest grids, with the exception that at the termination of each multigrid cycle on the current finest grid G^l, Gram–Schmidt orthonormalization is used to start the Ritz process. Thus, during each cycle, the separation exhibited by the fine grid vectors results only from them being the interpolants of the orthonormal grid $l-1$ approximants. This separation is, however, adequate to provide good numerical determination of the subspace in which they belong. Thus, our FAS orthonormalization conditions are designed to maintain this separation rather than the much more difficult task of maintaining actual orthogonality. The (approximate) orthonormalization constraint, analogous to that for normalization in (5.3), is given by

$$\langle u_j^k, I_{k+1}^k u_i^{k+1} \rangle = \sigma_{ij}^k, \qquad i = 1, \cdots, j, \tag{6.1}$$

where $\sigma_{ij}^k = \langle I_{k+1}^k u_j^{k+1}, I_{k+1}^k u_i^{k+1} \rangle$. Note that the case $i = j$ is the normalization constraint. Then the coarse grid equations are, for $i = 1, 2, \cdots, q$,

$$\begin{aligned} L^k U_i^k - \Lambda_i^l U_i^k &= I^k, \\ \langle U_i^k, I_{k+1}^k U_j^k \rangle &= \sigma_{ij}^k. \end{aligned} \tag{6.2}$$

Relaxation on the coarse grids tends to increase the components in the direction of previous eigenvectors, so for $i < j$, (6.1) is enforced by subtracting a multiple of $I_l^k u_i^l$ from u_j^k, and the equation for $i = j$ is enforced by multiplication by some constant. Since $\sigma_{ij}^k \neq 0$ for $i < j$, unlike true Gram–Schmidt orthogonalization, (6.1) is not exactly satisfied after one pass through the equations. However, experiments show that further work to enforce (6.1) is neither needed nor helpful.

We rely mainly on the coarse grid solution process to eliminate error components of u_i^l in the direction of U_{q+1}^l and other higher low frequencies. Since relaxation on intermediate grids does not significantly affect these frequencies, the approximate action of coarse grid solution on a component U_j is given by

$$I_1^l (L^1 - \Lambda_i^l I)^{-1} (L^1 I_l^1 - I_l^1 L^l) U_j^l.$$

Ignoring grid transfer errors, this means that the U_j^l component in u_i^l is approximately multiplied by γ_j, where

$$\gamma_j = \frac{\Lambda_j^1 - \Lambda_j^l}{\Lambda_j^1 - \Lambda_i^l}.$$

If γ_{q+1} is less than 1 for all $i \leqq q$, then the eigenspace approximation will improve since higher frequencies are reduced at a greater rate. However, this quantity generally depends on q and i and the type of discretization. For programming convenience in the work reported in this paper, 5-point stencil discrete operators were used on all grids. In this case, for a positive definite operator, the discrete eigenvalue approximations generally decrease as the mesh size increases. Thus, for some q, γ can be greater than 1 when coarse grid truncation error exceeds the eigenvalue separation $\Lambda_{q+1}^l - \Lambda_q^l$. For such problems, a variational formulation of the problem may be useful. Here, the coarse grid and grid transfer operators satisfy, up to a scalar factor,

1. $A^{2h} = I_h^{2h} A^h I_{2h}^h$,
2. $I_h^{2h} = I_{2h}^{hT}$.

Using such problem formulation yields discrete eigenvalues which decrease as h decreases. Thus if $\Lambda_i^l < \Lambda_j^l$ then

$$\gamma_j = \frac{\Lambda_j^1 - \Lambda_j^l}{\Lambda_j^1 - \Lambda_i^l} < \frac{\Lambda_j^1 - \Lambda_j^l}{\Lambda_j^1 - \Lambda_j^l} = 1.$$

This avoids any coarse grid anomalies and ensures coarse grid convergence, even when these grids are very coarse.

The coarsest grid used in multigrid cycling, when G is the currently finest grid in the FMG process, should be G^k, where k is $\min\{j: q \leqq c|G^j|\}$. This ensures that relaxation is not performed on levels for which the G^l eigenvectors we seek do not correspond to those on the coarse grid. On too coarse a level, eigenvalue ordering can be different than on G^l, and relaxation there can introduce error in the direction of higher G^l eigenvectors. Another reason not to go to such a coarse grid, even for low eigenvalues, is that most of the correction provided there will be in the direction of eigenvectors belonging to the subspace that we are trying to approximate. More specifically, if $|G^1|$ is not large compared to q, then there is only a negligible change

in the approximate subspace X on G^m due to the corrections computed on G^1. Use of the Ritz process allows us to take this view that we are concerned only with errors in the subspace approximation, not the eigenvectors themselves.

6.5. An alternate accuracy criterion. The results and methods so far have been geared towards finding q eigenvalues of L to within truncation error defined by a prespecified grid mesh size h. Truncation error generally increases with λ, so the same accuracy is not achieved for all eigenvalues. If one wishes to compute the first q eigenvalues of L to within some given tolerance ε, limited experiments indicate that this can be achieved without performing relaxation on the eigenvalues that have already been accepted because they are accurate to that tolerance. These accepted eigenvectors need not even be included in the Ritz process, although they should be carried to the finer grids demanded by the higher eigenvalues and used in the coarse grid and pre-Ritz orthogonalization processes there. Also, if for some reason the discrete eigenvalues are wanted to some level of tolerance smaller than truncation error for a fixed h, we can make the first cycle for all vectors as before, with more work (i.e., further cycles) performed only for higher vectors, using the lower ones again in orthogonalization only.

In both of the above cases, if all vectors are used in the Ritz projection, the convergence rates for the higher eigenvectors are close to those obtained by the more expensive algorithm that includes the lower eigenvalues in all processes. In the second case, the rates are almost identical.

7. Work and storage requirements. Since normalization, orthogonalization and computation of the orthonormalization correction terms and the Rayleigh quotient are performed primarily on coarser grids, we neglect these processes in developing the following overall work estimates. For convenience, let $n_k = |G^k|$ and assume $n_k = \frac{1}{4} n_{k+1}$ for $k = 1, 2, \cdots, m-1$. Let αn_k represent the number of operations in one G^k relaxation sweep. This is essentially the cost of one matrix multiply. α is approximately twice the number of nonzero entries per row in L^l. (In some cases, such as a uniform discretization of the Laplacian, α is much smaller than this.) An inner product on G^k is assumed to cost $2n_k$ operations. The amount of work involved at one stage of the FMG process results from the following computations:

1. Cubic interpolation of u_i^{l-1} to u_i^l for $i = 1, \cdots, q$.
2. $\nu_1 + \nu_2$ relaxation sweeps on u_i^k, $i = 1, \cdots, q$, $k = 1, \cdots, l$.
3. Coarse grid τ_i^k calculation for $k = 1, \cdots, l-1$, $i = 1, \cdots, q$.
4. FAS interpolation of u_i^{k-1} to u_i^k, $i = 1, \cdots, q$, $k = 2, \cdots, l$.
5. Ritz projection on G^l, which includes:
   a. orthonormalization of $u_1^l, \cdots, u_q^l$;
   b. compilation of the Ritz matrix $[u_i^{l\,T} L^l u_j^l]_{i,j}$, $i, j = 1, \cdots, q$;
   c. solving for all the eigenvectors and eigenvalues of the Ritz matrix;
   d. computing the new vectors $\bar{u}_1^l, \bar{u}_2^l, \cdots, \bar{u}_q^l$.

The work involved in each of these processes is measured as follows:

1. $\frac{15}{4} n_l \cdot q$.
2. $\frac{4}{3}\alpha(\nu_1 + \nu_2) n_l \cdot q$.
3. $\frac{8}{3}(\alpha+1) n_{l-1} q = \frac{2}{3}(\alpha+1) n_l \cdot q$.
4. $\frac{11}{4} n_l \frac{4}{3} \cdot q = \frac{11}{3} n_l \cdot q$.
5. a. $(2q^2 + q) n_l$;
   b. $\alpha q \cdot n_l + q(q+1) n_l$;
   c. $O(q^3)$;
   d. $q(2q-1) n_l$.

This gives a total G^l operation count, W_l, of

$$W_l = \tfrac{4}{3}(\nu_1+\nu_2+\tfrac{5}{4})q\alpha n_l + [\tfrac{109}{12}q + 5q^2]n_l + O(q^3).$$

The total operation count, $W = \sum_{l=1}^{m} W_l$, for the complete FMG algorithm is therefore

$$W \approx \tfrac{16}{9}(\nu_1+\nu_2+\tfrac{5}{4})q\alpha n_m + [12q + \tfrac{20}{3}q^2]n_m + mO(q^3).$$

Note that m should be proportional to $\log n_m$ so that

$$W = n_m O(q^2) + \log n_m O(q^3).$$

Storage is needed for all vectors on all grids, although future work on small-storage algorithms (see [2, § 7.5]) may drastically reduce that requirement. Since coarse grid problems are inhomogeneous, storage is also required for the right-hand side of these problems. However, the sequential manner in which the coarse grid cycles are performed means that only one vector per grid is needed to store the right-hand side τ^k. The finest grid actually needs no storage for this, since the problem is homogeneous. All other storage is at most proportional to the product of the number of points in one direction on the fine grid and the number of eigenvalues being computed. Thus, disregarding locations needed for bookkeeping, the total storage location requirements are approximately

$$S = \tfrac{4}{3}qn_m + \tfrac{1}{3}n_m.$$

8. Computational results. The model problem used in our experiments is given by

$$\begin{aligned} -\Delta u + 10y(\sin 3\pi x)u &= \lambda u \quad \text{on } \Omega = [0,1]\times[0,1], \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{8.1}$$

The Laplace operator by itself has several properties not typical of more general operators which made it unsuitable for reliable tests. The term added to the Laplacian in (8.1) causes the multiple eigenvalues of the Laplace operator to be perturbed, yielding instead sets of close eigenvalues. This poses more of a challenge to our algorithm. Moreover, the eigenvectors are altered so that they are not exactly represented by the discrete problem as they are for the Laplacian alone.

Unless otherwise indicated, $h_1 = \frac{1}{4}$ and $m = 4$ so that $h_m = \frac{1}{32}$. Also, $\nu_1 = \nu_2 = 2$.

A closed form solution for the eigenvalues and eigenvalue discretization error of this problem is not known, but can be closely approximated by carrying out the test results farther than the number of cycles and the fineness of h_m than we report. The "exact" values Λ_n^0 used in Table 1 were calculated by extrapolating the computed solutions on levels $m = 4$ and $m = 5$. h in Table 1 represents $h = h_4 = \frac{1}{32}$.

TABLE 1

n	Λ_n^0	Λ_n^h	$\lvert\Lambda_n^0 - \Lambda_n^h\rvert$	$\lVert L^h U_n^0 - \Lambda_n^0 U_n^0\rVert$
1	18.73558161	18.71847149	.0171	.0407
2	48.32534796	48.18927363	.136	.142
3	51.69556290	51.56004355	.136	.148
4	81.32645700	81.07201016	.254	.263
5	97.65037417	97.00117915	.649	.651
6	100.2221931	99.57484220	.647	.654
7	129.8746755	129.1084354	.766	.773
8	130.6674040	129.8996943	.768	.771
9	166.65623	164.6376509	2.02	2.02
10	169.0329	167.0085449	2.02	2.03

The last column in Table 1 is the norm of the residual of the solution to the continuous problem projected onto the $h = \frac{1}{32}$ grid. This is also an approximation, but is close enough to serve as a quantity against which to measure convergence of the discrete solution. That is, when the grid h residual is comparable to the residual formed by the projected continuous solution, then we conclude that the discrete error is the same order as the truncation error. The algebraic error in the eigenvalue is not necessarily a good measure of the algebraic error in the eigenvector, the former being approximately proportional to the square of the latter.

The method for one eigenvalue and $h = \frac{1}{32}$ is given in Table 2. The amount of work performed in relaxation, which dominates the overall work in this case, is equivalent to about seven sweeps on the fine grid.

TABLE 2

λ_1^h	$\lvert\lambda_1^h - \Lambda_1^h\rvert$	$\lVert L^h u_1^h - \Lambda_1^h u_1^h\rVert$
18.71871030	2.39×10^{-4}	1.40×10^{-2}

The results for ten eigenvalues and $h = \frac{1}{32}$ are shown in Table 3. A comparison with Table 1 shows that the problem is solved to below the level of truncation error. Note that the accuracy is better relative to this truncation error for the lower eigenvalues. If more eigenvectors are included in the process (whether or not they converge), all approximations are improved. When 15 eigenvectors were included in a test that we ran, although the last two failed to converge, the residual norm of the 10th eigenvector decreased by about a factor of 2. This is not of too much practical importance, however, since even though it suggests that less work is required for the lower eigenvalues when Ritz projection is used, the error decrease is actually due mainly to elimination of low-frequency error.

TABLE 3

n	λ_n^h	$\lvert\lambda_n^h - \Lambda_n^h\rvert$	$\lVert L^h u_n^h - \lambda_n^h u_n^h\rVert$
1	18.71847153	3.40×10^{-8}	4.26×10^{-3}
2	48.18927456	9.31×10^{-7}	2.04×10^{-2}
3	51.56004444	8.90×10^{-7}	2.32×10^{-2}
4	81.07201416	4.00×10^{-6}	3.80×10^{-2}
5	97.00123840	5.93×10^{-5}	1.64×10^{-1}
6	99.57489151	4.93×10^{-5}	1.56×10^{-1}
7	129.1088552	4.20×10^{-4}	2.64×10^{-1}
8	129.9001827	4.88×10^{-4}	2.77×10^{-1}
9	164.6602169	2.26×10^{-2}	1.74
10	167.0701555	6.16×10^{-2}	1.72

As explained in § 6, low-frequency error can arise if the coarse grid solution does not provide a good correction to the approximations on finer grids. In these cases a cycle with more visits to coarser grids may help. Table 4 gives the results for the so-called "W-cycle" for ten eigenvalues. A W-cycle means that the coarse grid correction to any grid k is calculated by *two* cycles on grid $k-1$. The amount of work performed in relaxation is $\frac{3}{2}$ times the amount needed for usual multigrid cycles. The total amount of work is then

$$W \approx \tfrac{8}{9}(3(\nu_1+\nu_2)+\tfrac{5}{2})q\alpha n_m + [12q + \tfrac{20}{3}q^2]n_m + mO(q^3).$$

TABLE 4

n	$\lvert\Lambda_n^h-\lambda_n^h\rvert$	$\lVert L^h u_n^h-\lambda_n^h u_n^h\rVert$
1	3.60×10^{-8}	4.21×10^{-3}
2	6.57×10^{-7}	1.83×10^{-2}
3	6.55×10^{-7}	2.12×10^{-2}
4	2.34×10^{-6}	3.10×10^{-2}
5	3.97×10^{-5}	1.47×10^{-1}
6	3.27×10^{-5}	1.40×10^{-1}
7	4.68×10^{-5}	1.53×10^{-1}
8	5.29×10^{-5}	1.58×10^{-1}
9	2.36×10^{-3}	7.15×10^{-1}
10	1.42×10^{-2}	8.83×10^{-1}

Tables 5 and 6 show the results obtained with red-black ordering in place of lexicographic Gauss–Seidel with usual cycling and W-cycling, respectively. This type of ordering is a more effective smoother than lexicographic, as evidenced by the higher accuracy obtained in the eigenvalues.

TABLE 5

n	$\lvert\lambda_n^h-\Lambda_n^h\rvert$	$\lVert L^h u_n^h-\lambda_n^h u_n^h\rVert$
1	1.23×10^{-8}	2.98×10^{-3}
2	8.40×10^{-8}	2.39×10^{-2}
3	1.03×10^{-7}	2.33×10^{-2}
4	5.54×10^{-7}	6.50×10^{-2}
5	5.64×10^{-6}	1.89×10^{-1}
6	8.51×10^{-6}	2.57×10^{-1}
7	1.44×10^{-5}	3.32×10^{-1}
8	2.72×10^{-5}	2.88×10^{-1}
9	9.84×10^{-3}	1.05
10	1.16×10^{-1}	1.96

TABLE 6

n	$\lvert\lambda_n^h-\Lambda_n^h\rvert$	$\lVert L^h u_n^h-\lambda_n^h u_n^h\rVert$
1	4.45×10^{-9}	1.65×10^{-3}
2	2.96×10^{-8}	1.35×10^{-2}
3	3.31×10^{-8}	7.48×10^{-3}
4	7.64×10^{-8}	2.08×10^{-2}
5	9.79×10^{-7}	6.19×10^{-2}
6	1.17×10^{-5}	3.00×10^{-1}
7	1.31×10^{-5}	3.18×10^{-1}
8	1.77×10^{-6}	9.22×10^{-2}
9	2.60×10^{-3}	4.40×10^{-1}
10	3.87×10^{-2}	1.31

The nature of the FMG process is ideal for the use of extrapolation on the eigenvalues approximations. Table 7 shows the eigenvalue approximations extrapolated from the $\frac{1}{16}$ and $\frac{1}{32}$ grids and the accuracy obtained.

TABLE 7

n	$(4\lambda_n^h - \lambda_h^{2h})/3$	$\lvert\Lambda_n^0 - (4\lambda_n^h - \lambda_n^{2h})/3\rvert$
1	18.73554270	3.89×10^{-5}
2	48.32469250	6.55×10^{-4}
3	51.69476513	7.98×10^{-4}
4	81.32502517	1.43×10^{-3}
5	97.64234667	8.03×10^{-3}
6	100.2140080	8.19×10^{-3}
7	129.8582268	1.64×10^{-2}
8	130.6508394	1.66×10^{-2}
9	166.4985283	1.58×10^{-1}
10	168.8655123	1.67×10^{-1}

Appendix. The algorithm is broken into two parts: the FMG Ritz procedure and the CYCLE procedure. CYCLE is called from FMG Ritz. The parameters of the algorithm are as follows:

q: The number of eigenvalues and eigenvectors desired.

m: The number of grids to be used.

ν_0: The number of iterations used to obtain a first approximation to each vector on the coarsest grid.

ν_1: The number of relaxation sweeps performed before transferring the problem to a coarser grid.

ν_2: The number of sweeps performed after the coarse grid correction.

FMG RITZ.

1. Set $n \leftarrow 1$, n max $\leftarrow 1$, $l \leftarrow 1$, $\lambda_0 \leftarrow 0$
2. Set $u_n^l \leftarrow$ Random function $\lambda_n \leftarrow \lambda_{n-1}$
   For $i = 1, 2, \cdots, \nu_0$, do
   $u_n^l \leftarrow$ Relax $(L^l u_n^l - \lambda_n u_n^l = 0)$
   For $j = 1, 2, \cdots, n-1$, do
   $u_n^l \leftarrow u_n^l - \langle u_n^l, u_j^l\rangle u_j^l$
   Set $\lambda_n \leftarrow \langle Lu_n^l, u_n^l\rangle / \langle u_n^l, u_n^l\rangle$
   Set $u_n^l \leftarrow u_n^l / \langle u_n^l, u_n^l\rangle$.
3. If $n = q$, set n max $= q$, k min $= l$,
   Ritz on u_i^l, $i = 1, 2, \cdots, n$ max & go to 4
   If $n < c|G^1|$, set $n \leftarrow n+1$ & go to 2.
   Set n max $\leftarrow n$, k min $\leftarrow l$ & go to 4.
4. If $l = m$, stop.
   Set $l \leftarrow l+1$
   For $i = 1, 2, \cdots, n$ max, do
   $u_i^l \leftarrow I_{l-1}^l u_i^{l-1}$
   cycle $(i, k$ min, $l)$
   Ritz on u_i^l, $i = 1, 2, \cdots, n$ max.
5. If n max $< q$, set $n \leftarrow n$ max $+1$ & go to 2.
   Otherwise go to 4.

CYCLE $(n, k$ min, $l)$.

1. Set $k \leftarrow l$, $\tau_n^k \leftarrow 0$
2. For $i = 1, 2, \cdots, \nu_1$, do
   $u_n^k \leftarrow$ Relax $(L^k u_n^k - \lambda_n u_n^k = \tau_n^k)$
   If $k \leqq k$ min

For $j = 1, 2, \cdots, n-1$, do

$u_n^k \leftarrow u_n^k - (\langle u_n^k, I_l^k u_j^l\rangle - \langle I_{k+1}^k u_n^{k+1}, I_l^k u_j^l\rangle)/\langle I_l^k u_j^l, I_l^k u_j^l\rangle \cdot I_l^k u_j^l$

Set $u_n^k \leftarrow u_n^k \langle I_{k+1}^k u_n^{k+1}, I_l^k u_n^l\rangle / \langle u_n^k, I_l^k u_n^l\rangle$

If $k = k$ min, set $\lambda_n \leftarrow \langle L^k u_n^k - \tau_n^k, u_n^k\rangle / \langle u_n^k, u_n^k\rangle$.

3. If $k = k$ min, go to 5.
4. Set $k \leftarrow k-1$

$\tau_n^k = \tau_n^{k+1} + L^k I_{k+1}^k u^{k+1} - I_{k+1}^k L^{k+1} u^{k+1}$

$u_n^k \leftarrow I_{k+1}^k u_n^k$

Go to 2.

5. For $i = 1, 2, \cdots, \nu_2$, do

$u_n^k \leftarrow \text{Relax}\,(L^k u_n^k - \lambda_n u_n^k = \tau_n^k)$

If $k \leqq k_1$

For $j = 1, 2, \cdots, n-1$, do

$u_n^k \leftarrow u_n^k - (\langle u_n^k, I_l^k u_j^l\rangle - \langle I_{k+1}^k u_n^{k+1}, I_l^k u_j^l\rangle)/\langle I_l^k u_j^l, I_l^k u_j^l\rangle \cdot I_l^k u_j^l$

Set $u_n^k \leftarrow u_n^k \langle I_{k+1}^k u_n^{k+1}, I_l^k u_n^l\rangle / \langle u_n^k, I_l^k u_n^l\rangle \cdot I_l^k u_j^l$

If $k = k$ min, set $\lambda_n \leftarrow \langle L^k u_n^k - \tau^k, u_n^k\rangle / \langle u_n^k, u_n^k\rangle$.

6. If $k = l$, stop.
7. Set $k \leftarrow k+1$

$u_n^k \leftarrow u_n^k + I_{k-1}^k (u_n^{k-1} - I_k^{k-1} u_n^k)$

Go to 5.

REFERENCES

[1] R. G. ALCOUFFE, A. BRANDT, J. E. DENDY, JR. AND J. W. PAINTER, *The multi-grid methods for the diffusion equation with strongly discontinuous coefficients*, Report LA-UR 80-1463, Los Alamos Scientific Laboratory, Los Alamos, NM, 1980.

[2] A. BRANDT, *Multi-level adaptive solutions to boundary value problems*, ICASE Report RC 76-27 1976, Hampton, VA, NASA Langley Research Center; Math. Comp., 31 (1977), pp. 336–390.

[3] A. Brandt, *Guide to multigrid development*, in Multigrid Methods, proceedings of a conference (Köln-Porz, No., 1981), W. Hackbusch and U. Trattenberg, eds. Springer-Verlag, New York, 1982.

[4] D. S. CLEMM AND K. G. GUDERLEY, *Eigenvalue and near eigenvalue problems solved by Brandt's multigrid method*, Univ. of Dayton Research Institute Interim Report AFFDL-TR-79-3147, Dayton, OH, 1979.

[5] W. HACKBUSCH, *Multigrid solutions to linear and nonlinear eigenvalue problems for integral and differential equations*, Report 80-3, Math. Inst. Univ. Köln, 1980.

[6] W. KAHAN, *Relaxation methods for an eigenproblem*, Stanford Technical Report AD-638818, Stanford Univ., Stanford, CA, 1966.

[7] S. F. MCCORMICK, *A mesh refinement method for* $Ax = \lambda Bx$, Math. Comp., 36 (1981), pp. 485–498.

[8] S. F. MCCORMICK AND T. NOE, *Simultaneous iteration for the matrix eigenvalue problem*, J. Lin. Alg. Appl., 16 (1977), pp. 43–56.

[9] L. STRAKHOVSKAYA, *An iterative method for evaluating the first eigenvalue of an elliptic operator*, USSR Comput. Math. and Math. Phys., 17 (1978), pp. 88–101.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0010 $01.25/0

SYSTOLIC NETWORKS FOR ORTHOGONAL DECOMPOSITIONS*

DON E. HELLER† AND ILSE C. F. IPSEN†

Abstract. An orthogonally connected systolic array, consisting of a few types of simple processors, is constructed to perform the QR decomposition of a matrix. Application is made to solution of linear systems and linear least squares problems as well as QL and LQ factorizations. For matrices A of bandwidth w the decomposition network requires less than w^2 processors, independent of the order n of A. In terms of the operation time of the slowest processor, computation time varies between $2n$ and $4n$ subject to the number of codiagonals.

Key words. systolic array, VLSI, QR decomposition, linear systems, least squares

Introduction. Systolic arrays are an architectural paradigm first proposed by Kung and Leiserson [11], [12] for the implementation of matrix operations in VLSI (very large scale integrated circuits). The object is to design special purpose devices which take advantage of new and emerging computer circuit technologies. A general overview of the design activity is given by Kung [9], [10].

A systolic array is a collection of parallel processors arranged, by their interconnections, into a regular planar network. The array operates in a pipeline mode. Data enter the processor array through its periphery, and (for the purposes of this paper) circulate through the array along fixed paths in synchrony with a global clock signal. Results emerge from the processor array along its periphery, at the same rate as data are entered; in some designs results may be accumulated for output after input is completed. Practical considerations for VLSI design require the processors to be simple and identical; those on the periphery may differ from those in the interior of the array, however.

This paper describes a systolic array to compute the QR factorization of a banded matrix, in the spirit of designs presented in [11], [12] for other band matrix computations. The LU factorization systolic array of [11], [12] implements Gaussian elimination without pivoting; modifications to support both pivoting and similar operating characteristics (time, I/O format) do not exist. Essentially, a high degree of data-dependent control for pivoting would have to be implemented by the network, violating some of the constraints on VLSI design. Here, the numerically stable Givens plane rotation [5] is repeatedly applied to obtain a reduction to triangular form.

Chaining three simple types of processors, which perform only basic arithmetic operations, produces a linearly connected mesh which is the integral component of the network for QR, QL and LQ decompositions, and hence solution of linear systems and linear least squares problems. Because the bandwidth, w, of a matrix A can be preserved through a proper order of elimination, the number of processors in our array is less than w^2 and independent of n, the order of A. Taking the operation time of the slowest processor as a unit, the total computation time varies between $2n$ and $4n$ subject to the number of codiagonals to be eliminated. The proposed networks follow the data I/O regimen established in [11], [12] for band matrices and are suited for implementation in VLSI as part of a family of special purpose devices.

After comments on plane rotations as well as exploitation of parallelism in orthogonal decompositions, appropriate processors are defined and the network for

* Received by the editors September 21, 1981, and in revised form May 10, 1982. This work was supported, in part, by the National Science Foundation under grant MCS 78-09126.

† Computer Science Department, The Pennsylvania State University, University Park, Pennsylvania 16802.

the QR decomposition is presented. Different indexing of matrix elements enables the QL factorization and a reversion of datalines yields a network for the LQ factorization. Addition of a processor incorporating a delay element allows solution of linear least squares problems following the QR factorization. Finally, some remarks concerning a VLSI implementation are made. An indication of how to process matrices too wide for available hardware will be made in a subsequent paper.

There are three related designs for the QR factorization by other authors. Bojancyzk, Brent and Kung [2] and Ahmed, Delosma and Morf [1] (cf. Gannon [3]) describe a systolic array for dense square matrices, storing Q in the array and propagating R; A is entered (essentially) one column per step. Gentleman and Kung [6] describe another systolic array for dense rectangular matrices, storing R in the array and propagating Q; A is entered (essentially) one row per step. Both designs are based on Givens rotations; the latter can be modified to solve least squares problems on the fly. The design presented here propagates both Q and R, making it more economical for band matrices. Johnsson [8] discusses a VLSI processor array implementing the Householder triangularization for band matrices, using w processors and $O(nw)$ time. Our design, when reduced to w processors, would also use $O(nw)$ time (cf. [1]), but the necessity to accumulate an inner product prevents the Householder reduction's effective use of more than w processors: w^2 processors would need $O(n \log w)$ time.

The final comparison of designs will depend on implementation issues not yet resolvable.

Plane rotations. The standard Givens rotation is a computationally stable device for introducing zeros into a matrix by premultiplication [14], [15]. Let

$$P = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad c^2 + s^2 = 1,$$

$$P \begin{pmatrix} x_1 & x_2 \cdots x_k \\ y_1 & y_2 \cdots y_k \end{pmatrix} = \begin{pmatrix} x_1' & x_2' \cdots x_k' \\ 0 & y_2' \cdots y_k' \end{pmatrix}, \qquad k \geqq 1.$$

The elimination is effected by

$$\text{(1)} \qquad \begin{aligned} &\text{if } y_1 = 0 \text{ then } x_1' = x_1,\ c = 1,\ s = 0 \\ &\qquad \text{else } x_1' = (x_1^2 + y_1^2)^{1/2},\ c = x_1/x_1',\ s = y_1/x_1', \end{aligned}$$

$$\text{(2)} \qquad x_i' = cx_i + sy_i, \quad y_i' = -sx_i + cy_i, \quad 2 \leqq i \leqq k.$$

Equations (1) should be modified for implementation, to avoid over- and underflow. Fast Givens rotations [4], [7] complicate data manipulation for VLSI implementation and are not considered here. By means of a few shifts and comparisons, P can be stably encoded into a single number [16]. In the sequel, P will denote either the matrix or its encoding.

Exploitation of parallelism. Considering a full matrix as an initial portion of an appropriate band matrix, attention can be restricted to band matrices A of order n. The size of the linearly connected mesh to follow depends on the bandwidth of A, $w = p + q + 1$ (q subdiagonals, p superdiagonals, $1 \leqq p,\ q \leqq n$), but not the length of the band, n.

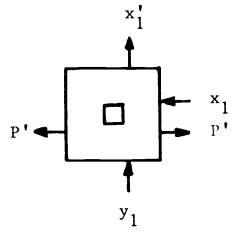
Arriving at the same conditions as in [13] and [14], subdiagonal elements y_1 and $\tilde{y}_1$ can be simultaneously eliminated by premultiplication, if

(a) y_1 and $\tilde{y}_1$ are in nonadjacent rows, and

(b) all elements to the left of $(x_1 y_1)^T$ and $(\tilde{x}_1 \tilde{y}_1)^T$, respectively, as well as below y_1 and $\tilde{y}_1$, respectively, are zero.
This means that annihilation proceeds subdiagonalwise from without toward the diagonal, implementing a *QR* factorization of A.

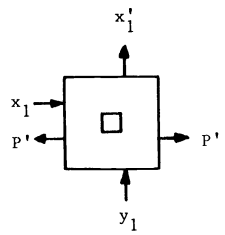
Processors. A synchronized network is assumed where the time between clock pulses is long enough to perform any required operation. Two kinds of processors are needed for the *QR* decomposition, one that generates plane rotations (see (1)) and a second which propagates them (cf. (2)).

Cell 1.



equations (1).

Cell 2.



equations (2).

In the absence of input, all datalines will assume the value 0 and consequently plane rotations will yield identity. P_{ij} denotes the rotation which eliminates a_{ij}. Primes denote values computed in the cell, so if x_i, y_i and P enter cell 2 at time t, then x_i', y_i' and the unchanged P exit cell 2 at time $t+1$ and are then available for input to another cell.

The linearly connected mesh. A linearly connected mesh of w processors elimnates the qth subdiagonal of A, resulting in a matrix A'. The bandwidth is preserved by fill-in of a $(p+1)$st superdiagonal. The product of plane rotations flowing to the right constitutes the matrix Q^T. Dataflow in and out of a particular processor occurs by codiagonals, so that succeeding codiagonal elements enter (and leave) every other time step. Figure 1 illustrates the mesh with appropriate dataflow.

If we index cells from left to right by $-q \cdots p$, then element $a(j, j+k)$, $-q \leqq k \leqq p$, arrives at time $|k|+2j-1$ as input to cell k. The updated value exits from cell $k-1$ two steps later.

Elimination of a codiagonal takes time $2n$, provided a_{11} enters at $t=1$. As for speedup and efficiency the same results as in [13] are obtained, indicating superiority over Gaussian elimination with pivoting, the Gram–Schmidt orthonormalization process and the Householder triangularization.

The *QR* decomposition. An array of q of the above meshes, comprising qw processors, performs the reduction $R = Q^T A$, where the lowermost mesh removes the qth subdiagonal and the uppermost one the first subdiagonal. The elements of R leave the top of the array. Delivering a_{11} at $t=1$, r_{11} is available at $t=2q$, bringing the total computation time to $T(n, q) = 2(n+(q-1))$. See the appropriate portion of Fig. 3.

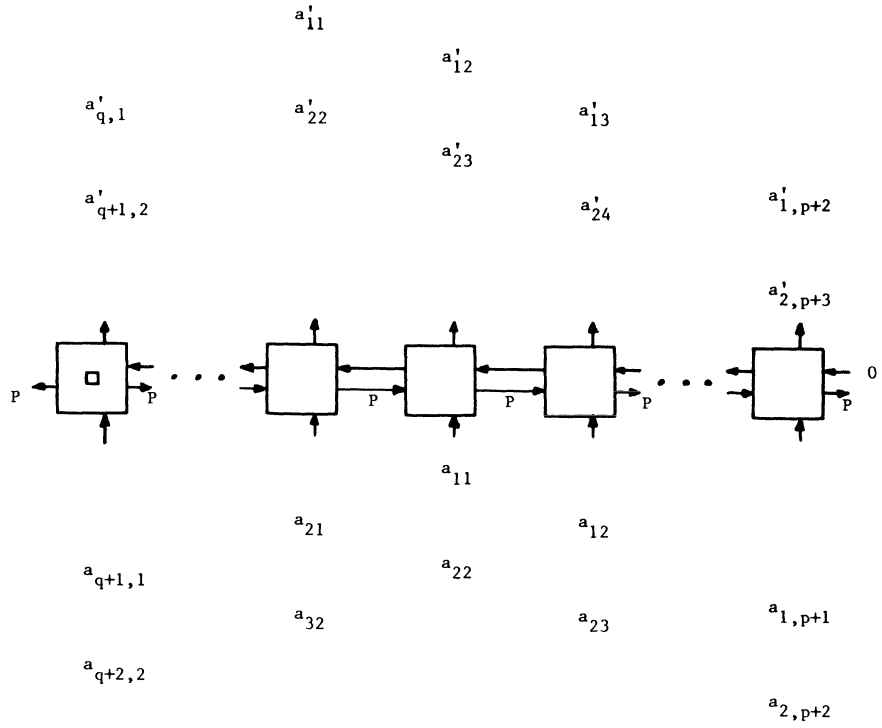


FIG. 1. *Linearly connected mesh, for elimination of qth subdiagonal, with* I/O *format.*

The *QL* and *LQ* decompositions. A different addressing scheme for matrix elements allows the *QR* network to be employed for the *QL* factorization. Input data consists of a matrix $\bar{A}$ which results from reversing the row and column order of the original matrix A. Let $J \in \mathbb{R}^{n\times n}$, $j_{ik} = \delta_{k,n-i+1}$ for $1 \leqq i, k \leqq n$, be a permutation matrix. Then

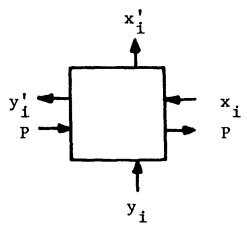
$$\bar{A} = JAJ = QR \quad \text{and} \quad A = (JQJ)(JRJ) = \bar{Q}L.$$

$p \cdot w$ processors are necessary, as is also a computation time of $T(n, p)$. Regarding the solution of $Ax = b$, one has to:

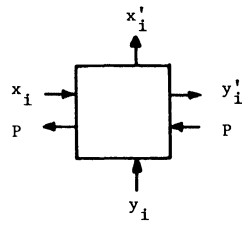
1. Determine $A = \bar{Q}L$.
2. Solve $L(Jx) = (\bar{Q})^T(Jb)$.

If it is not a question of having an upper or lower triangular matrix, the decomposition requiring the least resources should be applied. For example, for $p \ll q$, compute $A = \bar{Q}L$ where the number of processors $pw \ll qw$ and the time $T(n, p) < T(n, q)$.

As for the *LQ* factorization, superdiagonal elements are annihilated from without toward the diagonal by postmultiplications, and therefore the datalines of processors are reversed so as to join adjacent column elements:

Cell 1'.  equations (1).

Cell 2'.



equations (2).

The corresponding linear mesh is shown in Fig. 2. Processor and time requirements for the LQ array are the same as for the QL array. Routing of matrix elements, however, occurs as in the QR factorization.

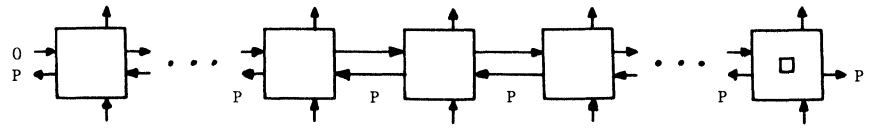


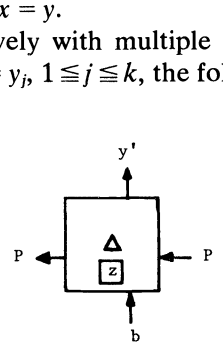
FIG. 2. *Linearly connected mesh for elimination of pth superdiagonal (cf. Fig. 1).*

Linear systems and least squares computations. A vector x has to be determined that minimizes $\|Ax-b\|_2$; for simplicity first assume A is square and nonsingular, so $Ax=b$. If $R=Q^TA$ is available then R is nonsingular [15] and $Rx=y$ for $y=Q^Tb$. Hence, two steps are necessary:

1. Determine $R=Q^TA$ and $y=Q^Tb$.
2. Solve $Rx=y$.

To deal effectively with multiple least squares problems, x_j minimizing $\|Ax-b_j\|_2$, $y_j=Q^Tb_j$, $Rx_j=y_j$, $1\leqq j\leqq k$, the following cell, which effects a delay, is needed

Cell 3.



input b, P
  local memory $=z$
compute $y'=cz+sb$
  $z'=-sz+cb$
output y', P
  local memory $=z'$

The value z' is retained as z for the next operation of the cell. Initially, z is 0. In Figs. 3 and 4 the local memory is not displayed.

The network in Fig. 3 for the case $q=2$ serves as an illustration for the general case. The first element of b_j, b_{1j}, enters $q-j$ steps before the first matrix element, a_{11}, and succeeding elements of b_j enter every two steps. Note that k is not restricted with respect to q, so $q-j$ may be negative. For any number of subdiagonals, q, y_{1j} always leaves the network j steps after r_{11}. Assuming b_{11} enters at $t=1$, r_{11} is available at $t=2q+(q-1)=3q-1$; therefore, the entire computation takes time $2n+k+3(q-1)=T(n,q)+k+q-1$. The network consists of $q(w+k)$ processors.

The second step, solving $Rx=y$, can be accomplished by a linearly connected mesh from [11], [12]. The input to the latter mesh, however, has to be a lower triangular matrix. Thus, the systolic arrays of steps 1 and 2 cannot be directly linked. Data must be stored in memory or passed through delay elements in order to obtain proper dataflow for the second network.

A partial trace of the dataflow is provided in Fig. 4 for the case $k = q = 2$, $p = 3$. Default values for data are omitted, matrix elements are identified by their indices: the ones entering the network stand for a_{ij}, the ones leaving for r_{ij}.

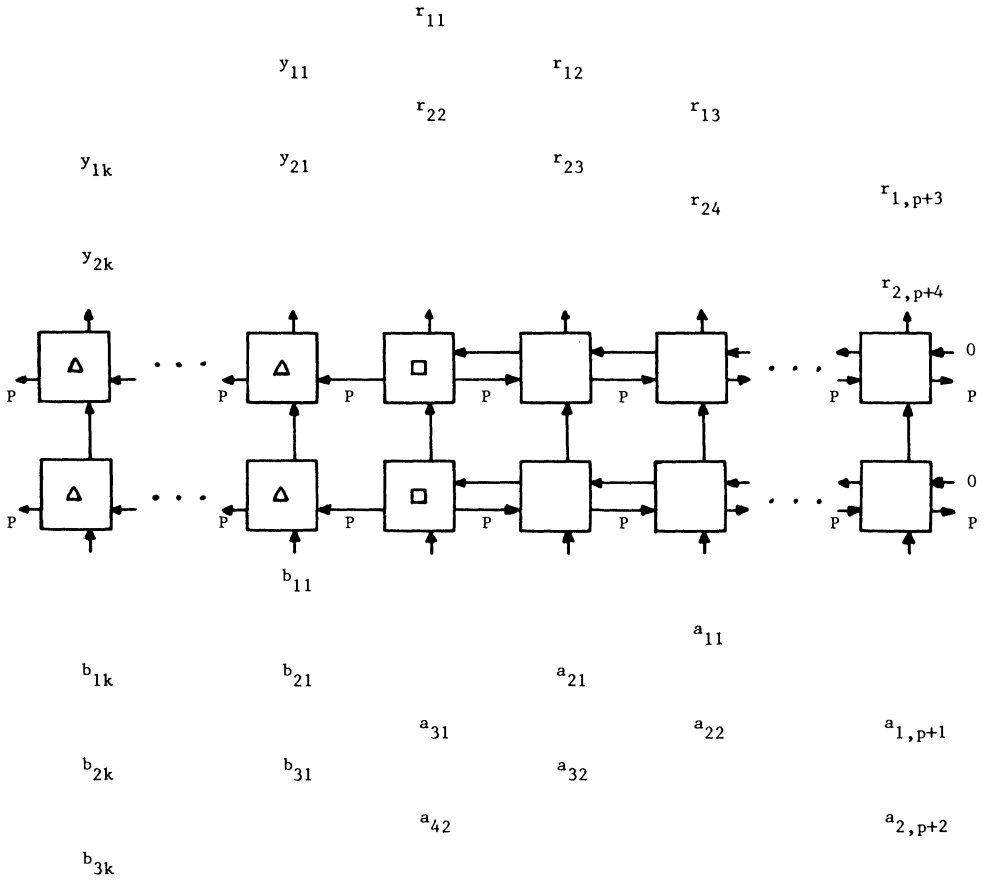


FIG. 3. *Computation of* $R = Q^T A$, $y_j = Q^T b_j$, *with* I/O *format*, $q = 2$.

Treatment of rectangular band matrices. Consider the minimization $\|Ax - b\|_2$ where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and A is assumed to be of full rank. Two cases can be distinguished.

Case 1. $m \geqq n$. Partition $A = (A_1^T 0)^T$, $b = (b_1^T b_2^T)^T$, $A_1 \in \mathbb{R}^{m_1 \times n}$, $b_1 \in \mathbb{R}^{m_1}$, for $m_1 = \min\{m, n+q\}$, so that $\|Ax - b\|_2^2 = \|A_1 x - b_1\|_2^2 + \|b_2\|_2^2$.

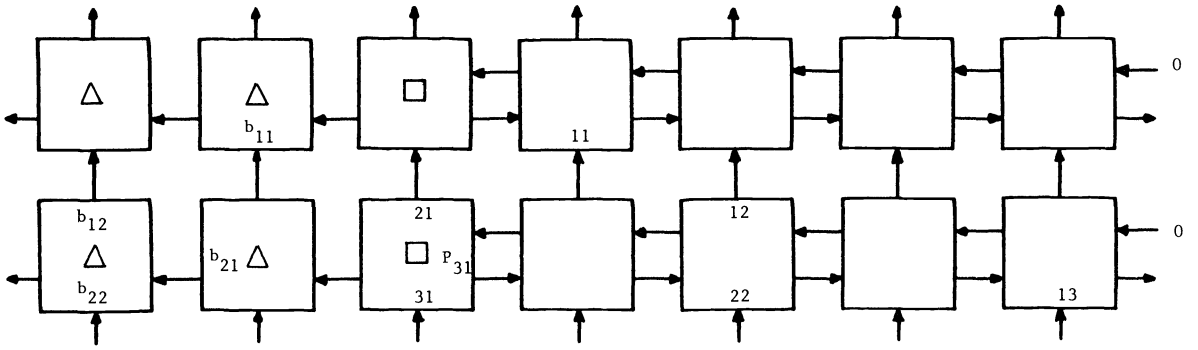
Thus, only A_1 and b_1 need to be entered into the QR network. Let the output be R_1 and y_1 with $R_1 = (R_{11}^T 0)^T$ and $y_1 = (y_{11}^T y_{12}^T)^T$, $R_{11} \in \mathbb{R}^{n \times n}$ and $y_{11} \in \mathbb{R}^n$. Then,

$$\|A_1 x - b\|_2^2 = \|R_1 x - y_1\|_2^2 = \|R_{11} x - y_{11}\|_2^2 + \|y_{12}\|_2^2.$$

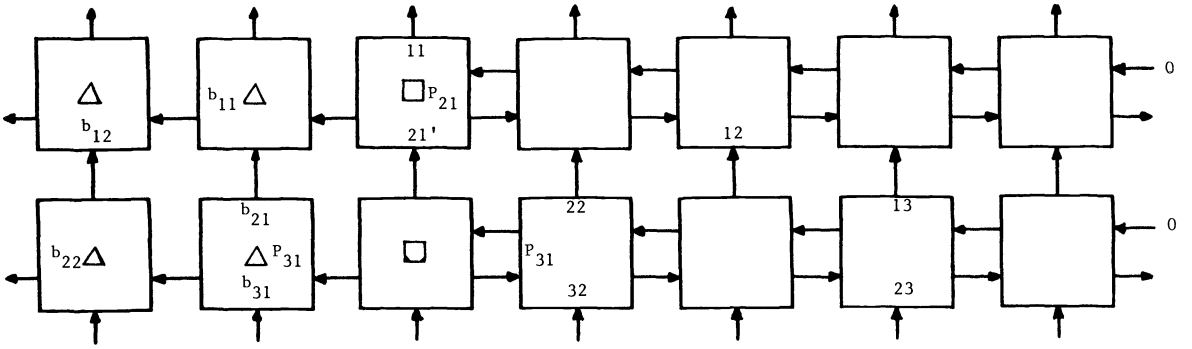
Solve $R_{11} x = y_{11}$ to find the minimizing x. The square of the residual is $\|b_2\|_2^2 + \|y_{12}\|_2^2$. The computation of R_1 and y_1 takes $T(m_1, q)$, while x can be determined in $O(n)$ steps.

Case 2. $m < n$. Again, partition $A = (A_1 0)$, $x = (x_1^T x_2^T)^T$, $A_1 \in \mathbb{R}^{m \times n_1}$, $x_1 \in \mathbb{R}^{n_1}$ for $n_1 = \min\{m+p, n\}$. Thus, $\|Ax - b\|_2 = \|A_1 x_1 - b\|_2$. Hence, only A_1 and b have to be input into QR network. If the output is R_1 and y with $R_1 = (R_{11} R_{12})$, $x_1 = (x_{11}^T x_{12}^T)^T$, $R_{11} \in \mathbb{R}^{m \times m}$ and $x_{11} \in \mathbb{R}^m$, then

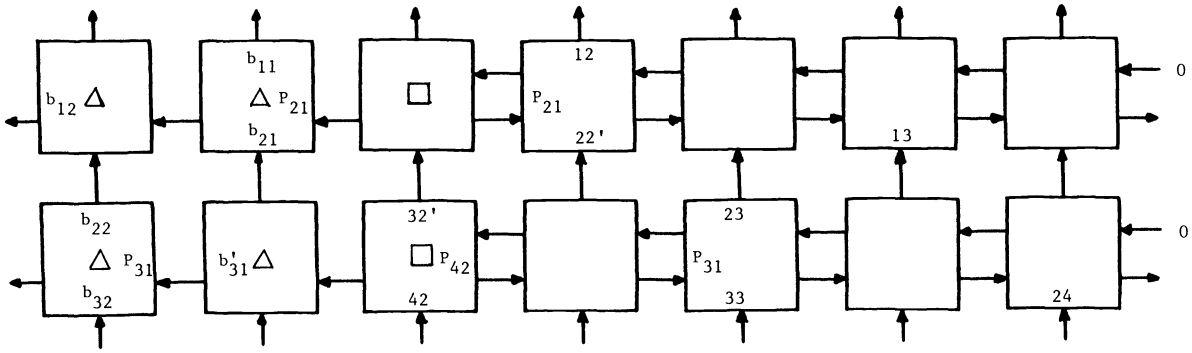
$$\|A_1 x_1 - b\|_2 = \|R_1 x_1 - b\|_2 = \|(R_{11} x_{11} - y) + R_{12} x_{12}\|_2.$$



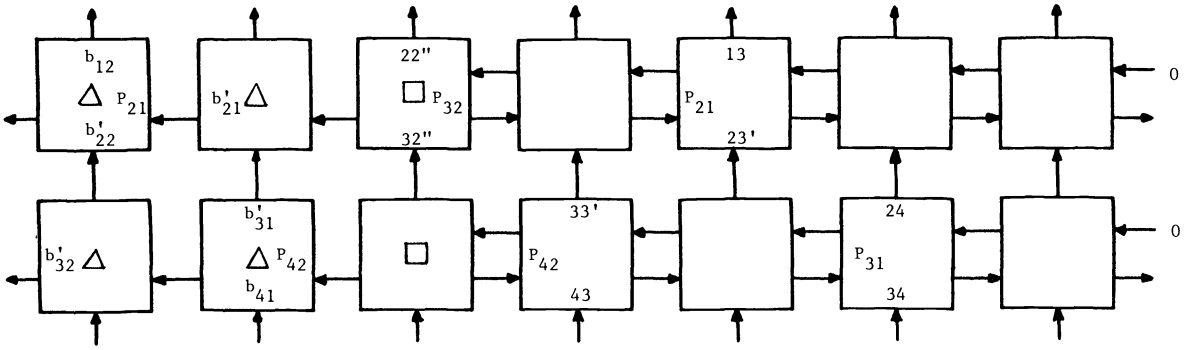
(a) $t = 3$.



(b) $t = 4$.



(c) $t = 5$.



(d) $t = 6$.

FIG. 4. *Partial time trace of Fig.* 3, $k = 2$, $q = 2$, $p = 3$.

Set $x_{12} = 0$ and $x_2 = 0$ and solve $R_{11}x_{11} = y$ to obtain the minimizer with smallest norm. Generation of R_1 and y can be accomplished in $T(n_1, q)$ and x_{11} in $O(m)$ steps.

In general, the time to solve the linear least squares problem for banded matrices comes to approximately T(smaller dimension of A, q). The treatment of $b \in \mathbb{R}^{m \times k}$, $k > 1$, follows the same lines.

VLSI implementation. The systolic array concept lends itself well to VLSI implementation of special purpose devices [11], [12]. The orthogonally connected grid of Fig. 3 may be built up from three basic rectangular cells:

1. to implement (1);
2. to implement (2), on the right of Cell 1 in Fig. 3;
3. to implement (3), on the left of Cell 1 in Fig. 3.

Cells 2 and 3 are nearly identical and should require far less area than Cell 1, as their functions are a small subset of those of Cell 1. Cell 1 is replicated q times, Cell 2 $q(p'+q)$ times, and Cell 3 qk times, so the largest cell is needed least often. The systolic array of Gentleman and Kung for dense matrices [6] has similar characteristics.

For connection to other systolic arrays, the bandmatrix input and output regimen established in [11], [12] is followed. Thus the input to one array may come from the output of another, and temporal displacement among elements does not have to be changed. As mentioned, however, the output from the decomposition array must be directed to memory for completion of the solution to a linear system or least squares problem. Memory devices supporting the I/O regimen would, of course, be available in a computer system adopting the systolic array concept.

REFERENCES

[1] H. M. AHMED, J.-M. DELOSME AND M. MORF, *Highly concurrent computing structures for matrix arithmetic and signal processing*, Computer, 15, no. 1 (Jan. 1982), pp. 65–82.

[2] A. BOJANCYZK, R. P. BRENT AND H. T. KUNG, *Numerically stable solution of dense systems of linear equations using mesh-connected processors*, this Journal, 5 (1984), to appear.

[3] D. GANNON, *A Note on Pipelining a Mesh-Connected Multiprocessor for Finite Element Problems by Nested Dissection*, in Proc. International Conference on Parallel Processing, IEEE Computer Society Press, Silver Spring, MD., 1980, pp. 197–204.

[4] W. MORVEN GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329–336.

[5] ———, *Error analysis of QR decompositions by Givens transformations*, Linear Algebra and Appl., 10 (1975), pp. 189–197.

[6] W. M. GENTLEMAN AND H. T. KUNG, *Matrix triangularization by systolic arrays*, in Real Time Signal Processing IV: SPIE Proceedings vol. 298, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 1981, pp. 19–26.

[7] SVEN HAMMARLING, *A note on modifications to the Givens plane rotation*, J. Inst. Math. Appl., 13 (1974), pp. 215–218.

[8] L. JOHNSSON, *A computational array for the QR-method*, in Proc. of a Conference on Advanced Research in VLSI, P. Penfield, ed., Artech House, Inc., Dedham, MA, 1982, pp. 123–129.

[9] H. T. KUNG, *The structure of parallel algorithms*, in Advances in Computers, 19, Academic Press, New York, 1980, pp. 65–112.

[10] ———, *Why systolic architectures*?, Computer, 15, no. 1 (Jan. 1982), pp. 37–46.

[11] H. T. KUNG AND CHARLES E. LEISERSON, *Systolic arrays (for* VLSI), in Sparse Matrix Proceedings, I. Duff and G. Stewart, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1978, pp. 256–282.

[12] C. MEAD AND L. CONWAY, *Introduction to* VLSI *Systems*, Addison-Wesley, Reading, MA, 1980.

[13] A. H. SAMEH AND D. J. KUCK, *Parallel direct linear system solvers—A survey*, in Parallel Computers—Parallel Mathematics, M. Feilmeier, ed., International Association for Mathematics and Computers in Simulation, Amsterdam, 1977.

[14] A. H. SAMEH AND D. J. KUCK, *On stable parallel linear system solvers*, J. Assoc. Comput. Mach., 25 (1978), pp. 81–91.
[15] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
[16] ———, *The economical storage of plane rotations*, Numer. Math., 25 (1976), pp. 137–138.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0011 $01.25/0

CONSTRUCTION OF A CURVILINEAR GRID*

BARBRO KREISS†

Abstract. The construction of overlapping grids is explained and applied to a system of hyperbolic differential equations.

Key words. overlapping grids, curvilinear grid, solution of PDE, interpolation.

1. Introduction. We want to solve a hyperbolic system of partial differential equations

$$\frac{\partial u}{\partial t} = A\frac{\partial u}{\partial x} + B\frac{u}{\partial y} + F \tag{1.1}$$

in a domain Ω_A bounded by a smooth curve $\partial\Omega_A$. At $t = 0$ we have the initial conditions

$$u(x, 0) = f(x),$$

and on $\partial\Omega_A$ boundary conditions

$$Lu = g$$

are given.

The crudest method is to cover Ω_A by a rectilinear grid (Fig. 1.1) and replace the differential equations and boundary conditions by difference equations.

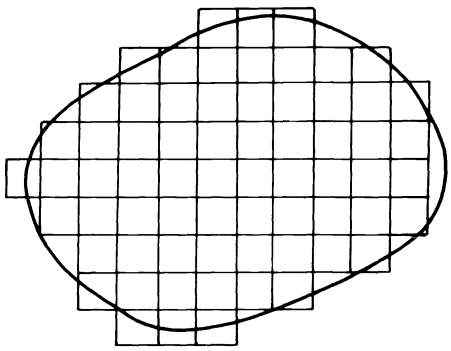


FIG. 1.1

Unfortunately this process is often very inaccurate. We shall instead use two different overlapping grids:

(1) One curvilinear grid, G_C, defined by the transformation

$$T: \begin{cases} x = (r, s), \\ y = y(r, s), \end{cases} \qquad 0 \leqq r, s \leqq 1, \tag{1.2}$$

* Received by the editors November 13, 1981, and in revised form June 25, 1982. This project was sponsored by the Office of Naval Research under contract N0014-80-C-0076, the U.S. Department of Energy under contract EY-76-S-03-070 and the Swedish Natural Science Council (NFR 2711-018) and the Swedish Board for Technical Development (STU 77-3690).

† Department of Computer Sciences, Uppsala University, Uppsala, Sweden. Present address, Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

which follows the boundary. It covers a domain $\Omega_C = \Omega_A - \Omega_B$, where Ω_B is bounded by another smooth curve $\partial\Omega_B$. This grid is constructed with help of splines.

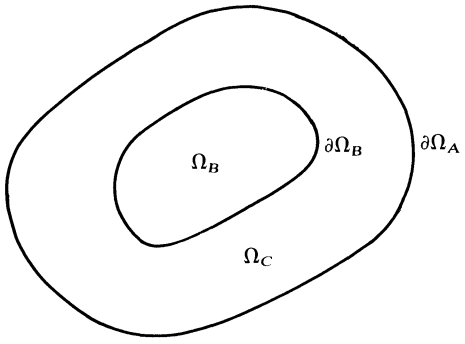


FIG. 1.2

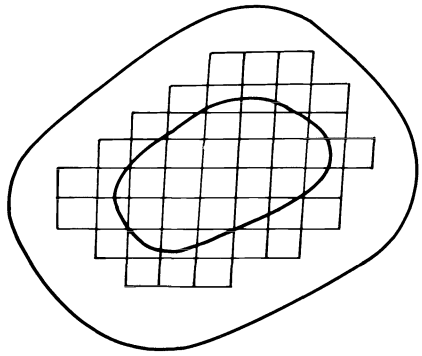


FIG. 1.3

(2) One rectilinear grid, G_B, covering Ω_B and partly overlapping Ω_C.

We replace (1.1) by difference equations. On Ω_B we use a standard leap-frog procedure. In order to obtain the difference equations on the curvilinear grid, we introduce r, s as new independent variables and obtain

$$\frac{\partial u}{\partial t} = \tilde{A}\frac{\partial u}{\partial s} + \tilde{B}\frac{\partial u}{\partial r} + F. \tag{1.3}$$

We replace (1.3) and the boundary conditions by difference equations and again use a standard leap-frog procedure.

The solutions on the two grids are connected by interpolation. The grid construction program generates a mapping function and its derivatives and the weight functions necessary for the interpolation.

Overlapping grids have earlier been used by [4]. However, we believe that our construction is simpler and more flexible. We are already working on extensions of the grid construction that include stretching in the r and s directions and allow other kinds of domains. Our applications include elliptic equations and a combination of hyperbolic and elliptic equations. The use of splines for grid construction has also been used by [5]. There is no overlapping of grids, though, which we think is a very important feature.

2. Grid construction. In practice the boundaries $\partial\Omega_A$ and $\partial\Omega_B$ are only known at J corresponding points

$$P_j, Q_j, j = 1, \cdots, J.$$

Therefore we define $\partial\Omega_A$ and $\partial\Omega_B$ by spline interpolation, as described in [1, p. 42] and [2, p. 50] and obtain a representation $P(s)$, $Q(s)$ of $\partial\Omega_A$, $\partial\Omega_B$ respectively, as functions of a parameter s, $0 \leqq s \leqq 1$. For every fixed s we connect $P(s)$, $Q(s)$ by a smooth curve $D(r, s)$, $0 \leqq r, s \leqq 1$. The simplest curve is a straight line, and this is the only type used so far (Fig. 2.1).

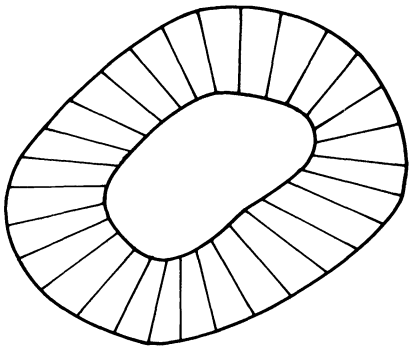


FIG. 2.1

This procedure defines the coordinate transformation (1.2).

We want to cover Ω_C by a curvilinear grid G_C. Let $\Delta s = 1/N$ and $\Delta r = 1/(M-1)$. The gridpoints of G_C are defined by

$$P_{i,\nu}(r, s) = P_{i,\nu}((i-1)\cdot\Delta r, \nu\cdot\Delta s), \qquad i = 1, \cdots, M, \quad \nu = 0, \cdots, N.$$

This curvilinear grid in the x, y-plane corresponds in the r, s-plane to a uniform grid over the rectangle $0 \leqq r, s \leqq 1$ (Fig. 2.2).

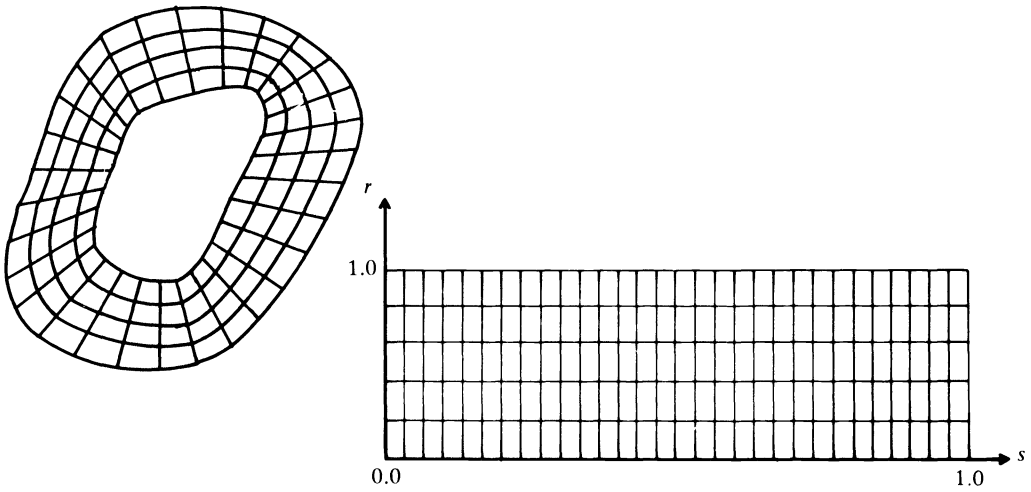


FIG. 2.2

We cover Ω_A by a uniform rectilinear grid. We use a method as given in [3, p. 358] and [4, p. 4.1] to restrict this grid in such a way that it covers Ω_B but is contained in Ω_A. Set $r = r_c$. We then obtain a coordinate curve

$$C(s)\colon x = x(r_c, s),\ y = y(r_c, s).$$

For every point on C we determine the closest point in the rectilinear grid. The points thus obtained are the boundary points R_L of the restricted grid. It is important that R_L forms a closed polygon. If two consecutive points on C give boundary points whose indices indicate a gap >1, then we define the intermediary boundary points as those which approximate the chord between the two points on C (Fig. 2.3).

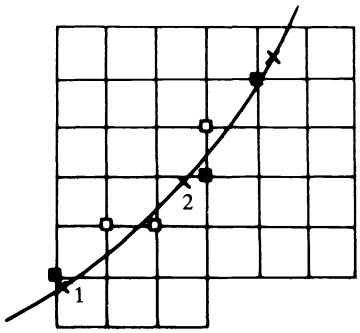


FIG. 2.3. $\times$ *Points on* C; ■ *boundary points from points* 1, 2, 3; □ *additional boundary points.*

This process needs a starting point that is known to be in the interior of Ω_B. Finally R_L is cleaned of corner points, which are not needed during the computation.

An integer image array is used to flag the points of the rectilinear grid. For the boundary points R_L the flag is 2. In these points we will interpolate. For interior points the flag is 1. There we will apply the leap-frog scheme. For outside points the flag is 0. The grid G_B consists of gridpoints, where the flag $\neq 0$.

3. Numerical solution of hyperbolic differential equations. Consider a symmetric hyperbolic system

$$\frac{\partial u}{\partial t} = A\frac{\partial u}{\partial x} + B\frac{\partial u}{\partial y} \tag{3.1}$$

for $(x, y) \in \Omega_A$, $t \geqq 0$. Here $u = (u^{(1)}, \cdots, u^{(n)})^T$ is a vector function and $A = A^*, B = B^*$ are $n \times n$ symmetric matrices which depend smoothly on x, y, t. At $t = 0$ initial conditions

$$u(x, 0) = f(x), \tag{3.2}$$

and on $\partial\Omega_A$ a number of linear relations between the components of u are given as boundary conditions

$$Lu = g, \qquad (x, y) \in \partial\Omega_A. \tag{3.3}$$

As we have described earlier, we cover Ω_A by two overlapping grids G_B and G_C. G_B is rectilinear in the x, y-plane and G_C in the r, s-plane, defined by the transformation (1.2). At all interior points of G_B we approximate the differential equation by the leap-frog scheme, i.e.,

$$u(x, y, t+\Delta t) = u(x, y, t-\Delta t) + 2\Delta t(A(x, y, t)D_{0x} + B(x, y, t)D_{0y})u(x, y, t).$$

Here

$$D_{ox}u(x, y, t) = (u(x+\Delta x, y, t) - u(x-\Delta x, y, t))/2\Delta x,$$

$$D_{0y}u(x, y, t) = (u(x, y+\Delta y, t) - u(x, y-\Delta y, t))/2\Delta y$$

denote the centered difference operators.

In order to obtain the difference approximation in G_C we use the transformation (1.2). We have

$$\frac{\partial u}{\partial x}=\left(\frac{\partial u}{\partial r}\cdot\frac{\partial y}{\partial s}-\frac{\partial u}{\partial s}\cdot\frac{\partial y}{\partial r}\right)\Big/\Delta, \qquad \Delta=\frac{\partial x}{\partial r}\cdot\frac{\partial y}{\partial s}-\frac{\partial x}{\partial s}\cdot\frac{\partial y}{\partial r},$$

$$\frac{\partial u}{\partial y}=\left(\frac{\partial u}{\partial s}\cdot\frac{\partial x}{\partial r}-\frac{\partial u}{\partial r}\cdot\frac{\partial x}{\partial s}\right)\Big/\Delta.$$

Therefore (3.1) becomes

$$\frac{\partial u}{\partial t}=\tilde{A}\frac{\partial u}{\partial r}+\tilde{B}\frac{\partial u}{\partial s}, \tag{3.4}$$

where

$$\tilde{A}=\frac{1}{\Delta}\left(A\frac{\partial y}{\partial s}-B\frac{\partial x}{\partial s}\right), \qquad \tilde{B}=\frac{1}{\Delta}\left(-A\frac{\partial y}{\partial r}+B\frac{\partial x}{\partial r}\right).$$

Here $u(r, s)$ is periodic in s. $r=0$ corresponds to $\partial\Omega_A$, and $r=1$ corresponds to $\partial\Omega_B$.

We approximate (3.4) for $0<r=j\cdot\Delta r<1$ and $s=0,\ \Delta s,\ 2\Delta s,\cdots,\ (N-1)\cdot\Delta s$ again by the leap-frog scheme

$$u(r, s, t+\Delta t)=u(r, s, t-\Delta t)+2\Delta t(\tilde{A}(r, s, t)D_{0s}+\tilde{B}(r, s, t)D_{0r})u(r, s, t). \tag{3.5}$$

For $r=0$ we use a one-sided formula. If

$$\tilde{A}(0, s)=\begin{pmatrix}-\Lambda_1 & 0\\ 0 & \Lambda_2\end{pmatrix}, \qquad \Lambda_j>0 \text{ diagonal}, \quad j=1, 2,$$

is diagonal, then the boundary formula becomes particularly simple. The positive eigenvalues correspond to the outgoing characteristics. Therefore we extrapolate these variables

$$\begin{aligned}u^{\mathrm{II}}(0, s, t+\Delta t)=u^{\mathrm{II}}(0, s, t)&+\frac{\sigma(\Delta s)^2}{2}D_{+s}D_{-s}u^{\mathrm{II}}(0, s, t)\\ &+\Delta t\,(\Lambda_2 D_{+r}u^{\mathrm{II}}(0, s, t)+(\tilde{B}D_{0s}u(0, s, t))^{\mathrm{II}}).\end{aligned} \tag{3.6}$$

Here σ is a parameter and

$$D_{+r}u^{\mathrm{II}}(0, s, t)=(u^{\mathrm{II}}(\Delta r, s, t)-u^{\mathrm{II}}(0, s, t))/\Delta r,$$

$$D_{+s}D_{-s}u^{\mathrm{II}}(0, s, t)=(u^{\mathrm{II}}(0, s+\Delta s, t)-2u^{\mathrm{II}}(0, s, t)+u^{\mathrm{II}}(0, s-\Delta s, t))/(\Delta s)^2.$$

For the other variables we use the boundary conditions.

If $\tilde{A}$ is not diagonal, then we proceed in the following way: At every boundary point $r=0$, $s=s_0$ we construct a unitary transformation U such that

$$U^*\tilde{A}U=\Lambda.$$

Then we introduce a new variable $\tilde{u}$ by $u=U\tilde{u}$ and transform (3.4). For the transformed equations we can use the approximation(3.6). Changing back to the original variables we obtain the boundary approximation.

Assume now that u is known at times t and $t-\Delta t$. Then we can use the above approximations to determine u at time $t+\Delta t$ in all gridpoints along $\partial\Omega_A$ and in all interior gridpoints of both G_C and G_B. The values of u at the gridpoints on the boundaries $\partial\Omega_B$ and R_L are obtained by interpolation, which will be explained in the next section.

4. Interpolation. Let $u(x, y)$ be a smooth function. Let P_i, $i = 1, \cdots, 9$ be nine points uniformly spaced in a rectilinear grid, and let $u(P_i)$ denote the function values in these points.

We want to determine an approximate value $u(Q)$ at a point Q inside the rectangle formed by P_1, P_3, P_7, P_9. See Fig. 4.1.

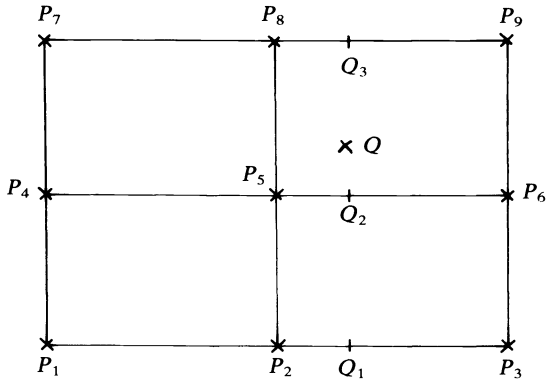


FIG. 4.1

Consider the 3-point formula

$$u(Q_I) = a_1(\alpha)u(P_{i-1}) + a_2(\alpha)u(P_i) + a_3(\alpha)u(P_{i+1}), \qquad 0 \leqq |\alpha| \leqq 1, \quad P_{i-1} \leqq Q_I \leqq P_{i+1} \tag{4.1}$$

with

$$a_1(\alpha) = \frac{\alpha(\alpha-1)}{2}, \quad a_2(\alpha) = \frac{(\alpha+1)(\alpha-1)}{-1}, \quad a_3(\alpha) = \frac{(\alpha+1)\alpha}{2}.$$

We use (4.1) with $\alpha = (x_Q - x_{P_5})/h_x$. First let $i = 2$ and we obtain

$$u(Q_1) = a_1(\alpha)u(P_1) + a_2(\alpha)u(P_2) + a_3(\alpha)u(P_3).$$

We obtain $u(Q_2)$ and $u(Q_3)$ when we let $i = 5, 8$, respectively. We replace α in (4.1) by $\beta = (y_Q - y_{P_5})/h_y$, P by Q and let $i = 2$ and obtain

$$u(Q) = a_1(\beta)u(Q_1) + a_2(\beta)u(Q_2) + a_3(\beta)u(Q_3)$$

or

$$u(Q) = a_1(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_i) + a_2(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_{i+3}) + a_3(\beta) \sum_{i=1}^{3} a_i(\alpha)u(P_{i+6}). \tag{4.2}$$

For every gridpoint Q on $\partial\Omega_B$ we find the gridpoint in G_B that is closest to Q. We denote this point P_5 and apply (4.2) using the function values of the nine gridpoints in G_B with P_5 in the center to obtain an approximate function value in Q.

The nine weight coefficients in (4.2) and the location of P_5 are computed once and stored to be available at computation time.

In our computation the grid G_C had to be considerably denser than G_B. Therefore we have found it to be satisfactory to use a four point formula when we interpolate among gridpoints in G_C to find an approximate value at the boundary points R_L of G_B.

Let $P \in R_L$. Denote by P_1 the gridpoint in G_C that has the shortest distance to P. We call three of $P_1 : s$ neighbor points in G_C P_2, P_3, P_4. See Fig. 4.2. Denote the vector from P_1 to P_2 by a, the vector from P_1 to P_4 by c and the vector from P_4 to P_3 by b.

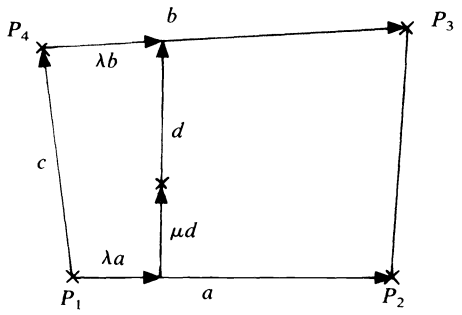


FIG. 4.2

The vector d through the point P is defined by

$$d = \lambda b + c - \lambda a, \qquad 0 \leqq \lambda \leqq 1. \tag{4.3}$$

We also have

$$P - P_1 = \lambda a + \mu d, \qquad 0 \leqq \mu \leqq 1. \tag{4.4}$$

Equations (4.3) and (4.4) give us

$$P - P_1 = \lambda a + \mu c + \lambda\mu (b - a). \tag{4.5}$$

We use an iterative process to solve (4.5) to obtain λ, μ. We have assumed that $\lambda, \mu \geqq 0$, i.e., P lies in "quadrant 1" of P_1. If the solution of (4.5) turns out negative values for λ or μ or both, then we try another "quadrant" of P_1, i.e., P_2, P_3, P_4 could be another set of points around P_1. Figure 4.3 shows the different "quadrants" and the arrows indicate the ascending order of the points P_2, P_3, P_4.

When we have reached satisfactory values for λ and μ, we proceed using the following standard interpolation formula, in order to obtain an approximate function

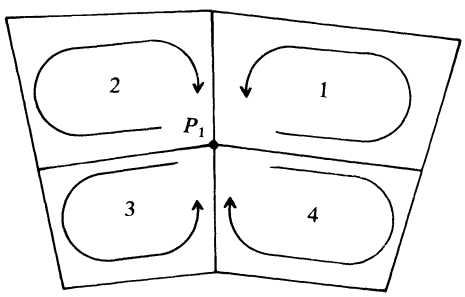


FIG. 4.3

value $u(P)$ from the function values $u(P_i)$, $i = 1, \cdots, 4$,

$$u(P) = (1-\lambda)(1-\mu)u(P_1) + \lambda(1-\mu)u(P2) + \lambda\mu \cdot u(P_3) + (1-\lambda)\mu \cdot u(P_4). \tag{4.6}$$

The four weight coefficients in (4.6) are computed once and stored to be available at computation time. Also available are the locations of P in G_B and of P_1 in G_C as well as the correct "quadrant" number. This is done for every point $P \in R_L$.

5. An example. As a test case we have solved (1.1) with

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad F = (\cos(x+2t), -3\sin(x+2t))^T, \tag{5.1}$$
$$u(x,0) = (\sin x, \cos x)^T, \qquad u(x, \Delta t) = (\sin(x+2\Delta t), \cos(x+2\Delta t))^T.$$

For the boundary $r = 0$ we have

$$a_1 u^{(1)} + a_2 u^{(2)} = g.$$

Remark. The coefficients a_1 and a_2 are chosen in such a way that we give the variable corresponding to the ingoing characteristic. The solution to this problem is

$$v = (\sin(x+2t), \cos(x+2t))^T.$$

Therefore we have been able to compute the error function

$$e = v - u.$$

Table 5.1 shows the maximum error, $e_{\max}$, and the global error for different grid densities. We used $\sigma = 0$ and $\Delta t = 0.001$. We constructed the grids from two sets of 7 points. They are listed in Table 5.2. In the iterative processes of the grid construction we used a tolerance $\varepsilon = 10^{-6}$. The grid construction program was run on a VAX 11/780 in time-sharing mode, the solution of (5.1) was done on an IBM 370/3032 at the California Institute of Technology, Pasadena.

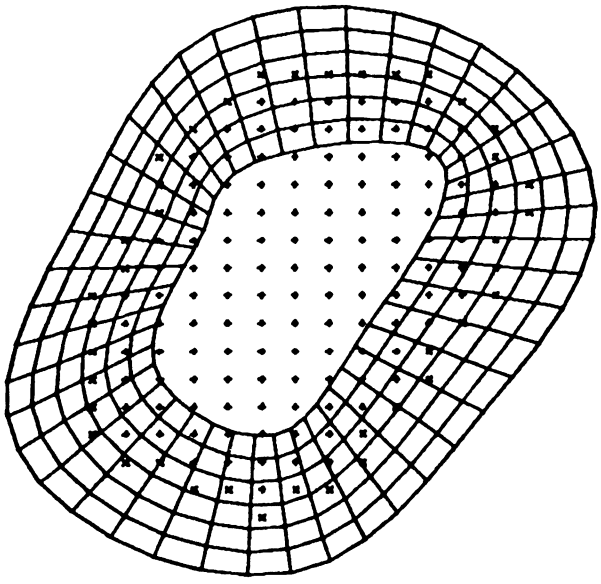


FIG. 5.1

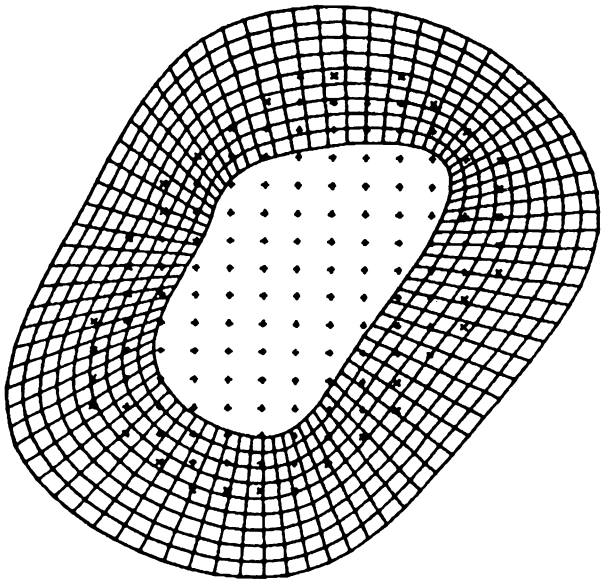


FIG. 5.2

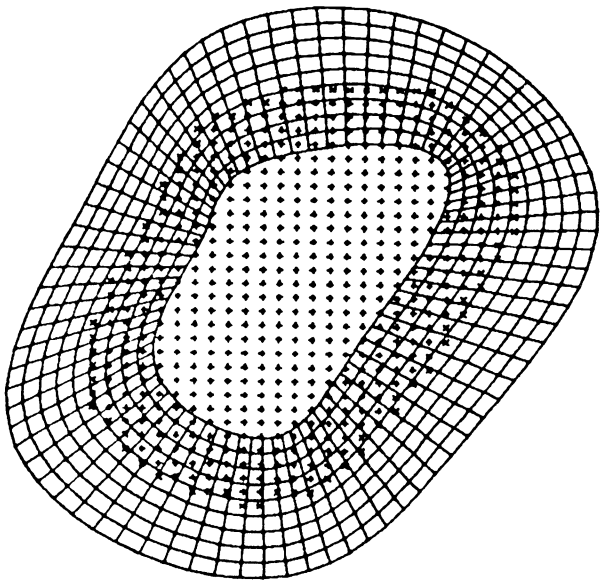


FIG. 5.3

TABLE 5.1

	time	G_C		G_B		G_C		G_B		global e
		N	M	h_x	h_y	$e_{\max}u^{(1)}$	$e_{\max}u^{(2)}$	$e_{\max}u^{(1)}$	$e_{\max}u^{(2)}$	
I	0.2	43	7	0.05	0.05	0.00289	0.00319	0.00228	0.00231	0.00118
	0.4					0.00280	0.00263	0.00269	0.00304	0.00147
	0.6					0.00224	0.00276	0.00497	0.00437	0.00130
	0.8					0.00195	0.00316	0.00689	0.00461	0.00114
	1.0					0.00252	0.00316	0.00712	0.00608	0.00159
II	0.2	78	10	0.05	0.05	0.00118	0.00158	0.00080	0.00127	0.00124
	0.4					0.00130	0.00180	0.00109	0.00183	0.00156
	0.6					0.00151	0.00144	0.00164	0.00184	0.00116
	0.8					0.00136	0.00114	0.00169	0.00183	0.00101
	1.0					0.00150	0.00181	0.00198	0.00336	0.00138
III	0.2	78	10	0.025	0.025	0.00100	0.00151	0.00082	0.00130	0.00079
	0.4					0.00130	0.00172	0.00125	0.00185	0.00098
	0.6					0.00154	0.00154	0.00160	0.00214	0.00067
	0.8					0.00139	0.00116	0.00183	0.00146	0.00064
	1.0					0.00153	0.00164	0.00208	0.00181	0.00076

The grids I, II, III (see Table 5.1) are plotted in Fig. 5.1, 5.2, 5.3 respectively.

TABLE 5.2.

$\partial\Omega_A$		$\partial\Omega_B$	
x	y	x	y
0.49195	0.00291	0.46714	0.24867
0.77552	0.28246	0.58529	0.40381
0.94802	0.66032	0.71172	0.63728
0.68927	0.99824	0.65264	0.77706
0.36907	0.96906	0.43406	0.73712
0.19184	0.67414	0.36198	0.61270
0.09495	0.22102	0.29229	0.37616

REFERENCES

[1] GÖRAN STARIUS: *Composite mesh difference methods for elliptic boundary value problems*, Numer. Math., 28 (1977), pp. 243–258.

[2] J. H. AHLBERG, E. N. NILSSON AND J. L. WALSCH: *The Theory of Splines and Their Applications*, Academic Press, London–New York, 1967.

[3] G. E. FORSYTHE AND W. R. WASOW: *Finite Difference Methods for Partial Differential Equations*, John Wiley, New York–London, 1960.

[4] GÖRAN STARIUS: *Numerical treatment of boundary layers for perturbed hyperbolic equations.* Depart. of Computer Science Report No. 69, Uppsala Univ., Sweden, 1978.

[5] PETER R. EISEMAN: *Geometric methods in computational fluid dynamics*, Rep. 80-11 Institute for Computational Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, April 18, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0012 $01.25/0

ESTIMATING THE DISTRIBUTION OF SPHERICAL PARTICLES FROM PLANE SECTIONS: AN OPTIMAL ALGORITHM FOR SOLUTION OF THE ABEL INTEGRAL EQUATION*

WILLIAM M. VISSCHER† AND AARON S. GOLDMAN†

Abstract. The persistently troublesome problem of solving the Abel integral equation with noisy data is reconsidered. A criterion is formulated for determination of a most probable solution which satisfies the reasonable requirement that frequency estimates be nonnegative. Our prescription, called the constrained minimum variance estimator (COMIVE) results in an optimum (in a sense described in the text) radius distribution for the spherical particles. Numerical examples are given which use input generated by Monte Carlo from various distributions and which use real experimental data. Results demonstrate considerable improvement over those obtained by other methods.

Key words. Abel integral equation, least squares, minimum variance, sampling distribution, stereology, parameter estimation

1. Introduction. Stereology is the art of obtaining information about three-dimensional objects from a study of their projections or their sections in the plane. [See Elias (1977)]. A prototypical stereological problem is that confronting an analyst who wishes to find the radius distribution of cells from observations of sections of tissues [Wicksell (1925)]. Similar problems arise in geology, metallurgy, materials science, ceramics, physiology, anatomy and other disciplines. Despite the fact that these studies have become highly automated, with microelectronics and many sophisticated instruments in general use, it has recently been remarked by Elias (1980) that "astonishingly enough, the determination of size distribution and numerical density are still the most difficult problems in the field." Their difficulty has caused Watson (1971) to advise, "this experimental method should be avoided when there is a practical alternative." Symptomatic of the intractability is the fact that all statistical methods given in the literature for estimating 3D distributions result in some negative estimates of frequencies or variances. Desperate measures are sometimes prescribed to circumvent these absurdities, such as visual smoothing [Elias (1980)], interpolation [Tallis (1970)] or summarily replacing negative frequencies with zero [Saltikov (1967)]. It is the aim of this paper to provide a method for solving this stereology problem which replaces these arbitrary prescriptions for treating the symptoms with an algorithm that actually cures the disease.

The original formulation of the stereology problem was given in the classic paper by Wicksell (1925). Work since that time has been summarized by Anderssen and Jakeman (1975) and by Nicholson and Merckx (1969). Some other references which are useful are Saltikov (1967), Minerbo and Levy (1969), DeHoff (1965) and Wahba (1979).

One of the simplest problems of stereology has been the subject of most quantitative investigations. It may be stated as follows: a set of spheres with a distribution of radii is dispersed randomly in a volume. An indefinite number of randomly placed planes pass through this configuration. Spheres which are sectioned appear as circles on these planes. The problem is, given a set of observations of the circle radii, what can be said about the number distribution of spheres of various radii in the volume?

* Received by the editors December 3, 1980, and in revised form November 2, 1981. This research has been supported by the U.S. Department of Energy.

† Los Alamos National Laboratory, University of California, Los Alamos, New Mexico 87545.

A problem which can be so simply stated ought to have a simple solution, but, until now, all attempts on this one have not excluded impossible answers, namely, negative density estimates.

2. Abel integral equation. If a plane passes a distance h from the center of a sphere of radius R, the intersection is a circle with radius $r=\sqrt{R^2-h^2}$. If there are many planes randomly uniformly distributed in h, then the distribution of circle radii is proportional to $|dh/dr|=r/h=r/\sqrt{R^2-r^2}$, and it is easy to see that, if $f(R)$ is the volume density of spheres of radius R and $\phi(r)$ is the surface density of circles of radius r on the sectioning planes, that

$$\phi(r)=2\int_r^\infty \left|\frac{dh(R,r)}{dr}\right| f(R)\,dR = 2r\int_r^\infty \frac{dR}{\sqrt{R^2-r^2}} f(R). \tag{1}$$

This is the Abel integral equation. [A more general form is given by Goldman and Visscher (1978).] As long as there are many sectioning planes, randomly positioned, it is valid regardless of the degree of order in the array of spheres. Our task is, given $\phi(r)$ in the form of discrete observations, to find $f(R)$. For a wide class of ϕ's this can be done exactly. But there are pitfalls, which we will now explain.

The Abel integral equation can easily be formally solved for $f(R)$ if $d\phi(r)/dr$ exists. The solution is

$$f(R)=\frac{\phi(R)}{\pi R}+\frac{R}{\pi}\int_R^\infty \frac{\phi(R)-\phi(r)}{(R^2-r^2)^{3/2}}\,dr, \tag{2}$$

as can be verified by direct substitution. But (2) is of limited usefulness in practical stereology problems for a couple of reasons. (i) Experimental data for $\phi(r)$ seldom, if ever, satisfy the differentiability requirement. Points at which $\phi(r)$ is discontinuous can give infinite contributions to (2), so $\phi(r)$ must somehow be smoothed. Unfortunately, the answers one gets for $f(R)$ depend on exactly how $\phi(r)$ is smoothed, and there are many reasonable ways to do it. (ii) $f(R)$ as given by (2) is not manifestly positive. For example, if for some R, $\phi(R)=0$, then, unless $\phi(r)=0$ for all $r>R$, then (2) says $f(R)<0$. This is just a reflection of the obvious fact that if one sees circular sections of radius r, then one should also see sections of radius r' for all $r'<r$.

In practice, experimental data for $\phi(r)$ consist of finite numbers of observations, usually in the form of histograms, so (i) says that $f(R)$ as given by (2) is not well defined and (ii) says that, for some values of R, $f(R)<0$, and the noisier the data $\phi(r)$, the more values of R there will be at which this physically absurd result obtains.

These deficiencies render (2) useless for direct stereological analysis. Although it is, under certain conditions, equivalent to (1) mathematically, it is not equivalent numerically when it is discretized, as it always is in practice. We will work henceforth directly with (1).

Observations are usually tabulated as frequencies per class interval. In terms of $\phi(r)$, these are given by

$$\phi_i=\int_{r_{i-1}}^{r_i}\phi(r)\,dr, \qquad 1\leqq i\leqq k, \tag{3}$$

which are the observed densities per unit area of circles with radius $r_{i-1}<r<r_i$. This amounts to a discretization of r into k histogram bins $(0,r_1),(r_1,r_2),\cdots,(r_{k-1},r_k)$.

Then, if the integral in (1) is approximated by some quadrature scheme, (1) becomes

$$\phi_i = \sum_{j=1}^{K} g_{ij} f_j, \qquad i = 1, \cdots, k, \tag{4}$$

where K is the number of intervals in the quadrature.

The form of the $k \times K$ matrix g is determined by the particular scheme chosen. If it is defined by the δ-function sum

$$f(R) = \sum_{j=1}^{K} f_j \delta(R - R_j) \tag{5}$$

[see Wicksell (1925) or Saltikov (1967)] or it is a histogram

$$f(R) = \sum_{j=1}^{K} f_j \theta(r_{j-1} | r | r_j), \tag{6}$$

(where

$$\theta(a|x|b) = \begin{cases} (b-a)^{-1}, & a < x \leqq b, \\ 0 & \text{otherwise}), \end{cases}$$

then the corresponding forms for g_{ij} are given in Appendix A. According to (5) there are assumed to be K different sphere species with radii $R_1 \cdots R_K$ and densities $f_1 \cdots f_K$. Saltikov makes a special choice of $\{R_j\}$ which are equally spaced on a logarithmic scale. According to (6) one has uniform distribution of sphere sizes in each of K histogram bins (class intervals) $(0, R_1), (R_1, R_2) \cdots (R_{K-1}, R_K)$ into which are placed $f_1, f_2, \cdots f_K$ spheres per unit volume.

Many other choices for the form of $f(R)$ besides (5) and (6) are possible. Tallis (1979) uses a discretization (ramp function) which involves piecewise straight lines. His example using Swiss cheese data is discussed in Appendix B.

The matrix g in (4) is upper-triangular, i.e.,

$$g_{ij} = 0 \quad \text{if } r_i > R_j, \tag{7}$$

for obvious reasons. So if $K = k$ it is very simple to solve (4) by successive elimination, starting with the largest spheres. It is often convenient to choose $k = K$; it is not, however, necessary. Equation (4) has a unique solution in general only in that case.

3. Constrained minimum variance estimator. Equation (4) does not determine $\{f_j\}$ unless $k = K$. If $k > K$, $\{f_j\}$ is overdetermined, and if $k < K$, it is underdetermined. Following Nicholson and Merckx (1969) we will adopt a minimum principle as a determinator of $\{f_j\}$ which gives the same answer as (4) when $k = K$, but also gives an answer when $k > K$, the more usual situation in practice.

Using a vector notation $\vec{f} = (f_1, \cdots, f_K)$, $\vec{\phi} = (\phi_1, \cdots, \phi_k)$, we define the positive bilinear form

$$\chi^2(\vec{f}) = (\vec{\phi} - g\vec{f})^T \Sigma^{-1} (\vec{\phi} - g\vec{f}), \tag{8}$$

where Σ is a $k \times k$ positive definite symmetric matrix. At the minimum of (8), clearly,

$$\frac{\partial \chi^2(\vec{f})}{\partial f_j} = 0, \qquad j = 1, 2, \cdots, K. \tag{9}$$

If $k = K$, (9) is identical to (4). If $k > K$, however, (9) has a solution but (4) does not. If $k < K$, (4) generally has multiple solutions and so does (9); this is a situation to be

avoided. One would be foolish, indeed, to choose the number of unknowns f larger than the number of observations ϕ!

We will now cease to regard (4) as the fundamental equation of our applied stereology problem. We replace it with the extremum principle, equivalent to (9),

$$\chi^2(\vec{f}) = \text{minimum with respect to variation of } \vec{f}. \tag{10}$$

But the solution $\vec{f}$ of (10) still suffers from the shortcoming (ii) of the preceding section; namely, some of its components may be negative. The obvious remedy to this situation is to seek a solution to (10) which also satisfies the credibility condition

$$f_i \geqq 0, \qquad i = 1, \cdots, K. \tag{11}$$

The only a priori universally applicable constraints on $\vec{f}$ are (11) and the normalization condition $\Sigma f_j^2 < \infty$ (which is always satisfied for the g_{ij}'s we use). We will call the stereology algorithm (10), (11) the constrained minimum variance estimator (COMIVE).

In $(K+1)$-space (K components of $\vec{f}$, plus χ^2) (8) is a parabolic ellipsoid. That is $\chi^2(\vec{f}) = \text{constant}$ is an ellipsoid in the K-dimensional $\vec{f}$-space, and if one of the components of $\vec{f}$ is fixed, $f_j = C$, then

$$\chi^2 = \chi^2(f_1, f_2, \cdots, C, \cdots, f_K)$$

is a $((K-1)+1)$-dimensional parabolic ellipsoid.

To illustrate the minimum principle, we have plotted χ^2 for $K = 2$ for a couple of contrived situations. In Fig. 1, $\chi^2(f_1, f_2)$ has a minimum for $f_1 > 0$ and $f_2 > 0$. In this case

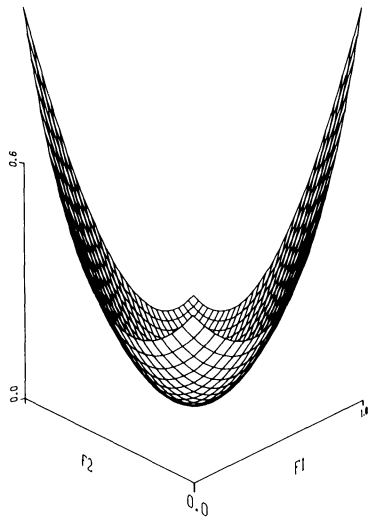


FIG. 1. *Illustration of the* COMIVE *algorithm for two degrees of freedom* ($K = 2$). *In this case the minimum of the parabolic ellipse* $\chi^2(f_1, f_2)$ *lies in the first quadrant, and the constraint* (11) *has no effect.*

the constraint (11) has no effect; the constrained minimum is the same as the unconstrained minimum. In Fig. 2, however, the unconstrained minimum would have $f_1 < 0, f_2 > 0$, which, when constrained, shifts to the origin $f_1 = f_2 = 0$. Figure 3 shows an example for which without constraints both $f_1 < 0, f_2 < 0$; with constraints $f_1 > 0, f_2 = 0$.

For $k = 2$ it is obvious from the geometry of Figs. 1, 2 and 3 that the constrained solution is unique. It is, in fact, true for any value of $K \leqq k$ [see Lawson and Hanson (1974, Exercise 23.55)].

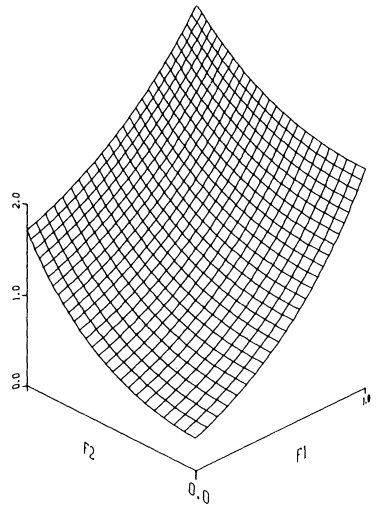


FIG. 2. *The minimum of the parabolic ellipse lies in the second quadrant* ($f_1 < 0, f_2 > 0$); *because of the orientation of the ellipse* $\chi^2 = constant$, *the constrained minimum is at* $f_1 = f_2 = 0$.

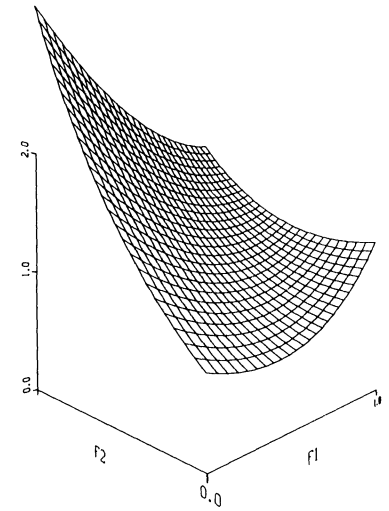


FIG. 3. *The minimum of the parabolic ellipse lies in the third quadrant* ($f_1 < 0, f_2 < 0$). *The constraint shifts it to* $f_1 > 0, f_2 = 0$.

4. Numerical illustration. In order to apply COMIVE one must assign values to the covariance matrix elements Σ_{ij} which appear in (8). Although one can conceive of situations in which it has nonvanishing off-diagonal elements, we will always take it to be diagonal. In the absence of correlated errors, the ϕ_i's in successive trials are integers which are actually Poisson distributed. As far as we can see, however, this does not necessarily mean that a factor ϕ_i should be included in Σ_{ii}. The simplest choice

$$\Sigma_{ii} = 1 \tag{12}$$

works quite well, unless correlated errors are suspected in a given interval i, in which case an appropriately increased value for Σ_{ii} should be assigned. In particular, since many experimental techniques tend to miss small circles, their contribution to (8)

should be unweighted, which obviously is accomplished by increasing the corresponding Σ_{ii}'s.

Once Σ is fixed and measurement gives us a set of values for $\{\phi_i\}$, then it is a straightforward matter to solve (10), (11) for f_i if $k \geqq K$. Our computer code performs the constrained minimization by a simple adaptation of Newton's method. If it weren't for the credibility condition (11) the problem would be trivial; it degenerates to ordinary least squares. With constraints it is still not hard [Lawson and Hanson (1974)].

The number of intervals k and K we choose depends on the number of circles observed. Since in our simulations this is at our disposal, for the sake of definiteness we will take $k = K = 20$ and use the following notation.

$\{\phi_i^I\}$ = input data; numbers of circles observed in ith class interval.

$\{f_i^R\}$ = raw output; defined by $\vec{\phi}^I = g\vec{f}^R$.

$\{f_j^0\}$ = optimum output; this is the solution of (10), (11) with $\vec{\phi} = \vec{\phi}^I$.

$\{\phi_i^B\}$ = back-substituted input; defined by $\vec{\phi}^B = g\vec{f}^0$.

We also take the input intervals to be the same as the output intervals, i.e., $R_i = r_i$. Empty intervals at the largest radii in practice have no effect; i.e., if $\phi_i^I = 0$ for $i > M$, then $f_j^0 = 0$ for $j > M$ also. We never predict spheres larger than the largest observed circles, although this is not guaranteed by the upper-triangularity of g_{ij}.

Although we will give other illustrations, we first concentrate on the simplest test, which is also a stringent one. This is the monosize distribution, for which the spheres have a volume density ρ and radius r_i

$$f_i = \delta_{i,m}\rho \tag{13}$$

in both the Saltikov and histogram discretizations. Any density can be constructed from a superposition of monosize distributions; although the constraints (11) spoil the linearity of the problem and variances are no longer additive, it is reasonable to suppose that if we can handle (13) we can obtain accurate results for any f_i.

Test data $\{\phi_i^I\}$ are generated for the monosize case as follows. The spheres with radius r_m are identically, independently, uniformly distributed in space with average density ρ per unit volume. A plane with unit area intercepts $z = 2r_m\rho$ spheres, on the average. The probability that exactly n spheres will be cut is $P_n = z^n e^{-z}/n!$, and it is a simple matter for the computer, with the aid of a pseudo-random number $0 < Q < 1$, to assign a value of n to each trial (i.e., n satisfies $\sum_{m=0}^{n-1} P_m < Q < \sum_{m=0}^{n} P_m$). Then the radius of the circle formed by the sectioning of each of n spheres is determined by n more random numbers according to $x = r_m\sqrt{1-Q^2}$, and the value of the appropriate ϕ_i^I is incremented (i.e., if $r_{i-1} < x < r_i$ then $\phi_i^I \to \phi_i^I + 1$).

A large ensemble of test data can be generated and processed very quickly. Typically, for an ensemble of 100 samples of approximately 100 circles each and a discretization into 20 bins, computing times are ten seconds on a CDC 6600. Means and variances of various densities for the monosize case are displayed in Figs. 4 and 5 for the Saltikov discretization of g_{ij}. Some of the features that show up deserve comment.

Although the raw output f^R (Fig. 4) often includes negative density estimates, which is reflected by the fact that the standard error bars as well as the means often go negative, the "optimum output" f^0 is nonnegative by the nature of the constrained minimization. As one might have expected, the standard deviation, σ_0, of f^0 is generally appreciably less than σ_R.

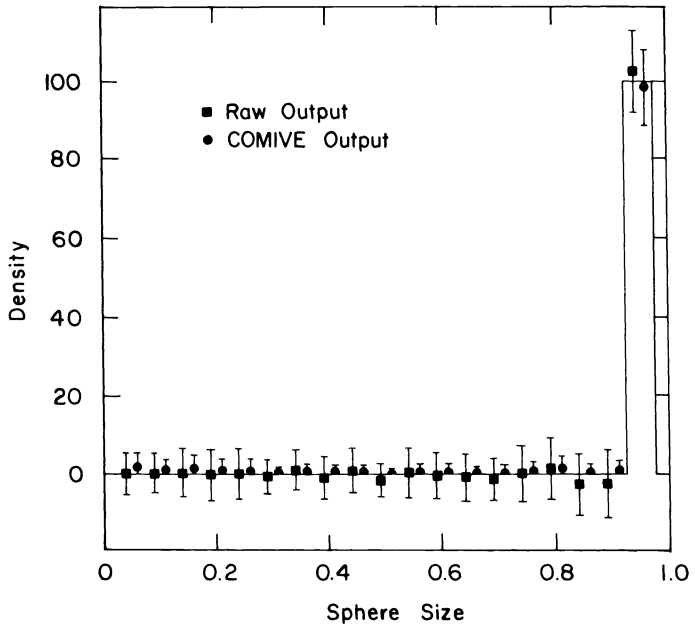


FIG. 4. *Results for Monte Carlo simulation with all spheres having radius* 0.95, *represented by the histogram. The "raw output" is obtained by applying the inverse Abel operator* g^{-1} *to the Monte Carlo input data (shown in Fig.* 5). *The data shown here is the result of taking* 100 *sections of unit area of a random array of spheres with density* 100 *per unit volume; the plotted points are means and the error bars are the standard deviations. One sees that the raw output is often negative, and thus cannot be accepted as a density estimate. The result of performing the* COMIVE *procedure is also shown ("optimum output"). The output values are always nonnegative now, which is indicated in the figure by not extending the error bars below the axis. The distance from the mean to the top of the error bar is one standard deviation.* COMIVE *increases the precision of the results.*

σ_0 is a measure of precision of the output data f^0. Because of the constraints imposed, the precision of f^0 is not only greater than that of f^R, but also is greater than that of the input data ϕ^I (Fig. 5). This is evidenced by the fact that the standard error bars for the back-substituted input ϕ^B are smaller than those of ϕ^I.

The *accuracy* of COMIVE does not increase as the ensemble size does, but will, of course, as the sample size increases. For a small sample, such as is illustrated in Figs. 4 and 5, the density of spheres with $R = .95$, which should be 100 is, on the average, less, because of the nonnegativity constraint. As the sample size increases the fraction in this interval increases as expected.

Results obtained with COMIVE for other pdf's (binary, lognormal) will be presented in Appendix B.

5. Conclusion. We have proposed an algorithm (COMIVE) for solving the stereology problem (Abel integral equation with noisy data) which we believe to be superior to any which have previously been used. The constrained least squares optimization procedure is designed to give the best possible answer which is consistent with positivity of density estimates. The procedure is completely stable and, in fact, enhances the precision of the input data.

The method has been illustrated using test data which was computer-generated by taking random slices of spheres. This idealized data lacks systematic effects which usually are present in real experimental data, but the principal benefits of our algorithm

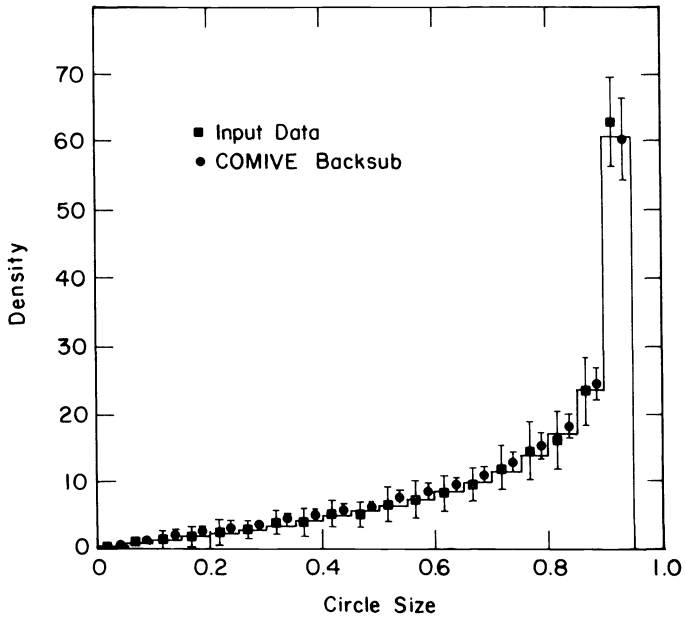


FIG. 5. *Same monosize case as Fig. 4. The histogram is the radius distribution which would be obtained for an infinitely large sample. "Backsub" is the result of applying the Abel operator g to the "optimum output" of Fig. 4. Closeness of "backsub" to the input, also shown here, is optimized by* COMIVE *subject to the constraint of nonnegativity of density estimates.*

will persist in any situation. It is a stable prescription for obtaining an optimal nonnegative density estimate f^0. From f^0, of course, all other statistics follow, such as moments and distributions.

One of the important unsolved problems in stereology is the determination of shape characteristics. Nicholson and Merckx (1969) use a matrix formulation to compute density and moment estimates for prescribed shapes of particles. Recent work by Beddow (1980) gives procedures for determining shape distribution parameters using 2D projections. A future project will be the adaptation of the methods of this paper to this morphological problem.

Appendix A. Discretization of (1). If (5) is substituted into (1), the result is

$$\phi(x) = 2x \sum_{i=1}^{k} \frac{f_i \theta(r_i - x)}{\sqrt{r_i^2 - x^2}} \tag{14}$$

(where $\theta(x) = 0$ if $x < 0$, $\theta(x) = 1$ if $x \geqq 0$), which is the density of circles for the density of spheres given by the superposition of δ-functions (5). The number of circles with radii in the interval $r_{i-1} < x < r_i$ is, per unit area,

$$\phi_i = \int_{r_{i-1}}^{r_i} \phi(x)\, dx. \tag{3}$$

When (14) is substituted into (3), one gets (4) with

$$g_{ij} = 2(\sqrt{R_j^2 - r_{i-1}^2} - \sqrt{R_j^2 - r_i^2}). \tag{15}$$

We call this the Saltikov discretization.

If, on the other, (6) is substituted into (1) we obtain the histogram discretization,

$$g_{ij} = \Delta^{-1}\Big\{ R_j\sqrt{R_j^2 - r_{i-1}^2} - R_{j-1}\sqrt{R_{j-1}^2 - r_{i-1}^2} - r_{i-1}^2 \log\left|\frac{R_j + \sqrt{R_j^2 - r_{i-1}^2}}{R_{j-1} + \sqrt{R_{j-1}^2 - r_{i-1}^2}}\right| - R_j\sqrt{R_j^2 - r_i^2} + R_{j-1}\sqrt{R_{j-1}^2 - r_i^2} + r_i^2 \log\left|\frac{R_j + \sqrt{R_j^2 - r_i^2}}{R_{j-1} + \sqrt{R_{j-1}^2 - r_i^2}}\right|\Big\}, \tag{16}$$

which is much more complicated than (15). It is understood in (17) that any term which contains the square root of a negative quantity is omitted, and we have assumed uniform interval size, with

$$r_i - r_{i-1} = \Delta. \tag{17}$$

Appendix B. Other applications of COMIVE. Figures 6 and 7 show results for binary and lognormal sphere pdf's, respectively. Again, the Saltikov discretization was used, and most of the remarks made in connection with Fig. 4 are relevant. The apparently large standard errors in the lognormal output should not mislead one—this is still an ensemble of 100 small samples and the scale on the ordinate is expanded.

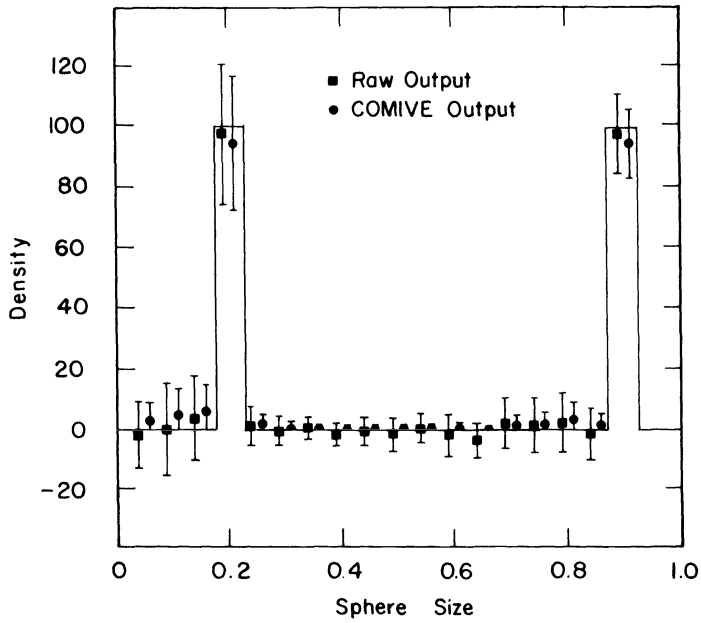


FIG. 6. *Output data when the input is generated by Monte Carlo on a sphere distribution consisting of equal numbers of radius* 0.9 *and* 0.2. *The statistics are generated from an ensemble of* 100 *sections like those used for Fig.* 4 *but with an equal density of the smaller spheres added. Other comments made about Fig.* 4 *also apply here.*

An example of the application of COMIVE to unfolding experimental data is shown in Fig. 8. This is a single sample of input consisting of observed circle sizes in slices of Swiss cheese which was used by Tallis (1970) to illustrate his method. As we mentioned above, Tallis interpolated to eliminate negative frequencies from his output.

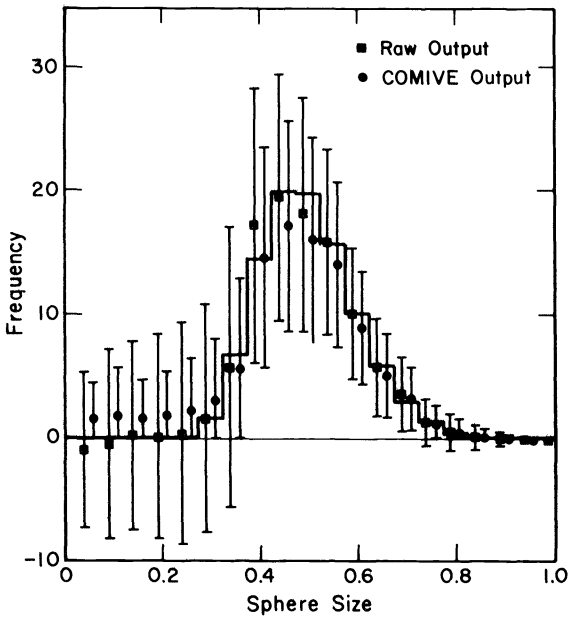


FIG. 7. *Like Fig. 6, but the input data are generated by Monte Carlo on a lognormal distribution of spheres, with mean radius .7 and standard deviation .2. The probability density is plotted here, rather than the number density of the previous figures.*

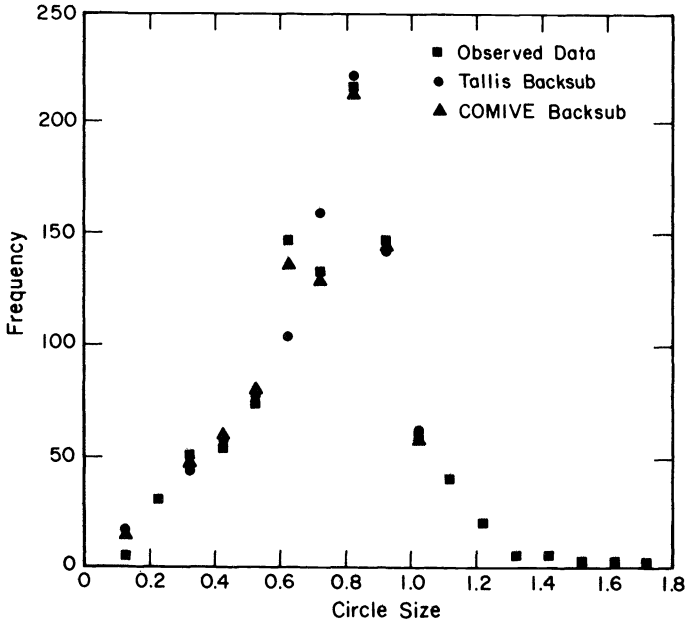


FIG. 8. *Example of Swiss cheese data used by Tallis* (1970) *to illustrate his unfolding method. Raw data (squares) is compared with the results of back-substituting Tallis output (circles) and the results of back-substituting the* COMIVE *optimum output (triangles). Only two points are significantly different; the discrepancy is caused by the different algorithms used by Tallis and* COMIVE *to eliminate negative components in the raw output. Tallis replaces raw output densities in the range of radii in which negative densities occur with a linear interpolation from the origin to the first positive density beyond the last negative one. This is reflected here by the Tallis back-substituted data being smoother than the raw data.*

Our algorithm is different; this is reflected by the figure, which shows the results of back-substituting Tallis' and our results contrasted with the original input.

Acknowledgment. The continued efforts of H. Dick Lewis to excite and sustain our interest in this subject deserve recognition and appreciation.

REFERENCES

R. S. ANDERSSEN AND A. J. JAKEMAN (1975), *Computational methods in stereology*, Fourth International Congress on Stereology, Natl. Bureau of Standards, Gaithersburg, MD.

J. K. BEDDOW (1980), *Particulate Science and Technology*, Chemical Publishing, New York.

R. T. DE HOFF (1967), *The estimation of particle size distributions from simple counting measurements made on random plane sections*, Trans. Metall. Soc. AIME, 233, pp. 25–29.

H. ELIAS (1967), *Editorial*, in Stereology, H. Elias, ed., Springer-Verlag, New York,.

——— (1980), *Stereology related to morphometry, tool of many disciplines.* Example, cancer prognosis, Abstracts of Papers of 146th Natl. Meeting AAAS, Jan. 3–8, San Francisco, AAAS Publication 80–2.

A. S. GOLDMAN, H. D. LEWIS AND W. M. VISSCHER (1979), *Constrained minimum variance estimator: A new procedure for estimating the distribution of spherical particles from plane sections* in Proc. of the Tech. Program Intl. Powder and Bulk Solids Handling and Processing, Philadelphia, May 15–17, pp. 151–160.

A. J. JAKEMAN (1975), *The numerical inversion of Abel-type integral equations*, Ph.D. Thesis, Australian National University, Canberra.

M. G. KENDALL AND P. A. P. MORAN (1963) *Geometrical Probability*, Hafner, New York.

C. L. LAWSON AND R. J. HANSON (1974) *Solving Least-Squares Problems*, Prentice-Hall, New York.

G. N. MINERBO AND M. E. LEVY (1969), *Inversion of Abel's integral equation by means of orthogonal polynomials*, SIAM J. Numer. Anal., 6, pp. 598–616.

W. L. NICHOLSON AND K. R. MERCKX (1969), *Unfolding particle size distributions*, Technometrics, 11, pp. 707–723.

S. A. SALTIKOV (1967), *The determination of the size distribution of particles in an opaque material from a measurement of the size distribution of their sections*, in Stereology, H. Elias, ed., Springer-Verlag, New York.

G. M. TALLIS (1970), *Estimating the distribution of spherical and elliptical bodies in conglomerates from plane sections*, Biometrics, 26, pp. 87–103.

G. WAHBA (1979), *Smoothing and ill-posed problems*, in Numerical Methods for Integral Equations with Applications, M. Golberg, ed., Plenum, New York.

G. S. WATSON (1971), *Estimating functions of particle size distributions*, Biometrika, 58, pp. 483–490.

S. D. WICKSELL (1925), *The corpuscle problem*, Biometrika, 17, pp. 84–99.

——— (1926), *The corpuscle problem, part* II, Biometrika, 18, pp. 151–172.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0013 $01.25/0

MULTIDIMENSIONAL ADDITIVE SPLINE APPROXIMATION*

JEROME H. FRIEDMAN,† ERIC GROSSE‡ AND WERNER STUETZLE§

Abstract. We describe an adaptive procedure that approximates a function of many variables by a sum of (univariate) spline functions s_m of selected linear combinations $a_m \cdot x$ of the coordinates

$$\phi(x) = \sum_{1 \leq m \leq M} s_m(a_m \cdot x).$$

The procedure is nonlinear in that not only the spline coefficients but also the linear combinations are optimized for the particular problem. The sample need not lie on a regular grid, and the approximation is affine invariant, smooth, and lends itself to graphical interpretation. Function values, derivatives, and integrals are inexpensive to evaluate.

Key words. multivariate approximation, surface fitting, projection pursuit

1. Introduction. Multidimensional surface approximation is recognized as an important problem for which several methodologies have been developed. The aim is to construct an approximation $\phi(x)$ to a p-dimensional surface $y = f(x)$ on the basis of (possibly noisy) observations $\{(y_i, x_i)\}_{1 \leq i \leq n}$. Most existing methods, such as tensor product splines, kernels, and thin plate splines (for a survey, see Schumaker [1976]), are linear in that

$$\phi(x) = \sum_{1 \leq i \leq n} w_i y_i,$$

where the weights $\{w_i\}$ depend only on x and $\{x_i\}_{1 \leq i \leq n}$, but not on $\{y_i\}_{1 \leq i \leq n}$. These methods have the advantage that they are straightforward to compute and their theory is tractable. In practice, however, they are limited because they cannot take advantage of special properties of the surface. Due to the inherent sparsity of high-dimensional sampling, procedures successful in high dimensions must be adaptive and thus nonlinear.

In this paper we describe an adaptive procedure that approximates $f(x)$ by a sum of (univariate) spline functions s_m of selected linear combinations $a_m \cdot x$ of the coordinates

$$\phi(x) = \sum_{1 \leq m \leq M} s_m(a_m \cdot x). \tag{1}$$

The procedure is nonlinear in that not only the spline coefficients but also the linear combinations are optimized for the particular problem.

2. The algorithm. The spline function s_m along $a_m \cdot x$ is represented as a sum of j_m B-splines (de Boor [1978]) of order q

$$s_m(a_m \cdot x) = \sum_{1 \leq j \leq j_m} \beta_{mj} B_{mj}(a_m \cdot x). \tag{2}$$

* Received by the editors May 8, 1980, and in revised form January 15, 1982. This work was supported by the Department of Energy under contracts DE-AC03-76SF00515 and DE-AT03-81-ER108 43, the Office of Naval Research under contract ONR N00014-81-K-0340, and the National Science Foundation under grant MCS 78-17697.

† Stanford Linear Accelerator Center, Stanford, California 94305.

‡ Computer Science Department, Stanford University, Stanford, California 94305. Present address: Bell Laboratories, Murray Hill, New Jersey 07974.

§ Stanford Linear Accelerator Center and Department of Statistics, Stanford University, Stanford, California 94305.

The approximation $\phi(x)$ (given by (1) and (2)) is specified by the directions $\{a_m\}_{1\leqq m\leqq M}$, the knot sequences along $a_m \cdot x$ for $1\leqq m\leqq M$, and the B-spline coefficients $\{\beta_{mj}\}_{1\leqq m\leqq M, 1\leqq j\leqq j_m}$. For particular $\{a_m\}$, the knots are placed heuristically and then the $\{\beta_{mj}\}$ are determined by (linear) least squares. The residual sum of squares from this fit is taken to be the inverse figure of merit for $\{a_m\}_{1\leqq m\leqq M}$.

Following Friedman and Stuetzle [1981], the approximation is constructed in a stepwise manner: given $\{a_m\}_{1\leqq m\leqq M-1}$, find a_M to optimize the figure of merit of $\{a_m\}_{1\leqq m\leqq M}$. Terminate when the figure of merit is below a user specified threshold.

3. Implementation. A difficult part of the algorithm is finding each direction a_m. We perform a numerical search using a Rosenbrock method (Rosenbrock [1966]). This method is easily modifiable to search over the unit sphere. We have found empirically that each iteration of the optimizer requires approximately $3.5p$ function evaluations, where p is the dimension of x. Two iterations are nearly always sufficient. As the search usually starts far from the solution and the solution does not have to be obtained with high precision, it does not seem likely that optimization procedures that estimate the Hessian would do better.

For high dimensionality, the computation is dominated by the evaluations of the object function. Since it is not crucial to find the precise optimum, considerable savings are achieved by substituting a similar, but much less expensive figure of merit during the search for a new direction. For this figure of merit not only the previously found directions but also the corresponding spline coefficients are held fixed. For a given direction, the residuals are modelled by (basically) a moving average smooth (see Friedman and Stuetzle [1981]). The characteristic bandwidth (the fraction of observations over which averaging takes place) is taken to be inversely proportional to the number of knots. The residual sum of squares from the smooth is the figure of merit used for the smooth. Solving the least squares problem for the original figure of merit requires

$$O\left[n\left(\sum_{1\leqq m\leqq M} j_m\right)^2\right]$$

operations, while the new figure of merit can be evaluated in roughly n operations using updating formulas for the moving average. The least squares problem has to be solved only once for each iteration to determine the new model after a_m has been found.

To solve the least squares problem, we form the normal equations and use a pseudo-inverse, since the design matrix might not have full rank. The singularity which arises from the inclusion of a constant term for each direction is remedied by simply dropping one column per direction from the design matrix. Higher order singularities caused, for example, by the linear terms for three co-planar directions, are not explicitly taken care of, but are handled by the pseudo-inverse.

Our knot placement procedure is motivated by the sequential nature of the algorithm. At each iteration, the knot positions are required for the least squares fit, after the new direction has been found. Our model at this point is the spline fit of the previous iteration, plus the moving average smooth along the newly found direction. The knot placement is based on the residuals $\{r_i\}$ from this model. Multidimensional structure in these residuals due to incompleteness of the model manifests itself as high local variability in the scatterplots of r_i against $a_m \cdot x_i$. In order to preserve the ability of fitting this structure in further iterations, it is important to avoid accounting for it by spurious fits along existing directions. For this reason we place fewer knots in regions of higher local variability. Since the residuals change, the knots are replaced along all directions at each iteration.

The knots along a direction a_m are placed as follows: the smooth described above is applied to $\{(r_i, a_m \cdot x_i)\}_{1 \leqq i \leqq n}$ and the local variability v_i at each point is taken to be the average squared residual from its local linear fit. The Winsorized local variabilities are defined by

$$w_i = \begin{cases} 2\bar{v} & \text{if } v_i > 2\bar{v}, \\ \frac{1}{2}\bar{v} & \text{if } v_i < \frac{1}{2}\bar{v}, \\ v_i & \text{otherwise} \end{cases}$$

(where $\bar{v} = (1/n) \sum_{1 \leqq i \leqq n} v_i$), and then are scaled so that $\sum_{1 \leqq i \leqq n} 1/w_i = 1$. The knots $\{t_i\}$ are placed to divide the line into intervals with equal content of $1/w_i$:

$$\text{for each } l, \quad \frac{1}{j_m - q + 1} = \sum_{a_m \cdot x_l \in [t_l, t_{l+1}]} \frac{1}{w_i}.$$

4. Procedure parameters. The operation of the procedure is controlled primarily by two parameters; these are the number of knots taken along each direction and the termination threshold. Both parameters can be adjusted using graphical output produced by the program. The adequacy of the number of knots and their placement can be judged by examination of the residuals from the final model plotted against each $a_m \cdot x$. A systematic pattern in any one of these plots indicates that either the number of knots is too small or that the knot placement algorithm did not perform well. Another indication that the number of knots might be insufficient is that the procedure chooses nearly the same direction twice, thereby effectively doubling the number of knots placed along that direction.

The value set for the termination threshold determines the number of terms making up the model. Various criteria can be used to decide whether a particular term should be included. In the case of noisy data, one can ask whether a term is significantly different from zero (given all previous terms), or whether the addition of the term reduces the predictive mean squared error of the model. Also, considerations outside the data having to do with the problem setting can influence such a decision. In order to judge statistical significance, it is necessary to know, by how much one would expect an additional term to increase the figure of merit if there were no structure in the residuals. This can be estimated with a permutation test. The residuals (from the previous terms) are randomly permuted among the observations, thereby guaranteeing no structure in the (permuted) data. MASA is applied to these residuals and the increase in figure of merit noted. This process can be repeated, obtaining a (null) distribution of the figure of merit. Either formal or informal hypothesis testing techniques can then be used to judge whether the nonpermuted figure of merit is significant.

The optimal number of terms with respect to prediction error can be estimated by cross validation. The observations are randomly divided into L (typically 5–10) subsamples. Each of the subsamples are in turn set aside and the model constructed from the remaining observations. Each observation is set aside exactly once. The mean squared prediction error averaged over the set aside observations is taken as an estimate of the model mean squared error. Such an estimate can be made for models with differing numbers of terms and that model minimizing the cross validated mean-squared error estimate is then selected. Both permutation tests and cross validation can be implemented in a small driver routine which calls MASA repeatedly.

5. Examples. In this section we present and discuss the results of applying the multidimensional spline approximation method (MASA) to four examples. (A

FORTRAN program implementing MASA is available from the authors.) The first three examples were suggested elsewhere for testing surface approximation procedures. The function in the fourth example was studied in connection with a problem in mathematical genetics.

The first example is taken from Friedman [1979]. In this example uniformly distributed random points $\{x_i | 1 \leqq i \leqq 200\}$ were generated in the six-dimensional hypercube $[0, 1]^6$. Associated with each point x_i was a surface value

$$y_i = 10 \sin(\pi x_i(1) x_i(2)) + 20[x_i(3) - 0.5]^2 + 10x_i(4) + 5x_i(5) + 0x_i(6) + \varepsilon_i,$$

where the $\{\varepsilon_i\}$ were independent identically distributed standard normal. The inverse figures of merit for the approximation with $M = 1, \cdots, 4$ terms were 6.71, 4.29, 1.87, 0.97. In three restarts, the figure of merit did not decrease below 0.86, so $M = 4$ was chosen. The four linear combinations and the corresponding spline functions are shown in Figs. 1a–1d. (The function value is plotted on the vertical axis, $a \cdot x$ on the horizontal axis. The "+" signs on the bottom of the graph indicate the knot positions. A "+" sign followed by a number indicates multiple knots. For completeness, the program parameters are also listed; see comments in the program source code for a detailed explanation.) The spline along the first linear combination (Fig. 1a) is seen to model the linear part of the surface. The second term in the approximation (Fig. 1b) models the additive quadratic dependence on $x(3)$. The final two terms (Figs. 1c, 1d) model the interaction between $x(1)$ and $x(2)$. The L_2 norm of the error $\|f - \phi\|_2$ was 0.57.

Although the full advantages of MASA compared to other procedures are realized in higher dimensional or noisy settings, we applied it to two bivariate examples used

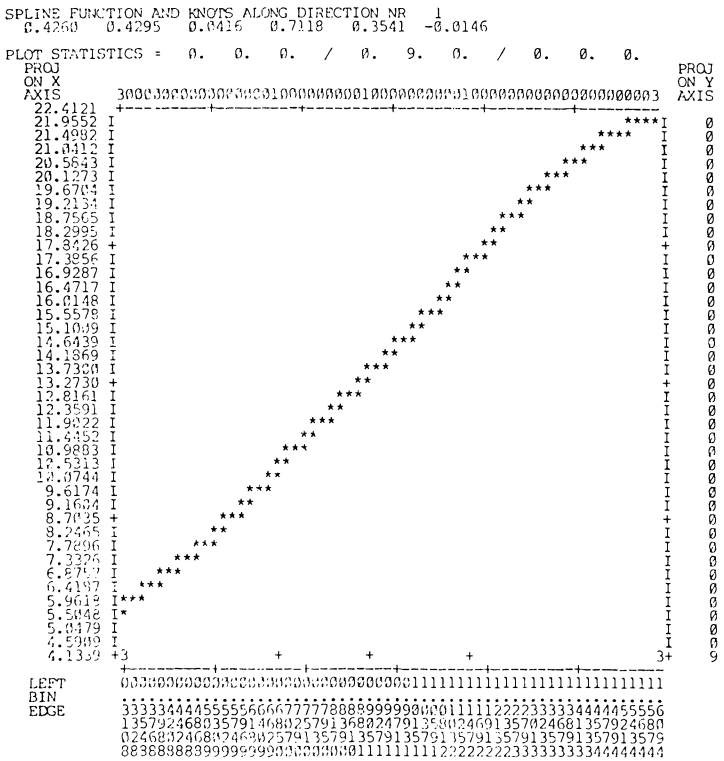


FIG. 1a

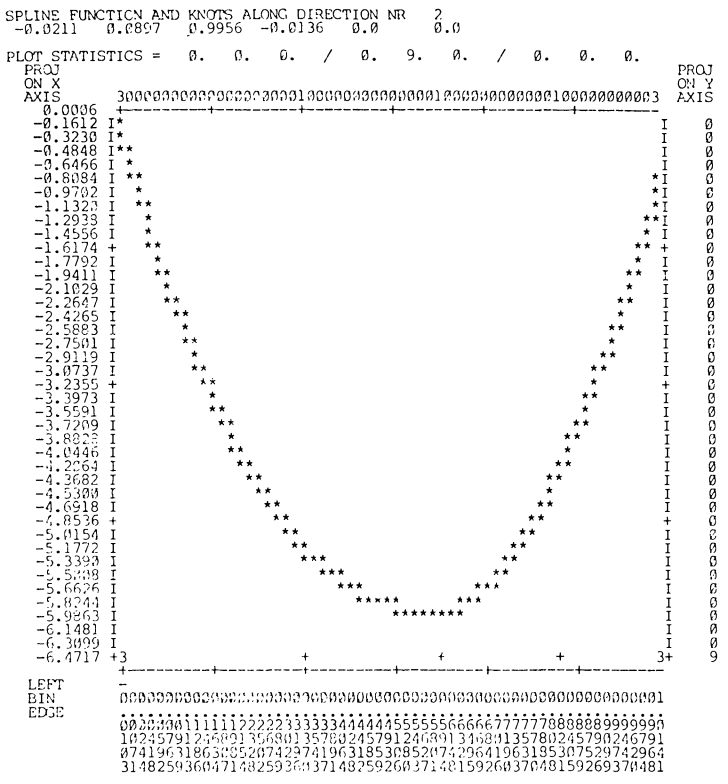

SPLINE FUNCTION AND KNOTS ALONG DIRECTION NR 2
-0.0211 0.0897 0.9956 -0.0136 0.0 0.0
PLOT STATISTICS = 0. 0. 0. / 0. 9. 0. / 0. 0. 0.
PROJ ON X AXIS
PROJ ON Y AXIS
LEFT
BIN
EDGE

FIG. 1b

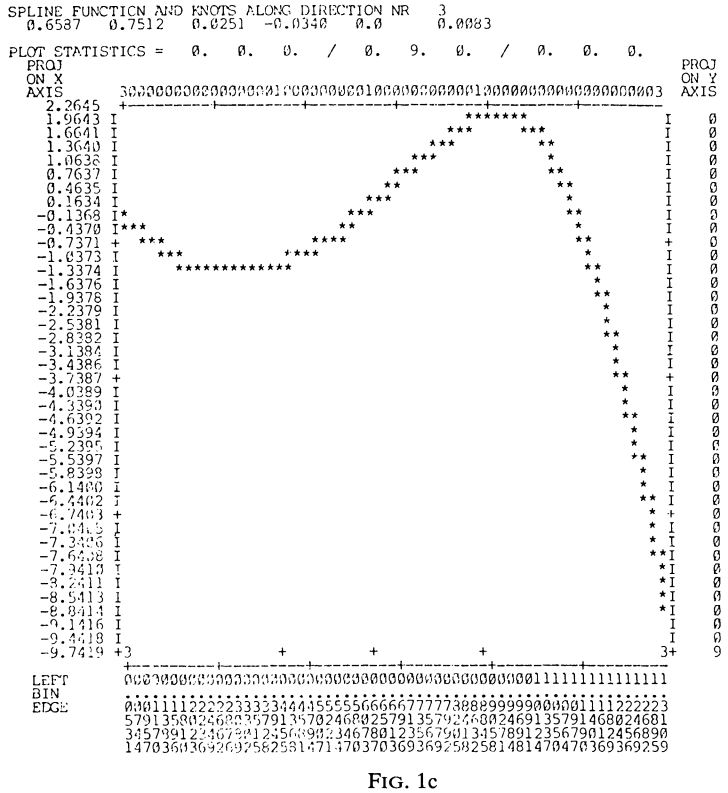

SPLINE FUNCTION AND KNOTS ALONG DIRECTION NR 3
0.6587 0.7512 0.0251 -0.0340 0.0 0.0083
PLOT STATISTICS = 0. 0. 0. / 0. 9. 0. / 0. 0. 0.
PROJ ON X AXIS
PROJ ON Y AXIS
LEFT
BIN
EDGE

FIG. 1c

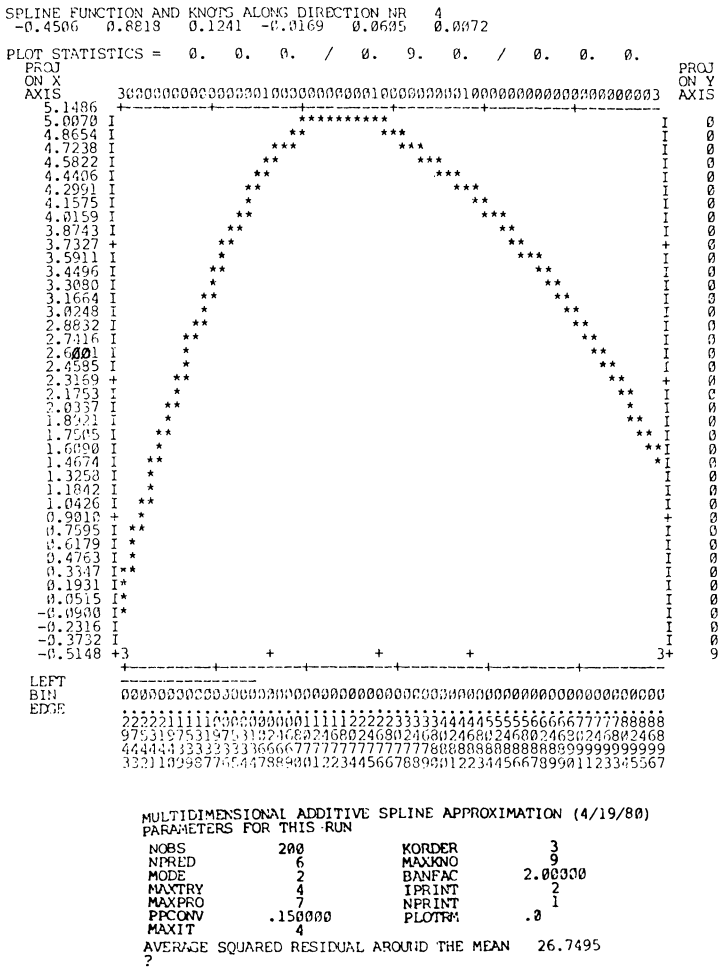


FIG. 1d

by Franke [1979] to compare a number of interpolatory surface approximation schemes. For both examples 100 uniformly distributed random points in the unit square $[0, 1]^2$ were generated. The function in Franke's first example is

$$f(x, y) = 0.75 \exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] + 0.75 \exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right]$$

$$+ 0.5 \exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] + 0.2 \exp\left[-(9x-4)^2-(9y-7)^2\right].$$

Considerations similar to those in the previous example led to an approximation with three terms. The linear combinations and corresponding spline functions are shown in Figs. 2a–2c.

The function in Franke's second example is

$$f(x, y) = \tfrac{1}{9}[\tanh(9y - 9x) + 1].$$

For this case the approximation used only one term, shown in Fig. 3.

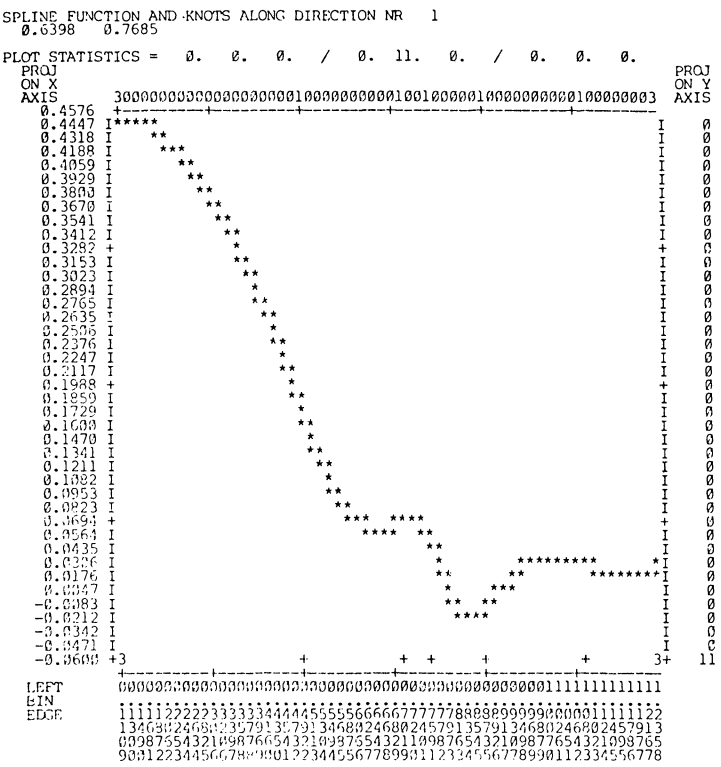

SPLINE FUNCTION AND KNOTS ALONG DIRECTION NR 1
0.6398 0.7685
PLOT STATISTICS = 0. 0. 0. / 0. 11. 0. / 0. 0. 0.
PROJ ON X AXIS
PROJ ON Y AXIS
0.4576
0.4447
0.4318
0.4188
0.4059
0.3929
0.3800
0.3670
0.3541
0.3412
0.3282
0.3153
0.3023
0.2894
0.2765
0.2635
0.2506
0.2376
0.2247
0.2117
0.1988
0.1859
0.1729
0.1600
0.1470
0.1341
0.1211
0.1082
0.0953
0.0823
0.0694
0.0564
0.0435
0.0306
0.0176
0.0047
-0.0083
-0.0212
-0.0342
-0.0471
-0.0600
LEFT BIN EDGE
RIGHT BIN EDGE

FIG. 2a

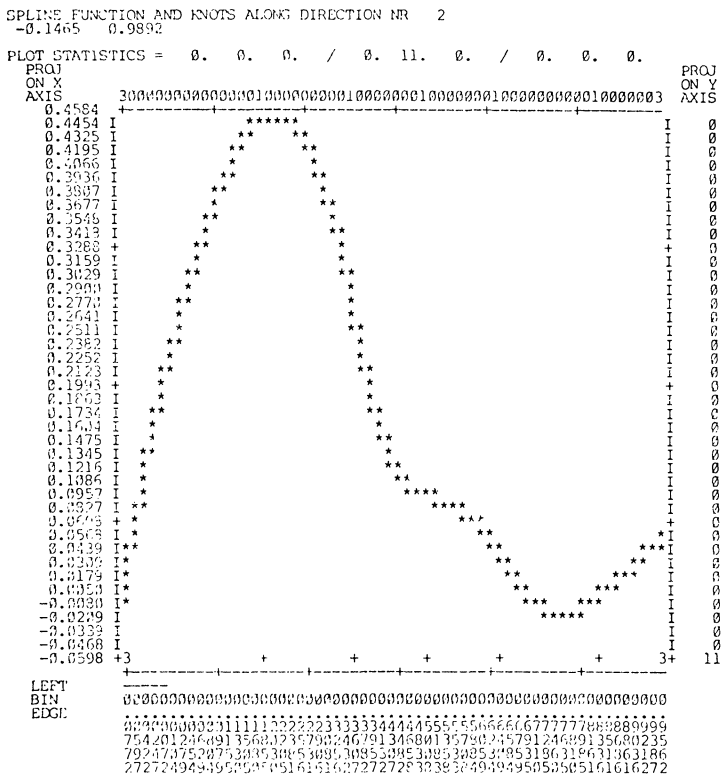

SPLINE FUNCTION AND KNOTS ALONG DIRECTION NR 2
-0.1465 0.9892
PLOT STATISTICS = 0. 0. 0. / 0. 11. 0. / 0. 0. 0.
PROJ ON X AXIS
PROJ ON Y AXIS
0.4584
0.4454
0.4325
0.4195
0.4066
0.3936
0.3807
0.3677
0.3548
0.3418
0.3288
0.3159
0.3029
0.2900
0.2770
0.2641
0.2511
0.2382
0.2252
0.2123
0.1993
0.1864
0.1734
0.1604
0.1475
0.1345
0.1216
0.1086
0.0957
0.0827
0.0698
0.0568
0.0439
0.0309
0.0179
0.0050
-0.0080
-0.0209
-0.0339
-0.0468
-0.0598
LEFT BIN EDGE
RIGHT BIN EDGE

FIG. 2b

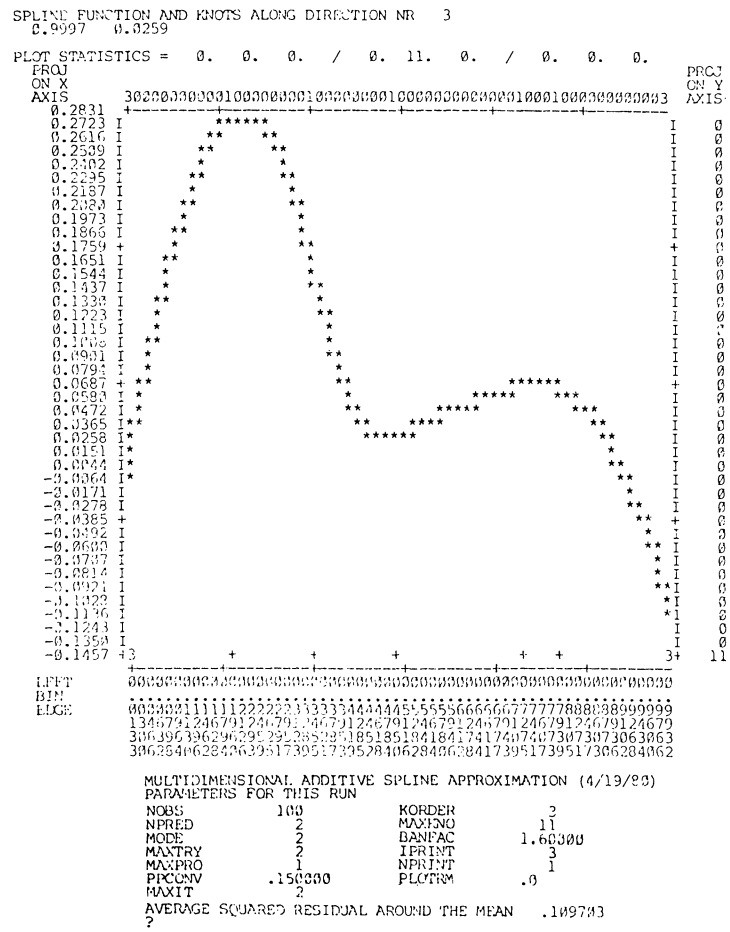


FIG. 2c

Since different random points were used in Franke's and our tests, precise comparisons are not possible. On the first example, MASA gave roughly an order of magnitude larger errors than the best methods in Franke's trials (global basis function methods) while on the second example, MASA gave an order of magnitude smaller errors than the best methods. These results are not surprising since the peak-shaped basis functions of the global basis methods are especially suited for representing the peaks of the first example, whereas the ridge-shaped basis functions of MASA are especially suited to the second example. Unfortunately, peak-shaped basis functions are not appropriate for moderate or higher dimensionality. The difficulty is that in order to achieve a smooth fit, the width of the basis peaks needs to be comparable to the distance between data points. For n uniformly distributed random points in a p-dimensional hypercube $[0, 1]^p$, the typical nearest neighbor distance is $(1/n)^{1/p}$. In particular for $n = 1000$ and $p = 10$, this distance is 0.5, and for $p = 20$ is 0.7. Thus variation of the surface over distances small compared to such large interpoint distances cannot be well approximated with these global basis functions methods.

Our final example is a 19-dimensional function encountered by Carmelli and Cavalli [1979]. An important question is the structure of this function near its

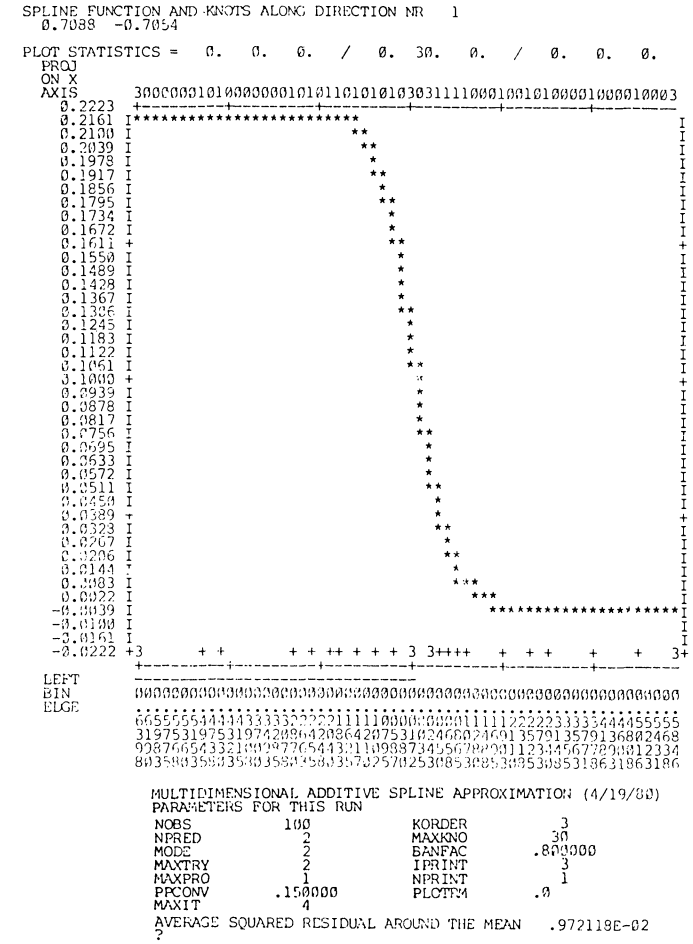


FIG. 3

minimum. We sampled the function at 200 points uniformly distributed in a small hypercube centered at the minimum found by numerical optimization and applied MASA. The inverse figure of merit for the best constant fit was 13.3. The inverse figure of merit for $M = 1$ was 0.78. In 30 restarts, the figure of merit did not decrease below 0.42. Figure 4 gives the linear combination and corresponding spline function. This shows that most of the structure in the likelihood function is revealed in this one projection. The structure certainly would not be easy to find by just looking at the definition of the function, and we know of no other approximation method that would yield this kind of information.

6. Discussion. MASA can be expected to work well to the extent that the surface can be approximated by a function of the form (1). Of course in the limit $M \to \infty$ all smooth surfaces can be represented by (1), but even for moderate M functions of this form constitute a rich class.

As seen in the previous section, an advantage of using essentially one-dimensional basis functions is the possibility of graphical interpretation. The entire model can be represented by graphing $s_m(a_m \cdot x)$ against $a_m \cdot x$ and by specifying $\{a_m\}_{1 \leqq m \leqq M}$ (perhaps

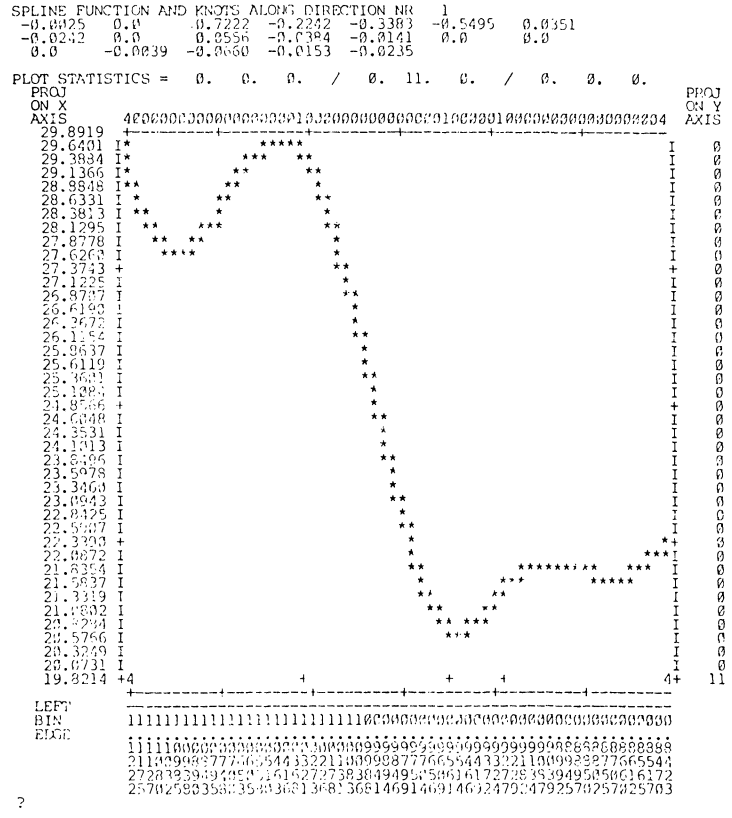


FIG. 4

graphically for $p = 2$ or 3). Additionally the graphical output is very helpful for setting the main procedure parameters, the number of knots along each direction and the termination threshold. Proper termination of the algorithm can be assured by monitoring at each iteration the plot of the residuals from the model of the previous iteration along the newly found direction.

The problem of sparse sampling in high dimensions is not encountered, since MASA is fitting one-dimensional projections of the entire sample. The sample need not lie on a regular grid, and the approximation is affine invariant and smooth. Function values, derivatives, and integrals are inexpensive to evaluate. In addition, since the approximation is locally quadratic for $q = 3$, optimization algorithms can be expected to converge rapidly. As only the directions, the knot positions and the B-spline coefficients have to be stored, MASA produces a very parsimonious description of the surface.

REFERENCES

CARL DE BOOR [1978], *A Practical Guide to Splines*, Springer-Verlag, New York.

DORIT CARMELLI AND L. L. CAVALLI-SFORZA [1979], *The genetic origin of the Jews: a multivariate approach*, Human Biology, 51, pp. 41–61.

WILLIAM S. CLEVELAND [1979], *Robust locally weighted regression and smoothing scatterplots*, J. Amer. Statist. Assoc., 74, pp. 829–836.

RICHARD FRANKE [1979], *A critical comparison of some methods for interpolation of scattered data*, Naval Postgraduate School report NPS-53-79-003.

JEROME H. FRIEDMAN AND WERNER STUETZLE [1981], *Projection pursuit regression*, J. Amer. Statist. Assoc., 76, pp. 817–823.

H. H. ROSENBROCK [1960], *An automatic method for finding the greatest or least value of a function*, Computer J., 3, pp. 175–184.

LARRY L. SCHUMAKER [1976], *Fitting surfaces to scattered data*, in Approximation Theory III, G. G. Lorentz, C. K. Chui and L. L. Schumaker, eds. Academic Press, New York, pp. 203–268.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0014 $01.25/0

ALGORITHMS FOR THE ANALYSIS OF SEVERAL 2×2 CONTINGENCY TABLES*

MARCELLO PAGANO† AND DAVID TRITCHLER‡

Abstract. This paper presents algorithms for calculating the exact permutation distributions of several 2×2 contingency tables. They address both the hypothesis of constant relative odds across the tables and inference about this common value. The efficiency of the algorithms is due to their not requiring the enumeration of all the tables consistent with the given marginals.

Key words. contingency tables, permutation distribution, relative risk, fast Fourier transform

1. Introduction. Zelen (1971) presents a methodology for analyzing k 2×2 contingency tables and lists several examples where it is useful. The complexity of the calculations required by the exact methods proposed by Zelen increases exponentially fast as the size of the problem increases. As a result, approximations based on asymptotics have been suggested. There has been some debate about these asymptotic methods (see Halperin, et al. (1977)). However, no clear and theoretically supported rules have been derived as to when these approximations are applicable or how close they are to the exact answer.

We propose algorithms to perform the exact calculations to determine whether the relative odds are constant across the k tables and to make inference about this common value. The algorithms take advantage of special features of the problem so that total enumeration of all possible tables, consistent with appropriate marginal restrictions, is unnecessary. Thus, as compared to total enumeration (Thomas (1975)), these algorithms have the effect of postponing the reliance on asymptotic methods to larger problems.

2. Theory. Using Zelen's (1971) notation, the jth table is

	Number of Successes	Number of Failures	Totals
Treatment 1	r_j	$m_j - r_j$	m_j
Treatment 2	s_j	$n_j - s_j$	n_j
Totals	t_j	$N_j - t_j$	N_j

for $j = 1, \cdots, k$.

Let θ_{ij} be the probability of success for treatment i in table j; $i = 1, 2, j = 1, \cdots, k$. Then the relative odds for the jth table are

$$\psi_j = \frac{\theta_{2j}/(1-\theta_{2j})}{\theta_{1j}/(1-\theta_{1j})}.$$

* Received by the editors October 23, 1980, and in revised form October 5, 1981.

† Harvard University and Sidney Farber Cancer Institute, Boston, Massachusetts 02115. The research of this author was supported in part by the National Institutes of Health under grants CA-23415 and CA-28066.

‡ Harvard University and Sidney Farber Cancer Institute, Boston, Massachusetts 02115. The research of this author was supported in part by the National Institutes of Health under grant CA-09337.

In order to determine whether the k tables may be combined, consider the hypothesis that all the ψ_j are equal. To this end, define the

$$C(s_j, t_j) = \binom{n_j}{s_j}\binom{m_j}{t_j - s_j}\binom{N_j}{t_j}^{-1}, \qquad j = 1, \cdots, k, \tag{1}$$

and let

$$s = \sum_{j=1}^{k} s_j$$

be the total number of successes observed for treatment 2. Let $\mathbf{s}$ be the vector whose jth component is s_j and define the set

$$\mathscr{S}_s = \left\{\mathbf{z}: \sum_{j=1}^{k} z_j = s,\ l_j \leqq z_j \leqq u_j\right\}, \tag{2}$$

where

$$l_j = \max(0, t_j - m_j) \quad \text{and} \quad u_j = \min(t_j, n_j).$$

Then, with $\mathbf{t}$ the vector whose jth component is t_j and

$$C(\mathbf{s}, \mathbf{t}) = \prod_{j=1}^{k} C(s_j, t_j), \tag{3}$$

the probability of the observed tables, under the hypothesis of constancy of the ψ_j is given by Zelen (1971) as

$$f_c(\mathbf{s}) = C(\mathbf{s}, \mathbf{t}) \Big/ \sum_{\mathbf{z} \in \mathscr{S}_s} C(\mathbf{z}, \mathbf{t}), \tag{4}$$

and the significance value of the observed tables is

$$P_c = \sum_{\mathbf{w} \in W_s} f_c(\mathbf{w}), \tag{5}$$

where W_s is the subset of $\mathscr{S}_s$ defined as

$$W_s = \{\mathbf{w}: \mathbf{w} \in \mathscr{S}_s \quad \text{and} \quad f_c(\mathbf{w}) \leqq f_c(\mathbf{s})\}.$$

Then the hypothesis of constant relative odds is rejected if P_c is too small (see Zelen (1971)).

If one is willing to accept the hypothesis that the ψ_j are all equal to ψ, say, then the next step is usually inference about ψ. To this end let

$$C(s, \mathbf{t}) = \sum_{\mathbf{z} \in \mathscr{S}_s} C(\mathbf{z}, \mathbf{t}).$$

Then the relevant frequency function, for the total number of successes on treatment 2, is

$$f(s) = C(s, \mathbf{t})\psi^s \Big/ \sum_{z \in Z} C(z, \mathbf{t})\psi^z, \tag{6}$$

where

$$Z = \left\{z: \sum_{j=1}^{k} z_j = z,\ l_j \leqq z_j \leqq u_j\right\}$$

is the set of all z's possible as $\sum z_j$, for admissible z_j's and

$$C(z, \mathbf{t}) = \sum_{\mathbf{z} \in \mathscr{S}_z} C(\mathbf{z}, \mathbf{t}).$$

As shown by Zelen (1971), this function can be used to make inference about ψ; for example, to test if ψ equals one, to set a confidence interval on ψ, to find a "maximum likelihood" estimate of ψ, to find the relative likelihood function, etc....

The two sets of quantities which are laborious to evaluate are:

(i) the $C(\mathbf{w}, \mathbf{t})$, for $\mathbf{w} \in W_s$, which appear in the numerator (implicitly) in (5) and

(ii) the $C(z, \mathbf{t})$, for $z \in Z$, which appear as coefficients in (6) and one of which $(z = s)$ appears in the denominators in (4) and (5).

These two sets of quantities require that $\prod_{j=1}^{k} (u_j - l_j + 1)$ configurations of possible successes for treatment 2 be generated. Then for each configuration, k probabilities must be evaluated and multiplied together and a possible "IF" executed (when $z = s$) to determine whether it belongs in W_s. Thus the amount of work increases exponentially as k and the number of observations increase.

This work can be significantly decreased by the algorithms we outline in the next three sections. Section 3 concerns itself with problem (i) above and § 4 with problem (ii) above. Section 5 applies the methods of § 4 to problem (i) above.

3. Tail probability for constant relative odds hypothesis. Since the quantity in the denominator in (4) is constant for all members of W_s appearing in (5), we can defer its calculation to the next section. This section addresses itself to the calculation of P_c under the assumption that the denominator in (5) is known.

To calculate P_c one could proceed by generating all the tables in $\mathcal{S}_s$ and determining which ones belong to W_s. As we show below, it is unnecessary to generate all tables in $\mathcal{S}_s$; but to make this argument more lucid it is advantageous to first illustrate how to calculate P_c by generating $\mathcal{S}_s$.

To determine all the tables in $\mathcal{S}_s$, generate all k-tuples $(z_1, \cdots, z_k)$ which are such that $l_j \leqq z_j \leqq u_j$ and $z_1 + \cdots + z_k = s$. To satisfy the summation constraint, consider the feasible region for the first table. Since the tables are generated in order, $z_2, z_3, \cdots, z_k$ could be as large as $\mu_2, \mu_3, \cdots, \mu_k$, so

$$z_1 \geqq \max\left(l_1, s - \sum_{j=2}^{k} u_j\right) \equiv l_1^*.$$

That is, z_1 must be large enough to permit the total to be s.

Similarly,

$$z_1 \leqq \min\left(u_1, s - \sum_{j=2}^{k} l_j\right) \equiv u_1^*.$$

In general, if the first $(i-1)z$'s have been chosen, then z_i must satisfy $l_i^* \leqq z_i \leqq u_i^*$, where

$$l_i^* = \max\left(l_i, s - \sum_{j=1}^{i-1} z_j - \sum_{j=i+1}^{k} u_j\right), \tag{7}$$

$$u_i^* = \min\left(u_i, s - \sum_{j=1}^{i-1} z_j - \sum_{j=i+1}^{k} l_j\right). \tag{8}$$

We use the convention that a summation is null if its lower bound is bigger than its upper bound.

Thus the complete enumeration algorithm for calculating P_c:

ALGORITHM 1.

1° Set $p = \prod_{j=1}^{k} C(s_j, t_j)$ and $P = 0$.

2° Set $i = 1$.

3° Calculate l_i^* and u_i^* as in (7) and (8).

4° Set $z_i = l_i^*$.
5° Set $i = i + 1$.
6° If $i < k$ go to 3°. Else,
7° Set $z_k = s - \sum_{i=1}^{k-1} z_i$.
8° Set $q = \prod_{j=1}^{k} C(z_j, t_j)$.
9° If $q \leqq p$, $P = P + q$.
10° Let $z_{k-1} = z_{k-1} + 1$ and $z_k = z_k - 1$.
11° If ($z_{k-1} \leqq u_{k-1}^*$ and $z_k \geqq l_k$) go to 8°. Else,
12° Set $i = k - 2$.
13° Set $z_i = z_i + 1$.
14° If $z_i \leqq u_i^*$ set $i = i + 1$ and go to 3°. Else,
15° Set $i = i - 1$. If $i > 0$ go to 13°. Else (z_1 has been exhausted),
16° $P_c = P / \sum_{z \in \mathcal{S}_s} C(z, t)$; the denominator is assumed known.

To obtain the coefficients $C(y_j, t_j)$ one can use the recursion

$$C(z+1, t) = \frac{(n-z)(t-z)}{(z+1)(m-t+z+1)} C(z, t),$$

where the subscripts have been dropped for clarity.

Another saving can be effected by the following considerations. Suppose the first $(k-2)$ tables have been set and let

$$p_j = C(z_j, t_j), \qquad j = 1, \cdots, k-2.$$

Also let

$$p = \prod_{j=1}^{k} C(s_j, t_j).$$

If

$$\prod_{j=1}^{k-2} p_j \leqq p, \tag{9}$$

then all the k-tuplets in $\mathcal{S}_s$ whose first $k-2$ components are $(z_1, \cdots, z_{k-2})$ are in W. This will save a number of "IF" executions (step 9°). If, on the other hand, (11) is not satisfied one must further check the last two tables.

To explain the checking of the last two tables consider a matrix of length $(u_{k-1}^* - l_{k-1}^* + 1)$ and width $(u_k - l_k + 1)$ whose (i, j)th entry is $C(x, t_{k-1})C(y, t_k)$, where $x = l_{k-1}^* + i - 1$ and $y = l_k + j - 1$. Then the entries of interest are the ones on the SW–NE diagonal where $x + y = a$, a constant. One can start at the SW corner ($z_{k-1} = l_{k-1}^*$, $z_k = u_k^*$) and proceed SW→NE. But one can take advantage of the fact that $C(x, t_{k-1})C(a-x, t_k)$ is unimodal (shown below). So one can proceed from the SW corner in the NE direction until one exceeds p, then descend from the NE corner ($z_{k-1} = u_{k-1}^*$, $z_k = l_k^*$) in the SW direction until p is again exceeded. No further checking need be done. If $(u_{k-1}^* - l_{k-1}^* + 1)$ and $(u_k - l_k + 1)$ are large, this directed enumeration can lead to a substantial saving, since a number of tables need not be generated (the loop 8° to 11°).

THEOREM. *The function*

$$f(j) = \binom{n}{j}\binom{N}{t-j}\binom{m}{s-j}\binom{M}{v-s+j}, \qquad j \text{ integer}$$

is unimodal as $l \leqq j \leqq u$, where l and u define the permissible range of j.

Proof. Consider

$$g(x) = \Gamma(x+1)\Gamma(s-x+1)$$

for continuous x. If it can be shown that g is log concave, any product of such functions is log concave, and the theorem is proved. To show that g is log concave,

$$\frac{d^2}{dx^2}\log\Gamma(x+1) = \int_0^\infty \frac{t\,e^{-(x+1)t}}{1-e^{-t}}\,dt, \qquad x > -1,$$

(see Abramowitz and Stegun (1964, p. 260)) which is nonnegative over the permissible range of x. Thus g is log concave and the theorem is proved. □

In order to take full advantage of the above theorem it would seem that the tables should be ordered so that Table 1 is the smallest and table k the largest, according to $(u_j - l_j)$. The procedure just described is implemented by replacing 9°–11° of the algorithm by the following steps.

9a° Set $\alpha = k-1$, $\beta = k$.
9b° If $q \leqq p$, $P = P + q$. Else,
9c° If $\alpha = k$, go to 12. Else,
9d° Set $\alpha = k$, $\beta = k-1$.
10° Let $z_\alpha = z_\alpha + 1$ and $z_\beta = z_\beta - 1$.
11° If ($z_\alpha \leqq \mu_\alpha^*$ and $z_\beta \geqq l_\beta^*$) go to 8°. Else,

All the calculations in this section pertain to the hypothesis that the relative risk is constant and the tables can be combined. If the hypothesis is going to be rejected, then P_c is small and quickly calculated, but presumably more often than not the tables can be combined. In this case P_c is large and it really is not necessary to know it exactly since the primary interest is whether P_c is greater than some value, such as the ubiquitous .05. If this is indeed the case, then, because P_c is accumulated for each new table (step 9°), once the value of P_c exceeds .05, no further tables need be generated if the actual probability is not of interest.

4. Probability function of total successes. Once it has been determined that the k tables can be combined, then attention focuses on the relative odds, ψ. One could evaluate the coefficients of the frequency function $f(s)$ (see (6)) by using an algorithm which is similar to the one in the previous section. Namely

1° Generate all k-tuples $(z_1, \cdots, z_k)$ with $l_j \leqq z_j \leqq u_j$.
2° For each k-tuple calculate $\prod_{j=1}^k C(z_j, t_j)$ and $z = \sum_{j=1}^k z_j$.
3° Combine the results of 2° to obtain $C(z, \mathbf{t})$ for each $z \in Z$.

Fortunately, it is unnecessary to perform any of these operations if one uses the algorithm described in this section.

Consider the independent random variables S_j with respective probability frequency functions

$$p_j(s) = C(s, t_j), \qquad l_j \leqq s \leqq u_j, \quad j = 1, \cdots, k.$$

Define the random variable

$$S = \sum_{j=1}^k S_j$$

with probability frequency function $p(\cdot)$. Then, with probability one, $l \leqq S \leqq u$, where $l = l_1 + \cdots + l_k$ and $u = u_1 + \cdots + u_k$. Furthermore, it is clear that the probability

function of S yields all the coefficients required in (6) and in the denominators of (4) and (5).

One can find the distribution of S, without generating all possible k-tuplets $(z_1, \cdots, z_k)$ in Z, by inverting its characteristic function.

Discrete Fourier inversion rule. For any $v \geqq u$,

$$p(z) = \frac{1}{v} \sum_{j=0}^{v-1} \chi\left(\frac{2\pi j}{v}\right) \exp\left(\frac{-2\pi i j z}{v}\right), \qquad l \leqq z \leqq u,$$

where $\chi(\cdot)$ *is the characteristic function of* S.

Proof. The random variable S takes on only a finite number of integer values and so $\chi(\cdot)$ is a trigonometric polynomial of order u. The rule is thus just a restatement of a basic theorem in Fourier series. □

Since the Fast Fourier Transform (FFT) can be used to invert the characteristic function, the v in the above theorem can be chosen so as to take full advantage of the FFT.

To find the characteristic function $\chi(\cdot)$, note that if $\chi_j(\cdot)$ is the characteristic function of S_j for $j = 1, \cdots, k$ then $\chi = \chi_1 \times \cdots \times \chi_k$, due to the independence of the S_j's.

Thus k calls to the FFT followed by a multiplication of the characteristic functions and a further call to the FFT to invert the resultant characteristic function, yields the probability frequency function of S. The total amount of work is approximately of the order of $(k+1)v \log v$ and extra storage requirement is approximately of the order of $3v$. These numbers should be contrasted with the previously noted exponential factors.

As a result of the above considerations, the coefficients of the frequency function $f(s)$ can be obtained without obtaining any elements in Z. Caution should be exercised when dealing with $f(s)$, which is possibly a very high degree polynomial. Overflow, underflow and ill-conditioning are dangers in calculating quantities of that form.

5. Tail probability for constant relative odds hypothesis II. One can apply the reasoning used in the previous section to the problem of finding P_c. For the random variables defined in the previous section, let

$$T_i = \sum_{j=i+1}^{k} S_j, \qquad i = 1, \cdots, k-1.$$

Thus T_i represents the total number of successes on treatment 2 in the last $(k-i)$ tables. Since the characteristic functions of the individual S have been calculated, it is straightforward to calculate the probability distribution of each of the T_i and to store this information. Suppose this has been done.

Now return to the algorithm, in § 3, for calculating P_c. Consider the first step, i.e., setting $z_1 = l_1^*$. If now $p_1 \Pr(T_1 = s - z_1) \leqq p$, then it is clear that each table in $\mathcal{S}_s$ whose first component is z_1 is also in W_s. Furthermore, it is unnecessary to generate all these tables to find the sum of their respective probabilities. Indeed, this sum is $p_1 \Pr(T_1 = s - z_1)$. The next set of tables to check would be those in $\mathcal{S}_s$ whose first component is $l_1^* + 1$ and continue in this manner until a z is found which is such that $C(z, t_1) \Pr(T_1 = s - z) > p$. Call this z, z^*. Now one can set $z = u_1^*$ and come in from the right until the appropriate probability exceeds p again. Call this value of z, z^{**}. Note, from the log concavity of the hypergeometric, that for all z_1, $z^* \leqq z_1 \leqq z^{**}$, $C(z_1, t_1) \Pr(T_1 = s - z_1) > p$. For these values of z_1 we can return to the algorithm described in § 3.

Clearly, once z_1 has been set, one can apply the above reasoning to the second component, z_2, and so on. One can thus modify the algorithm in § 3 by adding these two steps between Step 3° and Step 4°:

3.1° Call Algorithm 2.
3.2° If $l_i^* > u_i^*$, go to 15°. Else,

where

ALGORITHM 2.
1° Set $q = 1$ and $t = s$.
2° If $i > 1$, set $q = \prod_{j=1}^{i-1} C(z_j, t_j)$.
3° If $i > 1$, set $t = s - z_1 - \cdots - z_{i-1}$.
4° Set $j = l_i^* - 1$.
5° Do for $z = l_i^*, u_i^*$.
  5.1° Set $j = j + 1$.
  5.2° Set $q^* = q \times C(z, t_i) \times \Pr(T_i = t - z)$.
  5.3° If $q^* > p$, go to 6°. Else,
  5.4° $P = P + q^*$.
6° If $j = u_i^*$, go to 11°. Else,
7° Set $l_i^* = j$, $j = u_i^* + 1$, $k = l_i^* + 1$.
8° Do for $z = u_i^*, k, -1$.
  8.1° Set $j = j - 1$.
  8.2° Set $q^* = q \times C(z, t_i) \times \Pr(T_i = t - z)$.
  8.3° If $q^* > p$, go to 9°. Else,
  8.4° $P = P + q^*$.
9° If $q^* < p$, $u_i^* = l_i^*$. Else $u_i^* = j$.
10° Return.
11° If $q^* < p$, $k = 1$. Else $k = 0$.
12° Set $l_i^* = u_i^* + k$.
13° Return.

This algorithm requires more storage (approximately $\prod_{j=1}^{k} u_j$) than the algorithm given in § 3, but it seems clear that it should be much faster than that algorithm for large tables and/or large k.

6. Discussion. In order to obtain a measure of how the proposed algorithms compare with a complete enumeration algorithm, we performed some time trials. All computing was done on a DEC-2040 computer, and the times should be used for internal comparisons only.

Attention was focused on both the hypothesis of constant relative risk and calculating the coefficients in $f(\cdot)$ ((5) and (6)). We consider nine small tables,

7 6	1 3	5 14	7 2	1 6	8 5	4 5	3 6	6 6
1 2	5 3	2 2	5 3	7 3	7 3	6 7	7 5	7 5

We first took the first two tables ($k = 2$), then the first four ($k = 4$), then the first six ($k = 6$), and finally all nine tables ($k = 9$). We then multiplied each cell by two ($x = 2$) and repeated the experiment for $k = 2, 4, 6$; then repeated this experiment for $x = 3$. For $k = 9$ the experiment was not repeated for $x = 2$ or 3 because the complete enumeration algorithm would have taken much too much time. The results are displayed in Table 1.

TABLE 1
Timing in seconds.

k	x	P_c	CE	DS	DMS	$DS_{.05}$	$DMS_{.05}$
2	1	.41	.36	.44	.40	.42	.45
4	1	.43	.76	1.18	1.34	.93	1.24
6	1	.29	16.14	5.82	5.18	3.17	3.38
9	1	.57	23,715	1,289.98	134.70	326.45	14.64
2	2	.20	.38	.60	.60	.60	.60
4	2	.12	2.35	2.20	2.40	1.59	2.08
6	2	.02	522.8	59.79	44.86	59.79	44.86
2	3	.04	.69	.69	.71	.69	.71
4	3	.03	9.78	3.14	3.52	3.14	3.52
6	3	.001	5,796.5	305.45	227.78	305.45	227.78

The first three columns refer to which tables were chosen, the multiplier applied to each cell and the value of P_c (5) respectively. The fourth column refers to a complete enumeration algorithm. The fifth and seventh columns refer to the algorithms in §§ 3 and 4 without and with the .05 cutoff, respectively. The sixth and eighth columns refer to the algorithms in §§ 4 and 5 without and with the .05 cutoff, respectively.

Table 1 epitomizes our general findings:

(i) Except for small k and very small tables, the proposed algorithms are much faster than complete enumeration.

(ii) The order in which the tables are treated is important and the advice given in § 3 should be followed. For example, for $k = 4$, the data multiplier $= 2$ and the order of the tables reversed, the time rose from 1.59 secs to 2.29 secs (column 7).

(iii) The amount of time required increases dramatically as k and the total size of the problem increase.

(iv) The efficiency of the DMS algorithm is apparent when k is large, especially when a .05 cutoff is applied.

(v) In column 7, most of the required time is taken by the calculation of P_c. For example, when $k = 6$, $x = 1$, the calculation of f took only 1.91 secs—that is not to say that P_c took only 1.26 secs (3.17–1.91), since both calculations require the same overhead. But when the cell entries were multiplied by two, the time to calculate f increased to only 2.68 secs (of the total 58.36 secs).

This last point suggests that a mixed algorithm may be appropriate. Namely, if the size of the tables is relatively large, use an approximation to determine P_c, then use exact methods to determine $f(\cdot)$. This should be an acceptable compromise especially in the usual situation when P_c is of much less interest than $f(\cdot)$. But more research needs to be done on this point.

Acknowledgments. The authors gratefully acknowledge the help of Professors Leon J. Gleser and Marvin Zelen.

REFERENCES

M. ABRAMOWITZ AND I. A. STEGUN, eds. (1964). *Handbook of Mathematical Functions*, U.S. Government Printing Office, Washington, DC.

M. HALPERIN, J. H. WARE, D. P. BYAR, N. MANTEL, C. C. BROWN, J. KOZIOL, M. GAIL AND S. B. GREEN (1977), *Testing for interaction in an $I \times J \times K$ contingency table*, Biometrika, 64, pp. 271–275.

D. G. THOMAS (1975), *Exact and asymptotic methods for the combination of 2×2 tables*, Comp. Biom. Research, 8, pp. 423–446.

M. ZELEN (1971), *The analysis of several 2×2 contingency tables*, Biometrika, 58, pp. 129–137.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0015 $01.25/0

COMPUTING FORWARD-DIFFERENCE INTERVALS FOR NUMERICAL OPTIMIZATION*

PHILIP E. GILL,† WALTER MURRAY,† MICHAEL A. SAUNDERS† AND MARGARET H. WRIGHT†

Abstract. When minimizing a smooth nonlinear function whose derivatives are not available, a popular approach is to use a gradient method with a finite-difference approximation substituted for the exact gradient. In order for such a method to be effective, it must be possible to compute "good" derivative approximations without requiring a large number of function evaluations. Certain "standard" choices for the finite-difference interval may lead to poor derivative approximations for badly scaled problems. We present an algorithm for computing a set of intervals to be used in a forward-difference approximation of the gradient.

Key words. nonlinear optimization, finite-difference approximation, nonderivative method

1. Introduction. When minimizing a smooth multivariate function whose derivatives are not available, an obvious strategy is to use a so-called *finite-difference gradient method*, in which each occurrence of the exact gradient in a first-derivative method is replaced by a *finite-difference approximation*. The success of this approach depends on obtaining "good" approximations to the necessary first derivatives. However, in order for such methods to be competitive with alternative nonderivative methods, it is essential that the gradient approximation should require only a small number of function evaluations. (The efficiency can also be improved by modifications that tend to reduce the number of gradient evaluations—e.g., using a line-search procedure based on function values only.)

We shall describe an automatic procedure whose purpose is to compute finite-difference intervals that produce "acceptable" forward-difference approximations of the gradient during an iterative minimization procedure. Note that our intention is *not* to compute the most accurate possible estimate of the gradient at a single point. The motivation for the algorithm to be described is the same as that of an earlier algorithm suggested by some of the authors (Gill, Murray and Wright (1981)). The present algorithm differs in the initial choice of interval, in the estimates used to define termination and in some steps of its logic in certain situations; in addition, a different strategy is suggested when switching to a central-difference estimate and for computing intervals at an arbitrary point.

The general problem of finding approximate derivatives by finite differences is known as *numerical differentiation*. A recent discussion of methods for numerical differentiation is given by Lyness (1977). The topic is also considered in Anderssen and Bloomfield (1974), Dahlquist and Björck (1974), Lyness and Moler (1967), Lyness and Sande (1971), Oliver and Ruffhead (1975) and Oliver (1980). Many automatic differentiation routines attempt to minimize the total error in a finite-difference approximation at a given point and tend to require a significant number of function

* Received by the editors November 10, 1981, and in revised form March 25, 1982. This research was supported by the U.S. Department of Energy under contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018, the National Science Foundation under grants MCS-7926009 and ECS-8012974, the Office of Naval Research under contract N00014-75-C-0267, and the U.S. Army Research Office under contract DAAG29-79-C-0110.

† Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305.

evaluations—usually, at least ten per derivative (see, e.g., Dumontet and Vignes (1977) and Stepleman and Winarsky (1979)). Therefore, standard numerical differentiation techniques are not appropriate within a finite-difference gradient method.

Section 2 contains a brief overview of the errors associated with the relevant finite-difference formulae. The proposed algorithm for computing an interval is presented and discussed in § 3, along with numerical results. Section 4 discusses the procedure to be used in multivariate optimization and explains the differences between our approach and alternative methods.

2. Errors in finite-difference approximations. For simplicity of description, we shall initially consider the error in estimating the first derivative of the smooth *univariate* function $f(x)$. In general, the errors in finite-difference approximations depend on quantities that are *unknown*, such as higher derivatives of the function. Therefore, the derivation of a representation of the error is useful mainly in indicating how the error varies with the finite-difference interval. However, under certain assumptions about f and its derivatives, a good a priori finite-difference interval can be specified (see § 2.3).

2.1. The forward-difference formula. The simplest approximation involves the *forward-difference formula*, in which $f'(x)$ is approximated by the quantity

$$\varphi_F(f, h) = \frac{f(x+h)-f(x)}{h} \tag{1}$$

for some $h > 0$, where the subscript F denotes "forward difference". The analysis in this section also applies (with trivial modifications) to the *backward-difference formula*

$$\varphi_B(f, h) = \frac{f(x)-f(x-h)}{h}.$$

When using the finite-difference formula (1), there are three sources of error in the approximation to $f'(x)$.

(i) *Truncation error.* The *truncation error* consists of the neglected terms in the Taylor series, namely

$$\varphi_F(f, h) - f'(x) = \frac{h}{2}f''(x) + \frac{h^2}{6}f'''(x) + \cdots = \frac{h}{2}f''(\xi) \equiv T_F(h), \tag{2}$$

where ξ is a point in the interval $[x, x+h]$.

(ii) *Condition error.* In practice, the computed function values to be used in calculating φ_F will be subject to error. Let the positive quantitiy ε_A denote a bound on the absolute error in the computed function values at x and $x+h$. It will be assumed throughout this paper that the value of ε_A at the given point is available; an effective technique for computing ε_A is given by Hamming (1973), and is also described in Lyness (1977) and Gill, Murray and Wright (1981).

Let $\hat{f}(x)$ and $\hat{f}(x+h)$ denote the computed values of $f(x)$ and $f(x+h)$ (including errors that may result from representing x, h and $x+h$ in finite precision). By assumption, the computed function values satisfy

$$\hat{f}(x) = f(x) + \theta_0\varepsilon_A \quad \text{and} \quad \hat{f}(x+h) = f(x+h) + \theta_h\varepsilon_A,$$

where $|\theta_0| \leqq 1$ and $|\theta_h| \leqq 1$. If the inexact function values are used in (1) and no other errors are made, it follows that

$$\varphi_F(\hat{f}, h) - \varphi_F(f, h) = \frac{2\psi\varepsilon_A}{h} \equiv C_F(h) \tag{3}$$

for some ψ, $|\psi| \leqq 1$. The error $C_F(h)$ in the value of $\varphi_F(\hat{f}, h)$ due to inaccurate values of f will be termed the *condition error* (see Lyness (1977))—sometimes known as *cancellation error* (see, e.g., Kahan (1973))—and satisfies

$$|C_F(h)| \leqq \frac{2\varepsilon_A}{h}.$$

(iii) *Rounding error in the calculation of* φ_F. Given $\hat{f}(x)$ and $\hat{f}(x+h)$, the calculation of a forward-difference approximation involves rounding errors in performing the associated subtraction and division. However, these errors are generally *negligible* with respect to the truncation and condition errors, and will be ignored in our analysis. The error in the computed approximation can thus be viewed as *the sum of the truncation error and the condition error.*

Example 1. All calculations given in this paper were performed using short precision on an IBM 370. The machine precision ε_M is $16^{-5} \approx 9.537 \times 10^{-7}$, and numbers in the examples have therefore been rounded to (at most) six figures (irrelevant higher-order digits are omitted).

To illustrate the effects of errors in a forward-difference approximation, consider the function

$$f(x) = (e^x - 1)^2 + \left(\frac{1}{\sqrt{1+x^2}} - 1\right)^2. \tag{4}$$

Forward-difference approximations to f' were calculated at the point $x = 1$ for various values of h, starting with $h = \varepsilon_M$. Table 1 contains the results of the computation. The first column contains the values of h. The second column contains $|T_F(h)|$, the magnitude of the truncation error that would be incurred by using the exact $\varphi_F(h)$ to approximate f'; the third column contains the magnitude of the condition error, which was calculated using the exact value of φ_F. (The "exact" value of φ_F was obtained by double precision calculation.) The fourth column contains the computed values of $\varphi_F(\hat{f}, h)$ and the final column contains the exact error $|f'(x) - \varphi_F(\hat{f}, h)|$.

A convenient bound on the overall error in a forward-difference approximation is

$$E_F = \frac{h}{2}|f''(\xi)| + \frac{2}{h}\varepsilon_A. \tag{5}$$

If $f''(\xi)$ is nonzero, the interval that minimizes the bound (5) is

$$h_F^* = 2\sqrt{\frac{\varepsilon_A}{|f''(\xi)|}}. \tag{6}$$

TABLE 1

Condition and truncation errors in φ_F *for Example* 1. $f(x) = 3.03828$, $f'(x) = 9.54866$.

h	$\lvert T_F(h)\rvert$	$\lvert C_F(h)\rvert$	$\varphi_F(\hat{f}, h)$	$\lvert f'(x) - \varphi_F(\hat{f}, h)\rvert$
9.537×10^{-7}	1.15697×10^{-5}	2.54867	7.00000	2.54865
9.537×10^{-6}	1.15711×10^{-4}	4.87710×10^{-2}	9.50000	4.86536×10^{-2}
9.537×10^{-5}	1.15718×10^{-3}	2.98130×10^{-2}	9.52000	2.86541×10^{-2}
9.537×10^{-4}	1.15790×10^{-2}	2.23475×10^{-3}	9.55800	9.34600×10^{-3}
9.537×10^{-3}	1.16520×10^{-1}	2.75015×10^{-4}	9.66490	1.62462×10^{-1}
9.537×10^{-2}	1.24192	2.66112×10^{-6}	1.07906×10^{1}	1.24192

When $h = h_F^*$, the values of the bounds on the truncation and condition errors are equal, and the associated value of E_F is $2\sqrt{\varepsilon_A |f''(\xi)|}$.

It is of interest to analyze the conditions under which the *relative error bound* $E_F/|f'(x)|$ will be small. If we assume that the calculated values of f are an acceptable representation of the exact values, the value ε_A provides a measure of the "noise level", i.e., the smallest meaningful perturbation in f. (Note that ε_A *need not necessarily be "small" relative to* $|f|$ *at all points*; for example, it may happen that $\varepsilon_A = 10^{-6}$ and $f = 10^{-8}$.) This interpretation suggests using ε_A to estimate a corresponding "noise-level" perturbation h_A in x. Expanding f in a Taylor series, we obtain

$$\varepsilon_A \approx f(x + h_A) - f(x) = h_A f'(\xi),$$

where $\xi \in [x, x + h_A]$. If we assume that $f'(\xi)$ is similar in magnitude to $f'(x)$, then the value

$$h_A \approx \frac{\varepsilon_A}{|f'(x)|} \tag{7}$$

may be used as a *lower bound* on any meaningful finite-difference interval. Any perturbation h smaller than h_A will produce a change in f that is less than noise level; if h is close to h_A, the resulting change in f will be almost at noise level.

If the forward-difference interval h_F^* (6) satisfies $h_F^* \gg h_A$, then it follows from (6) and (7) that the relative error bound

$$\frac{E_F}{|f'(x)|} = \frac{2\sqrt{\varepsilon_A f''(\xi)}}{|f'(x)|}$$

will be small. On the other hand, we observe that this relative error bound will tend to increase as $|f'(x)|$ decreases, assuming that $|f''(\xi)|$ does not decrease correspondingly. Therefore, in general *even the best forward-difference approximation to* f' *will eventually be unreliable when* $|f'(x)|$ *becomes small.* (In terms of the preceding analysis, the relative accuracy deteriorates as h_F^* and h_A become similar in magnitude.)

We have emphasized the *relative error* in a forward-difference approximation to f' because *the relative accuracy of the gradient vector affects the quality of the search direction in finite-difference gradient methods.* For example, in a finite-difference quasi-Newton method, the search direction p is the solution of a linear system of the form

$$Mp = -\bar{g},$$

where $\bar{g}$ is the gradient approximation. If the relative accuracy in $\bar{g}$ compared to the exact gradient is poor, p may not have the properties required in the algorithm—for instance, p may not be a descent direction. (In § 4.3, we describe a procedure that may be used when the forward-difference approximation is no longer acceptable.)

2.2. The central-difference and second-order formulae. The first derivative can also be approximated using the *central-difference formula*

$$\varphi_C(f, h) = \frac{f(x + h) - f(x - h)}{2h},$$

where $h > 0$. In this case, the truncation error is bounded by $\frac{1}{6}h^2|f'''(\mu)|$, where $\mu \in [x - h, x + h]$; the condition error is bounded by ε_A/h, where ε_A is a bound on the error in the computed function values at x, $x - h$ and $x + h$. If $f'''(\mu)$ is nonzero, the

interval that minimizes the sum of bounds on the truncation and condition errors is given by

$$h_C^* = \sqrt[3]{\frac{3\varepsilon_A}{|f'''(\mu)|}}.$$

An approximation to $f''(x)$ can be obtained from the *second-order difference formula*

$$\Phi(f, h) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{8}$$

The truncation error is bounded by $\frac{1}{12}h^2|f^{(4)}(\eta)|$, where $\eta \in [x-h, x+h]$, and the condition error arising from inaccuracies in the computed values of $f(x)$, $f(x+h)$ and $f(x-h)$ is bounded by $4\varepsilon_A/h^2$. If $f^{(4)}(\eta)$ is nonzero, the interval that minimizes the sum of these bounds is

$$h_\Phi^* = 2\sqrt[4]{\frac{3\varepsilon_A}{|f^{(4)}(\eta)|}}.$$

2.3. The well-scaled case. In some circumstances, it is possible to estimate a priori values of the optimal intervals for each of the finite-difference formulae described above. In particular, suppose that f is nonzero and that ε_A can be expressed in terms of a known bound ε_R on the *relative error*, i.e.,

$$\varepsilon_A = |f|\varepsilon_R. \tag{9}$$

(In this case, ε_R is a measure of the number of correct figures in f.) A relationship like (9) often holds for a value of ε_R close to machine precision when f is a standard function or has a very simple form.

If f and all its derivatives are of comparable magnitude for all points in $[x, x+h]$, then it follows from (6) that

$$h_F^* \sim \sqrt{\varepsilon_R},$$

where "$\sim$" means "of similar size". Furthermore, under these circumstances the bound on the *relative error* in the approximation of $f'(x)$ is also of order $\sqrt{\varepsilon_R}$. This result leads to the "folklore" observation that, in general, the number of correct figures in a forward-difference approximation with the *best* finite-difference interval is *half* the number of correct figures in f.

Under the same assumption about f and its derivatives, it holds that

$$h_C^* \sim \sqrt[3]{\varepsilon_R} \quad \text{and} \quad h_\Phi^* \sim \sqrt[4]{\varepsilon_R}.$$

3. A procedure for automatic estimation of a forward-difference interval. In this section we outline an automatic procedure whose purpose is to obtain a "good" finite-difference interval h_F for use in a forward-difference approximation to the gradient at a given point. The use of this procedure within a minimization algorithm is described in § 4. For simplicity, we shall describe the procedure as applied to the univariate function f; for an n-variable function, the procedure is executed once for each component of the gradient.

3.1. Motivation for the procedure. The interval h_F computed by the procedure is given by

$$h_F = 2\sqrt{\frac{\varepsilon_A}{|\Phi|}}, \tag{10}$$

which is simply expression (6) for the interval h_F^* with Φ substituted for $f''(\xi)$. Since the point ξ is unknown and the second derivative is not available, the value Φ used in the procedure is an estimate of $f''(x)$ obtained from the second-order difference approximation (8).

Two assumptions underlie this choice of Φ. Firstly, it is necessary for $f''(x)$ to be an adequate approximation of $f''(\xi)$, which will be true only if the second derivative is not changing too rapidly near x. Secondly, the value of Φ must be a sufficiently accurate estimate of $f''(x)$. The procedure is directed toward finding an interval h_Φ whose associated value of Φ is a correct order-of-magnitude estimate of $f''(x)$ (i.e., has at least one correct decimal figure).

The procedure is based on the fact that the bound on the relative truncation error tends to be an increasing function of h, while the relative condition error bound is generally a decreasing function of h. In particular, the value of h_Φ is selected from a sequence of trial values $\{h_i\}$. The decision as to whether a given value of Φ is sufficiently accurate involves $\hat{C}(\Phi)$, the following bound on the *relative* condition error in Φ:

$$\hat{C}(\Phi) \equiv \frac{4\varepsilon_A}{h^2|\Phi|}. \tag{11}$$

(When Φ is zero, $\hat{C}(\Phi)$ is taken as an arbitrarily large number.) No attempt is made to compute an explicit estimate of the truncation error.

If the value of $\hat{C}(\Phi)$ for the trial value h_i is "acceptable" (i.e., lies in the interval $[.001, .1]$), h_Φ is taken as h_i, and the current value of Φ is used to compute h_F from (10). There is a clear need for an upper bound on the value of $\hat{C}(\Phi)$. A *lower bound* on the acceptable value of $\hat{C}(\Phi)$ is needed because a very small value of $\hat{C}(\Phi)$ indicates that the interval may be reduced (thereby reducing the truncation-error bound) and still yield a sufficiently small value of the bound on the relative condition error.

If $\hat{C}(\Phi)$ is unacceptable, the next trial interval is chosen so that the relative condition error bound will either decrease or increase, as required. If the bound on the relative condition error is "too large", a *larger* interval is used as the next trial value in an attempt to reduce the condition error bound. On the other hand, if the relative condition error bound is "too small", h_i is reduced. We have chosen simply to multiply h_i by a fixed factor to obtain the next trial value. More complicated iterative schemes can be devised, based on the special form of (11), but they tend to require further assumptions that do not necessarily hold in practice (e.g., that Φ remains of a similar magnitude as h changes).

The procedure will fail to produce an acceptable value of $\hat{C}(\Phi)$ in two situations. Firstly, if Φ is extremely small, then $\hat{C}(\Phi)$ may never become small, even for a very large value of the interval. This will happen if f is nearly constant, linear or an odd function. Alternatively, $\hat{C}(\Phi)$ may never exceed .001, even for a very small value of the interval. This implies that Φ is extremely large and usually occurs near a singularity. These situations will be illustrated in the examples of § 3.3.

As a check on the validity of the estimated derivative, the procedure provides a comparison of the forward-difference approximation computed with h_F and the central-difference approximation computed with h_Φ. If these values do not display some agreement, neither can be considered reliable.

In order to begin the procedure, an initial trial value h_0 is needed. Since the "ideal" h_0 is unknown, our aim in selecting h_0 is for the minimum number of function evaluations to be required for well-behaved functions (i.e., for the value of $\hat{C}(\Phi)$ computed with h_0 to be acceptable). Hence, our choice of h_0 is based on a relationship

that holds for many functions encountered in practice—namely that

$$|f''| \sim \frac{1+|f(x)|}{1+|x|^2}, \tag{12}$$

where $\sim$ means "of a similar order of magnitude". If $|x|$ is of order unity, (12) holds when f and its second derivative are of comparable size (a characteristic of many well-scaled functions). The additional factor involving $|x|$ is included to reflect to some extent the effect of a simple change in scale of x on the second derivative.

If f and x are such that (12) holds, the interval $\bar{h} = 2(1+|x|)\sqrt{\varepsilon_A/(1+|f|)}$ produces a condition error in φ_F comparable to its "optimal" value, but a relative condition error $\hat{C}(\Phi)$ of order one (i.e., the value of Φ computed with $\bar{h}$ would have *no* correct figures). Therefore, the initial h_0 is taken as $10\bar{h}$, in the hope that h_0 will produce an acceptable value (of order .01) for $\hat{C}(\Phi)$ (recall that $\hat{C}(\Phi)$ is inversely proportional to h^2). In this case, no further trial intervals need to be computed.

The value of h_0 is not critical to the ultimate success of the algorithm, but clearly may affect the number of trial intervals required before termination. Given suitable a priori information, it may be possible to choose h_0 so that fewer steps of the algorithm are needed for a particular class of functions.

3.2. Statement of the algorithm. Algorithm FD requires an initial point x, the value of $f(x)$ and the value of ε_A. The algorithm computes an interval h_F that should produce an adequate forward-difference approximation of f'. The algorithm also produces φ (an estimate of $f'(x)$), Φ (an estimate of $f''(x)$), and E_F (an estimate of the error bound in φ).

The positive integer K is an upper bound on the number of trial intervals. In situations where the trial interval is increased the maximum number of times, it is useful to store h_s, the smallest value (if any) of h_i for which the relative condition errors in the forward- and backward-difference approximations are acceptable. For this reason, we define the following bounds on the *relative* condition errors in φ_F and φ_B:

$$\hat{C}(\varphi_F) \equiv \frac{2\varepsilon_A}{h|\varphi_F|} \quad \text{and} \quad \hat{C}(\varphi_B) \equiv \frac{2\varepsilon_A}{h|\varphi_B|}.$$

(When φ_F or φ_B is zero, the corresponding error bound is taken as an arbitrarily large number.) The value h_s is the smallest interval in the sequence for which these bounds are"acceptable" (i.e., less than .1).

The formal statement of the algorithm is:

ALGORITHM FD (*Automatic estimation of h_F, f' and f'' using finite differences*).

FD1. [Initialization.] Choose K and evaluate $f(x)$. Define $h_0 = 10\bar{h}$, where

$$\bar{h} = 2(1+|x|)\sqrt{\frac{\varepsilon_A}{1+|f(x)|}}, \tag{13}$$

and set $k \leftarrow 0$. Evaluate $f(x+h_0)$, $f(x-h_0)$, $\varphi_F(h_0)$, $\varphi_B(h_0)$, $\varphi_C(h_0)$, $\Phi(h_0)$, $\hat{C}(\varphi_F)$, $\hat{C}(\varphi_B)$ and $\hat{C}(\Phi)$. Set $h_s \leftarrow -1$.

FD2. [Decide whether to accept the initial interval.] If $\max\{\hat{C}(\varphi_F), \hat{C}(\varphi_B)\} \leqq .1$, set $h_s \leftarrow h_0$. If $.001 \leqq \hat{C}(\Phi) \leqq .1$, set $h_\Phi \leftarrow h_0$ and go to Step FD5. If $\hat{C}(\Phi) < .001$, go to Step FD4. Otherwise, continue at Step FD3.

FD3. [Increase h.] Set $k \leftarrow k+1$ and $h_k \leftarrow 10h_{k-1}$. Compute the associated finite-difference estimates and their relative condition errors. If $h_s < 0$ and

$\max\{\hat{C}(\varphi_F), \hat{C}(\varphi_B)\} \leqq .1$, set $h_s \leftarrow h_k$. If $\hat{C}(\Phi) \leqq .1$, set $h_\Phi \leftarrow h_k$ and go to Step FD5. If $k = K$, go to Step FD6. Otherwise, repeat Step FD3.

FD4. [Decrease h.] Set $k \leftarrow k+1$ and $h_k \leftarrow h_{k-1}/10$. Compute the associated finite-difference estimates and their relative condition errors. If $\hat{C}(\Phi) > .1$, set $h_\Phi \leftarrow h_{k-1}$ and go to step FD5. If $\max\{\hat{C}(\varphi_F), \hat{C}(\varphi_B)\} \leqq .1$, set $h_s \leftarrow h_k$. If $.001 \leqq \hat{C}(\Phi) \leqq .1$, set $h_\Phi \leftarrow h_k$ and go to step FD5. If $k = K$, go to Step FD6. Otherwise, repeat Step FD4.

FD5. [Compute the estimate of the optimal interval.] Define h_F from (10), and set φ to $\varphi_F(h_F)$. Set the estimated error bound to

$$E_F = \frac{h_F|\Phi|}{2} + \frac{2\varepsilon_A}{h_F}. \tag{14}$$

Compute the difference between φ and $\varphi_C(h_\Phi)$ as

$$\bar{E} = |\varphi - \varphi_C(h_\Phi)|.$$

If $\max\{E_F, \bar{E}\} \leqq .5|\varphi|$, terminate successfully. Otherwise, terminate with an error condition.

FD6. [Check unsatisfactory cases.] If $h_s < 0$ (i.e., $\max\{\hat{C}(\varphi_F), \hat{C}(\varphi_B)\} > .1$ for all values of h_k), then f appears to be nearly constant; set $h_F \leftarrow \bar{h}$ (13), and set φ, Φ and E_F to zero. Otherwise, if $\hat{C}(\Phi) > .1$, then f appears to be odd or nearly linear; set h_F to h_s, set φ to $\varphi_F(h_F)$, set Φ to zero and compute E_F from (14). Otherwise, f'' appears to be increasing rapidly as h decreases (since $\hat{C}(\Phi) < .001$ for all values of h_k); set $h_F \leftarrow h_K$, set φ to $\varphi_F(h_F)$, set Φ to $\Phi(h_F)$ and compute E_F from (14). In all these cases, terminate with an error condition.

3.3. Numerical examples. In all the examples, K was taken as 6, and the value of ε_A was computed using the procedure given in Gill, Murray and Wright (1981).

When Algorithm FD is applied to the function of Example 1 at the point $x = 1$, with $\varepsilon_A = 4 \times 10^{-6}$, the relative condition error with the interval h_0 (3.98×10^{-2}) is $\hat{C}(\Phi) = 4.16 \times 10^{-4}$, which is less than the desired lower bound; the value of $\hat{C}(\Phi)$ corresponding to h_1 is 4.24×10^{-2} and thus $h_\Phi = h_1$. The value of Φ is 2.38294×10^1, which is a good order-of-magnitude estimate of the exact value of $f''(x)$ (2.42661×10^1). The value of h_F is 8.19×10^{-4}, and $\varphi_F(h_F) = 9.55636$, giving a relative error in f' of 8.07×10^{-4}. Note that the estimate h_F agrees with the results given in Table 1 and that the estimates of f' and f'' are as accurate as we might expect from six-decimal arithmetic.

Example 2. Consider the function

$$f(x) = (x-100)^2 + 10^{-6}(x-300)^3$$

at the point $x = 0$, with $\varepsilon_A = 9 \times 10^{-3}$. The exact values of $f(x)$ and $f''(x)$ are 9.97300×10^3 and 1.99820, so that condition (12) is not satisfied. Since h_0 is too small, the interval is increased twice before the relative condition error is acceptable (the final value of Φ is 1.99567). The value of h_F is 1.34×10^{-1}; the corresponding φ_F is -1.99632×10^2 and the exact value of $f'(x)$ is -1.99730×10^2, which gives a relative accuracy of 4.9×10^{-4}.

Example 3. To illustrate the performance of the algorithm in the opposite situation when $|f''| \gg |f|$, consider the function $f(x) = e^{100x}$ at the point $x = .01$, with $\varepsilon_A = 4 \times 10^{-6}$. Since f'' is four orders of magnitude larger than f, h_0 is much too large and the interval is decreased twice before the value of $\hat{C}(\Phi)$ is satisfactory; the final value of Φ was 2.71141×10^4, which is a good order-of-magnitude estimate of the

exact value of $f''(x)$ (2.71828×10^4). The value of h_F is 2.43×10^{-5}; the corresponding value of φ_F is 2.72090×10^2, which gives a relative error of 1.01×10^{-3} in $f'(x)$. The exact value of $f'(x)$ is 2.71828×10^2.

The remaining examples illustrate situations in which an abnormal exit is made from the procedure.

Example 4. As an example of a function that is nearly constant, we chose $f(x) = 1 - \cos^2(x) - \sin^2(x)$, at the point $x = 1$, with $\varepsilon_A = 10^{-6}$. Although this function would be identically zero with exact arithmetic, its computed value fluctuates around the level of machine precision. The relative condition error never becomes acceptably small, even after the interval is increased the maximum number of times. The algorithm terminates in Step FD6 with h_F set to $\bar{h}$ (4.0×10^{-3}), the interval associated with a well-scaled problem.

Example 5. The algorithm was tested on the linear function $f(x) = 3x + 1$ at the point $x = 1$, with $\varepsilon_A = 5 \times 10^{-6}$. As in Example 4, the relative condition error in Φ was not acceptable even when the interval was increased to its maximum value. The value of h_F was set to h_s, the smallest value for which $\max\{\hat{C}(\varphi_F), \hat{C}(\varphi_B)\}$ was less than .1 (in this case, $h_s = 4.0 \times 10^{-2}$). (The behavior of the algorithm was the same on the odd function $f(x) = x^3$ at $x = 0$, except that $h_s = 2.0 \times 10^{-1}$.)

Example 6. The remaining abnormal termination of the algorithm in Step FD6 occurs when the condition error in Φ remains too small even when h is decreased the maximum number of times. This situation usually occurs near a singularity and is illustrated by the function $\ln|x|$ at the point $x = 10^{-8}$, with $\varepsilon_A = 2.0 \times 10^{-5}$. The estimate of Φ increases by two orders of magnitude for every order-of-magnitude decrease in h, so that $\hat{C}(\Phi)$ consistently remains of order 10^{-6}, even when h is of order 10^{-7}. The procedure terminates with $h_F = h_6 = 2.03 \times 10^{-7}$.

Example 7. It was observed in § 2.1 that a forward-difference approximation will tend to have poor relative accuracy when $|f'|$ is small, even with the best choice of interval. As an example of this situation, consider the function $f(x) = x^4 + 3x^2 - 10x$ at the point $x = .99999$, with $\varepsilon_A = 7 \times 10^{-6}$. The exact value of $f'(x)$ is -1.80244×10^{-4}. The algorithm computes a good estimate of Φ with h_0 ($\Phi = 1.80008 \times 10^1$ compared to the exact value $f'' = 1.79999 \times 10^1$) and h_F is given by 1.25×10^{-3}. However, $\varphi_F(h_F)$ is 9.13×10^{-3}, so that the approximate derivative has poor relative accuracy, even though h_F is a good estimate of the optimal forward-difference interval. The central-difference approximation computed with h_Φ (-3.57×10^{-3}) also has poor relative accuracy. Note that the difference between the forward- and central-difference estimates would cause the algorithm to give an error message.

Example 8. Algorithm FD may fail to produce an acceptable estimate of the derivative even when f' is not small. The value of h_F is based on approximating $f''(\xi)$ in (6) by Φ, which is itself an approximation to $f''(x)$. Consequently, h_F will be a poor estimate of the optimal interval when $f''(\xi)$ in (6) differs widely from Φ—for example, if f''' is very large near x. In this case, the derivative approximation will be inaccurate because the underlying asssumptions used to derive (6) are not satisfied, even though the estimated condition error may be acceptable.

This situation often happens near a singularity, but may also occur in other situations. To illustrate the latter, consider the function

$$f(x) = 10000x^3 + .01x^2 + 5x$$

at the point $x = 10^{-9}$, with $\varepsilon_A = 10^{-6}$. The exact values are $f'(x) = 5.0$, $f''(x) = 2.006 \times 10^{-2}$ and $f'''(x) = 10000$. After one increase in the interval, we obtain the value $\Phi = 2.02177 \times 10^{-2}$ (a good estimate of $f''(x)$), and the resulting value of h_F is

1.41×10^{-2}. However, the value of $\varphi_F(h_F)$ is 6.98 and the value of $\varphi_C(h_\Phi)$ is 4.05×10^2; note that both estimates are poor. As in Example 7, the difference between these estimates produces an error message.

4. Gradient approximations in multivariate optimization. In this section we shall be concerned with the n-variable function $F(x)$, $x \in \mathcal{R}^n$. The gradient of F will be denoted by $g(x)$. Let x_0 denote the initial estimate of the solution.

4.1. Estimating the finite-difference interval at an arbitrary point. The procedure of § 3.2 requires at least three evaluations of the function for each component of the gradient, even for a well-scaled problem. An optimization algorithm that required this number of function evaluations at every iteration would be uncompetitive with alternative nonderivative methods. Ideally, an algorithm should be able to compute an approximation to the gradient with only one function evaluation per component.

Fortunately, this aim can often be achieved, for several reasons. First, and most important, it is our experience that, for many functions encountered in practice, the finite-difference intervals generated by a procedure such as that of § 3.2 do not vary significantly from one iteration to the next. Second, the intervals produced by the procedure of § 3.2 do not usually differ widely from the "optimal" intervals. Finally, finite-difference gradient methods can generally make satisfactory progress as long as the overall gradient vector has a "reasonable" level of accuracy; it is not essential for each component of the gradient to have close-to-maximal accuracy at every iterate.

Based on these observations, we suggest that the procedure of § 3.2 should be executed at a "typical" point (usually, x_0). The set of intervals obtained should then be used to compute forward-difference approximations at subsequent iterates.

We illustrate the effects of this strategy with the function from Example 1 when $x_0 = 1$. In § 3.3, it was shown that Algorithm FD produces the interval $h_F = 8.19 \times 10^{-4}$. When this interval is used to compute a forward-difference approximation at the point $x = 10$, we obtain a relative accuracy of 4.09×10^{-4} in the derivative (the approximate and exact values are 9.71×10^8 and 9.70286×10^8). When Algorithm FD is executed at the point $x_0 = 10$ (with $\varepsilon_A = 5.0 \times 10^2$), the value of h_F is 1.01×10^{-3}; if this interval is used to compute φ_F at $x = 1$, the relative error in the forward-difference approximation is 1.02×10^{-3}. *These results are typical of those in many numerical experiments.* Although it is not difficult to construct examples for which the optimal finite-difference intervals vary significantly from point to point, our experience is that the procedure works well in practice.

An additional benefit of the procedure of § 3.2 is that the quantity Φ associated with the ith variable is usually a good estimate of the ith diagonal element of the Hessian matrix at x_0, and thus we can obtain an initial diagonal approximation of the Hessian in a quasi-Newton method. An improved estimate of the Hessian often reduces the number of iterations required for convergence.

4.2. Other approaches. Several other approaches have been suggested for obtaining finite-difference approximations to the gradient. The procedure of Curtis and Reid (1974) is based on a central-difference approximation and hence requires at least $2n$ evaluations of the function at each iterate in order to approximate the gradient.

Stewart (1967) suggested a procedure for using a forward-difference approximation in a finite-difference quasi-Newton method and for switching to central differences under certain conditions. At each iteration, the interval with respect to each variable is chosen to minimize the sum of the relative truncation error and the relative condition error. The relative truncation error in the ith component of the forward-difference

approximation computed with interval h_i is estimated by

$$\frac{1}{2}\left|\frac{\Psi h_i}{\gamma}\right|,$$

where Ψ is the ith diagonal element of the current quasi-Newton approximation to the Hessian and γ is the ith component of the approximate gradient from the *previous* iteration. An assumption is made that the calculated value $\hat{F}$ of the exact function F satisfies

$$\hat{F}(x) = F(x)(1+\varepsilon),$$

such that $|\varepsilon| \leqq \varepsilon_R$, where ε_R is a known quantity; the relative condition error in the ith component of the approximate gradient is then given by

$$\frac{2\varepsilon_R|F(x)|}{|F(x+h_i e_i)-F(x)|}.$$

The reader should consult Stewart's original paper for a more detailed discussion of his assumptions and for a statement of the criteria used to switch to a central-difference approximation.

Our approach differs in at least two significant respects. First, our analysis is based on different estimates of the truncation and condition errors. Second, we suggest that additional function evaluations should be expended at the initial point in order to obtain accurate estimates of the second partial derivatives.

A modified version of Stewart's method could be developed in which the diagonal elements of a quasi-Newton approximation are used to estimate $f''(\xi)$ in (6). (Note that no additional evaluations of the function would be necessary.) With this strategy, the ith interval at the general point x would be given by

$$h_i = 2\sqrt{\frac{\varepsilon_A}{|\Psi|}}.$$

The unsatisfactory feature of this approach, as with Stewart's original procedure, is that the diagonal elements of a quasi-Newton approximation need not be good order-of-magnitude estimates of the diagonal elements of the true Hessian (see, e.g., Dennis and Moré (1977)). Even for a quadratic function, the Hessian approximation is exact only after n iterations and then only with exact arithmetic. This discrepancy is particularly severe for badly scaled problems, in which accurate gradient approximations are most crucial.

4.3. The switch to a central-difference approximation. The forward-difference formula requires only one additional evaluation of F for each component of the gradient and will usually provide approximate gradients of acceptable accuracy unless $\|g(x)\|$ is small. Since the gradient approaches zero at the solution of an unconstrained problem, this means that *the forward-difference approximation will eventually be unreliable*, even for a well-scaled problem with carefully chosen finite-difference intervals (see § 2.1 and Example 7).

The forward-difference approximation should be abandoned when there is a failure to find a lower point during the linear search. This tends to happen only when the relative error in the overall gradient vector becomes too large (note that the algorithm will not usually encounter difficulties if the relative error is poor only in some components). It may also be advisable to avoid the use of the forward-difference

formula under other circumstances—for example, if the change in x during any iteration is close to the perturbation associated with a finite-difference calculation.

When the forward-difference formula is unacceptable, we suggest a switch to a central-difference approximation (see § 2.2). The condition error in the central-difference formula is of order $(1/h)$, but the bound on the central-difference truncation error is of higher order than that for the forward-difference formula. When switching to central differences, we suggest using the set of intervals $\{(h_\Phi)_i\}$ computed at the initial point; note that, if Algorithm FD terminates successfully, $(h_\Phi)_i > (h_F)_i$.

5. Conclusions. We have described an algorithm for estimating a set of finite-difference intervals at a given point. In practice, the information accumulated about the function of interest at the given point usually provides an adequate set of finite-difference intervals at other points. If the intervals become inappropriate (for example, they may fail to produce a useful search direction in an optimization algorithm), a new set of intervals can be computed at the point where failure occurred.

The implementor of a finite-difference gradient method faces a dilemma: although good finite-difference intervals are often crucial to success (especially for badly scaled problems), an efficient method must not expend a large number of function evaluations in obtaining these intervals. The procedures in this paper are based on a compromise that generally produces sufficiently accurate derivatives at a reasonable cost.

Acknowledgment. We are indebted to James Lyness for many valuable comments.

REFERENCES

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

A. R. CURTIS AND J. K. REID (1974), *The choice of step lengths when using differences to approximate Jacobian matrices*, J. Inst. Maths. Applics. 13, pp. 121–126.

G. DAHLQUIST AND Å. BJÖRK (1974), *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ.

J. E. DENNIS JR. AND J. J. MORÉ (1977), *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19, pp. 46–89.

J. DUMONTET AND J. VIGNES (1977), *Determination du pas optimal dans le calcul des dérivées sur ordinateur*, Revue française d'automatique, d'information et de recherche opérationelle, Analyse numérique (RAIRO), 11, pp. 13–25.

P. E. GILL, W. MURRAY AND M. H. WRIGHT (1981), *Practical Optimization*, Academic Press, London and New York.

R. W. HAMMING (1973), *Numerical Methods for Scientists and Engineers*, 2nd edition, McGraw-Hill, New York.

W. KAHAN (1973), *The implementation of algorithms: Part* 1, Technical Report 20, Dept. Computer Science, Univ. of California, Berkeley.

J. N. LYNESS (1977), *Has numerical differentiation a future?* in Proc. 7th Manitoba Conference on Numerical Mathematics, pp. 107–129.

J. N. LYNESS AND C. B. MOLER (1967), *Numerical differentiation of analytic functions*, SIAM J. Numer. Anal., 4, pp. 202–210.

J. N. LYNESS AND G. SANDE (1971), ENTCAF *and* ENTCRE: *Evaluation of normalized Taylor coefficients of an analytic function*, Comm. ACM, 14, pp. 669–675.

J. OLIVER (1980), *An algorithm for numerical differentiation of a function of one real variable*, J. Comp. Appl. Math., 6, pp. 145–160.

J. OLIVER AND A. RUFFHEAD (1975), *The selection of interpolation points in numerical differentiation*, Nordisk Tidskr. Informationsbehandling (BIT), 15, pp. 283–295.

R. S. STEPLEMAN AND N. D. WINARSKY (1979), *Adaptive numerical differentiation*, Math. Comput. 33, pp. 1257–1264.

G. W. STEWART (1967), *A modification of Davidon's method to accept difference approximations of derivatives*, J. Assoc. Comput. Mach., 14, pp. 72–83.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0016 $01.25/0

COMPUTATIONAL ASPECTS OF OPTIMAL ALLOCATION IN MULTIVARIATE STRATIFIED SAMPLING*

BEVERLEY D. CAUSEY†

Abstract. We consider the problem of extending the Neyman allocation—which minimizes total sampling cost in stratified random sampling subject to an upper bound on the variance of a single estimated population total—the same situation with more than just one variance constraint. We deal with a series of computational aspects of solving this problem.

Key words. stratified random sampling, Neyman allocation, multiple variance constraints, convex programming, Newton approximation

1. Introduction. Suppose that we want to estimate total personal income in New York City. We have an enumeration of the N residents of the city; based on this enumeration we draw a simple random sample of size n from these N and compute the mean income for these n persons. Let $\bar{y}$ denote this mean income; then $N\bar{y}$, which we call $\hat{Y}$, is an unbiased estimator of total income.

For $i = 1, \cdots, N$ let Y_i denote the income for resident i; let $\bar{Y}$ denote $(\sum_{i=1}^{N} Y_I)/N$, the mean income for the N residents. Let S^2 denote the variance of income in the city; this variance is $(\sum_{i=1}^{N} (Y_i - \bar{Y})^2)/(N-1)$. Thus $\bar{Y}$ and S^2 are the true (but unknown) population mean and variance for all N residents—to be distinguished from (results based on) our sample of n residents. We let simply S denote the square root of S^2—that is, the standard deviation of income.

The variance of the estimator $\hat{Y}$, which we call Var $(\hat{Y})$, is equal to $NS^2(N-n)/n$. Hence if we have an advance idea of the value of S^2, we have an advance idea of the value of Var $(\hat{Y})$.

However, general reasoning suggests that we may reduce Var $(\hat{Y})$ as follows. Let the city be divided into rich and poor neighborhoods. We estimate the mean income for each neighborhood; then we form $\hat{Y}$ based on putting together these mean incomes to estimate the entire-city income. The variance of income within neighborhoods is much smaller than that between neighborhoods and thus is smaller than S^2; as a result, Var $(\hat{Y})$ will be less than for simple random sampling. Explicitly, we let N_h, n_h, $\bar{y}_h$ and S_h denote N, n, $\bar{y}$ and S defined for neighborhood h, i.e., "stratum" h, unto itself. Then $\hat{Y} = \sum N_h \bar{y}_h$ and Var $(\hat{Y})$ is $\sum N_h S_h^2 (N_h/n_h - 1)$. From this point, we proceed to the problem of this paper.

Suppose that we want to draw simple random samples from each of H given strata numbered $h = 1, \cdots, H$. Let:

Y be a variate of interest defined for each population unit;

N_h be the number of population units in stratum h;

$n_h (\leqq N_h)$ be the number of sample units to be drawn from stratum h;

(1.1) S_h^2 be the variance of Y in stratum h;

c_h be the cost per unit sampled in stratum h (often, one just takes $c_h = 1$);

$\bar{y}_h$ be the sample mean for stratum h; and

$\hat{Y} = \sum N_h \bar{y}_h$, an unbiased estimator of the population total for the variate Y.

* Received by the editors May 31, 1980, and in revised form February 15, 1982.
† Statistical Research Division, Bureau of the Census, Washington, DC 30233.

We want to choose the $n_h (\leqq N_h)$ so as to

$$\text{minimize } \sum c_h n_h \text{ subject to Var } (\hat{Y}) \leqq V \tag{1.2}$$

for $V(>0)$ specified by the analyst. We have $\text{Var}\,(\hat{Y}) = \sum N_h S_h^2 (N_h/n_h - 1)$. For $h = 1, \cdots, H$ let:

$$\begin{aligned} &x_h = 1/c_h n_h; \\ &\text{vector } X = (x_1, \cdots, x_H); \\ &f(X) = \sum (1/x_h); \\ &B_h = c_h N_h^2 S_h^2 / (V + \sum N_j S_j^2) \text{ (denominator independent of } h); \\ &\text{and } g(X) = 1 - \sum B_h x_h; \end{aligned} \tag{1.3}$$

the problem becomes that of

$$\text{minimizing } f(X) \text{ subject to } (X) \geqq 0. \tag{1.4}$$

Thus our problem may be mathematically characterized in terms of the parameters B_h, $h = 1, \cdots, H$, which combine information in N_h, c_h, S_h and V. Using a Lagrange multiplier, we obtain as a solution the familiar Neyman allocation (Cochran (1963)): $x_h = (B_h^{1/2} \sum_j B_j^{1/2})^{-1}$. There are also the constraints $n_h \leqq N_h$, i.e., $p_h \geqq 0$, where $p_h = x_h - 1/c_h N_h$; these constraints must be dealt with, but we do not do so until § 7.

Suppose now that we have, instead of just a single y, K variates of interest (with $K > 1$) Y_k, $k = 1, \cdots, K$, with associated constraints $\text{Var}\,(\hat{Y}_k) \leqq V_k$. Correspondingly, we have K constraints $g_k \geqq 0$, where

$$g_k(X) = 1 - \sum_h B_{kh} x_h, \qquad k = 1, \cdots, K. \tag{1.5}$$

In other words, for each constraint k, $k = 1, \cdots, K$, we define $g(X)$ and quantities B_h in the above fashion: we then denote these by $g_k(X)$ and B_{kh}. To insure that $n_h > 0$, so that x_h is well defined, we remove from consideration any strata for which, trivially, $B_{kh} = 0$ for all k (so that $n_h = 0$).

This, then, is the problem that we address. In practice, of course, the stratum variances "S_h^2" are determined as accurately as possible before drawing the sample from previous information. The solution vector X will in general be unique; there is no ambiguity concerning what is the correct answer. (We do not address the problem of how best to round the resultant n_h's to integers: clearly, if the n_h's are large, this problem is minor.) The uncertainty is over how best, computationally, to go about finding this solution.

In this paper we describe a series of steps that we have taken in obtaining a solution. We often point out advantages over familiar, existing techniques. Yet to a large extent each of our recommendations may be viewed separately. Depending on a particular problem or situation—or even on one's personal prejudices—one may accept or reject each of our proposals on an individual basis. Section by section we:

(1) For $K = 2$, in § 2, give a speedy algorithm based on an approach developed by Cochran (1963, § 5A.4). (For $K = 1$, of course, we have the Neyman allocation.)

(2) Indicate, in § 4, how, instead of dealing with a problem in H variables as considered in § 3, we may deal with a problem in only K variables—an approach advantageous for $K < H$—as has been noted, equivalently, by Al-Khayyal et al. (1978). Independently of our paper, these authors developed, for a somewhat different purpose, several of the ideas appearing herein.

(3) Show, in § 5, how, for $K \geqq 3$, instead of using a convex-programming algorithm to solve the K-variable problem, we may make efficient use of generalized Newton approximation.

(4) Show, in § 6, how to solve this K-constraint problem by adding one constraint (in many cases, several constraints) at a time to a subset of the full set of K, until the solution for a subset of, say, J constraints satisfies the remaining $K-J$.

(5) Deal, in § 7, with the constraints $n_h \leqq N_h$ according to Al-Khayyal et al. (1978); if necessary, we repeatedly do (3) and (4) here.

(6) Show, in § 8, how to measure, for $J \geqq 3$, closeness of cost for the solution at hand to the cost at the unattained optimum.

(7) Give, in § 9, an example to illustrate how, usefully, most of the quantities B_{kh} can be artificially 0 (with $K \ll H$, also).

(8) Consider, in § 10, some possible nonlinear cost functions (in contrast to $\sum c_h n_h$ of (1.2)).

Relative to other methods, our approach clearly becomes advantageous when J is very small relative to H and K and also when there are many stratum variances artificially zero. Our methods are not better for every imaginable situation, but seem to work well in practice for large problems. We have dealt with business surveys on monthly retail, wholesale and "service" sales, a voter registration survey and planned "post-enumeration" followup to the 1980 Census. We have developed computer programs which are available, one especially useful when most or all stratum variances are nonzero and one when most stratum variances are zero.

Other aspects of the general situation that we consider are discussed by Chatterjee (1968) and Folks and Antle (1965). Here we focus on computational issues for a specific problem—without considering surrounding issues such as, for example, how strata might best be formed.

2. Two constraints. For $K = 2$ the essence of a convenient solution is given by Cochran (1963, § 5A.4). By use of Lagrange multipliers it is established there (eq. (5A.14)) that the solution is of form which, for our purposes, may be viewed as $(u_1 B_{1h} + u_2 B_{2h})^{-1/2}$ for some $u_1, u_2 > 0$. Because of the approach that we take in § 6, we will know that, for our purposes, whenever $K = 2$: (1) we will never be in a situation where the Neyman allocation for either one of the two constraints can satisfy the other; and (2) we must have either $B_{1h} > 0$ or $B_{2h} > 0$. As a result, we know that $u_k > 0$ and $g_k = 0$, $k = 1, 2$, at the solution. From this point we use a binary search to obtain a solution which Cochran (1963) obtains by approximation in his equations (5A.15)–(17.)

The following is a concise statement of the algorithm that we thus use for $K = 2$. Re-express x_h as $r(tB_{1h} + (1-t)B_{2h})^{-1/2}$ for $r > 0$ and $0 < t < 1$; now we find the values of t and r. Let $D_h = B_{2h} - B_{1h}$; since $0 = g_2 - g_1 = \sum D_h x_h$, we find t by solving $C(t) = 0$, with $C(t) = \sum D_h / (B_{2h} - tD_h)^{1/2}$. With $dC/dt = \sum D_h^2 / 2(B_{2h} - tD_h)^{3/2} > 0$, we may narrow in on the solution t with a binary search. We compute $C(\frac{1}{2})$; if $C(\frac{1}{2}) > (<) 0$, we subtract (add) $\frac{1}{4}$ to t, i.e., use $\frac{1}{4}$ ($\frac{3}{4}$); likewise we subtract or add $\frac{1}{8}, \frac{1}{16}, \cdots, \frac{1}{2}^{16}$ (the last exponent being somewhat arbitrary). Having found the solution t in this manner, we have $r = \sum B_{2h} / (tB_{1h} + (1-t)B_{2h})^{1/2}$. Thus the 2-constraint situation is easily solved as an equation involving a single, bounded variable (t).

3. Three or more constraints. For $K > 2$ one may find the solution by an approach such as that of Chatterjee (1966) or those of references in Cochran (1977, § 5A.4), based on making a series of quadratic or linear approximations; or one may use a convex-programming technique such as "SUMT" (Fiacco and McCormick (1968)).

We may incorporate the constraints $p_h \geqq 0$ in these techniques. Here a direct approach is based on the H variables x_h, $h = 1, \cdots, H$. In § 4 we now take an approach which deals, instead, with only K variables.

4. Constraints as variables. We have (Fiacco and McCormick (1968, § 2.3, Thm. 4)) that the solution (without the constraints $p_h \geqq 0$) is given by X and $U = (u_1, \cdots, u_K)$ such that $u_k \geqq 0$, $g_k \geqq 0$, $u_k g_k = 0$ and $(d/dx_h)(f - \sum u_k g_k) = 0$—this solution being unique if, as should hold because of our § 6 approach, the rank of the $K \times H$ matrix $((B_{kh}))$ is K. Using the last, we may obtain $x_h = (\sum_k u_k B_{kh})^{-1/2}$ and thus

$$g_k(X) = g_k(U) = 1 - \sum_h B_{kh}/t_h^{1/2}, \quad \text{where } t_h = \sum_i u_i B_{ih}. \tag{4.1}$$

Thus in terms of U our problem is that of

$$\text{minimizing } F = \sum u_k g_k(U) \quad \text{subject to } u_k \geqq 0 \quad \text{and} \quad g_k(U) \geqq 0, \tag{4.2}$$

i.e., forcing F to 0. It may easily be established that the g_k are concave in U (the u_k are, of course, linear) and that F is strictly convex if BB' is positive definite, so that (4.2) has a unique solution. (A somewhat different approach was used by Al-Khayyal et al. (1978) to formulate a problem equivalent to (4.2).)

5. Newton approximation. Rather than *minimize* F, which is best done (in a series of iterative steps) by using the first and *second* derivatives of it, i.e., of the $u_k g_k$'s, with respect to the u's while taking care to avoid violating the constraints of (4.2), we may solve the K *equations* $u_k g_k(U) = 0$ by using only the *first* derivatives. We do this by means of generalized Newton approximation, as follows. Choose

$$U_0 = (u_{01}, \cdots, u_{0K}) \quad \text{such that } u_{0k} \geqq 0 \quad \text{and} \quad g_k(U_0) \geqq 0; \tag{5.1}$$

we let u_{0k} be $(\sum_h B_{kh}^{1/2})^2$, corresponding to Neyman allocation for constraint k alone. We compute the corresponding g's and then multiply U_0 by the factor $(1.001 \min(1, (1 - \min g_k)/.99))^2$. This somewhat arbitrary rule shrinks U_0 to a solution still satisfying (5.1) yet giving smaller cost. We denote

$$u_k g_k(U) \quad \text{by } v_k \quad \text{and} \quad dv_i/du_j \quad \text{by } a_{ij}; \tag{5.2}$$

then $a_{ij} = \delta_{ij} g_i + \frac{1}{2} u_i \sum_h B_{ih} B_{jh}/t_h^{3/2}$. Let V denote $(v_1, \cdots, v_K)$ and D the vector $A^{-1}V$, with D_0 denoting D evaluated at U_0. The vector D is the solution to $AD = V$. Computationally it is convenient to divide a_{ij}, $j = 1, \cdots, K$, and v_i by $u_i/2$; this does not change the value of D. The altered A and V are thus given by

$$a_{ij} = \delta_{ij} 2g_i/u_i + \sum_h B_{ih} B_{jh}/t_h^{3/2} \quad \text{and} \quad v_i = 2g_i \tag{5.3}$$

with A symmetric positive definite.

An off-diagonal entry of A is (always) zero for row i, column j if constraints i and j have either $B_{ih} = 0$ or $B_{jh} = 0$ (or both) for each h. (As indicated in § 9, we will often have many such zeroes.) Thus we deal repeatedly with A having 0's in the same positions; based on this information we may arrange the indexing of the constraints so that for whatever method is used to find D, many additions and multiplications may be bypassed. Here we omit details of the particular way in which we did this. We used the Crout method (Hildebrand (1956)) for solving a system of simultaneous linear equations with symmetric coefficient matrix; and it seemed straightforward to develop an accompanying arrangement of indexes to keep number of calculations at

a minimum. For some other method of solving the system of equations, a different arrangement and strategy might be appropriate.

The generalized Newton approximation to the solution U is $U_1 = U_0 - D_0$. But if we "step" too far from U_0 and come close to violating the constraints of (5.1)—even if we do not actually violate them—we will probably be thrown off the track, being close to a boundary but not close to the optimal vector U which is usually on a boundary (somewhere else). Hence instead of $U_0 - D_0$ we use $U_0 - cD_0$, where scalar c is as follows. Let U_1, with components u_{1k}, $k = 1, \cdots, K$, denote $U_0 - cD_0$.

(1) Let $c = 1.01$ (i.e., close to 1).

(2) If necessary, reduce c further so that $u_{1k} \geqq 0$ for all k.

(3) Divide c by 1.01. (Thus if no reduction is necessary in (2) we have $c = 1$; in any event, all the values u_{1k} are strictly positive and all the values x_h are well defined.)

(4) If necessary, divide c by 2, as many times as necessary up to 6 times, so that $g_k(U_1) \geqq 0$ for all k. (Thus all constraints are satisfied.)

(5) Divide c by 10.

This gives the final c.

We continue in this manner, computing U_{j+1} from U_j except that for $j \geqq 20$ we divide c by 2 instead of 10 in Step 5. We stop when: (a) the ratio $(\sum u_k g_k)/(\sum u_k)$ has become less than .001; (b) j reaches 500; or (c) Step 4 was repeated 6 times without $g_k \geqq 0$ for all k. Virtually never will (b) and (c) be encountered, in which case any further progress would be slow and time consuming. The meaning of the cutoff .001 in (a) is discussed in § 8.

As an alternative to the above approach, one might use convex-programming techniques as in § 3, to solve the problem in (4.2). Such techniques can avoid the arbitrariness in the above determination of scalar c. Using SUMT, we would again repeatedly solve a $K \times K$ system of linear equations with A now based on second derivatives of the function $F - r(\sum \log u_k + \sum \log g_k)$ for fixed, small and positive r; logs might be replaced by reciprocals. This A may be seen to be more cumbersome than the A above. Moreover, all its entries will be nonzero; thus, if for many pairs (i, j) we have $B_{ih} = 0$ and/or $B_{jh} = 0$ for each h, the above-discussed shortcut is not available.

On the other hand, if (few or) no entries of A are zero for the Newton method, a convex-programming approach (even with its added annoyances) may be especially worthwhile: in other words, even though we have found the Newton method to perform well for large problems, it should not be viewed as always faster.

6. Constraint subsets. If the solution for a subset S of the K constraints, say J constraints with $J < K$, satisfies the remaining $K - J$ constraints, we have a solution to the full problem. This enhances the potential usefulness of the Neyman allocation (§ 1) corresponding to $J = 1$ and of § 2 corresponding to $J = 2$. Accordingly, we build up S as follows, at each stage adding the constraint which appears most likely to have to be included in S.

First ($J = 1$), include in S the constraint for which the cost based on Neyman allocation (for that constraint alone) is largest.

Second, without loss of generality suppose the first constraint picked was # 1. If there are no strata for which $B_{1h} = 0$, we let the second constraint be that which is farthest from being satisfied by the sample allocation for the first constraint and go to the third step. Farthest from being satisfied means that the (negative) value of g_k is the most strongly negative. If $B_{1h} = 0$ for some h, we use the following iterative rule. For $k \notin S$ let a_k be the portion of the Neyman allocation cost for k that is associated with strata for which $B_{jh} = 0$ for all $j \in S$ and add to S the choice of k which maximizes

a_k. When this maximum is 0, i.e., there are no strata for which $B_{jh} = 0$ for all $j \in S$, rather than add to S we go to the third step. (At this point we observe again that there are many instances where B's can, usefully, be 0; this is discussed in § 9.)

Third, for $J \geqq 2$ find the solution; then let the next constraint added to S be that which is farthest from being satisfied by this solution. Keep doing this until done. At any point we are done if the current solution satisfies *all* the constraints $g_k \geqq 0$ (including, obviously, the case $J = K$ where there are no more constraints to add).

When many B's are zero, we add a set of constraints, rather than a single constraint that is "farthest from being satisfied", before finding a new solution. The idea is to add at once any mutually nonoverlapping constraints that are going to have to be added anyway. After a solution is obtained, set $a_h = 0$ for each stratum h. (1) Add the constraint k that is farthest from being satisfied, subject to $a_h = 0$ whenever $B_{kh} > 0$. (2) Set $a_h = 1$ wherever $B_{kh} > 0$. (3) Repeat # 1 and # 2 (without resetting any a_h to 0) until no more constraints can be added.

The matrix A should be well-conditioned since, by eliminating redundant constraints, we prevent, for example, two rows of the underlying matrix $((B_{kh}))$ from being very nearly collinear. This approach should preclude the computational difficulties that would occur (in practice) if we simply started with the full set of K constraints: as an extreme example, if two constraints were identical, and both were included, we would be trying, in effect, to invert a nearly singular A.

Ordinarily we will have $J \leqq H$, i.e., the number of constraints included in S will not exceed H. Thus for $H = 2$ (and $K \gg 2$) it should be simpler to use our approach than the graphical approach for $H = 2$ developed by Dalenius (1957).

7. Sample size greater than the universe. Now we consider the H constraints $p_h \geqq 0$. The recommended procedure to satisfy these is as follows. First, do the problem according to §§ 5–6. If $p_h \geqq 0$ for all h, we are done. Otherwise, for strata where $p_h \geqq 0$, sample all N_h units; then redo the problem with these strata removed from consideration. If under the new allocation additional strata are over-allocated, repeat the process; continue in this manner until $p_h \geqq 0$ for all strata. This procedure provides a solution to the full problem, as indicated by Al-Khayyal et al. (1978).

A variation on this idea is as follows. Suppose that sampling fractions are very small, so that, for certain, we will have $n_h \leqq N_h$; but that we want $n_h \geqq a_h \geqq 0$, where a_h is fixed (and, nontrivially, not always 0). We let $p_h = 1/c_h a_h - x_h$ and go through the above procedure, setting $n_h = a_h$ and adjusting the problem accordingly whenever there is *under*-allocation. We do not attempt to deal with the situation where both over- and under-allocation are simultaneously possible.

8. Closeness of solution to optimum. For $J = 1$ or 2, we are essentially at the exact optimum solution. For $J \geqq 3$, however, we only approximate it. Thus we have developed a way of estimating the cost associated with the exact (unattained) optimum, to see how much further cost reduction could be gained if we could attain this exact optimum.

Let S denote $\sum u_k$, R denote $\sum u_k g_k$, C denote cost, $U = (u_1, \cdots, u_K)$ and $D = (d_1, \cdots, d_K)$ based on the final printed solution. It may easily be shown, algebraically, that $S - R = C$. Let S^*, R^*, etc., denote these quantities at the exact solution; since $R^* = 0$, we have $S^* = C^*$. In the notation of § 5, it is approximately true that U^* will lie along the line $U - cD$ with scalar $c > 0$. The rate of change of S with respect to change in R is given, approximately, by the ratio of directional derivatives with respect to c of S and R along this line, evaluated at $c = 0$. The derivative for S is $\sum d_k$, for

$R\frac{1}{2}\sum d_k(1+g_k)$. We may thus approximate that S^*, corresponding to $R^*=0$, i.e., R decreasing to 0, is given by $S-2R(\sum d_k)/(\sum d_k(1+g_k))$.

Thus C^*, equal to S^*, is approximated by this quantity. When the g_k's are all close to 0, this is close to $S-2R$, i.e., $C-R$; thus the ratio C/C^* is approximately $1+R/C$, i.e., $1+R/S$. Our usual stopping rule as indicated in § 5 is $R/S<.001$; thus, typically, C will be greater than C^* by no more than 1 part in 1,000, and even less, possibly, if some g_k's are far from 0. (Yet if g_k is far from 0, then u_k and thus d_k, are close to 0; as a result, the difference should not be *much* less than 1 part in 1,000.) In other words, when we reach (a) at the end of § 5's Newton method, cost for the exact optimum differs by somewhat less than 1 part in 1,000 from that for the solution in hand.

These results may be generalized, of course, to bounds other than .001.

9. Zero variances and artificial variables. The following example, closely resembling an actual situation for the Census Bureau's monthly retail trade survey, is just one illustration of (1) how zero variances may arise and (2) the usefulness of artificial variables, in the following sense.

Suppose that we have a two-way stratification scheme with strata denoted, instead of by h, the pair of integers (i, j), $i=1,\cdots,25$, and $j=1,\cdots,20$. For a single variable of interest we may obtain an unbiased estimator of the population total in each region i, $i=1,\cdots,25$, and likewise in each industry j. Corresponding to region i we define a variable which artificially is 0, and thus has 0 variance, for strata (i', j) with $i'\neq i$—likewise industry j—and impose upper bounds on the variances of the corresponding estimated totals. Thus we have $H=25\times 20=500$ and $K=25+20=45$, so that $K\ll H$; also, out of the HK (22,500) stratum-constraint combinations we have nonzero variances for only $2H$ (1,000), a fraction 2/45. The 25 region constraints, and likewise the 20 industry constraints, are mutually nonoverlapping in the sense of § 6.

10. Nonlinear cost functions. Here we discuss results that we may obtain when the cost function is not the simple $\sum c_h n_h$ of (1.2).

The first extension is to $\sum c_h n_h^E$, with E an exponent independent of h. In essence, all the results of §§ 2 through 9 may be obtained; there is no proof for § 7; yet there appears to be no better procedure, and we conjecture that the proof holds. In practice, E will be between 0 and 1, reflecting a diminishing cost per additional unit as sample size increases.

A more general extension is to $\sum f_h(x_h)$ where: (1) $x_h=1/n_h$; and (2) $f_h''(x_h)>0$: f_h is convex with respect to x_h, not n_h. In this instance, the solution vector $X=(x_1,\cdots,x_H)$ is unique; in § 4, we may again minimize $\sum u_k g_k$, with the solution vector U being unique if (as is generally the case) the rank of $((B_{kh}))$ is K. Although $\sum u_k g_k$ is not convex (at least, not proven to be), we may show that the matrix A is always positive definite. Results for § 7 are as for the first extension. We cannot use the rule of § 8 (but may develop a cruder rule), for estimating closeness of cost at current solution to cost at optimum.

REFERENCES

F. A. Al-Khayyal, G. D. Capps, J. A. Dorsch, T. J. Hodgson, D. A. Kriegman and P. D. Pavnica (1978), *A Lagrangian dual approach for solving a structured geometric problem arising in sample survey design*, unpublished Census Bureau memorandum to be submitted for publication.

W. G. Cochran (1963), (1977) *Sampling Techniques*, John Wiley, New York, 2nd edition 1963, 3rd edition 1977.

S. CHATTERJEE (1966), *A programming algorithm and its statistical application*, Techn. Rep. 1, Harvard Univ., Cambridge, MA.

——— (1968), *Multivariate stratified surveys*, J. Am. Statist. Assoc., 63, pp. 530–34.

T. DALENIUS (1957), *Sampling in Sweden*, Almqvist and Wicksell, Stockholm.

A. V. FIACCO AND G. P. MCCORMICK (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York.

J. L. FOLKS AND C. E. ANTLE (1965), *Optimum allocation of sampling units to strata where there are R responses of interest*, J. Amer. Statist. Assoc., 60, 234–56.

F. B. HILDEBRAND (1956), *Introduction to Numerical Analysis*, McGraw-Hill, New York.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0017 $01.25/0

THE FINITE MEMORY PREDICTION OF COVARIANCE STATIONARY TIME SERIES*

H. JOSEPH NEWTON† AND MARCELLO PAGANO‡

Abstract. An algorithm is presented for conveniently calculating h step ahead minimum mean square error linear predictors and prediction variances given a finite number of observations from a covariance stationary time series Y. It is shown that elements of the modified Cholesky decomposition of the covariance matrix of observations play the role in finite memory prediction that the coefficients in the infinite order moving average representation of Y play in infinite memory prediction. A by-product of the algorithm is the extension of Pagano's result (J. Assoc. Comput. Mach., 23 (1976), pp. 310–316) on the convergence down the diagonal of the Cholesky factors of a banded Toeplitz matrix to a similar result for a general Toeplitz matrix. This result is applied to autoregressive-moving average time series. A numerical example illustrating the results of the paper is presented.

Key words. covariance stationary time series, minimum mean square error linear prediction, modified Cholesky decomposition algorithm, autoregressive-moving average time series

1. Introduction. An important part of the analysis of data $Y(1), \cdots, Y(T)$ from a time series Y is the construction of predictors $Y(t+h|t)$ and prediction variances $\sigma^2_{t,h} = E\{Y(t+h) - Y(t+h|t)\}^2$ of $Y(t+h)$ given the data $Y(1), \cdots, Y(t)$ up to time t. One often does this for several values of t and h; $t = t_1, \cdots, t_2 \leqq T$ and $h = h_1, \cdots, h_2$. Then the adequacy of a model for Y can be investigated by comparing the one step ahead prediction errors $e(t) = Y(t) - Y(t|t-1)$, $t = 1, \cdots, T$ with white noise while competing models can be compared in terms of how well they predict observed data for various horizons h. Finally, a basic aim of time series analysis is to construct predictors $Y(T+1|T)$, $Y(T+2|T), \cdots$ of unknown future values of Y. Calculation of predictors and prediction variances has been particularly important in the special case of finite parameter autoregressive-moving average (ARMA) processes and their special cases of pure autoregressive and pure moving average processes, particularly since their advocacy by Box and Jenkins (1970).

Many authors have proposed using approximate infinite memory predictors when available rather than finding the exact finite memory predictors. For example in ARMA time series one can fairly easily construct the infinite memory predictors (see Box and Jenkins (1970, p. 126) for example) while the exact finite memory quantities are not easy to determine. In many ARMA processes the infinite memory predictors are entirely adequate but often in small or moderate samples from close to linearly deterministic series the exact and approximate predictors can differ significantly. This fact has been discussed at length in studying the evaluation of the exact Gaussian likelihood $\mathscr{L}$ of the parameters of an ARMA process since $\mathscr{L}$ can be written as a function of the one step ahead prediction errors and prediction variances conditional on a particular set of parameters (see Ansley (1979) for example). Further, in the case of a general covariance stationary time series, infinite memory predictors are not available so one must construct finite memory predictors.

* Received by the editors October 14, 1980, and in revised form October 5, 1981.

† Institute of Statistics, Texas A & M University, College Station, Texas 77843. The research of this author was supported in part by Office of Naval Research grant N00014-78-C-0599.

‡ Harvard University and Sidney Farber Cancer Institute, Boston, Massachusetts 02115. The research of this author was supported by grants CA-23415 and CA-28066 from the National Cancer Institute, HHS.

Existing algorithms for finding exact finite memory predictors and prediction variances are of two basic types: 1) solving directly the defining Toeplitz normal equations in various efficient ways (see Kailath, Vieira and Morf (1978) or Grcar and Sameh (1981) for example); and 2) when possible calculating them recursively; for example using the Kalman filter algorithm (see Gardner, Harvey and Phillips (1980) for a description of the Kalman filter algorithm applied to ARMA processes).

The purpose of this paper is to describe an algorithm for the general case and to apply the results to ARMA processes. The algorithm is based on the modified Cholesky decomposition (MCD) of a Toeplitz covariance matrix. It extends the results of Pagano (1976), based on the work of Bauer (1955), concerning the MCD of the banded covariance matrix of a pure moving average process to the case of a general Toeplitz covariance matrix.

Section 2 contains the algorithm as Theorem 1 which also shows explicitly how the quantities required in the algorithm approach the quantities used in infinite memory prediction as memory length increases. In § 3, Theorem 2 presents the results of applying Theorem 1 to ARMA processes. Also Theorems 1 and 2 are used to prove a number of results about the factors in the MCD of various covariance matrices related to that of the ARMA process, thus greatly simplifying the algorithm. These results are collected as a lemma. We note that our algorithm avoids the necessity of calculating the entire state covariance matrix at each step as in the prediction algorithm using the Kalman filter algorithm while having the advantage of the numerical stability afforded by the MCD algorithm (see Wilkinson (1967) for example).

Finally a numerical example illustrating the lemma of § 3 is given in § 4.

2. Finite memory, horizon h, minimum mean square error linear prediction of covariance stationary time series. Consider a zero mean covariance stationary time series $\{Y(t), t=0, \pm 1, \cdots\}$ with autocovariance function $R(v)=E(Y(t)Y(t+v))$. Then given observations $Y(1), \cdots, Y(T)$, the horizon h, memory T, minimum mean square error linear predictor $Y(T+h|T)$ of $Y(T+h)$ is given by that linear combination of $Y(1), \cdots, Y(T)$ that is closest to $Y(T+h)$ in mean square error. The corresponding prediction variance $\sigma^2_{T,h}$ is given by this minimum mean squared error

$$\sigma^2_{T,h}=E\{Y(T+h)-Y(T+h|T)\}^2.$$

Thus (see Whittle (1963, p. 47))

$$Y(T+h|T)=\sum_{j=1}^{T}\lambda_{T,h}(j)Y(T+1-j),$$

where $\boldsymbol{\lambda}_{T,h}=(\lambda_{T,h}(1), \cdots, \lambda_{T,h}(T))^T$ satisfies the normal equations $\Gamma_T\boldsymbol{\lambda}_{T,h}=\mathbf{r}_{T,h}$, where $\mathbf{r}_{T,h}=(R(h), \cdots, R(h+T-1))^T$, and $\Gamma_T=\text{TOEPL}\,(R(0), \cdots, R(T-1))$, i.e., Γ_T is the $T\times T$ symmetric Toeplitz matrix having (j,k) element $R(|j-k|)$. We denote the transpose of a matrix A by A^T and the transpose of the inverse (or the inverse of the transpose) of A by A^{-T}.

Suppose that Y is purely nondeterministic, i.e.,

$$\sigma^2_\infty=2\pi\exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\log f(\omega)\,d\omega\right\}>0,$$

where f is the spectral density function of Y. Then $Y(t)$ can be represented as the limit in mean square of an infinite order moving average process, i.e.,

$$Y(t)=\sum_{k=0}^{\infty}\beta_\infty(k)\varepsilon(t-k), \tag{2.1}$$

where $\varepsilon(t)$ is the infinite memory horizon one error in predicting $Y(t)$, and $E(\varepsilon(T)\varepsilon(T-j))=\delta_j\sigma_\infty^2$ for all integer T and j, where δ_j is the Kronecker delta. Also the horizon h, minimum mean square error linear infinite memory predictor $Y(T+h|T, T-1, \cdots)$ and prediction variance $\sigma^2_{T,h,\infty}$ are given by

$$Y(T+h|T, T-1, \cdots)=\sum_{k=h}^{\infty}\beta_\infty(k)\varepsilon(T+h-k), \tag{2.2}$$

$$\sigma^2_{T,h,\infty}=\sigma^2_\infty\sum_{k=0}^{h-1}\beta^2_\infty(k). \tag{2.3}$$

The process Y being purely nondeterministic also means that its autocovariance function is positive definite. Thus for all T we can form the modified Cholesky decomposition (Wilkinson (1967)) $\Gamma_T=L_TD_TL_T^T$ of Γ_T, where L_T is a $T\times T$ unit lower triangular matrix and D_T is a $T\times T$ diagonal matrix. An important property of L_T and D_T is that they are nested for increasing T, i.e.,

$$L_{T+1}=\begin{bmatrix}L_T & \mathbf{0}\\ \mathbf{l}^T & 1\end{bmatrix},\qquad D_{T+1}=\begin{bmatrix}D_T & \mathbf{0}\\ \mathbf{0}^T & d_{T+1}\end{bmatrix}.$$

Thus the (j, k)th element of L_T will be referred to as L_{jk} and the jth diagonal element of D_T as d_j for any $T\geqq j, k$.

The following theorem shows the role played by L_T and D_T in finite memory prediction.

THEOREM 1. *Let Y be a purely nondeterministic covariance stationary time series with covariance function R. Let $\Gamma_T=L_TD_TL_T^T$ be the modified Cholesky decomposition of the covariance matrix of $\mathbf{Y}_T^T=(Y(1), \cdots, Y(T))$. Define $\mathbf{e}_T^T=(e(1), \cdots, e(T))$ by $L_T\mathbf{e}_T=\mathbf{Y}_T$. Then*

a) $Y(T+h|T)=\sum_{k=h}^{T+h-1}L_{T+h,T+h-k}e(T+h-k)$,

b) $\sigma^2_{T,h}=E\{Y(T+h)-Y(T+h|T)\}^2=\sum_{k=0}^{h-1}L^2_{T+h,T+h-k}d^2_{T+h-k}$,

c) i) $\lim_{T\to\infty}L_{T,T-j}=\beta_\infty(j), j\geqq 0$,

ii) $\lim_{T\to\infty}d_T=\sigma^2_\infty$.

Proof of Theorem 1. a) Defining the $T\times T$ permutation matrix P_T to be a matrix of zeros with ones on the main reverse diagonal, we have $Y(T+h|T)=\boldsymbol{\lambda}^T_{T,h}P_T\mathbf{Y}_T$, where $\Gamma_T\boldsymbol{\lambda}_{T,h}=\mathbf{r}_{T,h}$ since premultiplication (postmultiplication) by P_T reverses row (column) order of a matrix. Thus $Y(T+h|T)=\mathbf{r}^T_{T,h}\Gamma_T^{-1}P_T\mathbf{Y}_T=\mathbf{r}^T_{T,h}P_T\Gamma_T^{-1}P_TP_T\mathbf{Y}_T=\boldsymbol{\rho}^T_{T,h}\Gamma_T^{-1}\mathbf{Y}_T$, where $\boldsymbol{\rho}_{T,h}=P_T\mathbf{r}_{T,h}$ since $P_T^2=I_T$, and since for the symmetric Toeplitz matrix Γ_T^{-1}, we have $P_T\Gamma_T^{-1}P_T=\Gamma_T^{-1}$. Thus $Y(T+h|T)=\boldsymbol{\rho}^T_{T,h}L_T^{-T}D_T^{-1}L_T^{-1}\mathbf{Y}_T=\boldsymbol{\rho}^T_{T,h}L_T^{-T}D_T^{-1}\mathbf{e}_T$. To show that this is the result in a) we note 1) $\boldsymbol{\rho}^T_{T,h}=(R(T+h-1), \cdots, R(h))$ is the last row of Γ_{T+h} without its last h elements; 2) $\Gamma_TL_T^{-T}D_T^{-1}=L_T$ for all T; and 3) because of the nesting of the L_T and D_T, $L_T^{-T}D_T^{-1}$ is the $T\times T$ matrix consisting of the first T rows and columns of the upper triangular matrix $L_{T+h}^{-T}D_{T+h}^{-1}=\Gamma_{T+h}^{-1}L_{T+h}$.

Thus $(\boldsymbol{\rho}^T_{T,h}L_T^{-T}D_T^{-1})_k=(\Gamma_{T+h}L_{T+h}^{-T}D_{T+h}^{-1})_{T+h,k}=(\Gamma_{T+h}\Gamma_{T+h}^{-1}L_{T+h})_{T+h,k}=L_{T+h,k}$, proving a).

To prove b), note that $\sigma^2_{T,h}=R(0)-\mathbf{r}^T_{T,h}\Gamma_T^{-1}\mathbf{r}_{T,h}=R(0)-\boldsymbol{\rho}^T_{T,h}L_T^{-T}D_T^{-1}L_T^{-1}\boldsymbol{\rho}_{T,h}=R(0)-\mathbf{l}^T_{T,h}D_T\mathbf{l}_{T,h}$, where $\mathbf{l}^T_{T,h}=\boldsymbol{\rho}^T_{T,h}L_T^{-T}D_T^{-1}$ which as above is the row vector $(L_{T+h,1}, \cdots, L_{T+h,T})$. Also

$$R(0)=\Gamma_{T+h,T+h}=(L_{T+h}D_{T+h}L^T_{T+h})_{T+h,T+h}=\sum_{k=1}^{T+h}L^2_{T+h,k}d_k,$$

thus proving b).

To prove c) we first note that multiplying both sides of (2.1) for $t = T$ by $\varepsilon(T-j)$ and taking expectations gives

$$E(Y(T)\varepsilon(T-j)) = \beta_\infty(j)\sigma_\infty^2.$$

We next note that

$$E(e(T)e(T-j)) = \delta_j d_T,$$

$$E(Y(T)e(T-j)) = \sum_{k=1}^{T} L_{T,k}E(e(k)e(T-j)) = L_{T,T-j}d_{T-j}$$

and that $e(1) = Y(1)$, $e(t) = Y(t) - Y(t|t-1, \cdots, 1)$, $t = 2, \cdots, T$, where the notation $Y(t|t-1, \cdots, 1)$ makes explicit which Y's are used in predicting $Y(t)$. Then by the stationarity of Y we have that

$$\begin{aligned} L_{T,T-j}d_{T-j} &= E(Y(T)e(T-j)) = E\{Y(T)[Y(T-j) - Y(T-j|T-j-1, \cdots, 1)]\} \\ &= E\{Y(0)[Y(-j) - Y(-j|-j-1, \cdots, 1-T)]\} \end{aligned}$$

by subtracting T from all indices. Then letting T approach infinity gives that

$$E\{Y(0)[Y(-j) - Y(-j|-j-1, \cdots, 1-T)]\} \to E[Y(0)\varepsilon(-j)] = \beta_\infty(j)\sigma_\infty^2$$

since the memory T prediction error converges to the infinite memory prediction error. A similar argument shows $d_T \to \sigma_\infty^2$ thus proving c). □

Thus comparing a) with (2.2) and b) with (2.3), it is clear that the elements of L_T and D_T are playing the role in finite memory prediction of $B_\infty(\cdot)$ and σ_∞^2 in the infinite memory case, while c) makes explicit the connection. In the next section we describe how this algorithm can be simplified for an ARMA process.

3. Application to ARMA processes. The univariate ARMA process $\{Y(t), t = 0, \pm 1, \cdots\}$ of order (p, q) and parameters $\alpha(1), \cdots, \alpha(p), \beta(1), \cdots, \beta(q)$ and σ^2 is defined by

$$\sum_{j=0}^{p} \alpha(j)Y(t-j) = \sum_{k=0}^{q} \beta(k)\varepsilon(t-k), \qquad t = 0, \pm 1, \cdots,$$

where $\alpha(0) = \beta(0) = 1$, and $E(\varepsilon(t)) = 0$, $E(\varepsilon(t)\varepsilon(t+v)) = \delta_v\sigma^2$. We assume that the zeros of the complex polynomial $g(z) = \sum_{j=0}^{p} \alpha(j)z^j$ are all greater than one in modulus and that $g(z)$ and $h(z) = \sum_{k=0}^{q} \beta(k)z^k$ have no common zeros so that Y does indeed have an infinite order moving average representation.

Then given a realization $\mathbf{Y}_T = (Y(1), \cdots, Y(T))^T$ from $Y(\cdot)$ we define the following quantities:

i) The $T \times T$ covariance matrix $\Gamma_{Z,T} = \text{TOEPL}(R_Z(0), \cdots, R_Z(T-1))$ and its MCD $\Gamma_{Z,T} = L_{Z,T}D_{Z,T}L_{Z,T}^T$, where $Z(\cdot)$ is an autoregressive process of order p with coefficients $\alpha(1), \cdots, \alpha(p)$. Thus $Z(\cdot)$ is referred to as the autoregressive part of $Y(\cdot)$.

ii) The vector $\mathbf{X}_T = (X(1), \cdots, X(T))^T = L_{Z,T}^{-1}\mathbf{Y}_T$, its covariance matrix $\Gamma_{X,T} = E(\mathbf{X}_T\mathbf{X}_T^T) = L_{Z,T}^{-1}\Gamma_{Y,T}L_{Z,T}^{-T}$, where $\Gamma_{Y,T} = \text{TOEPL}(R_Y(0), \cdots, R_Y(T-1))$ and the MCD $\Gamma_{X,T} = L_{X,T}D_{X,T}L_{X,T}^T$.

iii) The vector $\mathbf{e}_T = (e(1), \cdots, e(T))^T = L_{X,T}^{-1}\mathbf{X}_T$.

We note that since $L_{Z,T}^{-1}$ and $L_{X,T}$ are nested for increasing T, then so are $\mathbf{X}_T$ and $\mathbf{e}_T$, i.e., $\mathbf{X}_{T+1}^T = (\mathbf{X}_T^T, X(T+1))$, $\mathbf{e}_{T+1}^T = (\mathbf{e}_T^T, e(T+1))$.

With these quantities defined, the basic ARMA algorithm is contained in the following theorem:

THEOREM 2. a) $Y(T+h|T)=X(T+h|T)-\sum_{j=1}^{p}\alpha(j)Y(T+h-j|T)$, $h\geqq 1$, *where*

i) $$X(T+h|T)=\begin{cases}\sum_{k=1}^{q} L_{X,T+h,T+h-k}e(T+h-k), & h=1,\cdots,q,\\ 0, & h\geqq q,\end{cases}$$

ii) $Y(T+h-j|T)=Y(T+h-j)$ *if* $j\geqq h$;

b) $\sigma_{T,h}^2=\sum_{k=0}^{h-1}(L_{Z,T+h}L_{X,T+h})_{T+h,T+h-k}^2\, d_{X,T+h-k}$, $h\geqq 1$;

c) $(L_{Z,T}L_{X,T})_{T,T-k}\to\beta_\infty(k)$ *and* $d_{X,T}\to\sigma^2$ *as* $T\to\infty$.

Proof. Since $\Gamma_{Y,T}=L_{Z,T}\Gamma_{X,T}L_{Z,T}^T=L_{Z,T}L_{X,T}D_{X,T}L_{X,T}^TL_{Z,T}^T$ and the MCD of a matrix is unique we have that $L_{Y,T}=L_{Z,T}L_{X,T}$ and $D_{Y,T}=D_{X,T}$. Thus Theorem 2b) and 2c) follow immediately from Theorem 1b) and 1c). We note also that this further factorization $L_{Y,T}=L_{Z,T}L_{X,T}$ produces in two steps the transformation $\mathbf{Y}_T=L_{Y,T}\mathbf{e}_T=L_{Z,T}L_{X,T}\mathbf{e}_T$ from $\mathbf{Y}_T$ to the uncorrelated set of one step ahead prediction errors $\mathbf{e}_T$.

To prove Theorem 2a), note that since $\mathbf{X}_T=L_{Z,T}^{-1}\mathbf{Y}_T$, we have $\boldsymbol{\rho}_{XY,T,h}\equiv E(\mathbf{X}_TY(T+h))=L_{Z,T}^{-1}\boldsymbol{\rho}_{Y,T,h}$. Also for $T+h>p$,

$$\begin{aligned}\boldsymbol{\rho}_{XY,T,h}&=E\left\{\mathbf{X}_T\left[X(T+h)-\sum_{j=1}^{p}\alpha(j)Y(T+h-j)\right]\right\}\\&=\boldsymbol{\rho}_{X,T,h}-\sum_{j=1}^{p}\alpha(j)\boldsymbol{\rho}_{XY,T,h-j}\\&=\boldsymbol{\rho}_{X,T,h}-\sum_{j=1}^{p}\alpha(j)L_{Z,T}^{-1}\boldsymbol{\rho}_{Y,T,h-j},\end{aligned}$$

where

$$\boldsymbol{\rho}_{X,T,h}=E(\mathbf{X}_TX(T+h)).$$

Thus $\boldsymbol{\rho}_{Y,T,h}=L_{Z,T}\boldsymbol{\rho}_{XY,T,h}=L_{Z,T}\boldsymbol{\rho}_{X,T,h}-\sum_{j=1}^{p}\alpha(j)\boldsymbol{\rho}_{Y,T,h-j}$. Therefore,

$$\begin{aligned}Y(T+h|T)&=\boldsymbol{\rho}_{Y,T,h}^TL_{Z,T}^{-T}L_{X,T}^{-T}D_{X,T}^{-1}\mathbf{e}_T\\&=\boldsymbol{\rho}_{X,T,h}^TL_{Z,T}^TL_{Z,T}^{-T}L_{X,T}^{-T}D_{X,T}^{-1}\mathbf{e}_T\\&\quad-\sum_{j=1}^{p}\alpha(j)\boldsymbol{\rho}_{Y,T,h-j}^TL_{Z,T}^{-T}L_{X,T}^{-T}D_{X,T}^{-1}\mathbf{e}_T\\&=\boldsymbol{\rho}_{X,T,h}^TL_{X,T}^{-T}D_{X,T}^{-1}\mathbf{e}_T-\sum_{j=1}^{p}\alpha(j)Y(T+h-j|T).\end{aligned}$$

An argument identical to that used in the proof of Theorem 1a) proves part i) of 2a). To verify 2aii) we substitute for $\mathbf{e}_T$ and $\mathbf{X}_T$ to obtain $\boldsymbol{\rho}_{Y,T,h-j}^TL_{Z,T}^{-T}L_{X,T}^{-T}D_{X,T}^{-1}\mathbf{e}_T=\boldsymbol{\rho}_{Y,T,h-j}^T\Gamma_{Y,T}^{-1}\mathbf{Y}_T=Y(T+h-j)$, since $\boldsymbol{\rho}_{Y,T,h-j}^T$ is the $(T+h-j)$th row of $\Gamma_{Y,T}$. □

From Theorem 2 and the definitions given above then, to find $Y(T+h|T)$ and $\sigma_{T,h}^2$ for $h=h_1,\cdots,h_2$ and $T=T_1,\cdots,T_2$ one needs the matrices L_{X,T_2+h_2}, L_{Z,T_2+h_2} and D_{X,T_2+h_2} as well as the vectors $\mathbf{X}_{T_2}$ and $\mathbf{e}_{T_2}$. However the following lemma provides significant reductions in the task of computing and storing these arrays.

LEMMA 1. a) *The jth row of* $L_{Z,T}^{-1}$ *is given by*

$$\mathbf{I}_j^T=\begin{cases}(1,\mathbf{0}_{T-1}^T), & j=1,\\(\alpha_{j-1}(j-1),\cdots,\alpha_{j-1}(1),1,\mathbf{0}_{T-j}^T), & j=2,\cdots,p,\\(\mathbf{0}_{j-p-1}^T,\alpha(p),\cdots,\alpha(1),1,\mathbf{0}_{T-j}^T), & j=p+1,\cdots,T,\end{cases}$$

where defining $\alpha_p(j)=\alpha(j)$, $j=1,\cdots,p$, *we have*

$$\alpha_{k-1}(j)=\frac{\alpha_k(j)-\alpha_k(k)\alpha_k(k-j)}{1-\alpha_k^2(k)}, \qquad 1\leqq j\leqq k-1\leqq p.$$

b) *The elements of* $\mathbf{X}_T$ *can be obtained by* $X(1)=Y(1)$ *while*

$$X(j)=\begin{cases} Y(j)+\sum_{l=1}^{j-1}\alpha_{j-1}(l)Y(j-l), & j=2,\cdots,p,\\ Y(j)+\sum_{l=1}^{p}\alpha(l)Y(j-l), & j>p.\end{cases}$$

c) *The* (i,j)th *element of* $\Gamma_{X,T}$ *is given by*

$$\Gamma_{X,i,j}=\begin{cases} \sum_{m=\max(1,j-p)}^{j}\alpha_{j-1}(j-m)\sum_{l=\max(1,i-p)}^{i}\alpha_{i-1}(i-l)R_Y(l-m), & i,j\geqq 1,\\ R_X(|i-j|), & i,j>p,\quad |i-j|\leqq q,\\ 0, & \text{if } 1\leqq j\leqq p \text{ and } i>p \text{ and } i-j>q \text{ or if } i,j>p \text{ and } |i-j|>q,\end{cases}$$

where $R_X(v)=\sigma^2\sum_{k=0}^{q-v}\beta(k)\beta(k+v)$, $v=0,\cdots,q$.

d) i) *The unit lower triangular matrix* $L_{X,T}$ *for* $T\geqq p+q$ *consists of zeros except for the elements of its first* $p+q$ *rows and columns and the* q *elements* $L_{X,k,k-q},\cdots,L_{X,k,k-1}$, $L_{X,k,k}=1$ *of row* k *for* $k>p+q$,

ii) $L_{X,T,T-k}\to\beta(k)$, $k=1,\cdots,q$ *as* $T\to\infty$,

iii) *The vector* $\mathbf{e}_T$ *defined by* $L_{X,T}\mathbf{e}_T=\mathbf{X}_T$ *can be obtained by* $e(1)=X(1)$ *while* $e(j)=X(j)-\sum_{l=\max(1,j-q)}^{j-1}L_{x,j,l}\,e(l)$, $j\geqq 2$.

e) *Let* $\gamma(0)$, $\gamma(1)$, $\gamma(2),\cdots$ *be the coefficients of the infinite order moving average representation of the autogressive part of* $Y(\cdot)$, *i.e.*, $\gamma(0)=1$ *while* $\gamma(j)=-\sum_{l=\max(0,j-p)}^{j-1}\alpha(j-l)\gamma(l)$, $j\geqq 1$.

Then

i) $L_{Z,p}=(L_{Z,p}^{-1})^{-1}\Rightarrow L_{Z,j,j-k}=-\sum_{r=j-k+1}^{j}L_{Z,j,r}L_{Z,r,j-k}^{-1}$, $k<j\leqq p$, *where* $L_{Z,j,k}^{-1}$ *denotes the* (j,k)*th element of* $L_{Z,p}^{-1}$.

ii) $L_{Z,p+j,p+j-k}=\gamma(k)$, $k=0,\cdots,j$, $j\geqq 0$,

iii) $L_{Z,p+j,l}=-\sum_{r=1}^{p}\alpha(r)L_{Z,p+j-r,l}$, $l=1,\cdots,p-1$, $j\geqq 1$,

iv) $\lim_{K\to\infty}\sum_{j=0}^{K}\gamma^2(k)=\lim_{K\to\infty}\sum_{j=1}^{K}L_{Z,K,j}^2=1/\prod_{j=1}^{p}(1-\alpha_j^2(j))$,

v) $\lim_{K\to\infty}\sum_{j=1}^{p-1}L_{Z,K,j}^2=0$.

Proof. The expression for $\mathbf{l}_j^T$ in a) is the well-known basis for Durbin's (1960) recursive algorithm for calculating autoregressive parameters $\alpha(1),\cdots,\alpha(p)$ and σ^2 from covariances $R_Z(0),\cdots,R_Z(p)$: $\alpha_1(1)=-R_Z(1)/R_Z(0)$, $\sigma_1^2=R_Z(0)(1-\alpha_1^2(1))$,

$$\alpha_k(k)=-\Big\{R_Z(k)+\sum_{j=1}^{k-1}\alpha_{k-1}(j)R_Z(k-j)\Big\}/\sigma_{k-1}^2,$$

$$\alpha_k(j)=\alpha_{k-1}(j)+\alpha_k(k)\alpha_{k-1}(k-j), \qquad 1\leqq j\leqq k-1,$$

$$\sigma_k^2=\sigma_{k-1}^2(1-\alpha_k^2(k)), \qquad k=2,\cdots,p.$$

Then $\alpha(j)=\alpha_p(j)$, $j=1,\cdots,p$ and $\sigma^2=\sigma_p^2$. If we write the second equation above for j and $k-j$ and solve for $\alpha_{k-1}(j)$ we obtain the backward recursion given in a).

The result in b) is a direct consequence of a) and the fact that $\mathbf{X}_T=L_{Z,T}^{-1}\mathbf{Y}_T$. Similarly the first expression in part c) is obtained by comparing (i,j) elements of $\Gamma_{X,T}=L_{Z,T}^{-1}\Gamma_{Y,T}L_{Z,T}^{-T}$, while the second expression is true since for $j>p$, $X(j)=$

$\sum_{l=0}^{p}\alpha(l)Y(j-l)=\sum_{k=0}^{q}\beta(k)\varepsilon(j-k)$, i.e., $X(p+1),\cdots,X(T)$ is a realization from a moving average process of order q with parameters $\beta(1),\cdots,\beta(q)$ and σ^2. Further, if $1\leqq j\leqq p, i>p$ and $i-j>q$ we have $\Gamma_{X,i,j}=\sum_{m=1}^{j}\alpha_{j-1}(j-m)\sum_{l=i-p}^{i}\alpha(i-l)R_Y(l-m)$ and this second summation is equal to $\sum_{l=0}^{p}\alpha(l)R_Y(l-(i-m))=0$ since $i-m\geqq i-j>q$ and $\sum_{l=0}^{p}\alpha(l)R_Y(l-v)=0$, $v>q$, for an ARMA process. Thus part c) details where the nonzero elements are in $\Gamma_{X,T}$, and since the pattern of zeros in $L_{X,T}$ is the same as that of the lower triangle of $\Gamma_{X,T}$ we have part di) from which diii) follows immediately.

To prove part dii) we note that Theorem 1c) shows that the rows of $L_{Z,T}$ are converging to the coefficients $\gamma(\cdot)$ of the infinite order moving average representation of $Z(\cdot)$, while Theorem 2c) shows that the rows of $L_{Z,T}L_{X,T}$ are converging to the coefficients of the infinite order moving average representation of $Y(\cdot)$. Thus the rows of $L_{X,T}$ must be converging to the coefficients of the infinite order representation of the moving average part, i.e., to $\beta(1),\cdots,\beta(q)$.

Part ei) is obvious, eii) follows because multiplying the $(p+j)$th row of $L_{Z,T}$ times the $(p+j-k)$th column of $L_{Z,T}^{-1}$ for $k\geqq 0$ gives $\sum_{l=0}^{\min(k,p)}\alpha(l)L_{Z,p+j,p+j-(k-l)}=\delta_k$, i.e., the $L_{Z,p+j,p+j-k}$ satisfy the same difference equation in k as $\gamma(k)$. We obtain eiii) by multiplying the $(p+j)$th row of $L_{Z,T}^{-1}$ times the kth column of $L_{Z,T}$ for $k=1,\cdots,p-1$ and $j\geqq 1$.

Finally ev) follows from eiv) and eii), while eiv) is proved as follows. The infinite order moving average representation of $Z(t)$ is given by $Z(t)=\sum_{k=0}^{\infty}\gamma(k)\varepsilon(t-k)$ and thus $R_Z(0)=\sigma^2\sum_{k=0}^{\infty}\gamma^2(k)$. But from Durbin's algorithm we have that $R_Z(0)=\sigma^2/\prod_{j=1}^{p}(1-\alpha_j^2(j))$ from which the equality of the first and third terms in eiv) follows. The equality of the first and second terms is a restatement of the fact that the rows of $L_{Z,T}$ are converging to the $\gamma(\cdot)$'s. □

Summary of the ARMA *algorithm.* Given $Y(1),\cdots,Y(T_2)$, $\alpha(1),\cdots,\alpha(p)$, $\beta(1),\cdots,\beta(q)$, σ^2 and convergence factors δ_1,δ_2 one obtains $Y(T+h|T)$ and $\sigma_{T,h}^2$ for $h=h_1,\cdots,h_2$ and $T=T_1,\cdots,T_2$ by performing the following steps:

1) Find $R_Y(0),\cdots,R_Y(p+q)$ via the algorithm of McLeod (1975), $L_{Z,p}^{-1}$ and $1/\prod_{j=1}^{p}(1-\sigma_j^2(j))$ by Lemma 1a), and $\mathbf{X}_{T_2}$ by Lemma 1b).

2) Calculate M_1 and $\gamma(0),\gamma(1),\cdots,\gamma(M_1)$, where M_1 is determined so that $\|\sum_{j=0}^{M_1}\gamma^2(j)-1/\prod_{j=1}^{p}(1-\alpha_j^2(j))\|<\delta_1$ for some norm. Then calculate the first $p-1$ elements of rows $1,\cdots,M_1$ of L_{Z,T_2+h_2} by Lemma 1 ei) and eiii). Note that L_{Z,T_2+h_2} is not needed if one is purely calculating predictors.

3) Calculate the first $p+q$ elements of the first $p+q$ rows of Γ_{X,T_2+h_2}, L_{X,T_2+h_2} and the first $p+q$ diagonal elements of D_{X,T_2+h_2}. Then successively calculate the q nonzero elements of rows $p+q+1,\cdots,M_2$ of L_{X,T_2+h_2} and the corresponding diagonal elements of D_{X,T_2+h_2}, where M_2 is determined so that $\|L_{X,M_2,M_2-k}-\beta(k)\|<\delta_2$, $k=1,\cdots,q$ and $\|d_{X,M_2}-\sigma^2\|<\delta_2$.

4) Find $\mathbf{e}_{T_2}$ by Lemma 1 diii) with $\beta(k)$ replacing $L_{X,j,j-k}$ for $j>M_2$.

5) Calculate the $Y(T+h|T)$ and $\sigma_{T,h}^2$ using Theorem 2a) and 2b) taking advantage of Theorem 2c) in evaluating $\sigma_{T,h}^2$ and Lemma 1 dii) in evaluating $Y(T+h|T)$.

We note the significant simplification of the algorithm if one only wants $h_1=h_2=1$ since then $e(T)=Y(T)-Y(T|T-1)$ and $d_{X,T}=\sigma_{T-1,1}^2$. This special case, with a slightly different factorization of $L_{Y,T}$ is essentially the algorithm of Ansley (1979) for evaluating the ARMA Gaussian likelihood. Thus the results of our Lemma 1 would greatly reduce the computations in Ansley's algorithm. We give a numerical example illustrating the results of the lemma in the next section.

We end this section by considering the special case of a pure autoregressive process ($q=0$) and a pure moving average process ($p=0$). If $q=0$ we have $\Gamma_{Y,T}=$

$\Gamma_{Z,T} = L_{Z,T} D_{Z,T} L_{Z,T}^T$ and thus $L_{X,T} = I_T$, $D_{X,T} = D_{Z,T}$. Thus

$$Y(T+h|T) = -\sum_{j=1}^{p} \alpha(j) Y(T+h-j|T), \qquad h \geqq 1,$$

$$\sigma_{T,h}^2 = \sum_{k=0}^{h-1} L_{Z,T+h,T+h-k}^2 d_{Z,T+h-k} = \sigma^2 \sum_{k=0}^{h-1} \gamma^2(k) \quad \text{if } T > p.$$

If $p = 0$ we have $\Gamma_{Z,T} = I_T$ which gives that $L_{Z,T} = D_{Z,T} = I_T$ and thus $\Gamma_{Y,T} = \Gamma_{X,T} =$ TOEPL$(R_X(0), \cdots, R_X(q), 0, \cdots, 0)$, so that

$$Y(T+h|T) = \begin{cases} \sum_{k=h}^{q} L_{X,T+h,T+h-k} e(T+h-k), & h = 0, \cdots, q, \\ 0, & h > q, \end{cases}$$

$$\sigma_{T,h}^2 = \sum_{k=0}^{h-1} L_{X,T+h,T+h-k}^2 d_{X,T+h-k}, \qquad h \geqq 1.$$

And $L_{X,T,T-k} \to \beta(k)$, $d_{X,T} \to \sigma^2$ as $T \to \infty$. This is the result given by Pagano (1976).

TABLE 1

Variances and first 10 *autocorrelations* $\rho_Y(\cdot)$, $\rho_Z(\cdot)$, $\rho_W(\cdot)$ *of* Y, *autoregressive part of* Y *and moving average part of* Y, *where* Y *is the above* ARMA (4, 3) *process.*

v	$\rho_Y(v)$	$\rho_Z(v)$	$\rho_W(v)$
1	−.2227	.3806	−.4548
2	−.0749	−.0112	−.0230
3	.0616	−.2897	.1347
4	−.1949	−.4128	0
5	−.0015	−.2107	0
6	.0250	.0115	0
7	.0233	.1603	0
8	.0560	.1919	0
9	.0134	.1036	0
10	−0.102	−.0091	0
Variance	1.1306	1.3891	1.4124

TABLE 2

First 10 *terms in infinite order moving average representation of each of series in Table* 1.

j	ARMA	AR part	MA part
1	−.272	.336	−.608
2	−.090	.031	.083
3	.025	−.174	.190
4	−.198	−.370	0
5	.015	−.201	0
6	.041	−.018	0
7	.037	.114	0
8	.058	.166	0
9	.006	.101	0
10	−.019	.007	0

4. A numerical example. Consider the ARMA process Y of order $p=4$ and $q=3$ with $\alpha(1)=-.3357$, $\alpha(2)=.0821$, $\alpha(3)=.1570$, $\alpha(4)=.2567$, $\beta(1)=-.6077$, $\beta(2)=.0831$, $\beta(3)=.1903$ and $\sigma^2=1$. Then the variances and first 10 autocorrelations of Y (denoted $\rho_Y(\cdot)$), the autoregressive part of Y (denoted $\rho_Z(\cdot)$) and the moving average part of Y (denoted $\rho_W(\cdot)$) are given in Table 1, while Table 2 gives the first 10 terms in the infinite order moving average representation of Y, Z and W.

Finally Table 3 illustrates the convergence proved in § 3.

TABLE 3
The matrices $L_{Z,10}L_{X,10}$, $L_{Z,10}$, $L_{X,10}$.

$L_{Z,10}L_{X,10}$:									
1									
−.223	1								
−.075	−.252	1							
.062	−.064	−.250	1						
−.195	.019	−.091	−.247	1					
−.002	−.205	.013	−.090	−.266	1				
.025	.004	−.206	.013	−.094	−.271	1			
.023	.032	.008	−.206	.021	−.091	−.270	1		
.056	.038	.041	.007	−.200	.024	−.090	−.271	1	
.013	.062	.042	.041	.014	−.199	.025	−.090	−.272	1
$L_{Z,10}$:									
1									
.381	1								
−.011	.450	1							
−.290	.116	.403	1						
−.413	−.155	.053	.336	1					
−.211	−.389	−.172	.031	.336	1				
.012	−.252	−.382	−.174	.031	.336	1			
.160	−.058	−.226	−.370	−.174	.031	.336	1		
.192	.102	−.031	−.201	−.370	−.174	.031	.336	1	
.104	.178	.112	−.018	−.201	−.370	−.174	.031	.336	1
$L_{X,10}$:									
1									
−.603	1								
.208	−.702	1							
.338	.102	−.653	1						
	.177	.075	−.583	1					
		.180	.075	−.602	1				
			.180	.077	−.607	1			
				.188	.082	−.606	1		
					.189	.083	−.606	1	
						.190	.083	−.607	1

Acknowledgments. The authors would like to thank Professor Emanuel Parzen for helpful discussions. We would also like to thank the referees for their helpful comments.

REFERENCES

C. ANSLEY (1979), *An algorithm for the exact likelihood of a mixed autoregressive-moving average process*, Biometrika, 66, pp. 59–65.

F. L. BAUER (1955), *Ein direktes Iterationsverfahren zur Hurwitz-Zerlegung eines Polynoms*, Archiv. Elekt. Ubertr., 9, pp. 285–290.

G. E. P. BOX AND G. M. JENKINS (1970), *Time Series Analysis: Forecasting and Control*, Holden-Day, San Francisco.

J. DURBIN (1960), *The fitting of time-series models*, Rev. Int. Statist. Inst., 28, pp. 233–244.

G. GARDNER, A. C. HARVEY AND G. D. PHILLIPS (1980), *An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering*, Appl. Statist., 29, pp. 311–322.

J. GRCAR AND A. SAMEH (1981). *On certain parallel Toeplitz linear systems solvers*, this Journal, 2, pp. 238–256.

T. KAILATH, A. VIEIRA AND M. MORF (1978). *Inverses of Toeplitz operators, innovations, and orthogonal polynomials*, SIAM Rev., 20, pp. 106–119.

I. MCLEOD (1975). *The derivation of the theoretical autocovariance function of autoregressive-moving average time series*. Appl. Statist., 24, pp. 255–256.

M. PAGANO (1976), *On the linear convergence of a covariance factorization algorithm*, J. Assoc. Comp. Mach., 23, pp. 310–316.

P. WHITTLE (1963). *Prediction and Regulation by Linear Least Squares Methods*, English Univ. Press, London.

J. H. WILKINSON (1967). *The solution of ill-conditioned linear equations*, in Mathematical Methods for Digital Computers II, A. Ralston and H. S. Wilf, eds. Wiley-Interscience, New York.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0018 $01.25/0

A MODIFIED GALERKIN PROCEDURE FOR BENDING OF BEAMS ON ELASTIC FOUNDATIONS*

J. BIELAK† AND E. STEPHAN‡

Abstract. In this paper we give a formulation for bending of beams supported on elastic foundations by incorporating the Green's function for different foundation models, e.g., Pasternak and elastic half space. For the corresponding Galerkin procedure we derive quasi-optimal asymptotic error estimates showing the *same* order of convergence as known for the standard Galerkin procedure for beam bending problems. The error analysis hinges on a Gårding inequality for our bilinear form which is proved with the calculus of pseudodifferential operators. The numerical experiments show superconvergence and underline the theoretical results.

Key words. beam bending, elastic foundation models, Green's function, Galerkin procedure, finite elements, error analysis.

1. Introduction. A large number of models have been developed to describe the behavior of an elastic foundation under surface load. Those most widely used are discussed in [1]. The simplest model of a continuous elastic foundation has been provided by Winkler [2] who assumed that the reactive forces of the foundation carrying a loaded beam were proportional at every point to the deflection of the beam at the point of application of the load. This is equivalent to assuming that the foundation consists of closely spaced, uncoupled linear springs. In order to achieve some degree of interaction between the springs, Filonenko-Borodich [3] introduced a modification of the Winkler model in which the top ends of the springs were connected to a stretched elastic membrane subjected to a constant tension field. To the same end Pasternak [4] assumed the existence of shear interactions between the spring elements. Both models have the effect of introducing in the expression for the reactive force a term which is proportional at every point to the second derivative of the deflection of the beam. This resulting model, generally known as the Pasternak foundation, is an improvement of the Winkler model.

Analytical solutions [5] are available for the problem of a prismatic beam on a Pasternak type foundation for several loading conditions. We develop a finite element analysis for beam bending problems with elastic foundations. First we consider a beam of variable cross-section on a Pasternak foundation and then we examine a foundation model obtained via the Green's function for the elastic halfspace.

The equation of equilibrium for the beam on an elastic foundation can be written in normalized form as

$$(\kappa w'')'' + q = p \quad \text{in } I = (-L, L), \quad L < \infty, \tag{1.1}$$

with κ as the bending stiffness of the beam, and p as the applied load. In the Pasternak foundation (Fig. 1) the deflection w of the beam and the contact pressure q are related by

$$q = -tw'' + kw \text{ in } I, \qquad t, k \in \mathbb{R}. \tag{1.2}$$

We assume throughout that the beam is in perfect contact with the soil, and therefore, that the deflection w and the contact pressure q of the beam and of the soil coincide

* Received by the editors October 16, 1981, and in revised form June 15, 1982.

† Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

‡ Department of Mathematics, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213 and Technische Hochschule, Darmstadt, German Federal Republic.

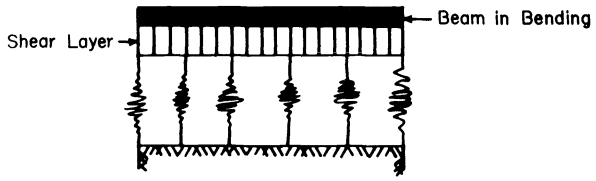


FIG. 1. *Beam on Pasternak-type foundation (Problem* I).

in I. Substituting (1.2) into (1.1) results in

$$(\kappa w'')'' - tw'' + kw = p \quad \text{in } I. \tag{1.3}$$

Here w satisfies the boundary conditions

$$((\kappa w'')' - tw')(\pm L) = 0 = w''(\pm L). \tag{1.4}$$

Thus we can pose:

Problem I. Given a force p, $\kappa(x) > 0$ and $t, k \in \mathbb{R}$ find the deflection w that satisfies (1.3)–(1.4).

Note. (i) The boundary conditions (1.4) reflect the fact that the total shear force and the total bending moment vanish at the end points $x = \pm L$. Here total denotes the sum of the effects on the soil and the beam.

(ii) Our variational formulation covers also the case of a concentrated load in I.

As a more realistic model (Fig. 2) we consider the problem in which the soil extends to infinity away from the beam. That is, instead of (1.2) there holds

$$q = -tw'' + kw \quad \text{in } (-\infty, \infty), \tag{1.5}$$

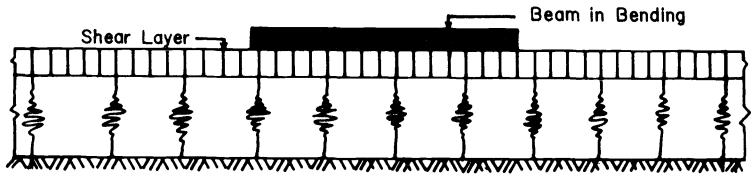


FIG. 2. *Beam on Pasternak foundation (Problem* II).

or, equivalently,

$$w(x) = \frac{\beta}{2k} \int_{-\infty}^{\infty} e^{-\beta|x-x'|} q(x')\, dx, \qquad x \in \mathbb{R}, \quad \beta^2 = \frac{k}{t}. \tag{1.6}$$

Since the soil surface is traction-free outside the beam, i.e., q vanishes in $\mathbb{R}\backslash\bar{I}$, (1.6) reduces to

$$w(x) = \frac{\beta}{2k} \int_{-L}^{L} e^{-\beta|x-x'|} q(x')\, dx', \qquad x \in \mathbb{R}. \tag{1.7}$$

Now (1.7), together with (1.3) and the boundary conditions

$$(\kappa w'')'(\pm L) = 0 = w''(\pm L), \tag{1.8}$$

yields the following problem for the deflection of the beam and the contact pressure in its interior.

Problem II. For given p, κ, t and k as above find (w, q) satisfying, in I,

$$(\kappa w'')'' + q = p, \tag{1.9}$$

$$w(x) = \frac{\beta}{2k} \int_{-L}^{L} e^{-\beta|x-x'|} q(x')\, dx' \equiv \int_{-L}^{L} g(|x-x'|) q(x')\, dx' = (Gq)(x)$$

with

$$(\kappa w'')'(\pm L) = 0 = w''(\pm L). \tag{1.10}$$

Note. (i) The boundary conditions (1.10) represent the vanishing of the shear force and bending moment at the end-points of the beam. Actually, we will see in § 3 that the contact pressure q includes concentrated forces at $x = \pm L$; this causes the shear force to be different from zero at both ends of the beam. Thus, if $Q^{\pm}$ is the concentrated soil reaction at $x = \pm L$, $(1.10)_1$ must be replaced by

$$(\kappa w'')'(\pm L) - Q^{\pm} = 0. \tag{1.11}$$

(ii) Once the solution (w, q) of Problem II has been found, the displacement w of the foundation in $\mathbb{R}\backslash\bar{I}$ can be obtained from (1.7).

The formulation (1.9), (1.10) can be used for studying the bending of beams supported on any other linear elastic foundation model by changing the Green's function in (1.9). As an example we consider the solution for an isotropic, homogeneous, elastic halfspace. The Green's function for points on the soil parallel to the beam axis is obtained by averaging the contact pressure across the width of the beam. Thus, by averaging Boussinesq's influence function for a point load, i.e., the Green's function for the elastic halfspace, we obtain [6]

$$\tilde{g}(|x-x'|) = \frac{\eta}{2\pi} \ln \frac{1+\sqrt{1+\eta|x-x'|}}{\eta|x-x'|}. \tag{1.12}$$

Here the dimensionless parameter $\eta \in \mathbb{R}$ measures the relative stiffness between the beam and the soil. The corresponding boundary-value problem is obtained by substituting $\tilde{g}$ for g in (1.9), and will be denoted as Problem III in the sequel.

The aim of this paper is to present a functional analytical setting for the foregoing beam bending problems and to derive an asymptotic error analysis for Galerkin procedures with finite elements. In § 2 we solve Problem I with the standard variational formulation. Problems II and III are treated in § 3 by using a modified Galerkin procedure incorporating the Green's function for the soil. Here we essentially use the calculus of pseudodifferential operators. In § 4 we give some numerical results and experimental error estimates in order to underscore our error analysis.

2. Standard Galerkin procedure for Problem I. A variational formulation for (1.3), (1.4) is obtained by multiplying (1.3) by a smooth test function $\bar{w}$ and then integrating the resulting equation by parts. Thus we are looking for $w \in H^2(I)$ such that

$$a(w, \bar{w}) = f(\bar{w}) \tag{2.1}$$

for all $\bar{w} \in H^2(I)$. Here $H^2(I)$ is the usual Sobolev space obtained by the closure of $C^\infty(I)$ in the norm

$$\|\bar{w}\|_2 = \sum_{j=0}^{2} \|D^j \bar{w}\|_0 \tag{2.2}$$

where $\|\cdot\|_0 = (\cdot,\cdot)_0^{1/2}$ denotes the L^2-norm induced by the L^2-scalar product

$$(v, \bar{v})_0 = \int_I v\bar{v}. \tag{2.3}$$

Due to the boundary condition (1.4) the bilinear form $a(\cdot,\cdot)$ in (2.1) reads, with (2.3),

$$a(w, \bar{w}) := (w'', \bar{w}'')_0 + t(w', \bar{w}')_0 + k(w, \bar{w})_0, \tag{2.4}$$

whereas for given p

$$f(\bar{w}) = (p, \bar{w})_0. \tag{2.5}$$

Obviously there exist constants $c, c', \gamma > 0$ such that for all $w, \bar{w} \in H^2(I)$

$$a(w, \bar{w}) \leqq c\|w\|_2\|\bar{w}\|_2, \tag{2.6}$$

$$|a(w, w)| \geqq \gamma\|w\|_2^2, \tag{2.7}$$

and

$$|f(\bar{w})| \leqq c'\|p\|_{-2}\|\bar{w}\|_2. \tag{2.8}$$

In (2.8) $\|\cdot\|_{-2}$ denotes the norm of the dual space $\tilde{H}^{-2}(I)$ of $H^2(I)$ (see [7]). Since the δ-distribution belongs to $\tilde{H}^{-1/2-\varepsilon}$ for any $\varepsilon > 0$, a point force p is also allowed in our formulation.

As usual (2.1) is solved approximately in a finite dimensional subspace $S_h^{t,k}$ of $H^2(I)$ [8], [9] with a conform Galerkin procedure, i.e., we are looking for $w^h \in S_h^{t,k} \subset H^2(I)$ such that

$$a(w_h, \bar{w}_h) = f(\bar{w}_h) \tag{2.9}$$

for all $\bar{w}_h \in S_h^{t,k}$.

Here h denotes the meshsize; $t-1$ is the degree of the piecewise polynomials and k describes the conformity, i.e. $S_h^{t,k} \subset H^k(I)$. It is well-known that the one-dimensional finite element spaces $S_h^{t,k}$ have the following properties:

Approximation property. For any $u \in H^r(I)$ there exists a $\tilde{u} \in S_h^{t,k}$ with $t \geqq r$ and a constant $c > 0$ independent of h and u such that for $q \leqq \min\{k, r\}$

$$\|u - \tilde{u}\|_{H^q(I)} \leqq ch^{r-q}\|u\|_{H^q(I)}. \tag{2.10}$$

Inverse assumption. For $q \leqq r \leqq k$ there exists a constant $M > 0$ independent of h such that for all $\tilde{u} \in S_h^{t,k}$

$$\|\tilde{u}\|_{H^r(I)} \leqq Mh^{q-r}\|\tilde{u}\|_{H^q(I)}. \tag{2.11}$$

Due to (2.6)–(2.8) the *Lax–Milgram* lemma gives unique solvability of the variational problem (2.1). Therefore by conformity of the method the Galerkin procedure converges with quasi-optimal order.

THEOREM 1. (i) *For given $p \in \tilde{H}^{-2}(I)$ there exists exactly one solution $w \in H^2(I)$ of the variational problem* (2.1).

(ii) *For any h, $0 < h \leqq h_0$, the finite system of algebraic equations* (2.9) *is uniquely solvable and Cea's lemma*:

$$\|w - w_h\|_2 \leqq C \inf_{\chi \in S_h^{t,k}} \|w - \chi\|_2 \tag{2.12}$$

holds.

(iii) *Furthermore with* (2.10) *the error estimate* (2.12) *implies*

$$\|w - w_h\|_2 \leqq ch^{\nu}\|w\|_s, \qquad \nu = \min\{t-2, s-2\} \tag{2.13}$$

By the standard arguments for elliptic b.v.p.'s [10] the solution w of (1.3), (1.4) satisfies the a priori estimate

$$\|w\|_{H^{s+4}(I)} \leqq c\|p\|_{H^s(I)}. \tag{2.14}$$

Therefore for a given point force $p = \delta \in H^{-1/2-\varepsilon}(I)$ we have $w \in H^{7/2-\varepsilon}(I)$ for any $\varepsilon > 0$. Thus from (2.13) with piecewise cubics in $S_h^{4,2}$ there holds the energy norm estimate, in agreement with [9, p. 347],

$$\|w - w_h\|_2 \leqq ch^{3/2-\varepsilon}\|w\|_{7/2-\varepsilon} \leqq c'h^{3/2-\varepsilon}\|\delta\|_{-1/2-\varepsilon}. \tag{2.15}$$

A simple application of Sobolev's inequality [11] with $\delta \geqq 1$, $|\gamma| \leqq 1$, $C > 0$,

$$|D^{\gamma}u(x)| \leqq C\delta^{-(r-1/2-|\gamma|)}(\|u\|_r + \delta\|u\|_0), \tag{2.16}$$

yields, for $u := w - w_h$ and $\delta = h^{-1}$, the pointwise estimate in the sup-norm

$$\|w - w_h\|_{\infty} = O(h^{3-\varepsilon}),$$

if together with (2.13) an L^2-error estimate is used. The L_2-estimate could be obtained via the Aubin–Nitsche trick. We omit the details and focus on superconvergence in the nodal points $z \in I$. This corresponds to [9, Thm. 2.4] for second order problems. Namely, there exists a $c > 0$ such that

$$|(w - w_h)(z)| \leqq ch^{2\nu}\|\delta\|_{-1/2-\varepsilon}, \qquad \nu = \min\{t-2, \tfrac{3}{2}-\varepsilon\}. \tag{2.17}$$

The error estimate (2.17) is proved as follows. Let z denote an arbitrary point in I and consider the auxiliary distributional problem

$$a(g, \bar{w}) = \bar{w}(z)$$

for all $\bar{w} \in H^2(I)$, where g is the Green's function for the beam. Then the choice $\bar{w} = e = w - w_h$ yields

$$\begin{aligned} |e(z)| &= |a(g, e)| = |a(g-\chi, e)| \leqq M\|g-\chi\|_2\|e\|_2 \\ &\leqq ch^{\nu}\|\delta\|_{-1/2-\varepsilon} \inf_{\chi \in S_h^{t,k}} \|g-\chi\|_2. \end{aligned} \tag{2.18}$$

For a nodal point z we have

$$\inf_{\chi \in S_h^{t,k}} \|g-\chi\|_2 \leqq ch^{\nu}, \qquad \nu = \min\{t-2, \tfrac{3}{2}-\varepsilon\}. \tag{2.19}$$

Hence (2.18) together with (2.19) gives (2.17). Thus (2.17), showing order 3 for the error in w, implies with (1.2) an error in q and w'' of order 1, whereas our numerical results show further superconvergence

$$\|e_q\|_{\infty} = O(h^{2-\varepsilon}) \quad \text{and} \quad \|e_{w''}\|_{\infty} = O(h^{2-\varepsilon}).$$

3. Modified Galerkin procedure for Problems II and III. For Problem II a variational formulation is obtained by multiplying (1.1) by $\bar{w}$ and (1.7) by $\bar{q}$. The resulting bilinear form

$$b((w, q), (\bar{w}, \bar{q})) := (w'', \bar{w}'')_0 + (q, \bar{w})_0 - (w, \bar{q})_0 + (Gq, \bar{q})_0 \tag{3.1}$$

is obviously continuous on $H \times L^2$ where $H = \{w \in L^2 : w'' \in L^2\} \subset H^2(I)$. Thus the weak formulation of Problem II reads as follows: Find $(w, q) \in H \times L^2$ such that for all $(\bar{w}, \bar{q}) \in H \times L^2$ there holds

$$b((w, q), (\bar{w}, \bar{q})) = (p, \bar{w})_0 \tag{3.2}$$

for given p. We remark that by allowing q to include point forces at $x = \pm L$, e.g., $q \in H^{-1}$, (3.2) yields (1.9) together with the boundary conditions $(1.10)_2$ and (1.11). Moreover, these conditions are *natural*, which means that when we approximate, as below, with finite elements there are no boundary restrictions on these elements.

The coercivity of $b(\cdot, \cdot)$ hinges on the mapping properties of the Green's function g in (1.9) being a pseudo-differential operator of order-2.

LEMMA 3.1. *For any $r \in \mathbb{R}$ and $q \in \tilde{H}^r(I) = \{\psi \in H^r(\mathbb{R}) : \psi = 0$ in $\mathbb{R} \backslash \bar{I}\}$ there holds*

$$\|Gq\|^*_{r+2} \leqq c \|q\|^*_r \tag{3.3}$$

with $\|q\|^*_r = \inf \|q\|_{H^r(\mathbb{R})}$.

Proof. Taking the Fourier transform of

$$Gq(x) = \int_{\mathbb{R}} g(x - x') q(x')\, dx', \tag{3.4}$$

we have

$$\widehat{Gq}(\xi) = \hat{g}(\xi) \hat{q}(\xi)$$

with the Fourier transformed Green's function $\hat{g}$ from (1.9)

$$\hat{g}(\xi) = \frac{\beta^2/k}{\beta^2 + \xi^2}. \tag{3.5}$$

But (3.5) shows that

$$\hat{g}(\xi) \sim (1 + |\xi|^2)^{-1}.$$

This implies that G is a pseudo-differential operator (ψdo) of order-2 and therefore (3.3) holds. □

Now (3.5) also implies coercivity of the bilinear form $b(\cdot, \cdot)$ on $H \times \tilde{H}^{-1}$, in the form of a Gårding inequality:

LEMMA 3.2. *There exists a constant $\gamma > 0$ such that for all $(w, q) \in H \times \tilde{H}^{-1}$*

$$b((w, q), (w, q)) \geqq \gamma \{\|w''\|_{L^2(I)} + \|q\|^2_{\tilde{H}^{-1}(I)}\} - |k((w, q), (w, q))|, \tag{3.6}$$

where k is a compact bilinear form on $(H \times \tilde{H}^{-1})^2$.

Proof. With the principal symbol in (3.5) of G we have, via Parseval's equation,

$$\begin{aligned} (Gq, q)_0 = (\widehat{Gq}, \hat{q})_0 &= \int_{\mathbb{R}} (1 + |\xi|^2)^{-2} |q^*(\xi)|^2\, d\xi + \int_{\mathbb{R}} (1 + |\xi|^2)^{-2-\varepsilon} |\hat{q}^*(\varepsilon)|^2\, d\xi \\ &\geqq \gamma \{\|q^*\|^2_{\tilde{H}^{-1}(\mathbb{R})} - c \|Rq^*\|_{\tilde{H}^{1+\varepsilon}} \|q^*\|_{\tilde{H}^{-1}}\}, \end{aligned} \tag{3.7}$$

where

$$q^* = \begin{cases} q & \text{in } I, \\ 0 & \text{in } \mathbb{R} \backslash \bar{I}. \end{cases}$$

Here we have used the Cauchy–Schwarz inequality and the fact that R is a ψdo of order $-(2+\varepsilon)(\varepsilon > 0)$. The last operator yields a compact bilinear form $k((w, q), (w, q)) = (Rq, q)_0$.

On the other hand the choice $w = \bar{w}, q = \bar{q}$ reduces (3.1) to

$$b((w, q), (w, q)) = \|w''\|_0^2 + (Gq, q)_0. \tag{3.8}$$

Hence (3.8) together with (3.7) gives the Gårding inequality (3.6). □

Now our modified Galerkin procedure corresponding to the weak formulation (3.2) of Problem II reads as: Find $(w_h, q_h) \in S_h = S_h^{t,k} \times S_h^{l,m} \subset H \times \tilde{H}^{-1}$ such that for all $(v_h, \psi_h) \in S_h$ there holds

$$b((w_h, q_h), (v_h, \psi_h)) = (p, v_h)_0 = b((w, q), (v_h, q_h)). \tag{3.9}$$

Actually for our numerical results, we use an equidistant mesh in I.

Now with the results in [12], [13] the Gårding inequality (3.6) together with uniqueness of (3.2) implies convergence of the Galerkin procedure (3.9).

LEMMA 3.3. *There exists a mesh width $h_0 > 0$ such that the Galerkin equations (3.9) are uniquely solvable for any h, $0 < h \leqq h_0$. For decreasing mesh size $h \to 0$ we have the asymptotic error estimate*

$$\|w'' - w_h''\|_{L^2(I)} + \|q - q_h\|_{\tilde{H}^{-1}(I)} \leqq c\{h^{\mu_1}\|w\|_{H^s(I)} + h^{\mu_2}\|q\|_{\tilde{H}^r(I)}\} \tag{3.10}$$

where $\mu_1 = \min\{t-2, s-2\}$, $\mu_2 = \min\{l+3, r+1\}$ and $c > 0$ independent of h.

Proof. The Gårding inequality (3.6) implies, together with the results in [13], the uniform boundedness of the Galerkin operator G_b: $(w, q) \mapsto (w_h, q_h)$ defined by (3.9), i.e., there exists a constant $c > 0$ such that for all $0 < h \leqq h_0$

$$\|G_b(w, q)\|_{H \times \tilde{H}^{-1}} \leqq c\|(w, q)\|_{H \times \tilde{H}^{-1}}.$$

Since, for an arbitrary pair $(\chi_h, \varphi_h) \in S_h$,

$$G_b(\chi_h, \varphi_h) = (\chi_h, \varphi_h),$$

we obtain (3.10) by triangle inequality, namely:

$$\begin{aligned}\|(w, q) - (w_h, q_h)\| &\leqq \|(w, q) - (\chi_h, \varphi_h)\| + \|G_b\{(\chi_h, \varphi_h) - (w, q)\}\| \\ &\leqq (1 + \|\|G_b\|\|)\{\|(w, q) - (\chi_h, \varphi_h)\|,\end{aligned}$$

where $\|(w, q) - (w_h, q_h)\| := \|D^2(w - w_h)\|_0 + \|q - q_h\|_{\tilde{H}^{-1}}$ and $\|\|G_b\|\|$ denotes the operator-norm. Finally, the approximation property (2.10) yields (3.10). □

Remark. For a point force $p = \delta$ at $x = 0$ $w \in H^{7/2-\varepsilon}(I)$ due to (1.9), since G is a ψdo of order-2. Hence (3.10) yields

$$\|w'' - w_h''\|_0 + \|q - q_h\|_{-1} \leqq \{h^{3/2-\varepsilon}\|w\|_{7/2-\varepsilon} + h^{5/2-\varepsilon}\|q\|_{3/2-\varepsilon}\}. \tag{3.11}$$

Furthermore since $q_h \in S_h^{l,m}$ has the inverse property (2.11) and

$$G_b(0, \varphi_h) = (0, \varphi_h)$$

for any $\varphi_h \in S_h^{l,m}$ there holds, in analogy to (3.11),

$$\begin{aligned}\|q - q_h\|_0 = \|q - q_h + \varphi_h - \varphi_h\|_0 &\leqq \|q - \varphi_h\|_0 + \|G_b(0, q - \varphi_h)\|_0 \\ &\leqq ch^{3/2-\varepsilon}\|q\|_{3/2-\varepsilon} + ch^{-1}\|G_b(0, q - \varphi_h)\|_{-1}.\end{aligned}$$

The last term on the right-hand side is bounded by $c'\|q - \varphi_h\|_{-1}$ by the stability of the Galerkin operator. Hence, from the preceding inequality together with the convergence property of our finite elements we obtain

$$\|q - q_h\|_0 \leqq ch^{3/2-\varepsilon}\|q\|_{3/2-\varepsilon}. \tag{3.12}$$

Therefore with the Sobolev inequality (2.16) from (3.11) and (3.12) there follows

$$\|w''-w_h''\|_\infty \leqq ch^{1-\varepsilon}\|w\|_{7/2-\varepsilon}, \qquad \|w-w_h\|_\infty \leqq C'h^{3-\varepsilon} \tag{3.13}$$

and

$$\|q-q_h\|_\infty \leqq ch^{1-\varepsilon}\|q\|_{3/2-\varepsilon}. \tag{3.14}$$

As for Problem I in this case our numerical results again show even superconvergence for w'' and q, namely

$$\|w''-w_h''\|_\infty = O(h^{2-\varepsilon}), \qquad \|q-q_h\|_\infty = O(h^{2-\varepsilon}). \tag{3.15}$$

Actually the solution w of Problem II decreases exponentially with the length L of the beam (see Hetenyi [5, p. 129]). Thus for a long beam the influence of the boundary conditions at the endpoints $x = \pm L$ is negligible for w near $x = 0$. This is also illustrated by our numerical experiments (Fig. 3).

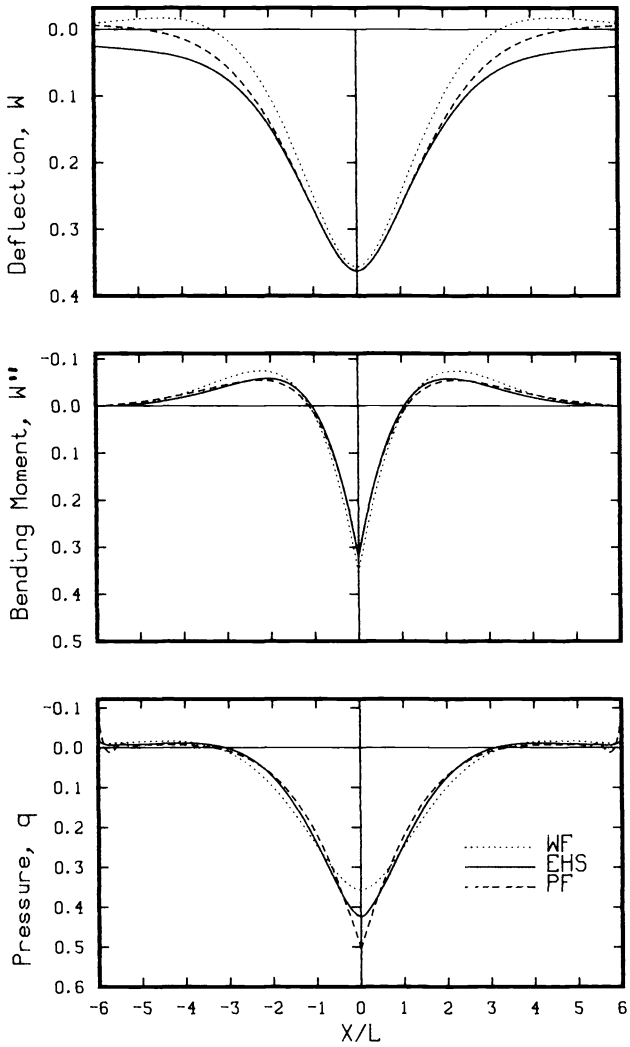


FIG 3. *Effect of point load on beam on* (i) *Winkler foundation* (WF), $k = 1.094$; (ii) *elastic halfspace* (EHS), $\eta = 2$; (iii) *Pasternak foundation* (PF), $t = 0.706$, $k = 0.864$.

A comparison of the theoretical error estimates and the numerical results for Problems I and II shows that the two formulations lead asymptotically to the same results—even the slopes in the figures give better constants for w and q than obtained via the modified method whereas numbers coincide for w''. On the other hand the modified method using the Green's function in (1.9) allows one to treat even halfspace problems whereas in a standard method like Problem I unbounded regions must be considered. Thus, incorporating the Green's function (1.12) for the halfspace we have analogously to (3.1), the weak formulation (3.2) with the new bilinear form

$$c((w,q),(w,\bar{q})) := (w'',w'')_0+(q,\bar{w})_0-(w,\bar{q})_0+(\tilde{G}q,q)_0. \tag{3.16}$$

The kernel (1.12) of $\tilde{G}$ has the decomposition

$$\tilde{g}(|x-x'|) := \ln\left\{\frac{1+\sqrt{1+\eta|x-x'|^2}}{|x-x'|}\right\} = -\ln|x-x'|+k(x,x') \tag{3.17}$$

with a smooth remainder $k(x,x')$. Taking Fourier transforms yields [12]

$$\widehat{-\ln|x-x'|} = |\xi|^{-1} \tag{3.18}$$

and thus $\tilde{G}$ is a pseudo-differential operator of order-1. $\tilde{G}$ is well defined by (3.4) with (3.17) since $q=0$ in $\mathbb{R}\backslash\bar{I}$.

Analogously to Lemma 3.2 for $c(\cdot,\cdot)$ there also holds a Gårding inequality on $H\times\tilde{H}^{-1/2}$.

LEMMA 3.4. *There exists a constant $\gamma>0$ such that for all $(w,q)\in H\times\tilde{H}^{-1/2}$*

$$c((w,q),(w,q)) \geqq \gamma\{\|w''\|^2_{L^2(I)}+\|q\|^2_{\tilde{H}^{-1/2}(I)}\}-|\tilde{k}((w,q),(w,q))|, \tag{3.19}$$

where k is a compact bilinear form on $(H\times\tilde{H}^{-1/2})^2$.

Furthermore since $c(\cdot,\cdot)$ is continuous on $H\times\tilde{H}^{-1/2}(I)$ Gårding's inequality (3.19) together with the assumed uniqueness of Problem III yields the convergence of a Galerkin procedure analogous to (3.9) and again quasi-optimal error estimates.

LEMMA 3.5. *For any $h, 0<h\leqq h_0$, the Galerkin equations (3.9) with $b(\cdot,\cdot)$ replaced by $c(\cdot,\cdot)$ have a unique solution $(w_h,q_h)\in S_h\subset H\times\tilde{H}^{-1/2}(I)$. Furthermore there holds*

$$\begin{aligned}\|w''-w_h''\|_0+\|q-q_h\|_{-1/2} &\leqq c\{h^{3/2-\varepsilon}\|w\|_{7/2-\varepsilon}+h^{7/2-\varepsilon}\|q\|_{5/2-\varepsilon}\}\\ &\leqq c' h^{3/2-\varepsilon}\|\delta\|_{-1/2-\varepsilon}.\end{aligned} \tag{3.20}$$

Remark. The estimate (3.20) differs from (3.10) since now $\tilde{G}$ being a pseudo-differential operator of order-1 yields $q\in H^{5/2-\varepsilon}$ for given $w=\tilde{G}q\in H^{7/2-\varepsilon}$.

4. Numerical experiments. The numerical results reported in this section are computed for prismatic beams ($\kappa=1$) of half-length $L=6$ and 12, and for a unit point load applied at the center of the beam.

Corresponding to Problem I the space $S_h^{4,2}$ consists of piecewise cubic functions. As usual the formulation (2.9) is equivalent to an algebraic system of linear equations. With a basis $\mu_1,\cdots,\mu_N$ in $S_h^{t,k}$ we have

$$w_h=\sum_{k=1}^{N}\alpha_k\mu_k \tag{4.1}$$

for unknown weights $\alpha_1,\cdots,\alpha_N$. The latter are determined by (2.9), i.e.,

$$\sum_{k=1}^{N}\alpha_k a(\mu_k,\mu_j)=(p,\mu_j)_0$$

for $k = 1, \cdots, N$, or equivalently by

$$\mathbf{K}\boldsymbol{\gamma} = \mathbf{p}. \tag{4.2}$$

Here the (j, k)-entry of the stiffness matrix $\mathbf{K}$ is given by

$$a(\mu_k, \mu_j) = (\mu_k'', \mu_j'')_0 + t(\mu_k', \mu_j')_0 + k(\mu_k, \mu_j)_0.$$

For Problem II the space S_h consists of piecewise cubic functions w_h, and piecewise linear functions q_h. With a basis $\nu_1, \cdots, \nu_M$ we have

$$q_h = \sum_{k=1}^{M} \gamma_k \nu_k \tag{4.3}$$

with unknown weights $\gamma_1, \cdots, \gamma_M$. The coefficients α_k in (4.1) and γ_m in (4.3) are obtained from the Galerkin equations (3.9), i.e., for $l = 1, \cdots, N$ and $n = 1, \cdots, M$

$$\sum_{k=1}^{N} \alpha_k \{(\mu_k'', \mu_l'')_0 - (\mu_k, \nu_n)_0\} + \sum_{m=1}^{M} \gamma_m \{(\nu_m, \mu_l)_0 + (G\nu_m, \nu_n)_0\} = (p, \mu_l)_0$$

or, in matrix notation

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{A} \\ -\mathbf{A}^T & \mathbf{F} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{Bmatrix} = \begin{Bmatrix} \mathbf{p} \\ \mathbf{0} \end{Bmatrix}. \tag{4.4}$$

Another system similar to (4.4) is obtained for Problem III:

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{A} \\ -\mathbf{A}^T & \tilde{\mathbf{F}} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{Bmatrix} = \begin{Bmatrix} \mathbf{p} \\ \mathbf{0} \end{Bmatrix}. \tag{4.5}$$

Here the (m, n)-entry $\tilde{F}_{mn}$ of $\tilde{\mathbf{F}}$ is given by

$$\tilde{F}_{mn} = (\tilde{G}\nu_m, \nu_n)_0.$$

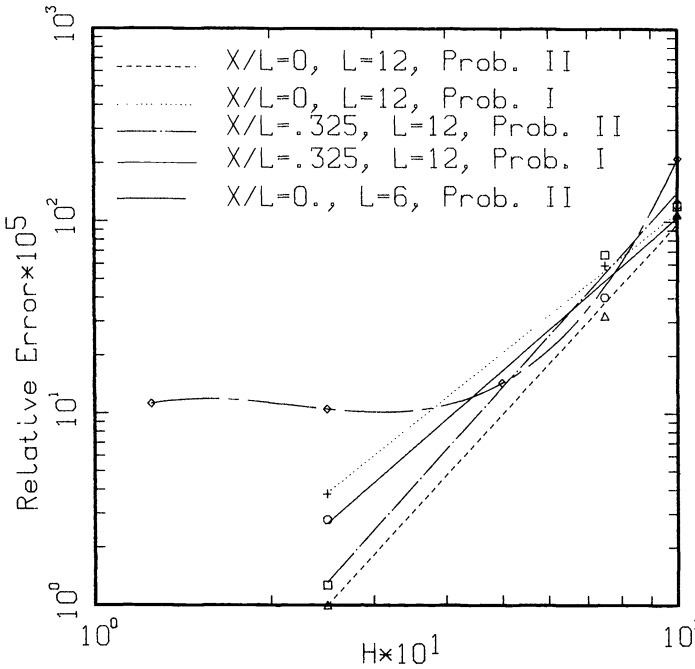


FIG. 4. *Relative error for displacement, w, at two points on beam on Pasternak foundations.*

The entries of the matrices $\mathbf{K}$, $\mathbf{K}_1$ and $\mathbf{A}$ are obtained explicitly from [14] in terms of the mesh-width h and the system parameters t and k, but those of $\mathbf{F}$ and $\tilde{\mathbf{F}}$ are evaluated by Gaussian quadrature. For $\mathbf{F}$ we integrate numerically with standard Gauss–Legendre formulas. Due to the logarithmic singularity of $\tilde{g}$ we use mesh refinement to compute $\tilde{F}_{mn}$ with the same quadrature formulas.

Equation (4.5) for the elastic halfspace is solved for $L = 6$ and $\eta = 2$ (in (1.12). The corresponding values of the deflection w, bending moment w'' and contact pressure q are represented by solid lines in Fig. 3. Also in Fig. 3 there are given the solution of (4.4) for the Pasternak foundation ($t = 0.706$, $k = 0.864$) and the solution of (4.2) for the limiting case of a Winkler foundation ($t = 0$, $k = 1.094$). A discussion of the above choice for the parameters t, k is given in [6]. Our numerical approximation for the pressure under Pasternak foundation reflects the delta-function behavior at the endpoints of the beam. The numerical experiments for Problem III show a similar effect (Fig. 3).

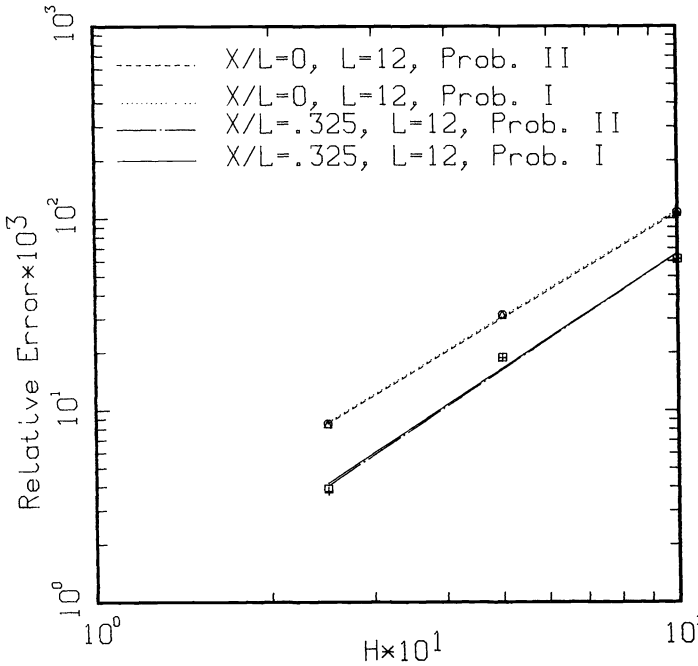


FIG. 5. *Relative error for bending moment, w'', at two points of beam on Pasternak foundations* ($L = 12$).

For a long beam ($L = 12$) in Figs. 4, 5 and 6 we plot against the mesh size the relative point errors for w, w'' and q, respectively, at the nodal point $x/L = 0$ and the interior point $x/L = 0.325$. These relative errors are obtained by comparing the numerical solutions of (4.2) and (4.4) with the analytical solution of Problem I for a finite free-end beam given by [5, (104), (105)]. The concentrated load is applied at the nodal point $x/L = 0$.

The numerical approximations for Problems I and II show essentially the same order of convergence. Note that in both cases w converges to the exact solution with the predicted rate in (2.17), (3.13) whereas w'' and q converge faster than predicted by the theoretical error estimates (super-convergence). These results are summarized in Table 1.

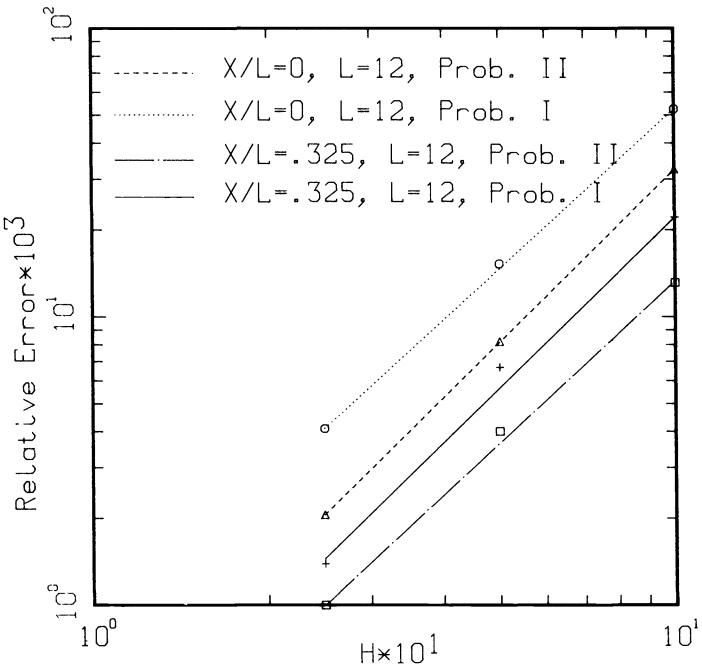


FIG. 6. *Relative error for pressure, q, at two points of beam on Pasternak foundation* ($L = 12$).

Figure 4 also shows the relative error in the displacement w corresponding to a shorter beam with a half-length $L = 6$. Notice that for this beam the rate of convergence decreases significantly for values of $h \leqq \frac{1}{2}$, as a consequence of the pollution from the singularities at the endpoints. This deterioration might be prevented either by introducing special singularity functions into the Galerkin procedure or by using mesh refinement.

TABLE 1
Relative errors for w, w'', q ($t = 0.856$, $k = 0.839$)

Abscissa		$x/L = 0$					
Relative error for		w		w''		q	
Problem		I	II	I	II	I	II
Relative error	numerical	2.8	3.4	1.9	1.9	1.9	2
	theoretical	$3-\varepsilon$	$3-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$
Abscissa		$x/L = 0.325$					
Relative error for		w		w''		q	
Problem		I	II	I	II	I	II
Relative error	numerical	2.7	3.3	2.1	2.1	1.9	2
	theoretical	$3-\varepsilon$	$3-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$	$1-\varepsilon$

Acknowledgment. We thank D. Sriram for performing the numerical computations.

REFERENCES

[1] A. D. KERR, *Elastic and viscoelastic foundation models*, J. Appl. Mech., 31 (1964), pp. 491–498.
[2] E. WINKLER, *Die Lehre von der Elasticitaet und Festigkeit*, Dominicus, Prag, 1867.
[3] M. M. FILONENKO-BORODICH, *Some approximate theories of the elastic foundation*, Uchenyie Zapiski Moskovskogo Gosudarstvennogo Universiteta Mekhanica, no. 46, (1940), pp. 3–18 (in Russian).
[4] P. L. PASTERNAK, *On a new method of analysis of an elastic foundation by means of two foundation constants*, Gosudarstvennoe Izdatelstvo Literaturi po stroitelstvu i Arkhitekture, Moscow, 1964 (in Russian).
[5] M. HETENYI, *Beams on Elastic Foundations*, Univ. Michigan Press, Ann Arbor, 1964.
[6] J. BIELAK AND D. SRIRAM, *Beams on elastic foundations revisited: a unified approach*, in preparation.
[7] J. L. LIONS AND E. MAGENES, *Non-homogeneous Boundary Value Problems and Applications* I, Springer-Verlag, Berlin-Heidelberg-New York, 1972.
[8] I. BABUSKA AND A. K. AZIZ, *Survey lectures on the mathematical foundations of the finite element method*, The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Academic Press, New York, 1972, pp. 5–359.
[9] J. T. ODEN AND J. N. REDDY, *An Introduction to the Mathematical Theory of Finite Elements*, John Wiley, New York-London-Sidney-Toronto, 1976.
[10] S. AGMON, *Lectures on Elliptic Boundary Value Problems*, Van Nostrand, Princeton, NJ, 1965.
[11] G. FAIRWEATHER, *Finite Element Galerkin Methods for Differential Equations*, Lecture Notes in Pure and Applied Mathematics 34, Marcel Dekker, Inc., New York-Basel, 1978.
[12] E. STEPHAN AND W. WENDLAND, *Remarks to Galerkin and least squares methods with finite elements for general elliptic problems*, Ordinary Partial Differential Equation, Proceedings 4th Dundee Conference, Lecture Notes in Mathematics 564, Springer-Verlag, New York, 1976, pp. 461–471.
[13] ST. HILDEBRANDT AND E. WEINHOLTZ, *Constructive proofs of representation theorems in separable Hilbert space*, Comm. Pure Appl. Math., 17 (1964), pp. 369–373.
[14] G. STRANG AND G. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 2, June 1983

0196-5204/83/0402-0019 $01.25/0

COMMENT: DISTRIBUTION OF QUADRATIC FORMS IN NORMAL RANDOM VARIABLES— EVALUATION BY NUMERICAL INTEGRATION*

CARL W. HELSTROM†

Abstract. Rice's method [SIAM J. Sci. Stat. Comput., 1 (1980), pp. 438–448] of calculating the cumulative distribution of a random variable by numerically integrating the inversion integral for the characteristic function, described in the paper named in the title, is modified by a different choice of the saddlepoint through which the path of integration passes, a modification that renders the method efficient for values of the variable not only in the tails of the distribution, as in Rice's paper, but even in the neighborhood of its mean. It is shown how the method can also be applied to calculating the cumulative distribution of a positive-integer-valued random variable from its probability generating function.

Key words. probability distribution, quadratic form, numerical integration

In [1] S. O. Rice treats the calculation of the complementary cumulative distribution $Q(y)$ of a quadratic form $\mathbf{y}$ in normal random variables by numerical integration of what is essentially the inversion integral for its Laplace transform,

$$(1) \qquad Q(y) = \Pr(\mathbf{y} > y) = \int_{c-i\infty}^{c+i\infty} \frac{e^{-uy+\phi(u)}}{2\pi i u}\, du = \int_{c-i\infty}^{c+i\infty} \frac{e^{\psi(u)}}{2\pi i}\, du, \qquad c > 0,$$

where $\phi(u)$ is the cumulant generating function [1, (8)] and

$$(2) \qquad \psi(u) = \phi(u) - uy - \ln u,$$

with $\arg(\ln u) = 0$ at $u = c$. He recommends taking the path of integration through the saddlepoint u_1 of $\exp[-uy + \phi(u)]$ when y is in one or the other tail of its distribution, but remarks in § 3 that when $Q(y)$ is not near 0 or 1, one may as well take the path along the imaginary axis, $\operatorname{Im} u = 0$. On the imaginary axis, however, the integration passes through a singularity at $u = 0$, which requires special treatment (Davies [2]).

The advantages of Rice's method can be preserved for all values of y if one takes the path instead through a saddlepoint u_0 of the *entire* integrand of (1). The saddlepoints are the roots of

$$(3) \qquad \psi'(u) = \phi'(u) - y - u^{-1} = 0,$$

the prime indicating differentiation. They can be quickly calculated by Newton's method starting from a trial value of u,

$$u \leftarrow u - \frac{\psi'(u)}{\psi''(u)}.$$

There are two principal saddlepoints, one to the left, the other to the right of the origin, each lying between the origin and the nearest singularity of the cumulant generating function $\phi(u)$. When $y \geqq E(\mathbf{y})$, the saddlepoint lying to the right of the origin, $u_0 > 0$, should be used; when $y \leqq E(\mathbf{y})$, use the saddlepoint to the left of the origin, $u_0 < 0$, and as in [1, (14)], add 1 to the result of the numerical integration.

* Received by the editors October 30, 1981.

† Department of Electrical Engineering & Computer Sciences, University of California at San Diego, La Jolla, California 92093.

When y is at or close to the mean $E(\mathbf{y})$, numerical integration along a path through one saddlepoint is just as efficient as along a path through the other.

This modified saddlepoint method was applied to the example of [1, § 4]. The path was taken parallel to the imaginary axis, so that as in [1, (14)] $u = u_0 + ibv$ with

$$b = \left[\frac{2}{\psi''(u_0)}\right]^{1/2}.$$

That form of the trapezoidal rule was used in which the integral of $f(v)$ from $v = 0$ to ∞ is approximated by $h[\frac{1}{2}f(0) + f(h) + \cdots]$, where h is the spacing and

$$f(v) = \operatorname{Re} F(v), \quad F(v) = \left(\frac{b}{\pi}\right) u^{-1} e^{-uy+\phi(u)}, \quad u = u_0 + ibv.$$

The summation $S = \frac{1}{2}f(0) + \sum_k f(kh)$ was stopped when the ratio $|F(v)|/|S|$ passed below 10^{-8}. Table 1 shows the convergence of the values of $Q(y)$ as the spacing h is progressively halved; N is the number of values at which the integrand had to be calculated. In this example $E(\mathbf{y}) = 120$.

TABLE 1

y	u_0	h	N	$Q(y)$
90	−0.08184771	0.5	18	0.85707656
		0.25	33	0.85707669
		0.125	62	0.85707669
120	0.03044865	0.5	21	0.46524863
		0.25	38	0.46524724
		0.125	71	0.46524724
150	0.04307360	1.0	11	0.14783405
		0.5	19	0.14764090
		0.25	34	0.14764089

In [3] the crude approximation

$$Q(y) \cong \begin{Bmatrix} 0, & u_0 > 0 \\ 1, & u_0 < 0 \end{Bmatrix} + [2\pi\psi''(u_0)]^{-1/2} \exp \psi(u_0)$$

to the complementary cumulative probability was proposed, with u_0 determined by (3). It follows from approximating the integrand in (1) as

$$\exp \psi(u) \cong \exp [\psi(u_0) + \tfrac{1}{2}\psi''(u_0)(u - u_0)^2].$$

Often simple to calculate, it is the more accurate, the closer $Q(y)$ lies to 0 or 1.

The method of saddlepoint integration can also be applied to calculating the complementary cumulative distribution of a positive-integer-valued random variable $\boldsymbol{n}$,

$$Q_n = \Pr(\boldsymbol{n} \geqq n) = \sum_{k=n}^{\infty} p_k, \quad p_k = \Pr(\boldsymbol{n} = k), \quad Q_0 = 1.$$

In terms of the probability generating function

$$h(z)=\sum_{k=0}^{\infty} p_k z^k,$$

$$Q_n=\begin{Bmatrix}0, & r>1\\ 1, & r<1\end{Bmatrix}+\int_{C_r}\frac{z^{-n}h(z)}{z-1}\frac{dz}{2\pi i}, \tag{4}$$

where C_r is a circle of radius r centered at the origin [3]. By putting $z=re^{i\theta}$ this can be written

$$Q_n=\begin{Bmatrix}0, & r>1\\ 1, & r<1\end{Bmatrix}+\pi^{-1}\,\mathrm{Re}\int_0^{\pi}\frac{r^{1-n}h(re^{i\theta})\,e^{i(1-n)\theta}}{re^{i\theta}-1}\,d\theta. \tag{5}$$

The circle C_r is made to pass through a saddlepoint of the integrand

$$\exp\psi(z)=z^{-n}\frac{h(z)}{z-1},$$

that is, through a root of

$$\psi'(r)=\frac{d}{dr}[\ln h(r)]-\frac{n}{r}-\frac{1}{r-1}=0.$$

Again there are two principal saddlepoints; they lie on the positive real axis, one to the left of $z=1$, the other to the right. The left-hand one, $r<1$, is used for $n\leqq E(\boldsymbol{n})$, the right-hand one, $r>1$, for $n\geqq E(\boldsymbol{n})$. The integral is evaluated by the trapezoidal rule as before, taking steps of size

$$\Delta\theta=\varepsilon[2/\psi''(r)]^{1/2}\text{ radians},\qquad \varepsilon=1, 0.5, 0.25, \cdots,$$

and starting at $\theta=0$. The integration can be stopped by the criterion given above; it is seldom necessary to take it all the way to $\theta=\pi$.

The method has been applied to evaluating the probabilities Q_n for the Laguerre distribution,

$$p_k=(1-v)^M v^k\exp[-(1-v)S]L_k^{M-1}(-x),\qquad x=\frac{(1-v)^2S}{v}, \tag{6}$$

where $L_k^{M-1}(\cdot)$ is the associated Laguerre polynomial. This distribution characterizes the number of photo-electrons emitted during a fixed interval when coherent laser light and incoherent background light fall simultaneously on an emissive surface. Summing the probabilities in (6) directly becomes tedious when S, M and n are large, and double precision may be required in order to avoid overflow or underflow [3]. Saddlepoint integration of (5) avoids these difficulties.

The mean value of the random variable $\boldsymbol{n}$ is $E(\boldsymbol{n})=S+Mv/(1-v)$, and the probability generating function is

$$h(z)=\left(\frac{1-v}{1-vz}\right)^M\exp\left[\frac{(1-v)(z-1)S}{1-vz}\right].$$

Table 2 shows the results of saddlepoint integration for $M=20$, $S=40$, $v=0.2$; N is the number of values of θ at which the integrand had to be evaluated.

TABLE 2

n	r	$\Delta\theta$	N	Q_n
20	0.520280	1.49600 (−1)	10	1−1.54282221 (−4)
		7.47998 (−2)	20	1−1.54282220 (−4)
45	0.881655	1.16355 (−1)	8	0.508049090
		5.81776 (−2)	15	0.509161169
		2.90888 (−2)	29	0.509162407
		1.45444 (−2)	57	0.509162410
70	1.366139	1.65347 (−1)	4	2.91437293 (−3)
		8.26735 (−2)	7	2.87700340 (−3)
		4.13367 (−2)	14	2.87700295 (−3)
		2.06684 (−2)	27	2.87700294 (−3)

REFERENCES

[1] S. O. RICE, *Distribution of quadratic forms in normal random variables—evaluation by numerical integration*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 438–448.

[2] R. B. DAVIES, *Numerical inversion of a characteristic function*, Biometrika, 60 (1973), pp. 415–417.

[3] C. W. HELSTROM, *Approximate evaluation of detection probabilities in radar and optical communications*, IEEE Trans. Aerospace Electro. Systems, AES 14 (1978), pp. 630–640.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0001/$01.25/0

MODELING OF THE SILICON INTEGRATED-CIRCUIT DESIGN AND MANUFACTURING PROCESS*

ROBERT W. DUTTON†

Abstract. The evolution of process modeling is traced starting with bipolar technology in the 1960's through recent processing concerns for oxide-isolated MOS devices. The kinetics of diffusion and oxidation are used to illustrate both physical and numerical effects. The interaction of device effects with process modeling is discussed as well as the statistical implications of process variables. The nature of computer-aided design tools for process and device modeling are discussed. This includes tools that bridge gaps between technology and system design with potential application for manufacturing.

I. Introduction. The fabrication and manufacturing of integrated silicon circuits has spawned a revolution equal in impact to that of the industrial revolution. This so-called information revolution differs substantially from the industrial revolution in its exploitation of computer-based technology in contrast to a technology focused primarily at mechanical advantage over the environment. On the other hand, the technology bases—steel and silicon—both require a sophisticated set of capital-intensive manufacturing techniques. The arsenal of silicon technology equipment in fact has a growing connection to metallurgy since the circuit design constraints which drive the technology are increasingly limited by interconnections—small metal lines; hence laser, ion, and other milling tools are now commonly used. However, the fact that active electronic elements are the primitive atoms of IC technology is the key driving force which separates this manufacturing endeavor from most other industrial technologies. It is the close interplay of the design of active transistor elements with the manufacturing technology which is the subject of this article.

Organization of the discussion which follows is intended both to chronicle the history and chart the future for IC manufacturing technology. The notion of process modeling will be introduced by means of examples related to bipolar transistor manufacturing. Next the features of MOS technology and the new concerns involved in modeling and manufacturing will be presented. Moving to the current issues of IC manufacturing technology, a discussion of state-of-the-art process kinetics will be given. Finally, the issues of CAD tools for both IC design and manufacturing will be summarized.

II. Bipolar technology and process modeling. The bipolar device technology dominated the decade of the 1960's while MOS technology was struggling with isolation and threshold control issues. Moreover, during this MSI era the off-chip drive capabilities of bipolar devices provided essential system leverage. From a production point of view the double-diffused technology dominated the high-speed market, with cut-off frequencies approaching 1 GHz. The desire to increase cut-off frequencies by shrinking base width led to a growing interest in emitter and base impurity profiles. During this period process models were developed in an effort to predict the process dependencies of double-diffused profiles [1]. Of special interest was the problem of push-out of the boron base impurity by heavily doped phosphorus emitters [2].

*Received by the editors May 4, 1983, and in revised form July 12, 1983. This research was supported in part by the U. S. Army Research Office under grant DAAG-29-80-K-0013, and by DARPA under contracts MDA903-80-C-0432 and MDA903-79-C-0257.

†Integrated Circuits Laboratory, Stanford University, Stanford, California 94305.

Although empirical models were developed and used [3], the dominant limitation of the modeling art arose from the lack of an adequate model for high-concentration coupled-species diffusion.

Fig. 1 shows the cross section of a typical double-diffused bipolar device along with one-dimensional impurity profiles in the emitter and base regions. A number of features of the device are apparent from the Fig. 1(b) and (c). First, the base-collector junction depth X_{BC} is different for the two regions. This indicates the fact that the boron diffusion is affected by the high-concentration phosphorus emitter. Second, the phosphorus profile exhibits a so-called "kink" in the high-concentration region of the profile. The implication of this kink on the resulting junction depths X_{BE} and X_{BC} as well as the base width X_B is of major importance in controlling electrical parameters of bipolar devices. Hence, the objective of process modeling is to provide basic understanding of the phenomena as well as engineering tools to assist in the design and process control during manufacturing. In the next few subsections the discussion will follow the evolution of one set of process models for bipolar device fabrication in order to demonstrate the tight interrelationship between the models and their application.

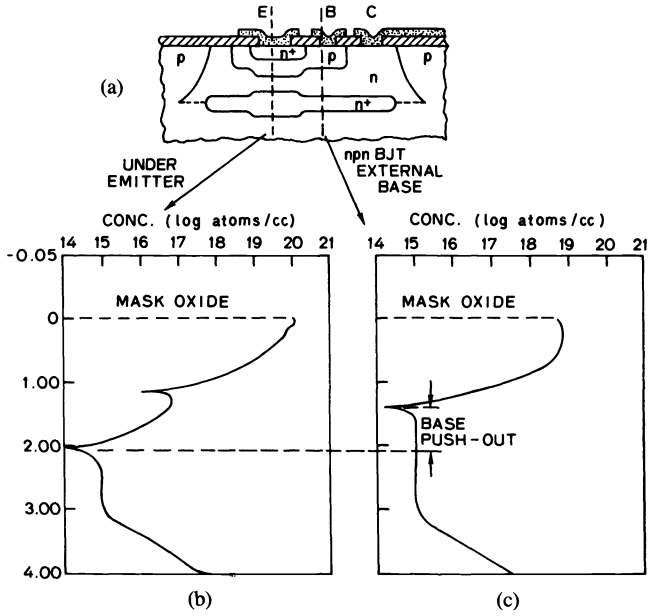


FIG. 1. *Double-diffused bipolar transistor*: (a) *cross section*, (b) *impurity profile under the emitter, and* (c) *impurity profile under the base.*

A. Models for impurity diffusion. The process of thermal diffusion of dopant impurities into a semiconductor is one of the key steps involved in creating integrated circuits. The dopant particles are charged and hence move by both diffusion and drift as given by the flux equation for positively charged species

$$F(x) = -D\frac{\partial C}{\partial x} + \mu \mathscr{E} C \tag{1}$$

where D is the diffusivity, μ is the mobility, $\mathscr{E}$ is the electric field, and C is the concentration of active dopant impurities per cubic centimeter. The diffusivity and mobility obey the Einstein relationship so that

$$\frac{D}{\mu} = \frac{kT}{q} \tag{2}$$

where k is the Boltzmann constant and T is absolute temperature. The diffusivity is a thermally activated process so that its temperature dependence is of the form

$$D = D_0 e^{-E_a/kT} \tag{3}$$

where E_a is an activation energy—typically in the range of 3.4–3.6 eV [4]. The conservation of particles during the diffusion process dictates that their time rate of charge obeys the transport equation

$$\frac{\partial C}{\partial t} = -\frac{\partial F}{\partial x}. \tag{4}$$

It is the solution of this continuity equation for concentration versus both time and distance which is the basis for early process-modeling efforts. Two physical conditions are commonly used in the solution of (4): constant source and fixed dose impulse. Both boundary conditions are applied at $x = 0$.

The solution of the continuity equation for these two cases gives the following:

$$C(x,t) = C_s \operatorname{erfc}\left[\frac{x}{2\sqrt{Dt}}\right] \tag{5}$$

for the constant source value of surface concentration C_s, and

$$C(x,t) = \frac{Q}{\sqrt{\pi Dt}} e^{-x^2/4Dt} \tag{6}$$

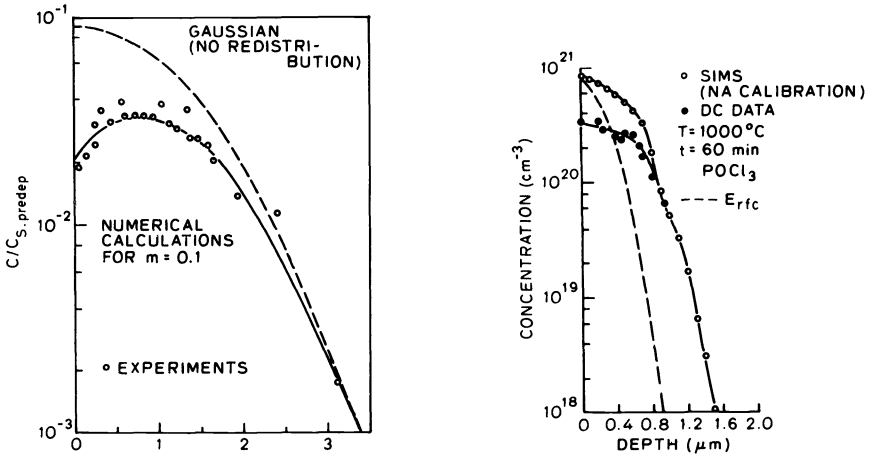
for the impulse dose Q per square centimeter.

The comparison of experiments with these two classical solutions—the complementary error-function and Gaussian forms—soon revealed that several physical effects altered the profiles substantially. For boron, although the diffusion process itself obeyed (1)–(4), the growth of an oxide layer at the surface during diffusion and the preference of the boron to be in silicon oxide rather than silicon give rise to the experimental results shown in Fig. 2(a) [5]. The segregation coefficient m is defined as follows:

$$m \equiv \frac{\text{Equilibrium impurity concentration in silicon}}{\text{Equilibrium impurity concentration in } SiO_2}. \tag{7}$$

Numerical solutions are typically required to properly correct for the impurity segregation; the best fit value of m is shown in Fig. 2(a). The figure also indicates that although the junction depth may well be predicted from (6), the peak concentration will be substantially incorrect. Since the total base doping and base width X_B will be determined by the surface concentration values, the difference between first-order models and experiment can be significant.

Fig. 2(b) shows the comparison between experiment [3] and (5) for a phosphorus diffusion. Clearly the results are not well represented by the complementary error function. Moreover, the two-region nature of the profile makes it difficult to use the equation even in an empirical sense for curve fitting since the sheet resistance will be controlled by the first portion of the profile whereas junction depth (hence X_B) will be



(a) (b)

FIG. 2. *Diffused impurity profiles*: (a) *boron* [5] *and* (b) *phosphorus* [3].

determined by the profile tail. In order to provide even a qualitative agreement with experiment, a more complex set of diffusion kinetics are needed.

Although it is not the purpose of this article to consider diffusion theory in detail, the following discussion illustrates the level of sophistication required in order to provide suitable process models for IC technology. Three key aspects of the diffusion process can be illustrated for arsenic; similar phenomena also apply for phosphorus, boron, and other common dopants. First, the electrical carrier concentration affects the diffusion process. Second, diffusing species can change state and may become inactive [6] or alter the conditions which affect the other mechanisms [7]. Finally, the generation and consumption of point defects by the surface boundary alter impurity diffusion.

The carrier concentration effect on diffusivity can be represented in a general empirical form as [8], [9]

$$D = D + D^{+}\left(\frac{n_i}{n}\right) + D^{-}\left(\frac{n}{n_i}\right) + D^{=}\left(\frac{n}{n_i}\right)^2 \tag{8}$$

where the various D's are superscripted to indicate components due to charge species, n_i is the intrinsic carrier concentration at the diffusion temperature, and n is the local concentration. To date the experimental evidence shows that for the p-type dopants (boron) the D_0 and D^+ terms contribute whereas for n-type dopants such as phosphorus and arsenic, the D_0 and D^- terms dominate. Hence for arsenic the extrinsic diffusivity ($n \neq n_i$) can be represented by

$$D = D_0 + D^{-}\left(\frac{n}{n_i}\right). \tag{9}$$

For n greater than n_i the second term dominates. Fig. 3(a) shows a comparison of calculated diffusion profiles assuming two different surface concentrations, one below n_i and the other substantially greater than n_i. Both calculations were made for the same time and temperature as indicated. From the figure it is clear that for increased n the

diffusivity is enhanced substantially. Classical theory as given by equation (5) would yield a constant junction depth as reflected by the shallower profile.

At high concentrations, ($n \gg n_i$) the phenomenon of clustering is hypothesized to remove arsenic from the diffusion process by forming energetically favorable collections of dopant. Current studies suggest that both solubility limits and electron concentrations affect clustering. One such equilibrium equation relating total number of atoms and active arsenic is given by [10]

$$C_T = C + mK_{eq}nC^m \tag{10}$$

where C_T is the total concentration, C is the actively diffusing dopant, m is the number of atoms per cluster, n is the electron concentration and K_{eq} is the equilibrium clustering coefficient. The results of the clustering are shown in Fig. 3(b) where the total arsenic, measured by Rutherford backscattering [11], as well as the electrically active portion of the profile are shown. The electrically active portion of the curve reflects a diffusion process given by (9) whereas the portion between the data and the dashed curve indicates the clustered portion given by (10). Although this discussion has been used to illustrate extrinsic diffusivity and clustering effects for arsenic, extrinsic diffusion phenomena are also observed for phosphorus and boron. Although recent reviews on the subject show phosphorus diffusion to be considerably more complex than the mechanisms considered earlier [7], an alternative approach considers phosphorus diffusivity to be controlled by the D_0 and D^- components of (8) with another enhancement in diffusivity at lower concentrations owing to generation of extrinsic point defects [3]. This latter approach has the advantage of being practical from an engineering point of view and it has been used widely in computer programs for process simulation [12].

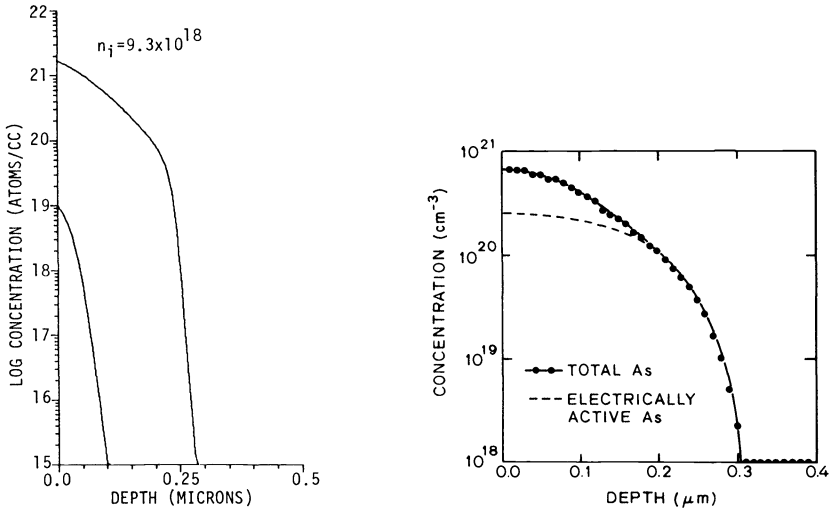


FIG. 3. *High-concentration diffusion effects*: (a) *numerical calculations based on* (9) *and* (b) *experimental results for arsenic* [11].

The final mechanism of primary importance in diffusion is surface generation and consumption of point defects. The oxidation of the silicon surface has been investigated as a generation source [13]. Fig. 4(a) shows the effect of oxidation on diffusivity of phosphorus in comparison to an inert surface condition. The semilog plot of diffusivity versus reciprocal of temperature shows that the difference between the boundary conditions is most pronounced at lower temperatures. This oxidation enhancement to diffusivity (OED) is experimentally determined to be related to the motion of the silicon dioxide interface. The difference in volumetric requirements for silicon in the SiO_2 and Si generate an excess of silicon determined by the oxidation rate according to the following empirical relationship [14]

$$\hat{C}_I = K_I \left(\frac{dX_{ox}}{dt} \right)^q \tag{11}$$

where $\hat{C}_I$ is the excess concentration of interstitial silicon, X_{ox} is the oxide thickness, K_I and q are empirical constants. The value for q is found to range between 0.4 and 0.6. The diffusivity enhancement is then written in terms of this excess concentration of interstitials as follows:

$$D = D^* + d_I \hat{C}_I \tag{12}$$

where D^* is the extrinsic diffusivity as discussed previously and d_I is a proportionality constant. Although the details of kinetic effects involved in oxidation-enhanced diffusion are a subject of great interest and investigation for their own sake, the implications

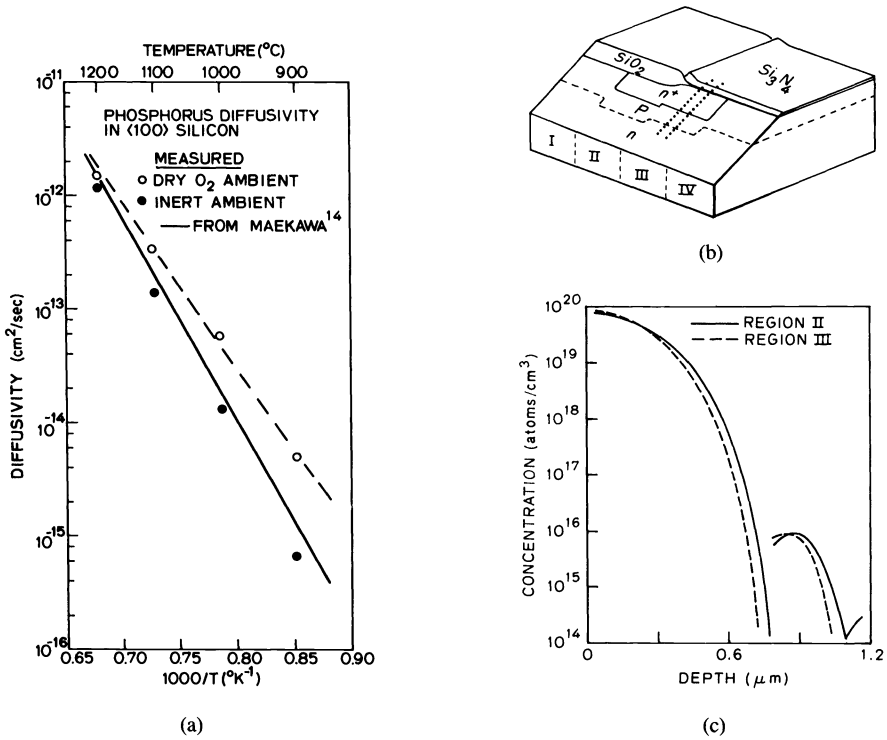


FIG. 4. *Phosphorus diffusivity*: (a) *effect of oxidation enhancement* (OED), (b) *cross section of test structures, and* (c) *measured profiles with and without* OED.

for device fabrication are indeed important. The technology trends toward lower-temperature processing and high-pressure oxidation both directly affect the OED and hence junction depths and electrical parameters. Both phosphorus and boron show similar trends in OED effects. Fig. 4(b) shows the cross section of an experimental structure used to investigate both OED and coupled diffusion effects for boron and phosphorus [15]. In region III the phosphorus enhances the boron diffusion compared to region IV. In region II the surface is oxidized and, as can be seen in Fig. 4(c), both the boron and phosphorus diffusion is enhanced. Fig. 4(c) compares the spreading resistance profiles in regions II and III. A more extensive discussion of OED and other point-defect mechanisms involved in diffusion is given elsewhere [14].

In summary, the generation of point defects is a major effect in altering diffusion. Moreover, the use of local oxidation for both bipolar and MOS device isolation suggests there is a need to understand two-dimensional device implications as well as the one-dimensional surface kinetics.

B. Effects of impurity profiles on bipolar devices. The previous subsection has considered the phenomena involved in impurity diffusion and rudiments of the process models used to describe the one-dimensional impurity distributions. In this section the implications of these impurity profiles on device behavior are discussed. As stated at the beginning of this section, the tight processing tolerances in the vertical dimension for bipolar technology are responsible for advances in process modeling during the 1960's.

Both the dc and ac parameters for bipolar devices are tightly linked to impurity distributions. Junction capacitances are directly related to impurity profiles and the statistics of electrical variations with process sensitivities have been studied [16]. The dc parameters such as current gain and base transport current show the greatest sensitivity to process variations. Two physical factors account for this sensitivity and the increased difficulty in modeling the electrical parameters directly from process models. First, the base width is extremely sensitive to the high-concentration emitter diffusion as well as the coupling of base and emitter profiles. As discussed in Subsection II-A, the complete set of kinetic effects in this regime of processing physics is indeed complex. Second, the base current, which directly affects current gain, is a function of both the emitter profile and minority-carrier recombination phenomena.

The problems associated with process modeling of the emitter profile have already been discussed. Considerations of minority-carrier effects in the emitter have resulted in improved models for carrier recombination as well as bandgap narrowing [17] which alter both the recombination and injected minority-carrier densities. Present trends in bipolar technology now emphasize polycrystalline emitter structures, both as a diffusion source [18] and to provide self-aligned contacts [19]. The exact role of this multilayer emitter is a current topic of investigation, and it appears that both interface phenomena [20] and impurity profile effects in the emitter [21] affect base current. Recent experimental evidence indicates that while the interface transport shows reproducible characteristics, the tight coupling of impurity diffusion and minority-carrier transport effects requires detailed knowledge of both in order to provide a first-principles model for base current [22]. In addition, recent efforts to model polysilicon diffusion effects have resulted in new simulation capabilities for multilayer systems [33].

While it has not yet been possible to model bipolar current gain directly from the specification of process variables, the modeling and characterization of the base transport has been a useful engineering tool. The well-known Moll–Ross relationship

[23] defines the key variables involved in understanding the process sensitivities of the current transported between emitter and collector

$$I_{CE} = I_S(e^{qV_{BE}/kT} - e^{qV_{BC}/kT}) \tag{13a}$$

where

$$I_s = \frac{qAn_i^2}{\int_0^{X_B} N_A(x)/D_n\,dx} \tag{13b}$$

and A is the emitter area, D_n is the electron diffusivity, $N_A(x)$ is the base doping profile, and X_B is the base width. It is clear from this equation that the integral quantity is the dominant process-dependent variable; it is controlled by both the impurity distribution $N_A(x)$ and its spatial extent X_B. During the 1970's it was found that the correlation of the pinched-base resistance[1] and I_s was an excellent means by which to monitor process control in bipolar device fabrication [24]. More recently, correlation of I_s directly with fabrication variables has shown the key role of both the integrated dopant concentration and subsequent thermal processing.

A conventional double-diffused process using $POCl_3$ predeposition shows an extreme sensitivity of both I_s and current gain to predeposition time as indicated by the simulated results in Table I. By comparison, the base-collector zero bias capacitance,

TABLE I
Effect of emitter predeposition time on the electrical characteristics

Emitter Predeposition Time	Pinched Base Sheet Resistance	β_{max}	I_s	C_{JCO}
(min)			A	pF
31.35	11.4 K Ω	170	1.4×10^{-15}	.182
33	37.3 K	260	3.2×10^{-15}	.179
34.65	117. K	410	1.2×10^{-14}	.178

CJCO, shows no sensitivity to this emitter process variation. The tightly coupled emitter and base profiles do not substantially alter the base-collector junction gradient. In order to overcome these process sensitivities, ion implantation is used to accurately control the total dose of boron and phosphorus and to reduce sensitivities to subsequent thermal cycles. Fig. 5 compares the sensitivity of I_s to variations in time and temperature for the ion-implanted process compared to the "standard" chemically predeposited process. Both processes were designed to have similar vertical junction depths and maximum dc current gain. For variations of 1 min of time on all steps and 5 degrees in temperature, the resulting percentage variations in I_s are shown. Clearly the temperature for the base predeposition and emitter drive-in temperatures has the greatest impact—the ion-implanted process shows a five-fold reduction in sensitivity to the latter variation. In addition, the sensitivity to time variation is dramatically reduced for the ion-implanted process. As a final point, Fig. 6 compares the measured current

[1]The pinched-base for an n-p-n bipolar device is the p-type resistance layer vertically bounded by emitter and collector n-type regions—essentially a JFET with emitter and collector as gate.

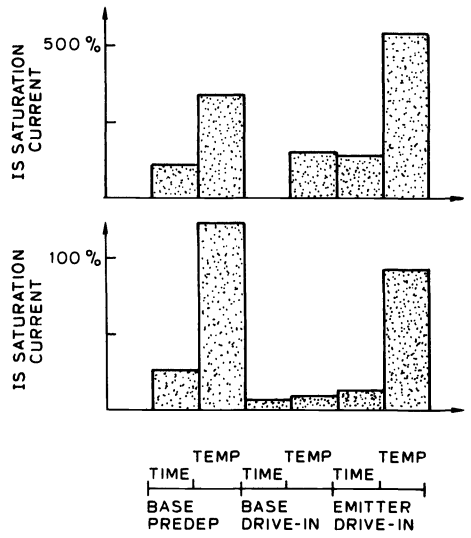


FIG. 5. *Comparison of process sensitivities of saturation current for a bipolar device (top) chemical phosphorus source, (bottom) ion-implanted emitter.*

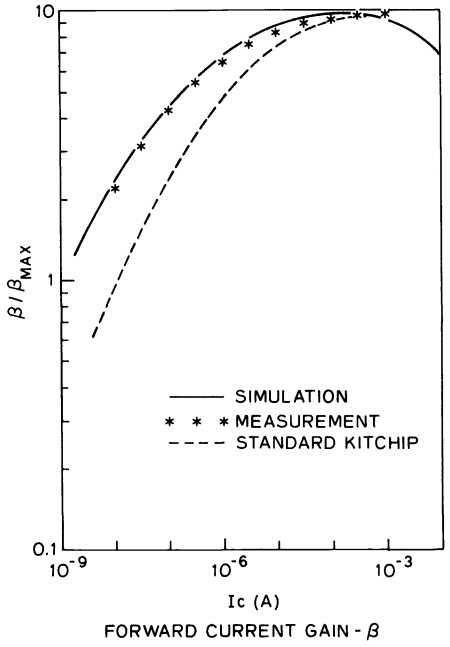


FIG. 6. *Comparison of experimental bipolar transistor current gain for chemically deposited and ion-implanted emitters.*

gain versus collector current for the two processes. As can be seen from the figure, the ion-implanted process shows more than a factor-of-two improvement in low-current gain as computed by means of one-dimensional device analysis [25] based on the appropriate simulated impurity profiles [26]. The comparison with experiment gives excellent agreement. The results show that for identically specified processes (in terms of junction depth and maximum current gain), the reduced surface doping for the emitter in the ion-implanted process reduces the low-level emitter recombination current substantially.

From a process design point of view, the control of I_s dominates all other considerations. A plot of the measured variation of maximum current gain versus I_s (Fig. 7) shows excellent correlation ($R = 0.55$ for a significance level of 0.28). Based on similar plots of data and using regression analysis to fit straight lines, the equations given in Table II are obtained [27]. The results show that both dc and ac parameters are accurately modeled based on the dominant statistical variable, the base transport current [27]. These linear relationships of model parameters provide a two-fold design advantage. First, the initial phase involving circuit versus process design trade-offs can be investigated directly. Second, the statistics of circuit behavior can be realistically assessed. To illustrate this latter point a low-power ECL circuit shown in Fig. 8 was fabricated using the "standard" bipolar process and characterized for input offset current and output offset voltage. The measured values for 35 circuits and their mean and standard deviations are given in Table III [28]. Using the correlated device model parameters given in Table II along with the low current coefficients n_e and C_2 from the Gummel–Poon model, the simulated distributions are also displayed in Table III. The results show excellent agreement with experiment. In contrast, the use of a Monte Carlo selection of model parameters from the complete measured set of device data gives more than 500-percent error in the standard deviation for V_{out} and 1500-percent error in the standard deviation for I_{in}—clearly unrealistic for yield modeling, despite the use of measured device data.

TABLE II

Simplified bipolar transistor model based on correlation to saturation current

$\beta_F = 6\ I_s + 140$
$\beta_R = .2\ I_s + 1.2$
$C_{JEO} = .6 - .01\ I_s$ (pF)
$f_T = 16\ I_s + 583$ (MHz)
I_s is a multiple of 10^{-16}A

TABLE III

Measured and simulated electrical results for ECL *circuit based on simplified model (Table* II)

	V_{out}(mV)		
	Measured	No Corr. (MC1)	With Corr. (MC2)
Minimum	-264.4	-489.72	-265.84
Maximum	-192.2	- 80.66	-167.18
Mean	-236.959	-214.76	-218.18
Std. Dev.	17.164	97.22	21.017
	I_{in}(nA)		
	Measured	No Corr. (MC1)	With Corr. (MC2)
Minimum	8.793	10.0	5.822
Maximum	90.14	1036.9	122.886
Mean	24.603	287.7	68.753
Std. Dev.	18.121	271.8	40.6

The conclusions from this last discussion are as follows. Process modeling allows a flexible tool for bipolar-device design. While process models for high-concentration profiles still require substantial improvement in terms of their physical basis, they are invaluable in designing processes with reduced sensitivity to process variations. Moreover, by constructing device models which incorporate the dominant process variations it is possible to develop realistic yield-oriented models for circuit design. In fact, the notion of a design centering tool which formalizes the method of relating process sensitivities to the relevant device and even circuit variables has recently been demonstrated [29].

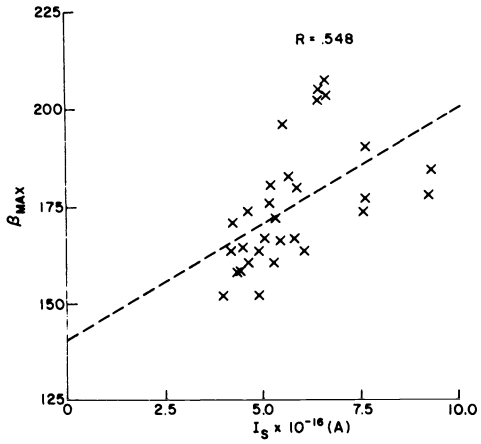


FIG. 7. *Scatter plot showing correlation of beta and saturation current.*

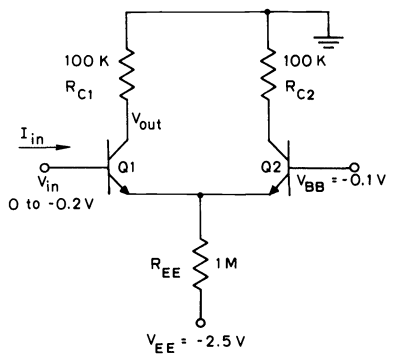


FIG. 8. *Low-power* ECL *circuit.*

III. MOS technology and process modeling. Early in the 1970's, MOS technology began to emerge as the dominant high-density IC technology. The announcements of memory chips with a factor-of-two greater storage have been the key news items each year. By the mid-1970's major portions of the CPU for computers were achieved on single IC's. Now the interest and emphasis in the 1980's is on large word-length computer components with extensive system features. The density of MOS provides a unique leverage in reducing interconnections off-chip and hence improving both cost and reliability.

Fig. 9 shows a portion of the cross section of a 2-μm channel length self-aligned MOS device with local oxidation used for device isolation. Several features of this structure illustrate the factors which have resulted in the density advantages of MOS as well as the process modeling concerns to be discussed next. First, the gate region is patterned in polysilicon, which allows the so-called self-alignment of source and drain impurities—they are introduced into the silicon after the etching of the gate. Moreover, polysilicon is used as a diffusion source to contact the bulk region in noncritical areas to form buried contacts. Both these features provide distinct advantages in terms of density, and similar techniques are now being applied to advanced bipolar devices. The second distinctive feature of the device relates to the oxide isolation. Since MOS technology relies on electric field penetration to induce mobile charge near the silicon surface, selectively thicker insulating regions are needed to provide isolation. The selective isolation requires care during fabrication from several perspectives: oxidation produces a volumetric expansion which leads to nonplanar surfaces, a potential problem for metallization step-coverage and electrical breakdown of devices. The oxidation process generates interstitial silicon which is thought to result in fixed charge in the oxide which alters threshold. Moreover, the local stress and point-defect generation can result in higher leakage currents and reduced breakdown due to generation of surface dislocations and growth of bulk stacking faults. In the next two subsections MOS technology will be considered with special emphasis on oxidation. First, the basic ideas of kinetic models for oxidation will be reviewed. Then the device implications, especially those related to local oxidation, will be discussed.

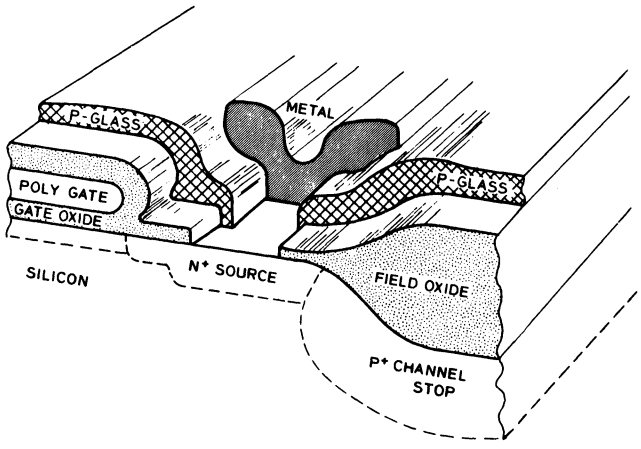


FIG. 9. *Cross section of oxide-isolated* MOS *device.*

A. Models for oxidation of silicon. The thermal oxidation of silicon and its excellent mechanical and electrical properties have been crucial to the advancement of silicon integrated circuits. The first-order one-dimensional model for oxide growth was presented by Deal and Grove in 1965 [30]. The basis for this model is the steady-state relationship among incoming reactants, diffusion across the growing layer, and surface reaction. For given effective oxidant concentrations C_0 and at the gas–oxide interface $X = 0$, and C_i at the oxide–silicon interface $X = X_{ox}$, the three fluxes can be written

$$F_1 = h(C^* - C_0), \tag{14a}$$

$$F_2 = D\frac{C_0 - C_i}{X_{ox}}, \tag{14b}$$

$$F_3 = k_s C_i \tag{14c}$$

where h is the gas phase mass-transfer coefficient, D is the diffusion coefficient for oxidant in the SiO_2, and k_s is the surface reactor rate constant for oxidation. The three fluxes are equal for the steady-state case. Thus solving for C_0 and C_i in terms of C^*, the flux reaching the interface to react and thereby move the oxide interface by dX_{ox} is given by

$$N_1 \frac{dX_{ox}}{dt} = F_3 = \frac{k_s C^*}{1 + \frac{k_s}{h} + \frac{k_s X_{ox}}{D}} \tag{15}$$

where N_1 is the number of oxidant molecules incorporated into a unit volume of oxide. The resulting classical relationship for oxide thickness is then found to be

$$X_{ox}^2 + AX_{ox} = B(t + \tau) \tag{16}$$

where

$$A = 2D\left(\frac{1}{k_s} + \frac{1}{n}\right), \quad B = \frac{2DC^*}{N_1}, \quad \text{and} \quad \tau = \frac{x_i^2 + Ax_i}{B}$$

for an initial oxide thickness of x_i.

The basic relationship given in (16) has proved to be an invaluable process model. A variety of ambient effects can be accounted for by changing the coefficients A or B. Dry oxidation, steam ambient, and even partial pressures of HCl have all been modeled in this way. Moreover, the change in dopant concentration in the silicon has been shown to affect oxidation rate, and this too can be accurately represented by changes in the appropriate coefficients. The following equations illustrate the nature of one such empirical formulation [31] which accounts for these doping-level-dependent kinetic effects

$$\frac{B}{A} = \left(\frac{B}{A}\right)^i [1 + \gamma(C^v - 1)] \tag{17}$$

where

$$C^v = \frac{1 + C^+\left(\frac{n_i}{n}\right) + C^-\left(\frac{n}{n_i}\right) + C^=\left(\frac{n}{n_i}\right)^2}{1 + C^+ + C^- + C^=}$$

and

$$\gamma = 2.62 \times 10^3 \exp\left(\frac{-1.10\,\text{eV}}{kT}\right).$$

Hence, the linear growth rate term is directly affected by the concentration of vacancies, which is in turn controlled by carrier concentration—much as dopant diffusivity.

The scaling of MOS devices has resulted in three important trends related to oxidation effects. First, the growth of thin dielectrics for the gate region has become a critical factor. Second, growth of locally oxidized regions controls device spacings. Finally, the redistribution of impurities in all regions of the device—channel, junctions, and isolation regions—affect performance. Examples of advances in modeling in each of these areas are now presented.

The growth mechanisms for thin oxides is a topic of great interest, and a definitive first-principles model is still to be developed; however, recent experiments involving *in situ* characterization of oxide thickness versus time have revealed new insight. Fig. 10 shows data of oxidation rate versus oxide thickness for two orientations of the silicon surface. In contrast to the thick oxide regime, the growth rates for thin oxides show sharp increases for short time and thin layers. The enhanced oxidation rate is clearly evident from Fig. 10. An extensive collection of data reveals that the oxidation rate can be adequately modeled using the following empirical formula [32].

$$\frac{dX_{ox}}{dt} = \frac{B + k_1 e^{-t/\tau_1} + k_2 e^{-t/\tau_2}}{2x + A}. \tag{18}$$

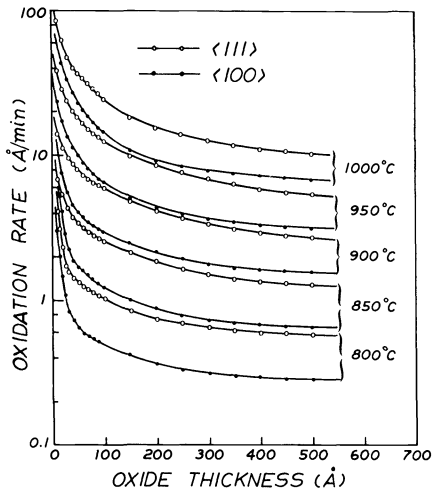


FIG. 10. *Oxidation rate versus thickness for thin oxides* [32].

The first term is the conventional dependence as reflected in the differential form of (16). The second two exponential terms involve exponential decaying functions, dominant in the 20- and 200-Å regimes of oxide growth, respectively. The pre-exponential coefficients show different Arrhenius plot behavior which may reveal information on the mechanisms responsible for both the ultra- and very-thin oxide growth mechanisms [32]. From a more practical viewpoint, a single exponential dependence using the spatial rather than time variable can be used to modify the B/A coefficient and give an excellent fit to data for oxides greater than 50 Å. Combining the thin-oxide and heavy-doping effects in a single expression gives the following equation:

$$\frac{B}{A} = \left(\frac{B}{A}\right)^{i} \left\{[1+\gamma(C^{v}-1)][1+ke^{-x/L}]\prod_{i=1}^{n}\right\} \tag{19}$$

where the heavy-doping term and thin-oxide terms are easily identified. The term loosely denoted as a product over the ith other effects suggests that partial pressure, orientation, chlorine ambient effects—to mention only a few—must also be accounted for. A more complete discussion of the present understanding related to these other effects is presented elsewhere [4].

As one can see from the previous discussion, the kinetics of oxidation are intimately tied to electrical activity in the substrate, ambient conditions, and even mechanisms within the layer itself. As stated at the beginning of this section, local oxidation (LOCOS) is a key component of modern MOS technology. Yet the growth of LOCOS results in still other kinetic effects beyond those described earlier. Fig. 11(a) shows the simulated cross section of a LOCOS structure during oxidation where the two-dimensional motion of the oxide elements is shown as well as the constraining nitride layer which exerts a normal stress. The arrows in the figure indicate oxide motion during growth. Note that because of the nitride stress, the oxide moves laterally out from the masking layer. Fig. 11(b) shows the measured cross section of the sample simulated from Fig. 11(a). The "bird's beak" and head are reflections of both the stress and flow as simulated. A full discussion of the two-dimensional kinetics is complicated. However, three key points should be mentioned. First, the concentration of oxidant C satisfies the two-dimensional Laplace equation

$$D\nabla^{2}\cdot C = 0. \tag{20}$$

Second, the oxide moves slowly and behaves as an incompressible fluid described by

$$\nabla\cdot V = 0 \tag{21}$$

where V is the velocity of oxide element.

Finally the boundary conditions involve both pressure and velocity and are mixed. The overall governing equation is a simplified form of the general Navier–Stokes hydrodynamic equation

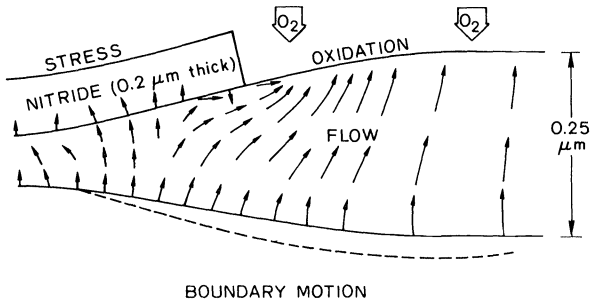
$$\mu\nabla^{2}V = \nabla P \tag{22}$$

where μ is the viscosity and P is the pressure. The velocity at the oxide–silicon interface is given by

$$V = -(1-\alpha)\frac{k_{s}C}{N}\hat{n} \tag{23}$$

where α is the volume ratio of consumed silicon to created oxide (0.473), k_{s} and N are given in (14c) and (15). The two-dimensional solution of (21) and (22) requires iterative numerical techniques and is discussed elsewhere [34]. The key point to emphasize is the fact that the LOCOS structure imposes a more complex physical set of constraints on

(a)



(b)

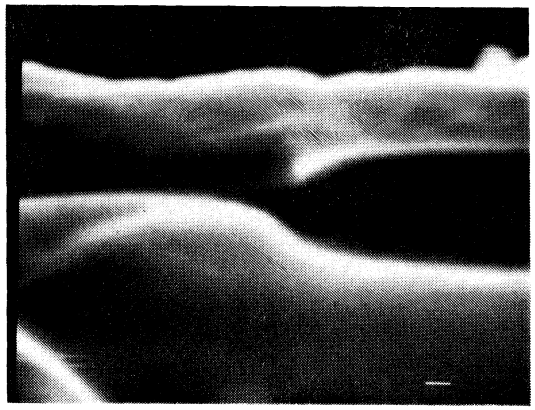


FIG. 11. *Simulated and measured local oxidation cross section*: (a) *numerical results showing key variables and* (b) *electron microscope picture.*

the kinetic models than the one-dimensional case. Indeed, the trends in kinetic models for IC processes as discussed in Section IV indicate the overall importance of advances in this kinetic understanding and the need to push further in these models to keep pace with present and future device technology.

Having discussed both the basic one- and two-dimensional kinetic models of oxidation as well as selected features involved in second-order phenomena, let us next consider the electrical effects on MOS devices.

B. Effects of oxidation and lateral diffusion on MOS devices. The basic operation of the MOS device involves field effects imposed by electrodes on doped regions of the silicon. Hence the two- and even three-dimensional solution of Poisson's equation, using the correct oxide topography and selective dopings in the substrate, is essential in understanding both the devices and their sensitivities to fabrication technology. Fig. 12(a) shows the plan view of an NMOS inverter [35] and Fig. 12(b) shows the technology cross section of the enhancement and depletion devices as well as the nature of the LOCOS isolation regions. In the examples which follow, the sensitivity of several device parameters to the underlying process variables are discussed. In most of the cases the process variations lead to two-dimensional effects. Although limited success has been realized in test structures and other measurement tools for these effects [15],

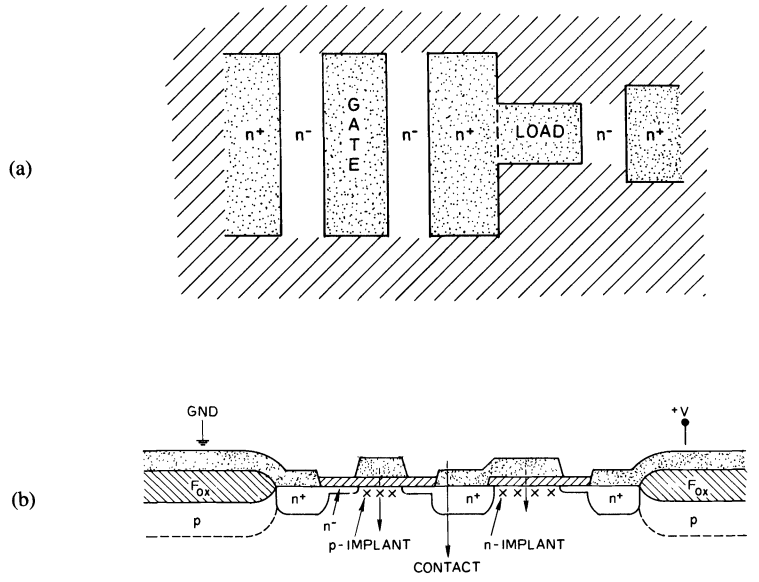


FIG. 12. NMOS *Inverter*: (a) *plan view and* (b) *technology cross section.*

further technology development will dominantly use two-dimensional device simulations to correlate with electrical effects. The simulations most clearly show the dominant effects and in fact have facilitated initial attempts to extract process-related device parameters similar to the bipolar quantities such as I_s which have been discussed in Section II.

Fig. 13 shows the well-known curve of threshold sensitivity (simulated) with channel length along with experimental data. The spread in δL along the channel-length axis is the same at all channel lengths, but the resulting δV_T increases dramatically for shorter channel lengths. Fig. 14 shows simulated punchthrough threshold curves for the 1-μm channel length device. A variation of 10 percent in junction depth, for a fixed channel length, results in an order of magnitude change in current at a fixed V_{DS} [36]. Although the gate patterning is the dominant factor [37] affecting the short-channel variations in threshold, it is clear that vertical dimensions—directly changed by oxide etching and OED—have a substantial role at micrometer device dimensions. Both Figs. 13 and 14 show extreme sensitivities of electrical parameters to physical parameters for short-channel MOS devices.

Turning to the depletion load we can observe critical mask and process sensitivities for this narrow-width device [35]. Fig. 15(a) shows the impurity and LOCOS oxide profiles for a nominal masked 4-μm-wide depletion device. Also shown with dashed lines on the figure is the calculated depletion edge. The depletion edge reveals two facts: 1) the narrow-width threshold for this device is quite different from short-channel effects due to the shape of the lateral boron diffusion; and 2) the region of strong inversion is noticeably smaller than the masked 4-μm dimension due to both oxide and dopant encroachment. Fig. 15(b) shows the simulated bias dependence of available channel charge for the depletion device as a function of masked device width. While the

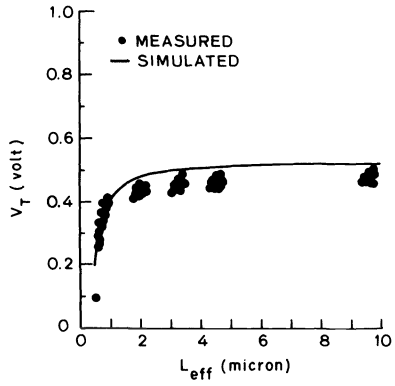


FIG. 13. *Threshold voltage versus channel length.*

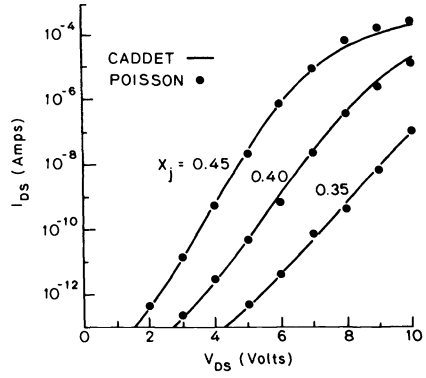
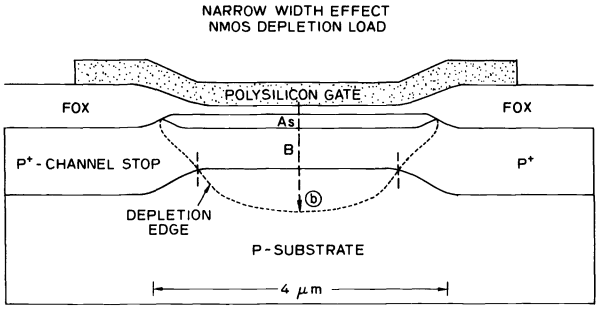


FIG. 14. *Punchthrough current versus drain voltage for a nominal* 1-μm *channel length device.*

extrapolated negative threshold shows minor variations, the slope for ± 1-μm variations is dramatic. The large change reflects the dominance of the LOCOS encroachment including diffusion especially for reduced widths. This example illustrates the need to develop new isolation methods for reduced-dimension devices. Moreover, the need for 2D oxidation and diffusion modeling has now become critical as we continue to reduce device dimensions.

As a final example in this section, consider the trade-offs to be made at the drain edge of the LOCOS region. Electrical factors include sidewall capacitance, breakdown, and leakage. From a structural point of view, the surface planarity and lateral bird's beak encroachment must be balanced with problems such as defect generation in the substrate. Fig. 16 shows the simulated and measured trade-offs to be considered in choosing boron dose in the field region [38]. The results show that over more than an order-of-magnitude change in dose, the capacitance can be reduced while continually increasing the breakdown. Obviously the field threshold must be factored into this

(a)



(b)

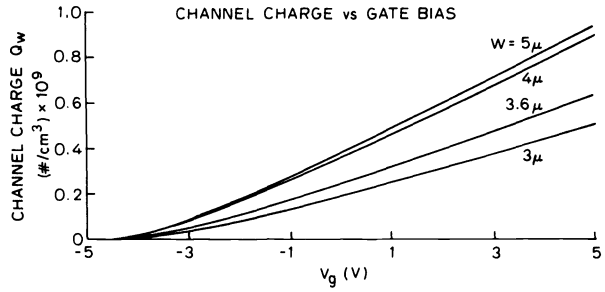


FIG. 15. NMOS *depletion load*: (a) *technology cross section including depletion edge and* (b) *simulated channel charge versus bias.*

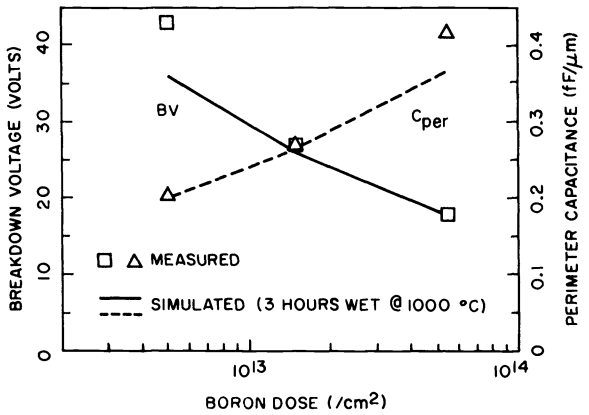


FIG. 16. *Simulated and measured sensitivity of breakdown voltage and perimeter capacitance versus field implant dose.*

analysis as well. Based on these experiments, the following empirical relationships are obtained:

$$BV\alpha(\text{dose})^{-1/3}(X_{ox})^{0.75}, \tag{24a}$$

$$C_{per}\alpha(\text{dose})^{+1/3}(X_{ox})^{-0.75}. \tag{24b}$$

These dependences are not immediately apparent from any first-order calculations [38]. However, the coupled 2D process and device analysis to be discussed in Section V clearly reveals these important design relationships. It should be emphasized, much as for the narrow-width depletion load, that the two-dimensional doping profile effects dominate the device performance.

While the previous example has emphasized the diffusive portion of the LOCOS step, the oxide growth itself plays a major role in determining circuit yields. In particular, the parameters such as pad oxide and nitride layer thickness used for the LOCOS directly affect defect generation. Recent simulations based on the model discussed in Subsection III-A reveal the correlation of calculated integral of stress and measured defect densities versus the nitride thickness as shown in Fig. 17. The trends show that factor-of-two changes in nitride thickness result in more than an order-of-magnitude increase in defects. Similarly, increased oxide-growth temperature reduces stress and defect generation [39]. This final example again illustrates the utility of modeling in the formulation of a design approach over a broad range of physical conditions.

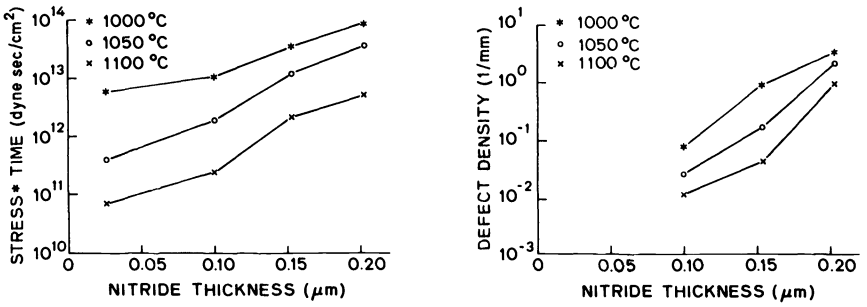


FIG. 17. *Correlation between stress and defect density as a function of nitride thickness*: (a) (*stress*) *x* (*time*) *integral versus nitride thickness, and* (b) *measured defect densities.*

IV. Trends in processing technology. The previous sections have presented an historic view of technology modeling. The phenomena of diffusion have been discussed, primarily from the perspective of bipolar technology. The modeling of oxidation has been driven by the MOS technology—especially the need for local oxidation to produce densely packed devices separated by oxide regions. These processing trends have illustrated the fundamental concerns and motivation for process modeling. The purpose of this section is to briefly provide a broader long-range perspective on the future trends of IC processing.

The equipment required to fabricate submicrometer-dimension devices has advanced rapidly over the past decade. In particular many new pieces of equipment have

been demonstrated to anneal, deposit, and etch films for a variety of silicon-compatible materials. The technology objectives related to each of these process steps is now considered.

Annealing originally referred to a process step which produced a positive device effect, the activation of implanted impurities, or the final alloying of an ohmic contact, for example by modest cycles in temperature and ambient. The present technologies for laser, pulsed light, or other beam processing have resulted in well-controlled temperature transients. The materials effects are indeed profound, ranging from solid-phase epitaxy through nonequilibrium creation of silicide materials. In addition the controllable growth of silicon on insulators (SOI) is likely to eliminate the need for the sapphire substrate used in SOS. The opportunities to completely alter the nature of thermal processing in silicon technology seem vast indeed [40]. The trends toward lower processing temperatures and the use of materials which are sensitive to surface interface reactions favor the nonequilibrium approach. From a process-modeling point of view, these new technologies pose a substantial challenge. The radiation effects result in short time-scale events driven by sharp transients and extreme temperature gradients. The activation of point-defect mechanisms as well as intrinsic concentrations seem to be altered substantially by beam processing. In short, the new techniques for annealing involve highly nonequilibrium events which occur over very short time periods.

The use of deposited films in silicon technology has grown rapidly in the past few years. Low-pressure deposition techniques in particular have experienced a phenomenal growth. In addition to the common dielectric materials—silicon dioxide and silicon nitride—both silicon epitaxy [41] and contact metallization [42] are benefiting from new deposition techniques and equipment. Silicon epitaxy is an especially exciting area from the device point of view since many desirable device effects can be altered by epitaxy. Emitter efficiencies for up-operated bipolar devices can be improved and latchup sensitivities for CMOS reduced, for example. The key concern, however, has been the defect generation properties during growth, and control of thickness and electrical parameters. Trends show favorable progress toward better control of the parameters and defects, suggesting that the use of epitaxial layers is likely to increase in the future. In the area of metallization, the need to reduce contact resistance and control interdiffusion suggest that more versatile deposition techniques are needed. The use of deposited polysilicon layers for improved contact properties and reduced junction depths have been discussed earlier in the context of polycrystalline bipolar emitter effects. The successful demonstration of selective CVD deposition of refractory metals [42] suggests one very positive trend in the direction to control MOS contact properties although vacuum-deposition techniques are likely to dominate for several years to come. As discussed shortly, the controlled etching of surfaces plays a key role in all IC processing, and certainly for metallization the interface properties are a factor of key concern in reliability and reproducibility. In the area of both metallization and dielectric materials, concerns with step coverage are crucial. For metal one must avoid openings due to thinning over steps while in the case of dielectrics it is essential to get good coverage over corners to increase breakdown and improve reliability. In summary, the role of deposition in lower-temperature processing is indeed important, and the need for process models in this area will continue to grow. Both dielectric and conducting layers are benefiting from new deposition techniques, especially those utilizing lowpressure equipment. Although it is too early to identify the leading approaches to selective deposition, the possibilities are quite attractive.

The final area suggested for special consideration is etching. The critical role of technology advances in this area cannot be overstated. The control of gate dimensions

and edges for MOS is a direct result of etching. New oxide isolation techniques use careful control of steeply etched trenches [43]. Even for steps without specific spatial constraints, the end point detection of etching can be crucial to device reliability and defects created during subsequent thermal processing. Despite the critical dimension constraints of etching, both lateral and vertical, the kinetic effects involved are more complex than any other step in the fabrication process [44]. Fig. 18 shows the cross section of a hypothetical structure during an etching step and a few of the possible mechanisms involved in the etching which results in the observed convex-curved surface. The set of events can range from electronic and ionic bombardment of the surface to surface migration and desorption of reactants and products. Most etching techniques presently use several reactants including fluorine, chlorine, and oxygen as key active ingredients. For example, a typical etching process for silicon might involve $C_2F_6 + Cl_2$. There is a wide range of intermediate reactions occurring in the etching process as indicated in Fig. 18. Moreover, the role of bombardment and surface kinetics is not generally included in the normal rate equations. These "ambient" and surface effects will further complicate the kinetic picture. Since patterning is the dominant driving force in achieving smaller devices, it is this technology which is expected to undergo the greatest advances in equipment technology—and hence be the most in need of improved process models. A further discussion of the device implications related to etching is given elsewhere [45].

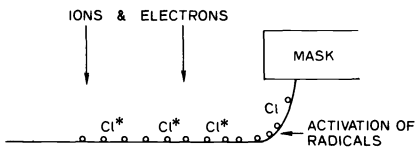


FIG. 18. *Cross section of a wafer during etching and mechanisms and species involved in the kinetics.*

In the previous discussion, I have emphasized annealing, deposition, and etching as key trends in processing technology. In particular, it is these areas in which new equipment will shape the directions of device fabrication, and hence new process models will be needed to keep pace with process design and control. As pointed out in both Sections II and III, the statistics of device characteristics depend critically on some aspects of the fabrication process. In the examples cited above, the doping profile sensitivities for the bipolar base region and the controlling factors of channel length and source–drain junction parameters for MOS have been demonstrated. Looking toward the next decade of IC technology, the lateral dimensional control of etched lines and electrical properties and integrity of thin layers will dominate process control concerns. In this spirit the trends related especially to deposition and etching will become more apparent. The use of recrystallization and special annealing techniques for multilayer interconnects and even active devices [46] is an area which is receiving increased attention. Moreover, the opportunities for innovative device structures in SOI will continue to be an exciting area for research [47]. In this field the concerns with the basic properties of oxides, interfaces and impurity diffusion again become first-order effects, much in the spirit discussed in Sections II and III.

Two areas of intense industrial activity and practical implication are lithography and multilayer metallization. The need to pattern fine lines prior to etching is crucial to

the device scaling efforts. The issues involved in this area are substantially different from the silicon-based materials kinetics discussed here. The interested reader is thus referred to [48], [49]. Similarly, in the area of metallization for multiple levels of interconnect, the concerns move more into the fields of packaging and even organic chemistry than thermal processing which affects the silicon. Again, the reader is referred elsewhere [45].

A final overview of both the process modeling as well as new trends discussed in this section is presented in Table IV. The dominant process steps for which process modeling is currently an active field of endeavor are listed. The second and third columns describe both the mathematical tools and the limiting physical kinetics associated with the several process steps. The reader will recognize several physical models discussed in Sections II and III as well as more complex kinetics not discussed here. Selected references are included in the table to point the way for further study in these areas. The first three rows in Table III refer to implantation, diffusion, and oxidations—the key processes which are better established. Moreover, the manufacturing equipment and the process modeling tools are most advanced in these areas. Both etching and deposition represent the avant garde, in terms of both equipment and the need for more complete process modeling.

TABLE IV
Relationship of process steps, models used for simulation, and limiting physical effects

PROCESS EFFECTS AND IMPLICATIONS			
Process Steps		Analysis Tools	Physical Limits
Ion Implantation		Distributions: Gaussian, Pearson...	Defects/Knock-ons
		Boltzmann Transport [4]	Transient Annealing [40]
		Monte Carlo [51]	
Diffusion		Fick's Law	Diffusivity
	"SUPREM" [12] "SUPRA" [38]	Multi-Stream Models [1] [7]	Precipitation Transient Kinetics
Oxidation		Laplace + Surface Reaction	Surface Kinetics
	"SOAP" [34]	Hydrodynamics (Navier-Stokes) [52]	Stress/Flow
Etching	"SAMPLE" [50]	Huygen Waves [49]	Multi-Stream Kinetics [44]

V. CAD tools for IC design and manufacturing. The previous sections have traced the evolution of process modeling beginning with bipolar concerns and moving to MOS devices and local oxidation effects. Now metallization, lithography, etching, and deposition will dominate IC processing through this decade. The development of process models have been an integral part of the fundamental understanding needed for control of manufacturing technology. The objective of this final section is to consider the computer-aided design (CAD) tools which have evolved in process modeling and to demonstrate their place in IC design and manufacturing.

For purposes of this discussion we limit our attention to the areas of technology, devices, and circuits. Fig. 19 shows parallel paths of development from a system point of view. The left path is that of the system design, ultimately leading to circuit design and IC layout for custom and semicustom blocks. The right path shows the

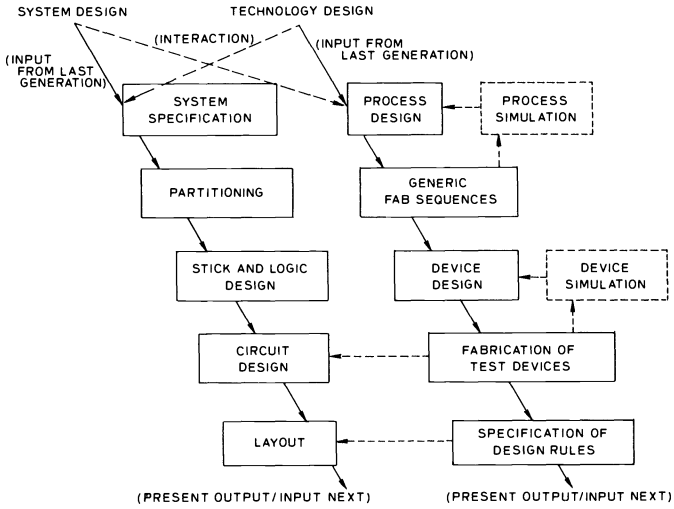


FIG. 19. *Schematic of "system" and "technology" design processes and their interrelationships.*

technology-based steps leading to the working IC devices and design rules used by the circuit designers to create the actual cells and subsystems. Three major interfaces exist between the two paths. At the device and layout levels, the technologists and circuit designers communicate critical performance and layout information—dominantly through the mechanisms of device models for circuit simulation and the rules to create these devices. Hence model coefficients and layout rules are two key exchange formats for communication. We will not discuss layout further, although this topic and its ramifications for achieving greater system complexity on single IC chips is of major importance for VLSI [53]. The third interface is less formal than the previous two, but a critical one in the evolution of technology. This is the projection phase shown as an arrow coming from the last generation circuit and system design efforts. At this interface the projections must be made as to needed technology features—for example, analog capabilities in a digital technology—and the hopes and expectations of both the systems and technology groups must be enunciated and defined. This aspect of the design interface will have a growing level of emphasis and importance with VLSI. There is a need to build cell libraries of substantial complexity, for example programmable ALU's and RAMs, and to scale them into new technologies and design rules. Moreover, the growing need to fully consider system-level factors such as hybrid and printed circuit interconnects, data conversion (A/D and D/A), and communication channels (busses and networks) suggests that chip technology will evolve in new directions. The choice to make on-chip hybrid technologies—for example, bipolar compatible MOS—will reflect a careful interchange across the system/technology interface.

We will define more narrowly the function of CAD for process design and manufacturing at the lower two interface levels shown in Fig. 19. Moreover, the dashed boxes shown as local feedback on the technology path (labeled "process" and "device") indicate that there are somewhat generic design steps involved in the development of technology. Fig. 20 shows a bottom-up slice of CAD at the process and device levels. The multifaceted equipment and process-models level shown at the bottom of the figure

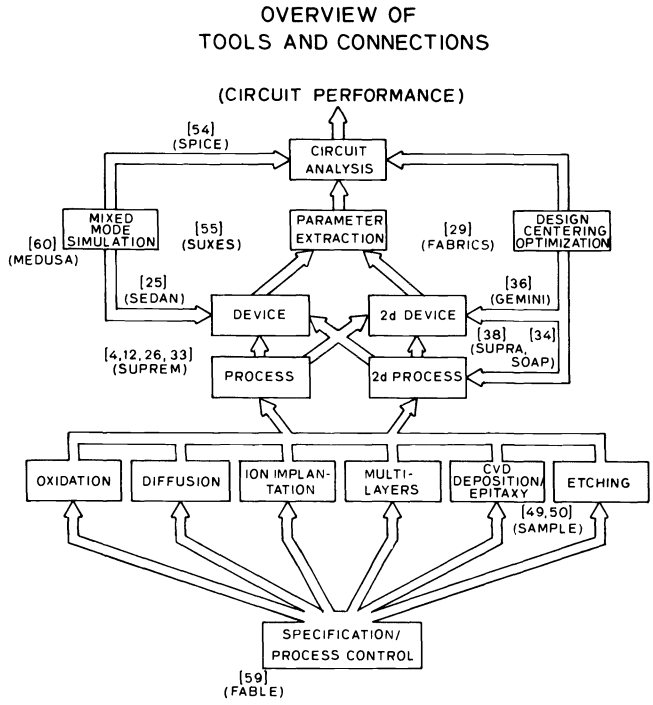


FIG. 20. *Range of* CAD *tools and their interconnection in going from process specification through circuit design.*

reflects the detailed considerations given in the previous sections. The incorporation of these process models into CAD programs, both one- and two-dimensional, provide the capability to simulate the effects of process steps such as oxidation and diffusion, and thereby "inspect" quantities such as junction depth and oxide thickness. Since it is time consuming and costly to fabricate IC's repeatedly during design, the simulation process is very efficient. Moreover, since measurement of physical parameters is difficult and often inaccurate, the use of simulation can be invaluable in providing feedback and cross-checking during both process development and manufacturing. Finally, the complexity of IC processing as reflected in all critical dimensions of submicrometer channel-length MOS devices requires careful design and process control over sequences of steps, not simply single oxidation or diffusion cycles. The predictive capabilities for process modeling CAD have grown rapidly and gained wide acceptance specifically because of the issues of process complexity and need for tight dimensional control.

Device modeling is the level just above process modeling in Fig. 20. Historically, these activities have been used extensively to understand fundamental limits of devices and predict performance. The successful evolution of process modeling—both one- and two-dimensional—now has established a basis for realistic and predictive device models. A number of the available tools at these various levels are included on the figure with appropriate literature references. The SUPREM program [12], [26], [4] is now the most mature of the process-modeling tools and is used extensively in process design. Industrial feedback suggests that there is a thousand-to-one cost savings in simulating a process step compared to laboratory cost. As we have demonstrated in

Section III the sensitivity of MOS devices to junction depths and oxide shape parameters such as oxide thickness and lateral extent for local oxidation plays a key, if not dominant, role in determining electrical characteristics. Hence, the availability of accurate process-modeling tools has laid the foundation for increased use and accuracy of device simulation.

The final levels of CAD tools shown in Fig. 20 are those for circuit simulation and device model parameter extraction. The circuit simulator SPICE [54] is shown as a specific tool because of its wide acceptance for more than a decade. The use of circuit simulation to evaluate performance is essential to the circuit designer if not also to the system designer. One can accurately determine speed limitations of gates and memory as well as off-chip current drive capabilities. Circuit simulation has replaced breadboarding at the chip level, primarily because of its demonstrated accuracy and speed compared to the inaccuracy and cost of breadboards.

A key limitation to simulation accuracy at the circuit level is device model parameters. The extraction capabilities shown in Fig. 20 primarily refer to extraction for prototype and production devices. Recent publications [55], [56], [37] have demonstrated efficient means to extract both single sets of parameters as well as statistically meaningful sets of parameters including critical correlations of parameters and physical effects. Sections II and III have shown the importance of such a statistical approach, especially for VLSI where the number of devices used and the distribution of their parameters limits design margins and performance. Fig. 20 also shows two alternative paths related to circuit simulation—mixed-mode simulation [60] and design centering of circuit parameters based on process and device inputs [29]. Each of these techniques tries to capture lower levels of detail in a form useful for circuit design. Mixed-mode simulation allows the direct embedding of device simulation in a circuit simulator environment. Design centering provides an environment to evaluate sensitivities of circuit parameters based on first-order analytic models for process and device effects.

The overall intent of Fig. 20 is to show an important coupling and synergism of process, device, and circuit CAD tools. While the present method for extracting circuit simulation model parameters relies heavily on measurements, the increasing use of the other tool sets will become dominant by the mid- to late 1980's. The reasoning parallels the motivation for process modeling itself—it will be faster and more accurate than experimentation, especially as process statistics become more significant. However, the concern with accuracy limitations of models and their coefficients will increasingly be pushed toward the processing end of the CAD tool chain. Limitations in device models related to resistances, capacitances, and even mobility each have an underlying set of technology variables such as impurity doses, junction depths, and oxide thicknesses. The factors-of-two change in device model parameters can increasingly be traced to these underlying technology parameters as well as the associated lithography variations. Hence, Fig. 20 contains two messages. First, the range of CAD tools between process design and circuit simulation will be of growing importance. Second, the use of process and device simulation as local feedback, as shown in Fig. 19, are well established for process design and will grow quickly as manufacturing tools as well.

It is beyond the scope of this paper to consider in detail the analysis techniques or even the state-of-the-art features of one- and two-dimensional process- and device-analysis tools. The previous sections have demonstrated several examples with a suitable sampling of current technology-oriented device effects. Both short-channel narrow-width effects of local oxidation on parasitic devices as well as defect generation can now be modeled in two dimensions. Rather than project into the future concerning how far the evolution of process modeling CAD will advance, I would like to reflect on the

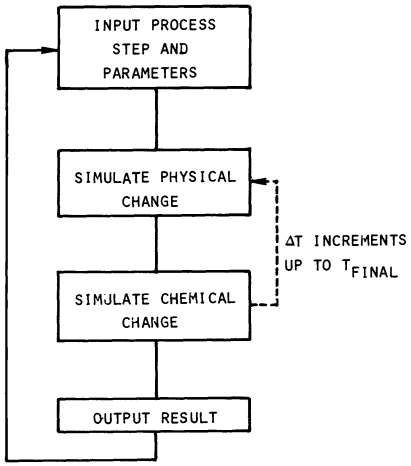
potential for the modeling concepts and technology depicted in Figs. 19 and 20 to become an integral part of IC manufacturing. Implicit in the discussion is a firm belief that process models will keep pace with the rapid development of manufacturing equipment technology and device innovation.

There are four attributes of CAD for process and device modeling which are essential to its long-term success in IC manufacturing: accuracy, speed, friendliness, and system integrability. I will briefly discuss each of these. Accurate determination of process and device parameters is a critical factor for scalability and manufacturing of VLSI. The basic premise for success of CAD tools in this area is to predict the critical performance and process control variables with sufficient accuracy so that devices can be well designed and, more importantly, manufactured. Examples in Sections II and III show clearly this dual role of design and manufacturability.

The factor of CAD tool speed has several implications. First, it is necessary that process and device designers get sufficient feedback quickly so that they reduce costs in the use of the fabrication facilities. Simulation obviously does not replace fabrication—it reduces the trial and error needed to converge to a stable process. Moreover, simulation allows the exploration of the process window and helps identify factors which can affect manufacturing and process control. A second factor related to speed is the growing possibility to use process models directly in the manufacturing environment—for example, as very sophisticated controllers for equipment. Although this aspect of simulation is only beginning to become practical, the prerequisite for its success is simulation speed in real time on the small computers used for process control. Finally, the growing complexity of process and device simulators, dictated by more complex kinetic models of physical processes, has caused a lag in available CPU cycles. The increased speed of next-generation computers has helped substantially. Nonetheless, there will be an ongoing effort in developing better algorithms to match the physics and yet provide real-time computations as suggested earlier.

The user interface for CAD tools, including process and device modeling, is a major factor in establishing broad usage in both engineering and manufacturing environments. In the area of process modeling the user input has evolved from text descriptions similar to SPICE-type syntax for device models. From the manufacturing point of view the text looks very similar to instructions used to control processing equipment. Fig. 21(a) shows one simple sequence of step specifications for an implantation and diffusion sequence. The internal program calculations used to solve for the resulting impurity distributions are shown schematically in Fig. 21(b). The physical changes refer to steps such as oxidation (16) and the chemical changes refer to diffusion (4). The discrete time solution of the respective partial differential equations results in the impurity profile shown in Fig. 21(c). Here the deposited oxide-layer thickness is shown as well as the impurity profile both in the oxide and into the silicon bulk. From the user interface point of view the pictures of profiles are extremely valuable, although more qualitative than quantitative. The transformation of impurity profiles into measurable quantities is the key step to get to the end result. For example, junction depth, sheet resistance, junction capacitance, and threshold voltage are typically desired user outputs. Fig. 21(d) shows one further post-processing capability for process simulation output. The profile from Fig. 21(c) has been converted into a plot of channel charge versus gate potential which can in turn be compared with measurements. From the viewpoint of manufacturing it will be increasingly important that such on-line monitoring of electrical as well physical outputs be generated so that feedback between results and design intent can be critically evaluated.

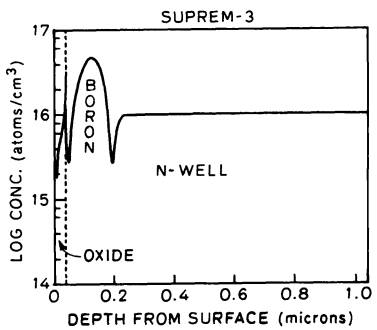
(a)



(b)

```
Title        Example 2 - Oxide and threshold implant

Initialize   <100> Silicon Thick=1 Dx=.01 Conc=1e16 Phosphorus

Deposit      Oxide Thick=.04 Dx=.01
Implant      Boron Dose=6e11 Energy=35 Pearson
Diffusion    Time=40 Temp=950 Nitrogen
Diffusion    Time=35 Temp=900 Nitrogen
Diffusion    Time=40 Temp=950 Nitrogen

Print        Layer
Plot         Cmin=1e14 Cmax=1e17

Save         Structure File=ex2ao
```

(c)



(d)

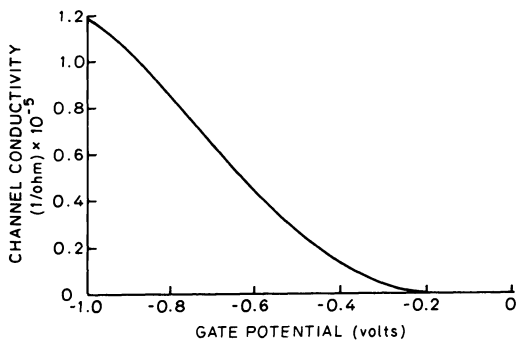
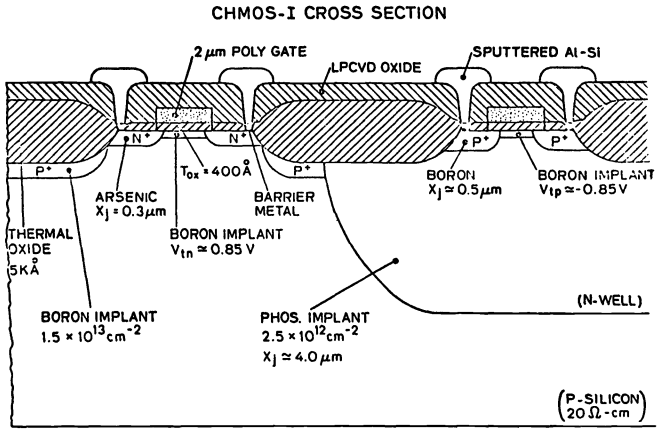


FIG. 21. *User interface and underlying analysis for process simulation*; (a) *input file*, (b) *schematic of computations performed*, (c) *output of impurity profile, and* (d) *computed channel charge versus bias.*

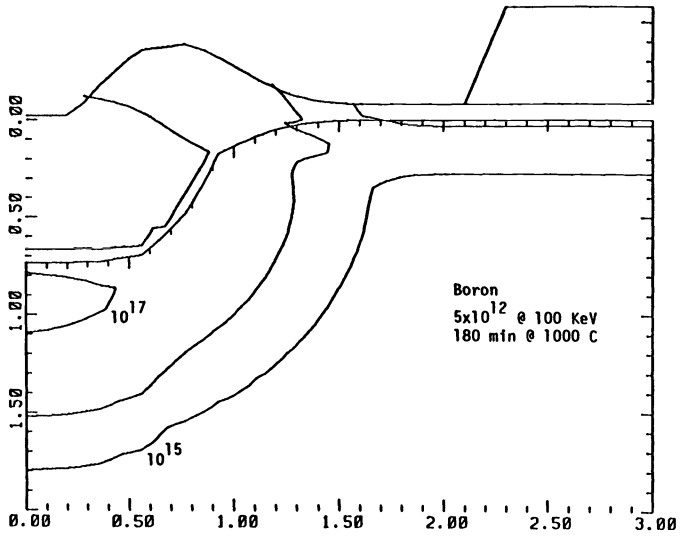
In the area of two-dimensional process and device simulation the user interface requires a different set of considerations. Here the topography is a dominant concern and text descriptions alone are not sufficient. Fig. 22 shows cross-sectional views of a portion of a CMOS device structure. Fig. 22(b) shows the cross section as viewed by the technologist when considering the trade-offs related to the boron implant and LOCOS steps. Fig. 22(c) shows the simulation grid used for 2D process modeling where high-concentration diffusion is to be modeled [57]. The discretization used to solve both the impurity diffusion and oxide interface effects place constraints on the most appropriate positioning of the grid. In addition, there are numerical analysis constraints implicit in the grid specification since the efficiency of the solution is affected by both the number of points and the form of their interconnection. For example, techniques such as line-relaxation (SLOR) or finite-difference (FD) methods each impose different constraints on the spacing of the grid [36]. The grid shown in Fig. 22(c) can also be used for device analysis of the physical structure shown in Fig. 22(a). As stated above, the physical as well as numerical constraints dictate how this grid must be refined or altered for the device modeling application. In this case, since electrical rather than impurity diffusion effects are involved, the grid must be altered in several regions. For example, the electronic analysis now requires solution for depletion edges and carrier distribution which occur in regions physically quite separate from the regions of maximum concentration as shown in Fig. 22(b). The user interface problems associated with the systematic progression from Fig. 22(c) into device analysis is still an art rather than a science. Fig. 23 defines schematically the problems of grid definition and the associated physical constraints imposed by the process and device simulation. The user interfaces indicate required interaction. Recent research results indicate successful automation of parts of the process, and hence the ability to relieve the user of these time-consuming and error-prone steps [58]. The vertical sequence of choices in process and device-modeling grids indicate options at several levels. The physical constraints of the systems of equations—Fick's law contrasted with the Poisson and continuity equations—suggest the need for substantially different grids. The total merging of grids for both process and device modeling offers the user a great relief, but may excessively compromise the underlying numerical solvers. The choices to be made both at a user level and in the underlying algorithms will evolve rapidly over the next decade. Certainly automation of algorithms to increase numerical efficiency and simultaneously enhance the user interface are the preferred directions of evolution.

The final essential attribute of CAD tools for process and device modeling is system integrability. The meaning of this term can be discussed at three levels. These levels of integration have been implicitly defined in Figs. 19, 20, and 23—each indicating a lower level of interfacing and simultaneously a higher level of demands on performance and algorithms. The long-range wish is for an "Integrated CAD System" —a cradle-to-grave approach in the design, production, and testing of IC's. Certainly the need for CAD tools at all levels shown in the figures discussed throughout this section are crucial to the success of IC design and manufacturing. Extensive algorithmic research will produce the desired interfaces depicted in Fig. 23. The integration of the CAD tools shown in Fig. 20 will be driven by the increasingly tight manufacturing controls needed for scaled-down devices. Moreover, the device and interconnect models needed for SPICE will dictate increased use of process and device simulation to extract the current physical parameters and coefficients. Finally at the system level shown in Fig. 19, the need to manage and control production will mandate the high-level integration of process and device modeling CAD. Recent efforts to create CAD languages for IC manufacturing [59] suggest a very positive and far-sighted perspective

(a)



(b)



(c)

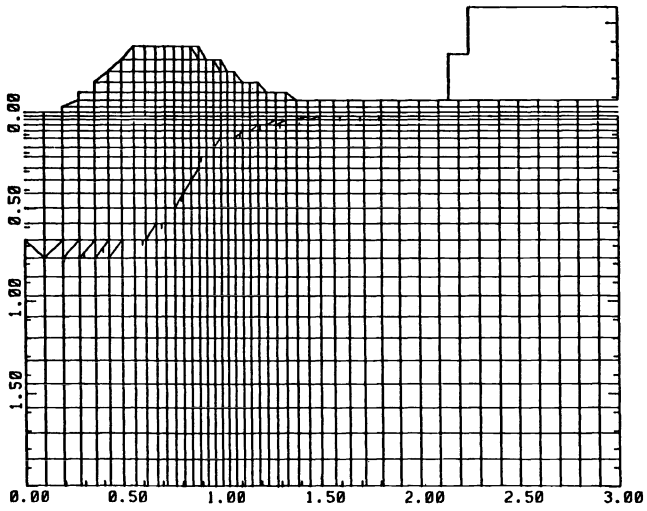


FIG. 22. CMOS *cross section and analysis "windows"*: (a) *complete inverter*, (b) *"window" for isolation analysis, and* (c) *numerical grid for device analysis.*

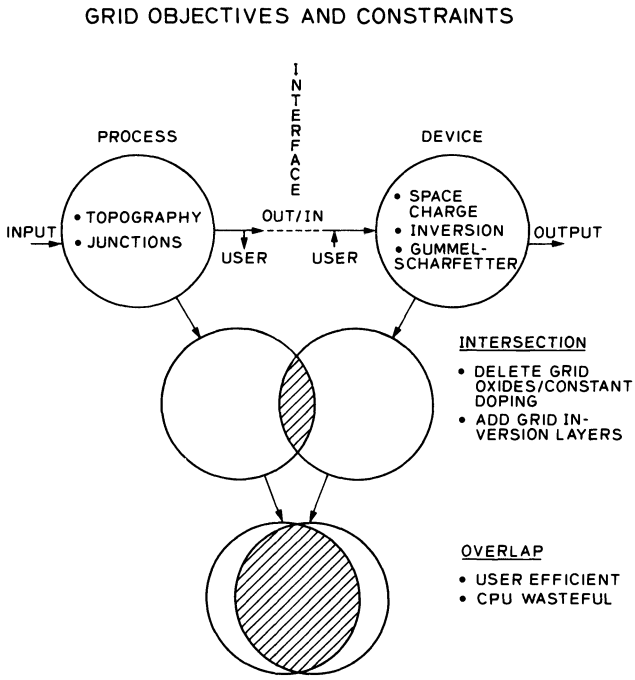


FIG. 23. *Grid objectives and constraints for process and device analysis.*

on solutions to the system needs implied by Fig. 19. The needs to automate production-trained personnel, schedule equipment, and document procedures and results are all key ingredients of manufacturing system. Indeed, the CAD tools discussed in this article are one set of specific computer aids to be incorporated into such a manufacturing system of the future.

VI. Conclusion. The intent of this article is to put the details of process and device modeling in perspective, relative to the history and future of IC manufacturing. Since its inception, technology modeling has been used to understand the fundamental limits of process and device control. This paper has traced primarily the evolution of process models for impurity diffusion and oxidation. Examples for bipolar and MOS technologies have been given to show details of application to both device design and manufacturing control. In considering the future for process and device modeling two aspects have been emphasized. The trends in equipment technology for IC manufacturing will play a major role in dictating the need for further modeling work. Furthermore, the CAD tools for technology modeling fit into an overall system-design approach. This system must integrate individual CAD tools to meet the overall objectives of IC manufacturing.

Acknowledgments. The author wishes to thank the many doctoral students who have contributed substantially to the work discussed, as is clear from the Reference section. The author also wishes to thank his colleagues in the Integrated Circuits Laboratory for a deep commitment to process modeling and a synergistic spirit to make the couplings with device and circuit analysis work—both in software and in the Laboratory. Special thanks go to Prof. J. D. Plummer and Prof. J. D. Meindl and to S. E. Hansen. Critical reading of the manuscript by D. Ward, P. Fahey, and D. Chin is greatly appreciated.

REFERENCES

[1] S. M. Hu and S. Schmidt, "Interactions in sequential diffusion processes in semiconductors," *J. Appl. Phys.*, vol. 39, pp. 4272–4283, 1968.

[2] S. M. Hu and T. H. Yeh, "Approximate theory of emitter-push effect," *J. Appl. Phys.*, vol. 40, pp. 4615–4620, 1969.

[3] R. B. Fair and J. C. C. Tsai, "A quantitative model for the diffusion of phosphorus in silicon and the emitter dip effect," *J. Electrochem. Soc.*, vol. 124, pp. 1107–1118, 1977.

[4] C. P. Ho *et al.*, "VLSI process modeling—SUPREM III," *IEEE Trans. Electron Devices*, vol. ED-30, no. 11, Nov. 1983, to be published.

[5] T. Kato and Y. Nishi, "Redistribution of diffused boron in silicon by thermal oxidation," *Japan J. Appl. Phys.*, vol. 3, p. 377–383, 1964.

[6] D. Nobili *et al.*, "Precipitation as the phenomena responsible for the electrically inactive phosphorus in silicon," *J. Appl. Phys.*, vol. 53, pp. 1484–1491, 1982.

[7] S. M. Hu, P. Fahey and R. W. Dutton, "On models of phosphorus diffusion in silicon," *J. Appl. Phys.*, to be published.

[8] S. M. Hu, "General theory of impurity diffusion in semiconductors via the vacancy mechanism," *Phys. Rev.*, vol. 180, pp. 773–784, 1969.

[9] R. B. Fair, *Impurity Doping Processes in Silicon*, F. F. Y. Wang, Ed. Amsterdam: North Holland Publishing, in press.

[10] E. Guerrero *et al.*, "Generalized model for the clustering of As dopants in silicon," *J. Electrochem. Soc.*, vol. 129, pp. 1826–1831, 1982.

[11] R. B. Fair and G. R. Weber, "Effect of complex formation on diffusion of arsenic in silicon," *J. Appl. Phys.*, vol. 44, pp. 273–279, 1973.

[12] D. A. Antoniadis and R. W. Dutton, "Models for computer simulation of complete IC fabrication processes," *IEEE J. Solid-State Circuits*, vol. SC-14, pp. 412–422, 1979.

[13] P. S. Dobson, "The effect of oxidation on anomalous diffusion in silicon," *Phil. Mag.*, vol. 24, pp. 567–576, 1971.

[14] A. M. Lin *et al.*, "The oxidation rate dependence of oxidation-enhanced diffusion of boron and phosphorus in silicon," *J. Electrochem. Soc.*, vol. 128, pp. 1131–1137, 1981; A. M. Lin, "The effect of oxidation on impurity diffusion and stacking fault growth in silicon," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1982.

[15] H-G. Lee and R. W. Dutton, "Two-dimensional low-concentration boron profiles: Modeling and measurement," *IEEE Trans. Electron Devices*, vol. ED-28, pp. 1136–1147, 1981; H-G. Lee, "Two-dimensional impurity diffusion studies: Process models and test structures for low-concentration boron diffusion," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1980.

[16] D. P. Kennedy *et al.*, "A statistical approach to the design of diffused junction transistors," *IBM Syst. J.*, vol. 8, p. 482–495, 1964.

[17] R. J. VanOverstraeten, H. J. DeMan, and R. P. Mertens, "Transport equations in heavy doped silicon," *IEEE Trans. Electron Devices*, vol. ED-20, pp. 290–298, 1973.

[18] J. Graul *et al.*, "High-performance transistors with arsenic implanted polysilicon emitters," *IEEE J. Solid-State Circuits*, vol. SC-11, pp. 491–495, 1976; T. Sakai *et al.*, "Elevated electrode integrated circuits," *IEEE Trans. Electron Devices*, vol. ED-26, pp. 379–385, 1979.

[19] D. D. Tang *et al.*, "Subnanosecond self-aligned I^2L/MTL circuits," *IEEE J. Solid-State Circuits*, vol. SC-15, pp. 444–449, 1980.

[20] H. C. deGraaff, and J. G. deGroot, "The SIS tunnel emitter: A theory for emitters with thin interface layers," *IEEE Trans. Electron Devices*, vol ED-26, pp. 1771–1776, 1979; B. Soerowirdjo *et al.*, "Influence of surface treatments on the electrical characteristics of polycrystalline emitter bipolar devices," in *IEDM Tech. Dig.*, pp. 668–671, Dec. 1982.

[21] T. H. Ning and R. D. Isaac, "Effect of emitter contact on current gain of silicon bipolar devices," in *IEDM Tech. Dig.*, pp. 475–477, Dec. 1979.

[22] Z. Yu *et al.*, "Box model of polyemitter contacts to bipolar transistors," *IEEE Trans. Electron Devices*, to be published.

[23] J. L. Moll and I. M. Ross, "The dependence of transistor parameters on the distribution of base layer resistivity," *Proc. IRE*, vol. 44, pp. 72–78, 1956.

[24] P. E. Fox *et al.*, "Statistical analysis of propagation delay in digital integrated circuits," in *Int. SSC Conf. Tech. Dig.*, pp. 66–67, Feb. 1972.

[25] D. C. D'Avanzo, "Modeling and characterization of short-channel double-diffused MOS transistors," Appendix C-SEDAN, Ph.D. dissertation, Stanford Univ., Stanford, CA, 1980.

[26] D. A. ANTONIADIS, S. E. HANSEN, AND R. W. DUTTON, "SUPREM II—A program for IC process modeling and simulation," Stanford Univ., Stanford, CA, Tech. Rep., June 1978.
[27] D. A. DIVEKAR *et al.*, "Experimental study of Gummel–Poon model parameter correlations for bipolar junction transistors," *IEEE J. Solid-State Circuits*, vol SC-12, pp. 552–559, 1977.
[28] D. A. DIVEKAR, "Device modeling for statistical circuit simulation," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1978.
[29] W. MALY AND A. J. STROJWAS, "Statistical simulation of the IC manufacturing process," *IEEE Trans. Computer-Aided Design of ICAS*, vol. CAD-1, pp. 120–131, 1982.
[30] B. E. DEAL AND A. S. GROVE, "General relationship for the thermal oxidation of silicon," *J. Appl. Phys.*, vol. 36, pp. 3770–3778, 1965.
[31] C. P. HO AND J. D. PLUMMER, "Si–SiO_2 interface oxidation kinetics: A physical model for the influence of high substrate doping levels," *J. Electrochem. Soc.*, vol. 126, pp. 1516–1522, 1979.
[32] H. Z. MASSOUD *et al.*, "Thin oxide growth kinetics in dry oxygen," in *Computer-Aided Design of Integrated Circuit Fabrication Processes for VLSI Devices*, J. D. Plummer, Ed. Stanford Univ., Stanford, CA, Tech. Rep., July 1982.
[33] L. MEI AND R. W. DUTTON, "A process simulation model for multilayer structures involving polycrystalline silicon," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 1726–1734, 1982.
[34] D. CHIN *et al.*, "Two-dimensional oxidation," to be published *IEEE Trans. Electron Devices*, vol. ED-30, pp. 744–749, July 1983; _____, "A general solution method for two-dimensional nonplanar oxidation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 993–998, 1983.
[35] R. W. DUTTON *et al.*, "Computer-aided process modeling for design and process control," in *Silicon Processing*, D. C. Gupta, Ed., ASTM, 1983 to be published.
[36] J. A. GREENFIELD AND R. W. DUTTON, "Nonplanar VLSI device analysis using the solution of Poisson's equation," *IEEE Trans. Electron Devices*, vol. ED-27, pp. 1520–1532, 1980; J. A. Greenfield, "Analysis of intrinsic MOS devices and parasitic effects using solutions of Poisson's equation," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1982.
[37] P. YANG AND P. CHATTERJEE, "Statistical modelling of small geometry MOSFET's," in *IEDM Tech. Dig.*, pp. 286–289, Dec. 1982; N. Herr *et al.*, "A statistical modeling approach for simulation of MOS VLSI circuit designs," in *IEDM Tech. Dig.*, pp. 290–293, Dec. 1982.
[38] D. CHIN *et al.*, "Process design using two-dimensional process and device simulations," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 336–340, 1982.
[39] _____, "Stresses in local oxidation," in *IEDM Tech. Dig.*, pp. 228–232, Dec. 1982.
[40] N. M. JOHNSON *et al.*, "Single crystal silicon transistor in laser-crystallized thin films on bulk glass," *IEEE Electron Device Lett.*, vol. EDL-3, pp. 369–372, 1982.
[41] E. KRULLMANN AND W. L. ENGL, "Low-pressure silicon epitaxy," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 491–497, 1982.
[42] P. A. GARGINI AND I. BEINGLASS, "WOS: Low resistance self-aligned source, drain, and gate transistors," in *IEDM Tech. Dig.*, pp. 54–57, Dec. 1981.
[43] R. D. RUNG *et al.*, "Deep trench isolated CMOS devices," in *IEDM Tech. Dig.*, pp. 237–240, Dec. 1982.
[44] L. EPHRATH, "Reactive ion etching for VLSI," in *IEDM Tech. Dig.*, pp. 402–404, Dec. 1980.
[45] L. MEI, "A review of VLSI interconnect technology," *IEEE Trans. Electron Devices*, submitted for publication.
[46] T. KAMINS, "MOS transistors in beam-recrystallized polysilicon," in *IEDM Tech. Dig.*, pp. 420–423, Dec. 1982.
[47] J. F. GIBBONS AND K. F. LEE, "A folding principle for generating three-dimensional devices in beam recrystallized polysilicon films," in *IEDM Tech. Dig.*, pp. 111–114, Dec. 1982.
[48] B. J. LIN, "Multilayer resist systems as a means to submicron optical lithography," in *IEDM Tech. Dig.*, pp. 391–394, Dec. 1982.
[49] W. G. OLDHAM *et al.*, "A general simulator for VLSI lithography and etching processes: Part II—Application to deposition and etching," *IEEE J. Solid-State Circuits*, vol. SC-15, pp. 1455–1459 1980.
[50] _____, "A general simulator for VLSI lithography and etching processes: Part I—Applications to projection lithography," *IEEE Trans. Electron Devices*, vol. ED-26, pp. 717–722, 1979.
[51] A. M. MAZZONE AND G. ROCCA, "Three-dimensional Monte Carlo simulations of implanted profiles for submicrometer device applications," *IEEE Trans. Computed-Aided Design of ICAS*, to be published.
[52] D. CHIN *et al.*, "A general solution method for two-dimensional nonplanar oxidation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 986–992, 1983.
[53] C. A. MEAD AND L. CONWAY, *Introduction to VLSI Systems*. Reading, MA: Addison–Wesley, 1980.

[54] L. W. NAGEL AND D. O. PEDERSON, "Simulation program with integrated circuit emphasis (SPICE)," presented at the *19th Midwest Symp. on Circuit Theory*, Waterloo, Ont., Canada, Apr. 1973.
[55] D. E. WARD AND K. DOGANIS, "Optimized extraction of MOS model parameters," *IEEE Trans. Computer-Aided Design of ICAS*, vol. CAD-1, pp. 163–168, 1982.
[56] P. YANG AND P. K. CHATTERJEE, "SPICE modeling for small geometry MOSFET circuits," *IEEE Trans. Computer-Aided Design of ICAS*, vol. CAD-1, pp. 169–182, 1982.
[57] M. R. PINTO, private communication.
[58] C. RAFFERTY, private communication.
[59] H. L. OSSHER AND B. K. REID, "FABLE: A programming-language solution to IC process automation problems," in *Proc. Symp. Programming Language Issues in Software Syst.*, June 1983.
[60] W. L. ENGL *et al.*, "MEDUSA—A simulator for modular circuits," *IEEE Trans. Computer-Aided Design of ICAS*, vol. CAD-1, pp. 85–93, 1982.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0002/$01.25/0

SEMICONDUCTOR DEVICE SIMULATION*

WOLFGANG FICHTNER†, DONALD J. ROSE† AND RANDOLPH E. BANK‡

Abstract. The most effective way to design VLSI device structures is to use sophisticated, complex two-dimensional (2D) and three-dimensional (3D) models. This paper and its companion [1] discusses the numerical simulation of such device models. Here we describe the basic semiconductor equations including several choices of variables. Our examples illustrate results obtained from finite-difference and finite-element implementations. We stress the necessary 3D calculations for small-size MOSFET's. Numerical results on inter-electrode capacitive coupling are included.

I. Introduction. The development of new semiconductor technologies and novel semiconductor device structures has traditionally been guided by an experimental approach. Starting from an *established* process sequence, fabrication steps are changed together with geometrical (feature size) dimensions. The *modified* process is then realized by fabricating several lots. Finished devices are tested to insure their performance conforms to the initial design. This approach usually includes several iterations of the processing to testing loop.

With the advent of increasingly complex integrated circuits, the traditional empirical approach has become expensive and time consuming. An alternative approach using sophisticated numerical simulations in process and device development has proved to be both cost effective and reliable.

For example, the development of a new CMOS process might involve nine lithography steps, six ion implantations and several diffusion, annealing, and oxidation steps. In a medium-size computer, one can simulate all critical process steps and the corresponding device performance in a matter of minutes to hours. A *real* experiment, on the other hand, would usually take from several weeks to months.

For devices and circuits of VLSI complexity, process conditions are tightly coupled to the behavior of the finished device. Therefore, *device simulation* cannot be a stand alone field, but has to be closely coupled to *process simulation*.

In this paper and its companion [1] to follow we present our approach to semiconductor device simulation. We assume that the necessary input from process simulation is available. We do not treat process simulation extensively here, although many of the numerical techniques presented in [1] are applicable. See other papers in this issue [2] and [3] for a summary of the current state of process simulation.

The coupled nonlinear partial differential equations describing the intrinsic behavior of semiconductor devices provide a significant challenge to the scientific computing community. While most of the work in this field has been performed by researchers with electrical engineering and physics backgrounds, there has recently been increased interest by numerical analysts in investigating this problem. As a result advanced numerical algorithms such as sparse-matrix techniques and adaptive multigrid methods have been applied to solve the semiconductor equations.

The guiding principles in the design of our software package were robustness, speed, and easy user access. Robustness is an absolute necessity for any software creator, otherwise his codes will not be used at all. In our computing environment with a large number of active users from different areas, robustness has priority. Neverthe-

*Received by the editors June 2, 1983.

†Bell Laboratories, Murray Hill, New Jersey 07974.

‡University of California, San Diego, California 92093. The research of this author was supported in part by the Office of Naval Research under contract N00014-82-K-0197.

less, execution times were of major concern to us. The search for a *best* device under worst case operating conditions can result in tens of different runs, where one run can be a complete device $I-V$ curve. The interface to a program is crucial for obtaining easy user access. We have attempted to relieve the user completely from any aspects of information transfer between the process and the device simulator, grid generation and numerical fine-tuning.

In Section II, we present the semiconductor equations, a set of coupled nonlinear partial differential equations (PDE's). After we formulate the equations, we discuss boundary conditions. The description of high-field phenomena and heavy doping effects is also included in this section.

Section III deals mainly with the simulation of intrinsic silicon device structures. We state our approach to hierarchical simulation as a cost effective albeit physical way to analyze devices. Several examples are included in this section, illustrating results obtained from finite-difference and finite-element implementations. Furthermore, we present methodology and results from three-dimensional (3D) simulations. We stress the necessity of 3D calculations for small-size MOSFET's. Part of this section is also devoted to user-oriented device simulation. One example illustrates how we have coupled a general purpose two-dimensional process simulator to our device simulation software.

Parasitic elements begin to have an increased impact on device and circuit performance. In Section IV, we shall present numerical results on inter-electrode capacitive coupling.

II. Semiconductor equations. In this section we formulate the partial differential equations (PDE's) which describe the static and dynamic behavior of carriers in semiconductors under the influence of external fields. We state the equations in a general form valid for all semiconductors of practical importance, although our primary interest is in silicon devices. However, the following discussion is *not* restricted to silicon; by using the appropriate material constants the equations can be easily transformed and generalized. We state various boundary conditions which are encountered in *real* device simulations. Several changes of variable which simplify the PDE system are discussed. Finally, we briefly discuss how hot-carrier effects and high-doping phenomena can be incorporated.

A. Equation formation. The equations which describe the transport of electrons and holes in semiconductor devices can be derived in a straightforward way from the *Boltzmann Transport Equation* (BTE), if the motion of carriers is treated to be *semiclassical* [4]. In this simplified, but nevertheless physical description, carrier motion in a semiconductor crystal with applied external field can be regarded as a series of acceleration (treated by *classical* mechanics) and scattering (treated by *quantum* mechanics) events. For the electric field strengths and temperatures encountered in small silicon devices, this semiclassical picture accounts very well for all carrier transport effects of interest (e.g., [4]–[6]).

The applicability of the BTE for the description of small semiconductor structures has been recently studied by various authors [7]–[9]. Based on the results of these papers, we can conclude that for silicon devices with *active* dimensions of 0.1 μm and larger, carrier transport can be described by the semiclassical BTE approach. The method of solution of the BTE is guided by the active dopant concentration level, which is normally well above 10^{16} cm^{-3} for small devices to insure proper current–voltage behavior. At these doping levels, however, it has been shown that carrier-carrier

interactions dominate the shape of the carrier distribution function [10]–[12]. This fact allows the assumption of an effective carrier temperature. Recently, Hess [13] has critically reviewed the applicability of the carrier temperature concept for the simulation of silicon device structures. His results indicate that for MOS devices and CCD's, the carrier temperature concept is well suited to describe high-field transport effects.

Based on these assumptions in the semiclassical picture, the operation of devices can be evaluated by solving the BTE [14]–[16], although for real applications the direct solution of the BTE is impractical. However, we can go one step further and make the quasi-static local field approximation [17], which reduces the problem from one in space and momentum coordinates to one in space coordinates alone. In this approximation, it is assumed that the response of carriers to a change in the electric field is much faster than the effective rate of change in the field; i.e., the device response and the dielectric relaxation processes in the device interior are slow compared to the fundamental response of the carriers. As a consequence, the product of the maximum particle velocities and relaxation times is negligible compared to active device dimensions, especially for the case of silicon.

Assuming the above conditions hold, we can write the basic equations of semiconductor transport in the form most commonly used in numerical device simulations [18]–[20].

The static and dynamic behavior of carriers under the influence of external fields can be partitioned into three groups: Maxwell's equation, the current-density relations and the carrier continuity equations.

a) The *Maxwell* equations form the basis of all electromagnetic phenomena in semiconductors. They are given by

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t}, \\ \nabla \times \mathbf{H} &= \frac{\partial \mathbf{D}}{\partial t} + J_{\text{cond}} = J_{\text{tot}}, \\ \nabla \cdot \mathbf{D} &= \rho, \\ \nabla \cdot \mathbf{B} &= 0 \end{aligned} \tag{1}$$

where the electric displacement vector $\mathbf{D}$ and the electric field vector $\mathbf{E}$ are related by the constitutive relation

$$\mathbf{D} = \varepsilon_s \mathbf{E}$$

with the semiconductor permittivity ε_s. A similar relation holds between the magnetic induction vector $\mathbf{B}$ and the magnetic field vector $\mathbf{H}$

$$\mathbf{B} = \mu \mathbf{H}$$

with the permeability μ. In (1), $\mathbf{J}_{\text{cond}}$ and $\mathbf{J}_{\text{tot}}$ are the total current and the conduction current, respectively; ρ is the total charge density.

When combined with the current equations, (1) provide a complete description of carrier dynamics in semiconductor structures.

For our purpose, the Poisson equation is of major interest. It relates the total space-charge to the divergence of the electric field, namely

$$\varepsilon_s \nabla \cdot \mathbf{E} = -\varepsilon_s \nabla^2 \psi = \rho \tag{2}$$

where the electrostatic potential ψ is defined via

$$\mathbf{E} = -\nabla \psi$$

and the total space-charge ρ, assuming complete ionization, is given by

$$\rho = -q(n - p + N) \tag{3}$$

where $N = N_D^+ - N_A^-$ is the electrically active net impurity concentration, q is the electric charge, and n and p are the electron and hole carrier densities.

b) The *current equations* for electrons and holes are given by

$$\mathbf{J}_n = q\mu_n n\mathbf{E} + qD_n \nabla n, \tag{4}$$

$$\mathbf{J}_p = q\mu_p p\mathbf{E} - qD_p \nabla p \tag{5}$$

where μ_n and μ_p are the electron and hole mobility, and D_n and D_p are the corresponding diffusion coefficients. Both mobilities and diffusion coefficients depend on the electric field.

Under nondegenerate conditions, the diffusion coefficients and the mobilities are related by the *Einstein* relation [5], [21], [22]

$$D_n = \mu_n \frac{kT}{q}, \qquad D_p = \mu_p \frac{kT}{q} \tag{6}$$

where k is *Boltzmann's* constant and T is the carrier temperature.

For nondegenerate materials, if the electrostatic potential is taken with reference to the intrinsic Fermi level E_i, the carrier densities are given by the Boltzmann approximation

$$n = n_i \exp\left[\frac{q(\psi - \phi_F)}{kT}\right], \tag{7}$$

$$p = n_i \exp\left[\frac{q(\phi_F - \psi)}{kT}\right] \tag{8}$$

where n_i is the intrinsic density and $E_F = q\phi_F$ is the Fermi level position under equilibrium conditions. From (7) and (8), it follows immediately, that

$$np = n_i^2. \tag{9}$$

Under nonequilibrium conditions, the electron and hole concentrations will depart from their equilibrium values and they can no longer be represented via the single quantity E_F. However, we may introduce two new energy parameters E_{Fn} and E_{Fp}, which allow us to write n and p in a form similar to that in (7) and (8), namely

$$n = n_i \exp\left[\frac{q(\psi - \phi_n)}{kT}\right], \tag{10}$$

$$p = n_i \exp\left[\frac{q(\phi_p - \psi)}{kT}\right] \tag{11}$$

where $E_{Fn} = q\phi_n$ and $E_{Fp} = q\phi_p$. The quantities E_{Fn} and E_{Fp} are the *quasi-Fermi levels* (also called *imrefs*), and ϕ_n and ϕ_p are the corresponding *quasi-Fermi potentials*.

The simple relationship in (9) is replaced by

$$np = n_i^2 \exp\left[\frac{E_{Fn} - E_{Fp}}{kT}\right] \tag{12}$$

and the difference in the quasi-Fermi levels indicates the deviation of the np-product from its equilibrium value.

Rewriting (4) and (5), using (6), (10), and (11), yields for the current densities in

terms of quasi-Fermi levels

$$\mathbf{J}_n = -q\mu_n n \nabla \phi_n, \tag{13}$$

$$\mathbf{J}_p = -q\mu_p p \nabla \phi_p. \tag{14}$$

c) The *continuity equations* for electrons and holes are given by

$$-\frac{1}{q}\nabla\cdot\mathbf{J}_n - G + R + \frac{\partial n}{\partial t} = 0, \tag{15}$$

$$\frac{1}{q}\nabla\cdot\mathbf{J}_p - G + R + \frac{\partial p}{\partial t} = 0 \tag{16}$$

where G incorporates generation phenomena, such as impact ionization or carrier generation by external radiation and R describes recombination processes. We shall comment on specific expressions for these terms in Subsection II-D.

Equations (1), (15), (16) with either (4) and (5) or (13) and (14) describe the current flow in the semiconductor and determine the electrical performance of the devices.

In this paper we are concerned with the numerical solution of the steady-state semiconductor equations. Thus we summarize (1)–(6) as

$$\varepsilon_s \nabla\cdot\mathbf{E} = q(p - n + N), \tag{17}$$

$$-\frac{1}{q}\nabla\cdot\mathbf{J}_n = \frac{1}{q}\nabla\cdot\mathbf{J}_p = G - R, \tag{18}$$

$$\mathbf{J}_n = q\mu_n n\mathbf{E} + qD_n\nabla_n, \tag{19}$$

$$\mathbf{J}_p = q\mu_p p\mathbf{E} - qD_p\nabla p. \tag{20}$$

Equations (17)–(20) are usually normalized into dimensionless form. We have followed the work of de Mari [23]; several other ways exist and have been reported e.g., [24]. If we express all spatial dimensions in terms of the intrinsic *Debye* length,

$$L_i = \sqrt{\frac{\varepsilon_s k_B T}{q n_i}},$$

all densities in units of n_i and all voltage terms in units of kT/q, (17)–(20) read

$$-\Delta u + n - p - k = 0, \tag{21}$$

$$-\nabla\cdot\mathbf{j}_n = \nabla\cdot\mathbf{j}_p = g - r, \tag{22}$$

$$\mathbf{j}_n = -\mu_n[n\nabla u - \nabla n] = -\mu_n n\nabla v, \tag{23}$$

$$\mathbf{j}_p = -\mu_p[p\nabla u + \nabla p] = -\mu_p p\nabla w, \tag{24}$$

with

$$n = e^{u-v}, \qquad p = e^{w-u}. \tag{25}$$

Here u, v, and w are the normalized potential and the electron and hole quasi-Fermi level, respectively, and k is the normalized impurity distribution. In the transition from (7)–(20) to (21)–(25), we have kept the symbols n, p, and μ_n and μ_p. The reader should be aware that in (21)–(25), these are actually normalized variables.

The PDE system (21)–(25) is posed on a region (in 2D) such as shown in Fig. 1. For MOS technology the gate contact is insulated from the channel between the source and drain contacts in the silicon region by SiO_2. In this oxide region, (21) is augmented

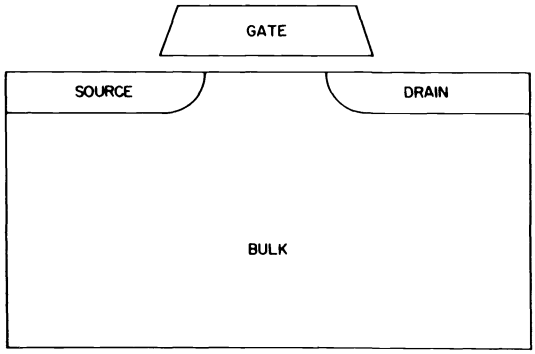


FIG. 1. *Idealized cross section of a planar* n-*channel* MOSFET.

by

$$-\varepsilon_0 \Delta U = 0. \tag{21a}$$

Equation (24) is normalized form, and ε_0 is the ratio of permittivities in silicon-dioxide and silicon.

B. Boundary conditions. Equations (21)–(25) summarize the coupled system of PDE's describing the semiconductor device. It remains to specify boundary conditions for a particular geometry.

Fig. 1 shows a representative example a two-dimensional cross section of an n-channel MOS device. For the ohmic contacts of the source, drain, and bulk we derive Dirichlet boundary conditions as follows. Assuming *space-charge neutrality* on these contacts ($\rho = 0$ in (23)) we obtain

$$n - p - k = 0. \tag{26}$$

Carrier equilibrium defines the condition

$$pn = 1. \tag{27}$$

Recalling (25) we obtain $v = w = u_F$, where u_F is the normalized equilibrium Fermi level (see (7) and (8)). Finally u_F is taken to be equal to the applied voltage at the contact; e.g., for the drain

$$v = w = u_{DS} \tag{28}$$

where u_{DS} is the normalized drain-to-source voltage.

From (26) and (27) we find

$$n = k/2 + \sqrt{(k^2/4)+1}$$

which combined with (25) and (28) produces the Dirichlet condition at the drain

$$u = u_{DS} + \ln\left(k/2 + \sqrt{(k^2/4)+1}\right). \tag{29}$$

The other contacts are treated similarly.

Along the side boundaries, homogeneous Neumann (reflecting) boundary conditions insure that no current can flow in or out of the device along these edges. This statement is equivalent to equality of drift and diffusion along the sides.

The gate contact is treated as a Dirichlet boundary condition with an appropriate work function for the material. Along the side of the gate oxide homogeneous Newmann boundary conditions are used.

C. Choice of variables and earlier work. The origin of semiconductor device modeling using numerical instead of analytical methods can be traced back to the early work of Gummel [25] on one-dimensional bipolar transistor modeling. This algorithm has since been extensively used both in its original one-dimensional form and also in two dimensions by Slotboom [26], [27], Mock [28], [29], Heimeier [30], Manck [31], and others. The major difference between the various attempts to solve the semiconductor equations lies in the choice of variables and the treatment of the carrier continuity equations [32]. In our companion paper, we present a discussion of the various possibilities that exist.

The basis for Gummel's algorithm [25] is as follows. Estimates are made for the quasi-Fermi level distributions v and w, and a solution for u is obtained from Poisson's equation. The solution for u so found is then used in the transport equations to improve the initial estimates of w and v. The cycle is repeated until some convergence criterion is satisfied. The important point is that the equations are decoupled. We shall call this solution process Block Nonlinear Iteration (BNI). Although Mock [33] has claimed that under certain circumstances the algorithm does not converge, the scheme itself has proved to be of considerable value.

Some workers have chosen the carrier concentration n and p as variables in which to apply the Gummel algorithm, for example de Mari [23], Petersen [34], Manck [31]. This choice of variables is attractive as it can be seen that if an estimate for u is known, (22) and (23) are linear in n and p, and that given an estimate for n and p, then (21) is linear in u. Consequently, only successive solutions of linear problems are required. However, because of the large range of values to be accommodated, care must be taken in the discretization scheme, as small errors may produce inadmissible (and unphysical) negative carrier concentrations. In addition, large spatial variations make an adequate numerical representation of derivatives difficult.

Others [26], [27], [35], [36], [37] used exponentials of the quasi-Fermi levels as dependent variables

$$\mathbf{j}_p = -\mu_p p \nabla w = -\mu_p e^{-u} \nabla e^{w} = -\mu_p e^{-u} \nabla \omega, \tag{30}$$

$$\mathbf{j}_n = -\mu_n n \nabla v = \mu_n e^{u} \nabla e^{-v} = \mu_n e^{u} \nabla \nu. \tag{31}$$

The BNI algorithm produces linear transport equations in the new variables $\nu = e^{-v}$ and $\omega = e^{w}$, although the Poisson equation is nonlinear. An advantage of this scheme is that the quantities ν and ω are nearly constant over large regions of the device, thus allowing a reduction of meshpoints. The range of numerical values, however, is greater than that in n and p. In our companion paper [1], we suggest several strategies to scale the original equations (30) and (31).

Mock [28], Sutherland [38], and others [39], [40] have adopted the so-called stream function as a variable in the continuity equations. The main assumption is that no generation and recombination process exist, i.e.

$$\nabla \cdot j_n = 0, \tag{32}$$

$$\nabla \cdot j_p = 0. \tag{33}$$

This divergence-free flow of the electron and hole current can be enforced by introducing a stream-function θ as a vector potential for j, e.g., for the electrons

$$j_n = \hat{\mathbf{J}}_n \nabla \times \theta_n \tag{34}$$

where $\hat{\mathbf{J}}_n$ is the total current. For a two-dimensional problem the components of j_n are,

respectively

$$j_{n,x} = \hat{\mathbf{J}}_n \frac{\partial \theta_n}{\partial y}, \tag{35}$$

$$j_{n,y} = -\hat{\mathbf{J}}_n \frac{\partial \theta_n}{\partial x}. \tag{36}$$

Identifying the x and y components of (24) with the corresponding expressions of (35) and (36) we have

$$\hat{\mathbf{J}}_n \frac{\partial \theta_n}{\partial y} = \mu_n e^u \frac{\partial}{\partial x}(ne^{-u}), \tag{37}$$

$$\hat{\mathbf{J}}_n \frac{\partial \theta_n}{\partial X} = -\mu_n e^u \frac{\partial}{\partial y}(ne^{-u}) \tag{38}$$

using $\partial/\partial x(ne^{-u}) = -e^{-v}(\partial v/\partial x)$ and (24).

By dividing both sides of these intermediate equations by $\mu_n e^u$, and differentiating the first equation partially with respect to y, the second partially with respect to x, and adding the two, we obtain

$$-\frac{\partial}{\partial x}\left\{\mu_n^{-1} e^{-u} \frac{\partial \theta_n}{\partial x}\right\} - \frac{\partial}{\partial y}\left\{\mu_n^{-1} e^{-u} \frac{\partial \theta_n}{\partial y}\right\} = 0. \tag{39}$$

A similar equation holds for the holes. The iterative procedure starts with initial guesses for u, θ_n, and θ_p. Then the nonlinear Poisson equation is solved, followed by a solution of (39) and the equivalent hole equation. The stream-function approach has recently been generalized to include generation–recombination processes [41].

One alternative to the above mentioned schemes is to work in the quasi-Fermi levels themselves, as defined in (9) and (10). This is a natural choice because, in addition to a range compression, the current equations simplify to some extent and also the boundary conditions are directly expressed in v and w and are therefore simple. The arising difficulty is that the continuity equations are nonlinear in v and w when an estimate for u is available, so that with this choice of variables all equations are nonlinear. However, with a proper nonlinear equation solver this is a small price to pay. If a simultaneous solution scheme is used, this choice of variables seems preferable.

A problem usually encountered with the decoupled solution scheme is slow convergence of the overall solution under certain physical conditions such as boundary conditions in high-current regimes. Physically, this implies a strong coupling between the Poisson and the continuity equations.

Historically, Manck [42] recognized this problem first and tried to avoid it by solving the equations simultaneously using a LSOR-scheme to solve the linearized system. Adler [43], and Buturla *et al.* [44] presented solutions of the coupled system using sparse matrix techniques. Recently, we have presented an alternative approach to the simultaneous solution of (21) to (23) [45]. By ordering the physical variables u, v, and w in blocks, we can utilize the advantages of both a global Newton strategy and efficient linear equation solution (for details, see [1]).

D. High-field phenomena and heavy doping effects. In (22) and (23), the terms g and r describe the variable generation–recombination phenomena which occur in silicon devices. The relative importance of recombination phenomena depends on the particular device; e.g., for bipolar devices, recombination strongly influences the current gain, but for MOS devices, it is usually not of importance. On the other hand, it

is imperative to include the avalanche generation term under high field conditions. Furthermore, under strong breakdown conditions, recombination phenomena become important due to the large increase of electron and hole densities. For complex structures, it is not always possible to choose the "right" physics *a priori* for the simulation, since the intrinsic behavior of the structure might change for different operating points. In Section III-B, we present as an illustrative example the case of parasitic structure which is typical for a CMOS bulk technology.

We can partition the various phenomena according to

$$g-r=(g-r)_{\text{thermal}}+(g)_{\text{av}}+(g-r)_{\text{Auger}} \tag{40}$$

where $(g-r)_{\text{thermal}}$ describes thermal processes in the volume and at the surface (Shockley–Read–Hall processes), the second term describes generation of carriers due to impact ionization and $(g-r)_{\text{Auger}}$ stands for Auger phenomena [46].

For silicon, the dominant recombination mechanism is an indirect process involving a trap center in the energy gap. This process can be well modeled by the expression

$$(g-r)_{\text{thermal}}=\frac{n_i^2-pn}{\tau_p(n+n_t)+\tau_n(p+p_t)} \tag{41}$$

where τ_n and τ_p are the electron–hole lifetimes and n_t and p_t are associated with defect levels in the energy gap. The most effective trap center occurs in the middle of the gap, and in this case $n_t=p_t=n_i$.

Surface recombination effects are modeled similar to (40), where the lifetimes are approximated by reciprocal surface velocities.

The term g_{av} describes the number of generated electron–hole pairs per unit volume and time

$$(g)_{\text{av}}=\frac{1}{q}(\alpha_n|j_n|+\alpha_p|j_p|) \tag{42}$$

where α_n and α_p are the ionization coefficients and j_n and j_p are the current densities in (24) and (25).

The coefficients α_n and α_p depend on the electric field. For multidimensional simulations, we treat the field dependence of α by the relations

$$\alpha_{n,p}=\begin{cases} A_n\exp\left(-\dfrac{\mathbf{B}_{n,p}|j_{n,p}|}{\mathbf{E}\cdot j_{n,p}}\right), & \mathbf{E}\cdot j_{n,p}>0, \\ 0, & \mathbf{E}\cdot j_{n,p}\leqslant 0, \end{cases} \tag{43}$$

thus taking only the field component parallel to the current flow into account.

Auger recombination is the reverse process to impact ionization, involving three particles. The recombination of an electron–hole pair releases energy for a third particle. The probability for this process is proportional to the density of *involved* particles, i.e.,

$$(g-r)_{\text{Auger}}=(n_i^2-pn)(C_n n+C_p p) \tag{44}$$

with the Auger coefficients C_n and C_p. It is obvious from (44) that Auger recombination will only become important if the electron and the hole density are high in the same area.

In various silicon devices, the active device regions contain doping levels above 10^{18} cm^{-3}. This is particularly true for bipolar devices, where heavily doped emitter-base

regions have a crucial influence on the transistor behavior. For essentially all *devices*, in which high doping levels are necessary, the transport of minority carriers is essential.

At these high doping levels, several phenomena occur which are unique. These phenomena can be classified in 1) effects concerning the band structure of silicon, 2) an increase in recombination with a corresponding decrease in minority-carrier lifetime, and 3) questions concerning the validity of Maxwell–Boltzmann statistics.

In this paper, we shall not dwell on the physics of high doping phenomena a complicated and still active area of research (for a review, see [47]).

To incorporate these high doping effects into our system (21) to (25), we have to take into account Auger recombination, (44), and correct the intrinsic carrier concentration, n_i, which is replaced by an effective intrinsic concentration n_{ie}

$$n_{ie}^2 = n_i^2 \exp\left(\frac{\Delta E_g}{kt}\right) \tag{45}$$

where ΔE_g describes bandgap narrowing. The effective carrier concentration has to be used in (10) and (11)

$$n = n_{ie} \exp\left[\frac{q(\psi - \phi_n)}{kT}\right], \tag{46}$$

$$p = n_{ie} \exp\left[\frac{q(\phi_p - \psi)}{kT}\right]. \tag{47}$$

Experimental values for n_{ie} are available in the literature [48].

III. Intrinsic device simulation. In this section, we present some practical aspects of numerical device simulation. Several examples should illustrate our points. These examples have been constructed from typical simulations of MOS (metal-oxide-semiconductor) devices. In Subsection III-A, we discuss hierarchical device simulation, and Subsections III-B and III-C *give* two-dimensional and three-dimensional examples.

A. Hierarchical device simulation. In simulations one is normally interested in the behavior of semiconductor devices over a wide range of operating conditions. In the case of MOS devices, these operating conditions range from the subthreshold region to high current conditions in the saturation region, possibly including impact ionization effects. If the device operates in the subthreshold region, it is essentially close to turn-off and a very small current is flowing between source and drain (see Fig. 1). This current flow is caused by diffusion only, resulting in a small perturbation in the quasi-Fermi level distributions, i.e.,

$$\nabla \cdot j_n \approx 0, \tag{48}$$

$$\nabla \cdot j_p \approx 0. \tag{49}$$

Therefore, constant quasi-Fermi levels can be assumed for low-current operating conditions. This fact has been utilized successfully by various authors [21], [49].

The assumption of a constant minority-carrier quasi-Fermi level breaks down under operating conditions resulting in medium to high current flow. For our hypothetic n-channel example, this means that (48) no longer holds. However, if generation phenomena such as impact ionization effects can be neglected, the majority-carrier current in the substrate is zero, i.e.

$$\nabla \cdot j_p \equiv 0 \tag{50}$$

for an n-channel device and the hole quasi-Fermi level w is constant. This reduces the problem to one of solving only two equations, namely (21) and (22) with (23). This case of constant quasi-Fermi level for majority carriers is the standard case for most MOS simulations.

Under bias conditions resulting in majority-current flow, all three equations have to be solved. For MOS-like structures, these situations usually arise under high bias conditions with impact ionization.

The concept of *modeling hierarchies* can be generalized for arbitrary semiconductor structures. Dirks and Engl [50] have successfully used their *stepsolving* strategy to analyze bipolar structures.

The utilization of the hierarchical solution scheme to a complex simulation problem can result in dramatic reduction of CPU-time needed. The *bottom-up* philosophy allows to start with the lowest level solution (e.g., a Poisson solution), which is a good starting approximation to the next level, and so on. Fig. 2 presents a schematic summary of the hierarchy concept. The motivation for the use of the hierarchical solution scheme is easy to understand when the alternative for obtaining the same information—a full solution of the coupled set of equations—is considered.

In the next subsection, we present two simulation examples where we utilized the hierarchy scheme.

I	$-\Delta u + e^{u-v} - e^{w-u} - k = 0$ v,w piecewise constant
II	$-\Delta u + e^{u-v} - e^{w-u} - k = 0$ $-\nabla \cdot j_n = r$ w piecewise constant
III	$-\Delta u + e^{u-v} - e^{w-u} - k = 0$ $-\nabla \cdot j_n = r - g$ $\nabla \cdot j_p = r - g$

FIG. 2. *Hierarchical device simulation.*

B. Intrinsic device simulation. We have developed a software package for the solution of (21)–(25) which is capable of dealing with most two-dimensional device structures arising in modern IC technology. The package consists of two parts, a finite-difference (FD) code and a finite-element (FE) code. In our environment, with many users having widely varying problems, this has proved to be a good solution to the tradeoff between general software and computer resource requirements.

The finite-difference code is used for device structures comprised of unions of rectangles. We use a tensor product mesh, which is the same for all three equations. The box method (see [1], Section 3.1) is used to discretize the equations. This allows us to take some advantage of vector processing capabilities available in our CRAY-1.

The finite-element code is used for device structures with nonrectangular geometry. Our implementation is based on Bank's PLTMG [51] package. We use elements of triangular shape with linear basis functions. This package utilizes a multigrid strategy with adaptive mesh refinement. For details on the discretization, we refer the reader again to our companion paper (see [1, sec. 3.5]). The implementation of our FE package is inherently more costly in the assembly process than the FD code.

In the following, we compare both implementations on an example of a planar MOSFET structure with a 1-μm gate length, 0.25-μm junctions and a gate oxide thickness of 250 Å. This device has a constant substrate doping of 10^{15} cm^{-3} and has not been implanted in the channel. We chose bias voltages of $V_{gs} = V_{bs} = 0$ V and $V_{ds} = 1$ V. Under these conditions, the device operates close to punchthrough mode with part of the total current flowing through the bulk. The total current is $I_d = 0.7$ mA ($W = 30$ μm). The device operation is illustrated in Figs. 3–6. Fig. 3 shows a surface plot (a) and the corresponding contour plot (b) of the electrostatic potential. We note that the depletion edge (-0.3-V line) has been pushed deep into the substrate. Fig. 4 presents the surface plot (a) and contour plot (b) of the electron concentration (note the different depth scales).

Electric field plots and a gate view of the lateral current density are shown in Figs. 5 and 6.

Although the current is relatively high under these bias conditions, no impact ionization occurs due to the modest drain bias ($V_{ds} = 1$ V). Therefore, we have only solved the Poisson equation and the electron continuity equation. We used a block nonlinear iteration scheme (BNI, see above) since our FE code presently does not allow a fully coupled, simultaneous scheme. For both cases, our choice of variables results in a nonlinear Poisson equation, and a linear continuity equation (see Equation (3.38), [1]). The finite-difference run was performed on one grid level whereas the finite-element run used to multigrid levels with uniform triangle refinement.

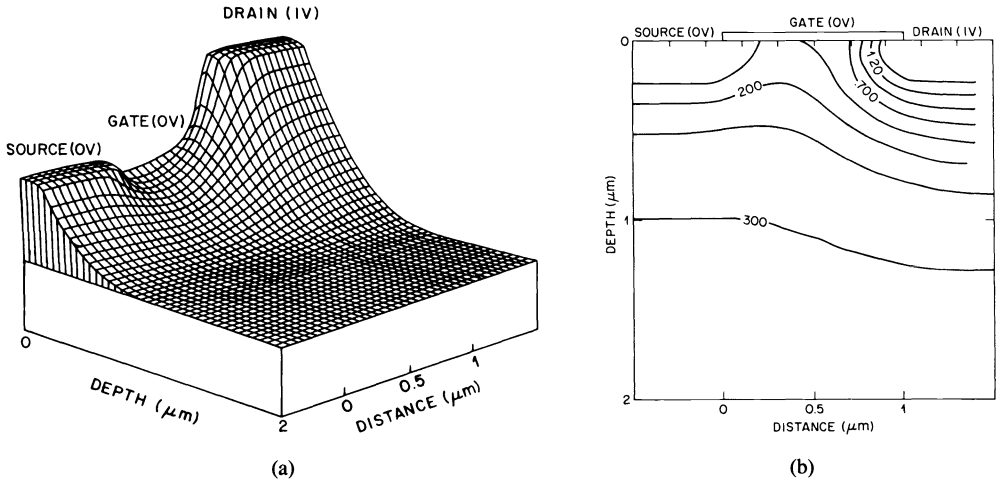


FIG. 3. (a) *Surface plot of the electrostatic potential,* (b) *contour plot as seen from the bulk for* $V_{gs} = V_{bs} = 0$ *and* $V_{ds} = 1$V.

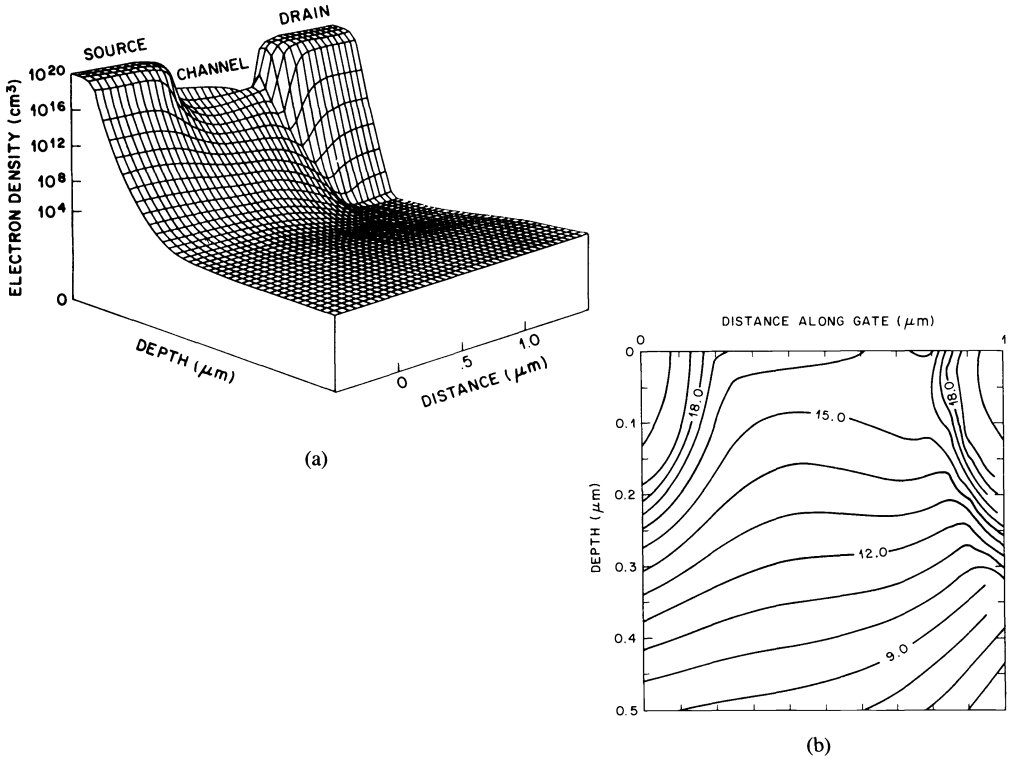


FIG. 4. (a) *Surface and* (b) *contour plot of the electron density.*

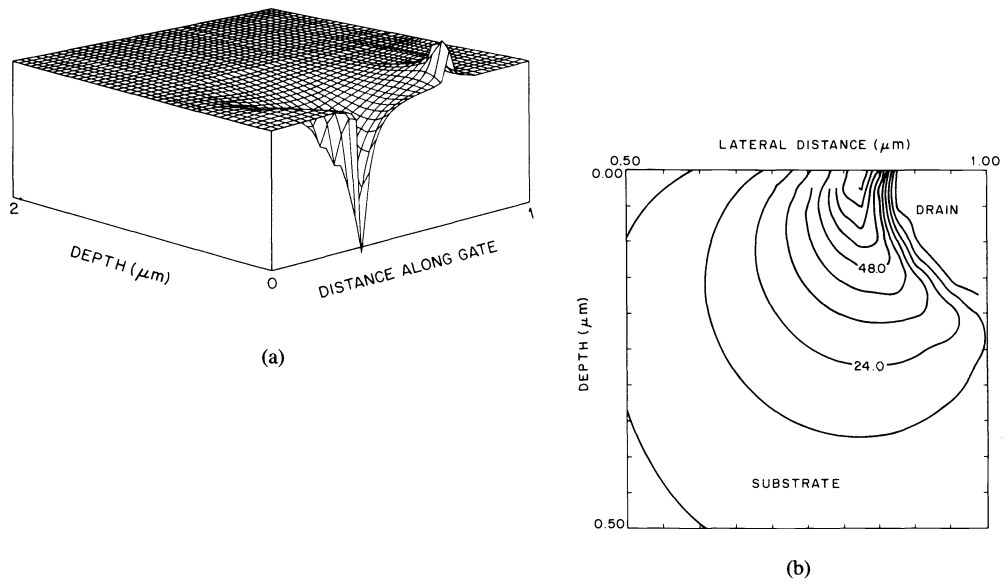


FIG. 5. (a) *Surface plot of the lateral electric field, and* (b) *the corresponding contour plot of the lateral electric field* (kV/cm) *near the drain contact.*

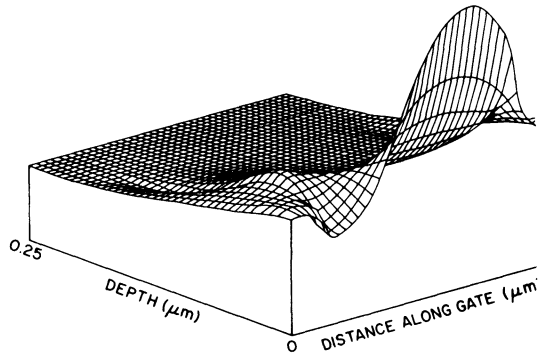


FIG. 6. *Surface plot of the electron current density, as seen through the gate.*

In Table I, we illustrate several performance aspects of the FD calculation. The nonlinear iteration scheme for the Poisson equation is the one we proposed in [53]. Initially, we solve the Poisson equation assuming constant quasi-Fermi levels (iteration = 0). This is equivalent to the lowest complexity in our solution hierarchy, and it gives a good initial condition for the electrostatic potential. Then, the electron continuity equation is solved, alternating with the Poisson equation, corresponding to next level of complexity in the solution hierarchy. The need for a stable algorithm to treat the nonlinear equations becomes obvious from the result in Table I. In order for the

TABLE I

Plug		$\|g_k\|$	$\frac{\|x\|}{\|u\|}$	t_k	function evaluations
0	0	0.17253e+04	0.00000e+00	0.10000e+01	0
	1	0.57084e+03	0.38984e+00	0.67667e+00	2
	2	0.60836e+02	0.18406e+00	0.10000e+01	1
	3	0.20709e+02	0.28712e−01	0.10000e+01	1
	4	0.65792e+01	0.86180e−02	0.10000e+01	1
	5	0.16669e+01	0.49293e−02	0.10000e+01	1
	6	0.19886e+00	0.22268e−02	0.10000e+01	1
	7	0.37518e−02	0.34863e−03	0.10000e+01	1
1	0	0.65712e+06	0.00000e+00	0.10000e+01	1
	1	0.64050e+06	0.15065e+01	0.25596e−01	4
	2	0.50558e+06	0.56559e−01	0.23063e+00	3
	3	0.15807e+06	0.13187e−01	0.10000e+01	1
	4	0.47666e+05	0.77781e−02	0.10000e+01	1
	5	0.14562e+05	0.58847e−02	0.10000e+01	1
	6	0.50547e+04	0.51173e−02	0.10000e+01	1
	7	0.17475e+04	0.43824e−02	0.10000e+01	1
	8	0.53914e+03	0.34663e−02	0.10000e+01	1
	9	0.11990e+03	0.21785e−02	0.10000e+01	1
	10	0.10935e+02	0.77098e−03	0.10000e+01	1
2	0	0.33754e+03	0.00000e+00	0.10000e+01	1
	1	0.14364e+03	0.21290e−01	0.70140e+00	2
	2	0.26625e+02	0.45202e−02	0.10000e+01	1
	3	0.31638e+01	0.13968e−02	0.10000e+01	1
	4	0.13390e+00	0.29081e−03	0.10000e+01	1
3	0	0.89378e+01	0.00000e+00	0.10000e+01	1
	1	0.32373e+01	0.61739e−02	0.10000e+01	1
	2	0.21459e+00	0.10030e−02	0.10000e+01	1
	3	0.10522e−02	0.63353e−04	0.10000e+01	1
4	0	0.19269e+01	0.00000e+00	0.10000e+01	1
	1	0.18654e+00	0.34266e−02	0.10000e+01	1
	2	0.20050e−02	0.29986e−03	0.10000e+01	1
5	0	0.68586e+00	0.00000e+00	0.10000e+01	1
	1	0.36458e−01	0.20029e−02	0.10000e+01	1
	2	0.10573e−03	0.95605e−04	0.10000e+01	1
6	0	0.29356e+00	0.00000e+00	0.10000e+01	1
	1	0.81300e−02	0.11044e−02	0.10000e+01	1
	2	0.61425e−05	0.27700e−04	0.10000e+01	1

equations to converge rapidly, it is imperative to select the right stepsize. The algorithm used allows a proper choice by reevaluating the right hand sides, which is a relatively cheap process.

The use of the hierarchical solution methodology can result in a large reduction of CPU time. In this example, eleven BNI iterations ("plugs") would have been necessary without the initial Poisson solution. For operating points resulting in high current or in cases requiring a solution of all three equations (such as avalanche breakdown) the BNI iteration strategy would be replaced by a (coupled) Newton-like iteration (see [1] and [45]).

The result obtained from the FE code is similar to the FD case. An initial triangulation with $NV = 847$ vertices and $NT = 1532$ triangles was used, which should be compared with a total number of 4070 gridpoints of the FD run.

The power of our FE package lies in its ability to model complex nonrectangular structures. To illustrate this point, we have chosen a structure which arises in twin-tub CMOS technology [54]. Fig. 7 schematically shows a cross section. Depending on the kind of dopant, this structure can contain several parasitic devices, and it can operate in various modes for different bias conditions. In Fig. 8(a) and 8(b), the parasitic element is either a p-channel field transistor (a) or a lateral p-n-p device (b). Situations c and d are equivalent to a SCR-structure and a vertical n-p-n device, respectively.

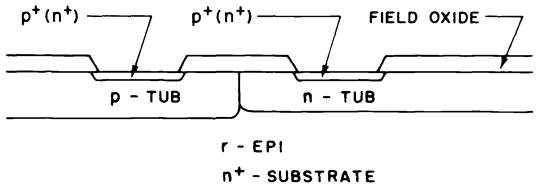


FIG. 7. *Schematic cross section of the parasitic devices which arise in* CMOS *technology.*

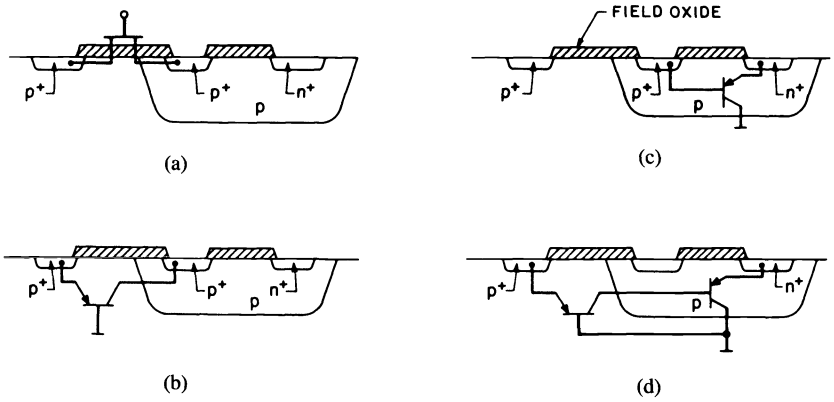
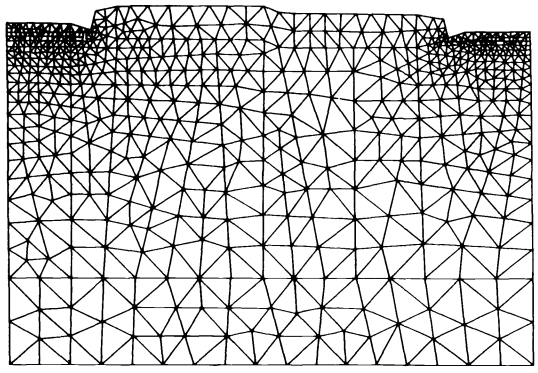
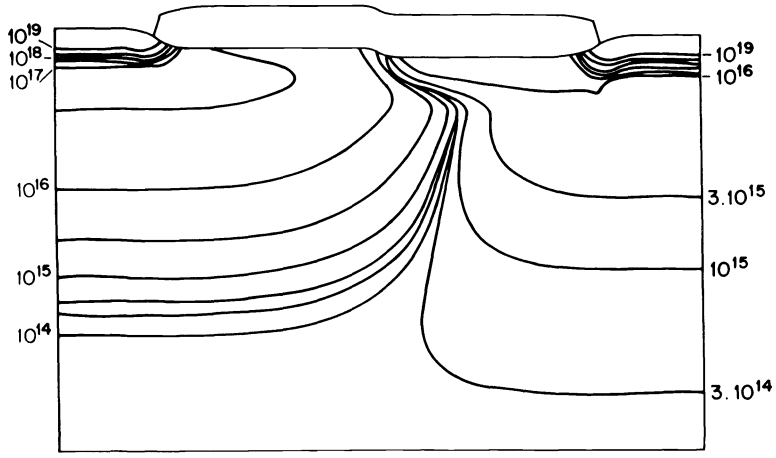


FIG. 8. (a) *Parasitic devices*: p-*channel* MOSFET, (b) *lateral bipolar* p-n-p *transistor*, (c) *vertical* n-p-n *transistor, and* (d) SCR-*structure.*



(a)



(b)

FIG. 9. (a) *Initial triangulation and* (b) *doping concentration.*

The automatically generated initial grid is shown in Fig. 9, together with the doping profile for the case in Fig. 8(a) and (b). Fig. 10 presents surface plots of the electrostatic potential for the cases in Fig. 8(a) and 8(b).

C. Numerical simulation of three-dimensional structures. Silicon devices are inherently three-dimensional (3D) structures. For most problems, however, the behavior of devices can be modeled in either one or two dimensions. This is normally the case for bipolar devices with a vertical emitter-base-collector structure or for MOS devices with a "wide" short-channel or a "small" long-channel geometry. For MOS devices with aspect ratios close to one, 3D simulation becomes necessary, especially in the subthreshold regime and under high-field conditions.

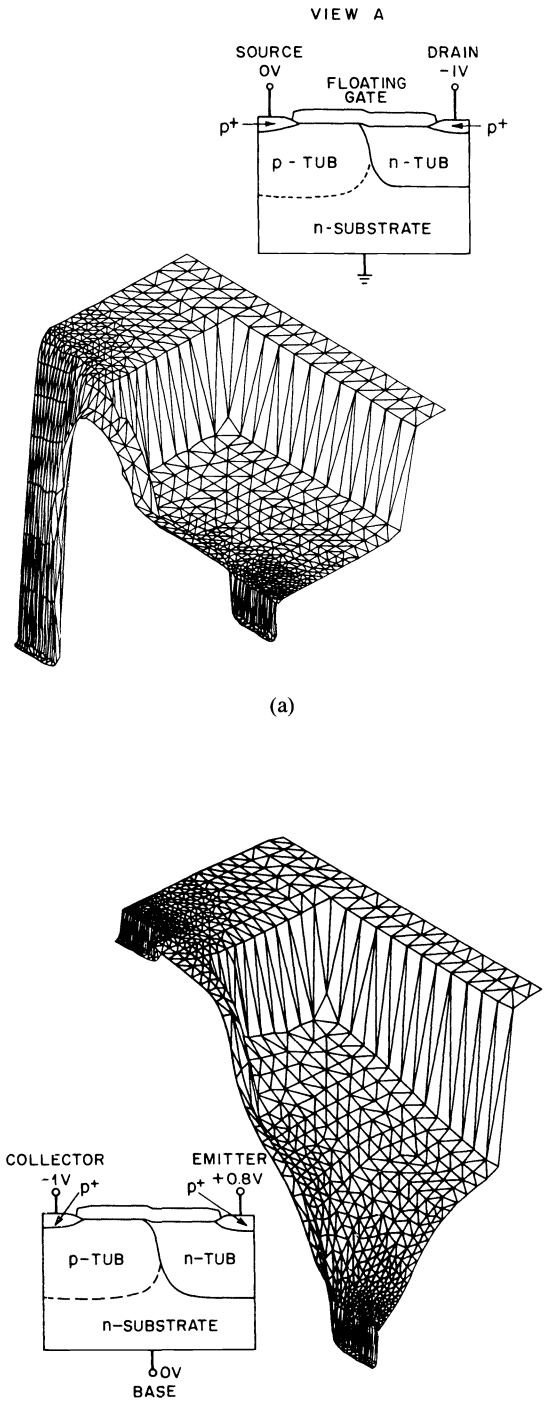


FIG. 10. *Surface plot of electrostatic potential distribution for parasitic devices in Fig.* 8(a) *and* (b).

The simulation of three-dimensional device structures presents a formidable problem requiring large computing resources. Nevertheless, several successful attempts have been reported in the engineering literature [55]–[57].

In the following, we present the basic methodology and several results of three-dimensional numerical solutions of the nonlinear system

$$h(z) = \begin{bmatrix} g_1(u,v,w) \\ g_2(u,v,w) \\ g_3(u,v,w) \end{bmatrix} = 0 \tag{51}$$

corresponding to the coupled PDE system in (21)–(25).

Fig. 11 shows the basic geometry of a planar MOSFET in three dimensions. We have employed a "standard" seven-point finite-difference scheme on *tensor-product* nonuniform grids to discretize the linearized equations. To account for the coupling along directions of current flow, the unknowns are arranged into blocks for different planes in the z-direction. The number p of planes varies between 15 and 30, depending on the operating conditions. To illustrate the interdependence of the variables, Fig. 12 shows the coupling graph $G(h')$, h' being the Jacobian of (51) for $p = 4$. Each node

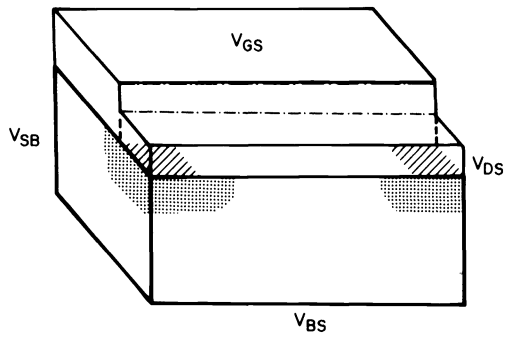


FIG. 11. *Basic geometry of a planar* MOSFET *in three dimensions.*

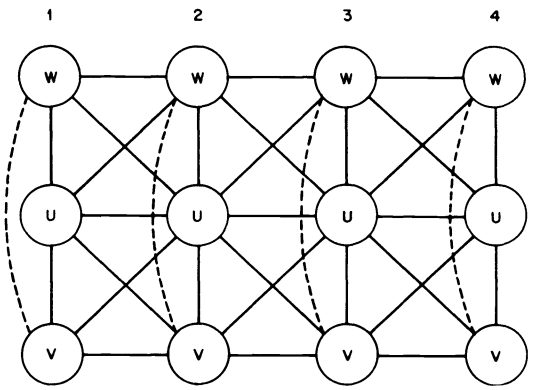


FIG. 12. *Coupling graph between variable planes.*

represents approximately 2K unknowns, 10K nonzeros and has the structure of a conventional 2D grid graph. Since the CRAY-1 has no virtual memory capabilities, we were forced to utilize out-of-core storage in the calculations and to recalculate matrix elements. In analogy to the 2D case, the linear systems of the form

$$Hx = -h$$

have been solved by block iteration. The sub-blocks corresponding to the variables u, v, and w have been solved by an iterative method (vectorized SOR with RB-ordering).

The results of 3D simulations have been successfully used to design small, submicrometer-size devices fabricated using a 1-μm NMOS process. As an example, we consider the enhancement-driver and the depletion-load device in this technology. Depletion devices are more susceptible to geometry effects than enhancement devices, because of the internal doping distribution. Fig. 13 shows a surface plot of the two-dimensional net impurity concentration profile in this device. In order to shift the threshold voltage to a proper negative value, the channel of this device is implanted with a shallow, low-dose arsenic implant. The maximum arsenic concentration in the channel is 3.10^{17} cm^{-3} with a junction depth of 800 Å. Depending on the operating conditions, the device works either in bulk mode ($V_{gs} < V_{FB}$) or in surface mode ($V_{gs} > V_{FB}$). Fig. 14 compares 2D and 3D results with experimental data for enhancement and depletion devices with a gatelength of 1 μm and different widths. In the case of the driver device we can observe a moderate short-channel effect (illustrated by the increase in I_d at higher V_{ds}) and a proper dependence of I_d on the channel width. In the case of the load device, however, we can observe a large short-channel effect (the current increases by two orders of magnitude for high V_{ds}). The width dependence of I_d

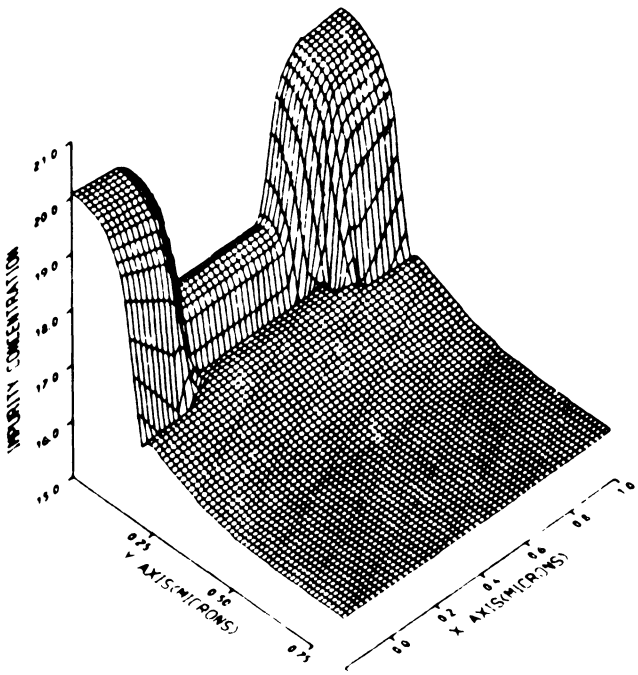
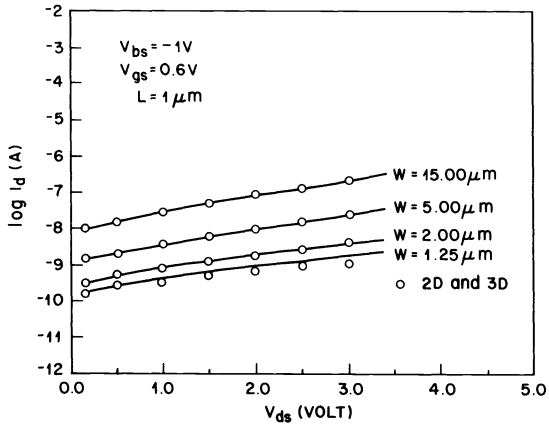
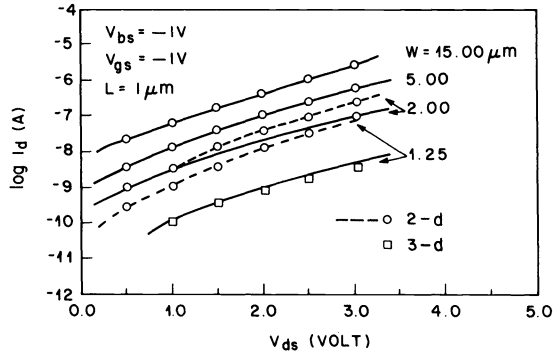


FIG. 13. *Surface plot of net impurity distribution of a depletion load* MOSFET.



(a)



(b)

FIG. 14. *Comparison of simulated* 2D *and* 3D *results with experimental data* (a) *enhancement device*, 2D *and* 3D *results are the same*; (b) *depletion device*, 2D *and* 3D *are the same for* $W = 15$, 5 *and* 2μm.

is again linear, if the width is 2 μm or larger. For smaller devices, however, the current is reduced drastically. As we can see from agreement between the simulated and experimental results, 2D simulation is certainly adequate for devices with $W \geqslant 2$ μm. Smaller devices, however, can only be modeled by a full 3D calculation. Similar conclusions have been obtained by the authors in the simulation of avalanche effects in depletion devices [56].

IV. Simulation of circuit interconnects. The ability to calculate accurately the capacitances of interconnection wires is of major importance as the circuit density increases and wire length and width decrease. For very large circuits, the total delay on an integrated circuit chip is given to a large extent by the capacitance of the interconnection wires.

In the past, capacitances of conductors and wires have been calculated using the parallel plate approximation, thus neglecting effects such as fringing and coupling. As transistors are scaled down, the width and the length of the conductor are also reduced

according to the scaling factor. However, the thickness of the conductor and that of the dielectric are not scaled down, but reduced by different factors. This fact has two serious consequences. It not only increases the two-dimensional fringe effect but results in a considerable increase of coupling capacitance where lines are neighboring.

A typical situation for today's integrated circuits is shown in Fig. 15. In this example, two levels of interconnects are used: a polysilicon level, made up of conductors 1 and 2, and an aluminum level at the top (conductors 3 and 4). Since the situation is symmetric, we shown only half of the geometry.

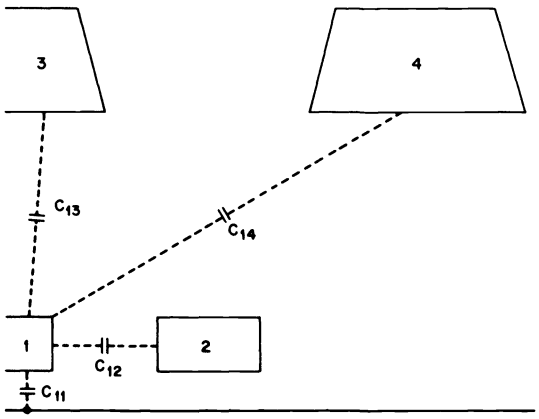


FIG. 15. *Schematic cross section illustrating a double level interconnect geometry.*

The capacitances associated with this system of conductors are given by the relation

$$Q_i = C_{i1}(V_i - V_1) + C_{i2}(V_i - V_2) + \cdots + C_{ii}V_i + \cdots + C_{in}(V_i - V_n) \tag{52}$$

where Q_i, V_i, and C_{ii} are the charge, the potential and the self-capacitance of the ith conductor, respectively; C_{ij} is the coupling capacitance between the ith and the jth conductor.

Generally, for a N-conductor problem, we obtain an $N \times N$ capacitance matrix $\mathbf{C}$ which is simply defined as a collection of all capacitances, analogous to (52). Historically, the diagonal elements of the capacitance are called coefficients of capacitance while the off-diagonal terms are called coefficients of electrostatic induction.

If Maxwell's equations hold, then (1),

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = 0, \tag{53}$$

can be rewritten in terms of a vector potential $\mathbf{A}$ using

$$\mathbf{B} = \nabla \times \mathbf{A}$$

to yield

$$\nabla \times (\nabla \times \mathbf{A}) = 0 \tag{54}$$

and since $\nabla \times (-\nabla\psi) = 0$, we obtain

$$\mathbf{E} = -\nabla\psi - \frac{\partial \mathbf{A}}{\partial t}. \tag{55}$$

In terms of the charge and current density, we can calculate the electric field at time t and at $\mathbf{r}$

$$\mathbf{E} = -\nabla\left(\frac{1}{4\pi\varepsilon}\int\frac{\rho' dr'}{|\mathbf{r}-\mathbf{r}'|}\right)-\frac{\partial}{\partial t}\left(\frac{\mu}{4\pi}\int\frac{\mathbf{J}' dr'}{|\mathbf{r}-\mathbf{r}'|}\right). \tag{56}$$

where ρ' and $\mathbf{J}'$ are the charge and current densities in the volume element dv' at time $[t-(|\mathbf{r}-\mathbf{r}'|/v)]$ with the propagation velocity v. The exact calculation of $E(\mathbf{r}, t)$ would involve the knowledge of the retarded source distributions. In integrated circuit problems, however, we can ignore retardation effects because the sources of interest are only microns away from each other, leading to propagation times much shorter than the time scales of circuit operation.

Estimating the magnitude of the two terms for $\mathbf{E}$ leads to the result that the second term may be neglected. Therefore, we can calculate $\mathbf{E}$ from

$$\mathbf{E} = -\nabla\psi. \tag{57}$$

It is therefore possible to use a solution of Laplace's equation

$$-\varepsilon\nabla\psi = 0 \tag{58}$$

to calculate the electric field. The total charge on a conductor is then obtained by integrating the normal field component along the conductor boundary.

Different techniques have been applied to analyze the interconnect problem. For simple conductor arrangements, analytical expressions [58] and numerical techniques [59] have been successfully utilized. Systems with conductors of nonrectangular shape are difficult to simulate with the above techniques. We have used a multigrid finite-element method to solve (58) for arbitrary geometries. The multigrid method lends itself naturally to problems of this kind. Fig. 16 shows the finite-element triangulation at increased levels of refinement around center conductor 1. Fig. 16(a) is the initial triangulation which is automatically generated. Fig. 16(b) and (c) show the adaptively refined meshes.

A typical result of a capacitance calculation is shown in Fig. 17 for a three-conductor problem. All variables are defined in the inset of the figure. The self-capacitance C_1, the capacitance to ground C_{11} and the coupling capacitance C_{12} have been calculated by biasing the center conductor at 1 V and by grounding all the others. The figure shows the basic problem that arises if we reduce conductor sizes. Below a certain W/H ratio, the coupling capacitance C_{12} increases above the capacitance-to-ground C_{11}. One possible way to make C_k smaller is to reduce the conductors thickness (see $T/H = 0.5$ in the figure), at the expense of an increased conductor resistance.

V. Conclusion. The development of new device technologies has traditionally been guided by an experimental approach. The use of software tools to aid process and device engineers presents a challenging alternative to the original approach. We feel that the simulation of semiconductor structures by accurate numerical models is an efficient way to optimize process conditions and the corresponding device performance.

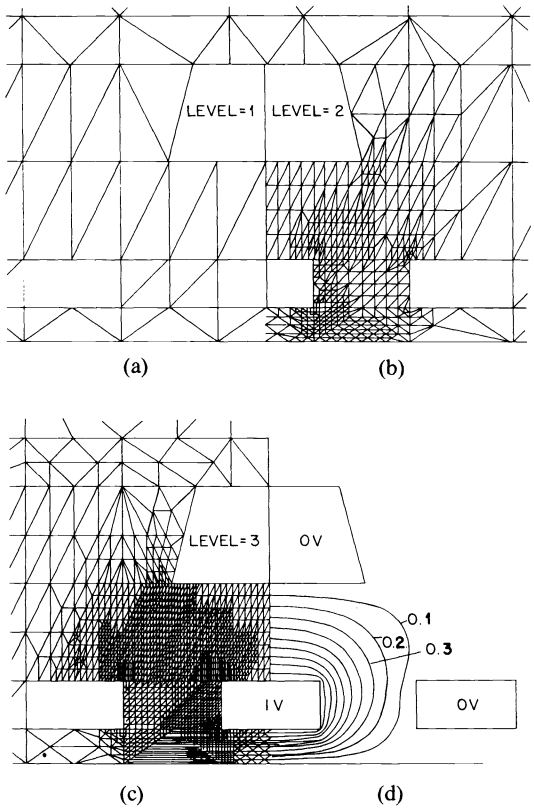


FIG. 16. *Finite element triangulation around center conductor*—(a) *initial triangulation, level one,* (b) *level two,* (c) *level three refinement, and* (d) *potential contours.*

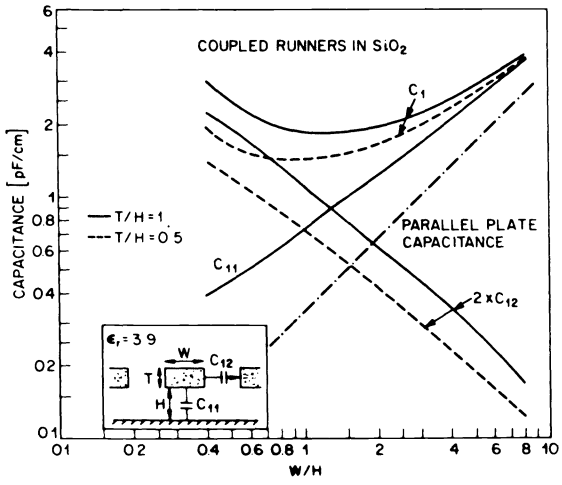


FIG. 17. *Capacitance between conductors as a function of conductor width / insulation thickness* (W/H) *for two thickness-to-height* (T/H) *ratios.*

REFERENCES

[1] R. E. BANK, D. J. ROSE, AND W. FICHTNER, "Numerical methods for semiconductor device simulator," *IEEE Trans. Electron-Devices*, vol. ED-30, pp. 1031–1041; this issue, pp. 416–435.

[2] See papers under Process Simulation, same issue.

[3] W. FICHTNER, *Process Simulation in VLSI Technology*, S. M. Sze, Ed. New York: McGraw-Hill, 1983.

[4] E. CONWELL, *High Field Transport in Semiconductors*. New York: Academic Press, 1967; see also *Handbook on Semiconductors*, vol. I, W. Paul, Ed. New York: North Holland, 1982.

[5] K. SEEGER, *Semiconductor Physics*, 2nd ed. New York: Springer, 1982.

[6] G. BAUER, in *Springer Tracts in Modern Physics*, vol. 74, G. Heber, Ed. New York: Springer, 1974.

[7] D. K. FERRY, in *VLSI Electronics — Microstructure Science*, vol. I, N. G. Einsprech, Ed. New York: Academic Press, 1981. See also D. K. Ferry, K. Hess, and P. Vogl, in *VLSI Electronics — Microstructure Science*, vol. II.

[8] K. HESS, in *Advances in Electronics and Electron Physics*, vol. 59, Peter W. Hawkes, Eds. New York: Academic Press, 1982.

[9] K. K. THORNBER, *J. Appl. Phys.*, vol. 51, p. 2127, 1980.

[10] C. J. HEARN, *Proc. Phys. Soc.*, vol. 86, 881, 1965; see also *Proc. Phys. Soc.*, vol. 88, p. 407, 1966; *Phys. Status Solidi (b)*, vol. 64, p. 527, 1974 and *Physics of Nonlinear Transport in Semiconductors*, D. K. Ferry, J. R. Barker and C. Jacoboni, Eds. New York: Plenum Press, 1980.

[11] R. STRATTON, *Proc. Roy. Soc.*, vol. A242, p. 355, 1957; see also *Proc. Roy. Soc.*, vol. A246, p. 406, 1958.

[12] H. FRÖHLICH, *Proc. Roy. Soc.*, vol. A188, p. 521, 1947; see also H. Fröhlich and B. V. Paranjape, *Proc. Phys. Soc.* (London), vol. B69, p. 21, 1956.

[13] K. HESS, in *Physics of Nonlinear Transport in Semiconductors*, D. K. Ferry, J. R. Barker and C. Jacoboni, Eds. New York: Plenum Press, 1980.

[14] P. A. LEBWOHL AND P. J. PRICE, *Solid-State Communications*, vol. 9, p. 1221, 1971.

[15] LEBWOHL,

[16] R. W. HOCKNEY AND J. W. EASTWOOD, *Computer Simulation Using Particles*. New York: McGraw-Hill, 1981, ch. 10.

[17] H. D. REES, *J. Phys. Chem. Solids*, vol. 30, p. 643, 1969.

[18] W. SHOCKLEY, *Electrons and Holes in Semiconductors*. Princeton, NJ: van Nostrand, 1950.

[19] W. VAN ROOSBROECK, *Bell Syst. Tech. J.*, vol. 29, p. 560, 1950.

[20] R. C. PRIM, III, *Bell Syst. Tech. J.*, vol. 30, p. 1174, 1951.

[21] A. H. MARSHAK, *Solid-State Electron.*, vol. 21, p. 429, 1978.

[22] R. A. SMITH, *Semiconductors*, 2nd ed. New York: Cambridge University Press, 1978, ch. 7.2.

[23] A. DE MARI, *Solid-State Electron.*, vol. 11, p. 1021, 1968; see also *Solid-State Electron.*, vol. 11, p. 33, 1968.

[24] F. DE LA MONEDA, *IEEE Trans. Circuit Theory*, vol. CT-20, p. 666, 1973.

[25] H. K. GUMMEL, *IEEE Trans. Electron Devices*, vol. ED-11, p. 455, 1964.

[26] J. W. SLOTBOOM, *Electron. Lett.*, vol. 5, p. 677, 1969.

[27] J. W. SLOTBOOM, *IEEE Trans. Electron. Devices*, vol. ED-20, p. 669, 1973.

[28] M. S. MOCK, *Solid-State Electron.*, vol. 16, p. 601, 1973.

[29] M. S. MOCK, *J. Eng. Math.*, vol. 7, p. 193, 1973.

[30] H. H. HEIMEIER, *IEEE Trans. Electron Devices*, vol. ED-20, p. 708, 1973.

[31] O. MANCK AND W. L. ENGL, *IEEE Trans. Electron Devices*, vol. ED-22, p. 339, 1975.

[32] D. SCHARFETTER AND H. K. GUMMEL, *IEEE Trans. Electron Devices*, vol. ED-16, p. 64, 1969.

[33] M. S. MOCK, *Solid-State Electron.*, vol. 15, p. 1, 1972.

[34] O. G. PETERSON, R. A. RIKOSKI, AND W. W. COULES, *Solid-State Electron.*, vol. 16, p. 239, 1973.

[35] P. DUBOCK, *Electron. Lett.*, vol. 6, p. 53, 1970.

[36] E. J. ZALUSKA, *Electron. Lett.*, vol. 9, p. 599, 1973.

[37] C. R. JESSHOPE, E. D. ZALUSKA, AND H. A. KEMHADJAN, *Electron. Lett.*, vol. 11, p. 285, 1975.

[38] A. D. SUTHERLAND, DARPA Rep. ECOM-75—1344-F-supp.

[39] T. TOYABE, K. YAMAGUCHI, S. ASAI, AND M. S. MOCK, in *IEDM Tech. Dig.*, paper 2006, Dec. 1977.

[40] E. TAKEDA, H. KUME, T. TOYABE, AND S. ASAI, *IEEE Trans. Electron Devices*, vol. ED-29, p. 611, 1982.

[41] D. N. PATTANAYAK, R. A. WILLIAMS, AND J. G. POKSHEVA, *Appl. Phys. Lett.*, vol. 41, p. 459, 1982.

[42] O. MANCK, H. HEIMEIER, AND W. ENGL, *IEEE Trans. Electron Devices*, vol. ED-21, p. 403, 1974.

[43] M. S. ADLER, in *Numerical Analysis of Semiconductor Devices*, B. T. Browne and J. J. H. Miller, Eds. Dublin, Ireland: Boole Press, 1979.

[44] E. M. BUTURLA, P. E. COTTRELL, B. M. GROSSMAN, AND K. A. SALSBURG, *IBM J. Res. Develop.*, vol. 25, p. 218, 1981.

[45] W. FICHTNER AND D. J. ROSE, in *Elliptic Problem Solves*, M. H. Schultz, Ed. New York: Academic Press, 1981.
[46] J. S. BLAKEMORE, *Semiconductor Statistics*. New York: Pergamon Press, 1962.
[47] R. P. MERTENS, R. J. VON OVERSTRAETEN, AND H. J. DE MAN, in *Advances in Electronics and Electron Physics*, vol. 55, p. 77, 1981.
[48] S. P. GAUR, G. R. SRINIVASAN, AND I. ANTIPOV, in *IEDM Tech. Dig.*, paper 11.2, 1980.
[49] J. A. GREENFIELD AND R. W. DUTTON, *IEEE Trans. Electron Devices*, vol. ED-27, p. 1520, 1980.
[50] H. DIRKS AND W. ENGL, in *Numerical Analysis for Semiconductor Devices*, B. T. Brown and J. J. H. Miller, Eds. Dublin, Ireland: Boole Press, 1981.
[51] R. E. BANK AND A. H. SHERMAN, in *Sparse Matrix Proceedings — 1978*, I. S. Duff and G. W. Stewart, Eds. Philadelphia, PA: SIAM, 1979.
[52] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*. New York: Academic Press, 1981.
[53] R. E. BANK AND D. J. ROSE, *Numer. Math.*, vol. 37, p. 279, 1981.
[54] L. C. PARILLO, L. K. WANG, R. D. SWENUMSON, R. L. FIELD, R. C. MELIN, AND R. A. LEVY, in *IEDM Tech. Dig.*, paper 29.3, 1982.
[55] E. M. BUTURLA, P. E. COTTRELL, B. M. GROSSMAN, B. M. LAWLOR, C. T. MCMULLEN, AND K. A. SALSBURG, in *IEEE Solid-State Circuits Conf. Dig. Tech.* Papers, 1980, pp. 76.
[56] W. FICHTNER, R. L. JOHNSTON, AND D. J. ROSE, in *Proc. 1981 Device Research Conf.*, Santa Barbara, CA., 1981.
[57] A. YOSHII, H. KITAZAWA, M. TOMIZAWA, S. HOIGUCHI, AND T. SUDO, *IEEE Trans. Electron Devices*, vol. ED-29, p. 184, 1982.
[58] W. H. CHANG, *IEEE Trans. Microwave Theory Tech.*, vol. MTT-24, 1976.
[59] A. RUEHLI, *IBM J. Res. Develop.*, vol. 23, p. 626, 1979.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0003/$01.25/0

NUMERICAL METHODS FOR SEMICONDUCTOR DEVICE SIMULATION*

RANDOLPH E. BANK†, DONALD J. ROSE ‡ AND WOLFGANG FICHTNER‡

Abstract. This paper describes the numerical techniques used to solve the coupled system of nonlinear partial differential equations which model semiconductor devices. These methods have been encoded into our device simulation package which has successfully simulated complex devices in two and three space dimensions. We focus our discussion on nonlinear operator iteration, discretization and scaling procedures, and the efficient solution of the resulting nonlinear and linear algebraic equations. Our companion paper [13] discusses physical aspects of the model equations and presents results from several actual device simulations.

I. Introduction. We focus our discussion here on the numerical techniques used to solve the coupled system of nonlinear partial differential equations describing the intrinsic behavior of semiconductor devices. For convenience in exposition we restrict our attention to the steady-state equations in two space dimensions, although the techniques and ideas presented are applicable to the time-dependent equations and to the system in three space dimensions. A physical description of these equations and results from numerical simulations several transistor devices are presented in our companion paper [13].

We begin in Section II by summarizing the PDE system and describing our basic approach to solving the equations. In Subsection II-B we discuss nonlinear iteration at the operator level, finding it convenient to defer until Section III discretization procedures. By considering nonlinear operator iteration we are led naturally to examine properties of the linear operators that will be solved at each iteration. For example, in Subsection II-C we examine the role of scaling and change of variables. Subsection II-D discusses aspects of grid selection and refinement and is a transition to Section III on discrete equation formulation.

We have used both a finite-difference and a finite-element approach to discretization. Subsection III-A reviews the box method for obtaining finite-difference equations. Subsections III-B and III-C apply the box method to derive the discrete equations for our particular PDE systems, and Subsection III-D compares these discrete equations with the commonly used, physically motivated discretization of Scharfetter and Gummel [16]. Our finite-element approach is presented in Subsection III-E. Although we use piecewise linear elements on triangles here as contrasted with rectangular-oriented finite differences, nevertheless, the two approaches lead to discrete systems with nearly identical properties when the finite-element quadrature rule is chosen to correspond to a generalized box-type method. Finally in Subsection III-F we make some comparative remarks on methods for solving the discrete equations.

II. Equations and basic methodology. In this section we present the central ideas used in our numerical approaches to solving the system of nonlinear PDE's modeling semiconductor devices. The discussion here is descriptive and methodologically oriented with pointers to the published literature included for many details of the algorithms and their analyses. After summarizing the equations, we focus on the issues of nonlinear iterations, choice of variables and scaling, and grid continuation.

*Received by the editors June 6, 1983.

†University of California at San Diego, California 92093. The research of this author was supported in part by the Office of Naval Research under contract N00014-82-K-0197.

‡ Bell Laboratories, Murray Hill, New Jersey 07974.

A. Equations. We begin by summarizing the static system of PDE's derived in [13]. We consider them in dimensionless form with mixed Neumann and Dirichlet conditions as described there. Furthermore, the Laplace equation, $-\varepsilon_1 \Delta u = 0$, and the constant ε_2 are explicitly suppressed and considered implicitly in the Poisson equation $g_1 = 0$ following, although they will be treated in more detail during discretization.

As in [13] we consider the coupled system of PDE's

$$g_1(u, n, p) = -\Delta u + n - p - k_1 = 0, \tag{1a}$$

$$g_2(u, n, p) = -\nabla \cdot j_n + k_2 = 0, \tag{1b}$$

$$g_3(u, n, p) = \nabla \cdot j_p + k_2 = 0. \tag{1c}$$

The continuity equations, (1b), (1c), are further specified by writing the current densities j_n and j_p in the (drift-diffusion) form

$$j_n = -\mu_n n \nabla u + D_n \nabla n, \tag{2a}$$

$$j_p = -\mu_p p \nabla u - D_p \nabla p \tag{2b}$$

where the mobility and diffusivity coefficients, μ_i and D_i, $i = n, p$, respectively, may be field dependent as well as spatially dependent. The doping profile k_1 is assumed to be spatially dependent only, while the recombination term k_2 is usually of the form $k_2 = k_2(u, n, p)$. Our methods apply to two- and three-dimensional geometries.

When the Einstein relation is valid, it takes the form $\mu_i = D_i$, and we write

$$n = e^{u-v}, \tag{3a}$$

$$p = e^{w-u}. \tag{3b}$$

Using these quasi-Fermi variables v and w, we may write (1) as

$$g_1(u, v, w) = -\Delta u + e^{u-v} - e^{w-u} - k_1 = 0, \tag{4a}$$

$$g_2(u, v, w) = \nabla \mu_n e^{u-v} \nabla v + k_2 = 0, \tag{4b}$$

$$g_3(u, v, w) = -\nabla \mu_p e^{w-u} \nabla w + k_2 = 0. \tag{4c}$$

B. Nonlinear iterations. There are two basic approaches to solving the system (1) or (4) which dominate the literature on device simulation. The first approach would be known to numerical analysts as a nonlinear operator Gauss–Seidel/Jacobi (G-S/J) iteration. The iteration associates with each g_i the highest-order differential dependent variable, e.g., u in (1a) and (4a), as the "output" variable for that equation. Given values of the dependent variables (including possibly the output variable itself), the operator equation $g_i = 0$ is solved (approximately) to obtain a new approximation to the output variable. Consider the formulation (4). Symbolically we can write the iteration as follows. Let $z_k = (u_k, v_k, w_k)^T$, z_0 given. The iteration proceeds by looping through solutions of

$$g_1(z_k \to u_{k+1}) = 0, \tag{5a}$$

$$g_2(z_k, u_{k+1} \to v_{k+1}) = 0, \tag{5b}$$

$$g_3(z_k, u_{k+1}, v_{k+1} \to w_{k+1}) = 0. \tag{5c}$$

The variables to the left of $\to$ are regarded as input variables; the output variable is on the right.

The symbolic representation of the iteration (5) allows considerable flexibility in determining the one-variable operator equation. For example, consider g_1 as in (4a). If

the entire vector $\mathbf{z}_k$ is "selected" by $\rightarrow$ the linear operator equation is

$$-\Delta u_{k+1} + e^{u_k - v_k} - e^{w_k - u_k} - k_1 = 0 \tag{6}$$

while the selection of only v_k and w_k explicitly leads to the nonlinear equation

$$-\Delta u_{k+1} + e^{u_{k+1} - v_k} - e^{w_k - u_{k+1}} - k_1 = 0. \tag{7}$$

Some suggestions on variants of the iteration (5) as well as methods to solve the individual nonlinear and/or linear discretized equations are given in [12], [5], [6]. For the types of problems we have encountered, the G-S/J iteration works well when the applied voltages insure that the solution currents are relatively small and the recombination-generation term k_2 can be neglected ($k_2 \equiv 0$). In this setting (4b) and (4c) can be regarded as small perturbations of the electrostatic potential equation (4a), and we suggest solving (7), rather than (6), fairly accurately. Equations (5b) and (5c) can be taken as linear equations by selecting v_k and w_k from z_k or by introducing the changes of variables $\nu = e^{-v}$, $\omega = e^{w}$ in (4b), (4c), respectively; however, in some computing environments such changes of variables may lead to undesirable numerical difficulties (underflow, overflow, loss of precision).

The major objection to G-S/J iteration is slow convergence for high currents and complicated functions $k_2 \not\equiv 0$. Although the multilevel adaptive refinement approach discussed in Subsection III-E attempts to circumvent this objection, an approach which couples the equations more tightly is desirable. This leads to the second basic approach which attempts to apply some form of Newton-like iteration.

In most of the literature dealing with the system (1) or (4), the (perhaps approximate) Jacobian needed for a Newton-like iteration is calculated *after* the discretization procedure. That is, (1) is discretized, perhaps by finite differences, yielding a system of nonlinear equations like (1) where each g_i is now a vector valued function of three vectors u, v, w. Each "block" of the Jacobian is a matrix; for example, the diagonal blocks are square matrices $\partial g_1/\partial u$, $\partial g_2/\partial n$ and $\partial g_3/\partial p$, respectively.

We have found an alternative view more convenient. Consider the formal Jacobian of the differential system (1) or (4). This is a 3×3 matrix of linear operators all in differential form. It is computed by repeatedly applying the identity

$$\{\partial/\partial u[d/dx f(x,u)]\} * = d/dx[\partial f/\partial u *]. \tag{8}$$

Here d/dx is a derivative, usually a gradient or a divergence operator, and $\partial f/\partial u$ is a linear operator acting on the "placeholder" function $*$. For example, (considering (1))

$$\partial g_1/\partial u = -\Delta *, \tag{9a}$$

$$\partial g_2/\partial n = -\nabla\cdot(\partial j_n/\partial n *) - \partial k_2/\partial n *, \tag{9b}$$

$$\partial g_3/\partial p = -\nabla\cdot(\partial j_p/\partial p *) - \partial k_2/\partial p *, \tag{9c}$$

are the diagonal entries ($*$ has been suppressed on the left-hand side). Similarly, using (2a) we see that

$$\partial j_n/\partial n * = [-\mu_n \nabla u + D_n \nabla] *. \tag{10}$$

We consider now the operator Jacobian of the system (4) in greater detail, assuming momentarily that the functions μ_n and μ_p are only spatially dependent. Using (8) and (4) we obtain the 3×3 operator Jacobian (letting $k_2 \equiv 0$ for convenience)

(11)

$$g' = \begin{bmatrix} -\Delta^* + (e^{u-v} + e^{w-u})* & -e^{u-v}* & -e^{w-u}* \\ \nabla(\mu_n * e^{u-v}\nabla v) & -\nabla(\mu_n e^{u}\nabla(-e^{-v}*)) & 0 \\ \nabla(\mu_p * e^{w-u}\nabla w) & 0 & -\nabla(\mu_p e^{-u}\nabla(e^{w}*)) \end{bmatrix}.$$

In a typical application of an approximate Newton method such as suggested in [6], the nonlinear iteration proceeds as

(12a) $$g_k' x = -g_k,$$

(12b) $$z_{k+1} = z_k + t_k x$$

for g_k' as in (11), $g_k = (g_1, g_2, g_3)^T$ as in (4), and $x \equiv (\delta u, \delta v, \delta w)^T$ corresponding to $z = (u, v, w)^T$. In the matrix vector product of (12a), the $*$ in each matrix entry indicates where to put the appropriate component of x. The parameters t_k are chosen to make the iteration converge in an orderly fashion; see [6].

There are several advantages in viewing the solution process abstractly as indicated by (11) and (12). First we see clearly the differential form of the linear operator g' we must ultimately discretize. Note that each matrix entry is a linear operator in *divergence form*. This is ideal for discretizations based on the box method or the finite-element method as discussed in Section III. Note also that the diagonal linear operators are in self-adjoint form if we view them as operating on δu, $\delta \nu = -e^{-v}\delta v$, and $\delta\omega = e^{w}\delta w$, respectively. $\delta\nu$ and $\delta\omega$ correspond to the variables $\nu = e^{-v}$ and $\omega = e^{w}$, although this change of variables is not applied explicitly to the entire system (4).

Another advantage of the operator view is that it immediately suggests solving (12a) by the *iterative* Gauss–Seidel splitting (or more general SOR splitting)

(13) $$\mathbf{L}_k(x_n - x_{n-1}) = -(g_k' x_n + g_k)$$

where $\mathbf{L}_k$ is the lower triangular part of g_k'. Note that each iteration (13) requires, in addition to the assembly of the system, the solution of three self-adjoint PDE's. In an abstract (conceptual) mathematical software environment, solving (13) might involve calls to a precomputed (divergence form) discretization procedure as part of the necessary calls to appropriate elliptic solvers. In practice, however, the distinction between discretization and solution of (12a) or (13) often becomes blurred as the details of implementation become important to the overall efficiency of the computation. It then becomes useful to view (13) as a block iterative method for the discretized block system arising from (12a). We have found such iterative methods for solving systems (12a) (after discretization) far more effective than solving the discrete analogue of (12a) by a sparse direct method. We comment in greater detail in Subsection III-F.

Computing the operator Jacobian has the additional advantage of providing motivation for the choice of an approximation to the exact operator Jacobian g' of the operator system $g(z) = 0$. Recall that (11) was derived under the tacit assumption that μ_n and μ_p were only spatially dependent. In most models, this function depends upon the dependent variable u in fact, usually on the field ∇u. This implies that (again take $k_2 \equiv 0$)

(14a) $$\partial g_2 / \partial u = -\nabla\cdot(\partial / \partial u\, j_n *),$$

(14b) $$j_n = -\mu_n e^{u-v}\nabla v.$$

Expanding, we see (suppressing $*$)

$$\partial j_n/\partial u = -(\partial \mu_n/\partial u\, e^{u-v} + \mu_n e^{u-v})\nabla v \tag{15}$$
$$= [1-(\partial \mu_n/\partial u)/\mu_n]\, j_n = [1-\partial/\partial u \ln \mu_n]\, j_n.$$

A similar expression holds for $\partial j_p/\partial u$. Hence (11) can be viewed as an approximation to the exact Jacobian where the factor $\partial/\partial u \ln \mu_n$ is neglected in the computation of $\partial g_2/\partial u$ (similarly for $\partial g_3/\partial u$). In many cases such approximations cause little deterioriation in the observed rate of convergence for the iteration (12), although, theoretically, (see [6]) the convergence rate cannot be superlinear in general. Of course, if $\partial/\partial u \ln \mu_n$ can be computed analytically, there is little reason not to include it. However, μ_n is often obtained experimentally, and in such cases either neglecting it as in (11) or approximating $\partial/\partial u \ln \mu_n$ by differences is indicated. In any event, we suggest approximating the individual terms in the operator Jacobian when necessary, rather than using some finite-difference approximation to the discretized system.

C. Choice of variables and scaling. As we have indicated in Subsection II-A, the assumption of the Einstein relation allows the electron and hole densities n and p, respectively, to be written in terms of the potentials u, v, and w as in (3). The functions v and w are "smoother" than n and p, although the variation of

$$j_n = -\mu_n(n\nabla u - \nabla n) = -\mu_n e^{u-v}\nabla v \tag{16a}$$

and

$$j_p = -\mu_p(p\nabla u + \nabla p) = -\mu_p e^{w-u}\nabla w \tag{16b}$$

due to the dependent variables u, v, and w is significant. Hence we may anticipate the linear operators $\partial j_n/\partial v$ and $\partial j_p/\partial w$ which appear on the diagonal of (11) to be somewhat ill-conditioned. Indeed, the coefficient functions e^u and e^{-u} in the expressions $-\nabla(\mu_n e^u \nabla(-e^{-v}*))$ and $-\nabla(\mu_p e^{-u}\nabla(e^w *))$ indicate that these operators can be poorly scaled due to the variation in $u(x, y)$; in some computing environments even the evaluation of the corresponding discrete operator may be difficult.

Part of the cause of the misscaling is the desire to force the diagonal operators to be self-adjoint and positive definite. Hence in (11) we implicitly consider the variables

$$\nu = e^{-v} \tag{17a}$$

and

$$\omega = e^w \tag{17b}$$

in the sense that we "solve" for $\delta\nu$ and $\delta\omega$ and then obtain δv and δw. We suggest that such a procedure is preferable, in a coupled approximate Newton strategy, to using the variables u, ν, and ω throughout the calculation since the range of values for ν and ω is much greater than for v and w. On the other hand, when the Poisson equation dominates and a G-S/J iteration strategy is used, the variables u, ν, and ω can often be used successfully. For $k_2 \equiv 0$, note that the linear PDE's $g_2(u, \nu)$ and $g_3(u, \omega)$ of (4) have the form

$$g_2(u, \nu) = -\nabla \mu_n e^u \nabla \nu = 0 \tag{18a}$$

and

$$g_3(u, \omega) = -\nabla \mu_p e^{-u}\nabla \omega = 0. \tag{18b}$$

We suggest that the scaling problem for the linear operators

(19a) $$\mathbf{L}_1\eta = -\nabla\cdot(\mu_n e^u \nabla\eta)$$

and

(19b) $$\mathbf{L}_2\eta = -\nabla\cdot(\mu_p e^{-u} \nabla\eta)$$

which appear on the diagonal of (11) and in (18) be addressed as follows. Consider (18a). Discretization of $\mathbf{L}_1\eta = 0$ as we shall see in Sections III and IV, leads to a matrix vector product

(20) $$A(\mu_n e^u)\eta = \mathbf{k}$$

where $A(\mu_n e^u)$ is an $m \times m$ matrix and η and k are now m vectors. For the discretizations we consider, $A(\mu_n e^u)$ will be symmetric positive definite; however, the entries a_{ij} will vary as e^u varies on the discretization mesh.

We have tried several scalings of (20). A symmetric scaling

(21a) $$A_1\eta_1 = D^{-1/2}AD^{-1/2}(D^{1/2}\eta) = D^{-1/2}k = k_1,$$

where

(21b) $$D = \operatorname{diag}[e^u]$$

is a diagonal matrix with entries e^u evaluated at meshpoints, preserves the symmetry and positive definiteness in the matrix $A_1 = D^{-1/2}AD^{-1/2}$. It is easy to compute the scaled matrix A_1 during assembly. Alternatively, we have considered the scaled linear system

(22) $$AD^{-1}(D\eta) = k,$$

D as in (21b). The linear system

(23) $$A_2\eta_2 = k$$

is no longer symmetric. However, since A_2 is a diagonal scaling of A, most iterative methods (SOR, SSOR, and SSOR-CG with slight modification), in exact arithmetic, have the same convergence rate when applied to (20), (21a), (23), and (24) following.

If an approximation to the quasi-Fermi variable v associated with (3a) is known, which would be the case in any setting discussed in Subsection II-B, the scaling (22) with

(24) $$D = \operatorname{diag}[e^v]$$

is natural. This leads to the linear system

(25) $$A_3\eta_3 = k$$

which we may further scale as

(26) $$D_1A_3\eta_3 = A_4\eta_3 = k_2 = D_1k, \qquad D_1 = \operatorname{diag}[e^{v-u}].$$

Note that A_3 and A_4 are not symmetric, although our comment regarding solution by iterative methods applies.

Some insight can be gained into the choice of scaling by examining the variables corresponding to the η_i given earlier. Suppose (20) is a discretization of (18a); then η represents the variable $\nu = e^{-v}$ in (17a). η_1 represents the variable $\theta = e^{u/2-v}$, a variable seemingly less volatile than ν but more so than $n = e^{u-v}$ represented by η_2 of (23). In the context of (25) and (26), η will usually represent $-e^{-v}*$ as in (11). Then $\eta_3 = -*$ and the matrix entries in A_3 are like the variable n while those in A_4 will contain exponentials of differences in v.

D. Grid continuation: two approaches. Our discussion of grid continuation is motivated by several concerns. First, as can be seen in the examples described in [13], the solutions of (1) or (4) exhibit large variations in the regions near the junctions but are generally quite smooth over most of the domain. In this situation, the straightforward use of uniform discretization meshes seems inappropriate, since a mesh sufficiently fine to achieve the desired accuracy in small regions where the solution is rough introduces many unnecessary unknowns in regions where the solution is smooth, greatly inflating the storage and computational time requirements.

Second, assuming that one has an appropriate mesh for a given problem, geometric complexity of the device and accuracy requirements can still lead to discrete systems of nonlinear equations whose dimension is quite large. Irrespective of the choice of nonlinear iterative method, it is clear one should seek to minimize the work associated with solving these nonlinear equations. One strategy for achieving this goal is to start from a good initial guess, so that only a few nonlinear iterations, perhaps only one or two, will be required to achieve the desired accuracy.

By *grid continuation* we mean the use of some strategy involving several grids—coarse grids with relatively few unknowns, inexpensive computational costs and low accuracy, and (locally) finer grids, on which accurate solutions may be computed. Coarse grids are used to obtain good starting iterates for the finer meshes, and can also be used in the creation of the finer meshes by indicating regions where refinement is necessary.

We have considered two different grid continuation approaches which we call *local mesh refinement* and *domain partitioning.*

The local mesh refinement scheme consists of generating a sequence of nested meshes on the entire domain. The meshes generally become increasingly nonuniform since most of the refinement is confined to small regions near the contacts. The decision of where to refine can be done manually (i.e., using human intervention) or automatically through the use of local *aposteriori* error estimates as in [1] and [9]. The problems are solved sequentially starting at the coarsest grid and using the jth computed solution as the initial iterate for the $(j+1)st$ problem. Since all meshes cover the entire domain, it is easy to make use of multilevel iterative methods in solving the systems of equations arising from any particular nonlinear iterative scheme. A mathematical analysis of such a scheme is given in [7]. Local mesh refinement seems most appropriate for finite-element discretization, since it can more easily cope with the host of technical difficulties which arise.

The domain partitioning idea makes use of (possibly nonuniform) tensor product grids. The coarsest grid(s) cover the entire domain. The problem is solved to (say) second-order accuracy, and then the accuracy is improved using Richardson extrapolation, deferred correction or some similar technique (see for example [4]). At this point we assume that the solution is sufficiently accurate in large portions of the domain where the solution is smooth. We take the improved solution and fit it with a tensor product (say, cubic) spline interpolant, which we take as the "exact" solution on the smooth region. We may then consider one or more small subregions of the original domain where the solution is still deemed inaccurate. On these subregions, we solve reduced problems, using the spline interpolants to set both the boundary conditions and provide the initial iterate. This approach seems well suited to finite-difference calculations, which are relatively easy to implement on tensor product grids. The tensor product structure also allows for easy vectorization of many of the inner loop

computations, and thus leads to extremely efficient implementations on vector computers.

III. Discrete equation formulation. As we suggested in Subsection II-B, the nonlinear iteration schemes we have discussed there implicitly assume that the operator systems (1), (2), or (4) will be solved approximately in the sense that they will be solved in discrete form. In this section we present some of the details of our discretization schemes. Both the finite-difference and finite-element methods discussed are suitable for dealing with systems such as (1), (2), or (4) which are in *divergence form*. Furthermore, linearization of the system which gives rise to the Jacobian operator g' of (11) or linearization of any component operator g_i again yield operators in divergence form. We will limit our detailed discussion to discretization of the component operators g_i of (1) and (4). Discretization of any of the operators in (11) not included in this discussion is similar and straightforward.

Our finite-difference discretization is based on the box method as discussed, for example, in [19, sec. 6.3]. Subsection III-A reviews this method and Subsections III-B and III-C apply the method to derive our discretizations. Subsection III-D discusses the relation of our discretizations to the Scharfetter–Gummel [16] approach often used in semiconductor simulation. In Subsection III-E we present our finite-element approach which is an adaptation of the piecewise-linear-triangular element approach of Bank *et al.* [2], [3], [7], [9]. Interestingly we show that, with the right choice of quadrature rule, this finite-element discretization is a box-method finite-difference discretization where now the boxes are general polygons rather than rectangles. Finally in Subsection III-F we discuss briefly some of the linear equations schemes for solving the linearized discrete equations.

A. Finite differences: box method. Often the domain of our PDE system is essentially a rectangle (or unions of rectangles). For such regions a rectangular-grid finite-difference approach is convenient and easy to implement allowing us, in solving the equations, to take some advantage of fast vector processing capabilities like those available on a CRAY-1. In both the silicon and oxide rectangles we use a tensor product mesh

$$\mathbf{M} = \{(x_i, y_j)\}, \qquad 0 \leqslant i \leqslant n+1, \quad 0 \leqslant j \leqslant m+1. \tag{27}$$

We use the same mesh for each of the three equations; although some authors (see [17], for example) favor using different meshes, our choice saves memory and implementation detail. The box method is well suited to equations in *divergence form*, i.e., in the form

$$\nabla \cdot \mathbf{F} = s \tag{28}$$

where $\nabla\cdot$ is the divergence operator. We briefly review this method in two dimensions, and then apply the method to derive various discretizations. The 3-D case is completely analogous; we restrict our attention in this section to 2-D.

Let the vector function $\mathbf{F}$ be written $\mathbf{F} = (f_1, f_2)$. Formally applying the divergence theorem, we recall

$$\iint_R \nabla \cdot \mathbf{F}\, dx\, dy = \int_C f_1\, dy - f_2\, dx \tag{29}$$

where R is a region of the x, y plane and C is its boundary curve. The right-hand side

of (29) is a line integral with counter-clockwise orientation. Let $h_i = x_i - x_{i-1}$ and $k_j = y_j - y_{j-1}$. For interior mesh points, i.e., points (x_i, y_i) with $i \neq 0$ or $n+1$ and $j \neq 0$, $m+1$, the region R is the box as indicated in Fig. 1. Notice that the edges of the box intersect the discretization mesh lines at midpoints; call the midpoint between (x_i, y_j) and (x_{i+1}, y_j) $(x_{i'}, y_j)$ or (i', j), etc.

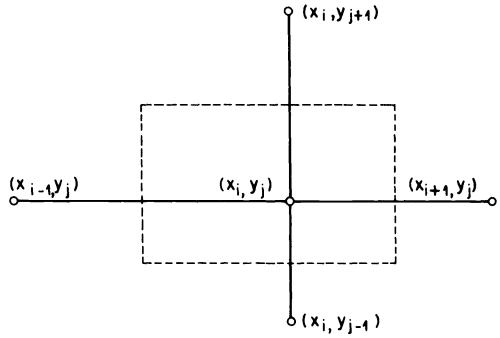


FIG. 1. *Finite-difference "box."*

Combining (28) and (29) in the box of Fig. 1 we obtain

$$I_{i-1',j} + I_{i,j-1'} + I_{i',j} + I_{i,j'} = \int_{\text{Box}} \int s\,dx\,dy \tag{30}$$

where

$$I_{i',j} = \int_{y_j - k_j/2}^{y_j + k_{j+1}/2} f_1(x_{i'}, y)\,dy, \tag{31a}$$

$$I_{i,j-1'} = \int_{x_i - k_i/2}^{x_i + k_{i+1}/2} - f_2(x, y_{j-1'})\,dx \tag{31b}$$

and similarly for $I_{i-1',j}$ and $I_{i,j'}$. Note that (30) is exact, and we now approximate both sides of this equation. The right-hand side of (30) is approximated by

$$\text{rhs}(i, j) = (s_{ij})\text{area(Box)} = s_{ij}\left[(k_j + k_{j+1})(h_i + h_{i+1})/4\right]. \tag{32}$$

Here $s_{ij} \equiv s(x_i, y_i)$. Similarly, the integrals in (31) can be approximated by

$$\hat{I}_{i',j} = \left[(k_j + k_{j+1})/2\right] f_1(x_{i'}, y_j), \tag{33a}$$

$$\hat{I}_{i,j-1'} = -\left[(h_i + h_{i+1})/2\right] f_2(x_i, y_{j-1'}). \tag{33b}$$

As we shall see below, f_1 and f_2 usually involve derivatives and other functions known (or to be determined) at the meshpoints **M**. Hence f_1 and f_2 are further approximated by functions, say, $\tilde{f}_1$ and $\tilde{f}_2$, usually using centered difference quotients and interpolated values of mesh functions. This leads to an approximate left-hand side of (30), say

$$\text{lhs}(i, j) = \tilde{I}_{i-1',j} + \tilde{I}_{i,j-1'} + \tilde{I}_{i',j} + \tilde{I}_{i,j'}. \tag{34}$$

The finite-difference equations are then

(35) $$\text{rhs}(i, j) = \text{lhs}(i, j)$$

for each (i, j) meshpoint where a box was defined; these will be all meshpoints except those on boundaries where Dirichlet boundary conditions are applied.

For meshpoints (i, j) where a homogeneous Neumann boundary condition is applied, one edge of the box is the mesh line segment which includes the (i, j) point (see Fig. 2). In this case $f_1 = 0$ or $f_2 = 0$ on the mesh line segment and hence an expression in (31), (33), and (34) will vanish. Also in (32) the area is determined by the smaller box of Fig. 2.

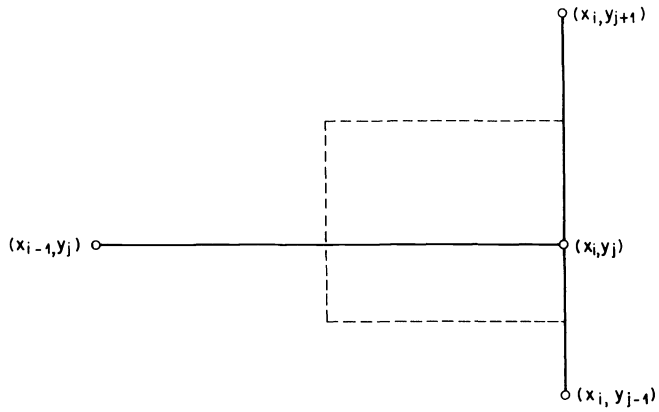


FIG. 2. *Finite-difference "box" at boundary.*

For meshpoints on the silicon/oxide interface of the Poisson equation, we will require f_2 to be continuous along the interface. If we then apply (29) separately to each of the two smaller rectangles in Fig. 1 above and below the interface line, the two contributions along the interface line will cancel. The contributions $I_{i, j-1'}$ and $I_{i, j'}$ in (30) will remain the same; however, $I_{i', j}$ and $I_{i-1', j}$ will consist of two parts as in

(36) $$I_{i', j} = \int_{y_j - k_j/2}^{y_j} f_1(x_{i'}, y)\, dy + \int_{y_j}^{y_j + k_{j+1}/2} f_1(x_{i'}, y)\, dy,.$$

allowing for piecewise continuous functions f_1. The corresponding modification to (33a) is

(37) $$\hat{I}_{i', j} = (k_j/2)\big[f_1(x_{i'}, y_j)\big]_{\text{silicon}} + (k_{j+1}/2)\big[f_1(x_{i'}, y_j)\big]_{\text{oxide}}.$$

Similarly (32) is replaced by

(38) $$\text{rhs}(i, j) = \big(k_j(h_i + h_{i+1})/4\big)[s_{ij}]_{\text{silicon}} + \big(k_{j+1}(h_i + h_{i+1})/4\big)[s_{ij}]_{\text{oxide}}.$$

B. The Poisson equation. We have from [13, eq. (21)] the equation

(39a) $$-\varepsilon_1 \Delta u = s$$

in the semiconductor and

(39b) $$-\varepsilon_2 \Delta u = 0$$

in the insulator regions. Along the interface we impose the continuity relation

(39c) $$[\varepsilon_1 u_y]_{\text{silicon}} = [\varepsilon_2 u_y]_{\text{oxide}}$$

as suggested earlier; hence f_2 is continuous there. Application of (32)–(35) is straightforward. Consider a point (i, j) as in Fig. 1 in the silicon region. Then, with $f_1 = -\varepsilon_1 u_x$ and $f_2 = -\varepsilon_1 u_y$, we will approximate $f_1(x_{i'}, y_j)$ and $-f_2(x_i, y_{j-1'})$ by

(40a) $$\tilde{f}_1(x_{i'}, y_j) = (\varepsilon_1/h_{i+1})(u_{i,j} - u_{i+1,j})$$

and

(40b) $$\tilde{f}_2(x_i, y_{j-1'}) = (\varepsilon_1/k_j)(u_{ij} - u_{i,j-1})$$

respectively, giving

(41a) $$\tilde{I}_{i',j} = \left[\varepsilon_1(k_j + k_{j+1})/2h_{i+1}\right](u_{ij} - u_{i+1,j})$$

and

(41b) $$\tilde{I}_{i,j-1'} = \left[\varepsilon_1(h_i + h_{i+1})/2k_j\right](u_{ij} - u_{i,j-1}).$$

Similar expressions complete the sum in (34), and (32) with (35) complete the discretization at this point.

If (i, j) is a meshpoint on the oxide/silicon interface we proceed as in (36)–(38), noting that (39c) insures that f_2 is continuous along the interface. Approximating $f_1 = -\varepsilon u_x$ for $\varepsilon = \varepsilon_1$ or $\varepsilon = \varepsilon_2$ where appropriate in (37), we obtain

(42) $$\tilde{I}_{i',j} = \left((\varepsilon_1 k_{j+1} + \varepsilon_2 k_j)/2h_{j+1}\right)(u_{ij} - u_{i+1,j}).$$

Since $[s_{ij}]_{\text{oxide}} = 0$ at this point (38) becomes

$$\text{rhs}(i, j) = \left(k_j(h_i + h_{i+1})/4\right)[s_{ij}]_{\text{silicon}}.$$

Note that in both these cases

(43) $$[s_{ij}]_{\text{silicon}} = -n(x_i, y_i) + p(x_i, y_j) + k_1(x_i, y_j)$$

for n, p, and k_1 as in (1). When n and p are as specified as in (3), the equations are nonlinear in the u_{ij}.

C. Continuity equations. We consider first the continuity equations of (1) written as

(44a) $$-\nabla \cdot j_n = s$$

and

(44b) $$\nabla \cdot j_p = -s$$

with the current densities in terms of quasi-Fermi levels

(45a) $$j_n = -\mu_n e^{u-v} \nabla v$$

and

(45b) $$j_p = -\mu_p e^{w-u} \nabla w.$$

We also consider simultaneously the variables $v = e^{-v}$, and $\omega = e^w$ which allows (45) to be rewritten as

(46a) $$j_n = \mu_n e^u \nabla v$$

and

(46b) $$j_p = -\mu_p e^{-u} \nabla \omega.$$

To minimize notation we first discretize the equation

(47a) $$-\nabla \cdot j = s,$$

(47b) $$j = \mu e^{u} \nabla \nu$$

the other carrier continuity equation in (44) being similar. We apply the box method as in (30)–(35) for the two cases of meshpoints (i, j) as in Figs. 1 and 2, respectively. Consider first the case of Fig. 1.

We approximate $\nabla \nu$ by centered differences exactly as in (40). Thus

(48a) $$\tilde{f}_1(x_{i'}, y_j) = -(\mu e^{u})_{i'j}(\nu_{i,j} - \nu_{i+1,j})/h_{i+1}$$

and

(48b) $$\tilde{I}_{i',j} = -\big((k_j + k_{j+1})/2h_{i+1}\big)(\mu e^{u})_{i'j}(\nu_{ij} - \nu_{i+1,j}).$$

In (48) we have not specified the interpolated value of $(\mu e^{u})_{i'j}$. Recall that μ may be field dependent, i.e., $\mu = \mu(\nabla u)$. While we are using the approximation $(u_{i+1,j} - u_{ij})/h_{j+1}$ for u_x at (i', j), we do not have readily available such an approximation for u_y. Hence we do not attempt to compute $u_{i'j}$ as $\mu((\nabla u)_{i'j})$, approximating $(\nabla u)_{i'j}$. Rather, we approximate $(\nabla u)_{i,j}$ at meshpoints **M**, using standard differences, and then evaluate μ with this approximation. Call these values μ_{ij}. We evaluate $(\mu e^{u})_{i'j}$ as the product $\mu_{i'j}(e^{u})_{i'j}$ using

(49) $$\mu_{i'j} = (\mu_{ij} + \mu_{i+1,j})/2.$$

The evaluation of $(e^{u})_{i'j}$ is more interesting and has been the subject of considerable discussion in the semiconductor simulation literature. The analogue of (49), namely

(50) $$E_1 = (e^{u_{ij}} + e^{u_{i+1,j}})/2,$$

is regarded to be a poor candidate even though it leads to fewer evaluations of e^{u}. If u varies significantly between meshpoints and if this variation is at least linear, then (50) will be relatively inaccurate. If u were indeed linear between meshpoints $(e^{u})_{i'j}$ would be

(51) $$E_2 = e^{(u_{ij} + u_{i+1,j})/2},$$

a tempting approximation for $(e^{u})_{i'j}$ in general. The use of E_2 is controversial; Mock [15], for example, claims that (51) will lead to inaccuracies.

An alternative expression for $(e^{u})_{i'j}$ can be motivated by considering the equation

(52) $$\frac{d}{dx} e^{-u} = -e^{-u} \frac{du}{dx}$$

and then integrating

(53) $$\int_i^{i+1} e^{u} \frac{d}{dx} e^{-u}\, dx = -\int_i^{i+1} \frac{du}{dx}\, dx = u_{ij} - u_{i+1,j}.$$

The mean value theorem says that the integral on the left-hand side is

(54) $$e^{u(x', y_1)} \int_i^{i+1} \frac{d}{dx} e^{-u}\, dx = e^{u'}\big(e^{-u_{i+1,j}} - e^{-u_{ij}}\big).$$

If we identify $x' \in [x_{ij}, x_{i+1,j}]$ with the midpoint, we obtain for $(e^{u})_{i'j}$

$$E_3 = \frac{(u_{ij} - u_{i+1,j})}{(e^{-u_{i+1,j}} - e^{-u_{ij}})} = B(\Delta u)e^{u_{ij}} = B(-\Delta u)e^{u_{i+1,j}} \tag{55}$$

where $\Delta u = u_{ij} - u_{i+1,j}$ and $B(x)$ is the Bernoulli function

$$B(x) = x/(e^x - 1).$$

Note that

$$E_3/E_2 = S(\Delta u/2), \tag{56a}$$

$$S(x) = x/\sinh x \tag{56b}$$

and that for Δu large E_3 can be considerably smaller than E_1, e.g., $S(10/2 = 5) = 0.0674$. Whether the use of (55) as opposed to (51) is more accurate *a priori* or rather that its use leads to a more stable computation with respect to roundoff is not well understood. The expression (55) will arise again in our discussion of the Scharfetter–Gummel discretization.

If (i, j) is a meshpoint as in Fig. 2, there is no term of the form $\tilde{I}_{ij'}$ in (34), and coefficients of $-f_2$ in $\hat{I}_{i,j-1'}$ and $\hat{I}_{i,j'}$ in the modified (33) will be $h_i/2$. For example, we obtain

$$\tilde{I}_{i,j-1'} = -(h_i/2k_j)(\mu e^u)_{i,j-1'}(\nu_{ij} - \nu_{i,j-1}).$$

Finally, in this case, we have

$$\mathrm{rhs}(i, j) = s_{ij}\left[(k_j + k_{j+1})h_i/4\right] \tag{57}$$

replacing (32).

The derivation of the discretization for the current densities as given by (45a) is completely analogous to (47)–(56). We obtain, in place of (48b), the expression

$$\tilde{I}_{i'j} = \left((k_j + k_{j+1})/2h_{i+1}\right)(\mu e^{u-v})_{i'j}(v_{i+1,j} - v_{ij}). \tag{58}$$

Here $(\mu e^{u-v})_{i'j}$ can be calculated as in our discussion of (49), (51), and (55). The analogue of (55) for $(e^{u-v})_{i'j}$ could be

$$B(\Delta(u - v))e^{u_{ij} - v_{ij}} \tag{59a}$$

or

$$(B(\Delta u)e^{u_{ij}})(B(-\Delta v)e^{-v_{ij}}, \tag{59b}$$

(59b) being derived from (55) and taking $(e^{u-v})_{i'j} = (e^u)_{i'j}(e^{-v})_{i'j}$. As an alternative to (58) we frequently take

$$\tilde{I}_{i'j} = -\left((k_j + k_{j+1})/2h_{i+1}\right)(\mu e^u)_{i'j}(e^{-v_{ij}} - e^{-v_{i+1j}}) \tag{60}$$

obtained by substituting $\nu_{ij} = e^{-v_{ij}}$ into the expression (48b). Interestingly, (58) and (60) are identical if $(e^u)_{i'j}$ is evaluated as in (55) and $(e^{u-v})_{i'j}$ is written as $(e^u)_{i'j}/(e^v)_{i'j}$ with $(e^u)_{i'j}$ and $(e^v)_{i'j}$ again evaluated as in (55).

The main difference in the discretization using ν and ω or v and w is in the form of the function $\mathrm{lhs}(i, j)$, $(i, j) \in \mathbf{M}$ and reflects our comments in Section II. It is easy to see that, given the u_{ij}, the function $\mathrm{lhs}(i, j)$ of (34) with the $\tilde{I}$ as in (48b) is a *linear function* of the ν_{ij}. However, this function with the $\tilde{I}$ as in (58) or (60) produces a *nonlinear function* of the v_{ij}. On the other hand, we anticipate that the v and w variables will be smoother (less range in value) than the ν and ω variables.

D. Physically motivated discretizations and comparative remarks. Consider the general equation (28) which led us (via Fig. 1) to the approximations (33)–(35). The essential difference in the approaches we wish to compare is in approximating the quantities f_1 and f_2 at midpoints of the mesh **M**.

Let **F** of (28) be $j = (j_1, j_2)$ of (47a) with j specified as in (2a), i.e., for j_1 we obtain

$$j_1 = -\mu(du/dx)n + D(dn/dx). \tag{61}$$

Assume that the current j_1, the field component $E = -du/dx$, and μ and D are constant from meshpoints (i, j) to $(i+1, j)$. Then (61) specifies a constant coefficient *ODE* as

$$dn/dx + \tilde{E}n + \tilde{j}_1 \tag{62}$$

where

$$\tilde{E} = \mu E/D, \tilde{j}_1 = j_1/D.$$

Suppressing the index j, the solution to (62) for $x \in [x_i x_{i+1}]$ is

$$n(x) = \left(n(x_i)e^{-\tilde{E}(x-x_i)} + (\tilde{j}/\tilde{E})\right)\left(1 - e^{-\tilde{E}(x-x_i)}\right). \tag{63}$$

Since E was assumed constant, we have $E = -(u_{i+1} - u_i)/h_{i+1}$; substituting into (63) and rearranging gives

$$j_1 = (\mu/h_{i+1})(u_i - u_{i+1})\left(e^{\alpha(u_i+u_{i+1})} - 1\right)^{-1}\left(n_{i+1}e^{\alpha(u_i-u_{i+1})} - n_i\right) \tag{64}$$

with $\alpha = \mu/D$. The component j_2 is obtained similarly.

The SG discretization uses the value of j_1 in (64) for the value of $\tilde{f}_1(x_{i'j})$ approximating $f_1(x_{i'j})$ of (33a). $\tilde{I}_{i'j}$ is computed as before, contributing a multiplicative factor of $-(k_j + k_{j+1})/2$ (recalling the "$-$" in (44a)). When $\alpha = 1$ and $n = e^{u-v}$ we obtain

$$\begin{aligned} \tilde{I}_{ij} &= -\left((k_j + k_{j+1})/2h_{i+1}\right)\mu B(\Delta u)\left(e^{u_i - v_i} - e^{u_i - v_{i+1}}\right) \\ &= -\left((k_j + k_{j+1})/2h_{i+1}\right)\mu B(\Delta u)e^{u_i}\left(e^{-v_i} - e^{-v_{i+1}}\right). \end{aligned} \tag{65}$$

This expression is identical to (48a) and (60) when μ is taken as $\mu = \mu_{i'j}$ of (49) and (55) is used to evaluate $(e^u)_{i'j}$. Hence an alternative view of the discretization (65) is that presented in (47), (48), and (52)–(55).

Note that the derivation of j_1 in (61)–(64) appears to be more general than our discussion in Subsection III-C since (64) does not explicitly assume the Einstein relation (3). However, the assumption that μ and D are constant from meshpoints (i, j) to $(i+1, j)$ implies a *localized Einstein relation* which leads to interesting algorithmic possibilities.

Let

$$\alpha_{i'j} = \mu_{i'j}/D_{i'j} \tag{66}$$

be assumed constant between (i, j) and $(i+1, j)$. Then (61) can be rewritten as

$$j_1 = \mu_{i'j}e^{\alpha_{i'j}(u-v)}\frac{dv}{dx} \tag{67a}$$

with

$$n = e^{\alpha_{i'j}(u-v)} \tag{67b}$$

between (i, j) and $(i+1, j)$. Assuming further that j_1 is constant between these meshpoints, we obtain

$$j_1 = \frac{D_{i'j}\left(e^{-\alpha_{i'j}v_i} - e^{-\alpha_{i'j}v_{i+1}}\right)}{\alpha_{i'j}\int_{x_i}^{x_{i+1}} e^{-\alpha_{i'j}u}\,dx}. \tag{68}$$

If the expression, $\int_{x_i}^{x_{i+1}} e^{-\alpha_{i'j}u}\,dx$, is evaluated by assuming u is linear between (i, j) and $(i+1, j)$, we find

$$j_1 = -D_{i'j}e^{\alpha_{i'j}u_i}B\left(\alpha_{i'j}\Delta u\right)\left(e^{-\alpha_{i'j}v_i} - e^{-\alpha_{i'j}v_{i+1}}\right) \tag{69}$$

where $B(x)$ is again the Bernoulli function. Note that (69) is a direct generalization of the appropriate expressions in (65) and (55).

We see that the use of (67) in conjunction with the box method as presented in Subsection III-C not only generalizes the SG discretization as in (69), but generalizes the entire discussion there. Note that in the complete discretization involving all the mesh constants of the form $\alpha_{i'j}$ and $D_{i'j}$, the quantities v_{ij} (and w_{ij}) are well defined at meshpoints while the quantities n_{ij} (and p_{ij}) are ambiguous (ill-defined) at meshpoints. Nevertheless, these quantities can be defined, when necessary, in an "average" or integrated sense, for example, in a generalized calculation of rhs(i, j) of (32) for the Poisson equation. The calculation of the currents as in (67a) is well defined.

We are presently examining the application of discretizations like (69) in cases of physical interest where the Einstein relation is suspect.

E. The finite-element discretization. The finite-element method offers several potential advantages as a discretization procedure. Perhaps most important is its ability to handle unusual geometry and nonuniform meshes in a straightforward fashion. In our experience, its main disadvantage has been the relatively costly process of assembling the Jacobian (stiffness) matrices and nonlinear residuals (right-hand sides) required by approximate Newton methods.

The finite-element procedure is based on a weak formulation of the system of partial-differential equations. (See [18] for standard terminology.) As with the finite-difference discretization, we consider the variables (u, v, w) and (u, ν, ω). Let $\mathbf{H}^1(\Omega)$ be the usual Sobolev space equipped with the norm

$$\|u\|_1^2 = \int\int_\Omega |\nabla u|^2 + u^2\,dx\,dy$$

and denote by $\mathbf{H}_0^1(\Omega)$ the subspace of $\mathbf{H}^1(\Omega)$ whose elements satisfy homogeneous boundary conditions on the Dirichlet portion of the boundary. Let $\mathbf{H}_u^1(\Omega)$ be the affine space whose elements satisfy the Dirichlet boundary conditions satisfied by u (i.e., $p, q \in \mathbf{H}_u^1$ implies $p - q \in \mathbf{H}_0^1$). Similar spaces can be defined for v, w, ν, and ω.

Let $H = \mathbf{H}_u^1 \otimes \mathbf{H}_v^1 \otimes \mathbf{H}_w^1$, $\hat{H} = \mathbf{H}_u^1 \otimes \mathbf{H}_\nu^1 \otimes \mathbf{H}_\omega^1$, and $H_0 = (\mathbf{H}_0^1)^3$. Then a weak form of (4) is: find $(u, v, w) \in H$, such that, for all $(\phi, \Psi, \gamma) \in H_0$

$$\begin{aligned}
&\int_\Omega \nabla u \nabla \phi + \left(e^{u-v} - e^{w-u} - k_1\right)\phi\,dx = 0,\\
&\int_\Omega \mu_n e^{u-v}\nabla v \nabla \Psi - k_2 \Psi\,dx = 0,\\
&\int_\Omega \mu_p e^{w-u}\nabla \omega \nabla \gamma + k_2 \gamma\,dx = 0.
\end{aligned} \tag{70}$$

Using the variables u, ν, and ω a weak form is: find $(u, \nu, \omega) \in \hat{H}$, such that, for all $(\phi, \Psi, \gamma) \in H_0$

$$\int_\Omega \nabla u \nabla \phi + (ve^u - we^{-u} - k_1)\phi \, dx = 0,$$

$$\int_\Omega \mu_n e^u \nabla v \nabla \Psi + k_2 \Psi \, dx = 0, \tag{71}$$

$$\int_\Omega \mu_p e^{-u} \nabla \omega \nabla \gamma + k_2 \gamma \, dx = 0.$$

Let $\mathbf{T}$ be a shape regular but possibly nonuniform triangulation of the domain Ω. Let $\mathbf{S} \subseteq \mathbf{H}^1(\Omega)$ be the space of C^0 piecewise linear polynomials corresponding to the triangulation $\mathbf{T}$. Let $\mathbf{S}_0$ denote the subspace of $\mathbf{S}$ whose elements satisfy homogeneous boundary conditions on the Dirichlet portion of the boundary. Similarly we define the affine spaces $\mathbf{S}_u, \mathbf{S}_v, \mathbf{S}_w$, etc. (The Dirichlet boundary conditions typically imposed are constant on each segment of the Dirichlet boundary, and hence can be exactly satisfied by piecewise linear polynomials provided the triangulation $\mathbf{T}$ satisfies some modest requirements.) Let $S = \mathbf{S}_u \otimes \mathbf{S}_v \otimes \mathbf{S}_w$ and define $\hat{S}$ and S_0 in an analogous fashion.

The finite-element discretizations of (70), (71) come from replacing H, $\hat{H}$, and H_0 by S, $\hat{S}$, and S_0, respectively. If the dimension of $\mathbf{S}_0$ is N, (70), (71) effectively become systems of $3N$ nonlinear equations.

Our strategy for the finite-element discretization closely parallels that for the finite-difference discretization in terms of our choice of variables (u, v, w), and most of the remarks made in Section II remain valid here. The important new consideration is in the choice of numerical quadrature rules to be used in the assembly of matrices and right-hand sides. While we certainly need quadrature rules of sufficient accuracy, we also want the matrices corresponding to the diagonal blocks of the linearized equations (11) to be either symmetric and positive definite or such that they may be transformed to that state using diagonal matrices in a fashion analogous to the finite-difference case.

This second requirement is not satisfied by many of the usual quadrature rules associated with triangles. The rules which we propose are motivated by the extension of the box method to triangular meshes. Consider vertex i in the triangulation depicted in Fig. 3. The region shown is the support of the nodal basis function associated with node i. We construct a box around each node using the perpendicular bisectors of each edge in the mesh to define the boundaries of the box (Fig. 4). The box method can be applied in a fashion analogous to the finite-difference case to obtain a difference equation corresponding to node i.

We now consider the implications of this example in the elementwise assembly process usually employed in the finite-element method. Let $t \in T$ be an element in the mesh as depicted in Fig. 5. Let ϕ_i, $1 \leqslant i \leqslant 3$ denote the linear polynomials satisfying $\phi_i(p_j) = \delta_{ij}$. These polynomials may be associated with the three nonzero basis functions in element t. We define the dot products d_i, $1 \leqslant i \leqslant 3$ by

$$d_i = l_j l_k \cos\theta_i \tag{72}$$

where the triple (i, j, k) is any cyclic permutation of $(1,2,3)$. Note

$$l_i^2 = d_j + d_k. \tag{73}$$

The d_i are not computed using (72), but rather as dot products which arise naturally in the affine mapping of t to the "reference element" $\hat{t}$ with vertices $(0,0)$, $(1,0)$, and $(0,1)$ and the ϕ_i to the reference basis functions $\hat{x}$, $\hat{y}$, and $1 - \hat{x} - \hat{y}$, as is typically done in finite-element assembly procedures.

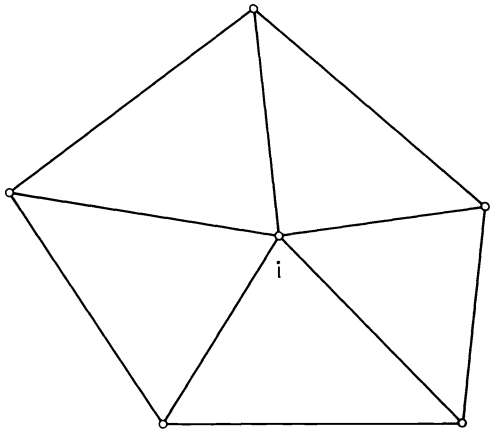


FIG. 3. *Vertex in triangulation.*

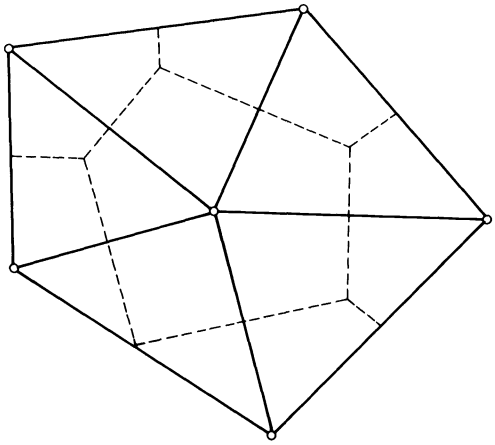


FIG. 4. *Box discretization.*

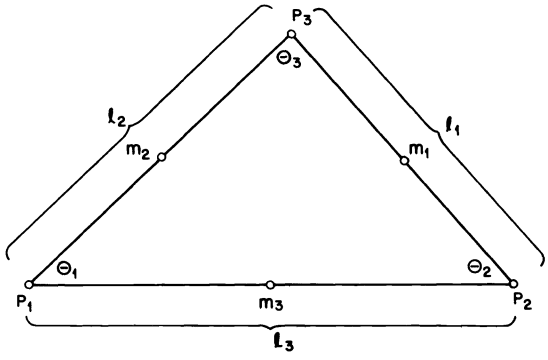


FIG. 5. *Definition of terms for triangular finite element.*

As an example, we consider in detail the approximation of

$$\int_t e^u \nabla\left(e^{-v}\phi_i\right)\nabla\phi_j\,dx\,dy, \quad 1 \leqslant i \leqslant 3,\ 1 \leqslant j \leqslant 3. \tag{74}$$

The 3×3 element stiffness matrix corresponding to one possible box-method quadrature rule is

(75)

$$\frac{1}{4|t|}\begin{bmatrix} d_3e^{u(m_3)-v(p_1)}+d_2e^{u(m_2)-v(p_1)} & -d_3e^{u(m_2)-v(p_2)} & -d_2e^{u(m_2)-v(p_3)} \\ -d_3e^{u(m_2)-v(p_1)} & d_3e^{u(m_2)-v(p_2)}+d_1e^{u(m_1)-v(p_2)} & -d_1e^{u(m_1)-v(p_3)} \\ -d_2e^{u(m_2)-v(p_1)} & -d_1e^{u(m_1)-v(p_2)} & d_2e^{u(m_2)-v(p_3)}+d_1e^{u(m_1)-v(p_3)} \end{bmatrix}$$

where $|t|$ is the area of t. The quadrature rule is exact for $u =$ constant, $v =$ constant; i.e., it is exact for the highest order term $\nabla\phi_i\nabla\phi_j$, which is the main requirement for any quadrature rule used in the finite-element method.

The global stiffness matrix can be symmetrized using the same diagonal scalings as in the finite-difference case. Indeed, one can compute the scaled element stiffness matrix directly in order to save storage, function evaluations, and arithmetic operations.

Quadrature rules for the remaining terms can be worked out using the same types of manipulations as in the previous example. When all interior angles of all the elements in $\mathbf{T}$ are bounded above by $\pi/2$, the d_i will be nonnegative (and the perpendicular bisectors of the three edges of an element t will meet at a point inside of t). In this case the resulting (scaled) Jacobians will be Stieltjes matrices (as in the finite-difference case). When angles are allowed to be larger than $\pi/2$ one d_i will be negative and the matrices will lose this property. Other box-method quadrature rules will still lead to positive definite (although not Stieltjes) matrices.

F. Solution of the discrete equations. After discretization the operator iterations of (5) and (11), (12) are viewed as finite dimensional systems of equations. In (5) each system g_i may be nonlinear or linear depending on the choice of variables and the particular choice of z_k as in (6) and (7). In the finite-difference package we solve the nonlinear equations by the approximate Newton schemes discussed in [5] and [6]. In our finite-element package we solve the nonlinear equations by the approximate Newton-multilevel iteration scheme discussed in [6], an extension of the linear multilevel iteration scheme [3] which we use for the linear equations. To date our finite-element package adapts the PLTMG software [2] for the nonlinear iteration (5) carrying any particular specification of (5) through a series of refined grids as discussed in Subsection III-E. There are several alternatives when considering a multilevel iterative method for solving systems of PDE either by (5) or (11), (12); see [8] for some discussion of continuing investigations.

Most of our simulations use the finite-difference package which has been tuned for efficient execution on a CRAY-1. Since both iterations (5) and (11), (12) use approximate Newton methods, sparse linear equations comprise an "inner loop" of the computation. We summarize our experience with various direct and iterative methods for solving the linear equations.

When solving the coupled system by a Newton-type iteration as indicated in (11), (12), we suggest that a block iterative method be used to solve (12a) as indicated in (13). We found an alternative which uses a sparse direct method to be less efficient (see [12]

for some comparisons). There are several reasons for this. First, even with a good ordering the arithmetic operation count is relatively high; since the equations are coupled, a 2D ordering analogous to nested dissection requires about 13 times the arithmetic operations required for a single PDE. (In 3D sparse direct solution methods can safely be labeled a disaster.) Second, sparse direct methods are relatively memory inefficient, and, again, this inefficiency increases with the number of coupled PDE's and certainly with the spatial dimension (2D to 3D). Finally, sparse direct methods do not vectorize well (with current compilers), whereas a block iterative approach allows the flexibility of solving some or all the diagonal block linear equations (which correspond to single PDE's) by highly specialized vectorized iterative methods. Note that the inefficiency of sparse direct methods in this context is not due to the sparse matrix software itself, but to the naive application of this software to a sparse linear system not well suited to that methodology.

We found the block iterative method to converge well for the problems we have encountered. Although a mathematical analysis of the convergence characteristics of block iteration of (12a) has proved illusive, it seems clear that the convergence rate does not depend upon the mesh parameter as the mesh is refined. There may conceivably be problems where block iteration converges so slowly (or even diverges) that the sparse direct solution (in 2D) becomes attractive. However, even in cases where we applied sparse direct methods to (12a), we used them most efficiently in conjunction with a Newton–Richardson iteration which only occasionally refactors the large linear systems; see [12] and [6]. Notice that such an approach still requires substantial memory.

If block iteration is used to solve the Newton equations as in (12a), the linear systems corresponding to the diagonal of (11) must be solved repeatedly in the inner loop of the overall computation. This is one reason for focusing attention in Sections II and III on obtaining self-adjoint elliptic operators on the diagonal of (11) which yield scaled symmetric positive definite systems of linear equations upon discretization. We solve these linear systems by vectorizable highly tuned iterative methods (such as SSOR with conjugate-gradient acceleration) [14] or by the Yale Sparse Matrix codes [10], [11]. In our CRAY-1 computing environment the iterative approach has proved to be somewhat more efficient, both in computing time and memory.

REFERENCES

[1] I. BABUSKA AND W. C. RHEINBOLDT, "Error estimates for adaptive finite element computations," *SIAM J. Numer. Anal.*, vol. 15, pp. 736–754, 1978.

[2] R. E. BANK, "PLTMG users' guide, June 1981 version," Dep. Math., Univ. of California, San Diego, Tech. Rep., Aug. 1982.

[3] R. E. BANK AND T. F. DUPONT, "An optimal order process for solving finite element equations," *Math. Comput.*, vol. 36, pp. 35–51, 1981.

[4] R. E. BANK AND D. J. ROSE, "Extrapolated fast direct algorithms for elliptic boundary value problems," in *New Directions and Recent Results in Algorithms and Complexity*, J. F. Traub, Ed. New York: Academic Press, 1976, pp. 201–247.

[5] _____, "Parameter selection for Newton-like methods applicable to nonlinear partial differential equations," *SIAM J. Numer. Anal.*, vol. 17, pp. 806–822, 1980.

[6] _____, "Global approximate Newton methods," *Numer. Math.*, vol. 37, pp. 279–295, 1981.

[7] _____, "Analysis of a multilevel iterative method for nonlinear finite element equations," *Math. Comput.*, vol. 39, pp. 453–465, 1982.

[8] _____, "Discretization and multilevel solution techniques for nonlinear elliptic systems," in *Elliptic Problem Solvers*, G. Birkhoff and A. Schoenstadt, Eds. New York: Academic Press, to be published.

[9] R. E. BANK AND A. WEISER, "Some *a posteriori* error estimates for elliptic partial differential equations," *Math. Comput.*, submitted for publication.

[10] S. EISENSTAT, M. GURSKY, M. SCHI AND A. SHERMAN, "Yale sparse matrix package I: The symmetric codes," *Int. J. Numer. Meth. Eng.*, vol. 18, pp. 1145–1151, 1982; also Dep. Comput. Sci., Yale Univ., New Haven, CT, Tech. Rep. 112, 1977.

[11] _____, "Yale sparse matrix package II: Nonsymmetric matrices," Dep. Comput. Sci., Yale Univ., New Haven, CT, Tech. Rep. 114, 1977.

[12] W. FICHTNER AND D. J. ROSE, "On the numerical solution of nonlinear elliptic PDEs arising from semiconductor device modeling," in *Elliptic Problem Solvers*, M. Schultz, Ed. New York: Academic Press, 1981, pp. 277–284.

[13] W. FICHTNER, D. J. ROSE, AND R. E. BANK, "Semiconductor device modeling," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1018–1030, 1983; this issue, pp. 391–415.

[14] R. G. GRIMES, D. R. YOUNG, AND D. M. YOUNG, "ITPACK 2A: A Fortran implementation of adaptive accelerated iterative methods for solving large sparse linear systems," Center for Numerical Analysis, Univ. of Texas, Austin, Tech. Rep. CNA-164, 1980.

[15] M. S. MOCK, "A two-dimensional mathematical model of the insulated-gate field-effect transistor," *Solid-State Electron.*, vol. 16, pp. 601–609, 1973.

[16] D. SCHARFETTER AND H. GUMMEL, "Large-signal analysis of a silicon Read diode oscillator," *IEEE Trans. Electron Devices*, vol. ED-16, pp. 64–77, 1969.

[17] J. W. SLOTBOOM, "Computer-aided two-dimensional analysis of bipolar transistors," *IEEE Trans. Electron Devices*, vol. ED-20, pp. 669–679, 1973.

[18] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method.* Englewood Cliffs, NJ: Prentice-Hall, 1973.

[19] R. S. VARGA, *Matrix Iterative Analysis.* Englewood Cliffs, NJ: Prentice-Hall, 1962.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0004/$01.25/0

NUMERICAL SIMULATION OF HOT-ELECTRON PHENOMENA *

DANIEL S. WATANABE† AND SUMANTRI SLAMET†

Abstract. An accurate two-dimensional numerical model for MOS transistors incorporating avalanche processes is presented. The Laplace and Poisson equations for the electrostatic potential in the gate oxide and bulk and the current-continuity equations for the electron and hole densities are solved using finite-difference techniques. The current-continuity equations incorporate terms modeling avalanche generation, bulk and surface Shockley–Read–Hall thermal generation-recombination, and Auger recombination processes. The simulation is performed to a depth in the substrate sufficient to include the depletion region, and the remaining substrate is modeled as a parasitic resistance. The increase in the substrate potential caused by the substrate current flowing through the substrate resistance is also included. The hot-electron distribution function is modeled using Baraff's maximum anisotropy distribution function.

The model is used to study hot-electron phenomena including negative-resistance avalanche breakdown in short-channel MOSFET's and electron injection into the gate oxide. The model accurately predicts the positive-resistance branch of the drain current–voltage characteristic and could, in principle, predict the negative-resistance branch and the sustain voltage.

The gate injection current is computed by summing the flux of electrons scattered into the gate oxide by each mesh volume element. The number of electrons in each element whose component of momentum normal to the oxide is sufficient to surmount the oxide potential barrier is approximated using Baraff's distribution function, and scattering along the electron trajectories is modeled using an appropriate mean free path. The flux scattered into the oxide can be expressed as an iterated six-fold integral which is evaluated using the potential and electron current density distributions produced by the model. Nakagome *et al.* [1] recently observed two new types of gate injection phenomena: avalanche injection and secondary ionization induced injection. The former is caused by carriers heated in the drain avalanche plasma, and the latter is caused by electrons generated by secondary impact ionization in the depletion region. The model yields gate current curves qualitatively similar to the experimental results.

I. Introduction. The high electric fields in scaled-down MOSFET's cause carrier multiplication and the injection of electrons into the oxide. Avalanche multiplication occurs when the field in the drain-depletion region becomes high enough for carriers to gain sufficient energy to create electron–hole pairs by impact ionization. The holes flow to the substrate, and a positive space charge accumulates in the ionization region. The process increases the drain current and thereby increases the creation of electron–hole pairs. This positive-feedback mechanism leads to avalanche breakdown when the gain is larger than unity. Avalanche breakdown limits the maximum voltage applicable to a MOSFET and hence limits the speed and power-handling capacity of the device. Water-related centers in silicon dioxide can trap electrons. Since these centers are present even in "dry" thermal oxides, any electron current in the oxide causes oxide charging. In a short-channel device, the charging may affect a significant portion of the channel. The charging increases with time, and eventually the device threshold drifts far enough to disable the device. Electron injection into the oxide also can create interface traps which degrade the transconductance of the device. Because of this cumulative degradation, oxide charging limits the maximum voltage that can be applied to a MOSFET given a specified device lifetime and duty cycle. A detailed understanding of these phenomena clearly is essential for the design of reliable MOSFET's.

This paper presents an accurate two-dimensional numerical model for MOSFET's incorporating avalanche processes. The Laplace and Poisson equations for the electrostatic potential in the gate oxide and bulk and the current-continuity equations for the

*Received by the editors April 14, 1983, and in revised form June 8, 1983. This research was supported in part by the U.S. Department of Energy under Contract DE-AC02-76-ER02383.

†Department of Computer Science, University of Illinois, Urbana, Illinois 61801.

electron and hole densities are solved using finite-difference techniques. The current-continuity equations incorporate terms modeling avalanche generation, bulk and surface Shockley–Read–Hall thermal generation-recombination, and Auger recombination processes. The simulation is performed to a depth in the substrate sufficient to include the depletion region, and the remaining substrate is modeled as a parasitic resistance. The increase in the substrate potential caused by the substrate current flowing through the substrate resistance is also included. The hot-electron distribution function is modeled using Baraff's maximum anisotropy distribution function.

In the following sections, the model is first described. The numerical procedure for solving the model equations is then described. Finally the model is used to study negative-resistance avalanche breakdown in short-channel MOSFET's and electron injection into the gate oxide, and examples illustrating the accuracy and utility of the model are presented.

II. Device model. The device is modeled by the Poisson and current-continuity equations

$$\begin{aligned} -\nabla^2 u + n - p &= k, \\ -\nabla\cdot\mathbf{J}_n &= G - R, \\ \nabla\cdot\mathbf{J}_p &= G - R \end{aligned}$$

where

$$\begin{aligned} \mathbf{J}_n &= -\mu_n n\nabla u + D_n\nabla n, \\ \mathbf{J}_p &= -\mu_p p\nabla u - D_p\nabla p. \end{aligned}$$

Here u is the electrostatic potential, n and p are the electron and hole densities, G and R are the generation and recombination rates, $\mathbf{J}_n$ and $\mathbf{J}_p$ are the electron and hole current densities, μ_n and μ_p are the electron and hole mobilities, and D_n and D_p are the electron and hole diffusion coefficients. The equations are nondimensionalized with respect to the Boltzmann voltage kT/q, intrinsic carrier density n_i, and Debye length $(\varepsilon_s kT/q^2 n_i)^{1/2}$. Here kT is the thermal energy, q is the magnitude of the electron charge, and ε_s is the permittivity of silicon. In the gate oxide, the electrostatic potential u satisfies the Laplace equation

$$-\nabla^2 u = 0.$$

At the oxide interface, the normal component of the electrical displacement vector $\varepsilon\partial u/\partial y$ is required to be continuous. The applied voltages and doping provide Dirichlet boundary conditions for the electrostatic potential at the gate, source, drain, and substrate and for the electron and hole densities at the source and drain electrodes and substrate. Neumann boundary conditions are assumed for the potential at the vertical boundaries, and current is not allowed to flow across the oxide interface and vertical boundaries.

The avalanche generation, bulk and surface Shockley–Read–Hall thermal generation-recombination, and Auger recombination processes are modeled by

$$\begin{aligned} G_a &= \alpha_n|\mathbf{J}_n| + \alpha_p|\mathbf{J}_p|, \\ (G-R)_t &= (n_i^2 - np)/\left[\tau_n(p+p_1) + \tau_p(n+n_1)\right], \\ (G-R)_s &= \delta(y-y_i)(n_i^2 - np)/\left[(p+p_1)/s_n + (n+n_1)/s_p\right], \\ (G-R)_A &= (n_i^2 - np)(C_n n + C_p p) \end{aligned}$$

where $\delta(\cdot)$ is the Dirac delta function and $y = y_i$ at the oxide interface. It is assumed that carriers gain energy only from the component of the electric field **E** parallel to the current density. Hence the ionization coefficients depend only on $\mathbf{E}\cdot\mathbf{J}/|\mathbf{J}|$ and have the form

$$\alpha \equiv ae^{-b|\mathbf{J}|/|\mathbf{E}\cdot\mathbf{J}|}.$$

The values of the parameters are taken from the literature. For example, the mobilities are taken from Yamaguchi [2] and the ionization coefficients are taken from Niehaus *et al.* [3].

Fig. 1 presents the device geometry. The voltages are measured with respect to the source voltage. The simulation is restricted for efficiency to the depletion region and the adjacent substrate, and an effective substrate voltage V_E is used on the computational substrate boundary. The remaining substrate is modeled by a parasitic resistance R_B. The resistance can be estimated, but it is generally chosen to match the predicted and experimental drain currents.

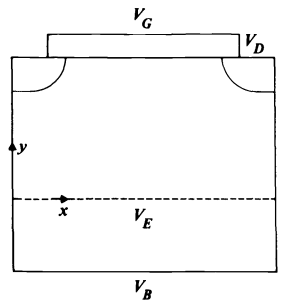


FIG. 1. *Device geometry.*

Baraff's maximum anisotropy approximation [4] to the hot-electron distribution function is

$$f(\mathbf{p}) = [m_0(E) + m_1(E)\cos(\theta)]/E$$

where $E = \mathbf{p}\cdot\mathbf{p}/2m$ is the energy of an electron with mass m, and θ is the angle between the electron momentum **p** and the local electric field **E**. The approximation is obtained by truncating the conventional spherical harmonics expansion of the velocity-distribution function under the assumption that all the electrons are traveling in the field direction. When $E \leqslant E_i$, the threshold energy for impact ionization, the distribution function is

$$m_0(E) = [Q/(3E_r)]\, m_1(E) + c,$$

$$m_1(E) = (Qbc/a)\left(1 + bE^{-a}e^{-bE}\int_E^{E_i} t^a e^{bt}\, dt\right) + (\alpha - Qbc/a)(E/E_i)^{-a} e^{b(E_i - E)}$$

where

$$Q = q\lambda_r|\mathbf{E}|,$$
$$a = (E_r - Q)/(2E_r + Q),$$
$$b = 3E_r/(2E_rQ + Q^2),$$
$$c = 1 - Q\alpha/(3E_r).$$

When $E > E_i$, the distribution function is

$$m_0(E) = (E/E_i)^{-a^*} e^{-b^*(E-E_i)},$$
$$m_1(E) = \alpha m_0(E)$$

where

$$a^* = (\alpha - 3)/(2\alpha + 3),$$
$$b^* = 3\alpha/(2\alpha Q^* + 3Q^*),$$
$$\alpha = \left\{3r + \left[r + 3(1-r)E_r^*/(2Q^*)\right]^2\right\}^{1/2} + r + 3(1-r)E_r^*/(2Q^*),$$
$$E_r^* = E_r\left[1 - (1-r)^{-1}\right]/\ln(1-r),$$
$$Q^* = Q\lambda_i/(\lambda_r + \lambda_i),$$
$$r = \lambda_r/(\lambda_r + \lambda_i).$$

Here E_r is the optical-phonon energy, λ_r is the mean free path for optical-phonon scattering, and λ_i is the mean free path for impact ionization. The approximation has several advantages: it is tractable, agrees well with numerical solutions of the Boltzmann equation in which the angular dependence is treated exactly, and reduces to Shockley's "spike" distribution which is valid in the limit $Q = E_r$ and Wolf's isotropic distribution which is valid in the limit $Q \gg E_r$.

Baraff's assumption that all electrons travel in the field direction is not always satisfied. Recall that the ionization coefficients are defined under the assumption that carriers gain energy only from the component of the electric field parallel to the current density. Consistency with this assumption requires that θ be redefined as the angle between the momentum $\mathbf{p}$ and the electron current density $\mathbf{J}$ and that the energy Q be redefined as $Q = q\lambda_r|\mathbf{E}\cdot\mathbf{J}|/|\mathbf{J}|$. The electron distribution is modeled using this modified distribution function aligned with the current density $\mathbf{J}$.

III. Numerical solution. The device is covered with a nonuniform mesh $\{(x_i, y_j)\}$. The mesh must be chosen with care to minimize the discretization error and the number of mesh points. Since it is impossible to determine an optimal mesh without knowing the solution, the mesh is refined adaptively during the computation so that the difference in the electrostatic potential u between any two mesh points is less than a prescribed value. This criterion is chosen because the ionization coefficients are exponentially sensitive to the electric field.

The equations are discretized using the box-integration method based on the divergence theorem

$$\int\int \nabla\cdot\mathbf{v}\,dx\,dy = \int \mathbf{v}\cdot\mathbf{n}\,ds.$$

Fig. 2 presents the influence region associated with the mesh point (x_i, y_j). Let $h_i \equiv x_{i+1} - x_i$ and $k_j \equiv y_{j+1} - y_j$. Approximating the line integral using the midpoint rule yields

$$\int \mathbf{v}\cdot\mathbf{n}\, ds \approx (h_{i-1}+h_i)(v_n - v_{n-1})/2 + (k_{j-1}+k_j)(v_m - v_{m-1})/2.$$

It is well known that this procedure when applied to Poisson's equation yields a system of difference equations with a symmetric positive definite coefficient matrix. Consider the electron current-continuity equation where $\mathbf{v} = \mathbf{J}$. If the Einstein relation $\mu = D$ is assumed, and the new dependent variable $\eta \equiv e^{-u}n$ is introduced, then the current density $\mathbf{J}$ can be written as

$$\mathbf{J} = De^{u}\nabla\eta.$$

If the x component of the current density $\mathbf{J}$ is assumed constant on the interval $[x_i, x_{i+1}]$, and the exponent u is modeled by the linear function interpolating $u_{i,j}$ and $u_{i+1,j}$, then it follows that

$$J_m \approx D_m e^{u_{i,j}} b(u_{i,j} - u_{i+1,j})(\eta_{i+1,j} - \eta_{i,j})/h_i$$

where the Bernoulli function $b(x) \equiv x/(e^x - 1)$. This procedure yields a system of difference equations $A(\mathbf{u})\boldsymbol{\eta} = \mathbf{k}$ with a symmetric positive definite coefficient matrix. The elements of the coefficient matrix $A(\mathbf{u})$ may be large because of the factor $e^{u_{i,j}}$. This factor can be "absorbed" by transforming $\boldsymbol{\eta}$ to $\mathbf{n}$ through the linear transformation $\mathbf{n} = D\boldsymbol{\eta}$, where the diagonal matrix $D \equiv (e^{u_{i,j}})$. This transformation yields a system of difference equations $AD^{-1}\mathbf{n} = \mathbf{k}$ for the electron density $\mathbf{n}$ with a nonsymmetric positive definite coefficient matrix. The hole current-continuity equation is discretized in a similar manner.

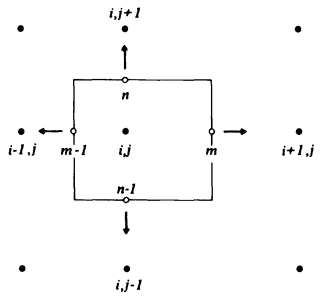


FIG. 2. *Box-integration method.*

The discretized equations have the form

$$A_1\mathbf{u} + \mathbf{n} - \mathbf{p} - \mathbf{k}_1 = 0,$$
$$A_2(\mathbf{u})\mathbf{n} - \mathbf{k}_2(\mathbf{u},\mathbf{n},\mathbf{p}) = 0,$$
$$A_3(\mathbf{u})\mathbf{p} - \mathbf{k}_3(\mathbf{u},\mathbf{n},\mathbf{p}) = 0$$

where the A's are sparse positive definite matrices, and the $\mathbf{k}$'s represent the doping profile k, the boundary conditions, and the generation and recombination terms.

For high fields $D \neq \mu$. The equations in this case are more complicated but have similar characteristics. The electron current density can be written as

$$\mathbf{J}_n = De^u \nabla \eta + (D - \mu)\eta e^u \nabla u$$

and the discretization procedure yields

$$J_m \approx D_m e^{u_{i,j}} b(u_{i,j} - u_{i+1,j})(\eta_{i+1,j} - \eta_{i,j})/h_i$$
$$+ (D - \mu)_m \left[(\eta_{i,j} + \eta_{i+1,j})/2\right] e^{u_{i,j}} b(u_{i,j} - u_{i+1,j})(u_{i+1,j} - u_{i,j})/h_i.$$

The difference equations have the form $[A(\mathbf{u}) + B(\mathbf{u})]\boldsymbol{\eta} = \mathbf{k}$. The matrix $B \equiv (b_{i,j})$ is generated by the second term in $\mathbf{J}$, and the elements contain a first-order difference of the potential u and satisfy $b_{i,j} = -b_{j,i}$ for $i \neq j$. For sufficiently small mesh stepsizes, the elements of B are small in magnitude relative to the corresponding elements of A, and it can be shown that the coefficient matrix $A + B$ is an M-matrix. However, the Einstein relation is assumed for simplicity in the following sections.

The discretized equations are solved using the sequential approach first used by Gummel. The equations are decoupled and solved sequentially until a self-consistent solution is obtained. The Newton-LSOR method is used to solve Poisson's equation, and the LSOR method is used for the continuity equations. These methods were chosen for simplicity after experiments with various methods including the preconditioned conjugate-gradient method and sparse-direct methods. The mobility and generation and recombination terms are evaluated periodically for efficiency.

IV. Avalanche breakdown. The breakdown phenomenon determines the maximum voltage applicable to a MOSFET, and hence limits the speed and power-handling capacity of the device. Normal breakdown occurs in long-channel n-MOSFET's, where the drain current increases rapidly at a breakdown voltage which increases with increasing gate voltage. Negative-resistance breakdown occurs in short-channel n-MOSFET's. The drain current increases rapidly at a breakdown voltage which decreases with increasing gate voltage. With further increases of the drain current, the drain voltage decreases to a sustain voltage following a current-controlled negative-resistance characteristic. The sustain voltage puts a practical limit on the maximum voltage applicable to a MOSFET, and hence is as important as the transconductance or threshold voltage in device design.

Eitan and Frohman-Bentchkowsky [5] explain avalanche breakdown as follows. Impact ionization by hot electrons in the drain-depletion region creates electron–hole pairs. The electrons are removed through the drain terminal, and the holes flow toward lower potential terminals. A positive space charge accumulates in the ionization region because the hole-collection efficiency and mobility are lower than those of electrons. They propose the following current-enhancement mechanisms: 1) the net positive charge in the pinchoff region increases the inversion-layer conductivity which, in turn, increases the source current; 2) the hole-current flow through the depletion region toward the substrate lowers the potential barriers between the source and channel and drain which, in turn, increases the current injected from the source and channel to the drain; and 3) the hole-current flow through the neutral-substrate resistance increases the substrate potential, resulting in current injection from the source through the substrate to the drain. Fig. 3 presents the current components in avalanche breakdown. The increased current then increases the impact-ionization current. This positive-feedback mechanism leads to avalanche breakdown when the gain is larger than unity. Eitan and Frohman-Bentchkowksy present experimental results and a qualitative

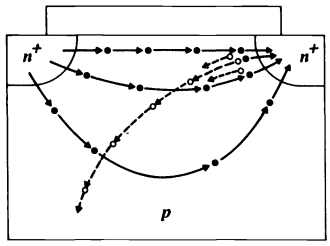


FIG. 3. *Current components in avalanche breakdown.*

model demonstrating the primary importance of the first two current-enhancement mechanisms.

The effective substrate-to-source voltage is

$$V_E = -I_B R_B + V_B.$$

The substrate current I_B is a function of u, n, and p which are functions of the effective substrate voltage. Hence the problem reduces to the solution of the scalar nonlinear equation

$$F(V_E) \equiv I_B - (V_B - V_E)/R_B.$$

Note that the evaluation of F requires the solution of the system of nonlinear partial differential equations describing the device.

Toyabe *et al.* [6] analyzed avalanche breakdown using the Poisson and electron current-continuity equations with $G = R = 0$. The equations are solved and the solution is used to evaluate the ionization integral

$$I = \int \alpha_n \exp\left(-\int (\alpha_n - \alpha_p)\, d\eta\right) d\xi$$

along the channel-current path to obtain the multiplication factor $M = 1/(1-I)$ describing the increase in the drain current due to avalanche multiplication. The increase in the substrate potential caused by the substrate current flowing through the substrate resistance is also included in their model. This approach, while efficient, ignores the primary current-enhancement mechanisms. Kotani and Kawazu [7] use a similar approach but include the avalanche-multiplication term in the electron-continuity equation. However, they assume a zero ionization rate for holes and ignore the effect of the substrate current on the substrate potential. Schütz *et al.* [8], [9] use the present model equations, but their procedure is designed to compute only the positive-resistance branch of the current–voltage characteristic. The present procedure, however, yields both branches of the characteristic. For a given gate and drain voltage, the substrate potential is incremented in small steps until the first root of $F(V) = 0$ corresponding to the lower branch is bracketed. The process is repeated for the second root corresponding to the upper branch. This scheme was used for simplicity, but it is possible to use more sophisticated root-finding algorithms.

The model was applied to a device considered by Toyabe *et al.* [6]. The device parameters are: effective channel length 2 μm, width 13 μm, gate-oxide thickness 48 nm, substrate doping 3.2×10^{21} m^{-3}, drain-junction depth 0.4 μm, surface concentration of diffused layer 2.4×10^{26} m^{-3}, flatband voltage -0.91 V, and applied substrate

voltage 0 V. Fig. 4 presents the computed and experimental current–voltage characteristics for $V_G = 1$ V. A substrate resistance of 9 kΩ is assumed. Toyabe *et al.* were forced to use a resistance of 20 kΩ to match the data because they ignored two of the three current-enhancement mechanisms. The predicted results are in good agreement with the experimental results. Figs. 5, 6, and 7 present the electron density, hole density, and electrostatic potential distributions for $V_G = 1$ V and $V_D = 8$ V. Consider the hole density distribution. All holes outside the undisturbed bulk region are generated by impact ionization. Note the accumulation of holes at the interface and near the source. The holes near the source lower the source-potential barrier. The distribution demonstrates that the holes do not flow directly to the bulk substrate, but instead first flow toward the interface, then toward the source, and finally into the substrate. Consider the electrostatic potential distribution. The lateral electric field is small in the channel and very large in the pinchoff region. Hence impact ionization should occur mainly in that region, and the hole density distribution confirms this expectation. The transverse electric field attracts the holes to the interface near the drain, but repels them in the adjacent channel region.

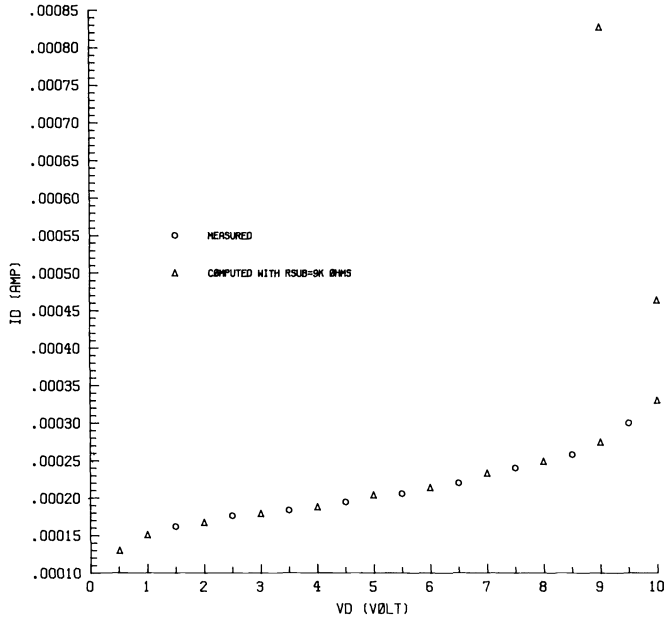


FIG. 4. *Drain current versus drain voltage.*

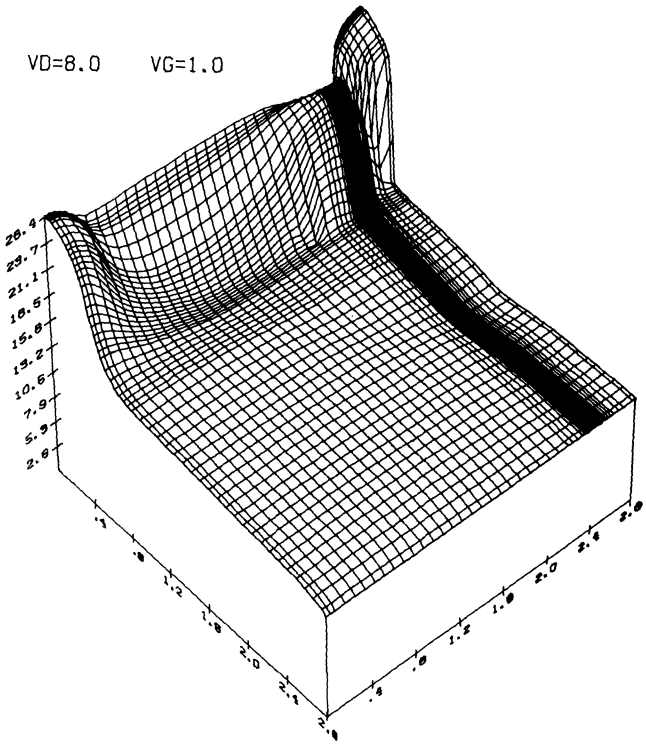


FIG. 5. *Electron density distribution.*

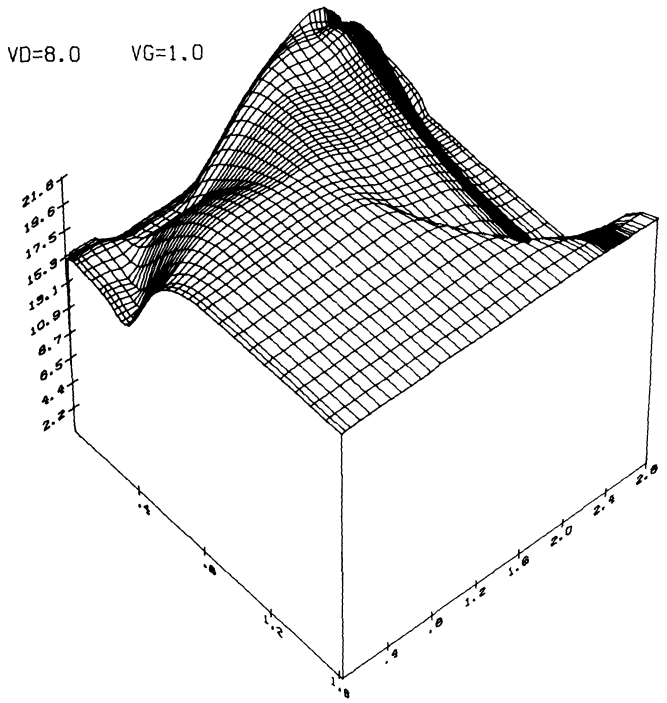


FIG. 6. *Hole density distribution.*

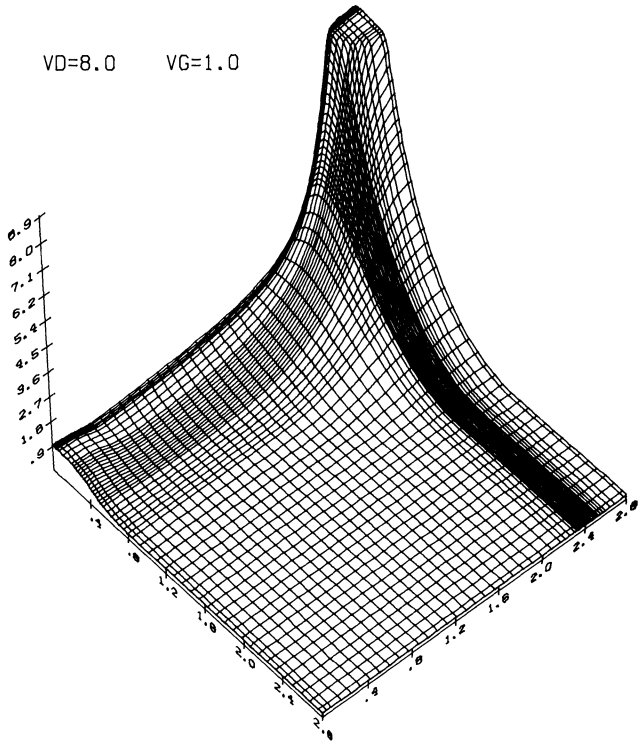


FIG. 7. *Electrostatic potential distribution.*

V. Gate injection current. Nakagome *et al.* [1] recently observed two new types of hot-carrier injection phenomena in short-channel MOSFET's: avalanche injection and secondary ionization induced injection. They attribute avalanche injection to carriers heated in the avalanche region between the pinchoff point and the drain junction. Holes are injected into the gate in the region near the drain where the y component of the electric field $E_y > 0$, while electrons are injected in the region where $E_y < 0$. Since the point at which $E_y = 0$ moves toward the drain as V_G increases, electron injection should dominate at higher gate voltages. They attribute secondary ionization induced injection to electrons generated by secondary impact ionization. Holes generated in the drain-avalanche region are accelerated toward the substrate and generate electron–hole pairs in the depletion region. Some of the excess electrons are accelerated toward the surface and are injected into the gate. Figs. 8 and 9 illustrate these injection mechanisms.

The gate-injection current is computed by summing the flux of electrons scattered into the gate by each mesh element in the plane $z = 0$. It is assumed that an electron originating at the point $(x, y, 0)$ will cross the oxide potential barrier if: 1) it is directed at a point (x_i, y_i, z_i) on the interface where the y component of the electric field $E_y < 0$; 2) the y component of the electron momentum $p_y > (2mE_e)^{1/2}$, where the effective potential barrier

$$E_e \equiv E_b + u(x, y, 0) - u(x_i, y_i, z_i)$$

and E_b is the silicon silicon-dioxide potential barrier; and 3) the electron is not scattered by an optical phonon. Assumption 2) implies that the difference in potential

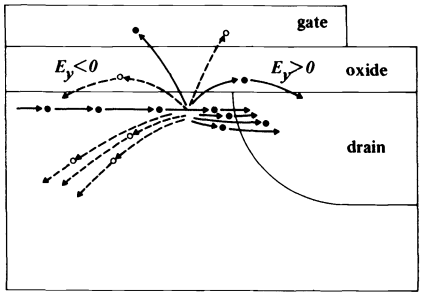


FIG. 8. *Avalanche injection.*

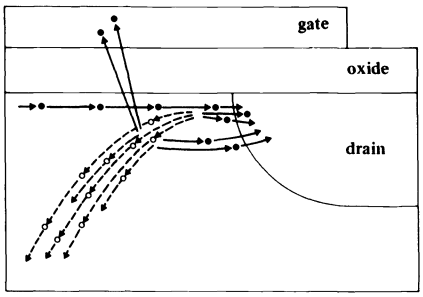


FIG. 9. *Secondary ionization induced injection.*

energy between the points $(x, y, 0)$ and (x_i, y_i, z_i) is converted into kinetic energy. The fraction of the electrons in the neighborhood of $(x, y, 0)$ scattered into the gate is the injection efficiency

$$\Gamma = \int\int_S \int p_y e^{-d/\lambda_r} f(\mathbf{p})\, d\mathbf{p} \Big/ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\mathbf{p}\cdot\mathbf{J}/|\mathbf{J}|) f(\mathbf{p})\, d\mathbf{p}$$

where S is the set of $\mathbf{p}$ for which $\mathbf{p}$ intersects the gate at a point (x_i, y_i, z_i) where $E_y < 0$ and $p_y > (2mE_e)^{1/2}$, and d is the distance from $(x, y, 0)$ to the point of intersection. The factor e^{-d/λ_r} is the probability that an electron reaches the interface without being scattered by an optical phonon. The denominator in Γ is the modulus of the electron current density $|\mathbf{J}|$ at $(x, y, 0)$. Now $\int \Gamma |\mathbf{J}|\, dy$ is the flux scattered into the gate as the current passes through the plane with fixed value x. Hence the gate current is given by

$$I_G = \int\int \Gamma |\mathbf{J}|\, dy\, dx \Big/ \int dx.$$

Note that this is a six-fold integral because the distribution function $f(\mathbf{p})$ contains an integral.

The injection efficiency Γ at the point $(x, y, 0)$ is more conveniently written in terms of a new set of variables X_i, ϕ, and E. Let (X, Y, Z) denote the coordinate system with origin at the point $(x, y, 0)$, and let (X_i, Y_i, Z_i) denote the point of intersection. Let ψ denote the angle between the X axis and $\mathbf{J}$, and recall that θ is the angle between the momentum $\mathbf{p}$ and $\mathbf{J}$. Let ϕ denote the angle between $\mathbf{p}$ and the line passing through the origin and $(X_i, Y_i, 0)$, and let E denote the energy of an electron measured from the conduction band edge. Then

$$p_x = (2mE)^{1/2}\cos(\phi) X_i/\delta,$$

$$p_y = (2mE)^{1/2}\cos(\phi) Y_i/\delta,$$

$$p_z = (2mE)^{1/2}\sin(\phi)$$

where $\delta = (X_i^2 + Y_i^2)^{1/2}$. The numerator in Γ can be written as

$$4mY_i^2 \int_{a(x)}^{b(x)} dX_i \int_0^{\pi/2} d\phi \int_{E_0(X_i,\phi)}^{\infty} dE f(X_i, \phi, E)$$

where

$$E_y < 0 \quad \text{for } a(x) \leqslant X_i \leqslant b(x), \qquad E_0 = E_e\{\delta/[Y_i\cos(\phi)]\}^2,$$

and

$$f(X_i,\phi,E) = \cos^2(\phi)e^{-d/\lambda_r}[m_0(E)+m_1(E)\cos(\theta)]/\delta^3,$$
$$d = \delta/\cos(\phi),$$
$$\cos(\theta) = \cos(\phi)[X_i\cos(\psi)+Y_i\sin(\psi)]/\delta.$$

It is impractical to evaluate the gate current using only numerical quadrature methods because it involves a six-fold integral. A combination of numerical and analytical techniques is required. The integration with respect to E is performed analytically. For example, the integral term in $m_1(E)$ for $E \leqslant E_i$ has the series expansion

$$bE^{-a}e^{-bE}\int_E^{E_i} t^a e^{bt}\,dt = (E/E_i)^{-a}\sum_{n=1}^{\infty} c_n b^n (E_i - E)^n/n!$$

where the coefficient c_n is the series

$$c_n = 1 - a/(bE_i) + a(a-1)/(bE_i)^2 + \cdots$$
$$+(-)^{n-1}a(a-1)\cdots(a-n+2)/(bE_i)^{n-1}.$$

Hence a series expansion of $\int_E^{E_i} m_1(E)\,dE$ can be derived by integrating this series. The integration yields the integral

$$\int_E^{E_i}(t/E_i)^{-a}b^n(E_i-t)^n\,dt = E_i(bE_i)^n\int_{E/E_i}^1 t^{-a}(1-t)^n\,dt$$

which is related to the incomplete beta function. An integration by parts yields

$$\int_\tau^1 t^{-a}(1-t)^n\,dt = -\tau^{1-a}(1-\tau)^n/(1-a)$$
$$-\tau^{2-a}p(1-\tau)^{n-1}/[(1-a)(2-a)] - \cdots$$
$$-\tau^{n+1-a}p!/[(1-a)(2-a)\cdots(n+1-a)]$$
$$+p!/[(1-a)(2-a)\cdots(n+1-a)].$$

Thus the coefficients in the series expansion are themselves products of series. The integral

$$\int_E^\infty m_0(t)\,dt = \int_E^\infty (t/E_i)^{-a^*}e^{-b^*(t-E_i)}\,dt$$

for $E > E_i$ can be written as $(b^*E_i)^{a^*-1}E_i e^{b^*E_i}\Gamma(1-a^*, b^*E)$, where $\Gamma(\alpha,\beta)$ is the incomplete gamma function. Hence this integral can be evaluated using the power series or continued-fraction representations of the incomplete gamma function. The representations are

$$\Gamma(1-a^*, b^*E) = \Gamma(1-a^*) - (b^*E)^{1-a^*}\sum_{n=0}^{\infty}(-b^*E)^n/[(1-a^*+n)n!]$$

and

$$\Gamma(1-a^*, b^*E) = e^{-b^*E}(b^*E)^{1-a^*} \cdot \left(\frac{1}{b^*E+} \frac{a^*}{1+} \frac{1}{b^*E+} \frac{a^*+1}{1+} \frac{2}{b^*E+} \frac{a^*+2}{1+} \cdots \right).$$

The more rapidly convergent of the two representations of $\Gamma(1-a^*, b^*E)$ is used for any given set of arguments. Aitken's δ^2-method is used, if necessary, to accelerate the convergence of the various series and continued-fraction representations. The remaining integrations with respect to ϕ and X_i are performed numerically using the adaptive quadrature routine QUANC8 [10] which is based on the eight-panel Newton–Cotes rule. The product $\Gamma|\mathbf{J}|$ is evaluated at the centers $(x_{i+1/2}, y_{j+1/2})$ of the relevant mesh elements and the gate current is approximated using the midpoint rule.

Phillips *et al.* [11] first used the present approach with a simple model of the electron distribution function. Eitan and Frohman-Bentchkowsky [12] present a qualitative model based on similar assumptions. Rather than evaluate their integral expression for the gate current, they use it to provide an explanation of the gate-current parameter and voltage dependence and to predict a correlation between the gate and substrate currents. Wada *et al.* [13] combine an expression for the injection efficiency based on Baraff's distribution function and a uniform channel electron density with the scattering factor e^{-d/λ_r}. They also assume that an electron will reach the gate only if $\tan^{-1}(E_y/E_x)$ is greater than a critical injection angle. By assuming a critical angle of 60 degrees, they obtain good agreement between the computed and experimental results.

The preceding papers are concerned only with the channel-injection current due to channel electrons accelerated by the high field near the drain junction. The present model, however, includes avalanche generation and should, in principle, predict the avalanche-injection phenomena reported by Nakagome *et al.* They report measurements for two types of devices. For the first the electron-injection current versus gate-voltage curve exhibits a secondary peak at $V_G \approx 3.5$ V attributed to avalanche injection as well as the primary peak at $V_G \approx V_D$ associated with channel injection. For the second the channel-injection current is suppressed, the gate-current curve is generally unimodal, and the gate current is strongly correlated with the substrate current. The peak in the gate-current curve occurs at a gate voltage slightly higher than that for the substrate-current curve. At sufficiently high drain voltages, however, the gate-current curve is bimodal. Both devices have ion-implanted channels. Since the doping profiles are proprietary information, a hypothetical short-channel n-MOSFET with a uniformly doped substrate was studied instead. The device parameters are: effective channel length 1 μm, width 14 μm, gate-oxide thickness 10 nm, substrate doping 3.5×10^{22} m^{-3}, drain-junction depth 0.2 μm, surface concentration of diffused layer 5.5×10^{26} m^{-3}, and applied substrate voltage 0 V. Since the substrate doping is uniform, the device presumably may exhibit a combination of the properties of the two devices studied by Nakagome *et al.* Figs. 10 and 11 present the gate and substrate currents for $V_D = 3.0$, 3.5, 4.0, 4.5, 5.0, and 5.5 V. The gate current is strongly correlated as expected with the substrate current for $V_D \leqslant 4.5$ V, and the first peak in the gate-current curve occurs at a gate voltage slightly higher than that for the peak in the substrate-current curve. It is tempting to speculate that the peak in the gate-current curve at $V_G \approx 3.5$ V for $V_D = 3.5$, 4.0, and 4.5 V is due to avalanche injection. For

$V_D = 5.0$ and 5.5 V, however, the gate-current curve resembles the normal unimodal channel-injection current curve. The substrate current saturates as expected with increasing drain bias. It saturates because the source-substrate junction becomes forward biased, and the substrate current saturates at the value needed to make the voltage drop across the substrate equal the forward bias.

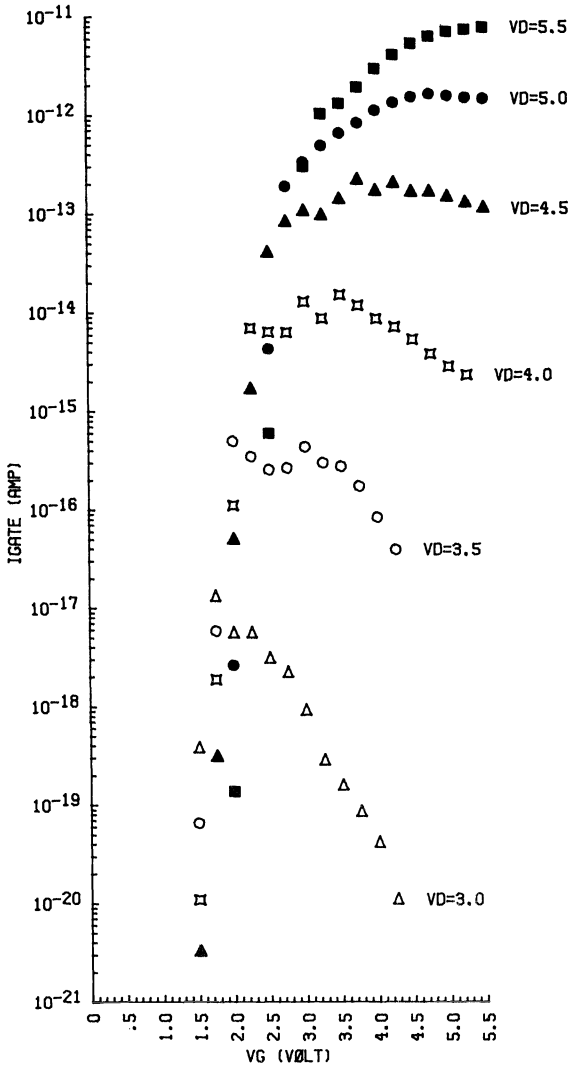


FIG. 10. *Gate current versus gate voltage.*

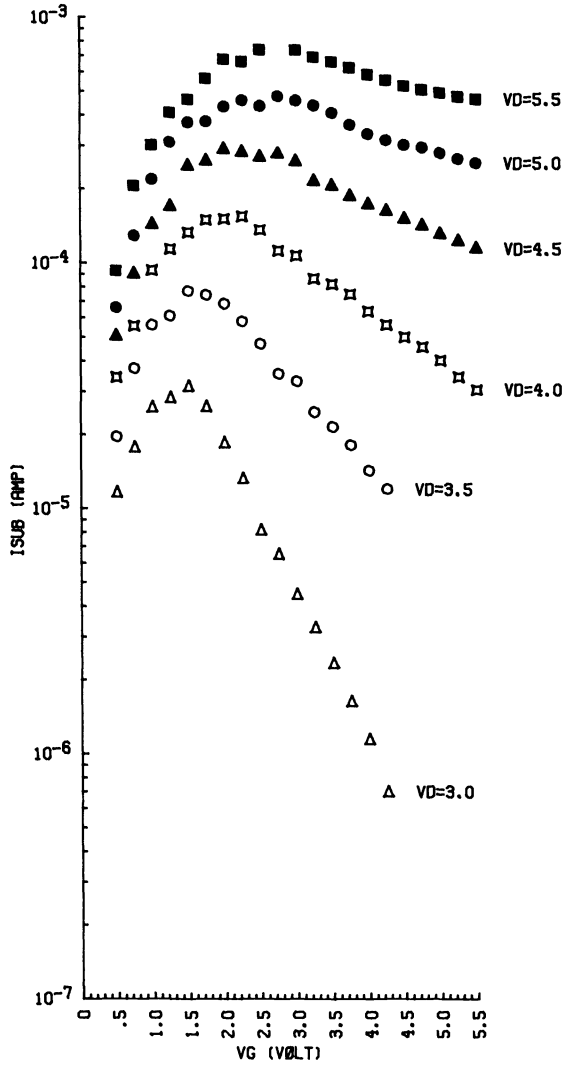


FIG. 11. *Substrate current versus gate voltage.*

Acknowledgment. The authors wish to thank J. Murata of Hitachi, Ltd., for his invaluable assistance while he was a Visiting Research Associate at the Department of Computer Science of the University of Illinois.

REFERENCES

[1] Y. NAKAGOME, E. TAKEDA, H. KUME, AND S. ASAI, "New observation of hot-carrier injection phenomena," in *ICSSD Dig. Tech. Papers*, pp. 63–64, 1982.

[2] K. YAMAGUCHI, "Field-dependent mobility model for two-dimensional numerical analysis of MOSFET's," *IEEE Trans. Electron Devices*, vol. ED-26, pp. 1068–1074, 1979.

[3] W. C. NIEHAUS, T. E. SEIDEL, AND D. E. IGLESIAS, "Double-drift IMPATT diodes near 100 GHz," *IEEE Trans. Electron Devices*, vol. ED-20, pp. 765–771, 1973.

[4] G. A. BARAFF, "Maximum anisotropy approximation for calculating electron distributions; application to high field transport in semiconductors," *Phys. Rev.*, vol. 133, pp. A26–A33, 1964.

[5] B. EITAN AND D. FROHMAN-BENTCHKOWSKY, "Surface conduction in short-channel MOS devices as a limitation to VLSI scaling," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 254–266, 1982.

[6] T. TOYABE, K. YAMAGUCHI, S. ASAI, AND M. MOCK, "A numerical model of avalanche breakdown in MOSFET's," *IEEE Trans. Electron Devices*, vol. ED-25, pp. 825–832, 1978.

[7] N. KOTANI AND S. KAWAZU, "A numerical analysis of avalanche breakdown in short-channel MOSFETs," *Solid-State Electron.*, vol. 24, pp. 681–687, 1981.

[8] A. SCHÜTZ, S. SELBERHERR, AND H. W. PÖTZL, "A two-dimensional model of the avalanche effect in MOS transistors," *Solid-State Electron.*, vol. 25, pp. 177–183, 1982.

[9] A. SCHÜTZ, S. SELBERHERR, AND H. W. PÖTZL, "Analysis of breakdown phenomena in MOSFET's," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 77–85, 1982.

[10] G. E. FORSYTHE, M. A. MALCOLM, AND C. B. MOLER, *Computer Methods for Mathematical Computations*. Englewood Cliffs, NJ: Prentice-Hall, 1977, pp. 102–105.

[11] A. PHILLIPS JR., R. R. O'BRIEN, AND R. C. JOY, "IGFET hot electron emission model," in *IEDM Tech. Dig.*, pp. 39–42, 1975.

[12] B. EITAN AND D. FROHMAN-BENTCHKOWSKY, "Hot-electron injection into the oxide in n-channel MOS devices," *IEEE Trans. Electron Devices*, vol. ED-28, pp. 328–340, 1981.

[13] M. WADA, T. SHIBATA, M. KONAKA, H. IIZUKA, AND R. L. M. DANG, "A two-dimensional computer simulation of hot carrier effects in MOSFET's," in *IEDM Tech. Dig.*, pp. 223–226, 1981.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0005/$01.25/0

AUTOMATIC PROBLEMSIZE REDUCTION FOR ON-STATE SEMICONDUCTOR PROBLEMS*

SIMON J. POLAK,† WILLY H. A. SCHILDERS,† COR DEN HEIJER,†
ARTHUR J. H. WACHTERS,† AND HARRY M. VAES†

Abstract. Two algorithms are presented for the analysis of a large class of on-state semiconductor problems.

The algorithms only involve the strictly necessary equations in the discretized problem.

Equations that are satisfied and solution values that are found are not unnecessarily present in the calculations. Some examples of their use are shown.

I. Introduction. The analysis of 2D on-state semiconductor problems usually involves a large computational effort. Special device properties are therefore used to minimize computing times.

In analyzing a MOS device, for example, one may omit the equations for the minority carriers or only calculate the quasi-Fermi level in the channel.

The two algorithms presented here automatically recognize device properties that can be used to omit equations and unknowns from the discretized problem.

The basis for the solution of the semiconductor problems with both algorithms is the following: The continuous problems are discretized using a FEM. Then a continuation method [6], [11] is used to step up from a nearby problem, e.g., with a lower applied voltage or with a different dopant, to the problem to be solved. At each step of the continuation method a damped Newton method is used.

In both algorithms the set of discrete equations involved in one damped Newton iteration is kept as small as possible in some sense. In one algorithm we always start with the full set of equations and omit equations thus reducing the set until it is empty. Then we return to the full set as a safeguard. This algorithm is called *Down Subsetsolving* (D3S).

In the other algorithm we start with only those equations giving large residuals from the initial estimate for a Newton process. Then, after one iteration on this (usually small) set we omit all equations with small residuals involving only small corrections and we omit the associated variables. Furthermore, equations and variables outside the original set which are associated with too large residuals are added to the set. This algorithm is called *Up* and *Down Subsetsolving* (U + D3S).

We have developed a program package CURRY for the solution of 2D time independent on-state semiconductor problems. Both 2D MOS and transistor devices can be analyzed with it. For the solution one can choose either of the two algorithms. Some examples and comparisons are presented in this paper.

In Section II we describe the basic equations and some notations. In Section III we briefly present the continuation method and the damped Newton method. In Section IV the linear algebra is discussed. In Section V the *Down Subsetsolving* algorithm and in Section VI the *Up* and *Down Subsetsolving* algorithm are presented. Section VII contains remarks on convergence of both algorithms. In Section VIII some practical remarks concerning the algorithms are given. In Section IX the package CURRY is briefly described together with some implementation remarks. In Section X we give examples calculated with both algorithms.

*Received by the editors November 2, 1982, and in revised form May 16, 1983.
†Philips Research Laboratories, Eindhoven, the Netherlands.

II. Fundamental equations and notations. The equations to be solved have been much discussed, e.g., [5] and [10]. Therefore it is sufficient to give the equations here without much comment.

Furthermore, the algorithms presented in this paper may be used for other classes of problems as well. So a symbolic representation of the problem often will be enough for the discussion in this paper. The equations are:

$$(1) \qquad -\operatorname{div}\varepsilon \operatorname{grad}\psi - \rho(\psi,\phi_n,\phi_p) = F_1(\psi,\phi_n,\phi_p) = 0,$$

$$(2) \qquad -\operatorname{div}\mu_n n(\psi,\phi_n)\operatorname{grad}\phi_n + R(\psi,\phi_n,\phi_p) = F_2(\psi,\phi_n,\phi_p) = 0,$$

$$(3) \qquad -\operatorname{div}\mu_p p(\psi,\phi_p)\operatorname{grad}\phi_p - R(\psi,\phi_n,\phi_p) = F_3(\psi,\phi_n,\phi_p) = 0$$

where

$$p(\psi,\phi_n,\phi_p) = q\big(p(\psi,\phi_p) - n(\psi,\phi_n) + D\big)$$

and

$$p(\psi,\phi_p) = n_i \exp\big((q/KT)(\phi_p - \psi)\big)$$

and

$$n(\psi,\phi_n) = n_i \exp\big((q/KT)(\psi - \phi_n)\big).$$

R is the recombination and may be of SRH or Auger type [8]. They are discretized with a FEM with bilinear quadrilaterals and linear triangles as described for (1) in [6]. Then we have a set of N nonlinear equations with N unknowns indicated by $r_i(x) = 0$, $i = 1,\cdots,N$ or $R(\mathbf{x}) = 0$ where $\mathbf{x}$ is the vector of nodal ψ, ϕ_n, and ϕ_p values. S will indicate the index set $\{i\}_{i=1}^{N}$. Equation and unknown will be termed associated if they have the same index $i \in S$.

III. The continuation method. Descriptions of the continuation method can be found in [4] and [7]. A brief explanation follows here.

Suppose we want to solve a nonlinear $N \times N$ set of equations, $R(\mathbf{x}) = 0$. Then we replace the equation by a family of sets of equations $R(\mathbf{x}(t), t) = 0$, $t \in [0,1]$ with $\mathbf{x}(0)$ known or easily calculated and $R(\mathbf{x}(1),1) = R(\mathbf{x}(1)) = 0$. Let $t_0 = 0$; for t_0 the solution is known. If the solution has been found for $t = t_{k-1}$, $k \geqslant 1$, then a small enough step τ_k will give a convergent Newton process for $t = t_k = t_{k-1} + \tau_k$ with $\mathbf{x}(t_{k-1})$ as an initial estimate.

We use the following damped Newton method:

1) calculate $dx = -J^{-1}R(\mathbf{x})$, J is the Jacobian of $R(\mathbf{x})$
2) if $\|dx\| \leqslant \varepsilon_{abs} + \varepsilon_{rel} * \|x\|$, terminate
3) else calculate $\bar{\lambda}$ with
   $\|R(\mathbf{x} + \bar{\lambda}\, dx)\|_2 \leqslant \|R(\mathbf{x} + \lambda\, dx)\|$ for all $\lambda \geqslant 0$
4) set $x := x + \bar{\lambda}\, dx$, return to 1).

In the subset solving algorithms we also encounter the following situation. Suppose $S = S_1 \cup S_2$, $S_1 \cap S_2 = \varnothing$.

J_p is the submatrix of J with coefficients $j_{m,n}$ with $m, n \in S_1$. So all the rows and columns with indices in S_2 are omitted. Also $R_p(\mathbf{x})$ is the vector of $r_i(\mathbf{x})$ with $i \in S_1$. Then $dx_1 = -J_p^{-1}R_p(\mathbf{x})$. dx is defined by assigning the dx_1 components to the associated dx components and by setting all other components to zero. $\bar{\lambda}$ is established from $\|R_p(x + \bar{\lambda}\, dx)\|_2 \leqslant \|R_p(x + \lambda\, dx)\|_2$ for all $\lambda \geqslant 0$. $\bar{\lambda}$ is obtained numerically by the method of Davies, Swann, and Campey (see [15]).

IV. Linear algebra. The problem $Jdx = -R(\mathbf{x})$ is equilibrated [12] and all the considerations involving J and $R(\mathbf{x})$ concern the equilibrated problem. Only in the damping procedure the equilibration factors are kept constant as updating them would involve a re-evaluation of J for each λ. (For equilibration, optimal scaling etc., see e.g. [1], [9].) The Jacobian J can be conceived as a 3×3 block matrix, with main diagonal blocks $(\partial/\partial\psi)F_1$, $(\partial/\phi_n)F_2$, and $(\partial/\partial\phi_p)F_3$ (see also Section II.)

Instead of $Jdx = -R(\mathbf{x})$, we solve the preconditioned system $J^*dx = -R(\mathbf{x})^*$, where $J^* = (L+D)^{-1}J$ and $R(\mathbf{x})^* = (L+D)^{-1}R(\mathbf{x})$. Here $(L+D)$ is the block lower-diagonal part of J. Then we use ORTHOMIN (see [3], [14]) to solve this problem. In J^* we use the $L-U$ decomposition of the main diagonal blocks. We are investigating cheaper preconditionings.

V. Down subsetsolving (D3S). The simplest way to reduce the number of equations in the discrete set is the following.

After each iteration of the Newton algorithm select those corrections satisfying the convergence criteria and omit the associated equations and variables from the set until it is empty. Then again restart the process with the full set. Terminate if the empty set is found with one Newton iteration for the whole set. More precisely the algorithm has the following form:

$t = 0$, set x, $T = 0$, $R = S$
while $t < 1$
do evaluate τ, $t = t + \tau$
  while $T \neq R$
  do $R = S$
    while $R \neq 0$
    do $S_1 = R$, $S_2 = S \setminus R$
      λdx is calculated as in Section III, $x = x + \bar{\lambda} dx$
      $T = R$
      $R = \{ i \in R \| dx_i | > C \}$
    od
  od
od

Remark 1. The disadvantage of this algorithm is that the total set of equations has to be solved at least twice per Newton process whereas most of the solutions may remain unchanged. In the U+D3S algorithm this is not the case.

Remark 2. We decide that the algorithm diverges if
(the $\|dx\|$ increases "OR" $\|dx\| > M\|x\|$)"AND" $\bar{\lambda}$ decreases
"AND" if the number of points in S^1 stays constant
then the step τ_k is decreased.
We chose $M = 10^3$, because of the expected range of the solution.

VI. Up and down subsetsolving (U+D3S). A more complicated algorithm that keeps the set of equations each Newton iteration as small as possible is the following. Select the equations with a large residual from the substituted initial estimate. This usually is a far smaller set than the total set. After each following Newton iteration those equations and associated variables with both a small residual and a small associated correction are omitted. Also equations with too large a residual are added.

This way we start with a small set and keep the set small if possible. More precisely the algorithm has the following form:

$t = 0$, set x
while $t < 1$
do evaluate τ, $t = t + \tau$
  $T = \{i \in S \| r_i(\mathbf{x})| > C = \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \|x\|\}$
  while $T \neq 0$
  do $S_1 = T$, $S_2 = S \setminus T$
    calculate $\bar{\lambda} dx$ as in Section III, $x = x + \bar{\lambda} dx$
    $T_d = \{i \in T \| dx_i| \geqslant C\}$
    $T_r = \{i \in S \| r_i(\mathbf{x})| \geqslant C\}$
    $T = T_r \cup T_d$
  od
od

Remark 1. Although the set is kept minimal, the process of finding the largest of the S^1 does involve an amount of work (see examples) that decreases the expected advantage. Therefore one might consider further combinations of D3S and U + D3S. Starting with a larger set than is done is U + D3S, e.g., the second set of the previous iteration may be faster. We are investigating such possibilities. See also Remark 1 in Section XI.

Remark 2. S^1 can be established by considering only

$$\left\{ i \left| \frac{\partial r_i}{\partial x_m} \neq 0 \text{ and } \frac{\partial r_m}{\partial t} \neq 0 \text{ for some } m \right. \right\}.$$

Remark 3. Suppose the emitter potential is increased in step k of the continuation process from V_{k-1} to V_k. Then the set S_1 will contain only indexes of the points adjacent to the emitter contact. The set then will grow to involve all points influenced by the voltage change, though the original points may drop out earlier.

Remark 4. We take the criterion $C = \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \|x\|$. This is a necessary condition (not sufficient) for $\|dx\| \leqslant \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \|x\|$ as can easily be seen from the equilibrated problem.

Remark 5. We decide that the algorithm diverges if
(($\|dx\|$ increases "OR" $\|dx\| > M\|x\|$)"AND" $\bar{\lambda}$ decreases)
"OR" if $\bar{\lambda} < \varepsilon_\lambda$
"OR" if $\|dx\| > N$,
then the step τ_k is decreased.
We chose $M = 10^3$, $N = 10^4$, and $\varepsilon_\lambda = 10^{-4}$.

VII. Convergence considerations. There is no proof of the convergence of the algorithms presented. However, it is reasonable to expect the Newton process for the full problem to converge if τ_k is small enough. It is less obvious that this is also the case for the subsetsolving process. Still, one Newton iteration for a subset may be considered out of a Newton process for this subset. Then if the previous iterand had been sufficiently close, which for a small enough τ_k is the case, the new iterand will be closer. No thoroughness is pretended in this reasoning of course.

Let us now consider the functional $\|R(\mathbf{x})\|_2$. This functional has a minimum zero for the solution. In the algorithms presented, there is no guarantee that the functional will decrease from step to step even when a $\bar{\lambda} > 0$ is found. This is so because the

corrections dx_i, $i \in S^1$ may change some $r_m(\mathbf{x})$ with $m \notin S^1$. One might consider the adding of all those $r_m(x + \lambda\, dx)^2$ with $(\partial/\partial x_i) r_m(\mathbf{x}) \neq 0$ to the functional giving the functional $\|\underline{R}(\mathbf{x})\|_2$. However, this easily introduces the situation where the Newton direction dx is not a descent direction for the functional. So $\bar{\lambda} = 0$. Or the functional generates a series of λ's converging to zero. Theoretically the descent direction can be found by not solving $J\,dx = -R(\mathbf{x})$ but by taking the overdetermined Jacobian $\underline{J}$ with coefficients $(\partial/\partial x_i) r_m(\mathbf{x})$ for $(\partial/\partial x_i) r_m(\mathbf{x}) = 0$ for some $i \in S^1$ and solving $\underline{J}^T \underline{J}\, dx_1 = -\underline{J}^T \underline{R}(\mathbf{x})$. Then $d/d\lambda \|\underline{R}(x + \lambda\, dx)\|_2^2 = 2(\underline{J}(x + \lambda\, dx)\, dx_1,\ \underline{R}(x + \lambda\, dx))$. For $\lambda = 0$ this gives

$$d/d\lambda \|\underline{R}(x + \lambda\, dx)\|_{2\lambda=0}^2 = -2\left([\underline{J}^T \underline{J}]^{-1} J^T R_p,\ J^T R_p\right) < 0$$

from substitution of dx_1. However, $\underline{J}$ is a badly conditioned matrix already so that it is not advisable to use $\underline{J}^T \underline{J}$.

VIII. Practical remarks. Some remarks can be made on the implementation and actual usage of the algorithm.

Remark 1. Step strategies for continuation methods are as yet all unsatisfactory in practice. Theoretical information can be found in [16]. However, for this particular problem we simply apply multiplication by 1.3 after two consecutive successful steps and we divide by 2 in case of rejected step.

Remark 2. The damping procedure shows sometimes λ's converging to zero, sometimes diverging λ's. In the first case we reject the step, in the second case we take $\bar{\lambda} = 1$.

Remark 3. The computational effort involved in calculating $\bar{\lambda}$ appeared to be approximately half of the time needed to solve the linear system.

IX. CURRY. The examples were calculated with the program package CURRY. This is a package similar to MAGGY [13] and SEMMY [6]. The problem oriented language of SEMMY has been extended for the specification of mobilities, recombination, and contact potentials.

X. Examples. In this section two examples of totally different devices are discussed to show flexibility of the algorithm and the program. In Figs. 1 and 2, cross sections of both devices are depicted. The meshes used for the calculations are also shown in the same figures.

The first example is a bipolar transistor for low-voltage analog applications. For the calculations the mobility is modeled dependent on electric field, dopant, and temperature. As recombination mechanisms both SHR and Auger have been taken into account.

In Figs. 3 and 4 the full $\log I_C/V_{BE}$ and $\log I_B/V_{BE}$ curves are given. The resulting current gain is shown in an $h_{FE}/\log I_C$ plot in Fig. 5.

The second example is an MOS transistor with a gate length of 15 μm. Both electron and hole currents have been taken into account. In Fig. 6 the $\log I_d/V_s$ characteristic is shown at gate voltage $V_{gs} = 3$ V.

XI. Algorithm behavior. The behavior of the algorithms for the two examples is shown in Tables I–IV. The tables show the number of unknowns in each Newton step (vertical) for each t-level in the continuation process (horizontal). So a column represents a t-step. Table I shows the behavior of D3S for the bipolar transistor problem. Table II shows the behavior of U+D3S for the same problem. Table III

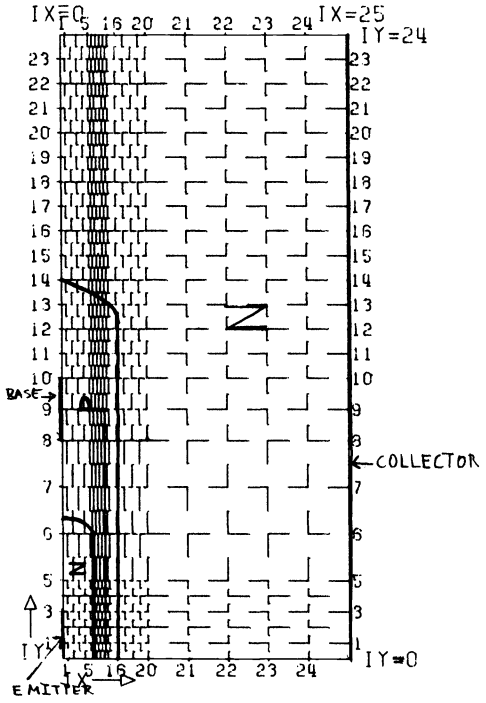


FIG. 1. *Mesh for bipolar transistor.*

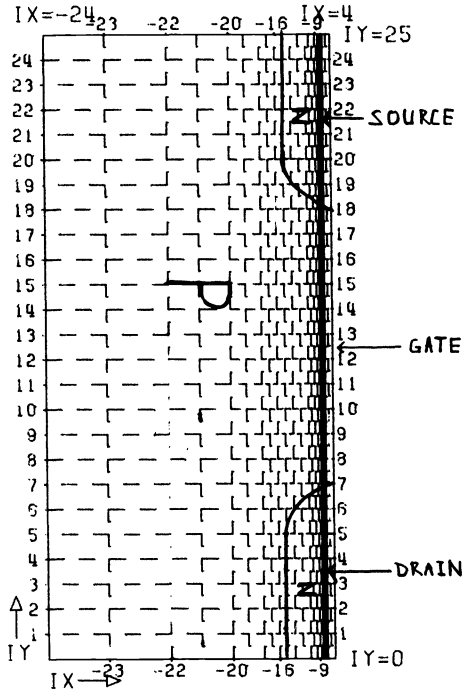


FIG. 2. *Mesh for* MOS *transistor.*

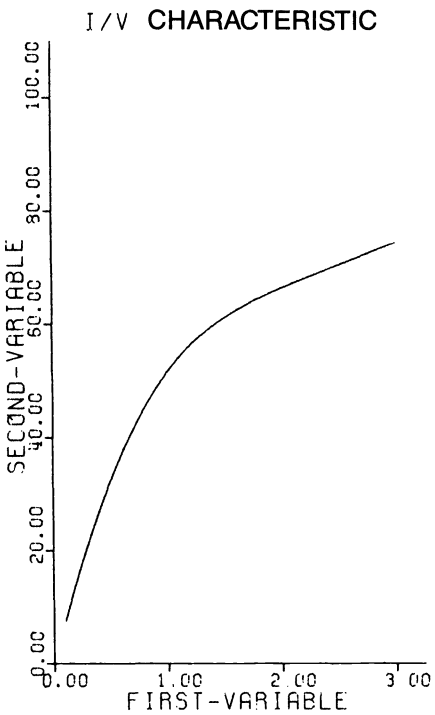


FIG. 3. $\log I_C/V_{BE}$.

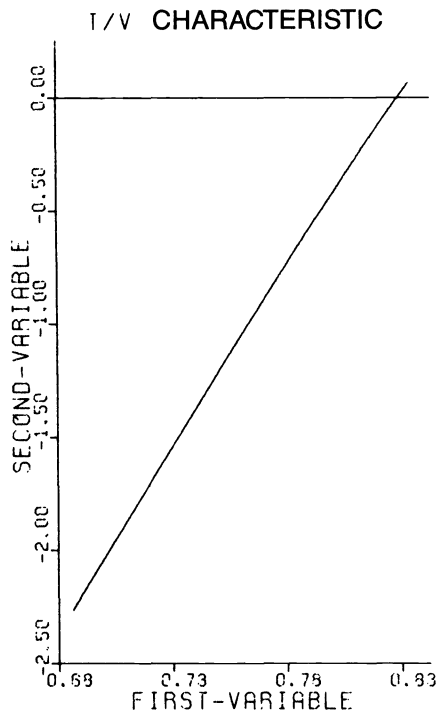


FIG. 4. $\log I_B/V_{BE}$.

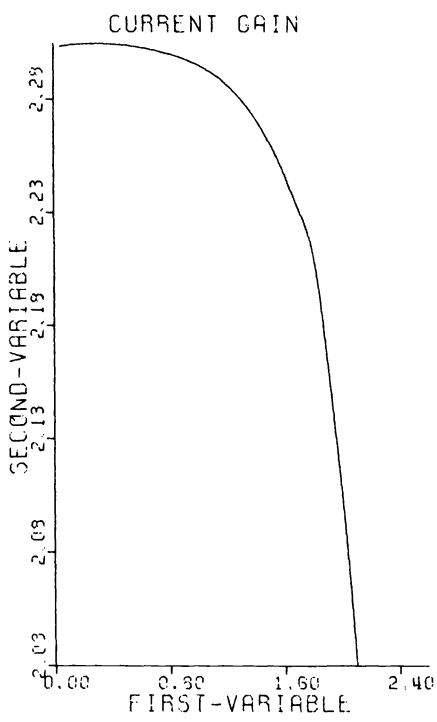


FIG. 5. $h_{FE}/\log I_C$.

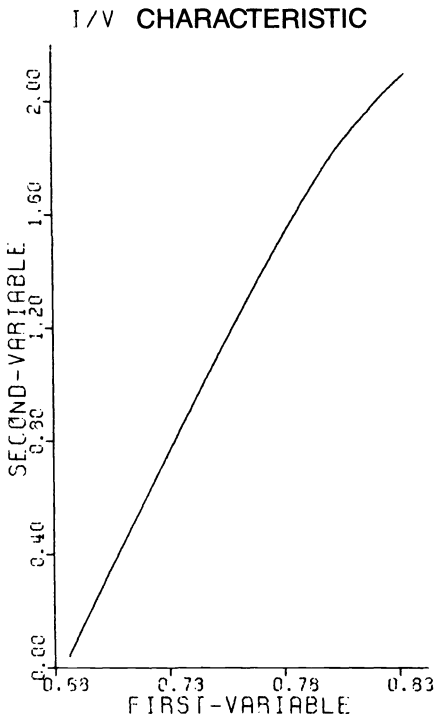


FIG. 6. $\log I_D/V_{DS}$.

shows the behavior of D3S for the MOS transistor problem and Table IV shows part of the U + D3S for the same problem. The performance of U + D3S may be improved by calculating a better starting point for the process to solve the problem. As a referee pointed out, by routinely initializing the majority Fermi potential to the applied bias in the region corresponding to the contact, while choosing the electrostatic potential to maintain the previous space-charge density.

TABLE I
D3S *for bipolar transistor*

BASE-EMITTER VOLTAGE							
0.09	0.14	0.18	0.24	0.31	0.39	0.44	0.49
MAXIMUM NUMBER OF UNKNOWNS IN NEWTON PROCESS							
1950	1950	1950	1950	1950	1950	1950	1950
NUMBER OF UNKNOWNS PER NEWTON STEP							
1950	1950	1950	1950	1950	1950	1950	1950
273	227	229	286	297	319	337	364
265	221	216	277	296	313	318	351
258	220	215	277	296	299	317	351
252	206	210	275	296	299	315	351
243	60	49	264	294	298	170	210
238	1950	1950	118	294	298	1	2
229	31	124	1950	293	296	1950	1950
199	7	20	56	268	296	94	117
16	1950	1950	1950	44	290	1950	1950
1950				1950	136		
78				15	1950		
3				1950	56		
1950					1950		

TABLE II
U + D3S *for bipolar transistor*

BASE-EMITTER VOLTAGE												
0.09	0.18	0.21	0.24	0.28	0.32	0.36	0.39	0.43	0.49	0.52	0.56	0.60
MAXIMUM NUMBER OF UNKNOWNS IN NEWTON PROCESS												
220	265	196	202	232	271	215	217	246	314	232	234	308
NUMBER OF UNKNOWNS PER NEWTON STEP												
24	24	24	24	24	24	24	24	24	24	24	24	24
45	45	45	45	45	45	45	45	45	45	45	45	45
68	68	62	62	68	62	62	62	68	68	62	62	68
94	94	87	87	87	87	87	87	93	93	87	87	93
116	116	113	113	113	113	113	113	113	119	113	113	119
135	138	138	141	141	141	141	141	141	147	141	141	147
157	163	160	166	166	167	168	167	167	174	168	167	173
171	181	177	185	186	195	186	185	195	192	186	186	191
184	198	196	202	204	204	203	204	203	210	203	204	204
201	210	156	165	221	223	215	217	221	228	221	221	223
214	222	133	139	232	235	203	208	235	241	232	234	235
218	230	133	146	232	247	187	202	246	253	229	230	247
220	241	129	146	225	260	179	178	244	267	223	223	259
99	249	117	138	213	271	177	178	242	273	209	210	270
63	257	97	120	190	255	155	178	235	285	197	197	272
72	262	97	130	148	214	159	174	214	296	203	203	271
71	262	102	134	149	210	162	181	215	310	217	225	289
41	265	101	106	146	206	156	179	215	314	217	232	308
1	242	21	43	57	175	150	172	207	313	138	142	207
	61	21	5		20	19	25	148	164	59	65	139
		20	1		2		16	26	36	39	50	125
		18			5			18	36	40	53	143
		18			8			2	42	45	55	149
		11			11			2	54	61	70	100
		4						2	12	24	30	88
		3						2	1	8	15	77
		4						2		1	1	33
		5						2				31
		5										15
												9
												4
												2
												1

TABLE III
D3S *for* MOS *transistor*

DRAIN-SOURCE VOLTAGE						
0.05	0.09	0.18	0.27	0.42	0.54	0.69
MAXIMUM NUMBER OF UNKNOWNS IN NEWTON PROCESS						
2054	2054	2054	2054	2054	2054	2054
NUMBER OF UNKNOWNS IN NEWTON PROCESS						
2054	2054	2054	2054	2054	2054	2054
1023	1085	1142	1193	1235	1257	1275
927	973	1081	1139	1178	1235	1252
845	854	955	1066	1141	1228	1245
109	551	771	873	1113	1210	1234
2054	2054	2054	389	1030	1199	1222
	80	112	2054	745	1179	1211
	2054	2054	32	67	1162	1194
			2054	2054	1125	1172
				228	956	886
				2054	843	449
					36	48
					6	2054
					2054	294
					257	1
					2054	2054

TABLE IV
U + D3S *for* MOS *transistor*

DRAIN-SOURCE VOLTAGE									
0.05	0.09	0.18	0.27	0.42	0.57	0.78	1.02	1.14	1.29
MAXIMUM NUMBER OF UNKNOWNS IN NEWTON-PROCESS									
380	390	507	652	634	837	779	756	736	711
NUMBER OF UNKNOWNS PER NEWTON STEP									
14	14	14	14	20	25	16	14	15	16
39	39	39	39	46	58	43	39	40	42
63	63	64	64	76	89	74	72	76	76
98	97	99	101	116	120	114	111	115	119
127	129	126	137	154	162	164	163	172	176
144	148	144	163	194	195	211	217	215	223
138	141	165	184	215	229	244	252	246	254
157	161	189	208	248	266	270	281	279	279
180	180	214	231	271	302	301	309	304	310
205	202	231	254	302	340	329	336	339	342
220	217	251	275	332	379	356	370	370	375
240	240	270	307	358	418	390	409	405	408
274	270	298	357	398	476	429	449	442	443
310	302	329	400	432	523	468	497	482	480
333	333	372	453	471	563	509	521	516	521
359	359	411	489	501	593	538	559	549	551
374	386	450	549	547	628	579	597	595	594
380	390	482	593	569	660	607	631	623	635
365	386	507	602	596	690	639	668	660	667
158	180	490	634	621	719	669	701	687	689
172	199	469	652	634	748	694	723	724	708
143	162	400	639	634	765	708	746	736	711
113	101	137	621	593	791	724	756	695	709
128	95	73	284	258	809	730	750	342	459
130	95	61	133	95	811	733	559	92	227
126	97	51	126	46	814	748	235	39	201
121	103	42	124	17	825	759	202	32	148
116	105	37	125	15	835	779	151	30	22
122	96	33	115	11	837	743	119	25	18
122	89	26	103	7	455	604	27	17	12
115	82	24	51	5	272	597	25	11	6
106	59	13	53	3	165	597	11	9	2
130	30	10	28	3	147	593	10	6	
140	22	7	8	3	155	589	9	5	
127	15	6	8		146	597	7	4	
55	18	4	6		188	604	3	2	
36	8	3	5		219	604	2	1	
16	3	2	5		242	604		3	
1	2	1	3		223	607		2	
	1		1		192	610			
					181	601			
					145	538			
					119	150			
					65	89			
					49	38			
					40	38			
					37	40			
					35	45			
					33	45			
					30	41			
					28	31			
					29	26			
					27	10			
					17	1			
					15				
					14				
					8				
					8				
					3				
					3				

REFERENCES

[1] F. L. BAUER, *Numer. Math.*, vol. 5, pp. 73–87, 1963.
[2] S. C. EISENSTAT, *SIAM J. Sci. Stat. Comp.*, vol. 2, pp. 1–4, 1981.
[3] O. AXELSSON, *Linear Alg. and Appl.*, vol. 29, pp. 1–16, 1980.
[4] C. DEN HEYER, "The numerical solution of nonlinear operator equations by imbedding methods," Ph.D. dissertation, Math. Centrum, Amsterdam, 1979.

[5] S. J. POLAK *et al.*, in *Proc. NASECODE II Conf*, B. T. Browne and J. J. H. Miller, Eds., Dublin, Ireland, pp. 182–187, 1981.
[6] S. J. POLAK *et al.*, *Int. J. Numer. Math. Engineering*, vol. 17, pp. 1591–1604, 1981.
[7] W. C. RHEINBOLDT, in *Mathematics of Finite Elements and Applications*, J. Whiteman, Ed. New York: Academic Press, 1976, pp. 465–482.
[8] J. W. SLOTBOOM, "Analysis of bipolar transistors," Ph.D. dissertation, Technical University Eindhoven, 1977.
[9] A.V.D. SLUIS, *Numer. Math.*, vol. 14, pp. 14–23, 1969.
[10] S. M. SZE, *Physics of Semiconductor Problems*. New York: Wiley, 1969.
[11] E. WASSERSTROM, *SIAM Rev.*, vol. 15, pp. 89–119, 1973.
[12] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*. New York: Clarendon Press, 1965.
[13] S. J. POLAK *et al.*, *Int. J. Numer. Math. Engineering*, vol. 15, pp. 113–127, 1980.
[14] P. K. W. VINSOME, in *Proc. 4th SPE Symp. on Resevoir Simulation* (Los Angeles, CA), pp. 149–160, 1976.
[15] J. KOWALIK AND M. R. OSBORNE, *Methods for Constrained Optimization Problems*. New York: Elsevier, 1968.
[16] C. DEN HEYER AND W. C. RHEINBOLDT, *SIAM J. Numer. Anal.*, vol. 18, pp. 925–948, 1981.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

TWO-DIMENSIONAL ANALYSIS OF SEMICONDUCTOR DEVICES USING GENERAL-PURPOSE INTERACTIVE PDE SOFTWARE *

JAMES L. BLUE† AND CHARLES L. WILSON‡

Abstract. Analyzing currents and fields in VLSI devices requires solving three coupled nonlinear elliptic partial differential equations in two dimensions. Historically, these equations have been solved using a special-purpose program and batch runs on a large fast computer. We use a general-purpose program and interactive runs on a large minicomputer. We discuss the physical formulation of the semiconductor equations and give three example solutions: a short-channel MOSFET near punchthrough, a DMOS power transistor in the ON state, and a beveled p–n junction. These examples demonstrate that solutions to a very general class of semiconductor-device problems can be obtained using these methods.

Nomenclature

E_a	Energy of acceptor-like trap.	Q_{ss}	Total interface charge.
E_d	Energy of donor-like trap.	q	Electronic charge.
E_g	Bandgap.	R	Recombination-generation rate.
g_a	Acceptor-state spin degeneracy.	s_n	Electron surface recombination velocity.
g_d	Donor-state spin degeneracy.	s_p	Hole surface recombination velocity.
N	Net ionized impurity density.	T	Kelvin temperature.
N_c	Density of states in the conduction band.	V	Applied voltage.
N_v	Density of states in the valence band.	ε_0	Permittivity of free space.
n	Electron density.	κ	Dielectric constant.
n_0	Equilibrium electron density.	μ_n	Electron mobility.
n_1	Electron density at trap energy.	μ_p	Hole mobility.
n_i	Intrinsic carrier concentration.	τ_{n0}	Electron lifetime.
$\mathbf{n}$	Unit vector normal to boundary.	τ_{p0}	Hole lifetime.
p	Hole density.	ϕ_{MS}	Metal-semiconductor work function.
p_0	Equilibrium hole density.	ϕ_n	Electron quasi-Fermi level.
p_1	Hole density at trap energy.	ϕ_p	Hole quasi-Fermi level.
Q_{itc}	Interface trapped charge.	ϕ_S	Surface potential.
Q_{ox}	Oxide trapped charge.	ψ	Electrostatic potential.

I. Introduction. Analyzing currents and fields in VLSI devices requires solving three coupled nonlinear elliptic partial differential equations in two dimensions. Historically, these equations have been solved using a special-purpose program and batch runs on a large fast computer. We use a general-purpose program and interactive runs on a large minicomputer.

We have developed a package which is capable of solving a wide variety of semiconductor-device problems. The package analyzes coupled systems of semiconductor equations in two dimensions. The package is hierarchical, consisting of a top level, or driver, which calls a general-purpose driver (B2DE), which calls a general-purpose elliptic solver. The top level depends on the particular semiconductor-device technology being modeled. This organization has several advantages. We can change the physical model of the system easily. We can test the package on problems with known solutions.

*Received by the editors November 5, 1982, and in revised form March 28, 1983.

†Scientific Computing Division, Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234.

‡Semiconductor Devices and Circuits Division, Center for Electronics and Electrical Engineering, National Bureau of Standards, Washington, DC 20234.

We discuss the physical formulation of the semiconductor equations and give three example solutions: a short-channel MOSFET near punchthrough, a DMOS power transistor in the ON state, and a beveled p–n junction. The MOSFET is a typical present-day short-channel VLSI device. The DMOS example shows how power devices and other structures with large ratios of parasitic to active volume, such as gate structures, can be analyzed. The p–n junction problem shows how nonplanar devices with nonideal contacts can be analyzed. These examples demonstrate that solutions to a very general class of semiconductor device problems can be obtained using these methods.

Our use of B2DE has convinced us that we are now able to analyze large enough VLSI structures to do gate-level modeling.

A. Why use a general-purpose box? Our working environment has affected our choices of methods for analyzing semiconductor devices (SCD's). The National Bureau of Standards does not design, manufacture, or sell SCD's. We do not need the world's fastest computer program for analyzing short-channel MOSFET's or any other kind of SCD.

We do need flexibility in analyzing SCD's. We need to be able to analyze many kinds of devices, with many different geometries, in many modes of operation, including nonstandard modes. We need to be able to change the physical model included in the analysis.

We want to have confidence that our partial differential equations (PDE's) are solved sufficiently accurately. With a general-purpose box (GPB), we can solve PDE's with known solutions, not just problems arising from SCD's. Since the same GPB does all the kinds of devices we are interested in, we have comparatively little new computer code to write and to test when we start to analyze a new kind of device.

B. Where do you get a suitable GPB? We are firm believers in building on other people's work. We have saved much development time by using existing quality software instead of starting at the beginning. If there had been a suitable GPB in the public domain, we would not have built our own. We started in 1979 with the first version of PLTMG, written by R. E. Bank and A. H. Sherman [2]. PLTMG had many of the features we thought were essential, but lacked others. We have rewritten it in another language, modified nearly every line of it, and added many features to it. Since 1979, there have been new versions of PLTMG [3]. We have not incorporated Bank's recent modifications, but have made some of them independently.

II. Design principles for a GPB. In this section we discuss principles that guided us in the development of B2DE. Throughout, we have striven to keep B2DE as general as possible while giving it the capability to analyze SCD's.

A. How wide a class of PDE's should the GPB solve? The simplest realistic formulation of the semiconductor-device equations includes a two-dimensional calculation of the electrostatic potential and the hole and electron quasi-Fermi levels in the semiconductor. The basic semiconductor equations are given in Section III. To analyze the steady-state behavior of SCD's, the GPB has to solve three coupled nonlinear elliptic PDE's. B2DE can solve k PDES in k unknown functions u_i. The PDE's are of the form

$$(1) \qquad \nabla\cdot\left(a_i(u, \partial u/\partial x, \partial u/\partial y, x, y)\nabla u_i\right) = f_i(u, \partial u/\partial x, \partial u/\partial y, x, y).$$

a_i and f_i can be any function of their variables. To maintain generality, the user of B2DE provides a_i and f_i as subprograms; each of the programs must calculate the function and its first partial derivatives.

At each boundary point, there must be k boundary conditions. B2DE allows two types of nonlinear boundary condition

$$g_i(u, x, y) = 0 \tag{2}$$

or

$$\partial u_i / \partial \mathbf{n} + g_i(u, x, y) = 0. \tag{3}$$

The type is allowed to be different for different PDE's and is allowed to vary along the boundary.

This framework is general enough for the standard SCD equations and the standard physics. More generality is needed for more complex physical models. The framework is so general that it includes many problems that are mathematically or physically ill-posed. We do not expect B2DE to work well for such problems.

The geometry of the region involved is also important. A single rectangle is not sufficient, nor is a union of rectangles. A union of triangles is sufficient. Curved boundaries are infrequently required, but are included in B2DE because they were in PLTMG.

B. What kind of accuracy is desired? How does this affect the GPB? If a very accurate solution of a smooth problem is wanted, higher-order methods are probably indicated. If less accuracy is wanted, or if the typical problem is not smooth, lower-order methods are indicated. Our solutions are not smoothly varying; they have much of their variation in a small part of the region, as discussed later.

In analyzing SCD's, the physical parameters are generally not known accurately. In addition, some regions of a device may not have much effect on the behavior of the device, and thus require less accuracy. Similarly, some variables do not need to be known accurately to predict the behavior of the device.

For our GPB, occasionally we will want high accuracy, and want methods that will not prevent our obtaining a very accurate solution, but "engineering accuracy," a few percent, will usually be enough. For testing the GPB, we will need higher accuracy.

We frequently simulate VLSI devices for the purpose of designing and explaining experiments. We frequently simulate these devices in modes of operation that "production"-oriented device modeling does not usually require. In this type of modeling, we do not know a good solution strategy, and must compute interactively to develop one.

B2DE allows the user to monitor the solution variables and residuals to aid in finding a solution strategy. The adaptive methods in B2DE, to be described later, allow the user to control the evolution of the mesh to achieve the accuracy required for the particular problem.

C. What are the special features of SCD problems? SCD problems are qualitatively different from the well-known PDE's of mathematical physics, and present different challenges to the numerical analyst and the software designer.

The standard form of the SCD equations includes exponential nonlinearities, and the arguments of the exponentials can be large. The software designer must protect against overflow and against destructive underflow. In some cases, accuracy can be limited by cancellation of large terms, as in (4), where with the usual normalization N can be as large as 10^{10}.

In order to solve the PDE's, they first must be reduced to a set of nonlinear equations. (We use linear finite elements on triangles; the nonlinear equations are the usual Galerkin equations.) In solving these nonlinear equations, by some variant of Newton's method, the approximate solution is not close to the answer until the end of the solution process; standard numerical analysis proofs are mostly irrelevant.

The three PDE's have typical terms that vary greatly in magnitude, both from each other and in different parts of the device. Each PDE must be scaled by the user, and the scaling should be different for each equation.

In some regions of devices, either the electron or the hole density is negligible. This causes the Jacobian matrix of the nonlinear equations to be nearly singular, and a calculated Newton correction may be inaccurate, especially on a computer with short word length. We have often found it necessary to regularize the Jacobian matrix to avoid singularity. We do not have an automated method to recognize near-singularity.

Present-day devices have solutions characterized by steep fronts, as in Section III. The locations of the fronts depend on the particular device because of the doping profile, and on the boundary conditions, the applied voltages. A uniform mesh throughout the device is inadequate; even a tensor-product mesh is wasteful. The mesh must be refined locally to resolve the steep fronts as well as to cope with mathematical singularities in the solution. For these reasons, high-order methods are usually inappropriate.

The whole character of the solution can change drastically with boundary conditions. Providing a reasonable initial guess for the general SCD is difficult; providing one for a particular class of device in a restricted range of operation may not be difficult.

There are always three coupled equations, but the degree of coupling and the importance of each equation vary with device type, with region in a given device, and with the applied boundary condition. The variation makes it difficult to specify a universal iterative method for solving the coupled nonlinear PDE's. In the general case, we cannot count on knowing an iteration algorithm which will reliably solve the nonlinear equations without human intervention.

In the general case, we cannot count on knowing a correct mesh or even an adequate one. If the mesh has too little resolution in critical regions, there may not be a solution to the set of nonlinear equations.

D. What software features do you want in the GPB? The box should be in portable Fortran, not optimized for any particular computer. B2DE is in the PFORT subset of Fortran 66; adherence to this subset has been checked by the PFORT verifier program [13]. Machine-dependent parameters, such as the largest and smallest floating-point numbers, are isolated. System-dependent features, such as the lowest-level graphics calls, are also isolated.

Although the GPB should be in portable Fortran, we prefer to write in a higher-level language, both because Fortran is an unpleasant language and because we are more productive programmers in a higher-level language. We use Ratfor [12], a language with modern control structures and a few conveniences; a translator converts the Ratfor to Fortran.

It should be possible to run the GPB interactively for program development and to run it in batch mode for production runs.

In order to help understand both the solution process and the nature of the solution, graphical output is essential. Interactive graphics on a video terminal is necessary; diversion of the graphical output to files for postprocessors is also necessary.

Many types of graphics are useful.

The user of the GPB must be able to save and to restart solutions. The user can test various solution strategies and compare the results. Since the box can never be completely general, it should have built-in places where a user can insert a subroutine to replace a null subroutine for special purposes.

The program must be tolerant. No calculation should ever overflow. No destructive underflow should occur. The program must never abort because of faulty inputs from the user.

E. What is B2DE like? B2DE has a hierarchical structure. On the bottom are a number of programs from well-established software libraries, including sparse matrix software [6], [16] and the kernel of the PORT library [9]. On the middle layer is a large assemblage of general-purpose finite-element programs and graphics programs. Much of this layer is descended from PLTMG.

The top layer, the layer with which a user interacts, may be one of several existing top layers. One of them is a general-purpose interactive driver. It is used for "one-shot" problems and for developing specialized drivers. The user provides a few subroutines, as part of the top layer, to define the PDE's and the boundary conditions.

Other top layers are specialized drivers for particular classes of SCD's. These drivers simplify the inputs to B2DE and provide the necessary subroutines.

The general-purpose interactive driver is typical of all the top layers in having a menu-driven structure. At each step, the user is presented with a menu of all the actions that are possible at this time. The possible menus have a tree structure. For example, at the highest level of the tree, the user has the following options:

print statistics about the current solution,
plot the current solution,
iterate on solving the nonlinear finite-element equations,
refine a mesh, producing a new mesh,
compress the finest mesh into the next-coarser mesh,
compare the current solution with a saved solution,
save the current solution,
change parameters, and
quit.

If, for example, the plot option is selected, the user has the choice of plotting the following:

triangles,
contours,
surface plots,
profile plots, and
flow-line plots.

Then, if surface plots are selected, the user may select

surface views of the triangles,
surface views of level lines (contours), or
surface views of grids on the surface.

Each of these plots then requires further information from the user.

B2DE uses linear finite elements on triangles, and in standard fashion [17] converts the PDE's to a set of nonlinear equations. The nonlinear equations are solved by a

damped Newton's method; the Jacobian matrix of the nonlinear equations is similar to a finite-element matrix from a set of linear PDE's. The heart of B2DE is software to solve the linear finite-element equations and is the part of B2DE which has changed least from PLTMG.

In solving the nonlinear finite-element equations, Newton iterations may be carried out until a given degree of convergence is achieved. The Newton iterations require solving linear equations. These may be solved directly, using sparse Gaussian elimination [16], by standard iterative methods of several possible types, or by multilevel iterative methods of several possible types. Depending on the progress of the solution, the user can decide to do more iterations, to change the type of iterations, to change from working on all three equations at once to working on two or on one, or refine the mesh and then iterate. At every stage, the user can (and must) make choices about the solution process.

Since the user does not know a reasonable mesh, B2DE generates a reasonable mesh adaptively. Starting with the user-specified mesh, an approximate solution to the nonlinear finite-element equations is found. The error in the solution on this mesh is estimated, and a refined mesh is generated by subdividing the triangles with the largest estimated error. The degree of refinement is governed by the refinement threshhold; all triangles with at least this fraction of the worst estimated error are subdivided. Alternatively, this fraction of triangles may be refined. This process can be repeated. For SCD's, the final mesh is highly nonuniform.

Asymptotically, in the limit as all triangle dimensions become small, linear finite elements on triangles gives second-order convergence to the solution. That is, if each triangle is divided into four congruent triangles, the error in the solution is divided by four. In practice, for SCD's, the solution is almost always as accurate as desired before asymptotic behavior holds everywhere in the device.

Both in developing iteration strategies and in interpreting the physical meaning of solutions, the availability of graphics is essential. Otherwise it is difficult for the user to make any sense out of what is happening. If the graphical output must be specified before the computer run is started, the making-sense process is drastically slowed.

At every step, the user is also able to find out what is going on in the solution process by looking at the current approximate solution, the Newton iterates, and the residuals; by looking at many kinds of graphics; and by looking at timing results. There are many choices to be made. B2DE is not a black box, safe for use by amateurs, but a box to be run by an expert. For a limited class of problems, an expert can construct a new top-level driver suitable for use by amateurs.

F. What kind of computer is suitable? The majority of the computer time is spent in constructing Jacobian matrices for solving the nonlinear finite-element equations. Much of this work is not floating-point calculations, but, broadly speaking, bookkeeping. It is not easy to arrange much of the code to take advantage of the special capabilities of vector or parallel machines.

The data structures needed to implement arbitrary meshes of triangles are fairly wasteful of memory. For k PDE's and m vertices, approximately $m(13+6k+19k^2)$ words are necessary. Typical total-memory requirements for an accurate solution of three coupled equations ranges from 300 000 to 1 800 000 floating-point words. Keeping many levels of triangulation and using multilevel iteration requires less memory than using only the finest level and using direct solution of the linear equations. (More data space is used, but less space is needed for factoring the level 1 matrix.) Data references are sufficiently local that the program runs well under a

properly configured virtual-memory operating system.

Interactive operation of the GPB, including the graphics, is needed to take full advantage of the box's capabilities.

In summary, B2DE will run well on a computer with a modern operating system, either with large memory or with efficient virtual memory. It will not run well on old-fashioned operating systems, even those supposedly designed for scientific use.

III. Physical formulation of SCD problems. The physical formulation of the device simulations discussed here requires solution of the basic semiconductor-device equations using the appropriate carrier distribution functions, Boltzmann or Fermi–Dirac, with appropriate boundary conditions. In addition, the interior structure of each device requires the specification of a two-dimensional doping profile.

The model includes a two-dimensional calculation of the electrostatic potential in the semiconductor and in the oxide (if one is present) and a two-dimensional calculation of the hole and electron quasi-Fermi levels in the semiconductor. With an oxide present, these equations are more general than B2DE can solve directly, since B2DE assumes that all PDE's are defined on the same region. We use a fast Poisson solver [18] in planar oxides.

A. The semiconductor equations. The model is based on the standard semiconductor equations

$$\nabla\cdot(\kappa\nabla\psi) = -\frac{q}{\varepsilon_0}(p - n + N), \tag{4}$$

$$\nabla\cdot(\mu_n n\nabla\phi_n) = R, \tag{5}$$

$$\nabla\cdot(\mu_p p\nabla\phi_p) = -R. \tag{6}$$

The symbols have their usual meanings, as defined in the nomenclature list, and are derived from the basic equations given in standard texts such as [19]. For Boltzmann statistics, the electron density n and the hole density p are

$$n = n_i \exp\big(q(\psi - \phi_n)/kT\big), \tag{7}$$

$$p = n_i \exp\big(q(\phi_p - \psi)/kT\big). \tag{8}$$

For Fermi–Dirac statistics, n and p are

$$n = N_c \mathscr{F}_{1/2}\left(q(\psi - \phi_n - E_g/2)/kT - \frac{1}{2}\ln\left(\frac{N_c}{N_v}\right)\right), \tag{9}$$

$$p = N_v \mathscr{F}_{1/2}\left(q(\phi_p - \psi - E_g/2)/kT + \frac{1}{2}\ln\left(\frac{N_c}{N_v}\right)\right) \tag{10}$$

where

$$\mathscr{F}_j(\eta) = \frac{1}{\Gamma(j+1)}\int_0^\infty \frac{\varepsilon^j\, d\varepsilon}{1+\exp(\varepsilon - \eta)}. \tag{11}$$

In (5) and (6), R stands for a general recombination-generation term; in the examples in this paper, we use Shockley–Read–Hall recombination

$$R = \frac{(pn - n_i^2)}{\tau_{n0}(n + n_1) + \tau_{p0}(p + p_1)}. \tag{12}$$

Auger recombination could also be used; avalanche generation could also be included. Both are easy to insert into B2DE.

Doping and mobilities are specified by the user as subroutines.

The validity of (4) is limited by the validity of characterizing the polarizability by a simple dielectric constant. This is not a significant limitation in most applications. Using a tensor instead of a constant would require a fairly simple change in B2DE.

The restriction imposed by the formulation of (5) and (6) is more basic. Each of these equations depends on the existence of an isotropic mobility and on the correctness of a distribution function relating carrier densities to electrostatic and quasi-Fermi potentials. In most applications, a mobility can be defined which meets the above requirements [20]. Anisotropic mobilities could be used with the altered form of B2DE mentioned in the previous paragraph.

Tensor polarizability and anisotropic mobilities are necessary for superlattice devices.

In each of the triangular elements the carrier densities used in (5) and (6) are approximated by one of the distribution functions. The factor multiplying the gradient is an exponential function of the solution variables. In the finite-element solution in B2DE, integrals over triangles are approximated by a 4-point quadrature rule, using the triangle vertices and the midpoint. The finite-element solution maintains the continuity of the currents only approximately, since the integrals are not done exactly. In high-current regions of the device, the mesh needs to be refined to improve current continuity. The accuracy of the current will be discussed in a later section.

The recombination term given in (11) is the usual Shockley–Read–Hall term. For the MOS transistors considered here, where recombination takes place in the lightly doped substrate, this term is usually sufficient. The accuracy of this model is dependent on the models used for τ_{n0} and τ_{p0} as functions of position and doping.

The validity of the distribution functions is more suspect. Boltzmann statistics are only an approximation to Fermi–Dirac statistics. When the latter are used, the distribution functions for holes and electrons given in (9) and (10) assumes parabolic densities of states in the valence and conduction bands. This may not always be sufficient.

When a solution has been found, the difference between Boltzmann and Fermi–Dirac statistics is significant only when the carrier densities become large, approaching the densities of states. Including Fermi–Dirac statistics improves the stability of the solution process by bounding the exponential terms found in Boltzmann statistics.

This simplified model neglects carrier interactions and other effects which would effect carrier statistics in a strongly inverted device. In addition all heavy-doping effects which would affect the density of states in the band have been excluded.

B2DE is adequate to handle most of this more complicated physics. It is harder to decide what physics to use than to fit it into the model.

B. Boundary conditions. Most previous models of semiconductor devices have used idealized metallic contacts and idealized oxide–semiconductor interfaces. In the models developed here more realistic boundary conditions are used, including both ideal ohmic contacts and contacts with Schottky barriers. For Boltzmann statistics, contacts with an applied potential V are characterized by

$$\psi = V + \ln(N/n_i) + \phi_{MS}, \tag{13}$$

$$\phi_n = V + \phi_{MS}, \tag{14}$$

$$\phi_p = V + \phi_{MS}. \tag{15}$$

ϕ_{MS} is zero for a perfect ohmic material. These functions cause the right-hand side of (4) to be zero at the metal–semiconductor interface.

In the case of Fermi statistics on n-type material

$$\psi = V + \mathscr{F}_{1/2}^{-1}(N/N_c) + E_g/2 + \frac{1}{2}\ln\left(\frac{N_c}{N_v}\right) + \phi_{MS}, \tag{16}$$

$$\phi_n = V + \phi_{MS}, \tag{17}$$

$$\phi_p = V + \phi_{MS}. \tag{18}$$

On p-type material (16) becomes

$$\psi = V + \mathscr{F}_{1/2}^{-1}(-N/N_v) - E_g/2 + \frac{1}{2}\ln\left(\frac{N_c}{N_v}\right) + \phi_{MS}. \tag{19}$$

These functions also cause the right-hand side of (4) to be zero at the metal–semiconductor interface.

Any of these sets of boundary conditions is also equivalent to the case of an infinite surface recombination velocity in that the value of the excess carrier density goes to zero. The model of the Schottky contact shown in (13)–(19) is a simple metal–semiconductor work function difference, ϕ_{MS}. Inclusion of a boundary of this type requires, or will cause, additional mesh refinement in the region of the Schottky contact.

The oxide boundary conditions used are

$$\frac{\partial \psi_{\mathrm{Si}}}{\partial \mathbf{n}} = -\frac{\kappa_{\mathrm{ox}}}{\kappa_{\mathrm{Si}}}\frac{\partial \psi_{\mathrm{ox}}}{\partial \mathbf{n}} - \frac{Q_{ss}}{\varepsilon_0 \kappa_{\mathrm{Si}}}, \tag{20}$$

$$\frac{\partial \phi_n}{\partial \mathbf{n}} = 0, \tag{21}$$

$$\frac{\partial \phi_p}{\partial \mathbf{n}} = 0. \tag{22}$$

The interface charge is given by

$$Q_{ss} = Q_{\mathrm{itc}} + Q_{\mathrm{ox}} \tag{23}$$

where

$$Q_{\mathrm{itc}}(\phi_S) = \int_{E_v}^{E_c} \frac{Q(E_d)\,dE_d}{1 + g_d \exp\left((\phi_S - E_d)/kT\right)} - \int_{E_v}^{E_c} \frac{Q(E_a)\,dE_a}{1 + g_a^{-1}\exp\left((E_a - \phi_S)/kT\right)}. \tag{24}$$

The evaluation of (24) requires a knowledge of the energy distribution of traps at the interface. This data must be obtained experimentally. The integration of (24) uses the value of the surface potential ϕ_S, which is the value of ψ evaluated at the interface of the semiconductor.

In the present model the value of the field in the oxide is calculated outside the Newton-iteration loop in B2DE. The calculation for planar oxides is done using a fast Poisson solver [18]. The change in potential perpendicular to the surface is large compared to the change parallel to the surface is MOS transistors; the oxide is "thin." This makes it necessary to have a relatively dense mesh in the oxide parallel to the interface, but allows a relatively coarse mesh perpendicular to the interface. In many applications it is possible to approximate the field in a "thin" oxide by dividing the potential difference across the oxide by the oxide thickness. This approximation is useful because the interface portion of the Jacobian matrix can be obtained exactly.

The case of a boundary which is controlled by surface recombination is given by

$$\frac{\partial \phi_p}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}} = \frac{s_p}{\mu_p}\left(1 - \frac{p_0}{p}\right), \tag{25}$$

$$\frac{\partial \psi}{\partial \mathbf{n}} - \frac{\partial \phi_n}{\partial \mathbf{n}} = \frac{s_n}{\mu_n}\left(1 - \frac{n_0}{n}\right) \tag{26}$$

where p and n are given by either (7) and (8) or by (9) and (10), and where p_0 and n_0 are given by these equations with $\psi = \psi_0$, the equilibrium value, and $\phi_n = \phi_p = 0$. The equilibrium value of ψ_0 is given by (13), (16), or (19), as appropriate. In the unbiased equilibrium case, since $\phi_p = \phi_n = 0$, (25) and (26) reduce to a boundary condition on the potential alone

$$\frac{\partial \psi}{\partial \mathbf{n}} = \frac{s_n}{\mu_n}\left(1 - \exp\left(q(\psi_0 - \psi)/kT\right)\right) = -\frac{s_p}{\mu_p}\left(1 - \exp\left(q(\psi - \psi_0)kT\right)\right) \tag{27}$$

for Boltzmann statistics. (The corresponding boundary condition for Fermi–Dirac statistics is left as an exercise for the reader.) This condition can be satisfied for arbitrarily large values of surface recombination velocity only if $\psi = \psi_0$ and $\partial\psi/\partial\mathbf{n} = 0$. As discussed previously, a surface with large surface recombination is in some sense equivalent to an ohmic metal contact. The boundary condition given in (25) and (26) with mixed nonlinear right-hand sides is not usually available in conventional PDE software. B2DE will need be generalized to provide this capability.

Two types of symmetry lines are used in the simulations presented here. These lines of symmetry represent boundaries at which $\partial\psi/\partial\mathbf{n} = \partial\phi_n/\partial\mathbf{n} = \partial\phi_p/\partial\mathbf{n} = 0$. No currents flow into or across these boundaries and the electric field has no component perpendicular to these boundaries. Actual lines of symmetry are required to model devices which have symmetric doping and symmetric biases. Typical examples of these structures are DMOS power transistors, bipolar transistors with symmetric base-emitter structure, and parasitics in NMOS and CMOS structures. These truly symmetric structures cause few calculational problems.

False symmetry lines are often introduced to bound the device being simulated. These false symmetry lines have boundary conditions identical to real symmetry lines, but are used as boundaries only to limit the computational region of a large device, saving mesh points and reducing calculation cost. The intersection of p–n junctions with these boundaries can cause accuracy and convergence problems. If the false lines of symmetry are too close to the active region of the device, accuracy problems are caused; field distortion propagates into the electrically active region of the device. This also reduces convergence rates. The mesh must be refined in such regions.

C. Doping profiles. The doping profiles used in these calculations enter into the equations only through $N(x, y)$, and possibly some parameters in R, such as τ_{n0} and τ_{p0}. The functions used are not attached to the mesh in any way. This allows the doping to be altered during the calculation.

The most commonly used two-dimensional doping profiles result from the redistribution of impurities by diffusion [11], [21], [7]. The simplest form of two-dimensional profile is the profile [11] combining a Gaussian profile perpendicular to the surface with an error function complement under the mask edge. This profile is a reasonable representation of the redistribution in ion-implanted impurities at low dose and high annealing temperatures. More abrupt two-dimensional profiles result at high dose and at lower annealing temperatures. These profiles have been studied in two dimensions

[21] using the concentration-dependent diffusion model of Fair [8]. Profiles of this type can be modeled by generating two-dimensional profiles of the constant-coefficient type with concentration-dependent diffusion lengths.

Low annealing temperatures generate impurity profiles which are not characteristic of diffusion processes. Either these profiles show little or no impurity redistribution [14], or they are the result of impurity redistribution processes in which the ion implantation damage has a significant effect [1]. Profiles of this type are well approximated by a Gaussian under the mask window and a Gaussian rotated about the median-range point under the mask. A Gaussian profile approximation results in an error of the type shown in Fig. 1 near the surface, but is generally an efficient method of approximating these profiles.

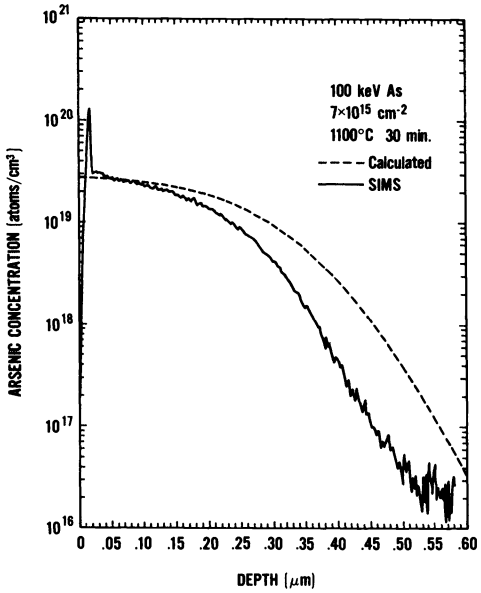


FIG. 1. *Comparison of modeled and measured doping profiles.*

IV. Interactive solution strategies. Each type of problem solved requires that a specific solution strategy be developed; finding an appropriate strategy is usually done with the interactive driver. Once the specific strategy has been determined, subsequent solutions can be obtained using batch calculations and later viewed interactively.

This procedure has allowed us to solve, to any desired level of accuracy, all of the problems we have attempted. We have solved them by starting from a rather simple initial guess with no need for "sneaking up" on the solution through gradually changing device parameters or biases.

The examples presented here include a short-channel MOSFET operated near punchthrough, a DMOS power transistor in the high current (ON) state, and an unbiased submicrometer p–n junction which is intersected by a beveled surface with high surface-recombination velocity. The most important input parameters of each of these devices are given in Tables I, II, and III, respectively. For all examples we have used Shockley–Read–Hall recombination, constant carrier mobilities, Fermi–Dirac statistics, $Q_{ss} = 0$, and $\phi_{MS} = 0$.

TABLE I
Simulation parameters—MOSFET

L_{poly}	2.222μm
L_{ch}	1.25μm
Q_{ss}	0.0
V_D, V_G, V_{SUB}	1.0V,1.0V,−0.100V
t_{ox}	0.0515μm
N_{ch}	1.0×10^{15} *to* 4.0×10^{15}cm^{-3}
N_{sd}	1.0×10^{20}cm^{-3}
L_{jun}	0.486μm
D_{jun}	0.687μm

TABLE II
Simulation parameters—DMOS *transistor*

L_{poly}	28.0μm
L_{ch}	0.40μm
Q_{ss}	0.0
V_D, V_G, V_{Body}	5.0V,5.0V,0.00V
t_{ox}	0.0515μm
N_{ch}	1.0×10^{17} *to* 5.0×10^{14}cm^{-3}
N_s	1.0×10^{20}cm^{-3}
N_d	5.0×10^{14}cm^{-3}
L_{Source}	1.6μm
D_{Source}	2.0μm
L_{Body}	3.2μm
D_{Body}	4.0μm

TABLE III
Simulation parameters—p–n *junction*

s_p/μ_p	1.0×10^5 V/cm
$N_{surface}$	1.0×10^{20}cm^{-3}
N_{bulk}	1.0×10^{15}cm^{-3}
D_{jun}	0.248μm
Bevel Angle	2.86°

The mesh used in these calculations is generated in two steps. First a primitive mesh of the type shown in Fig. 2 is generated. The initial mesh is generated by specifying boundary segments which represent the geometry of the device. The minimum number of boundary segments is determined by the shape of the region of interest and by the boundary conditions. At each transition between boundary condition (2) and (3), a vertex is required.

After the primitive mesh has been generated, user-specified prerefinements are used to place mesh points where the user thinks they are needed based on physical understanding of the device. In the case shown in Figs. 2 and 3, the prerefinements were used to increase the mesh density in the inversion layer. Additional mesh points were also placed at the gate edge in the source and drain, where there is a mathematical singularity in the potential.

We have found that when p–n junctions intersect Dirichlet edges, additional prerefinement is required. This is shown in the DMOS power transistor whose schematic is shown in Fig. 4(a). The shaded region is modeled; the level 1 triangulation is in Fig. 4(b). A view of the DMOS transistor potential solution is shown in Fig. 4(c). This device is almost as complex as a single gate, and suggests that gate-level modeling will be possible with B2DE.

Using a mesh of the type shown in Fig. 3, an initial attempt at the problem solution is attempted. The success of this attempt is partially determined by the adequacy of the mesh. If the initial level 1 mesh is not adequate, the nonlinear iterations may not converge, or may even diverge; additional refinement of the mesh then is required at level 1. If a small area of the device is causing most of the solution error, the mesh refinement will occur in a small percentage of the triangles, or even in a single triangle.

In such cases, compression back to level 1 is essential; the initial level 1 mesh is replaced by the new mesh. Such compressions are usually caused by effects near the edge of boundary regions or by the intersection of p–n junctions with surfaces. This type of mesh inefficiency slows convergence; the iterations only suffice to generate a mesh which is adequate. The residuals on the new mesh are still large, and discretization error is still large.

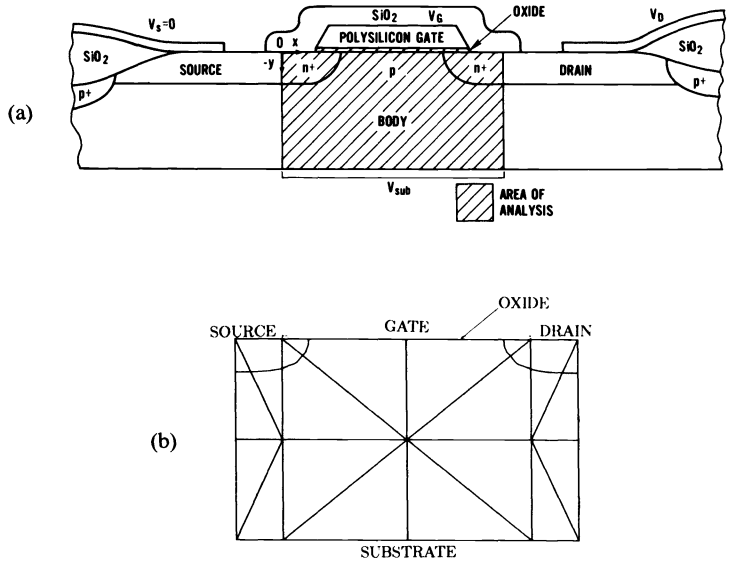


FIG. 2. (a) *Schematic of* MOS *transistor.* (b) *Primitive mesh for* MOS *transistor.*

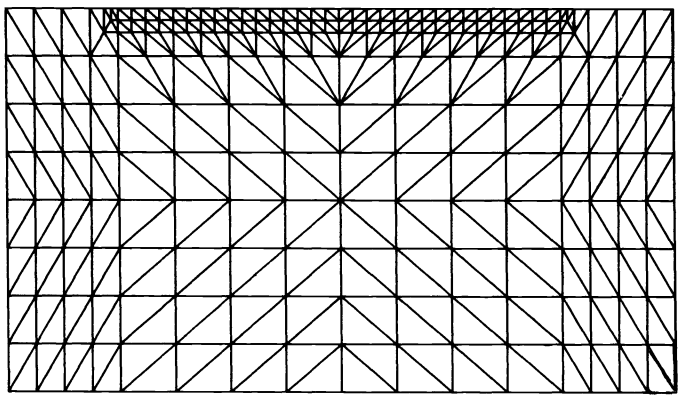


FIG. 3. *Level* 1 *mesh for* MOS *transistor.*

The user may not be able to predict where the mesh should be refined. As an example, an initial mesh for a beveled p–n junction is shown in Fig. 5. The beveled surface is assumed to have an infinite surface-recombination velocity. One might expect that the mesh needs to be refined near the arrow, the intersection of the junction with the surface. A refined mesh is shown in Fig. 6; the surface recombination distorts the electric field and the charge densities, and refinement is not needed near the arrow.

The difficulty of obtaining the level 1 solution is strongly dependent on the accuracy of the initial guess. The most primitive initial guess is to make the solution variables satisfy (13), (14), and (15), with $\phi_{MS} = 0$ at each point in the device. This is the local space-charge neutral condition. When Fermi statistics are used (16), (17), and (18) are used instead. Initial guesses of this type were used for the DMOS transistor and for the p–n junction. This type of initial guess is a type of worst-case guess; it is rarely very good, but can be used when nothing better is known. We have always been able to make B2DE converge with this guess.

The MOS2 driver, for MOS transistors, uses a more elaborate initial guess for the potential. In heavily doped material, local space-charge neutrality is assumed; this applies in the source and drain. The two-dimensional fields around each junction are rotated one-dimensional abrupt p–n junctions. An idealized one-dimensional inversion layer is assumed in the channel. In any region below the channel where junction and inversion-layer fields are both present, the larger of the two possible potentials is used. An initial guess of this type is shown in Fig. 7(a), and the final solution is shown in Fig. 7(b).

After an accurate solution of the nonlinear finite-element equations for a particular level, which is characterized by small L_2 and L_∞ norms of the residuals, and the L_2 norm of the Newton step less than kT/q (1 in normalized units), the discretization error in the solution can be reduced to any desired level of accuracy by refining the mesh adaptively, reducing the discretization error. For computational efficiency, several meshes are retained and the linear equations solved using multilevel iteration [2], [3].

The strategy for obtaining the level 1 solution is usually significantly different from the strategy used at higher levels, because the initial guess usually has low accuracy. In devices where one of the carriers is only perturbing the potential, it is fastest to start the iteration process by solving the potential alone, then solving the potential and the dominant carrier, and finally solving all three carriers together. This procedure should

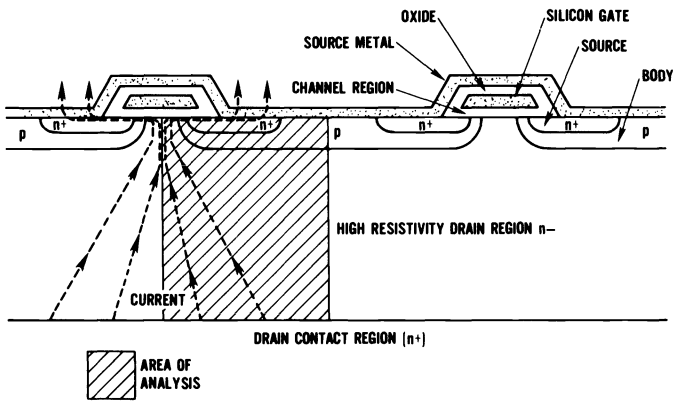


FIG. 4. (a) *Schematic of* DMOS *transistor. The shaded region is the region modeled.*

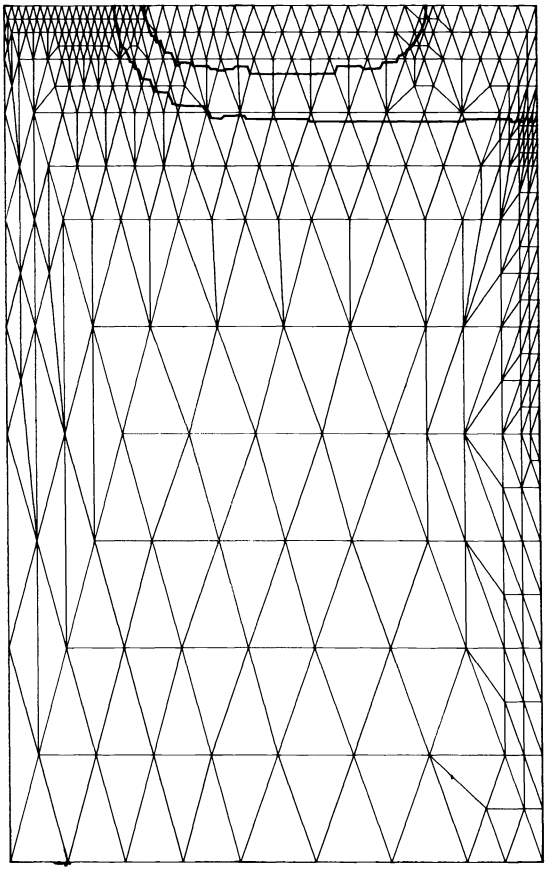


FIG. 4. (b) *Level* 1 *mesh for* DMOS *transistor and position of* p–n *junction.*

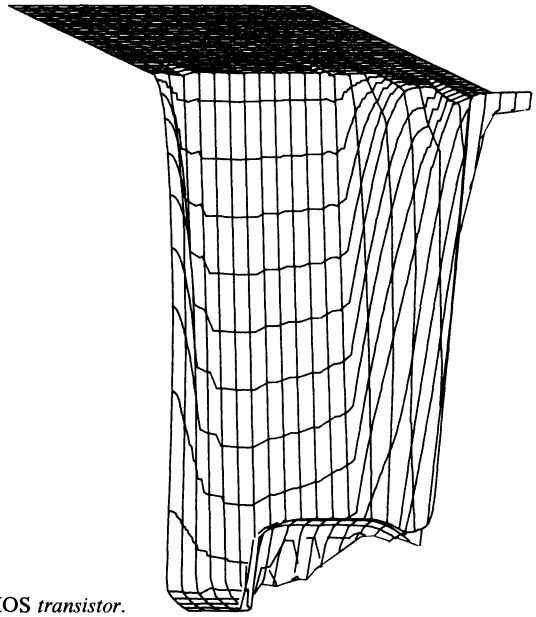
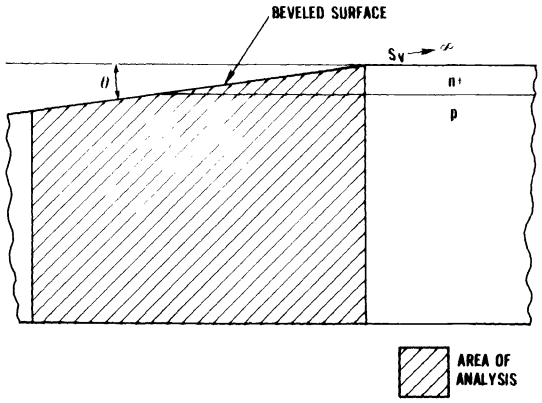


FIG. 4. (c) *Final potential solution for* DMOS *transistor.*

(a)



(b)

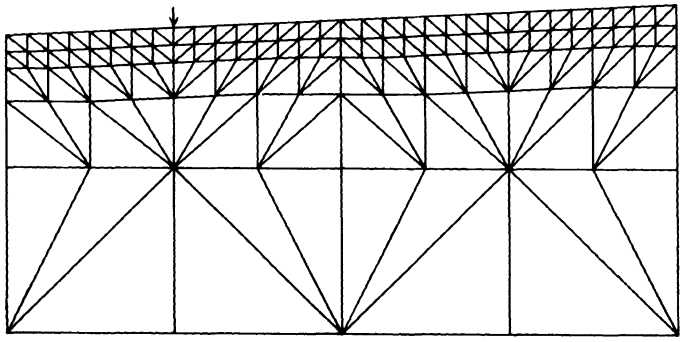


FIG. 5. (a) *Schematic of beveled* p–n *junction.* (b) *Level* 1 *mesh for beveled* p–n *junction. The junction is horizontal and intersects the surface at the arrow.*

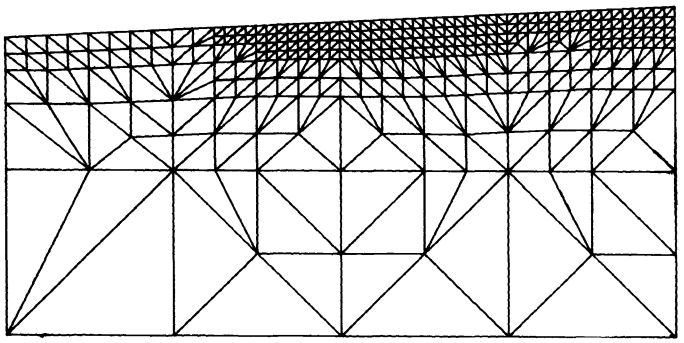


FIG. 6. *Level* 2 *mesh for beveled* p–n *junction.*

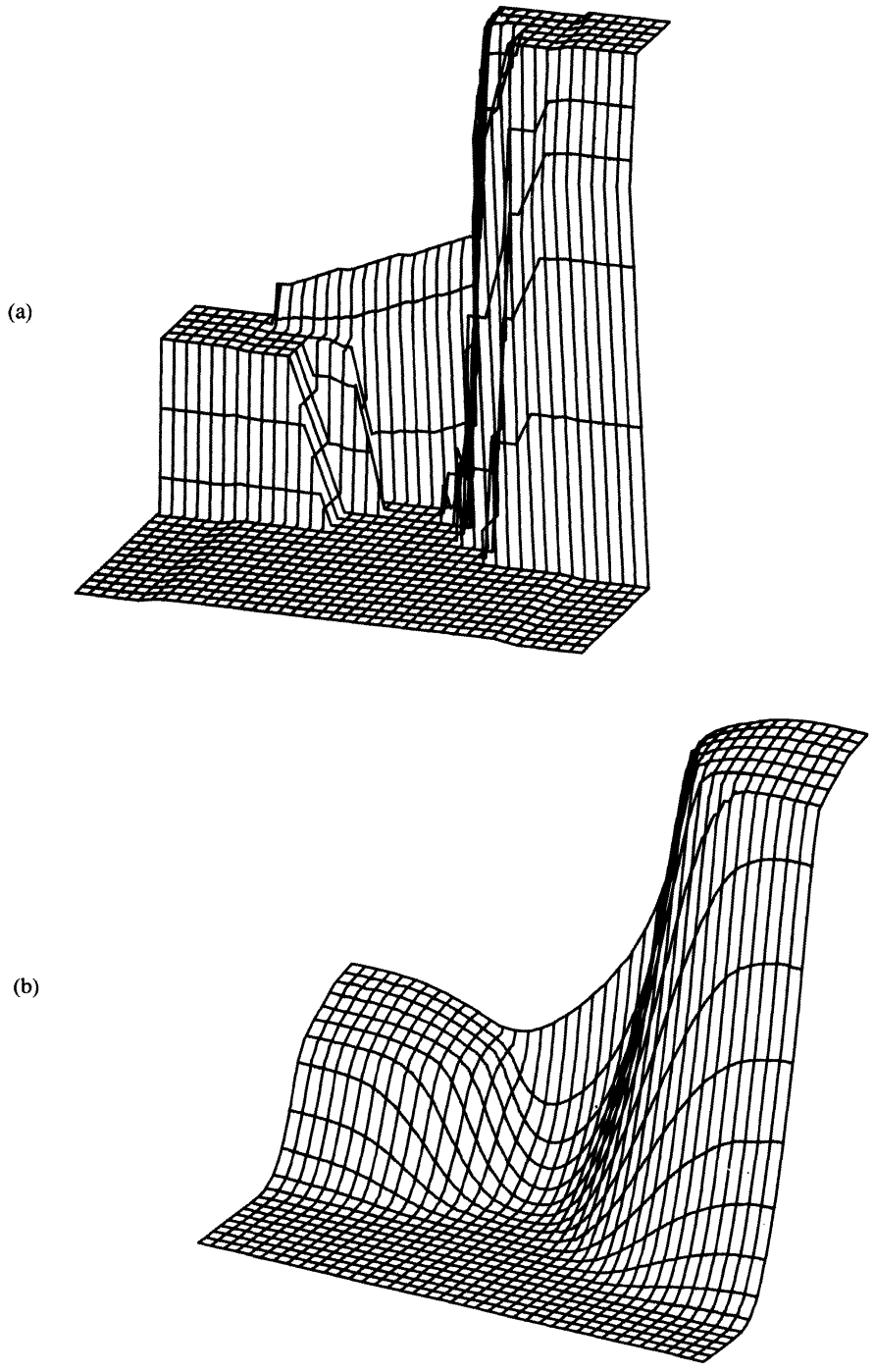


FIG. 7. (a) *Initial guess for* MOS *transistor problem.* (b) *Final potential for* MOS *transistor problem.*

be avoided when a good initial guess is available, since it adds to the total number of level 1 iterations. A typical level 1 potential solution is shown in Fig. 8.

Expanding the mesh from level 1 to level 2 may cause some difficulties. The initial mesh expansion may introduce too few new mesh points. When this occurs, it is necessary to compress level 2 into level 1 and treat the resulting mesh as level 1. Sometimes it is preferable to solve on the level 2 mesh, then to refine to level 3, and then compress level 3 into level 2 until the number of vertices on level 2 is at least twice the number on level 1.

The first Newton step taken on level 2 is critical. Since the solution is an exponential function of the change in potential and quasi-Fermi levels, the L_2 norm of the residual after taking this first step can be very large. Small reductions in the size of the Newton step bring exponential reductions in the L_2 norms. At level 2 and higher levels, the error in the potential usually dominates the iteration. This seems to be caused by the more rapid variation of the right-hand side of this equation with solution variables. A typical level 2 solution is shown in Fig. 9.

Mesh refinements beyond level 2 are usually routine. All equations may be used and the pattern of refinement is such that compressions and solution-norm problems are unusual. Convergence is rapid, usually five iterations or less. The quasi-Fermi levels converge more rapidly than the potential. A typical level 3 solution is shown in Fig. 10.

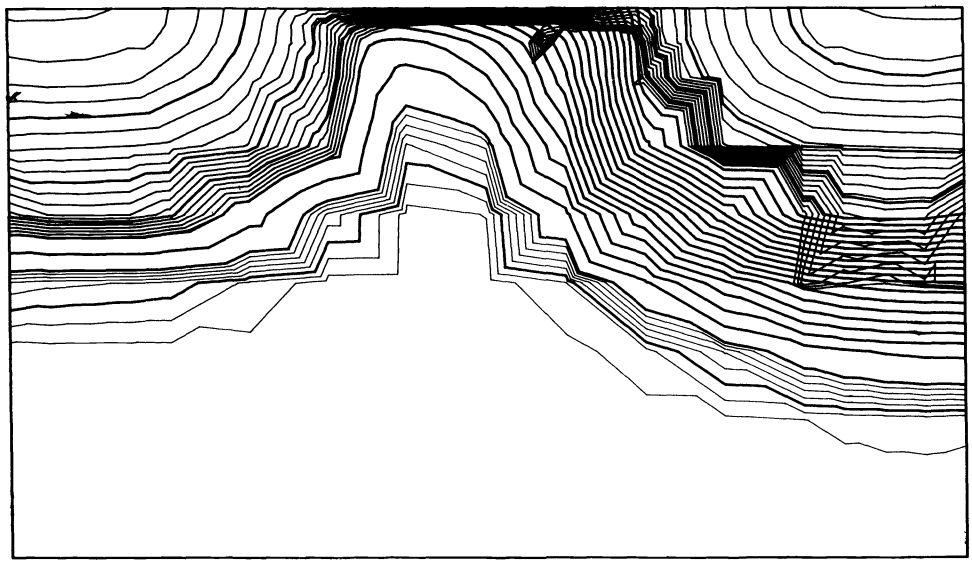
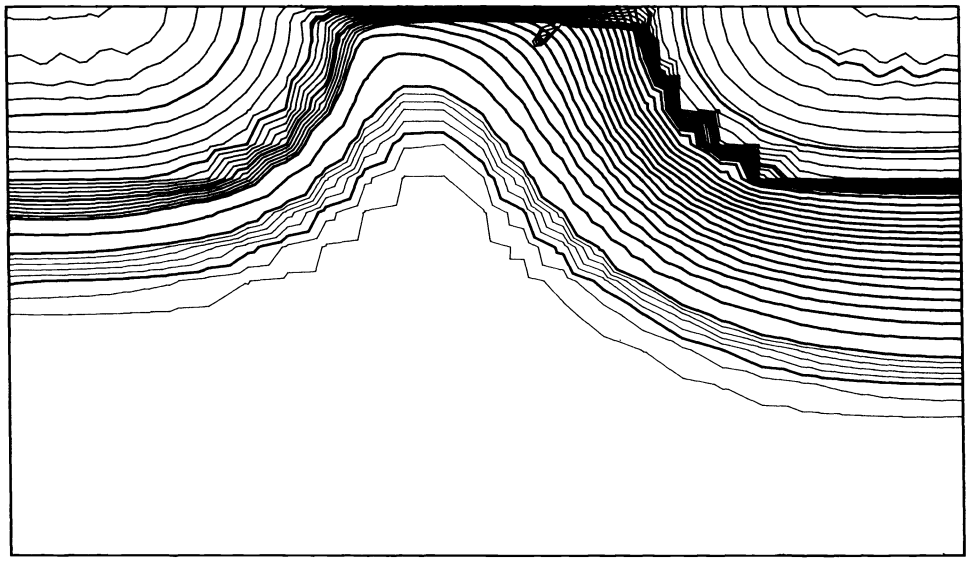


FIG. 8. *Level* 1 *solution for* MOS *transistor problem. The heavy lines are contours of* ψ *at* $5kT/q$ *intervals. The upper light lines are contours of* $\ln(n/n_i)$, *ranging from* $5kT/q$ *upward, in steps of* $1kT/q$. *The lower light lines are contours of* $\ln(p/n_i)$, *ranging from* $5kT/q$ *upward, in steps of* $1kT/q$.

(a)



(b)

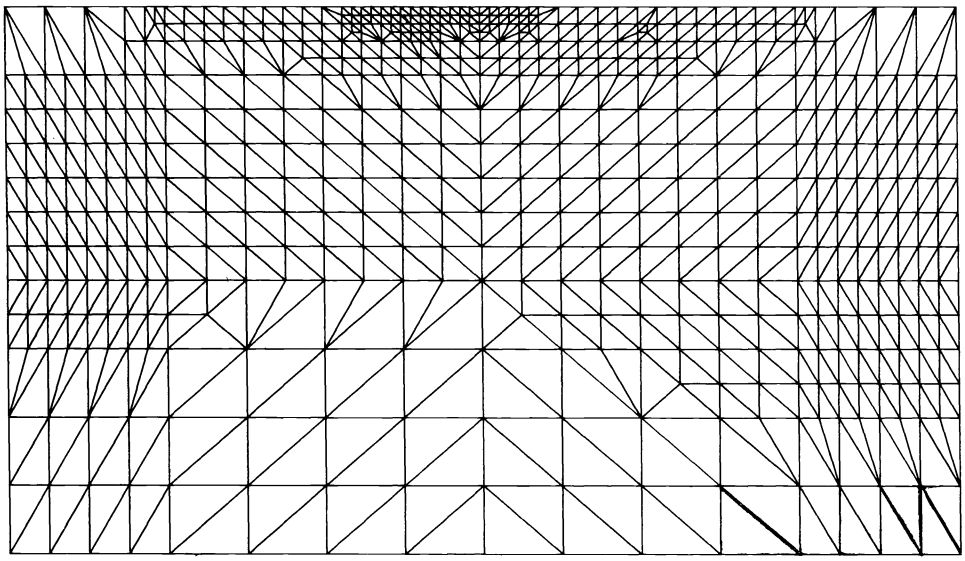
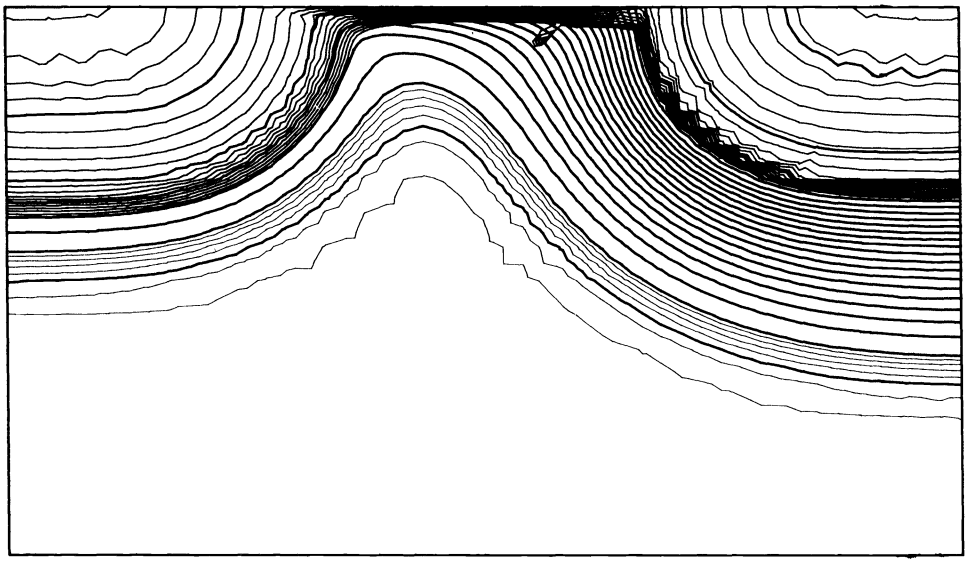


FIG. 9. (a) *Level 2 solution for* MOS *transistor problem. Contour lines are as in Fig.* 8. (b) *Level 2 mesh for* MOS *transistor problem.*

(a)



(b)

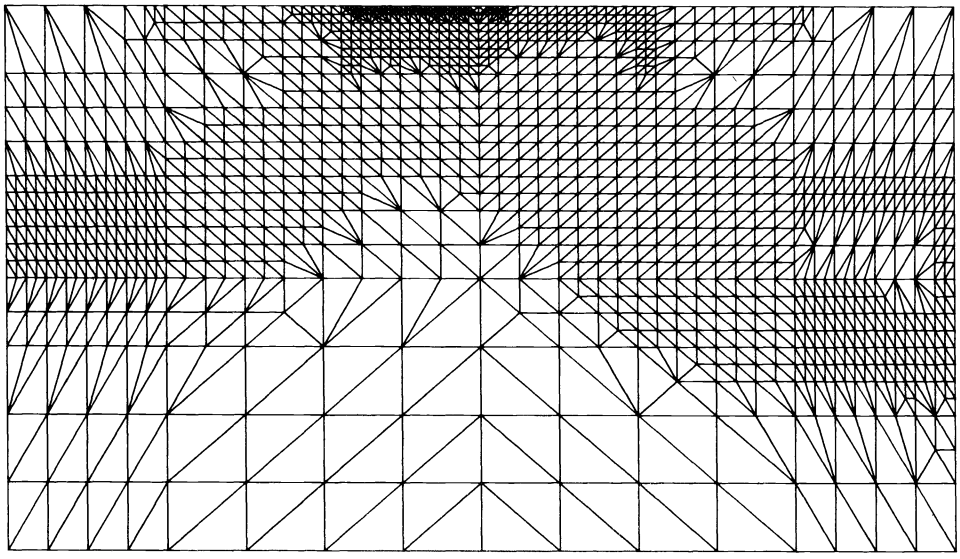


FIG. 10. (a) *Level* 3 *solution for* MOS *transistor problem. Contour lines are as in Fig.* 8. (b) *Level* 2 *mesh for* MOS *transistor problem.*

V. Accuracy requirements for calculation of currents and fields. In the previous section we discussed methods for obtaining accurate values of the potential and quasi-Fermi levels. Quasi-Fermi levels are not measurable quantities, and potentials can only be measured at the surface of the device.

The physically measurable quantities in semiconductor devices are related to the two- and three-dimensional vector fields in the device. The quantities of interest are currents and electric fields. These quantities involve gradients of the solution variables. If solutions are to represent physical quantities accurately, they must provide accurate values of the gradients. This is unfortunate, since B2DE can find the PDE variables to second-order accuracy, but the gradients only to first-order accuracy.

Calculation of currents has been widely used to evaluate the accuracy of device models. Currents calculated at poorly chosen points in the device may not so much reflect the solution as the location and method of calculation. The interpretation of current measurements depends in part on the accuracy of the physical parameters used in the device simulation compared. Uncertainties in these input parameters can make these comparisons difficult. As long as the model is applied to a narrow range of parameter values, empirically determined input parameters should be adequate. Disagreements arise when the model is applied outside the range where the empirical parameters have been verified.

Terminal currents are line integrals of the gradients of quasi-Fermi levels. Since this integration contains exponential functions of the potential and carrier densities, the choice of integration method and the location of the line are critical.

We have found that it is easy to be fooled into thinking that a solution on a mesh is sufficiently good to give an accurate current value. A smooth solution on a mesh which is apparently fine enough may not be adequate. To be sure of the solution, a solution on a more-refined mesh must be obtained; for safety, the mesh should be refined everywhere, since the adaptive mesh refinement is only approximate.

The hole and electron currents must be integrated by a combination of numerical and analytical methods in order to calculate them accurately, since they depend exponentially on the solution variables. We use a geometric mesh to perform this integration in the inversion region of a MOS transistor. The integrand is evaluated at half the total device depth, a quarter of the depth, and so on, so that the bulk of the quadrature points are near the inverted surface. Between any two mesh points the gradient of the quasi-Fermi level is assumed to be constant and the carrier density is assumed to vary exponentially. Using this procedure, the current can be calculated so that the principal source of error is the error in the gradient of the quasi-Fermi levels.

Electric fields provide another critical measure of model accuracy. Many physically important processes in devices depend on the value of the electric field in a small region of the device. Avalanche, carrier generation by band-to-band tunneling, and punchthrough all are effects which are localized in small high-field regions of the device. Avalanche and tunneling are localized because of their exponential dependence on field, and punchthrough is localized by two-dimensional electrostatics. The magnitude of the field in the inversion region is also important for models of field-dependent mobility [5], [15]. The other area where electric fields are important is at the oxide–silicon interface. Radiation effects in the oxide are determined by the size and polarity of the oxide field [23]. The oxide field is coupled to the field in the inversion layer by (20). The combination of these field effects is critical in scanning electron-microscope measurements of the oxide field [10].

The electric field is also the critical boundary condition in the operation of

field-effect devices, since it determines the current in the channel. The electric field boundary condition couples to the bulk of the semiconductor after being screened by the total channel charge. This coupling through the channel charge is critical in the formation of the channel inversion layer in both one and two dimensions [4], [22]. If the interface condition (20) is inaccurately satisfied, the calculated channel current will be wrong.

The accuracy of the potential solution in the oxide is also critical in solving (20). In devices with a thin oxide, it is more efficient to approximate the field as the local potential difference divided by t_{ox}. In devices with a thick oxide, it is important to solve Laplace's equation in the oxide accurately, so that the discretization error in the calculation is below the level that would change the charge in the channel.

In the level 1 solution process, errors in the initial guess at Dirichlet edges can be important. These errors often result from the intersection of p–n junctions with false symmetry lines, lines which are used to reduce the size of the mesh at the edges of the active device. More refined meshes in these regions must be used at level 1 to reduce these errors. At mathematical singularities, such as the edge of the gate or metal on a device surface, errors from these singularities can significantly effect the calculation of the electric field. Again the problem can be reduced by proper level 1 mesh refinement.

In the solution of SCD's, regional accuracy is necessary for calculation of vector fields. In regions where electrons dominate, accuracy in the electron solution is needed; in regions where holes dominate, accuracy in the hole solution is needed.

We have also learned that visual estimates and nice plots are not reasonable criteria for solution accuracy. The visual estimates of solution accuracy lead one to attempt uniform accuracy everywhere. Because of the exponential behavior, accuracy in ϕ_n is important mainly where $\psi - \phi_n$ is positive, and similarly for ϕ_p. Uniform accuracy is not usually desired.

This is particularly true in regions which have two strongly coupled variables interacting. In these regions, color graphics and careful choice of the plotting range of the solution variables is essential for visual analysis of solution errors. Two other constraints are always present in using interactive adaptive mesh-generation methods. All of the triangles in any high-level mesh will be similar to the triangles in the primitive mesh, affecting the shape of the other triangles used and the possible location of the allowed vertices. Even at level 1, the vertices must sample the basic equations sufficiently densely to show the effects of boundary conditions and internal parameters, such as doping.

VI. Summary. We have developed a capability for analyzing semiconductor devices in two dimensions, using a general-purpose PDE box, B2DE. We have been able to solve a wide variety of semiconductor devices, even starting from a poor initial guess. We have shown that these methods work for ideal and nonideal contacts and oxide interfaces, and for either Boltzmann or Fermi–Dirac statistics. Our analysis of a DMOS power transistor suggests that we will be able to analyze devices as complex as a single gate.

REFERENCES

[1] J. H. ALBERS, P. ROITMAN, AND C. L. WILSON, "Verification of models for fabrication of shallow arsenic junctions," to be published.

[2] R. E. BANK AND A. H. SHERMAN, "PLTMG users' guide," Center for Numerical Analysis, Univ. of Texas, Austin, Rep. CNA 152, Sept. 1979.

[3] R. E. BANK, "PLTMG users' guide, June, 1981 version," Dept. Math., Univ. of California, San Diego, Tech. Rep., Aug. 1982.
[4] J. R. BREWS, "A charge-sheet model of the MOSFET," *Solid-State Electron.*, vol. 21, pp. 345–355, 1978.
[5] J. A. COOPER, JR. AND D. F. NELSON, "Measurement of high-field drift velocity of electrons in inversion layers on silicon," *IEEE Electron Device Lett.*, vol. EDL-2, pp. 171–173, 1981.
[6] S. C. EISENSTAT, M. C. GURSKY, M. A. SCHULTZ, AND A. H. SHERMAN, "Yale sparse matrix package I. The symmetric codes," Dep. Comp. Sci., Yale University, New Haven, CT, Res. Rep. 112.
[7] W. FICHTNER, E. N. FULS, R. L. JOHNSTON, T. T. SHENG, AND R. K. WATTS, "Experimental and theoretical characterization of submicron MOSFET's," in *Proc. 1980 Int. Elec. Dev. Meeting*, pp. 24–27, Dec. 1980.
[8] R. FAIR, "Concentration profiles of diffused dopants in silicon," in *Applied Solid State Science Suppl. 2B*, Dawon Kahng, Ed. New York: Academic Press, 1981, pp. 1–108.
[9] P. A. FOX, A. D. HALL, AND N. L. SCHRYER, "The PORT mathematical subroutine library," *ACM Trans. Mathemat. Software*, vol. 4, pp. 104–126, 1978.
[10] R. C. FRYE AND J. H. LEAMY, "Direct observation of the gate oxide electric field distribution in silicon MOSFET's," *IEEE Electron Device Lett.*, vol. EDL-3, pp. 1–3, 1982.
[11] D. P. KENNEDY AND R. R. O'BRIEN, "Analysis of the impurity atom distribution near the diffusion mask for a planar p–n junction," *IBM J. Res. Develop.*, vol. 9, pp. 179–186, 1965.
[12] B. W. KERNIGHHAN, "RATFOR—A preprocessor for a rational Fortran," *Software — Practice and Experience*, vol. 5, 1975.
[13] B. G. RYDER, "The PFORT verifier," *Software — Practice and Experience*, vol. 4, pp. 359–377, 1974.
[14] P. ROITMAN, D. R. MYERS, J. H. ALBERS, J. R. EHRSTEIN, AND J. LOWNEY, "Direct observation of lateral redistribution profiles of shallow ion implants," *IEEE Trans. Electron. Devices*, vol ED-28, p. 1240, 1981.
[15] S. A. SCHWARZ AND S. E. RUSSEK, "Simple physical models for bulk and surface electron velocities in silicon," presented at the Device Res. Conf., June 21–23, 1982, Paper VA-1.
[16] A. H. SHERMAN, "Algorithms for sparse Gaussian elimination with partial pivoting," *ACM Trans. Mathemat. Software*, vol. 4, pp. 330–338, 1978.
[17] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method.* Englewood Cliffs, NJ: Prentice-Hall, 1973.
[18] P. SWARZTRAUBER AND R. SWEET, "Efficient FORTRAN subprograms for the solution of elliptic partial differential equations," NCAR Tech. Note NCAR-TN/IA-109, 1975.
[19] S. M. SZE, *Physics of Semiconductor Devices.* New York: Wiley 1981, pp. 16–57.
[20] K. K. THORNBER, "Current equations for velocity overshoot," *IEEE Electron Device Lett.*, vol. EDL-3, pp. 69–71, 1982.
[21] D. D. WARNER AND C. L. WILSON, "Two-dimensional concentration dependent diffusion," *Bell Syst. Tech. J.*, vol. 59, pp. 1–41, 1980.
[22] C. L. WILSON AND J. L. BLUE, "Two-dimensional finite element charge-sheet model of a short-channel MOS transistor," *Solid-State Electron.*, vol. 25, pp. 461–477, 1982.
[23] P. S. WINOKUR AND H. E. BOESCH, JR., "Interface-state generation in radiation-hard oxides," *IEEE Trans. Nucl. Sci.*, vol. NS-27, p. 1647, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403—0007/$01.25/0

RELAXATION-BASED ELECTRICAL SIMULATION*

ARTHUR RICHARD NEWTON† AND ALBERTO L. SANGIOVANNI-VINCENTELLI†

Abstract. Circuit simulation programs have proven to be most important computer-aided design tools for the analysis of the electrical performance of integrated circuits. One of the most common analyses performed by circuit simulators and the most expensive in terms of computer time is nonlinear time-domain transient analysis. Conventional circuit simulators were designed initially for the cost-effective analysis of circuits containing a few hundred transistors or less. Because of the need to verify the performance of larger circuits, many users have successfully simulated circuits containing thousands of transistors despite the cost.

Recently, a new class of algorithms has been applied to the electrical IC simulation problem. New simulators using these methods provide accurate waveform information with up to two orders of magnitude speed improvement for large circuits. These programs use *relaxation* methods for the solution of the set of ordinary differential equations, which describe the circuit under analysis, rather than the direct sparse-matrix methods on which standard circuit simulators are based.

In this paper, the techniques used in relaxation-based electrical simulation are presented in a rigorous and unified framework, and the numerical properties of the various methods are explored. Both the advantages and the limitations of these techniques for the analysis of large IC's are described.

I. Introduction.

Circuit simulation programs, such as SPICE2 [1] and ASTAP [2], have proven to be most important computer-aided design tools for the analysis of the electrical performance of integrated circuits (IC's). These programs can perform a variety of analyses, including dc, ac, and time-domain transient analysis of circuits containing a wide range of nonlinear active circuit devices such as MOSFET's and bipolar junction transistors [3].

One of the most common analyses performed by circuit simulators and the most expensive in terms of computer time is nonlinear time-domain transient analysis. By performing this analysis, precise electrical waveform information can be obtained if the device models and parasitics of the circuit are characterized accurately. However, conventional circuit simulators were designed initially for the cost-effective analysis of circuits containing a few hundred transistors or less. Because of the need to verify the performance of larger circuits, many users have successfully simulated circuits containing thousands of transistors despite the cost. For example, a 700 MOSFET circuit, analyzed for 4 μs of simulated time with an average 2-ns time step, takes approximately 4 CPU hours on a VAX11/780 VMS computer with floating-point accelerator hardware.

Gate-level logic simulators (e.g., [4], [5]) and switch-level simulators [6]–[8] can verify circuit function and provide first-order timing information more than three orders of magnitude faster than a detailed circuit simulator. However, to verify circuit performance for critical paths, memory design, and analog circuit blocks, it is often essential to perform accurate electrical simulation. In some companies the simulation of circuits containing many thousands of devices is performed routinely and at great expense. In recent years, considerable effort has been focussed on techniques for improving the speed of time-domain electrical analysis while maintaining acceptable waveform accuracy.

*Received by the editors May 25, 1983. This research was supported in part under contract N00039-K-0251, by JSEP under contract AFOSR-F49620-79C0178, by Army Research Office under contract DAAG29-81-K-0021, and by the National Science Foundation under contract ECS-7913148.

†Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720.

A number of approaches have been used to improve the performance of conventional circuit simulators for the analysis of large circuits. The time required to evaluate complex device model equations has been reduced using table-lookup models [9]–[13]. Techniques based on special-purpose microcode have been investigated for reducing the time required to solve sparse linear systems arising from the linearization of the circuit equations [14]. Node-tearing techniques have also been used to exploit circuit regularity by bypassing the solution of subcircuits whose state is not changing [15], [16] and to exploit the vector-processing capabilities of high-performance computers such as the CRAY-1 [17]. In all cases, the overall speed improvement of the simulation has been at most an order of magnitude, for practical circuits.

Recently, a new class of algorithms has been applied to the electrical IC simulation problem. New simulators using these methods provide as accurate, or more accurate, waveforms than standard circuit simulators such as SPICE2 or ASTAP with up to two orders of magnitude speed improvement for large circuits. These simulators have been used for the analysis of both digital and analog MOS IC's. They use *relaxation* methods for the solution of the set of ordinary differential equations, (ODE's) which describe the circuit under analysis, rather than the direct sparse-matrix methods on which standard circuit simulators are based.

A broad survey of decomposition techniques for the simulation of large-scale integrated circuits can be found in [18]. In this paper, the techniques used in relaxation-based electrical simulation are presented in a rigorous and unified framework and the numerical properties of the various methods are explored. Both the advantages and the limitations of these techniques for the analysis of large IC's are described. In Section II, some of the fundamental problems associated with conventional circuit simulation algorithms as circuit size increases are exposed and the mathematical basis for the relaxation approach is introduced. In Section III, the special relaxation methods called *timing simulation* algorithms are described and their numerical properties are investigated. In Section IV, *iterated timing analysis*, which applies relaxation techniques at the nonlinear equation level [19], is described briefly and its convergence properties are proven. The *waveform relaxation* method [20], [21], which applies relaxation techniques at the differential equation level, is presented in Section V, and various techniques which can be used to improve its performance for electrical simulation are described. Concluding remarks and areas requiring further research are presented in Section VI.

II. Circuit equation formulation and standard relaxation techniques.

A. Equation formulation. Before the techniques used in relaxation-based simulation are presented, the particular electrical simulation problem to be solved must be defined. Although relaxation-based methods can be used with a variety of technologies (e.g., [23]), they are particularly suited to the analysis of large MOS digital IC's, as will become clear later. Thus to help clarify the presentation, the following simplifying assumptions are made:

- All resistive elements, including active devices, are characterized by constitutive equations where voltages are the controlling variables and currents are the controlled variables.
- All energy storage elements are two-terminal, possibly nonlinear, voltage-controlled capacitors.
- All independent voltage sources have one terminal connected to a ground or can be transformed into independent current sources with the use of the Norton transformation.

Under these assumptions, the circuit equations can be formulated in terms of a nodal analysis that yields N equations in N unknown node voltages [24], where there are $N+1$ nodes in the circuit and node $N+1$ is the reference node, or ground.

An important assumption required by relaxation-based electrical simulators is that a two-terminal capacitor be connected from each node of the circuit to the reference node. This assumption is satisfied by circuits where lumped parasitic capacitances are present between circuit interconnect and ground or on the terminals of active circuit elements.

Under these assumptions, the nodal equations can be written in the form

$$C(v(t), u(t))\dot{v}(t) = -f(v(t), u(t)), \qquad 0 \leqslant t \leqslant T, \tag{1}$$
$$v(0) = V$$

where $v(t) \in \mathbb{R}^n$ is the vector of node voltages at time t; $\dot{v}(t) \in \mathbb{R}^n$ is the vector of time derivatives of $v(t)$; $u(t) \in \mathbb{R}^n$ is the input vector at time t, $C(\cdot)$: $\mathbb{R}^n \to \mathbb{R}^{n \times n}$ represents the nodal capacitance matrix, f: $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, and

$$f(v(t), u(t)) = \left[f_1(v(t), u(t)), f_2(v(t), u(t)), \cdots, f_N(v(t), u(t))\right]^T$$

where $f_i(v(t), u(t))$ is the sum of the currents charging the capacitors connected to node i. In the following sections (1) will be referred to in a simplified form where the time dependencies are expressed implicitly, i.e.,

$$C(v, u)\dot{v} = -f(v, u). \tag{2}$$

B. Standard circuit simulation. A simplified flow diagram for the solution of these equations by a conventional circuit simulator is shown in Fig. 1. Once the circuit description has been read by the program and the data structures required for simulation have been assembled, the main analysis loop (Steps (2)–(13)) is entered.

At each new analysis time point, t_{n+1}, the information from previous time points is used to predict the solution at t_{n+1}. Stiffly stable integration formulas, such as Backward Euler (BE), the Trapezoidal Rule (TR), or Gear's Variable-Order Method

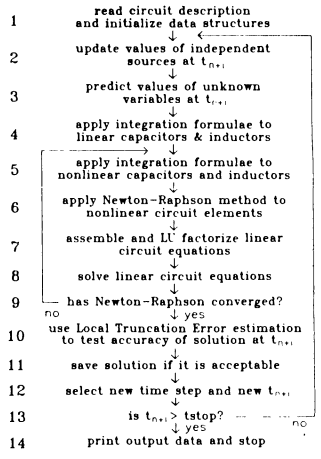


FIG. 1. *Circuit simulator flow diagram for transient analysis.*

(GE), with variable time steps, are used to discretize (1) at Steps (4) and (5) [3]. This process yields a set of nonlinear, algebraic difference equations of the form

$$g(x) = 0 \tag{3}$$

where $x \in \mathbb{R}^N$ is the vector of node voltages at time t_{n+1}.

These equations are solved using a damped Newton–Raphson algorithm to yield a set of sparse linear equations of the form

$$Ax = b \tag{4}$$

where $A \in \mathbb{R}^{N \times N}$ is a matrix related to the Jacobian of g and $b \in \mathbb{R}^N$ [3]. Typically, less than 2 percent of the entries of A are nonzero for $N > 500$. These equations are then solved using direct methods, such as sparse LU decomposition or Gaussian Elimination, Steps (7) and (8).

Steps (5)–(9) are repeated until the Newton–Raphson process converges or the upper bound on the number of iterations is reached. The program then decides whether to accept the solution, based on its estimate of local truncation error (LTE) and the number of Newton–Raphson iterations required in Steps (5)–(9). A new time step is computed, and Steps (2)–(13) are repeated until the simulation is complete [3].

This procedure has proven to be reliable and accurate. For large circuits, the process can take a considerable amount of computer time, as illustrated in Section I. The majority of the time spent in Steps (2)–(13) can be lumped into two categories: the time required to solve the system of sparse linear equations, SOLVE (Steps (7) and (8)), and the time required to form the entries of A and b in (4), FORM (Steps (5) and (6)).

Fig. 2 shows the amount of CPU time required to perform a transient analysis of a set of typical circuits of increasing size. For this example, the number of circuit nodes N is used as a measure of circuit size. The time required for equation preprocessing is not included here; only time involved in the actual time-domain transient portion of the simulation is shown. A simple RC circuit was chosen for this example to emphasize the increasing cost of matrix solution time. The example was constructed by calling an

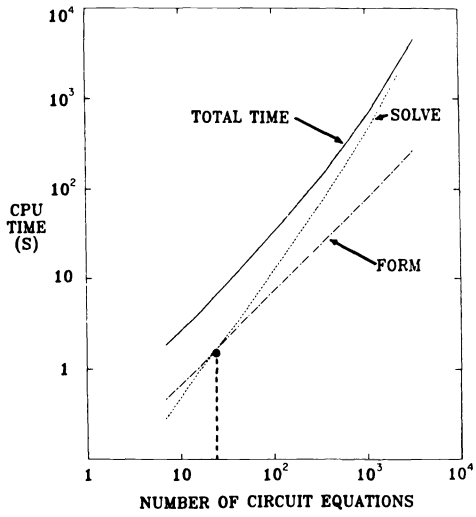


FIG. 2. *Transient analysis time for circuits of increasing size.*

increasing number of cells, in an hierarchical manner, each with the same matrix structure and an average number of fanouts between 2.5 and 3. This approach preserved the observed properties of most real circuits while providing a uniform technique for increasing circuit size.

As can be seen in Fig. 2 for small circuits ($N < 20$), the majority of the solution time is spent performing FORM. However, when the size of the circuit grows, an increasing percentage of the time is spent in the SOLVE phase. While the actual percentages may vary depending on the circuit under analysis, the complexity of the nonlinear device models used by the program, and the computer on which the simulator is running, this trend is true for all standard circuit simulators running on conventional computers. For MOS circuits analyzed on a VAX11-780 UNIX computer, the crossover point is at around 500 nodes. The time spent in the equation solution phase has been measured to grow as $O(N^{\beta})$, where $1.1 < \beta < 1.5$. In particular, for large circuits β has been found to depend on the difference between the time required to perform arithmetic operations and the memory bandwidth of the computer. On the other hand, the time required for FORM grows linearly with the number of circuit elements and, therefore, with the number of circuit equations for typical circuits. The time spent in the load phase can be reduced by simplifying the device model equations, using table look-up models [9]–[13], or providing special-purpose instructions to update A and b [14].

For most circuits the fraction of nodes which are changing their voltage value at a given point in time decreases as the circuit size increases. For circuits containing over 500 MOSFET's, fewer than 20 percent of the node voltages change significantly over a simulation time step. Only the circuit equations representing these active nodes must be solved at any time. Circuit simulators exploit this *time sparsity* or *latency* by using device-level [1] or block-level [16], [25], [17] bypass schemes. In a device-level bypass scheme, if the terminal voltages and branch currents of a circuit element did not change significantly in the previous Newton–Raphson iteration, its contributions to A and b in (4) are not reevaluated, and the values computed during the previous iteration are used. In block-level bypass, both the matrix element evaluation and the node solution steps are bypassed for each block of inactive connected circuit elements. While the aforementioned techniques do reduce the total execution time for conventional circuit simulators, the savings are often not sufficient for the cost-effective electrical simulation of LSI circuits.

C. Linear relaxation methods. Relaxation methods can be used for the solution of (1) in a number of ways. In all cases, their principal advantages stem from the fact that they do not require the direct solution of a large system of linear equations and from the fact that they permit the simulator to exploit latency efficiently.

Relaxation methods can be applied at different stages in the solution of (1), as illustrated in Fig. 3. The two most common methods used in electrical simulation are the Gauss–Jacobi method and the Gauss–Seidel method [26], [27].

For the solution of the linear equations, relaxation methods can replace direct methods for the solution of (4). Let A be split into $L + D + U$, where $L \in \mathbb{R}^n$ is strictly lower triangular, $D \in \mathbb{R}^n$ is diagonal, and $U \in \mathbb{R}^n$ is strictly upper triangular. Then the two methods mentioned earlier have the following form when applied to the solution of (4):

Gauss–Jacobi:

$$Dx^{k+1} = -(L+U)x^k + b \tag{5a}$$

or

$$x^{k+1} = -D^{-1}\big((L+U)x^k - b\big) \triangleq M_{GJ}x^k + D^{-1}b; \tag{5b}$$

Gauss–Seidel:

$$(L+D)x^{k+1} = -Ux^k + b \tag{6a}$$

or

$$x^{k+1} = -(L+D)^{-1}(Ux^k - b) \triangleq M_{GS}x^k + (L+D)^{-1}b \tag{6b}$$

where x^k is the value of x at the kth iteration.

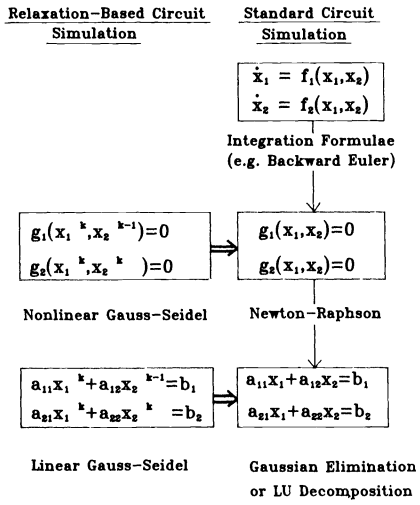


FIG. 3. *Parallel between standard circuit simulation techniques and relaxation-based techniques.*

Since relaxation methods are iterative methods, it is important to ask under what conditions they are guaranteed to converge to the solution of (4).

Note that the iterations are not well defined if D is singular. That is, if there is a zero on the main diagonal of A. It is well known that a necessary and sufficient condition for the iterations defined by (5b) and (6b) to converge to the solution of (4), independent of the initial guess x_0, is that the eigenvalues of M_{GJ} and M_{GS} be inside the unit circle in the complex plane [26]. However, this condition is not practical from a computational point of view and other conditions, in general sufficient conditions, are used to check the convergence of these methods. In particular, it can be shown that if A is strictly diagonally dominant, then both the Gauss–Jacobi and the Gauss–Seidel iteration converge to the solution of (4). Other sufficient conditions can be found in [26], [28].

Another important convergence property of iterative methods is rate of convergence. It can be shown that if the Gauss–Jacobi and the Gauss–Seidel iteration converge, they converge at least linearly. That is, after a sufficiently large number of iterations, the error at each iteration decreases according to

$$\|x^{k+1} - \hat{x}\| \leqslant \varepsilon \|x^k - \hat{x}\|$$

where $\hat{x}$ is the solution of (4).

The computational cost of both of these methods is $O(N)$, compared with $O(N^{1.1-1.5})$ for direct, sparse-matrix techniques. Thus relaxation methods are advantageous from a computational point of view with respect to sparse-matrix techniques only if the number of iterations needed to obtain convergence is of the order of $N^{0.1}$. In addition, sparse-matrix techniques are based on Gaussian elimination or LU decomposition and, if exact arithmetic is used, they obtain the exact solution of (4) in one step. Relaxation techniques, as mentioned previously, are not guaranteed to converge. Reliability is the basic reason why sparse-matrix techniques have been used more frequently than relaxation techniques in conventional circuit simulators. If the Gauss–Seidel method is used, reordering of the equations has an effect on the number of iterations needed to obtain a solution of (4). For example, if A is upper triangular, N iterations are needed to obtain the exact solution of (4). However, if A is reordered into lower triangular form, the solution of (4) is obtained in a single iteration. If the Gauss–Jacobi iteration is used, reordering of the equations has no effect on the speed of the algorithm.

The Gauss–Seidel method can be shown to converge faster than the Gauss–Jacobi method on a class of problems [26].[1] For example, if A is lower triangular, Gauss–Seidel converges to the exact solution of (4) in one iteration while Gauss–Jacobi converges in N iterations. However, the fact that at each iteration each $x_i^{k+1}, i=1,\cdots,N$, does not depend on any $x_j^{k+1}, j=1,\cdots N; j\neq i$ in the Gauss–Jacobi method means that the computation of all $x_i^{k+1} i=1,\cdots,N$ can proceed in parallel. This method is, therefore, well suited to modern multiprocessor computers.

D. Nonlinear relaxation methods. Relaxation methods can also be used at the nonlinear-equation solution level to augment the Newton–Raphson method, and hence replace the linear-equation solution based on sparse-matrix techniques. Let x^k denote the value of x at the kth iteration. The Gauss–Jacobi and Gauss–Seidel algorithms when applied to (3) have the following form:

Nonlinear Gauss–Jacobi algorithm:

(7)

repeat {
  forall (j **in** N) {
    solve $g_j(x_1^k,\cdots,x_j^{k+1},\cdots,x_N^k)=0$ for x_j^{k+1};
  }
}
until ($\|x^{k+1}-x^k\|\leqslant\varepsilon$)

that is, until convergence is obtained. The **forall** (i **in** J) construct specifies that the computations for all values of i in the set J may proceed concurrently, i.e., in parallel and in any order.

Nonlinear Gauss–Seidel algorithm:

(8)

repeat {
  foreach (j **in** N) {
    solve $g_j(x_1^{k+1},\cdots,x_j^{k+1},\cdots,x_N^k)=0$ for x_j^{k+1};
  }
}
until ($\|x^{k+1}-x^k\|\leqslant\varepsilon$)

[1] Note that examples can be found where Gauss–Jacobi converges faster than Gauss–Seidel.

The **foreach** (i **in** J) construct specifies that the computations for each value of i in the ordered set J must proceed sequentially and in the order specified by the set. For this method the actual order in which the node equations are solved may be determined either statically or dynamically, as described later in Subsection III-B.

The nonlinear Gauss–Jacobi and Gauss–Seidel iterations are well defined only if each equation described in (7) and (8) has a unique solution in some domain under consideration. In the linear case, the iterations were well defined if D was nonsingular. In the nonlinear case we have a similar condition. In addition, the conditions under which these methods converge are also analogous to the ones given for the linear case.

Let $g'(x)$ denote the Jacobian of g computed at x. Let g be continuously differentiable in an open neighborhood S_0 of $\hat{x}$ for which $g(\hat{x}) = 0$. Let $g'(\hat{x})$ be split as $L(\hat{x}) + D(\hat{x}) + U(\hat{x})$ where $L(\hat{x})$, $D(\hat{x})$, and $U(\hat{x})$ are, respectively, the strictly lower triangular part, the diagonal part and the strictly upper triangular part of $g'(\hat{x})$. Let $M_{GJ}(\hat{x})$ and $M_{GS}(\hat{x})$ be defined as follows:

$$M_{GJ}(\hat{x}) = -D(\hat{x})^{-1}(L(\hat{x}) + U(\hat{x})) \tag{9}$$

and

$$M_{GS}(\hat{x}) = -(D(\hat{x}) + L(\hat{x}))^{-1}U(\hat{x}). \tag{10}$$

Assume that $D(\hat{x})$ is nonsingular and that all the eigenvalues of $M_{GJ}(\hat{x})$ and $M_{GS}(\hat{x})$ are inside the unit circle. Then there exists an open ball $S \subset S_0$ such that the nonlinear Gauss–Jacobi and the Gauss–Seidel iterations are well defined and for any $x_0 \in S$, the sequence generated by the iterations converges to $\hat{x}$.

This result assumes that (7) and (8) can be solved exactly. Since these equations are nonlinear, there is no hope of computing the solutions exactly in finite time. Therefore an iterative method must be used. In general, the Newton–Raphson method is used to solve these equations. Note that for each relaxation iteration, N decoupled equations, each in one unknown, must be solved. Thus the implementation of the Newton–Raphson method is straightforward. These "composite" methods are called the Gauss–Jacobi–Newton and Gauss–Seidel–Newton methods to specify that the Newton iteration is performed inside the nonlinear Gauss–Jacobi and Gauss–Seidel iterations, respectively [27].

It is important to determine when to stop the iteration of the "inner" Newton–Raphson loop to achieve the same convergence as in the ideal case when the solutions of (7) and (8) are computed exactly. It turns out rather surprisingly that *one iteration only of the Newton method on* (7a) *and* (8a) *is sufficient to preserve the convergence properties of the nonlinear relaxation methods* [27]. In particular, the rate of convergence of the nonlinear Gauss–Seidel method is the same as the rate of convergence of the Gauss–Seidel–Newton method.[2]

Note that the convergence result presented above is local in the sense that the iterations are guaranteed to converge only if the initial guess is sufficiently close to a solution. In this respect, the convergence properties of relaxation methods are similar to the ones of the Newton–Raphson method. However, the eigenvalue condition of relaxation methods is much stronger than the other conditions of Newton–Raphson methods. Moreover, the rate of convergence of relaxation methods is only linear while it is quadratic for Newton–Raphson methods. This explains why Newton–Raphson methods are preferred in standard circuit simulation. However, each iteration of a

[2]Note that rate of convergence is an asymptotic measure of the speed of the algorithm, i.e., of the number of iterations needed to achieve a given accuracy. Performing additional iterations of the inner Newton–Raphson loop may make the outer relaxation loop converge in fewer iterations, in some cases.

relaxation method involves a set of decoupled equations while Newton–Raphson methods require the solution of a set of simultaneous equations. In addition, relaxation methods are ideally suited to exploit the latency of the circuit under analysis as described in the following sections.

A comparison can be made of the use of relaxation methods at the linear and nonlinear equation level. If the relaxation methods are applied at the linear equation level and the iteration of the inner relaxation loop is carried to convergence, then the convergence of the Newton methods is not affected. However, if the inner loop is not carried to convergence, but a fixed number of iterations is allowed, then the convergence of the outer Newton loop is affected. In fact, if only one iteration of the inner relaxation loop is taken, then the convergence of the "Newton–Gauss" methods is only linear. If more iterations are taken, then the rate of convergence asymptotically improves to be quadratic [27]. The use of relaxation at the linear equation level involves the computation of the Jacobian of g, which is quite expensive as mentioned earlier. Nonlinear relaxation methods coupled with an inner Newton–Raphson loop only need the computation of the partial derivative of g_i with respect to x_i, resulting in a considerable saving of computer time per iteration.

As in the linear case, the Gauss–Seidel method tends to converge faster than Gauss–Jacobi. Reordering of the equations affects the speed of the Gauss–Seidel method crucially. In this task, the *dependency matrix* of (3) plays an important role. The dependency matrix is defined to be a zero-one matrix $P = [p_{ij}]$ such that $p_{ij} = 1$ if g_i depends on x_j, $p_{ij} = 0$ otherwise. Note that P also represents the zero-nonzero structure of the Jacobian of g.

If P is lower triangular, then only one iteration of the outer Gauss–Seidel relaxation loop is needed, provided that the inner Newton–Raphson loop is run to convergence. If P is not lower triangular, but the dependency of the g_i component of g on x_j, $j < i$, is "weak," then the Gauss–Seidel method converges rather quickly. Then a key issue in applying relaxation techniques to the solution of circuit equations is the reordering of the equations so that P is almost lower triangular. This task can be performed both statically and dynamically, as described in the next section. Since MOS devices are almost unidirectional from gate to drain and gate to source due to the electrical decoupling between the gate and the source and drain of the device, and if all capacitors used in the simulation have one node tied to a ground and the circuit does not contain any MOS transmission gates and no feedback connections, then the equations can be reordered statically to yield a lower triangular P. This property provides an intuitive explanation as to why relaxation methods are successful for the simulation of MOS digital circuits.

E. Conclusions. To conclude this preliminary section, note that in Fig. 3 there is a "hole" in the relaxation counterpart of the flow diagram of standard circuit simulation at the differential equation level. Until recently, relaxation techniques had been used only at the linear and nonlinear equation levels. The *waveform relaxation* method, presented in Section V, fills that gap.

III. Timing simulation.

A. Introduction. The first successful application of relaxation methods to electrical-circuit analysis was in *timing* simulation [9]–[12]. In timing simulation, only one relaxation iteration is performed per time step while one or more Newton–Raphson iterations may be performed to solve each nodal equation. Since the relaxation loop is not taken to convergence, a small time step must be used to bound local errors and the saturating properties of digital MOS circuits are exploited to bound error propagation.

Timing simulators have proved successful when applied to constrained IC design methods, such as standard cell [31] or gate array, but have not been as successful in the custom-design environment. Since there is no way to guarantee accuracy for an arbitrary connection of MOSFET's unless at least two relaxation iterations are performed per time step, timing simulators have produced incorrect results in some situations. A circuit designer will use a program that gives the correct simulation result and occasionally gives no result (e.g., no convergence at a time point). A circuit designer soon loses confidence in a program that occasionally gives an incorrect answer! Many timing simulators that were developed in-house in industry are no longer in use although where they do remain in use, they continue to be very successful. When used correctly, timing simulators can provide over two orders of magnitude speed improvement over conventional circuit simulators for comparable waveform accuracy.

As described in detail later, timing simulation has problems analyzing circuits containing tight feedback loops, pass transistors or *floating elements*.[3] In particular, floating capacitors are not handled satisfactorily. Early timing simulators avoided the problem of analyzing circuits with floating capacitors by not allowing the user to include them in the circuit description. Hence, it is assumed here that the nodal capacitance matrix is diagonal, that it is nonsingular for the entire range of node voltages of interest (this implies nonzero grounded capacitances), and that the circuit equations are written as

$$\dot{v} = -C(v,u)^{-1} f(v,u) = -F(v,u). \tag{11}$$

Algorithms used for timing analysis often discretize the derivative operator by Backward Euler [9], [10], [30] or the Trapezoidal Rule [29]. For the sake of simplicity, in the following description the Backward Euler formula will be used:

$$\dot{v}_{k+1} = (v_{k+1} - v_k)/h,$$

where the time step $h = t_{k+1} - t_k$ and v_{k+1} and v_k are the computed values of the node voltages at time t_{k+1} and t_k, respectively. The solution of the resulting nonlinear system of equations

$$v_{k+1} - v_k + hF(v_{k+1}, u(t_{k+1})) = 0 \tag{12}$$

is then approximated by *one sweep* of a relaxation technique.

Program MOTIS [9] used a modified Gauss–Jacobi technique which yields the following set of decoupled equations:

$$v_{k+1}^n = v_k^n - hF_n\left(v_k^1, \cdots, v_k^{n-1}, v_{k+1}^n, v_k^{n+1}, \cdots, v_k^N, u_n(t_{k+1})\right), \qquad n = 1, \cdots, N. \tag{13}$$

The solution of the decoupled nonlinear equations of (6) is then approximated by taking a single step of a *regula falsi* iteration [32].

The MOTIS-C [10] and SPLICE1 [29] programs use a modified Gauss–Seidel technique. In SPLICE this technique yields

$$v_{k+1}^n = v_k^n - hF(\tilde{v}_{k+1,n}, u(t_{k+1})), \qquad n = 1,2,\cdots,N \tag{14}$$

where

$$\tilde{v}_{k+1,n} = \left[v_{k+1}^1, \cdots, v_{k+1}^n, v_k^{n+1}, \cdots, v_k^N\right]^T. \tag{15}$$

The solution of (14) is then approximated by using one or more steps of the Newton–Raphson algorithm. The program can cut the time step locally at a node and

[3] A floating element is a two-terminal capacitor or resistor whose terminals are not ground or power supply.

use a number of small time steps to achieve a satisfactory solution before the next equation in the relaxation solution is to be processed.

B. Network ordering. Unless some form of connection graph is used to establish a precedence order for signal flow, the new node voltages will be computed in an arbitrary order. As pointed out in Section II, in a Gauss–Jacobi-based simulator, where only node voltages at t_k are used to evaluate the node voltage at t_{k+1}, the order of processing elements will not affect the results of the analysis. However, substantial timing errors may occur. For example, consider the inverter chain of Fig. 4. If the input to inverter I_1 changes at time t_k, that change cannot appear at node (1) before time t_{k+1}. For a chain of N inverters, the change will not appear at output I_N before N time steps have elapsed. If the time step is very small with respect to the response time of any one inverter, this error may not be significant.

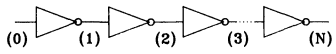


FIG. 4. *Inverter chain example.*

In a Gauss–Seidel-based simulator, where node voltages already computed at t_{k+1} are made available for the evaluation of other node voltages at t_{k+1}, the order of processing elements can affect the simulator performance substantially. In the previous example, if the inverters were processed in the order $1,2,3,\cdots,N$ then v_{k+1}^1 can be determined from v_{k+1}^0, v_{k+1}^2 from v_{k+1}^1, and so on. The result will be zero accumulated timing error. Should the nodes happen to be processed in the reverse order, $N, N-1,\cdots,1$, then a timing error of N time steps will occur, the same error as in the Gauss–Jacobi–Newton iteration.

If it were possible to order the processing of nodes in the Gauss–Seidel–Newton iteration so as to follow the flow of the signal through the circuit, the timing error would be kept small. A signal flow graph would provide this information. An example of a circuit fragment and associated signal flow graph, illustrating the fanins and fanouts of the nodes, is shown in Fig. 5. One way to generate this graph is to consider the dependency matrix introduced in Section II. This zero-one matrix can be considered as the adjacency matrix of a directed graph $G = G(X, E)$, where X is the set of vertices and E is the set of directed edges of the graph. An edge connects nodes x_i to node x_j if $p_{ij} = 1$. By the definition of dependency matrix, given a circuit and its node equations written as in (11), this graph indicates that if an edge connects x_i to x_j, then the voltage of node i, v_i, can affect the value of the voltage of node j, v_j via the device equation. Thus the set of vertices that are connected by an edge going in x_i identifies all the nodes in the circuit that affect the value of the voltage of node i. These are called *fanin nodes* of node i. Similarly, the set of nodes that are connected to node x_i by an edge going out of x_i identifies all the nodes in the circuit whose voltage is affected by the voltage of node i. These are called *fanout nodes* of node i. Note that a node can be both a fanin and a fanout node of node i.

Fanin and fanout *elements* can also be defined. The fanin elements of node i are defined as those which play some part in determining the voltage at node i, i.e., those elements that cause some entries of row i in the dependency matrix to be one. For example, any MOS transistor, modeled by the simple Shichman–Hodges [33] equations

shown in Fig. 6, whose drain or source is connected to a node would be classified as a fanin element at that node since its drain or source current may affect the node voltage.

A fanout element of node i is one whose operating conditions are directly influenced by the voltage at node i, i.e., those elements that cause some entries of column i in the dependency matrix to be one. For MOS transistors, connection to any of the three independent ports (drain, gate, or source) would cause that MOS transistor to be included in the fanout-element set at the node. It is therefore possible for an element to appear as both a fanin and a fanout at the node.

SPLICE1 builds the signal flow graph by constructing two tables for each node as the circuit is read. First, all circuit elements are classified as fanin and/or fanout elements of the nodes to which they are connected. The two tables constructed for each node contain the names of the fanin and fanout elements at the node. These tables are

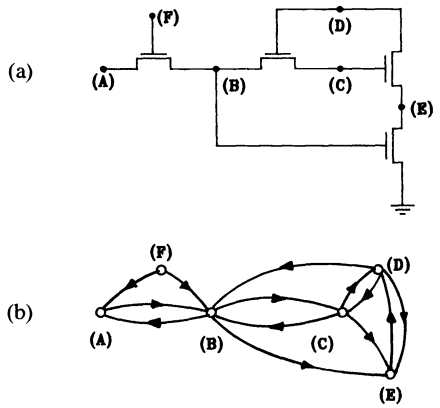
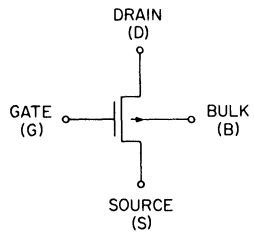


FIG. 5. (a) *Circuit fragment.* (b) *Associated signal flow graph.*



$$V_t = VTO + \gamma(\sqrt{V_{bs} + \phi} - \sqrt{\phi})$$
$$V_{gse} = V_{gs} - V_t$$
$$V_{dse} = MAX(V_{ds}, V_{gse})$$
$$I_{ds} = K(V_{gse} - V_{dse}/2)V_{dse}(1 + \lambda V_{ds})$$

FIG. 6. (a) n-*channel* MOS *transistor.* (b) *Simple Shichman–Hodges n-channel* MOS *model equations.*

generated as the elements are read into memory. If this graph is acyclic, then it can be *levelized* [12, p. 427], where a level number corresponding to the longest path (most branches traversed) from any independent source to a node is assigned to each node. If the graph does contain cycles, special steps must be taken to break these feedback loops. Then if the nodes are processed in the order of level numbers, it is clear that an optimal static ordering for Gauss–Seidel processing will be achieved. An ordering can also be achieved by finding the strongly connected components of the graph [34]. The levelized graph provides a *static* ordering of the network.

Whenever the voltage at a node changes, it is possible to *schedule* all of its fanouts to be processed. In this way, the effect of a change at the input to a circuit may be *traced* as it propagates to other circuit nodes via the fanout tables, and thus via the circuit elements which are connected to them. Since the only nodes processed are those which are affected directly by the change, this technique is *selective* and hence its name: *selective trace*. If a selective-trace algorithm is used with the fanin and fanout tables, the order in which the node voltages are updated becomes a function of the signals flowing in the network, and is therefore a dynamic ordering. This approach is often used in modern logic simulators and is also the ordering technique used in the SPLICE program.

Even with selective trace some timing errors can occur. For example, wherever feedback paths exist, one time step of error may be introduced. Consider the circuit fragment and its associated signal flow graph shown in Fig. 5. Assume v_1 and v_2 are such that both M_1 and M_2 are conducting. If a large input appears at node (1) at time t_{k+1}, it will be traced through nodes (1), (2), (3), and (4), respectively. Now, however, the change in voltage at node (4) caused node (1) to be marked to be processed again at this time. Since only one sweep of Gauss–Seidel is being used, the solution of node (1) a second time would be illegal. Rather, the node (1) is scheduled to be processed one time step in the future, and thus it is possible that one time step of timing error has been introduced.

C. Exploiting latency. As mentioned earlier, large digital circuits are often relatively inactive. A number of schemes can be used to avoid the unnecessary computation involved in the reevaluation of the voltage at nodes which are inactive or *latent*.

A scheme used in many electrical simulators is the "bypass" scheme, described in Section II for conventional circuit simulators. This scheme has also been employed in a number of timing simulators. However, when the majority of the nodes in the circuit are latent, the task of simply checking each node to determine if it can be bypassed can dominate the total run time.

The use of the selective-trace technique for dynamic ordering can provide a major time saving here. By constructing a list of all nodes which are active at a time point and excluding those which are not, selective trace allows circuit latency to be exploited without the need to check each node for activity. The elimination of this checking process, used in both the bypass approach and the static levelizing scheme described previously, can save a significant amount of computer time for large circuits at the cost of some extra storage for the fanin and fanout tables at each node.

In an efficient implementation of the selective-trace technique, the fanin and fanout tables do not contain the "names" of fanin and fanout elements, respectively, but rather a *pointer* to the actual location in memory where the data for each element is stored.

D. Numerical properties of timing algorithms. A major drawback with the use of timing analysis is that tightly coupled feedback loops, or bidirectional circuit elements, can cause severe inaccuracies and even instability during the analysis. For example, if the Gauss–Seidel "one-sweep" timing-analysis method is applied to the circuit of Fig. 7 limiting the time-step to 0.1 s, the waveforms of Fig. 8(a) are obtained. However, if the time step is set to 0.8 s, then the computed solution blows up as shown in Fig. 8(b). This demonstrates that timing algorithms do not inherit the numerical properties of the discretization formulate used to approximate the time derivative. In fact, Backward Euler is well known to be A stable, i.e., the computed solution of the circuit differential equations should not "blow up" independent of the choice of time step as long as the simulated circuit is stable.

The reason why this idiosyncrasy is observed is that timing algorithms do not solve (5) since only one sweep of the relaxation iteration is taken. Therefore the stability and accuracy properties of the integration method used to discretize the derivative operator no longer hold. As a matter of fact, the combination of the discretization formula, the various relaxation steps, and the Newton–Raphson method form a *set of new integration algorithms*. These integration methods use an implicit formula to discretize the differential equations, but they do not solve the nonlinear equation obtained. Thus they are somewhat in between explicit and implicit methods.

In the following description, the "time-advancement" algorithms which use the Gauss–Jacobi and the Gauss–Seidel relaxation step will be referred to as *Gauss–Jacobi* and *Gauss–Seidel integration algorithms*, respectively. This perspective allows the understanding of the numerical behavior of timing algorithms and the development of better techniques for timing simulation.

An analysis of numerical properties of the Gauss–Jacobi and Gauss–Seidel integration algorithms when applied to MOS circuits has been carried out in [35], [36]. Only the most important results are outlined here. First consider the case where no floating capacitors are present in the circuit to be analyzed.

The numerical properties of an integration method, such as stability, are studied on test problems [37], [38], which are simple enough to allow a theoretical analysis but still sufficiently general that some insight can be obtained about how the method will behave in general. For the widely used linear multistep methods, the test problem consists of a linear time-invariant asymptotically stable autonomous differential equation. Unfortunately this simple test problem cannot be used to evaluate relaxation-based time-advancement techniques. In fact, each variable of the system of differential equations is treated differently according to the ordering in which equations are processed. Hence a more complex test problem is needed. The test problem chosen here is a linear time-invariant asymptotically stable *system* of autonomous differential equations, i.e.,

$$\dot{x} = Ax, \qquad x(0) = X \tag{16}$$

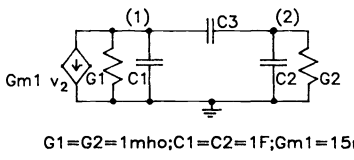


FIG. 7. *Schematic diagram of the example circuit.*

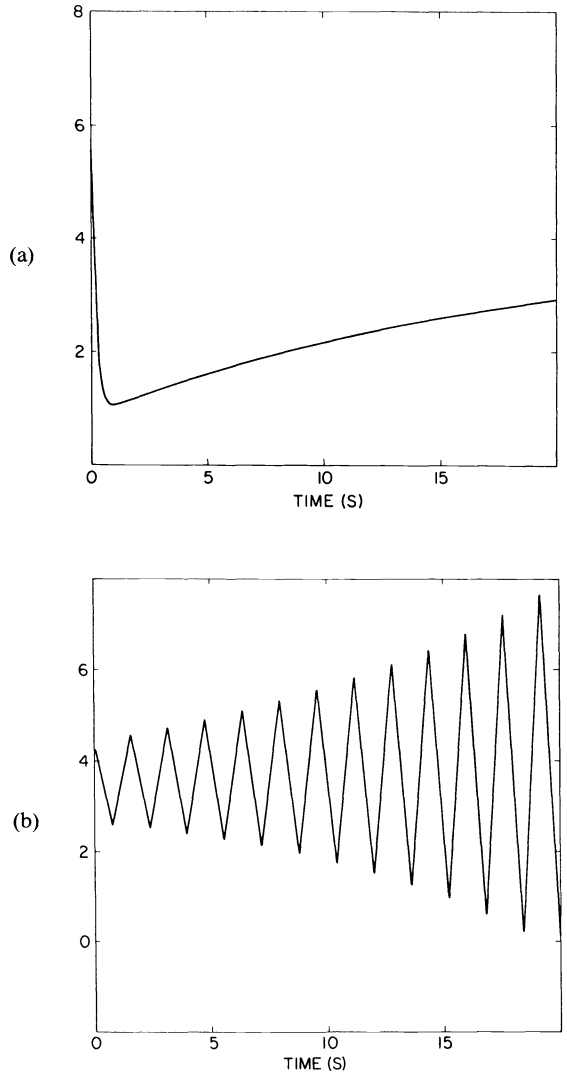


FIG. 8. (a) *Accurate waveform of voltage* v_1 *computed with a* 0.1-s. *time-step.* (b) *Waveform of voltage* v_1 *computed with* 0.8-s *time-step.*

where $A \in R^{n \times n}$ and the set of eigenvalues (spectrum) of A, $\sigma(A)$, is in the open left-half complex plane, i.e., $\sigma(A) \in C_0^-$.

In circuit theoretic terms, linear circuits whose natural frequencies are in the open left-half plane and which satisfy the assumptions described in Section II are considered as test circuits. Let $A = L + D + U$, where L is strictly lower triangular, D is diagonal, and U is strictly upper triangular. The time-advancement methods presented in Section III applied to the test system of (16) yield the following recursive relations:

Gauss–Jacobi integration algorithm:

$$[I - hD]x_{k+1} = [I + h(L+U)]x_k, \tag{17}$$

$$x_{k+1} = M_{GJ}(h)x_k \tag{18}$$

where I is the identity matrix and

$$M_{GJ}(h) = [I - hD]^{-1}[I + h(L+U)]. \tag{19}$$

Gauss–Seidel integration algorithm:

$$[I - h(D+L)]\, x_{k+1} = [I + hU]\, x_k, \tag{20}$$

$$x_{k+1} = M_{GS}(h)\, x_k \tag{21}$$

where

$$M_{GS}(h) = [I - h(D+L)]^{-1}[I + hU]. \tag{22}$$

The matrices $M_{GJ}(h)$ and $M_{GS}(h)$ are called the companion matrices of the methods. If the generic companion matrix of a method is denoted $M(h)$, then

$$x_k = [M(h)]^k x_0. \tag{23}$$

The numerical properties of the integration algorithms described by (23) are now described following the outline of one-step integration methods applied to ordinary differential equations [37].

The first numerical property of the integration methods to investigate is accuracy. This property relates the error introduced by the discretization process and the time step.

DEFINITION III-D-1. Let $x(t_k)$ be the exact value of the solution of the test problem at time t_k. Let x_k be the computed solution at time t_k assuming $x_{k-1} = x(t_{k-1})$, i.e., that no error has been made in computing the value of x at the previous time point. If $h = t_k - t_{k-1}$, the local truncation error is defined to be

$$\varepsilon = \|x(t_k) - x_k\|. \tag{24}$$

If $\varepsilon = O(h^{r+1})$, r is said to be the order of the integration method [37].

It has been observed experimentally that if the time step is decreased, the accuracy of the solution computed by timing algorithms improves in almost the same way as Backward Euler. In fact, the Gauss–Jacobi and the Gauss–Seidel integration algorithms have the same accuracy as the Backward Euler integration method.

THEOREM III-D-2. *Gauss–Jacobi and Gauss–Seidel integration algorithms are first order integration algorithms.*

In circuit analysis, another important criterion for evaluating the accuracy of an integration method, can be defined as *waveform accuracy*. In general, the computed solution of a system of differential equations is the superposition of a principal solution and associated parasitic solutions. Parasitic solutions are generated by the numerical approximations of the integration methods. In particular, an nth order integration algorithm yields $n-1$ parasitic solutions when applied to the test problem. For the algorithms under consideration in this paper, the displacement techniques introduce additional spurious components called *numerical solution components*.

If the original system to be analyzed does not contain an oscillatory component, the presence of such a component in the computed solution can be misleading in the evaluation of the performances of the system. As was shown in the previous subsection, the Gauss–Seidel integration algorithm introduces spurious oscillations in the computed solution of the equations describing a circuit where the exact solution does not have any oscillations. It is necessary to introduce methods that allow the evaluation of the "waveform accuracy" of the integration methods. To this end, a subclass of the test problem is now introduced, characterized by $\sigma(A) \in R_0^-$, i.e., the set of test problems

which does not have an oscillatory component in the solution, and bounds on the oscillatory components of the computed solutions must be established.

Theorem III-D-2 provides a bound on the oscillatory components of all the methods. In particular it is clear that, by choosing an appropriately small step size h, the numerical solution oscillatory components can be made negligible with respect to the principal solution.

Another fundamental property of integration methods is stability. Stability can also be defined in terms of the test problem.

DEFINITION III-D-3 (*stability*). An integration algorithm is stable if $\exists \delta > 0, \exists N > 0$ such that $\forall x_0 \in R^n, \exists \bar{k} > 0$ and

$$\|x_k\| < N, \quad \forall k \geqslant \bar{k}, \quad \forall h \in [0, \delta] \tag{25}$$

where x_k is the sequence generated by the algorithm applied to the test problem according to (23).

It is obvious that a numerical method that is not stable is of no practical use. It happens that both relaxation-based integration algorithms are stable as stated in the following theorem which is proven in [35].

THEOREM III-D-4. *Gauss–Jacobi and Gauss–Seidel integration algorithms are stable.*

The accuracy and the stability of the integration algorithms explain the success of timing simulators, but they also point out what the problems are with their use. In particular, note that δ of Definition III-D-3 can be quite small, i.e., that the time step may have to be reduced not for accuracy reasons, but to make sure that the computed solution does not blow up. Moreover, it is difficult to identify oscillations in the computed solutions as spurious.

To cope at least in part with these problems, another displacement technique for the solution of (1) has been proposed for a simple circuit in [42]. This algorithm is a symmetric-displacement method reminiscent of the alternating-direction implicit method [32] and is based on a class of methods proposed by Kahan [39]. The basic idea here is to "symmetrize" the Gauss–Seidel scheme with a method that takes two half-steps of size $\frac{h}{2}$ each: one half-step is taken in the usual "forward" (i.e., lower triangular) direction, the second half-step is taken in the backward (i.e., upper triangular) direction.

This method is introduced with the help of the linear system described by (16). The first half-step corresponds to the Gauss–Seidel method. Let $A = L + D + U$, then

$$\left(I + \frac{h}{2}\left(\frac{1}{2}D + L\right)\right)x_{k+1/2} = \left(I - \frac{h}{2}\left(\frac{1}{2}D + U\right)\right)x_k. \tag{26}$$

Note that there is a difference between (20) and (26) since D has been split into two parts here. This splitting of D is necessary to "symmetrize" the method. The backward half-step is then

$$\left(I + \frac{h}{2}\left(\frac{1}{2}D + U\right)\right)x_{k+1} = \left(I - \frac{h}{2}\left(\frac{1}{2}D + L\right)\right)x_{k+1/2}. \tag{27}$$

Consider the simple example of Fig. 9. The first half-step yields

$$\left(1 + \frac{h}{4}\frac{G_1 + G_3}{C_1}\right)x^1_{k+1/2} = \left(1 - \frac{h}{4}\frac{G_1 + G_3}{C_1}\right)x^1_k + \frac{h}{2}\frac{G_3}{C_1}x^2_k,$$

$$\left(1 + \frac{h}{4}\frac{G_2 + G_3}{C_2}\right)x^2_{k+1/2} = \frac{h}{2}\frac{G_3}{C_2}x^1_{k+1/2} + \left(1 - \frac{h}{4}\frac{G_2 + G_3}{C_2}\right)x^2_k.$$

The backward step involves the solution of

$$\left(1+\frac{h}{4}\frac{G_2+G_3}{C_2}\right)x^2_{k+1}=\frac{h}{2}\frac{G_3}{C_2}x^1_{k+1/2}+\left(1-\frac{h}{4}\frac{G_2+G_3}{C_2}\right)x^2_{k+1/2},$$

$$\left(1+\frac{h}{4}\frac{G_1+G_3}{C_1}\right)x^1_{k+1}=\frac{h}{2}\frac{G_3}{C_1}x^2_{k+1}+\left(1-\frac{h}{4}\frac{G_1+G_3}{C_1}\right)x^1_{k+1/2}.$$

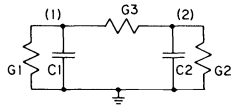


FIG. 9. *Simple circuit to illustrate the modified symmetric Gauss–Seidel method.*

If these formulas are generalized for the nonlinear case, and with

$$\bar{v}_{l,i}=\begin{cases}\left[v^1_l,\cdots,v^i_l,v^{i+1}_{l-1/2},\cdots,v^n_{l-1/2}\right]^T & \text{if } 2l \text{ is odd,} \quad (28)\\ \left[v^1_{l-1/2},\cdots,v^{i-1}_{l-1/2},v^i_l,\cdots,v^n_l\right]^T & \text{if } 2l \text{ is even,} \quad (29)\end{cases}$$

the forward step yields

$$v^i_{k+1/2}-v^i_k+\frac{h}{4}F_i\left(\bar{v}_{k+1/2,i},u(t_{k+1/2})\right)+\frac{h}{4}F_i\left(\bar{v}_{k+1/2,i-1},u(t_{k+1/2})\right)=0,\qquad i=1,2,\cdots,N \quad (30)$$

and the backward step yields

$$v^i_{k+1}-v^i_k+\frac{h}{4}F_i\left(\bar{v}_{k+1,i},u(t_{k+1})\right)+\frac{h}{4}F_i\left(\bar{v}_{k+1,i+1},u(t_{k+1})\right)=0,\qquad i=N,N-1,\cdots,1. \quad (31)$$

The solution of the decoupled equations is then approximated by taking one step of the Newton–Raphson algorithm.

This method can be proven to be more accurate and more stable than the previous one. The additional work required to perform the intermediate step is compensated by the additional accuracy as specified by the following theorem.

THEOREM III-D-5. *The modified symmetric Gauss–Seidel algorithm is a second-order integration algorithm.*

In addition, the "waveform accuracy" of this method is better than that of the other integration algorithms. If the class of the test problems is restricted to the subclass characterized by a symmetric A matrix, then a strong result for the modified symmetric Gauss–Seidel integration method can be obtained. In circuit theoretic terms, this analysis applies to linear circuits whose node equations yield a symmetric nodal admittance matrix when only the resistive part of the circuit is considered. Moreover it is required that this matrix remain symmetric when premultiplied by C^{-1}, the diagonal matrix of the grounded capacitors. A sufficient condition for this to occur is that the circuit consists of two terminal linear resistors and capacitors and that the grounded capacitors be of equal value. The case where the capacitors are not of equal value can

also be included in this class provided that a scaling of the rows of the matrix is performed.

THEOREM III-D-6. *If A is a real, symmetric matrix, the spectrum of the companion matrix of the modified symmetric Gauss–Seidel integration method is real, i.e., no oscillatory parasitic components are present in the computed solution.*

This theorem guarantees that the time step does not have to be limited to eliminate spurious oscillations at least for reciprocal circuits. Numerical results obtained with an experimental timing simulator show that spurious oscillations do not appear in the solution of nonlinear nonreciprocal circuits as well [36], [40].

For computational efficiency, it is desirable that the step size be limited only by accuracy considerations as in the case of the implicit backward differentiation formulas [37]. In the case of classical multistep methods, the concept of A-stability [38] and stiff-stability [37] have been introduced to test the "unconditional" stability of multistep methods. For the "time-advancement" techniques presented in this paper, it makes sense to define a similar concept. Unfortunately, general results of "unconditional" stability are not available for the test problem defined previously, but only for a subclass; once more the subclass characterized by a symmetric A matrix.

DEFINITION III-D-7 ($\tilde{A}$ *stability*). *An integration method is $\tilde{A}$ stable if* $\exists N > 0$ *such that* $\forall x_0 \in R^n, \exists \bar{k}$

$$\|x_k\| < N, \quad \forall k \geq \bar{k}, \quad \forall h \in [0, \infty) \tag{32}$$

where $\{x_k\}$ is the sequence generated by the method applied to the test problem of (16) *with A symmetric.*

THEOREM III-D-8. *The modified symmetric Gauss–Seidel method is $\tilde{A}$ stable.*

Note that no $\tilde{A}$ stability result for the Gauss–Jacobi and the Gauss–Seidel integration methods has been proven. In our practical experiments, we have seen that when applied to real circuit problems, the modified symmetric Gauss–Seidel method is indeed "more stable" than the other two methods.

E. Floating capacitors. As mentioned in the introduction to this section, a circuit element that has limited the application of timing analysis is the floating capacitor [41], [42].

The floating capacitor is often an important element in the design of integrated circuits. In Fig. 10(a) the value of the bootstrap capacitor C_b is generally large compared with the values of the associated parasitic grounded capacitors C_1 and C_2. The value of the intrinsic gate-drain feedthrough capacitance C_{gd} in Fig. 10(b) is often small compared with other circuit parasitics at the gate and drain nodes; however, the effect of C_{gd} on circuit performance can be significant due to the large voltage gain of the stage.

When floating capacitors are present in the circuit to be analyzed, the timing simulation algorithms presented in the previous sections take a different form. For the sake of simplicity, consider a linear time-invariant circuit described by

$$C\dot{v} = -Gv, \qquad v(0) = V \tag{33}$$

where C is the node capacitance matrix and G is the node conductance matrix. If C is inverted, the methods described previously apply. However, inverting C is expensive and most of the advantages of timing simulation algorithms would be lost. Thus if the Backward Euler formula is used to discretize the circuit equations at time t_{k+1}, v_{k+1} is given by

$$C(v_{k+1} - v_k) = -hGv_{k+1} \tag{34}$$

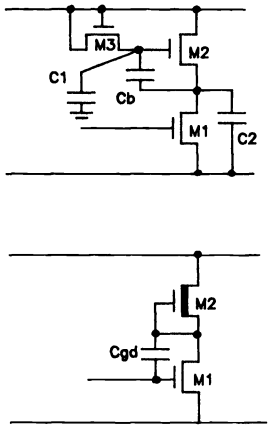


FIG. 10. (a) *Bootstrap inverter circuit and* (b) *effect of* C_{GD} *feed-through.*

or, rearranging (34), v_{k+1} is given by

$$(C + hG)v_{k+1} = Cv_k. \tag{35}$$

Let C be split as $C_d + C_l + C_u$ and G as $G_d + G_l + G_u$, where C_d and G_d are diagonal matrices, C_l and G_l are strictly lower triangular matrices, and C_u and G_u are strictly upper triangular matrices. Then, the time-advancement Gauss–Jacobi algorithm for circuits described by (33) becomes

$$(C_d + hG_d)v_{k+1} = (C - C_l - C_u - hG_l - hG_u)v_k = (C_d - h(G_l + G_u))v_k \tag{36}$$

and the Gauss–Seidel time-advancement algorithm is

$$(C_d + C_l + h(G_d + G_l))v_{k+1} = (C - C_u - hG_u)v_k = (C_l - hG_u)v_k \tag{37}$$

and finally the modified symmetric Gauss–Seidel algorithm is

$$\left(\frac{1}{2}C_d + C_l + \frac{h}{2}\left(\frac{1}{2}G_d + G_l\right)\right)v_{k+1/2} = \left(\frac{1}{2}C_d + C_l - \frac{h}{2}\left(\frac{1}{2}G_d + G_u\right)\right)v_k, \tag{38a}$$

$$\left(\frac{1}{2}C_d + C_u + \frac{h}{2}\left(\frac{1}{2}G_d + G_u\right)\right)v_{k+1} = \left(\frac{1}{2}C_d + C_u - \frac{h}{2}\left(\frac{1}{2}G_d + G_l\right)\right). \tag{38b}$$

When these algorithms are applied to circuits where C is not diagonal, serious stability and accuracy problems may arise. In the example of Fig. 7, the Gauss–Seidel integration algorithm with a time step equal to 0.6 s computes the solution shown in Fig. 11, with oscillations that are not present in the accurate solution of the circuit equations shown in Fig. 8(a). In addition, these methods are not even consistent. That is, when the time step h is reduced to 0, the sequence of voltages computed according to (36) or to (37) does not converge to the solution of (33). In fact, since (36) is the same as the equation obtained by applying the Gauss–Jacobi algorithm to a circuit where only grounded capacitors are present, the effect of the floating capacitors is completely neglected (note that in (36), C_l and C_u do not appear). The Gauss–Seidel algorithm neglects C_u only, and hence is more accurate than the Gauss–Jacobi algorithm.

The modified symmetric Gauss–Seidel integration algorithm presented in the previous subsection has been proven to be accurate, stable, and even $\tilde{A}$ stable, but only for the particular classes of circuits characterized by G and C matrices with appropriate

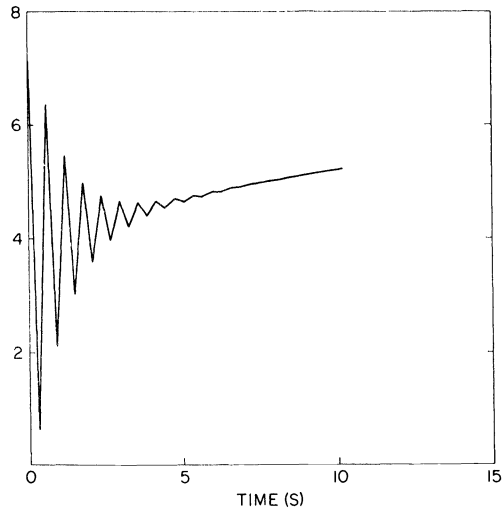


FIG. 11. *Waveform of node voltage* v_1 *computed with* 0.6-s *time-step.*

mathematical properties [36]. However, the modified symmetric Gauss–Seidel algorithm is not consistent in the general case either. On the other hand, since C_u is neglected in the first half-step while C_l is neglected in the second half-step, the effect of the floating capacitors on the solution of (33) is modeled more precisely than in the other methods, and better accuracy is obtained as a consequence [36].

Early timing simulators avoided the problem of analyzing floating capacitors by not allowing the user to include them in the circuit description. The effect of a floating capacitor may then be approximated by altering the values of the grounded capacitors at appropriate nodes in the circuit. If the operation of a circuit depends on a floating capacitor, a functional macromodel may be used (e.g., [9]–[11]) where the effect of the floating element is hidden from the relaxation iteration by special processing of the circuit fragment in which it is embedded, perhaps involving local matrix solution [29].

Another approach called the *Implicit-Implicit-Explicit* (IIE) method has been proposed [41] for circuits with floating capacitors. This method can be generalized and explained using (33). Let (33) be rearranged as

$$(C_d + C_l)\dot{v}^l + C_u\dot{v}^u = -Gv, \qquad v(0) = V \tag{39}$$

where $\dot{v}^l = \dot{v}^u = \dot{v}$. The Backward Euler method is used to approximate $\dot{v}^l$ and *Forward Euler is used to approximate* $\dot{v}^u$. Then (39) becomes

$$(C_d + C_l)(v_{k+1} - v_k) + C_u(v_k - v_{k-1}) = -hGv_{k+1}. \tag{40}$$

Applying the Gauss–Seidel integration scheme to (40) and rearranging terms, the IIE equations for the linear circuit described by (33) become

$$(C_d + C_l + h(G_d + G_l))v_{k+1} = -C_u(v_k - v_{k-1}) - hG_uv_k. \tag{41}$$

Note that the effects of both the lower triangular part and the upper triangular part of the capacitance matrix are now taken into account. However, to date the numerical properties of this scheme have only been published for a simple two-node linear test circuit [41], [42]. The method has been difficult to characterize rigorously since it involves information from two previous time points (t_k and t_{k-1}). Recently the

stability, consistency, and order of accuracy of this approach have been determined for the general case [43] and the method looks very promising. The IIE method is used in the SPLICE1 program for the analysis of floating capacitors [12].

F. Conclusions. Timing simulation algorithms are fast and rather accurate for the electrical simulation of MOS circuits with no tight feedback loops. However, several stability and accuracy problems hamper the use of timing simulation as a standard simulation tool. In particular, major drawbacks of timing simulation algorithms are

1) The selection of an appropriate step size is difficult. If a fixed step size is used, heuristics must be introduced to estimate the time constants of the circuit [10]. If a variable step size is used, the local truncation error must be estimated. In fact, the other technique used to control step size in standard circuit simulators, the so-called iteration count method [1], cannot be applied here since the relaxation techniques are not carried to convergence. Unfortunately, the local truncation error cannot be estimated accurately since the error in the voltage computed at time t_k by the timing simulation algorithms is the sum of the truncation error due to the integration method and of the error due to the inaccurate solution of the discretized nonlinear equations. These two errors can be of the same order and sometimes the latter component can even be larger than the truncation error.

2) The step size may be limited by stability considerations since timing simulation algorithms are A stable only in particular cases. This may force the use of small step sizes even though large step size may be possible from accuracy point of view.

3) With the exception of the IIE method, timing simulation algorithms are not even consistent when applied to the analysis of circuits containing floating capacitors. This means that their accuracy cannot be improved over a certain limit by further step-size reductions.

All of these problems stem from the fact that the relaxation methods are not carried to convergence. The fear that carrying the relaxation iteration to convergence would reduce the speed advantages of timing simulation prevented the adoption of the obvious remedy to this situation for a number of years. In the following sections, techniques and simulators based on convergent relaxation methods are introduced and shown to be highly competitive with standard circuit simulators for accuracy and with timing simulators for speed.

IV. Iterated timing analysis.

A. Introduction. Iterated Timing Analysis (ITA) [19] is a new form of electrical analysis which can be derived from timing analysis. This form of relaxation-based electrical analysis has shown promising results over a wide class of circuits, from large digital circuits to complex analog designs. The technique is accurate, fast for large digital circuits, and amenable to implementation on advanced computer architectures, such as vector and array processors [44]–[47] as well as data-flow machines [48], [49].

The starting point for a description of ITA is the circuit equation formulation of (2). The differential equations are converted to a set of nonlinear, algebraic difference equations (3) using a stiffly stable integration formula, and an iterative relaxation method (Gauss–Jacobi or Gauss–Seidel) is then used to solve them. However, unlike timing analysis where a single relaxation iteration is used per time point, in the ITA approach *the relaxation process is continued to convergence* at a time point.

Only one Newton–Raphson iteration is used to approximate the solution of each nodal equation per relaxation iteration and event-driven selective trace techniques may still be used to exploit latency, as for timing simulation. Thus the mathematical framework of ITA is the nonlinear Gauss–Seidel (Gauss–Jacobi)–Newton method presented in Section II.

B. Numerical properties of ITA. Since in ITA the nonlinear circuit equations are solved by an iterative method until satisfactory convergence is achieved, the numerical properties of the integration methods used to discretize the circuit equations are retained. Thus the stability and the accuracy problems typical of the timing simulation algorithms presented in Section III are not an issue. However, the basic question is whether the relaxation iteration will converge at each time point when solving the discretized circuit equations.

The conditions under which the relaxation iteration is guaranteed to converge were presented in Section II. Note that these conditions require the diagonal dominance of the Jacobian of the discretized nonlinear equations. Returning once again to (2), the circuit equations may be formulated as

$$C(v,u)\dot{v} + f(v,u) = 0, \qquad v(0) = V \tag{42}$$

where C: $\mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^{n\times n}$ is a symmetric diagonally dominant matrix-value function in which $-C_{ij}(v,u)$; $i \neq j$ is the total floating capacitance between nodes i and j, $C_{ii}(v,u)$ is the sum of the capacitances of all capacitors connected to node i, and f: $\mathbb{R}^l \times \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is a continuous function, each component of which represents the net current charging the capacitor at a node due to other conductive elements. If the capacitance matrix $C(v,u)$ is assumed to be symmetric and positive definite (and hence strictly diagonally dominant), as is the case if all the capacitors in the circuit are two-terminal elements and are positive for all values of v, it is intuitive to see that the Jacobian matrix of the discretized nonlinear circuit equations is diagonally dominant provided that the time step is small enough. In fact, the time step is acting as a scaling parameter that increases the role of the capacitance matrix in the Jacobian matrix when it is decreased. More formally, the convergence properties of ITA can be proven rather easily for circuit equations of the form of (42) where $C(v,u)$ is a matrix of real numbers, i.e., the capacitors present in the circuit are all linear. Then the discretized equations become

$$C(v_{k+1} - v_k) - h_{k+1} f(v_{k+1}, u_{k+1}) = 0 \tag{43}$$

where h_{k+1} is the time step selected at time t_k. The following strong Theorem has been proven in [50].

THEOREM IV-B-1. *There exists a time step $\hat{h}$ strictly positive such that for all $h_{k+1} \leqslant \hat{h}$ the nonlinear Gauss–Jacobi and the nonlinear Gauss–Seidel iteration applied to* (43) *converge to the solution of the discretized circuit equations independent of the initial guess.*

C. Implementation of ITA. In Theorem IV-B-1, the value of $\hat{h}$ can be quite small if the Jacobian of f is not diagonally dominant at the time point of interest and if the C matrix has large off-diagonal elements, i.e., when large floating capacitors and/or tight feedback loops are present in the circuit. Hence, such an iterative method would not appear well suited to the analysis of circuits with strong bilateral coupling. However, an ITA capability has been implemented in the SPLICE1 program [19], [51], and while strong bilateral coupling does increase simulation time, the correct solution is obtained even for analog circuits. With the event-driven selective trace scheduling as implemented in SPLICE1, less than a factor of two increase in CPU time has been observed compared with SPLICE1 timing simulation, for large digital circuits. For small tightly coupled MOS analog circuits, the ITA program may take even longer than SPICE2, as illustrated in the next section. However, for large integrated circuits such tight coupling

is local to a small block and the advantages obtained from circuit latency, as well as the ability to exploit parallel processing effectively, far outweigh the disadvantages.

The following algorithm illustrates the principal steps involved in ITA analysis for use on a conventional computer. Only the Gauss–Seidel form is shown here, but the Gauss–Jacobi form can be obtained as outlined in Section II. At each time at which one or more nodes are scheduled to be processed, two event lists, $E_A(t_n)$ and $E_B(t_n)$, are used to separate the nodes to be processed in successive iterations, k and $k+1$, of the Gauss–Seidel–Newton process.

Gauss–Seidel iteration:
put all nodes that are connected to independent sources in event list $E_A(0)$:

$t_n = 0$;
while ($t_n < TSTOP$){

  $k \leftarrow 0$;
  while (event list $E_A(t_n)$ is not empty){

    foreach (i **in** $E_A(t_n)$){
      obtain v_i^{k+1} from $g_i(v_1^{k+1},\cdots,v_i^{k+1},\cdots,v_N^k) = 0$ using a single Newton–Raphson step;

      if ($|v_i^{k+1} - v_i^k| \leqslant \varepsilon$; i.e. convergence is achieved){
        use LTE to determine the next time, t_s for processing node i;
        add node i to event list $E_A(t_s)$;

      }
      else{
        add node i to event list $E_B(t_n)$;
        add the fanout nodes of node i to event list $E_A(t_n)$ if they are not already on $E_A(t_n)$;

      }
    }

    $E_A(t_n) \leftarrow E_B(t_n)$; $E_B(t_n) \leftarrow$ empty;
    $k \leftarrow k+1$;
  }
  $t_n \leftarrow t_{n+1}$;
}

where t_n is the present time for processing and t_{n+1} is the next time in the time queue at which an event was scheduled. In this way, the "time step" is handled independently for each node.

This simplified algorithm does not illustrate how such issues as time-step reduction and local truncation-error estimation are handled. These and other important details of the algorithm are described elsewhere [53].

D. Circuit examples. Iterated Timing Analysis is an integral part of the SPLICE1.6 mixed-mode simulator [19], [51] and a number of example runs performed by this program are included below. SPLICE1.6 uses a Successive Over Relaxation (SOR)-Newton method [29], which defaults to Gauss–Seidel–Newton for solving the nonlinear

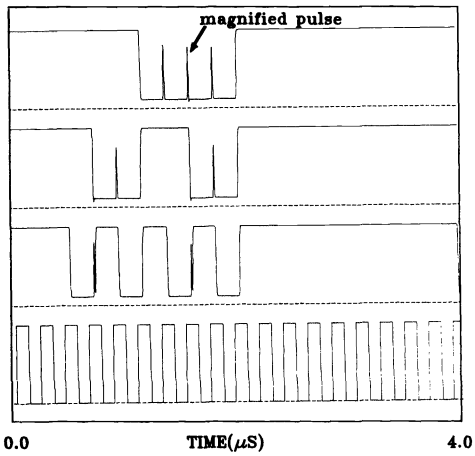
difference equations of (3). The program uses event-driven selective trace analysis to exploit latency [52]. A fundamental limitation of the present implementation of ITA, which reduces its overall effectiveness, is its simple time-step control mechanism. Another ITA program, SPLICE2 [53], is under development which overcomes this and other limitations of SPLICE1.

For large digital circuits, SPLICE1.6 is still 10–50 times faster than SPICE2 for the same output waveforms. The actual speedup factor depends on the nature of the digital circuit (highly pipelined, random logic, etc.) and the type of MOS technology in use (2-phase static, 4-phase dynamic, etc.). In each case, the circuit latency and strength of the bilateral coupling between circuit blocks determines the actual speedup. Table I contains results for the analysis of a large, random logic circuit. The corresponding output waveforms are shown in Fig. 12(a). Fig. 12(b) shows a comparison with SPICE2 for a small glitch in the waveform. Note that the program is significantly faster than SPICE2 for substantially the same waveform information. The precise tradeoff between accuracy and speed can be adjusted by varying the convergence criteria of both programs.

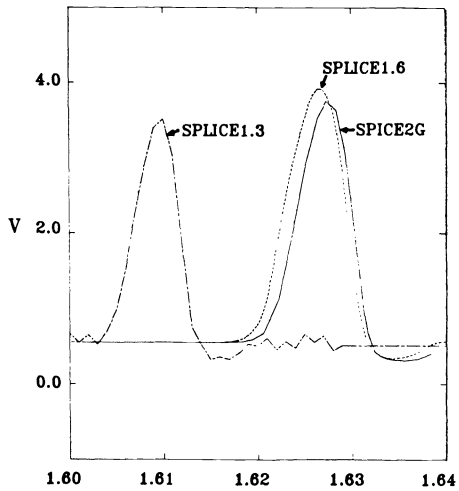
TABLE I

Comparisons of conventional circuit simulation, iterated timing analysis, and timing simulation for two example circuits

Circuit:	Encoder/decoder		Operational Amplifier	
MOSFETS:	1,326		15	
Nodes:	553		14	
	Time (s)	Memory (Kbyte)	Time (s)	Memory (Kbyte)
SPICE2G	115,840	2,420	59.8	29.0
SPLICE1.6	1,740	88.9	114.3	9.3
SPLICE1.3	789	64.4	–	–



(a)



(b)

FIG. 12. (a) *Selected waveforms from the encode/decode circuit obtained by* SPLICE1.6. (b) *Expanded view of a small pulse in the encode/decode circuit waveforms.*

A major drawback of standard timing simulators is their inability to handle floating elements accurately, in particular, floating capacitors. As shown in the previous section, this is most apparent when the value of the floating capacitor is large with respect to the grounded capacitors at each of its terminals. Strong feedback and high gain also present a difficult problem for a relaxation-based ITA program. Fig. 13(a) shows the schematic diagram of an MOS operational amplifier used as part of a phase-locked loop circuit [54]. The circuit was analyzed by both SPLICE1.6 and SPICE2 in a unity-gain configuration. Such circuits have always proved the most difficult even for conventional circuit simulators. Note the large capacitive feedback provided by the floating compensation capacitor. All transistors included parasitic capacitors C_{gs} and C_{gd}. The output waveforms for both SPLICE1.6 and SPICE2, for the same step input, are shown in Fig. 13(b). The only differences in the waveforms are at the beginning of the analysis and are due to slightly different initial conditions

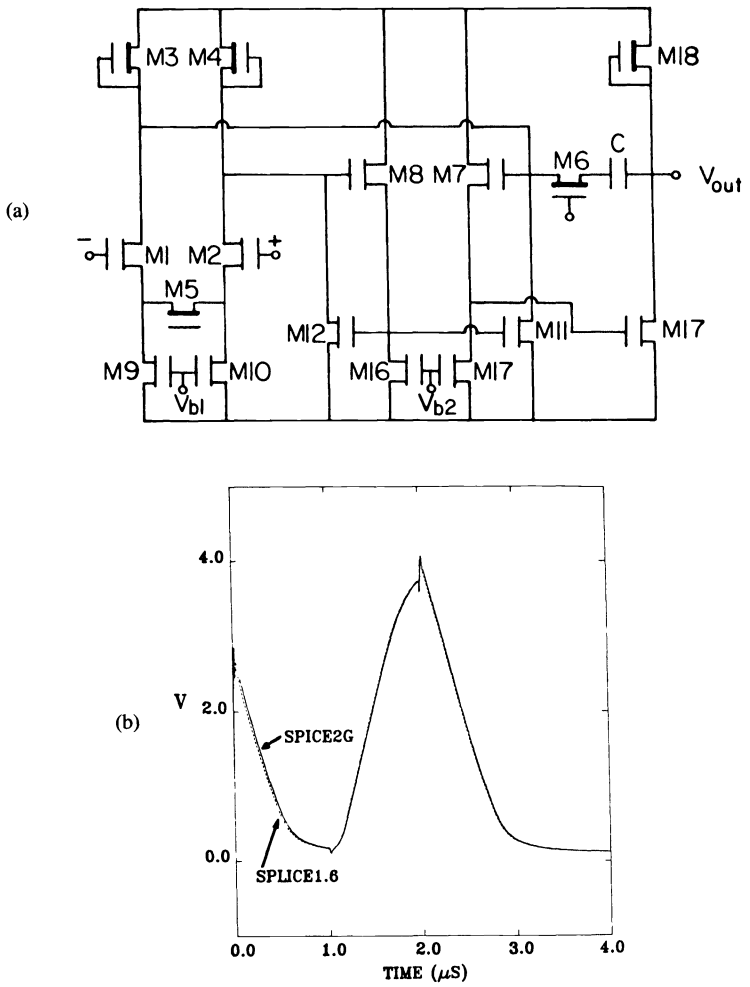


FIG. 13. (a) *Schematic diagram of operational amplifier circuit.* (b) *Response of operational amplifier to pulse input.*

assumed by each program. In this case, however, the prototype SPLICE1.6 program ran two times longer than SPICE2, as shown in Table I. While it is to be expected that such a worst-case circuit would reduce the performance of the program due to the strong capacitive feedback and high forward gain of the circuit, it is anticipated that this time difference will be reduced as the SPLICE1 program is developed further.

E. Conclusions. As previously illustrated, Iterated Timing Analysis has a great deal of potential for improving the performance of electrical simulation. Not only can this technique outperform standard circuit-simulation programs for the analysis of large circuits on standard computers, but it offers the possibility of a further substantial speedup when implemented on special-purpose hardware.

V. Waveform relaxation techniques.

A. Introduction. Both timing algorithms and iterated timing analysis are based on the application of relaxation techniques to the solution of circuit equations at the nonlinear algebraic equation level. As pointed out in Section II, there is a part missing in Fig. 3: a relaxation method at the differential equation level. While relaxation techniques in the linear and nonlinear algebraic case deal with vectors in $\mathbb{R}^n$, relaxation techniques at the differential equation level must deal with elements in function spaces, i.e., *waveforms*. Recently, a family of relaxation techniques applied to the differential equation level, called *Waveform Relaxation* (WR), has been proposed in [20], [21]. WR methods have been implemented in an experimental circuit simulator called RELAX [21] that has proven to be effective for the accurate analysis of some MOS digital circuits with more than an order of magnitude speed improvement over standard circuit simulators.

In this Section, the basic ideas of WR methods are reviewed and some of WR applications and extensions are presented. To begin, a simple example is used to illustrate the method, and then the general "Gauss–Seidel" algorithm in the WR family for MOS digital circuits is described. A more detailed and complete description of these techniques is available in [20], [21].

B. The waveform-relaxation Gauss–Seidel algorithm. Consider the first-order two-dimensional differential equation in: $x(t) \in \mathbb{R}^2$ on $t \in [0, T]$.

$$\dot{x}_1 = f_1(x_1, x_2, t), \qquad x_1(0) = x_{10}, \tag{44a}$$

$$\dot{x}_2 = f_2(x_1, x_2, t), \qquad x_2(0) = x_{20}. \tag{44b}$$

The basic idea of the "Gauss–Seidel" waveform-relaxation algorithm is to fix the waveform x_2: $[0, T] \to \mathbb{R}$ and solve (44a) as a one-dimensional differential equation in $x_1(\cdot)$. The solution thus obtained for x_1 can be substituted into (44b) which will then reduce to another first-order differential equation in one variable, x_2. Equation (44a) is then resolved using the new solution for $x_2(t)$, and the procedure is repeated.

In this fashion, an iterative algorithm has been constructed. It replaces the problem of solving a differential equation in two variables by one of solving a sequence of differential equations in one variable. As described earlier, the waveform relaxation algorithm can be seen as an analogue of the Gauss–Seidel technique for solving nonlinear algebraic equations. Here, however, the unknowns are waveforms (elements of a function space), rather than real variables. In this sense, the algorithm is a technique for time-domain decoupling of differential equations.

WR algorithms applied to circuits can be formulated in a number of ways. A "Gauss–Seidel" WR algorithm for MOS circuits will be considered in the following

analysis. Recall that, according to (2), the circuit equations are formulated as

$$C(v,u)\dot{v} + f(v,u) = 0, \qquad v(0) = V \tag{45}$$

where C: $\mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^{n\times n}$ is a symmetric diagonally dominant matrix-value function in which $-C_{ij}(v,u)$; $i \neq j$ is the total floating capacitance between nodes i and j, $C_{ii}(v,u)$ is the sum of the capacitances of all capacitors connected to node i, and f: $\mathbb{R}^l \times \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is a continuous function, each component of which represents the net current charging the capacitor at each node due to the pass transistors, the other conductive elements, and the controlled current sources.

Algorithm V-B-1 (WR *Gauss–Seidel algorithm for solving* (45)):

Comment: The superscript k denotes the iteration count, the subscript i denotes the component index of a vector, and ε is a small positive number.

$k \leftarrow 0$;
guess waveform $v^0(t)$; $t \in [0,T]$ such that $v^0(0) = V$
  (for example, set $v^0(t) = V$, $t \in [0,T]$);

repeat {
  $k \leftarrow k+1$

  foreach (i **in** N) {
    solve

$$\sum_{j=1}^{i} C_{ij}(v_1^k,\cdots,v_i^k,v_{i+1}^{k-1},\cdots,v_N^{k-1},u)\dot{v}_j^k + \sum_{j=i+1}^{n} C_{ij}(v_i^k,\cdots,v_i^k,v_{i+1}^{k-1},\cdots,v_N^{k-1},u)\dot{v}_j^{k-1} + f_i(v_1^k,\cdots,v_i^k,v_{i+1}^{k-1},\cdots,v_N^{k-1},u) = 0$$

    for ($v_i^k(t)$; $t \in [0,T]$), with the initial condition
    $v_i^k(0) = V_i$.
    }
  }
until ($\max_{1\leqslant i\leqslant n}\max_{t\in[0,T]}|v_i^k(t) - v_i^{k-1}(t)| \leqslant \varepsilon$)
that is, until the iteration converges.

Note that (45) has only one unknown variable v_i^k. The variables $v_{i+1}^{k-1},\cdots,v_N^{k-1}$ are known from the previous iteration and the variables $v_1^k,\cdots,v_{i-1}^k$ have already been computed. Note also that the Gauss–Jacobi version of the WR algorithm presented earlier can be obtained simply by replacing the **foreach** statement with the **forall** statement and adjusting the iteration indices in the same way as can be done in the Gauss–Jacobi version of ITA. In the Gauss–Jacobi case, the computation can be performed in parallel, and hence it is more suitable for implementation on special-purpose hardware.

As an example of how Algorithm V-B-1 can be applied, consider the MOS circuit shown in Fig. 14. For the sake of simplicity it is assumed that all capacitors are linear. Hence the dynamical behavior of the circuit can be described as follows:

$$\begin{aligned}(c_1+c_2+c_3)\dot{v}_1 - i_1(v_1) + i_2(v_1,u_1) + i_3(v_1,u_2,v_2) - c_1\dot{u}_1 - c_3\dot{u}_2 &= 0,\\ (c_4+c_5+c_6)\dot{v}_2 - c_6(\dot{v}_3) - i_3(v_1,u_2,v_2) - c_4\dot{u}_2 &= 0,\\ (c_6-c_7)\dot{v}_3 - c_6\dot{v}_2 - i_4(v_3) + i_5(v_3,v_2) &= 0.\end{aligned} \tag{46}$$

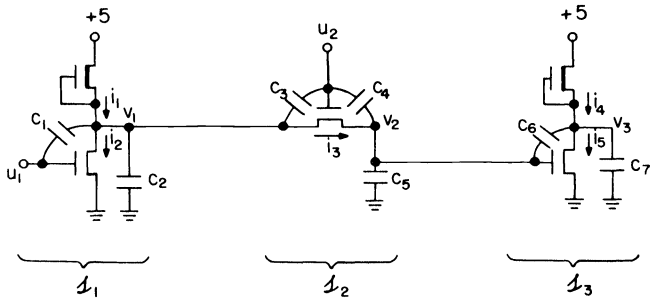


FIG. 14. *Circuit example for Gauss–Seidel waveform relaxation algorithm.*

Applying the WR procedure to (46) the kth iteration corresponds to solving the following equations:

$$
\begin{aligned}
(c_1+c_2+c_3)\dot{v}_1^k - i_1(v_1^k) + i_2(v_1^k, u_1) + i_3(v_1^k, u_2, v_2^{k-1}) - c_1\dot{u}_1 - c_3\dot{u}_2 = 0,\\
(c_4+c_5+c_6)\dot{v}_2^k - c_6\dot{v}_3^{k-1} - i_3(v_1^k, u_2, v_2^k) - c_4\dot{u}_2 = 0,\\
(c_6+c_7)\dot{v}_3^k - c_6\dot{v}_2^k - i_4(v_3^k) + i_5(v_3^k, v_2^k) = 0.
\end{aligned}
\tag{47}
$$

The circuit interpretation of (47) is shown in Fig. 15.

If the original circuit in Fig. 14 consists of 3 subcircuits s_1, s_2, and s_3, then the decomposed subcircuits $\tilde{s}_1$, $\tilde{s}_2$, and $\tilde{s}_3$ together with additional components to approximate the loading effects due to the rest of the circuit. Hence, the WR procedure for analyzing the circuit in Fig. 14 can be described in circuit terms as follows:

$k \leftarrow 0$;
make an initial guess of $v_2^0(t), v_3^0(t)$; $t \in [0, T]$;
repeat {
  $k \leftarrow kp$;

  analyze s_1 for its output waveform $v_1^k(\cdot)$ by
    approximating the loading effect due to s_2;

  analyze s_2 for its output waveform $v_2^k(\cdot)$ by
    using $v_1^k(\cdot)$ as its input and approximating
    the loading effect due to s_3;

  analyze s_3 for its output waveform $v_3^k(\cdot)$ by
    using $v_2^k(\cdot)$ as its input;
  }
until (the difference between $\{(v_1^k(t), v_2^k(t), v_3^k(t);\ t \in [0, T]\}$ and $\{v_1^{k-1}(t), v_2^{k-1}(t), v_3^{k-1}(t);\ t \in [0, T]\}$ is sufficiently small)

From the previous procedure and example it can be seen that each component of the decomposition is a dynamical subcircuit which is processed for the entire time interval $[0, T]$ in a fixed order. When each subcircuit is being processed, the iterations (coupling or loads) from the rest of the circuit are approximated by using the information obtained from the most recent iteration. The iteration is carried out until satisfactory convergence of all waveforms is detected. It can be shown that for the MOS circuit in Fig. 14 the sequence of waveforms generated by the WR procedure *will always converge to the correct waveform independent of the initial guess provided that* c_2, c_5, and

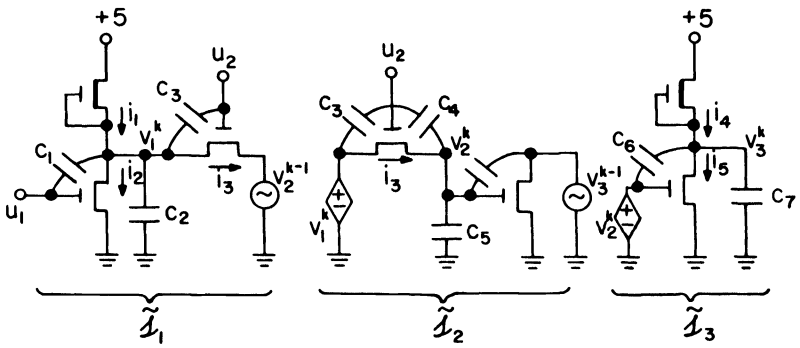


FIG. 15. *Circuit interpretation of Gauss–Seidel waveform relaxation applied to the circuit of Fig.* 14.

c_7 *are not zero.* In [20], [50], a strong convergence result was proved that can be applied to Algorithm V-B-1 as follows.

THEOREM V-B-1. *Assume that:*

1) *the charge-voltage characteristic of each capacitor, the current–voltage characteristic of each conductor, and the drain-current characteristic of each MOS device are Lipschitz continuous with respect to their controlling variables;*

2) $C_{min} > 0$ *and* $C_{max} < \infty$ *where* $C_{min} \in \mathbb{R}$ *is the minimum value of all grounded capacitors at any permissible value of node voltage, and* $C_{max} \in \mathbb{R}$ *is the maximum value of all floating capacitors at any permissible value of node voltages; and*

3) *the current through any controlled conductor (e.g., the drain current of an MOS device) is uniformly bounded throughout the relaxation process.*

Then, for any given set of initial conditions and any given piecewise continuous input $u(\cdot)$, *Algorithm V-B-1 generates a converging sequence of iterated solutions whose limit satisfies the circuit equations and the given initial conditions.*

C. Waveform relaxation in RELAX. The WR "Gauss–Seidel" algorithm was implemented in an experimental circuit simulator, RELAX. Actually, RELAX implemented a modified version of the WR algorithm described previously. These modifications were introduced to improve the speed of convergence of the algorithm and exploit the structure of the class of circuits to be analyzed, i.e., MOS digital circuits. These modifications are as follows.

1) Rather than having strictly one unknown per each component of the decomposition as stated in Algorithm V-B-1, RELAX allows each decomposed subcircuit to have more than one unknown. This corresponds to a block relaxation method that can be proven to have similar convergence properties. In fact, the analysis of the circuit is decomposed into the analysis of subcircuits each of which corresponds to a physical subcircuit that is built into the program and called by the user as a unit, such as a NOR or a NAND.

2) Each decomposed subcircuit is processed by using standard circuit-analysis techniques. The Backward Euler integration method with variable time steps is used to discretize the differential equations associated with the subcircuit, and the Newton–Raphson method is used to solve the nonlinear algebraic equations resulting from the discretization. Since the number of unknowns associated with a subcircuit is usually small, the linear equation solver used by the Newton–Raphson method is implemented by using standard full-matrix techniques rather than using sparse-matrix techniques. Note that in RELAX each subcircuit is analyzed independently from $t = 0$ to $t = T$, using its own time-step sequence, controlled by the integration method,

whereas in a standard circuit simulator the entire circuit is analyzed from $t = 0$ to $t = T$ using only one common time-step sequence. In RELAX, the time-step sequence of one subcircuit is usually different from the others, but contains, in general, a smaller number of time steps than that used in a standard circuit simulator for analyzing the same circuit.

3) The order according to which each subcircuit is processed is determined in RELAX prior to starting the iteration by a subroutine called "scheduler." Although, according to Theorem V-B-1, scheduling is not necessary to guarantee convergence of the iteration, it does have an impact on the speed of convergence as is the case for the relaxation methods at the linear- and nonlinear-equation levels. Assume now that the circuit consists of unidirectional subcircuits with no feedback path. Exactly as for the other relaxation methods introduced before, if the subcircuits are processed according to the flow of signals in the circuit, the algorithm used in RELAX will converge in just two iterations (actually the second iteration is needed only to verify that convergence has been obtained). For MOS digital circuits which contain almost unidirectional subcircuits, it is intuitive that convergence of the WR procedure will be achieved more rapidly if the subcircuits are processed according to the flow of signals in the circuit. The scheduler traces the flow of signals through the circuit and generates a static order for the processing of subcircuits. To be able to trace the flow of signals, the scheduler requires the user to specify the flow of signals through each subcircuit by partitioning the terminal of the subcircuit into input and output terminals. This is needed since in RELAX the basic unit is a subcircuit. In general, a designer can easily specify what the flow of the signals is intended to be even in a subcircuit which is not unidirectional such as a transmission gate or a subcircuit containing floating capacitors between its input and output terminals. The analysis algorithm in RELAX will indeed take into account the bidirectional effects correctly.

4) The first iteration in RELAX is carried out by assuming that there is no loading effect due to fanouts. The "standard" WR procedure actually begin at the second iteration in RELAX. Hence, strictly speaking, the first iteration in RELAX is used to generate a good initial guess for the actual WR procedure.

D. Speed-up techniques. In addition to the previous modifications, RELAX incorporates two bypass techniques to speed up the process of analyzing a subcircuit. The key idea is once more to bypass the analysis of a subcircuit for certain time intervals without losing accuracy by exploiting the information obtained from previous time points and/or from previous iterations.

The two techniques used in RELAX are presented here by showing their application for the analysis of the subcircuit s_1 of the circuit shown in Fig. 16, which is a schematic diagram of the circuit in Fig. 14. The output voltages of s_1 and s_2 at the kth iteration are denoted by v_1^k and v_2^k, respectively.

The first technique is based on the latency of s_1 and is similar to the technique described in Section III. According to (47), s_1 is analyzed in the first iteration with no loading effect from s_2. After it has been analyzed for a few time points, its output voltage v_1^1 is found to be almost constant with time, i.e., $v_1^1(0.01) \approx 0$ (see Fig. 16(b)). Since the input of s_1, i.e. u_1 is also constant during the interval [0.01, 1.9], the subcircuit s_1 is said to be "latent" in the first iteration during the interval [0.01, 1.9], and its analysis during this interval is bypassed. From Fig. 16(b), s_1 is latent again in the interval [2.15, 3]. Note that, according to (47), the check for latency of s_1 after the first iteration will include u_2 and v_2 as well as u_1 since they can affect the value of v_1. For most digital circuits, the latency intervals of a subcircuit usually cover a large portion of

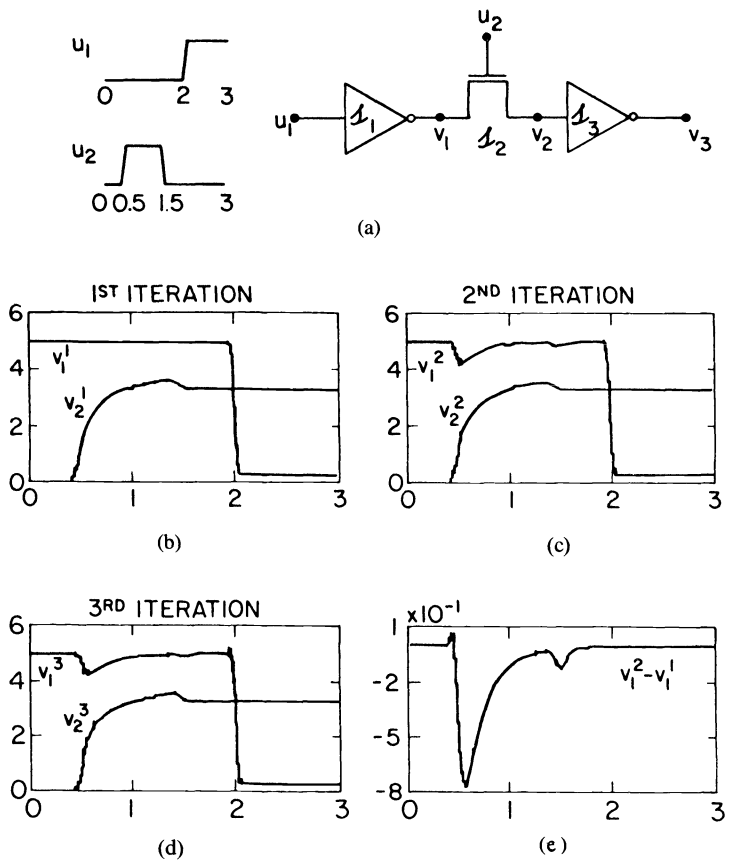


FIG. 16. (a) *Schematic diagram of the circuit of Fig.* 14. (b), (c), *and* (d): *The waveforms of the two node voltages at the first, second and third waveform relaxation iteration, respectively.* (e). *The difference of the waveforms between the first and second iteration.*

the entire simulation time interval $[0, T]$ and hence the implementation of this technique can provide a considerable saving of computing time as shown in Table II.

The second technique is based on the *partial convergence* of a waveform during the previous two iterations. This technique is introduced by using the example of Fig. 16. After the first two iterations, we observe that the values of v_1^1 and v_1^2 during the interval [1.7, 3.0] do not differ significantly (see Fig. 16(b), (c)), i.e., the sequence of waveforms of v_1 seem to converge in this interval after two iterations. In the third iteration, shown in Fig. 16(d), s_1 is analyzed from $t = 0$ to $t = 1.8$ and v_1^3 (1.8) is found to be almost the same as v_1^2 (1.8). Moreover, during the interval [1.8, 3], the value of v_2^2 which affects the value of v_1^3 also does not differ significantly from the values of v_2^1 (which affects the value of v_1^2). Hence the value of v_1^3 during the interval [1.8, 3] should remain the same as v_1^2 and the analysis of s_1 during this interval in the third iteration will be bypassed. This technique can provide a considerable saving of computing time as shown in Table II since the intervals of convergence can cover a large portion of the entire simulation time interval $[0, T]$, especially in the last few iterations. Note that the subcircuit need not be latent during the intervals of convergence although overlapping of these intervals with the latency intervals is possible.

TABLE II

Comparison of CPU *times used by* RELAX *for analyzing the circuit of Fig.* 0 *with and without the latency and the partial waveform convergence techniques*

(Case 1: Without the latency and the partial waveform convergence techniques.
Case 2: With only the latency technique.
Case 3: With only the partial waveform convergence technique.
Case 4: With both the latency and the partial waveform convergence techniques.)

	CPU time (seconds)			
Iteration #	case 1	case 2	case 3	case 4
1	0.363	0.255	0.360	0.258
2	0.824	0.704	0.814	0.695
3	0.821	0.705	0.242	0.238
4	0.828	0.704	0.151	0.104
5	0.832	0.695	0.097	0.016
Total	3.668	3.063	1.664	1.311

E. Some extensions of waveform-relaxation techniques: RELAX2. RELAX is written in FORTRAN77. It can handle MOS digital circuits containing NOR gates, NAND gates, transmission gates, multiplexers (or banks of transmission gates whose outputs are connected together), super buffers, and cross-coupled NOR gates (or RS flip-flops). It uses the Shichman–Hodges model [33] for the MOS device. All the computations were performed in double precision and the results were also stored in double precision. Although the RELAX code is rather small, approximately 4000 FORTRAN lines, it requires a large amount of storage for the waveforms, especially when large circuits are analyzed. For an MOS circuit containing 1000 nodes with 100 analysis time points per node, the waveform storage requires approximately $3 \times 1000 \times 1000$ floating point numbers (corresponding to 2.4 Mbytes if each number is stored in 64 bits).

A new version of RELAX, RELAX2 [55], written in C, has extended and made waveform-relaxation algorithms more practical. The first important extension consists of allowing arbitrary subcircuits to be defined by the user, as was provided in the original SPLICE1 program. These subcircuits are analyzed by a subroutine patterned after SPICE, while the original RELAX used "hard-wired" subcircuit analyzers made possible by the fixed structure of the subcircuits allowed by the program. Thus RELAX2 is slower than RELAX, but still maintains a definite advantage over standard circuit simulators, approximately one order of magnitude speed improvement.

The second extension consists of allowing the decomposition of the time interval of interest for the time-domain simulation into subintervals, called *windows* [56].

Digital circuits can be broken up into two very broad classes: circuits with logic feedback loops (finite-state machines, asynchronous circuits, digital oscillators) and circuits without logic feedback loops (most combinational logic, programmable logic arrays). Experience simulating MOS digital circuits using RELAX2 shows that most

MOS digital circuits without logic feedback loops converge in less than ten iterations. However, circuits with logic feedback loops may take many more iterations to converge, and the number of iterations required is proportional to the length of the simulation interval.

This suggests that the interval of simulation should be broken into "windows", $[0, T_1]$, $[T_1, T_2], \cdots, [T_{n-1}, T_n]$, so that the relaxation will converge rapidly in each window. Waveform relaxation is applied to the first window, $[0, T_1]$ and the values of the node voltages at T_1 are then used as initial conditions for the analysis of the second window. This procedure is repeated until all windows have been analyzed.

Consider the analysis of the cross-coupled NAND gate circuit shown in Fig. 17 with the "window" approach provided by RELAX2; convergence is quite rapid (see Table III). There is a trade-off, however. As the window size gets smaller some of the advantages of waveform relaxation are lost. One cannot take advantage of a digital circuit's natural latency over the entire waveform, but only within that window. The scheduling overhead increases when the windows become smaller, as each subcircuit must be scheduled once for each window, and if the windows are made very small, time steps chosen to calculate the waveforms will be limited by the window size rather than by the local truncation error and unnecessary calculations will be performed.

Breaking the interval simulation into windows also has the advantage of reducing the memory requirements of WR, since these are proportional to the number of time points used by the numerical integration algorithm used to solve the subcircuit equations.

Other extensions to WR included in RELAX2 involve the approximate solution of the subcircuit equations [56], [57]. When using iterative decomposition methods for solving systems of nonlinear equations, it may be possible to reduce the calculations required by not solving the decomposed nonlinear equations exactly at each iteration.

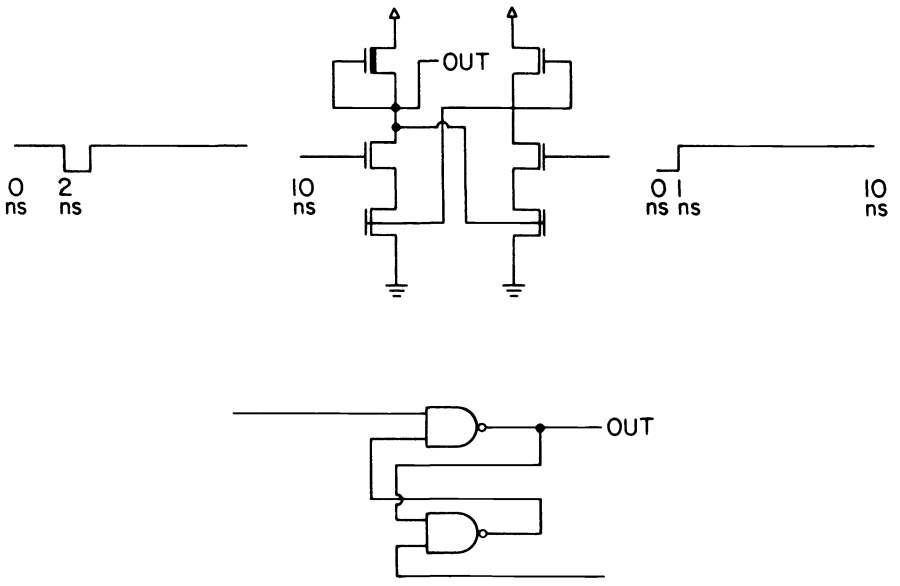


FIG. 17. *A two cross-coupled* NAND-*gate circuit.*

TABLE III
Windowing experiments for the cross-coupled NAND *gates performed on a* VAX11/780 *Running Berkeley* UNIX4.1*c*

CROSS-COUPLED NAND GATES			
# Windows	# Timepoints	Max # Iterations	CPU time*
SPICE	–	–	21.75
1	50	>100	--
2	50	10	13.59
4	48	4	4.48
8	59	4	5.38
16	69	4	5.91

In some cases the convergence of the algorithm is not affected by the inaccurate solutions. An example is the Gauss–Seidel–Newton method [27] described in Section II. In the WR case, simpler approximate methods for calculating the node waveforms are used for the first few iterations; more complex and more exact methods are used for the last few iterations.

One way of simplifying the calculation of the node-voltage waveforms is to use a simple model for the MOS devices and then to switch to the more detailed model as the waveforms approach convergence. The simple device model used in RELAX2 is a resistor in series with a switch, where the size of the resistance is scaled with the device size. Using such a model in the calculation of waveforms is not straightforward, because the equations describing the model can not be solved easily using the Newton–Raphson method. The Newton–Raphson method will often oscillate about the point where the simple model's switch changes state. One solution to this problem is not to carry out the Newton–Raphson method to convergence, but to do only one iteration. The result is that the calculation of the waveforms using the simple model is quite fast, but only approximate, even if the simple model is assumed to be correct. The results obtained using the simple model and then changing to the more detailed model have been disappointing so far, as demonstrated in the examples of Table IV. In circuits without logic feedback, the simple model did not provide a better guess for the waveforms than one iteration using more complex models. It is possible that the addition of another term to the simple model, to make it more smooth, may help. Then the Newton–Raphson algorithm can be used and achieve the accuracy required to produce a useful first guess for the iterations using the more detailed models.

Another approach to simplifying the calculations performed in the first few iterations of the WR algorithm is to allow the numerical integration algorithm, which is used to solve for the node waveforms of the decomposed subcircuits, to use a larger local truncation error. Here, unlike changing the device models, it is possible to increase the accuracy of the calculation of the node waveforms at each iteration by tightening the local truncation error limit. In the case of RELAX, since most circuits converge in about 5 iterations, a local truncation error was chosen that is about 3 times larger than

TABLE IV
Experiments with variable accuracy model and variable local truncation error (LTE)

TEST CIRCUITS						
Method	Shift Cell		Two Phase Clk		Memory Cell	
	# Iter	Time	# Iter	Time	# Iter	Time
SPICE	–	12.52	–	43.13	--	13.63
RELAX2	4	2.10	4	5.47	4	2.98
Simple model	4	3.20	4	8.75	4	4.45
Simple model only	4	0.49	4	0.69	4	0.88
LTE	5	1.24	3	3.81	4	2.71

the local truncation error that would be chosen to calculate the waveforms for the final answer. Then after each iteration the local truncation error is multiplied by 0.7. The results from this approach are shown in the last row of Table IV.

A key question to be asked is whether the convergence of WR algorithms is affected by these approximation techniques. Note that Theorem V-B-1 assumes that the solutions of the differential equations are computed exactly. However, the convergence of the WR algorithm to the exact solution of the original differential equations has been proven provided the accuracy of the integration is increased while the WR iterations are converging [56]. The framework necessary to prove this result is the one of nonstationary WR algorithms.

Algorithm V-B-1 is a stationary algorithm in the sense that the iteration process is performed with the same set of equations. Nonstationary WR algorithms are characterized by the fact that the equations describing the system at each iteration can change from one iteration to the other. Note that the approximations computed by the integration methods can be viewed as the exact solutions of perturbed differential equations. Thus the iteration equations of the WR algorithms can be seen as changing from iteration to iteration. The convergence theorem proven in [50], [56], assumes that the accuracy of the integration methods is controllable and that in the limit, the exact solution of a differential equation is achieved. Fortunately, this property is obviously satisfied when using simpler models in the first iterations to switch to more complex and accurate models at the last iterations. In addition, it is known that consistent integration methods can be made as accurate as desired by controlling parameters such as the local truncation error. Thus the speed-up techniques presented here are theoretically sound.

F. Conclusions. Waveform-relaxation methods have been proven to be effective decomposition methods for the analysis of large-scale MOS circuits. In particular, the methods have guaranteed convergence properties. Since WR algorithms are quite new, more research is needed to characterize completely the trade-offs involved in the choice of a particular method in the class (e.g., Gauss–Jacobi versus Gauss–Seidel). In addition, an accurate comparison between iterated timing analysis and waveform relaxation must be carried out.

It is clear that WR methods are quite suitable for implementation on a parallel or pipeline architecture since they allow different subcircuits to be analyzed concurrently on different processors.

In addition, WR algorithms have recently been extended to piecewise linear circuit analysis [57] and to other fields such as electrophoresis process simulation [58].

VI. Summary and directions for future work.

A. Introduction. Relaxation-based simulation techniques have been used for the analysis of electronic circuits in many ways. Timing simulators were the first relaxation-based electrical simulators to gain widespread use and are still being used successfully in many companies today. Unfortunately, since only a single sweep of a relaxation method is used to approximate the solution of the set of nonlinear algebraic equations obtained at a time point, these simulators suffer from severe accuracy problems when used to analyze circuits containing tight feedback loops or floating circuit elements. In Section III we presented an analysis of the numerical techniques used in timing simulation, and described a number of algorithms which can be used to improve the accuracy of timing simulators. In particular, methods suited to the analysis of floating capacitors have been described.

While timing simulators will continue to be used for the analysis of circuits where constrained circuit design methods, such as cell-based approaches, limit the likelihood of simulation errors, they cannot be used for the analysis of complex digital and analog circuits where feedback effects are significant. A new relaxation-based approach, called iterated timing analysis, can be used for the analysis of these circuits. We have described the basic algorithms used for ITA and their associated numerical properties in Section IV. This approach provides accurate simulation results while still achieving a substantial speed improvement over conventional circuit-analysis techniques. As shown in Section IV, ITA has also been used successfully for the analysis of tightly coupled analog circuits containing a number of floating capacitors.

Both of the aforementioned techniques apply relaxation methods to the solution of a set of nonlinear algebraic equations. In contrast to these techniques, the waveform-relaxation method uses a relaxation approach at the differential equation level. In Section V we have shown that waveform relaxation has guaranteed convergence properties for a wide class of electrical circuits, and has performed over one order of magnitude faster than standard circuit simulators on a number of test circuits while maintaining the same, or even better, accuracy. A number of improvements and extensions to the basic waveform-relaxation method were also presented.

On conventional computers, the speed advantage of relaxation-based analysis over matrix-based techniques can vary from a slight slow-down, for small tightly coupled analog circuits, to a maximum of about two orders of magnitude speedup, for large semi-static digital circuits.

B. Special-purpose hardware. The use of special-purpose computer instructions for sparse-matrix solution [14] and the use of vector computers, such as the CRAY-1 [17], can improve the speed of conventional circuit simulators by about an order of magnitude over their nonoptimized versions on the same machine. In the latter case, the speedup is limited by the *gather/scatter* problem [46] associated with arranging the data so that effective parallel computation can be performed. Unfortunately, the irregular structure of a circuit sparse matrix is the limiting factor here.

If relaxation techniques are used to replace these direct methods, the solution of each node equation is effectively *decoupled* from the others. While such decoupled-analysis techniques would be suitable for use on a vector computer, it seems that other architectures, in particular *data-flow* computers and related dependency-driven approaches [48], [49], [59]–[62], will allow the decoupling to be exploited more effectively. A straight-forward approach to the implementation of an electrical relaxation simulator on such a computer would be to allocate a separate processor for the solution of each decoupled-node equation. For an ITA algorithm, the calculation performed by each processor would be a single Newton–Raphson step on a nonlinear algebraic equation in one unknown. In the case of WR, each calculation would involve the computation of a partial waveform, or set of waveforms, on the processor. While the performance of a practical multiprocessor depends on many factors, a simplified analysis is presented here to illustrate the potential savings of such a machine.

For a circuit containing N nodes with M nodes actively changing at any time, $M \leqslant N$, the total time spent solving the independent node equations on a serial computer is approximately

$$T_s(M) \propto Mt_s \tag{48}$$

where t_s is the time required to solve the single-node equation in either scheme.

Consider a multiprocessor using a single- or multiple-stage shuffle network [61], [62] with an element cycle time of t_c and a latency proportional to $k\log(P)$, where P is the number of ports in the shuffle network and k is a constant. For now assume $P > M$; then the total analysis time on such a network is approximately

$$T_p(P) \propto t_s + t_c k \log(P). \tag{49}$$

Equation (49) is in fact a worst-case figure because it assumes all communication is on the critical path of the computation and that there is no pipelining of requests. The speed-up factor for the parallel computation is then

$$\frac{T_s}{T_p} \propto \frac{Mt_s}{t_s + t_c k \log(P)}. \tag{50}$$

If k is from 1 to 3 [63]; if $M = P$; if $t_c \approx t_s$, the speed-up becomes approximately $M/\log M$. However, if the equation solution time is larger than the network cycle time, or if a better than random placement of the circuit nodes on the network can be performed, then the speed-up factor will be closer to M. Note that it will generally be true in practice that $M > P$. In that case, more than one node will be allocated to each processor. Techniques for the implementation of "virtual processors" can be used to solve this problem [64], and scheduling algorithms can be used to allocate the nodes to processors in such a way that network loading is uniform.

It is clear that relaxation-based algorithms for electrical simulation are well suited to the use of special-purpose hardware. Future work in this area includes the investigation of the best match of hardware and algorithms, and the investigation of optimal techniques for simulation time advancement in a parallel computational environment.

Acknowledgments. The authors wish to thank the many talented graduate students with whom it has been their pleasure to work. M.-Y. Hsueh, J. E. Kleckner, J. D. Crawford, and J. Straus participated in the early development of the MOTIS-C timing simulator and SPLICE1 mixed-mode simulator. N. DeMicheli contributed in the development of new algorithms for timing simulation. E. Lelarasmee played a key role in the development of the waveform-relaxation techniques and the RELAX simulator. J. Kaye extended WR algorithms to the analysis of piecewise linear circuits. J. E. Kleckner and R. Saleh implemented the iterated timing analysis algorithm in SPLICE1.6. J. White wrote the RELAX2 program.

A. Ruehli participated in the early development of waveform relaxation. The authors also thank G. D. Hachtel for helpful discussions and D. O. Pederson for his continuous support, encouragement, and inspiration.

Many people have helped during the preparation of this paper. In particular, the authors thank N. DeMicheli, R. Saleh, and J. White for their assistance in the preparation of examples and critical reading of the manuscript. They would also especially like to thank J. T. Deutsch and J. E. Kleckner for helpful discussions. They would also like to thank Digital Equipment Corporation, IBM, Harris Semiconductors, Hewlett-Packard, and Tektronix for their support throughout the course of this work.

REFERENCES

[1] W. NAGEL, "SPICE2, A computer program to simulate semiconductor circuits," Univ. of California, Berkeley, Memo No. ERL-M520, May 1975.

[2] "Advanced statistical analysis program (ASTAP)," IBM Corp. Data Proc. Div., White Plains, NY, Pub. No. SH20-1118-0.

[3] A. SANGIOVANNI-VINCENTELLI, "Circuit simulation," in *Computer Design Aids for VLSI Circuits*, P. Antognetti, D. O. Pederson, and H. De Man, Eds. Groningen, The Netherlands: Sijthoff and Noordhoff, 1981, pp. 19–113.
[4] F. JENKINS, "ILOGS: User's manual," Simutec, 1982.
[5] S. A. SZYGENDA AND E. W. THOMPSON, "Digital logic simulation in a time-based, table-driven environment. Part 1. Design verification," *Comput.*, March 1975, pp. 24–36.
[6] R. E. BRYANT, "An algorithm for MOS logic simulation," *LAMBDA*, 4th quarter, pp. 46–53, 1980.
[7] C. M. BAKER AND C. TERMAN, "Tools for verifying integrated circuit designs," *LAMBDA*, 4th quarter 1980.
[8] M. H. HEYDEMANN, G. D. HACHTEL, AND M. LIGHTNER, "Implementation issues in multiple delay switch level simulation," in *Proc. 1982 Int. Conf. Circ. Comp.*, pp. 46–52, Sept. 1982.
[9] B. R. CHAWLA, H. K. GUMMEL, AND P. KOZAK, "MOTIS-an MOS timing simulator," *IEEE Trans Circuits Syst.*, vol. CAS-22, pp. 901–909, Dec. 1975.
[10] S. P. FAN, M. Y. HSUEH, A. R. NEWTON, AND D. O. PEDERSON, "MOTIS-C A new circuit simulator for MOS LSI circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Apr. 1977.
[11] N. TANABE, H. NAKAMURA, AND K. KAWAKITA, "MOSTAP: An MOS circuit simulator for LSI," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1035–1039, Apr. 1980.
[12] A. R. NEWTON, "Timing, logic and mixed-mode simulation for large MOS integrated circuits" in *Computer Design Aids for VLSI Circuits*, P. Antognetti, D. O. Pederson, and H. De Man, Eds. Groningen, The Netherlands: Sijithoff and Noordhoff, 1981, pp. 175–240.
[13] J. L. BURNS, A. R. NEWTON, AND D. O. PEDERSON, "Active device table look-up models for circuit simulation," in *Proc 1983 Int. Symp. Circuits Syst.*, May 1983.
[14] E. COHEN, "Performance limits of integrated circuit simulation on a dedicated minicomputer system," ERL Memo. UCB/ERL M81/29, May 22, 1981.
[15] A. SANGIOVANNI-VINCENTELLI, L. K. CHEN, AND L. O. CHUA, "A new tearing approach-node tearing nodal analysis," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 143–147, 1977.
[16] P. YANG, I. N. HAJJ, AND T. N. TRICK, "SLATE: A circuit simulation program with latency exploitation and node tearing," in *Proc. IEEE Int. Conf. Circuits Comput.*, Oct. 1980.
[17] A. VLADIMIRESCU AND D. O. PEDERSON, "Performance limits of the CLASSIE circuit simulation program," in *Proc. Int. Symp. Circuits Syst.*, May 1982.
[18] G. D. HACHTEL AND A. L. SANGIOVANNI-VINCENTELLI, "A survey of third-generation simulation techniques," *Proc. IEEE*, vol. 69, no. 10, Oct. 1981.
[19] J. KLECKNER, R. SALEH, AND A. R. NEWTON, "Electrical consistency in schematic simulation," in *Proc. IEEE Int. Conf. Circuits Comput.*, pp. 30–34, Oct. 1982.
[20] E. LELARASMEE, A. RUHELI, AND A. L. SANGIOVANNI-VINCENTELLI, "The waveform relaxation method for the time-domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Design of ICAS*, vol. CAD-1, no. 3, pp. 131–145, Aug. 1982.
[21] E. LELARASMEE AND A. SANGIOVANNI-VINCENTELLI, "RELAX: a new circuit simulator for large scale MOS integrated circuits," In *Proc. 1982 Design Automation Conf.*, June 1982.
[23] G. R. BOYLE, "Simulation of integrated injection logic," Univ. of California, Berkeley, ERL Memo. No. ERL-M 78/13, Mar. 1978.
[24] C. A. DESOER AND E. S. KUH, *Basic Circuit Theory*. New York: McGraw-Hill, 1969.
[25] K. SAKALLAH AND S. W. DIRECTOR, "An activity-directed circuit simulation algorithm," in *Proc. IEEE Int. Conf. Circ. Comput.*, pp. 1032–1035, Oct. 1980.
[26] J. VARGA, *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1969.
[27] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.
[28] A. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*. Boston, MA: Ginn, 1964.
[29] A. R. NEWTON, "The simulation of large-scale integrated circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 741–749, Sept. 1979.
[30] G. ARNOUT AND H. DE MAN, "The use of threshold functions and boolean controlled network element for macromodelling of LSI circuits," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 326–332, June 1978.
[31] G. PERSKY, D. N. DEUTSCH, AND D. G. SCHWEIKERT, "LTX—A system for the directed automation design of LSI circuits," in *Proc. 13th Design Automation Conf.*, 1976.
[32] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*. New York: Wiley, 1966.
[33] H. SHICHMAN AND D. A. HODGES, "Modeling and simulation of insulated gate field-effect transistor switching circuits," *IEEE J. Solid-State Circuits*, vol. SC-3, pp. 285–289, Sept. 1968.
[34] A. RUEHLI, A. SANGIOVANNI-VINCENTELLI, AND G. RABBAT, "Time analysis of large-scale circuits containing one-way macromodels," *IEEE Trans. Circuits Syst.*, vol. CAS-29, pp. 185–191, Mar. 1982.

[35] G. DE MICHELI AND A. SANGIOVANNI-VINCENTELLI, "Characterization of integration algorithms for the timing analysis of MOS VLSI circuits," *Int. J. Circuit Theory Appl.*, pp. 299–309, Oct. 1982.

[36] G. DE MICHELI, A. R. NEWTON, AND A. SANGIOVANNI-VINCENTELLI, "Symmetric displacement algorithms for the timing analysis of MOS VLSI circuits," *IEEE Trans. Computer-Aided Design of CAS*, to be published.

[37] C. W. GEAR, *Numerical Initial Value Problems for Ordinary Differential Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[38] G. DAHLQUIST AND A. BJORCK, *Numerical Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1974.

[39] W. KAHAN, private notes, 1975.

[40] G. DE MICHELI, "New algorithms for the timing analysis of MOS circuits," M.S. Thesis, Univ. of California, Berkeley, 1980.

[41] A. R. NEWTON, "The analysis of floating capacitors for timing simulation," in *Proc. 13th Asilomar Conf. Circuits Syst. Comput.*, Nov. 1979.

[42] G. DE MICHELI, A. SANGIOVANNI-VINCENTELLI, AND A. R. NEWTON, "New algorithms for timing analysis of large circuits," in *Proc. 1980 Int. Symp. Circuits Syst.*, 1980.

[43] T. HUANG AND A. SANGIOVANNI-VINCENTELLI, "Analysis of a method for the timing simulation of large-scale MOS circuits containing floating capacitors," in preparation.

[44] *The CRAY X-MP Series of Computers*, CRAY Research, Inc., Mendota Heights, MN, Pub. MP-0001, 1982.

[45] *CDC CYBER 200 MODEL 203 Computer System Hardware Reference Manual (Preliminary Edition)*, Control Data Corporation, St. Paul, MN, Pub. 60256010, May, 1980.

[46] D. A. CALAHAN AND W. G. AMES, "Vector processors: Models and applications," *IEEE Trans. Circuits Syst.*, vol. CAS-26, Sept. 1979.

[47] A. E. CHARLESWORTH, "An approach to scientific array processing: The architectural design of the AP-120B/FPS-164 family," *Comput.*, vol. 14, Sept. 1981.

[48] J. B. DENNIS, "Data flow supercomputers," *Comput.*, vol. 13, no. 11, pp. 48–56, Nov. 1980.

[49] I. WATSON AND J. R. GURD, "A practical data flow computer," *Comput.*, vol. 15, no. 2, pp. 51–57, Feb. 1982.

[50] E. LELARASMEE, "The waveform relaxation method for the time-domain analysis of large scale integrated circuits: Theory and applications," Ph.D. dissertation, Univ. of California, Berkeley, 1982; also Memo UCB/ERL M82/40, 1982.

[51] R. SALEH, "Iterated timing analysis and SPLICE1," M.S. thesis, Univ. of California, Berkeley, 1983.

[52] A. R. NEWTON, "The simulation of large scale integrated circuits," Ph.D. dissertation, Univ. of California, Berkeley, July 1978; also Univ. of California, Berkeley, Memo UCB/ERL M78/52, July 1978.

[53] J. KLECKNER AND A. R. NEWTON, "Advanced techniques for iterated timing analysis," in preparation.

[54] D. SENDEROWICZ, "An NMOS integrated vector-locked loop," Univ. of California, Berkeley, Memo. No. UCB/ERL M82/83, Nov. 12, 1982.

[55] J. WHITE AND A. SANGIOVANNI-VINCENTELLI, "RELAX2: A new waveform relaxation approach for the analysis of LSI MOS circuits," in *Proc 1983 Int. Symp. Circuits Syst.*, May 1983.

[56] E. LELARASMEE AND A. SANGIOVANNI-VINCENTELLI, "Some new results on waveform relaxation algorithms for the simulation of integrated circuits," in *Proc. 1982 IEEE Int. Large-Scale Syst. Symp.*, pp. 371–376, Oct. 1982.

[57] J. KAYE AND A. SANGIOVANNI-VINCENTELLI, "Solution of piecewise linear ordinary differential equations using waveform relaxation and Laplace transforms," in *Proc. 1982 Int. Conf. Circuits Comput.*, pp. 180–183, Oct. 1982; also *IEEE Trans. Circuits Syst.*, to be published.

[58] M. GUARINI AND O. A. PALUSINSKI, "Integration of partitioned dynamic systems using waveform relaxation and modified functional linearization," *1983 Summer Computer Simulation Proc.*, July 11–13, 1983.

[59] T. Y. FENG, "A survey of interconnection networks," *Comput.*, no. 11, pp. 12–27, 1981.

[60] D. D. GAJSKI, D. J. KUCK, AND D. A. PADUA, "Dependence driven computation," *Proc. IEEE Spring Compcon*, pp. 168–172, Feb. 1981.

[61] D. A. PADUA, D. J. KUCK, AND D. H. LAWRIE, "High-speed multiprocessors and compilation techniques," *IEEE Trans. Comput.*, vol. C-29, no. 9., pp. 763–776, Sept. 1980.

[62] D. H. LAWRIE, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1145–1155, Dec. 1975.

[63] J. T. DEUTSCH AND A. R. NEWTON, "Data-flow based behavioral-level simulation and synthesis," *Proc. IEEE ICCAD Conf.*, Sept. 1983.

[64] J. T. DEUTSCH, "Multiprocessor computer architecture," Worcester Polytechnic Institute, MQP Rep. June 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403-0008/$01.25/0

MATRIX METHODS FOR QUEUING PROBLEMS*

LINDA KAUFMAN†

Abstract. Queuing networks are often analyzed to determine their behavior under different traffic situations. The analysis may indicate the effect of increasing servers on the waiting times of customers. It may also indicate whether the network can become so overloaded that no customer can be served. Most of the quantities of interest (the blocking probabilities and waiting times for various traffic streams) can be expressed in terms of the steady-state probabilities that are the solution of the local balance or Kolmogorov equations. For most models currently used, these probabilities can be expressed analytically as a product of quantities that can be easily ascertained. However, when the current state of the system dictates future action, such an analytic expression is not always available and the balance equations must be solved explicitly. Even for systems with relatively small numbers of queues (say 4) and a small number of waiting spaces and servers per queue (say 20), the number of linear equations can be huge, i.e. much more than 10,000, and hence these equations are rarely solved. However, most of the equations are sparse, highly structured, and possess enough algebraic structure that it is possible to solve some modest systems. In this paper we will discuss various methods to solve for the steady state probabilities which form a normalized null-vector of a singular matrix. These methods have been traditionally used to solve nonsingular linear systems that arise during the solution of partial differential equations. We show that when applying certain well known iterative techniques the ordering of equations is important, but that the traffic flow usually dictates an appropriate ordering. Our experience in applying various techniques to a queue overflow problem and to a tandem queue problem is described.

Key words. M-matrices, singular matrices, queuing, Gauss-Seidel

1. Introduction.

Queuing networks are often analyzed to determine their behavior under different traffic situations. The analysis often indicates the effect of increasing servers on the waiting times of customers and on the blocking of customers. It may indicate whether the network can become so overloaded that no customer can be served. Most of the quantities of interest (the blocking probabilities and waiting times for various traffic streams) can be expressed in terms of the steady-state probabilities that are the solution of the local balance or Kolmogorov equations. For most models currently used, these probabilities can be expressed analytically as a product of quantities that can be easily ascertained. However, when the traffic routing is dependent not only on the global parameters of the system but also on the local traffic situation, such an analytic expression is not always available and the balance equations must be solved explicitly. Even for systems with relatively small numbers of queues (say 4) and a small number of waiting spaces and servers per queue (say 20), the number of equations can be huge, i.e. much more than 10,000, and hence these equations are rarely solved. However, most of the equations are sparse, highly structured, and possess enough algebraic structure that it is possible to solve some modest systems. In this paper we will discuss various methods which have been used to solve these problems.

Section 2 will discuss the setting up of the balance equations and properties possessed by the resulting system of equations. The steady-state probabilities form a

*Received by the editors March 1, 1982, and in final revised form November 8, 1982. This paper was typeset at Bell Laboratories, Murray Hill, New Jersey, using the ***troff*** program running under the UNIX™ operating system. Final copy was produced on July 26, 1983.

†Bell Laboratories, Murray Hill, New Jersey 07974.

normalized null-vector of a matrix with positive diagonal elements and nonpositive off-diagonal elements. Section 3 will discuss determining this null-vector using direct methods. Often the matrix possesses some algebraic structure which may be used to determine a matrix decomposition swiftly and using less space than traditional Gaussian elimination. Section 4 will discuss various methods based on matrix splitting for finding the probability distribution. These methods have traditionally been used to solve nonsingular linear systems with M-matrices that have arisen during the solution of partial differential equations. When the singular matrix is 2-cyclic, Gauss-Seidel and its variants will converge. When the matrix does not have this property, which may occur in a tandem queue situation, Gauss-Seidel will converge for all traffic situations only if the rows and columns are ordered appropriately. A correct ordering can easily be determined by considering the traffic flow. Although the singular system can be easily modified to a nonsingular system for which convergence is assured, we suggest that this approach not be taken. The splitting methods, which have been traditionally applied to nonsingular problems, will normally converge much, much faster when applied directly to the singular system which has been appropriately ordered.

At the end of the paper we consider two examples. The first involves overflowing queues which generates a singular system with a symmetric zero structure. In this case taking advantage of the algebraic structure of the problem is worthwhile and we approach the problem in a manner similar to that taken by Buzbee, Golub, and Nielson [4] for solving Poisson's equation. The second problem, which stems from a model of a packet switch network, generates a problem in which the ordering of the equations is important. We show how the traffic flow dictates an appropriate ordering.

2. The equations. A Markovian analysis of a queuing network based on solving the Kolmogorov equations (see Neuts[12]) for the steady state probability distribution involves finding the null-vector of a large, sparse, structured matrix. In this section we will detail the construction of this matrix, consider the effect of various network protocols on this matrix and discuss certain properties of the matrix which might help us determine its null-vector. We will be mainly concerned with problems having interacting queues because these problems lead to systems whose solutions cannot be found using simple analytic techniques and are usually solved explicitly via the balance equations.

In general let us assume that the system to be analyzed has k queues, with $n_1,n_2,\ldots,n_k$ spaces in each queue. Solving the balance equations yields the steady state probability distributions $p_{i_1,\ldots,i_k}$ giving the probability that there are i_1 occupied spaces in queue 1, i_2 occupied spaces in queue 2,..., and i_k occupied spaces in queue k. Note that $0 \le i_j \le n_j$ for $1 \le j \le k$, so that there are $n = \prod_{j=1}^{k} (n_j+1)$ states and hence n equations, 1 per state.

The probabilities are rarely of interest in themselves. They may be used however to compute such quantities as the loss (or blocking) probabilities, the probability of overflow from one queue to another, the average waiting times per queue etc.

The balance equations describe the rate at which state $(i_1,i_2,\ldots,i_k)$ is left and the rate and states from which that state is entered. For example consider the network with 2 queues, 5 servers per queue, and 9 spaces in each queue. Suppose customers enter the first queue at rate λ_1 and leave with rate μ_1 and enter the second with rate λ_2 and leave with rate μ_2. Let us also assume that a customer entering the first

queue can be serviced by the second queue if all spaces in the first queue are filled. This last assumption insures that most analytical solution techniques will fail. If

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases}$$

then the balance equations describing the system can be written as

$$\begin{aligned} &\left(\lambda_1\left(1-\delta_{i9}\delta_{j9}\right)+\lambda_2\left(1-\delta_{j9}\right)+\mu_1 \min(i,5)+\mu_2 \min(j,5)\right)p_{ij} \\ &\quad = \lambda_1\left(1-\delta_{i0}\right)p_{i-1,j}+\left(1-\delta_{j0}\right)\left(\lambda_1\delta_{i9}+\lambda_2\right)p_{i,j-1}+\mu_1\left(1-\delta_{i9}\right)\min(i+1,5)p_{i+1,j} \\ &\qquad +\mu_2\left(1-\delta_{j9}\right)\min(j+1,5)p_{i,j+1} \qquad \text{for } i,j=0,1,2,\ldots,9. \end{aligned} \tag{2.1}$$

The coefficient of $p_{i,j}$ in (2.1) indicates when (the terms with δ) and the rate at which state (i,j) is left. In particular, it indicates that unless both queues are filled, customers enter the network, changing the state of the system, at rate λ_1. The right hand side of (2.1) indicates from which states, when (the terms with δ), and the rate at which state (i,j) is entered. For example, the coefficient of $p_{i,j-1}$ says that ordinarily one gains a customer in the second queue at rate λ_2 but if the first queue is filled, additional customers arrive at rate λ_1. The expressions of the form $1-\delta$ on the right hand side insure that inadmissible states are not accessed.

Let $\mathbf{p}$ be the vector $(p_{00},p_{01},\ldots,p_{09},p_{10},p_{11},\ldots,p_{99})^T$. Then (2.1) can be written as

$$D\mathbf{p} = C\mathbf{p}$$

where D is a 100×100 diagonal matrix and C is a matrix with 0 on the diagonal. If $A = D-C$, then the p_{ij}'s form the null-vector of the matrix A, i.e.

$$A\mathbf{p} = 0. \tag{2.2a}$$

Since equation (2.2a) determines $\mathbf{p}$ up to a scalar multiple, we add the normalizing condition that

$$\sum_{l=1}^{n} p_l = 1. \tag{2.2b}$$

The matrix A has a number of interesting properties which we will explore using the example in (2.1).

First of all, the matrix A is singular and since each diagonal element is the negative of the sum of the off-diagonal elements in its column, each column sum is 0. The matrix A, like all those we will consider in this paper, is irreducible, that is, there is *no* symmetric permutation of the rows and columns of A which will change it into the form

$$\begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$$

where B and D are square matrices. The fact that A is irreducible is equivalent to the assumption that the underlying Markov chain is ergodic. (See Berman and Plemmons[2], chapter 8.) It also means that there is a unique positive stationary probability distribution vector associated with the problem.

Secondly, A has positive diagonal elements, nonpositive off-diagonal elements, and the diagonal elements are not less than the sum of the magnitudes of the off-diagonal elements in their columns. Matrices having these properties have been called L matrices by Young[17]. The term Z-matrix has also been used to denote matrices with nonpositive off-diagonal elements. A Z-matrix has been called an M-matrix if, as in our case, all principal minors of A are nonnegative or equivalently, the real parts of all eigenvalues of A are nonnegative. (See Berman and Plemmons[2].) We will use the term singular M-matrices. Probably because M-matrices often appear when solving elliptic partial differential equations, many theorems exist governing the handling of such matrices (see [15]).

In the third place, the matrix is large and sparse. Our example of 2 queues led to a 100×100 matrix with at most 5 nonzero elements per row. If we had considered a network of three queues with 9 spaces in the third queue, our matrix would have 1000 rows and at most 7 nonzeroes per row. It is not difficult to contemplate systems with 1,000,000 equations. In general adding queues tends to do more damage than adding to the number of spaces per queue. The matrices are highly structured and many of the reordering techniques developed to handle sparse matrices of modest size ($<$10,000 rows) can often be applied successfully to these problems.

Fourthly, often the matrix exhibits some block structure. The matrix A for the equations in (2.1) can be written as

$$A = \begin{pmatrix} B_1 & C_1 & & & & \\ D_2 & B_2 & C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & D_9 & B_9 & C_9 & \\ & & & D_{10} & B_{10} \end{pmatrix} \tag{2.3}$$

where $C_i = -\mu_1 \min(i,5)I$ is a 10×10 diagonal matrix for $i = 1,2,\ldots,9$, $D_i = -\lambda_1 I$ is a 10×10 diagonal matrix for $i = 2,3,\ldots,10$ and B_i is a 10×10 tridiagonal matrix for $i = 1,2,\ldots,10$. In fact, for $i = 1,2,\ldots,9$ we see that $B_i = B+\min(i-1,5)\mu_1 I$, where B is the matrix

$$\begin{pmatrix} \sigma & -\mu_2 & & & & & & & \\ -\lambda_2 & \sigma+\mu_2 & -2\mu_2 & & & & & & \\ & -\lambda_2 & \sigma+2\mu_2 & -3\mu_2 & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & -\lambda_2 & \sigma+5\mu_2 & -5\mu_2 & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & -\lambda_2 & \sigma+5\mu_2 & -5\mu_2 & \\ & & & & & & -\lambda_2 & \sigma+5\mu_2-\lambda_2 \end{pmatrix}$$

where $\sigma = \lambda_2+\lambda_1$. For B_{10} the subdiagonal elements are $-\lambda_1-\lambda_2$.

Sometimes the blocks themselves can be further partitioned into blocks. For example, if the example in (2.1) had three queues with 9 spaces in each where both the first and second queues can overflow into the third queue, the figure in (2.3)

would be appropriate, but the D's and C's would be 100×100 diagonal matrices and the B's would be a 100×100 matrix having exactly the same block structure as A in (2.3).

Even if the network had tandem queues, the matrix A would have block structure. For 2 queues in tandem the equations would have the form

$$ap_{i,j} = bp_{i-1,j}+cp_{i+1,j-1}+dp_{i,j+1} \tag{2.4}$$

for some a,b,c,d's. The matrix A would have the same block structure as (2.3), but, assuming the i^{th} diagonal block corresponds to holding i fixed and varying j, the B's would be bidiagonal, the D's would again be diagonal, and the C's would have only one diagonal but it would be a subdiagonal.

Exposing the block structures of the system is a useful notational tool, but it may also have some computational advantages. Often iterative algorithms which treat a submatrix as an individual unit converge faster than those which treat each element in the submatrix as a unit. Usually the faster convergence rate is coupled with more work per iteration, but in the case when the matrix is banded with very small bandwidths, the extra work is negligible.

Submatrices of the matrix A often show additional algebraic structure associated with the separability of the original system, which can be exploited as shown in section 3.2. We will call a matrix A separable if it can be written as the sum of the tensor product of matrices

$$A = \sum_{j=1}^{q} B_j \otimes C_j \tag{2.5}$$

where the B's have more than 1 row and there exist matrices Q and Z such that for every j, QB_jZ is a diagonal matrix. Usually q denotes the number of queues in the system. Often all of the B's, except one, are identity matrices, Z is Q^{-1}, and Q contains the eigenvectors of the one B_i which is not an identity matrix. If the system corresponds to more than two queues, the C matrices may also be separable matrices. A separable submatrix corresponds to those states in which the action in any one queue is independent of the state of the other queues. For example, in (2.1) traffic enters queue 1 at the same rate regardless of the state of queue 2, but the incoming rate into queue 2 is increased when queue 1 is full. Thus the first 90 rows and columns of A in (2.3), which correspond to the states in which the first queue is not full, form a separable submatrix, but the whole matrix itself is not separable. In general, separable submatrices correspond to the interior of the state space plus a few, but not all, faces of the boundary. In computational terms one can reduce a separable matrix (or submatrix) to a diagonal matrix by solving small eigenvalue problems. For 3 overflowing queues with 9 spaces in each, for many protocols about 800 of the 1000 resulting equations can be reduced to a diagonal representation by solving 3 eigenvalue problems each 10×10.

Finally, (2.2) contains $n+1$ equations for determining n unknowns. Assuming that n is small, Paige, Styan, and Wachter[13] present about 8 ways, including some least squares approaches, to solve (2.2). Their methods make no attempt to use the underlying structure of the A matrix. In (2.6) several more obvious methods are proposed, some of which use the fact that A may be partitioned as

$$A = \left(\begin{array}{c|c} B & \mathbf{d} \\ \hline \mathbf{c}^T & f \end{array}\right)$$

where B is a nonsingular $(n-1)\times(n-1)$ M-matrix.

(2.6a) (1) Solve (2.2a) and scale $\mathbf{p}$ so (2.2b) is satisfied

(2) Let $\hat{\mathbf{p}} = (p_1,\ldots,p_{n-1})^T$

(2.6b) Solve $B\,\hat{\mathbf{p}} = -\mathbf{d}$

Set $\mathbf{p} = \begin{pmatrix}\hat{\mathbf{p}}\\ 1\end{pmatrix}$

Scale $\mathbf{p}$ so (2.5b) is satisfied

(3) Solve the system

$$\text{(2.6c)} \qquad \left(\begin{array}{c|c} B & \mathbf{d} \\ \hline \mathbf{e}^T & \end{array}\right)\mathbf{p} = \mathbf{e}_n$$

where $\mathbf{e}^T$ is the vector (1, 1, ..., 1) and $\mathbf{e}_n$ is the last column of the identity matrix.

The coefficient matrices in (2.6b) and (2.6c) may be ill-conditioned and several examples have been found in which they have appeared singular up to the precision of the computer, although they have independent rows theoretically. For these ill-conditioned problems, small changes in the matrix might produce large changes in the solution although the exact perturbation is of course also dependent on the right hand side. (See Skeel[14].) In (2.6b) the first $(n-1)$ columns of A have been arbitrarily chosen as a basis for A. Perhaps, a better conditioned system might arise if a different column were deleted, although results by Harrod and Plemmons[8] indicate that it usually does not matter which column is eliminated. In section 4 we discuss various iterative methods for determining $\mathbf{p}$. We have found that applying some of these iterative schemes to (2.6a) requires much fewer iterations than applying the same scheme to (2.6b). Again the results may vary if a different basis were chosen for A. In approach (2.6c) one loses some of the nice properties of the system. When using a direct approach like Gaussian elimination, a naive user faced with banded matrix solvers and general solvers in his local subroutine library, may choose to ignore the fact that B is banded and use an expensive general solver because the coefficient matrix in (2.6c) does not have exactly a banded structure. Also the matrix is not an M-matrix, which eliminates the underpinning of the iterative methods in section 4. Thus it seems most natural to approach the problem using method (2.6a). In our future discussions we will mainly be concerned with (2.6a) but we will allude to the other approaches as well.

3. Direct methods. In this section we discuss direct methods for solving

$$\text{(3.1)} \qquad A\mathbf{p} = \mathbf{0},$$
$$\sum_{i=1}^{n} p_i = 1$$

where A is a sparse, nonsymmetric, irreducible, square matrix arising during a queuing analysis. The matrix A is often banded and usually has a regular, discernible zero structure. It may also possess some algebraic structure which one may wish

to take advantage of. In most analyses the problem will be solved many times with coefficient matrices having the same zero structure but with slightly different values for the nonzero elements.

Direct methods are used when n is not large, say $n < 600$. Even when the proposed problem is much larger, one might be able to study the salient features of the network by considering a system which can be solved directly rather than iteratively. For example, an analysis of a system with 4 or 5 buffer spaces in 4 queues will probably tell a designer as much about whether a given protocol will produce negligible throughput in a heavy traffic situation as a similar analysis with 20 buffer spaces per queue. In many instances the data that is entered into A is so imprecise that only trends and the qualitative behavior of a system can be gleaned from any analysis. An accurate numerical forecast might be meaningless and hence there might not be any good scientific reason to solve a large problem. On the other hand, if a designer is sure of his data and wishes to determine the minimum number of buffer spaces required by various queues to maintain a given throughput level, one might have to solve large problems and the techniques in the next section might be more amenable.

3.1 General direct methods. Funderlic and Mankin[7] have shown that for these problems A may be written as

$$A = L\,U_1 \tag{3.2}$$

where L is a nonsingular lower triangular matrix and U_1 is the upper triangular matrix

$$\begin{pmatrix} U & \mathbf{d} \\ 0 & 0 \end{pmatrix} \tag{3.3}$$

where U is an upper triangular nonsingular matrix. If one tried to obtain the LU factorization using Gaussian elimination with partial pivoting, one would discover that no pivoting would be performed and the resulting decomposition would look like (3.2). Thus one can used the following algorithm for solving (3.1)

(A) Set $p_n = 1$

(B) Solve $U\tilde{\mathbf{p}} = -\mathbf{d}$ where $\tilde{\mathbf{p}}^T = (p_1, \ldots, p_{n-1})^T$

(C) Let $c = \sum_{i=1}^{n} p_i$

(D) For $i = 1,..,n$
Set $p_i = p_i/c$.

Notice that the L matrix is not needed explicitly after U has been formed and if one performed Gaussian elimination as is usually done, there would be no reason to store L. In a typical situation with a sparse matrix package(see [5],[6]), about half the space could be saved. Bandsolvers usually overwrite a band matrix A with L and U and require extra space for the overflow diagonals of U which might occur because of pivoting for stability. As Funderlic and Mankin have shown, this extra space is unnecessary and requires typically 50% more storage. If the bandsolver does not accept A initially as a whole matrix, but receives it one row or column at a time, as in most sparse matrix packages, L need not be saved, thus reducing the storage costs by about 50%.

Whether a bandsolver is preferable to a sparse matrix solver usually depends on the nature of the problem, the available resources and the architecture of the machine. The first phase of most sparse matrix solvers is the determination of an ordering of the rows and columns of A to minimize the number of nonzeros in L and U. For these problems the row ordering should be exactly that of the column ordering, which would correspond to reordering the states in the system. To obtain this ordering one might apply an ordering routine designed for problems with symmetric zero structure to the zero structure of the matrix $A + A^T$. When the network has queues in tandem, the zero structure of A is often nonsymmetric. For problems in which space is crucial, the sparse matrix solver will usually produce a decomposition having 90% fewer nonzeroes than that produced by the bandsolver. However, for each nonzero element in the sparse decomposition one needs a numerical indication and some mechanism to indicate which element it is. Thus the saving in space might be only 45%. Because queuing problems are often multidimensional(8-dimensional problems are not uncommon), the extra space released by the sparse matrix solver might only mean that the buffer size in each queue might be increased by 1, before the available resources are exhausted. Theoretically, one would expect about a 90% decrease in the execution time of the numerical phase of a sparse solver over a bandsolver. However, sparse matrix solvers always involve more complicated indexing procedures which cut the percentage. Moreover, if time is really crucial, one might use a vector machine like the Cray-1. The bandsolver provided by the Cray-1 calls assembly language subroutines for inner and outer products and is very good on large bandwidth problems. Even when the numerical phase of a fast sparse matrix solver has been modified to call hand coded assembly language subroutines for sparse inner and outer products, the bandsolver supplied by Cray is usually faster.

3.2 Decompositions with algebraically structured matrices. Often linear systems arising from queuing problems have sufficient algebraic structure that it pays to exploit this structure rather than use a general sparse matrix solver. Usually in these problems, A can be partitioned as

$$A = \begin{pmatrix} C & D \\ E & F \end{pmatrix} = \begin{pmatrix} I & 0 \\ EC^{-1} & I \end{pmatrix} \begin{pmatrix} C & D \\ 0 & F-EC^{-1}D \end{pmatrix} = LU \tag{3.4}$$

where C is very large, nonsingular, and easy to factor, and $N=F-EC^{-1}D$ is a dense matrix that is small enough that solving a linear system with N as a coefficient matrix is relatively simple.

For example, consider the problem with two queues, 1 server per queue in which a job destined for the first queue can be handled by the second queue if no space in the first queue is available. Assume the mean service rate for each queue is μ, the mean arrival rate for the first queue is λ_1, the mean arrival rate for the second queue is λ_2, the first queue has n_1 spaces and the second queue has n_2 spaces. Let p_{ij} indicate that there are i spaces filled in the first queue and j in the second queue. The balance conditions lead to the following system of equations:

$$\begin{aligned} &\left(\lambda_1\left(1-\delta_{in_1}\,\delta_{jn_2}\right) + \lambda_2\left(1-\delta_{jn_2}\right) + \mu\left(2-\delta_{i0} - \delta_{j0}\right)\right)p_{ij} \\ &\quad = \lambda_1\left(1-\delta_{i0}\right)p_{i-1,j} + \left(1-\delta_{j0}\right)\left(\lambda_1\,\delta_{in_1} + \lambda_2\right)p_{i,j-1} \\ &\qquad + \left(1-\delta_{in_1}\right)\mu p_{i+1,j} + \left(1-\delta_{jn_2}\right)\mu p_{i,j+1}\,. \end{aligned} \tag{3.5}$$

If the natural lexicographical ordering is used, then A may be written as

$$A = \left(\begin{array}{ccc|c} & & & 0 \\ & C & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 \; H & J \\ & & & F \end{array}\right) \tag{3.6}$$

where J is the $(n_2+1)\times(n_2+1)$ diagonal matrix $-\mu I$, H is an $(n_2+1)\times(n_2+1)$ diagonal matrix $-\lambda_1 I$, F is a tridiagonal matrix and C is a separable matrix of the form

$$C = I \otimes G + V \otimes I$$

where G is the $(n_2+1)\times(n_2+1)$ tridiagonal matrix

$$\begin{pmatrix} \lambda_2 & -\mu & & & & & \\ -\lambda_2 & \mu+\lambda_2 & -\mu & & & & \\ & -\lambda_2 & \mu+\lambda_2 & -\mu & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\lambda_2 & \mu+\lambda_2 & -\mu & \\ & & & & -\lambda_2 & \mu & \end{pmatrix}$$

and V is the $n_1\times n_1$ tridiagonal matrix

$$\begin{pmatrix} \lambda_1 & -\mu & & & & & \\ -\lambda_1 & \mu+\lambda_1 & -\mu & & & & \\ & -\lambda_1 & \mu+\lambda_1 & -\mu & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\lambda_1 & \mu+\lambda_1 & -\mu & \\ & & & & -\lambda_1 & \mu & \end{pmatrix}.$$

We will show that C is simple to work with because it is separable. Our ideas are similar to those found in [4] for solving large linear systems arising when solving elliptic partial differential equations.

First of all, because of the sign structure of the off-diagonal elements of G, one can find a real matrix Q such that $QGQ^{-1} = K$, a real diagonal matrix with diagonal elements $k_1,\ldots,k_{n_2+1}$.

Thus

$$(I \otimes Q)C(Q^{-1} \otimes I) = I \otimes K + V \otimes I. \tag{3.7}$$

It should also be noted that forming Q^{-1} is easy. Since G is diagonally similar to a symmetric matrix, the matrix Q may be written as

$$Q = \hat{Q}S \tag{3.8}$$

where S is a diagonal matrix and $\hat{Q}$ is an orthogonal matrix. Thus

$$Q^{-1} = S^{-1}\hat{Q}^T.$$

To solve a system

$$C\mathbf{x}=\mathbf{y}, \tag{3.9}$$

it is simplest to consider $\mathbf{x}$ as an $(n_2+1\times n_1)$ array X and $\mathbf{y}$ as an array Y of the same size. To solve (3.9) one might proceed as follows

(A) Set $Z=\hat{Q}SY$
Set $M=Z^T$

(B) For $i=1,2,\ldots,n_2+1$
Solve $(V+k_iI)\mathbf{w}_i=\mathbf{m}_i$
Let $W=(\mathbf{w}_1,\ldots,\mathbf{w}_{n_2+1})$

(C) Set $X=S^{-1}\hat{Q}^TW^T$.

Steps (A) and (C) of the above algorithm each require $(n_2+1)^2n_1$ operations and step (B) about $5(n_2+1)n_1$ operations. The matrix-matrix multiplications can of course be vectorized, and all the systems in step (B) are independent so at the expense of storing all of them, one could do them all simultaneously.

If $n_1+1 = n_2+1 = m$, only m^2 locations are needed to store Q, another m^2 to store

$$N=F - (O \ : H)\ C^{-1} \begin{pmatrix} O \\ \cdot \\ \cdot \\ J \end{pmatrix}$$

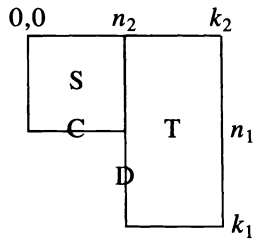
and $\approx 2m^2$ to store the intermediate quantities while working with C. Even if $m = 200$, which would give us a problem with 40,000 variables, the problem could be solved in about 160,000 locations. A sparse matrix package, which reordered for sparsity, would need at least $2(m^2 \log m)$ or at least 640,000 locations. A bandsolver would need at least $2m^3$ locations or 8 million locations to store the original matrix.

The operation count is also smaller for the separable approach. Forming Q and solving a system with N each requires $O(m^3)$ operations. Forming N is the expensive step because one must solve m systems with C as the coefficient matrix, requiring $O(m^4)$ operations. However, each Y in (3.9) has only 1 nonzero element in the whole matrix, so that the operation count for step (A) is negligible and since only the last column of each X matrix is needed, step (C) can be cut to $O(m^2)$ operations. Thus the whole separable algorithm really requires $O(m^3)$ operations. In contrast, a bandsolver would require $O(m^5)$ operations.

We chose a simple problem to show how to use the algebraic structure of the problem. Even examples with much different geometries have separable portions. For example, consider the 2-queue overflow problem with n_1 servers in the first queue and w_1 waiting spaces and n_2 servers in the second queue and w_2 waiting spaces for that queue. Assume that a customer waiting for service in the first queue must be handled by the second queue if a second queue server is available. If we let $p_{i,j}$ denote the probability that i customers are being served or waiting in the first group and j in the second group, then

$$p_{ij} = 0 \quad \text{for} \quad i > n_1+1 \text{ and } j \leq n_2.$$

If we let $k_1 = n_1+w_1$ and $k_2 = n_2+w_2$, we may think of the probabilities as values associated with the L shaped domain



If we denote by region **S**, the values associated with p_{ij} for $0 \le i < n_1$, $0 \le j < n_2$, by region **T** the values associated with p_{ij} for $0 \le i \le k_1$, $n_2+1 \le j \le k_2$, by region C the points on line $p_{n_1,j}$, $0 \le j < n_2$ and by region D the points p_{in_2}, $0 \le i \le k_1$, then the matrix A will have the following zero structure:

$$A = \begin{pmatrix} S & 0 & X & X \\ 0 & T & 0 & X \\ X & 0 & C & X \\ X & X & X & D \end{pmatrix}$$

where the large matrices S and T will each be separable. Indeed for many protocols with 2 queues, this basic idea of using small eigendecompositions to help solve a linear system is viable. When the queues are in tandem, however, the approach does not carry over easily.

Unfortunately, the idea usually cannot be extended to 3 queues of size n_1, n_2, and n_3 spaces in each if these quantities are even of moderate size. Almost every protocol entails finally solving a general matrix of size say n_2n_3 by n_2n_3. Thus only if these numbers were say less than 20 each, could one fit the problem in the core of a midsize machine.

Our presentations of the solution method involved a small eigenvalue-eigenvector calculation. Bank [1] has presented another method for dealing with separable matrices which would be more efficient, especially if one is solving systems with multiple right hand sides of random zero structure. Such a situation might arise if one chose to solve the problem using an iterative technique, as given in the next section, in which the D matrix of (3.4) was put on the right hand side of the equation and the rest of the $\tilde{A}$ matrix was left on the left hand side. In the problem presented, the algebraic structure of the E and D matrices of (3.4) was so special that it was cost effective to use the eigenvalue-eigenvector decomposition.

4. Iterative methods. In this section we will discuss methods for finding the null-vector of the matrix A defined in section 2 that are based on splitting that matrix A into two matrices M and N such that

$$A = M-N \tag{4.1}$$

where M is a nonsingular M-matrix and $n_{ij} \ge 0$. Such a splitting is called a regular splitting and can always be found. For example in the Jacobi method M is D, the diagonal of A, and in the Gauss-Seidel method M is $D+L$ where L is the strictly

lower triangular portion of A and N is $-U$, where U is the strictly upper triangular portion of A.

Our methods will all have the form

(4.2)
guess $\mathbf{p}^{(0)}>\mathbf{0}$
Until convergence
Solve $M\mathbf{p}^{(k)} = N\mathbf{p}^{(k-1)}$
Normalize $\mathbf{p}^{(k)}$ so that $||\mathbf{p}^{(k)}|| = 1$.

In section 4.1 we give theoretical convergence theorems. In section 4.2 some computational evidence of the performance of various splitting methods is given for 2 different types of examples. In particular we consider whether (2.6a), solving a singular problem, is preferable to (2.6b), solving a nonsingular problem, and whether the ordering of the equations is important given a particular matrix and iteration scheme. For one of our examples we show the effect of using the algebraic structure of the matrix to form a block iterative method. We also indicate our experience with various accelerating techniques such as SOR and Chebychev acceleration. In section 4.3 we give some implementation advice and draw some conclusions about the feasibility of solving large problems.

4.1 Convergence results. In this section we will discuss convergence properties of algorithms of the form (4.2) to find the null-vector of $A = M-N$.

In this section we will use the following notation:

$$T = M^{-1}N,$$

$$\rho(T) = \text{modulus of largest eigenvalue of } T,$$

$$\sigma(T) = \max\{|\lambda| \mid \lambda \text{ is an eigenvalue of } T \text{ and } \lambda \neq 1\}.$$

The matrix T is central to the convergence theory since (4.2) may be rewritten as

$$\mathbf{p}^{(k)} = T\mathbf{p}^{(k-1)}. \tag{4.3}$$

Our analysis below based on the eigenvalues of T is similar to that in [2], section 7.6.

LEMMA 4.1. $\mathbf{p} = T\mathbf{p}$ *iff* $A\mathbf{p} = 0$.

Proof. If $\mathbf{p} = T\mathbf{p}$, then $\mathbf{p} = M^{-1}N\mathbf{p}$ which implies $(M-N)\mathbf{p} = 0$, or $A\mathbf{p} = 0$.

If $A\mathbf{p} = 0$, then $(M-N)\mathbf{p} = 0$ implying $M\mathbf{p} = N\mathbf{p}$ or $\mathbf{p} = M^{-1}N\mathbf{p}$. Thus the fixed point of our algorithm, the eigenvector corresponding to the eigenvalue 1 of T, is the null-vector of A.

Since the null-vector of A is assumed to be unique, the eigenvalue of T at 1 corresponds to only 1 linear independent eigenvector. Moreover, this eigenvalue is the largest one of T as Lemma 4.2 indicates.

LEMMA 4.2. $\rho(T) = 1$.

Proof. Since $\sum_{i=1}^{n} a_{ij} = 0$ for $j = 1,2,\ldots,n$, i.e. A is stochastic

$$\mathbf{e}^T A = 0$$

where **e** is the vector of all 1's. This implies

$$\mathbf{e}^T M = \mathbf{e}^T N$$

or

$$M^T\mathbf{e} = N^T\mathbf{e}$$

implying

$$\mathbf{e} = M^{-T}N^T\mathbf{e}.$$

This means that the maximum row sum of $M^{-T}N^T$ is 1 or $||M^{-T}N^T||_\infty = 1$. But

$$||M^{-T}N^T||_\infty \geq \rho(M^{-T}N^T) = \rho(NM^{-1}) = \rho(M^{-1}N) = \rho(T) .$$

Thus

$$1 \geq \rho(M^{-1}N) .$$

But we know 1 is an eigenvalue of T. Therefore

$$1 = \rho(M^{-1}N).$$

To determine when algorithm (4.2) converges let us first consider the case when T has a full set of linearly independent eigenvectors. Let $\mathbf{x}_i$ represent the eigenvector corresponding to the eigenvalue λ_i of T. We may write $\mathbf{p}^{(0)}$ as a linear combination of these eigenvectors, i.e.

$$\mathbf{p}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

and if $\lambda_1 = 1$, then $\alpha_1 \neq 0$ since all indices of $\mathbf{p}^{(0)}$ and $\mathbf{x}_1$ are positive. Therefore after k steps of (4.2)

$$\mathbf{p}^{(k)} = \beta \sum_{i=1}^{n} \alpha_i \lambda_i^k \mathbf{x}_i$$

where β is some normalizing constant. If for $j>1, |\lambda_j|<1$, we see that

$$\lim_{k\to\infty} \mathbf{p}^{(k)} = \mathbf{x}_1/||\mathbf{x}_1|| = \mathbf{p}$$

which means that not only is convergence is assured but also that the normalization step in (4.2) is superfluous until the end.

If there are other eigenvalues on the unit circle besides the one at 1, algorithm (4.2) will not converge. In particular if $\lambda_2 = -1$, and all the other eigenvalues are within the unit circle, eventually every other iterate would repeat itself and neither would be the solution. This in fact can happen. According to Theorem 4.3 of Varga[15], when A is 2-cyclic, that is, the directed graph associated with A is strongly connected and the greatest common divisor of the lengths of all closed paths is 2, as in the case of a tridiagonal matrix, the Jacobi method produces a T matrix, T_J, with the only eigenvalues on the unit circle at -1 and 1. The same theorem however indicates that for this class of problems, for the Gauss-Seidel, the eigenvalues of the T matrix T_{GS} are those of T_J raised to the second power. Thus the 2 eigenvalues of T_J on the unit circle are both represented by an eigenvalue at 1 for T_{GS}. Thus if $|\lambda| = \rho(T_{GS})$, then $\lambda = 1$, and the Gauss-Seidel method converges.

When T does not have a full set of linearly independent eigenvectors, the results of the analysis is unchanged. Without loss of generality let us assume T has $n-1$ linearly independent eigenvectors, λ_m corresponds to a Jordan block of rank 2, and that $\mathbf{x}_n$ is a principal vector of grade 2 of that block. Thus

$$T\mathbf{x}_m = \lambda_m \mathbf{x}_m$$

and

$$T\mathbf{x}_n = \lambda_m \mathbf{x}_n + \mathbf{x}_m.$$

After k steps of algorithm (4.2) we would have

$$\mathbf{p}^{(k)} = \beta\Big(\sum_{i=1, i\neq m}^{n-1} \alpha_i \lambda_i^k \mathbf{x}_i + (k\alpha_n \lambda_m^{k-1} + \alpha_m \lambda_m^k)\mathbf{x}_m + \alpha_n \lambda_m^k \mathbf{x}_n \Big).$$

If $m=1$, $\mathbf{p}^{(k)}$ will grow in the direction of the null-vector of A and in no other direction. Thus we have to normalize the iterate at each step to detect convergence, but basically we are not hurt if the eigenvalue at 1 is not simple. If λ_m is on the unit circle but not at 1, the algorithm, as in the case of a full set of eigenvectors, will not converge. If λ_m is within the unit circle, growth will initially take place in the direction of its eigenvector but eventually the portion of the iterate in that direction will decrease. If there is more than one Jordan block greater than 1 in rank or some of higher rank than 2, the conclusions are the same.

As we have shown, eigenvalues of T on the unit circle, that are not at 1, are responsible for nonconvergence and for 2-cyclic matrices, the Gauss-Seidel algorithm converges. When the matrix A is not 2-cyclic, the convergence of the Gauss-Seidel algorithm for singular matrices is dependent on the ordering of the rows and columns of the matrix. A matrix is considered p-cyclic if the block directed graph associated with the matrix T_j is strongly connected and the greatest common divisor of the lengths of all closed paths is p. A p-cyclic matrix is considered consistently ordered if the eigenvalues of the matrix

$$K_\eta = \eta D^{-1}L + \eta^{(-p+1)} D^{-1}U \tag{4.4}$$

are independent of η. Since the eigenvalues of T_{GS} of a p-cyclic consistently ordered matrix are those of T_J raised to the p^{th} power and the p^{th} roots of unity are eigenvalues of T_J, Gauss-Seidel converges for p-cyclic consistently ordered matrices. (See Varga[15].) For example, for the 3×3 matrix

$$A = \begin{pmatrix} 1 & & -1 \\ -1 & 1 & \\ & -1 & 1 \end{pmatrix},$$

condition (4.4) is satisfied, the T matrix for Gauss-Seidel has eigenvalues at (0,0,1), and convergence is assured. However if the states had been reordered so that the A matrix was

$$\begin{pmatrix} 1 & -1 & \\ & 1 & -1 \\ -1 & & 1 \end{pmatrix},$$

then condition (4.4) is not satisfied, the eigenvalues of the T matrix of Gauss-Seidel are $(0,1,-1)$, and every other iterate repeats. In general if the underlying graph is p-cyclic, a matrix is consistently ordered if it has the form

$$\begin{pmatrix} x & & & x \\ x & x & & \\ & x & x & \\ & & x & x \\ & & & x & x \end{pmatrix}.$$

In networks in which different traffic flows have different destinations and some traffic resides in fewer buffers than other traffic, the cycles in the graph suggested by the balance equations are not all of the same length. In this case, the ordering of the equations is still important because with a given traffic mix, the matrix may effectively be p-cyclic. For example, consider the matrix suggested by the network analyzed in [9]

$$\begin{pmatrix} 1 & -b & & -1 \\ -1 & b+c & & \\ & -c & 1 & \\ & & -1 & 1 \end{pmatrix}. \tag{4.5}$$

The eigenvalues of T for point Gauss-Seidel are $(0,0,0,1)$, which means convergence is immediate. If the states were renumbered so that the matrix looked like

$$\begin{pmatrix} 1 & -1 & & \\ & 1 & -c & \\ & & c+b & -1 \\ -1 & & -b & 1 \end{pmatrix} \tag{4.6}$$

the eigenvalues of T for the point Gauss-Seidel method would be 0,1, and the 2 roots of $(b+c)\lambda^2+c\lambda+c$. If b were 0, the eigenvalues would be 0 and the cube roots of unity. Thus as the ratio of b to c decreases, the number of iterations required rises dramatically. If the matrix A had been further rearranged to

$$\begin{pmatrix} 1 & & -c & \\ & 1 & -b & -1 \\ & -1 & c+b & \\ -1 & & & 1 \end{pmatrix} \tag{4.7}$$

the eigenvalues of T for point Gauss-Seidel would be $(0,0,-c/(b+c),1)$. If b were zero, every other iterate of point Gauss-Seidel would be repeated.

Even in situations when convergence is guaranteed, the ordering is important as Table 4.4 in section 4.2 indicates. If under certain parameter settings, the second largest eigenvalue lies on the unit circle, then a slight change in these parameters might yield a convergent algorithm. However, the second largest eigenvalue might be so close to the unit circle that convergence is excruciatingly slow.

The problem of having 2 or more eigenvalues of T on the unit circle can be avoided without reordering by using a step length algorithm. Consider the algorithm

$$\mathbf{p}_{k+1} = \frac{1}{\alpha+1}(\alpha I+T)\mathbf{p}_k \tag{4.8}$$

for some fixed α and let $\bar{T} = (\alpha I+T)/(\alpha+1)$. The matrix $\bar{T}$ has an eigenvalue at 1 whose corresponding eigenvector is a null-vector of A; the rest of the eigenvalues of $\bar{T}$ are within the unit circle and the method will always converge. In fact for every eigenvalue of T, there exists an eigenvalue of $\bar{T}$ which lies on the line connecting the eigenvalue of T to 1. This means that if all the eigenvalues corresponding to $\sigma(T)$ have negative real parts, then for small values of α, (4.8) would be faster than (4.2). On the other hand if $\sigma(T)$ corresponded to a positive real eigenvalue, (4.2) would be faster than (4.8) for all values of α. A similar suggestion is made in [2], p. 234.

If A were a nonsingular M-matrix, rather than a singular M-matrix, the theorems of section 3.6 of Varga [15] would assure us that giving two regular splittings $A = M_1-N_1$ and $A = M_2-N_2$ then the method based on N_1 would require no fewer iterations than those based on N_2 if $N_1 \geq N_2$. This fact is certainly reassuring because supposedly the method based on N_2 would require more work per iteration. The proofs in Varga all depend on $\rho(T)$. In our case A is singular and according to Neumann and Plemmons[11], the rate of convergence is $-\ln(\sigma(T))$. It would be nice if the same result would follow for singular matrices but as the following example indicates, this is not the case.

$$\text{Let } A = \begin{pmatrix} 1 & -1/2 & -1/2 & 0 \\ -1/2 & 1 & 0 & -1/2 \\ -1/2 & 0 & 1 & -1/2 \\ 0 & -1/2 & -1/2 & 1 \end{pmatrix}.$$

$$\text{Let } N_1 = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } N_2 = \begin{pmatrix} 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

so that $N_1 \geq N_2$. One would expect that since M_2 has more nonzero elements than M_1, that $\sigma(M_2^{-1}N_2) \leq \sigma(M_1^{-1}N_1)$. However, $\sigma(M_1^{-1}N_1) = 0$ and $\sigma(M_2^{-1}N_2) = 1/9$. Another example showing the same phenomenon was devised by Hans Schneider and given in [3].

Considering the fact that the convergence theory has all been worked out for nonsingular M-matrices, it would seem natural to suggest to fix one element in the probability vector, delete one row and column from A and solve a linear system with a nonsingular coefficient matrix for the remaining elements of the probability vector as in (2.6b). This scheme is a viable alternative to working with the singular matrix only if it does not converge significantly slower. In certain cases one can theoretically indicate the best choice. Assume A is a symmetric matrix and partition A as

$$A = \begin{pmatrix} B & \mathbf{c} \\ \mathbf{c}^T & f \end{pmatrix}.$$

Let us consider the convergence rate of Gauss-Seidel applied to the problem $A\mathbf{p} = 0$ and the convergence rate of the Gauss-Seidel algorithm applied to the problem

$B\hat{\mathbf{p}} = -\mathbf{c}$. We may write B as $B = L_B + D_B + L_B^T$, where L_B is the strict lower triangular part of B. Similarly we may write A as $A = L + D + L^T$ where L is the strict lower triangular part of A. The rate of convergence of the Jacobi method applied to B is governed by the eigenvalues of $T_{JB} = D_B^{-1}(L_B + L_B^T)$. The eigenvalues of T_{JB} are also the eigenvalues of $S_B = D_B^{-1/2}(L_B + L_B^T)D_B^{-1/2}$ which interleave the eigenvalues of $S_A = D^{-1/2}(L + L^T)D^{-1/2}$, because S_B is the principal $(n-1)\times(n-1)$ submatrix of the symmetric matrix S_A. Now the Jacobi algorithm applied to A is governed by the eigenvalues of T_J, which are the eigenvalues of S_A. Thus, the relationship of the eigenvalues of T_{JB}, represented by y to those of T_J represented by x is

$$-1 \le x_1 \le y_1 \le x_2 \le y_2 \le \ldots \le x_{n-1} \le y_{n-1} \le x_n = 1.$$

If A were 2-cyclic and consistently ordered, $x_1 = -1$, and the eigenvalues of the Gauss-Seidel method applied to both problems are the squares of the eigenvalues of the Jacobi method applied to both problems. Thus in this case, Gauss-Seidel applied to the B matrix is controlled by $\max(y_1^2, y_{n-1}^2)$. and Gauss-Seidel applied to the A matrix is controlled by $\max(x_2^2, x_{n-1}^2)$ and is definitely faster than the same method applied to B. Thus the condition that guarantees convergence, also guarantees that Gauss-Seidel applied to the singular matrix would be faster.

As another case consider the following matrix

$$A = \begin{pmatrix} \alpha & c & e \\ a & \beta & f \\ b & d & \gamma \end{pmatrix} \quad \text{where} \quad \begin{aligned} \alpha &= -(a+b), \\ \beta &= -(c+d), \\ \gamma &= -(e+f). \end{aligned} \tag{4.9}$$

The controlling eigenvalue for the Gauss-Seidel method applied to the singular problem is $bcf \,/\, (\alpha\beta\gamma)$ while that for the nonsingular scheme involving the top 2×2 matrix is $ac \,/\, (\alpha\beta)$. Thus if $a=c=e=1$ and $b=d=f=9$, then the controlling eigenvalue for Gauss-Seidel for the singular scheme is 81/1000 while if one deletes the last row and column, the controlling eigenvalue is 1/100. However, given a matrix of the form (4.9), there always exists a symmetric permutation of A, so that the singular system is better.

In practice, we have observed that the magnitude of the controlling eigenvalue for the singular scheme has always been less than that for the nonsingular iteration. When n is large, the largest eigenvalue for the nonsingular iteration matrix will normally be so close to 1 that the method will be excruciatingly slow. In fact, the first time the nonsingular scheme was tried by the author, convergence became so slow that she suspected that the program was faulty and only inspection of the eigenvalues for a small problem allayed those fears. In section 4.2 the controlling eigenvalues for the singular scheme and nonsingular scheme are given for several examples.

4.2 Numerical examples. In this section we will describe our experiences in applying various splitting schemes to two very different types of problems. The first problem is one of overflowing queues described in [10]; the second concerns tandem queues in a packet switch network and is described in [9].

In Problem (I) traffic is offered by s independent streams to s groups of servers with some overflow capability. In this paper we will concentrate on the case when $s=3$. We will assume that there are n_l, $l=1,2,3$ servers in the primary, secondary, and tertiary groups and demands arrive with Poisson arrival rates λ_l, $l=1,2,3$, and all demands are satisfied at an exponential service rate with mean rate μ . If $p_{i,j,k}$ is

the steady state probability that there are i demands in the primary, j demands in the secondary and k demands in the tertiary, then the Kolmogorov equations are

$$
\begin{aligned}
(4.10)\quad & (\lambda_1(1-\delta_{i,n_1}\delta_{j,n_2}\delta_{k,n_3}) + \lambda_2(1-\delta_{j,n_2}\delta_{k,n_3}) + \lambda_3(1-\delta_{k,n_3}) + (i+j+k)\mu)p_{i,j,k} \\
&= \mu(i+1)(1-\delta_{i,n_1})p_{i+1,j,k} + \mu(j+1)(1-\delta_{j,n_2})p_{i,j+1,k} + \mu(k+1)(1-\delta_{k,n_3})p_{i,j,k+1} \\
&\quad + (1-\delta_{j,0})(\lambda_1\delta_{i,n_1}+\lambda_2)p_{i,j-1,k} + (1-\delta_{k,0})(\lambda_1\delta_{i,n_1}+\lambda_2\delta_{j,n_2}+\lambda_3)p_{i,j,k-1} \\
&\quad + \lambda_1(1-\delta_{i,0})p_{i-1,j,k}.
\end{aligned}
$$

These equations indicate that a demand arriving at the $group_i, i=1$ or 2, is served there unless all n_i servers are busy, in which case it is served by the next group. A demand arriving at the third group is served there, unless all n_3 servers are busy, in which case it is lost.

The matrix A looks like the standard seven point operator that one encounters while discretizing an elliptic partial differential equation in three dimensions. Since the matrix is block tridiagonal in which the diagonal blocks are also block tridiagonal and are almost separable, one is faced with many options for the M-matrix in (4.1). One could use point Gauss-Seidel and set M to the lower triangular portion and the diagonal of A. One could use line Gauss-Seidel and set M to the lower triangular portion, the main diagonal, and the first superdiagonal. Thirdly, one could take advantage of the algebraic structure and partition the problem as

$$
(4.11)\qquad \begin{pmatrix} A_{11} & A_{12} & A_{13} & 0 \\ A_{21} & A_{22} & 0 & A_{24} \\ A_{31} & 0 & A_{33} & A_{34} \\ 0 & A_{42} & A_{43} & A_{44} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{pmatrix} = \mathbf{0}
$$

where A_{11} is separable and $\mathbf{p}_1$ corresponds to $i=0,1,\ldots,n_1-1, j=0,1,\ldots,n_2-1$, and $k=0,1,\ldots,n_3$; A_{22} is separable and $\mathbf{p}_2$ corresponds to $i=n_1, j=0,1,\ldots,n_2-1$, and $k=0,1,\ldots,n_3$; A_{33} is separable and $\mathbf{p}_3$ corresponds to $i=0,1,\ldots,n_1-1, j=n_2$, and $k=0,1,\ldots,n_3$; and A_{44} is tridiagonal and $\mathbf{p}_4$ corresponds to $i=n_1, j=n_2$, and $k=0,1,\ldots,n_3$. Moreover A_{21}, A_{31}, A_{42} and A_{43} have at most one nonzero element per row and A_{13}, A_{24}, A_{34}, A_{12} have at most one nonzero per column.

If M represented the lower triangular and diagonal blocks of (4.11) and one used the separability of the diagonal blocks as in section 3.2, the following block Gauss-Seidel scheme is reasonable:

Until convergence
Solve $A_{11}\mathbf{p}_1^{(k)} = -A_{12}\mathbf{p}_2^{(k-1)} - A_{13}\mathbf{p}_3^{(k-1)}$ for $\mathbf{p}_1^{(k)}$
Solve $A_{22}\mathbf{p}_2^{(k)} = -A_{21}\mathbf{p}_1^{(k)} - A_{24}\mathbf{p}_4^{(k-1)}$ for $\mathbf{p}_2^{(k)}$
Solve $A_{33}\mathbf{p}_3^{(k)} = -A_{31}\mathbf{p}_1^{(k)} - A_{34}\mathbf{p}_4^{(k-1)}$ for $\mathbf{p}_3^{(k)}$
Solve $A_{44}\mathbf{p}_4^{(k)} = -A_{42}\mathbf{p}_2^{(k)} - A_{43}\mathbf{p}_3^{(k)}$ for $\mathbf{p}_4^{(k)}$.

Because of the separability of A_{11}, A_{22}, and A_{33}, solving linear systems with these huge matrices reduces to a number of matrix by matrix multiplications as we shall now show.

Let $T(n,a_j,b_j,c_j)$ denote an $n\times n$ tridiagonal matrix with a_j as its j^{th} subdiagonal element, b_j as its j^{th} diagonal element and c_j as its j^{th} superdiagonal element. Let

$$T_1 = T(n_3+1,-\lambda_3,\lambda_3(1-\delta_{j,n_3+1})+(j-1)\mu,-j\mu),$$

$$T_2 = T(n_2,-\lambda_2,\lambda_2+(j-1)\mu,-j\mu),$$

$$T_3 = T(n_1,-\lambda_1,\lambda_1+(j-1)\mu,-j\mu),$$

$$T_4 = T(n_2,-\lambda_1-\lambda_2,\lambda_1+\lambda_2+n_1\mu+(j-1)\mu,-j\mu),$$

$$T_5 = T(n_3+1,-\lambda_2-\lambda_3,(\lambda_2+\lambda_3)(1-\delta_{j,n_3+1})+n_2\mu+(j-1)\mu,-j\mu).$$

Then

$$A_{11} = I\otimes I\otimes T_1+I\otimes T_2\otimes I+T_3\otimes I\otimes I,$$

$$A_{22} = I\otimes T_1+T_4\otimes I,$$

$$A_{33} = I\otimes T_2+T_5\otimes I.$$

Because the T matrices are tridiagonal with nonpositive off-diagonal elements, there exist Q matrices, which are easily computable, such that

$$Q_jT_jQ_j^{-1} = D_j \quad \text{for } j=1,2,3,4,5 \tag{4.12}$$

where D_j is a diagonal matrix. Thus

$$Q_3\otimes Q_2\otimes Q_1\otimes A_{11}\otimes Q_1^{-1}\otimes Q_2^{-1}\otimes Q_3^{-1} = I\otimes I\otimes D_1+I\otimes D_2\otimes I+D_3\otimes I\otimes I$$

which is a diagonal matrix. Similarly, A_{22} and A_{33} can be transformed into diagonal form.

We considered accelerating the Gauss-Seidel algorithms using the popular SOR technique. If A is written as $A=L+D+U$, where L represents the lower triangular blocks, D the diagonal blocks, and U the upper triangular blocks, then the SOR iteration can be written as

$$\mathbf{p}^{(k+1)} = (rL+D)^{-1}((1-r)D-rU)\mathbf{p}^{(k)} \tag{4.13}$$

which may also be written as

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)}-(L+r^{-1}D)^{-1}A\mathbf{p}^{(k)} \tag{4.14}$$

where r is a relaxation parameter. When $r=1$, SOR reduces to Gauss-Seidel. One can show that if the matrix is q-cyclic and consistently ordered then the best r, r^*, is the unique real root less than $q/(q-1)$ of the equation(see [15], p.109)

$$(\mu r^*)^q = q^q(q-1)^{-(q-1)}(r^*-1) \tag{4.15}$$

where μ is the controlling eigenvalue of the Gauss-Seidel method.

In Table 4.1 we chose one particular problem with $n=1000$ and indicate the sensitivity of point SOR, line SOR, and block SOR to changes in r. Notice that the optimum r value was not the same for the three schemes and that r value which worked well for one scheme would not necessarily work well for another. Also notice that sometimes a small change in r could double the number of iterations. Considering the number of variables, it was surprising how few iterations were required to decrease the residual to 10^{-6}, the convergence criteria. Because the problem was 3-dimensional, for good values of r, the decrease in the number of iterations for line-SOR was not as significant as would be expected for 2-dimensional problems. The

decrease in the number of iterations for the block scheme was certainly significant. Moreover, the block scheme was significantly less sensitive to changes in r.

Table 4.1
Sensitivity of SOR to r when
$\lambda_1=\lambda_2=\lambda_3=6$, $n_1=n_2=n_3=9$, $\mu=1$

	No. of iterations		
r	point SOR	line SOR	block SOR
1.0	118	100	32
1.1	98	82	26
1.2	81	67	20
1.3	67	53	14
1.4	54	40	18
1.5	43	26	24
1.6	35	34	32
1.7	39	57	
1.8	45	106	

Table 4.2
Timings for the SOR schemes on the Cray-1

Problem 1: $n=1000$			
	point SOR	line SOR	block SOR
r	1.6	1.5	1.3
time in sec.	.202	.157	.006+.044
no. of iterations	35	26	14
time per iteration	5.7×10^{-3}	6.0×10^{-3}	3.15×10^{-3}
Problem 2: $n=3375$			
	point SOR	line SOR	block SOR
r	1.8	1.6	1.3
time in sec.	.822	.757	.014+.182
no. of iterations	42	37	18
time per iteration	1.95×10^{-2}	2.04×10^{-2}	1.01×10^{-2}

In Table 4.2 timing information is given for the three schemes for two different problems. The programs were run on the Cray-1 and for the block separable scheme, which involved many matrix by matrix multiplications per iteration, the system provided software was used for the linear algebra tasks. The first problem was the same as that given in Table 4.1. In the second problem $n_1=n_2=n_3=14$, so that $n=3375$, $\mu=1.0$, and $\lambda_1=\lambda_2=\lambda_3=9.0$. For all methods r was varied in increments of .1, and the times are reported for the best r for that problem. Notice that the reported r varied somewhat between the two problems. As would be expected the line method was not significantly more expensive than the point method per iteration. The parallel nature of the block scheme really is evident in the timings. The first number in the time row for the block scheme indicates the startup time required to determine the $Q's$ in (4.12). The second number indicates the time for the iteration portion. If $n_1 = n_2 = n_3 = m$, then point Gauss-Seidel and line Gauss-Seidel require

$9m^3+O(m^2)$ operations per iteration, while the block separable method requires $6m^4+7m^3+O(m^2)$ operations. The operations in the block scheme are highly vectorizable. For problem 1, the block scheme theoretically requires 6 times more multiplications per iteration than the point scheme and for problem 2, 10 times, but as the timings indicate each iteration is about half as expensive as the point scheme. Certainly on a vector machine the block scheme is the preferable method. On a machine that does not vectorize matrix-matrix multiplications, the block scheme would certainly not have enjoyed such a position.

In problem (II) the question of how much should be in M for the Gauss-Seidel method was of secondary consideration. A more fundamental question was the ordering of the states because the Kolmogorov matrix was the sum of a 2-cyclic and a 4-cyclic matrix. Thus consistent ordering was important for the convergence of the method. The problem stemmed from the modeling of a 2-node packet switch network[9]. Each node consisted of three tandem queues, which meant that the problem was 6-dimensional. The state space for each node was the base of a tetrahedron and if one considered the steady state probabilities as given by the 6-dimensional matrix

$$p_{i,j,h,i',j',h'}$$

then for a given b, the size of a shared buffer space, and w, the size of a window buffer,

$$0 \le i+j \le b,$$

$$0 \le i'+j' \le b,$$

$$j \le h \le w,$$

$$j' \le h' \le w.$$

Table 4.3
State count for 2-node packet switch network

buffer size	window size	state count	space required by naive data structure
4	2	961	5625
5	2	2116	11664
6	2	4096	21609
8	4	21025	164025
9	4	34225	250000

Because of the geometry of the state space, the representation of the space in the computer added another facet to the problem. Table (4.3) compares the number of states in the state space for given values of b and w with the number of spaces that would be required if the state space were represented by a six-dimensional matrix with $i,j,i',j' \le b$ and $h,h' \le w$. Obviously one had to be careful in the choice of data structures so that a problem that was of sufficient interest to the modelers could be solved given the storage space available on the machine. Note that most implementations of any iterative scheme require at least one scratch vector of the size of the state space besides that containing the probability vector.

The Kolmogorov equations for problem (II) have the form

$$ap_{i,j,h,i',j',h'} = bp_{i-1,j,h,i',j',h'} + cp_{i,j,h,i'-1,j',h'} + dp_{i+1,j,h,i',j',h'} + ep_{i,j,h,i'+1,j',h'}$$
$$+ fp_{i+1,j-1,h,i',j',h'} + gp_{i,j,h,i'+1,j'-1,h'} + sp_{i,j,h-1,i',j',h'}$$
$$+ tp_{i,j,h,i',j',h'-1} + up_{i,j+1,h+1,i',j',h'} + vp_{i,j,h,i',j'+1,h'+1}.$$

where a,b,c,d,e,f,g,s,t,u,v depend on the indices of p. The zero structure of the A matrix for a very small problem is given in Fig. 4.1 The off-diagonal x's in Fig 4.1 should be considered as diagonal matrices. The diagonal x's should be thought of as matrices having the same zero structure as given in the whole figure. Obviously, the matrix does not have symmetric zero structure and is not reminiscent of problems one usually encounters in modeling in the physical sciences. Along the diagonal one notices tridiagonal blocks of varying sizes. They are caused by local traffic that will not be transferred to the other node in the network. The 4 cycles in the graph are caused by foreign traffic which is ultimately destined for the other node in the network. Before it gets there, the foreign traffic visits several queues while waiting to be processed and sent over the transmission lines. If all the traffic were local, the matrix would be 2-cyclic and the ordering of the elements in the state space would not be of ultimate consideration. If all the traffic were foreign, the matrix would be 4-cyclic and the ordering of the elements in the state space would be highly significant.

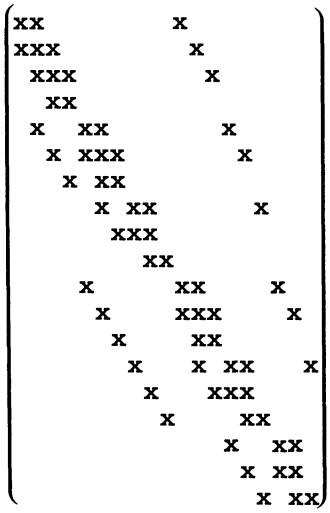


FIG. 4.1

Table 4.4 gives the controlling eigenvalues of point Gauss-Seidel for various traffic mixes and various ordering schemes for a small problem. Scheme A uses the ordering given in (4.5), scheme B uses that in (4.6), scheme C that in (4.7). Note that schemes B and C do not always converge to a single vector. When there is no local traffic, after a while every third iterate for scheme B will be the same and every second iterate for scheme C. The table also indicates the controlling eigenvalues if one had deleted the last equation and pushed the last column of the singular matrix to the right hand side. This always produces a convergent scheme because the coefficient is now a nonsingular M-matrix no matter what the ordering. However,

convergence can be excruciatingly slow. The rate of convergence for the nonsingular iterations is also dependent on the ordering. Since one may think of the nonsingular scheme as a slight perturbation of the singular scheme, if the singular is misordered so that it has an eigenvalue of the T-matrix other than 1 on the unit circle, then there should be an eigenvalue for the nonsingular scheme close to the unit circle retarding convergence. Thus an ordering that would guarantee convergence for the singular scheme would probably improve the convergence rate for the nonsingular scheme. Thus the ordering of the equations may be considered important for both schemes and if one is going to worry about ordering for the nonsingular scheme, one might as well use the singular scheme.

Table 4.4
Convergence behavior for Problem (II)

scheme	percentage of local traffic	controlling eigenvalues singular iterations	controlling eigenvalues nonsingular iterations
A	0.0	$-.24\pm.05i$	.70
A	50.0	$.15\pm.14i$	.89
B	0.0	cube roots of unity	.89 $-.44\pm.77i$
B	50.0	$-.25\pm.66i$	.89 $-.26\pm.63i$
C	0.0	± 1	$\pm.84$
C	50.0	$-.5$	.92

Queuing problems involving tandem queues often give rise to balance equations with nonsymmetric zero structure for which the ordering of the states in the state space is important to the convergence of Gauss-Seidel. Fortunately, in these problems the traffic flow itself often suggests an appropriate ordering. For example, in problem (II) let us set $b=1$ and $w=1$ and consider only the first three indices. The state space will contain the states (0,0,0),(1,0,0),(0,1,0), and (0,1,1). A packet entering the system will visit the states in exactly the order just given and that is exactly the ordering for Scheme A. Any cyclic permutation of the ordering will also give rise to a convergent scheme, but inverting the order of any of the states will not. The point is that the traffic flow itself defines a convergent ordering.

For problem (II), point SOR tended to be highly sensitive to changes in r. It was not unusual that changes of 10% in r would double the number of iterations. Furthermore, this problem was not like those found say in the usual discretization of Laplace's problems in which point SOR will converge for $0\leq r\leq 2$. Often values of $r=1.5$ would cause divergence. Moreover, the optimum value of r would change for different traffic mixes. Using an estimate of the controlling eigenvalue of the Gauss-Seidel method that was determined by the change in iterates, an algorithm was implemented which determined r^* via equation (4.15). The program, which determined r dynamically, tended to require about 20% more iterations than beginning initially with the optimal r^*. We often found that increasing r greatly at first produced very large changes in the iterates and it was not always clear whether the iterates were diverging or whether they would eventually settle down. Thus when trying to estimate r^*, our program incremented the current value slowly.

Chebychev acceleration (See [15]). of Gauss-Seidel was also applied to this problem. Chebychev acceleration entails knowing the location of the foci$(b-a)$ and $(b+a)$ of the ellipse enclosing the eigenvalues of $M^{-1}N$ for the Gauss-Seidel method. In Chebychev acceleration the **p**'s are generated using the following 3-term recurrence relation:

$$\mathbf{p}^{(k+1)}=\delta_{k+1}(\gamma(L+D)^{-1}U\mathbf{p}^{(k)}+(1-\gamma)\mathbf{p}^{(k)})+(1-\delta_{k+1})\mathbf{p}^{(k-1)} \tag{4.16}$$

where

$$\delta_2=2/(2-\sigma^2),$$

$$\delta_{k+1}=(1-(\sigma/2)^2\delta_k)^{-1} \text{ for } k>2$$

and $\gamma=1/b,\sigma=a/b$. As Table 4.5 indicates Chebychev acceleration of Gauss-Seidel seemed to be less sensitive to changes in the parameters for that scheme than SOR. In Table 4.5 the problem had 4096 variables so that convergence was really very fast.

Table 4.5
SOR and Chebychev acceleration

r in SOR	No. of iterations	σ in Chebychev	No. of iterations $\gamma=1.5$	No. of iterations $\gamma=1.0$
1.00	44	.1	44	38
1.05	30	.3	41	35
1.10	27	.5	36	28
1.15	>100	.7	28	33
		.9	60	>100

For problem (II) taking advantage of the tridiagonal blocks on the diagonal did not produce a significant change in the number of iterations when foreign traffic dominated because the matrix was largely 4-cyclic. Moreover, since each row had about 11 nonzeroes, increasing the number of nonzero diagonals in the M matrix from 5 to 6 would not be expected to generally produce grand results. Table 4.6 indicates the behavior of line and point Gauss-Seidel for various traffic mixes.

Table 4.6
Line Gauss-Seidel vs. point Gauss-Seidel

percentage local traffic	No. of iterations point G.S.	No. of iterations line G.S.
10	45	44
30	68	60
50	82	70
70	94	68
90	80	52

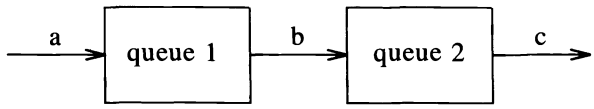
Again the problem had 4096 variables. Because convergence was so swift it was considered unnecessary to consider partitioning A into larger blocks as in problem(I). If we had, the natural partitioning would have grouped together all the queues at one node and each block would have had $\sqrt{n}$ rows. The storage required for the sparse LU decomposition of the diagonal blocks would have overflowed the machine if the LU decomposition had been computed for each block for any problem of say

$n>1000$. The eigendecomposition approach, as used in problem(I), would have been preferable storagewise, for the eigenvectors for most of the diagonal blocks would have been the same. However, the overhead for initially computing the eigenvectors, the Q matrices, would probably require more time than doing a point SOR scheme because now the matrices are not similar to tridiagonal symmetric matrices.

4.3 Implementation advice. In general because the queuing problem statement describes each column of the A matrix, a column orientation of an implementation of an iterative scheme is easier to implement than a row orientation. Unfortunately, a column orientation usually entails storing another scratch vector. From our experience, equation (4.14) is the easiest form of SOR to work with because it naturally lends itself to a column orientation. Equation (4.14) requires 50% more work per iteration than the standard row oriented version of SOR, but for this additional effort, one receives the residual, $A\mathbf{p}$. Because it is possible that the convergence will be so slow that the difference between iterates will be small when one is not near the solution, it is much safer to compute the residual and use it as a convergence criteria.

We have also found that it is much more economical to store the diagonal of the A matrix if there is sufficient space rather than to regenerate it each iteration. Because of the regularity of the off-diagonal elements, it is rarely necessary to store them.

It is not that difficult to detect if the problem is inconsistently ordered and to correct the situation. First of all, if the transition matrix describing the flow in the state space changes in only 1 dimension at a time, then the problem is 2-cyclic and is consistently ordered. Difficulties arise when there are tandem queues and hence the A matrix is not 2-cyclic. Let us assume we have the following queue situation



with k spaces in the first queue and m spaces in the second queue. Let $p_{i,j}$ denote the probability of having i spaces in queue 1 occupied and j spaces in queue 2 occupied. Transitions between states may be specified by the following rules:

(4.17)

if $i<k$ $(i,j) \rightarrow (i+1,j)$ at rate a
if $i>0$ and $j<m$ $(i,j) \rightarrow (i-1,j+1)$ at rate b
if $j>0$ $(i,j) \rightarrow (i,j-1)$ at rate c

The following algorithm for numbering the columns of the A matrix is very natural, but unfortunately leads to an inconsistent ordering

```
L←0
FOR I = 0 TO K
   FOR J = 0 TO M
      {
       L←L+1
       COLUMN(L) CORRESPONDS TO STATE (I,J)
      }
```

If one set m and k both to 1, the A matrix would have the following zero structure:

$$\begin{pmatrix} x & x & & \\ & x & x & \\ x & & x & x \\ & x & & x \end{pmatrix} \tag{4.18}$$

which is obviously not consistently ordered.

If the algorithm had been rearranged to

```
L←0
FOR J = 0 TO M
   FOR I = 0 TO K
     {
      L←L+1
      COLUMN(L) CORRESPONDS TO STATE (I,J)
     }
```

the matrix A would have been consistently ordered and would have had the following zero structure for $m=1$ and $k=1$:

$$\begin{pmatrix} x & & x & \\ x & x & & x \\ & x & x & \\ & & x & x \end{pmatrix}. \tag{4.19}$$

Thus one would suspect that the innermost loop should correspond to the first queue in the line and the outermost to the last queue. As mentioned earlier, the traffic pattern itself dictates the ordering. Unfortunately, the situation is not always that simple. It is not unusual for the $p_{i,j}$ to represent the probability that there are i spaces *available* in queue 1 and j spaces *available* in queue 2. The following rules would then govern the traffic:

if $i>0$ $(i,j) \rightarrow (i-1,j)$ at rate a
if $i<k$ and $j>0$ $(i,j) \rightarrow (i+1,j-1)$ at rate b
if $j<m$ $(i,j) \rightarrow (i,j+1)$ at rate c

As far as the zero structure of the A matrix is concerned, i and j have switched roles from the previous case. Thus the ordering of the states which previously had led to an inconsistent ordering of the A matrix would now lead to a consistently ordered matrix and vice versa. Moreover the following ordering scheme would also lead to a consistently ordered A matrix:

```
L←0
FOR J = M TO STEP BY -1 TO 0
   FOR I = K TO STEP BY -1 TO 0
     {
      L←L+1
      COLUMN(L) CORRESPONDS TO STATE (I,J)
     }
```

Thus if one makes sure that the innermost loop index corresponds to the first queue that traffic enters, one should also make sure that the loop index is running in the correct direction. If traffic entering the queue causes the state space index to increase, the loop should be run forward. If traffic entering the queue causes the state space index to decrease, as in the last example, the loop should be run backwards. It is not completely unusual for someone modeling a system to think of space "occupied" in one queue and space "available" in another. In fact the whole question of consistent ordering was brought to this author's attention by just such a problem.

For most people with queuing problems specifying the matrix is a major problem. Even when the iteration scheme converges, most people still are uncertain whether the singular matrix accurately reflects their model. Often they have been forced to solve the Kolmogorov equations explicitly because their model is not simple and it is not easy to verify that they have encoded their model correctly. The data structure used to represent the state space can sometimes make the task more difficult. Sometimes the geometry of the state space has necessitated a complicated translation of the state space into an array which can be squeezed into the machine. If the state space is a generalization in many dimensions of a tetrahedron, it is not unusual for the user to decide to store only the admissible states in the system and to store them in one long array. The programming language also might impose a complicated data structure. For example, in FORTRAN, the maximum array dimension is 7 and if one has an 8-dimensional state space, a bit of computation is required to access the correct elements in the space. The fact that a queuing problem is usually relatively simple to specify but might be difficult for the ordinary modeler to solve suggests that a compiler be written that takes as input some specification of a queuing problem and creates a program to solve the problem. The compiler would remove from the user the task of dealing with a data structure and deciding the ordering of the columns of the matrix.

In conclusion, we have shown that some of the techniques that have been used to solve nonsingular M-matrix problems may be applied to solving singular problems that have arisen during the modeling of queuing networks. We have solved problems with as many as 35,000 variables on relatively small machines. Some care must be taken in ordering the equations to insure convergence of some of the methods but it is not difficult to determine this ordering. Taking advantage of the algebraic structure of the singular matrix may sometimes be cost effective and the concept of separability that is utilized with great success in [4] can also be applied here. We have shown that in many cases, if a Gauss-Seidel like algorithm converges for the singular system, it will probably be faster than applying the same scheme to the problem in which the singularity has been deleted. Lastly, it should be mentioned that many of the people who design networks and wish to model them are not extremely conversant with the concepts that are basic to numerical linear algebra. They are hardly as sophisticated mathematically as the typical physicist who wishes to solve a large linear system that has arisen during the solution of a partial differential equation. Thus any means of presenting a solution technique in terms of his language is encouraged.

5. Acknowledgment. The author wishes to acknowledge the helpful comments of D.J. Rose, R.E. Funderlic and the referees.

REFERENCES

[1] R. E. BANK, *Marching algorithms for elliptic boundary value problems. II. the variable coefficient case,* SIAM J. Numer. Anal, 14 (1977), pp. 950-970.

[2] A. BERMAN AND R. J. PLEMMONS, *Nonnegative Matrices in the Mathematical Sciences,* Academic Press, New York, 1979.

[3] J. L. BUONI, M.NEUMANN, AND R. S. VARGA, *Theorems of Stein-Rosenberg type III. the singular case,* Linear Algebra and its Applications, 42 (1982), pp. 183-198.

[4] B. L. BUZBEE, G. H. GOLUB, AND C. W. NIELSON *On direct methods for solving Poisson's equations,* SIAM J. Numer. Anal., 7 (1970), pp. 627-656.

[5] I. DUFF AND J. K. REID, *Some design features of a sparse matrix code,* ACM Transactions on Mathematical Software, 5 (1979),pp. 18-36.

[6] 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, Inc., New York, 1976.

[7] R.E. FUNDERLIC AND J. B. MANKIN, *Solution of homogeneous systems of linear equations arising from compartmental models,* this Journal, 2 (1981), pp. 375-383.

[8] W. J. HARROD AND R. J. PLEMMONS, *Comparison of Some Direct Methods for Computing Stationary Distributions of Markov Chains,* N.C. State University Departments of Mathematics and Computer Science Technical Report, 1982.

[9] L. KAUFMAN, B. GOPINATH, AND E. F. WUNDERLICH, *Analysis of packet network congestion control using sparse matrix algorithms,* IEEE Transactions on Communications, 4 (1981), pp. 453-465.

[10] L. KAUFMAN, J. B. SEERY, AND J. A. MORRISON, *Overflow models for dimension PBX feature packages,* The Bell System Technical Journal, 5 (1981), pp. 661-676.

[11] M. NEUMANN AND R. J. PLEMMONS, *Convergent nonnegative matrices and iterative methods for consistent linear systems,* Numer. Math., 31 (1978), pp. 265-279.

[12] M. F. NEUTS, *Algorithmic Methods in Probability,* North-Holland, Amsterdam, 1977.

[13] C. PAIGE, G. STYAN, AND P. WACHTER, *Computation of the stationary distribution of a Markov chain,* J. Stat. Comp. and Simul., 4 (1975), pp.173-186.

[14] R. D. SKEEL, *Scaling for numerical stability in Gaussian elimination,* Journal of the ACM, 26 (1979), pp.494-526.

[15] R. S. VARGA, *Matrix Iterative Analysis,* Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[16] J. H. WILKINSON, *The Algebraic Eigenvalue Problem,* The Clarendon Press, Oxford, 1965.

[17] D. M. YOUNG, *Iterative Solution of Large Linear Systems,* Academic Press, New York, 1972.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 3, September 1983

0196-5204/83/0403-0009/$01.25/0

COMPUTING A TRUST REGION STEP*

JORGE J. MORÉ AND D. C. SORENSEN †

Abstract. We propose an algorithm for the problem of minimizing a quadratic function subject to an ellipsoidal constraint and show that this algorithm is guaranteed to produce a nearly optimal solution in a finite number of iterations. We also consider the use of this algorithm in a trust region Newton's method. In particular, we prove that under reasonable assumptions the sequence generated by Newton's method has a limit point which satisfies the first and second order necessary conditions for a minimizer of the objective function. Numerical results for GQTPAR, which is a Fortran implementaton of our algorithm, show that GQTPAR is quite successful in a trust region method. In our tests a call to GQTPAR only required 1.6 iterations on the average.

Key words. Newton's method, trust region, ellipsoidal constraint, global convergence

1. Introduction.

In an important class of minimization algorithms called "trust region methods" (see, for example, Sorensen [1982]), the calculation of the step between iterates requires the solution of a problem of the form

$$\min\{\psi(w) : \|w\| \leqslant \Delta\} \tag{1.1}$$

where Δ is a positive parameter, $\|\cdot\|$ is the Euclidean norm in R^n, and

$$\psi(w) \equiv g^T w + \tfrac{1}{2} w^T B w, \tag{1.2}$$

with $g \in R^n$, and $B \in R^{n \times n}$ a symmetric matrix. The quadratic function ψ generally represents a local model to the objective function defined by interpolatory data at an iterate and thus it is important to be able to solve (1.1) for any symmetric matrix B; in particular, for a matrix B with negative eigenvalues.

In trust region methods it is sometimes helpful to include a scaling matrix for the variables. In this case, problem (1.1) is replaced by

$$\min\{\psi(v) : \|Dv\| \leqslant \Delta\} \tag{1.3}$$

where $D \in R^{n \times n}$ is a nonsingular matrix. The change of variables $Dv = w$ shows that problem (1.3) is equivalent to

$$\min\{\hat{\psi}(w) : \|w\| \leqslant \Delta\} \tag{1.4}$$

where $\hat{\psi}(w) \equiv \psi(D^{-1}w)$, and that the solutions of problems (1.3) and (1.4) are related by $Dv = w$. Because of this equivalence, we only consider problem (1.1). Also note that if, as is usually the case, D is a diagonal matrix, then it is easy to explicitly carry out the change of variables and solve problem (1.4).

The use of a trust region method in a nonlinear optimization problem requires the solution of many problems of type (1.1). These problems do not usually require accurate solutions, but in all cases we must be able to find an approximate solution with a reasonable amount of computational effort, and the approximate solution found must guarantee that the trust region method has the

* Received by the editors December 16, 1981 and in revised form July 15, 1982.

† Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439. Work supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-Eng-38.

right sort of convergence properties. In this paper we are concerned with these two issues; namely, robust and stable algorithms for the solution of (1.1) and the impact of these algorithms on the convergence properties of trust region methods.

Goldfeld, Quandt, and Trotter [1966], Hebden [1973], Fletcher [1980], Gay [1981], and Sorensen [1982], have discussed (1.1) in connection with trust region methods. Their algorithms are based on the fact that if (1.1) has a solution on the boundary of $\{w : \|w\| \leq \Delta\}$ then, in most cases, a solution of (1.1) can be found by determining $\lambda \geq 0$ such that $B + \lambda I$ is positive definite and

$$\|(B + \lambda I)^{-1} g\| = \Delta \tag{1.5}$$

In one case - the hard case - equation (1.5) has no solution with $B + \lambda I$ positive definite, and this leads to numerical difficulties. Hebden [1973] proposed an algorithm for the solution of (1.1) which is basically sound except for its treatment of the hard case. Gay [1981] improved Hebden's scheme and showed that under certain conditions the approximate solution determined by his algorithm is nearly optimal. His algorithm, however, may require a large number of iterations in the hard case.

We propose an algorithm for the solution of (1.1) which is guaranteed to produce a nearly optimal solution in a finite number of steps. Specifically, given parameters σ_1 and σ_2 in (0,1), the approximate solution s satisfies

$$\psi(s) - \psi^* \leq \sigma_1(2-\sigma_1)\max\{|\psi^*|, \sigma_2\}, \quad \|s\| \leq (1+\sigma_1)\Delta,$$

where ψ^* is the optimal value of (1.1). We also consider the use of our algorithm in a trust region Newton's method. In particular, we prove that under reasonable assumptions the sequence $\{x_k\}$ generated by Newton's method has a limit point x^* which satisfies the first and second order necessary conditions for a minimizer of the objective function f. Numerical results for GQTPAR, which is a Fortran implementation of our algorithm, show that GQTPAR is quite successful in a trust region method. In our tests a call to GQTPAR only required 1.6 iterations on the average.

The outline of the paper is as follows. The theoretical basis of an algorithm for the solution of (1.1) is laid out in Section 2, while in Section 3 we present the algorithm and show that the solution generated by the algorithm is nearly optimal. In Section 4 we consider the use of this algorithm in a trust region Newton's method and prove that the combined algorithm has very strong convergence properties. Numerical results are presented in Section 5.

2. Structure of the problem. Problem (1.1) has a tremendous amount of structure and it is important to understand this structure in order to construct a suitable algorithm. The following results expose this structure and provide a theoretical basis for the numerical algorithm. Note that these results provide necessary and sufficient conditions for a point $p \in R^n$ to be a solution to (1.1) and that there is no "gap" between the necessary and sufficient conditions.

LEMMA(2.1). *If p is a solution to* (1.1) *then p is a solution to an equation of the form*

$$(B + \lambda I)p = -g, \tag{2.2}$$

with $B + \lambda I$ positive semidefinite, $\lambda \geq 0$, and $\lambda(\Delta - \|p\|) = 0$.

LEMMA(2.3). *Let* $\lambda \in R$, $p \in R^n$ *satisfy* (2.2) *with* $B + \lambda I$ *positive semidefinite.*

(*i*) *If* $\lambda = 0$ *and* $\| p \| < \Delta$ *then* p *solves* (1.1).

(*ii*) p *solves* $\psi(p) = \min\{\psi(w) : \| w \| = \| p \|\}$.

(*iii*) *If* $\lambda \geqslant 0$ *and* $\| p \| = \Delta$ *then* p *solves* (1.1).

If $B + \lambda I$ *is positive definite then* p *is the only solution to* (1.1).

Simple proofs of these lemmas are given by Sorensen [1982]. Lemma (2.3) is important from a computational standpoint since it provides a perturbation result that is useful in setting termination rules for the iterative method used to solve (1.1).

The solution of (1.1) is straightforward if (1.1) has no solutions on the boundary of $\{w : \| w \| \leqslant \Delta\}$. In fact, (1.1) has no solution p with $\| p \| = \Delta$ if and only if B is positive definite and $\| B^{-1} g \| < \Delta$. To prove this claim, first note that if B is positive definite and $\| B^{-1} g \| < \Delta$ then Lemma (2.3) immediately shows that $p = -B^{-1} g$ is the solution to (1.1). On the other hand, if (1.1) has a solution p with $\| p \| < \Delta$ then Lemma (2.1) shows that $\lambda = 0$ and that B is positive semidefinite. If B were singular then $Bz = 0$ for some z with $\| p + z \| = \Delta$ and then Lemma (2.3) implies that $p + z$ is a solution to (1.1) on the boundary. This contradiction establishes our claim.

Now assume that (1.1) has a solution on the boundary of $\{w : \| w \| \leqslant \Delta\}$. If g is not perpendicular to the eigenspace

$$S_1 \equiv \{z : Bz = \lambda_1 z, \ z \neq 0\}$$

where λ_1 is the smallest eigenvalue of B, then the nonlinear equation $\| p_\alpha \| = \Delta$ where

$$p_\alpha = -(B + \alpha I)^{-1} g$$

has a solution $\lambda \geqslant 0$ in $(-\lambda_1, \infty)$. Moreover, Lemma (2.3) implies that p_λ is the solution of problem (1.1). Reinsch [1967,1971] and Hebden [1973] observed independently that to solve $\| p_\alpha \| = \Delta$ great advantage could be taken of the fact that the function $\| p_\alpha \|^2$ is a rational function in α with second order poles on a subset of the negatives of the eigenvalues of the symmetric matrix B. To see this consider the decomposition

$$B = Q \Lambda Q^T \text{ with } \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n) \text{ and } Q^T Q = I,$$

and observe that

$$\| p_\alpha \|^2 = \| Q(\Lambda + \alpha I)^{-1} Q^T g \|^2 = \sum_{j=1}^{n} \frac{\gamma_j^2}{(\lambda_j + \alpha)^2} \tag{2.4}$$

where γ_i is the ith component of $Q^T g$. In the next section we elaborate on the importance of this observation.

If g is perpendicular to S_1 then the equation $\| p_\alpha \| = \Delta$ may still have a solution $\lambda \geqslant 0$ in $(-\lambda_1, \infty)$, and in this case the solution to (1.1) can be obtained as above. If, however, $\| p_\alpha \| = \Delta$ has no solutions in $(-\lambda_1, \infty)$ then this leads to numerical difficulties. We call this situation the "hard case".

A characteristic difficulty of the hard case is that $\| p_\alpha \| < \Delta$ whenever

$B + \alpha I$ is positive definite. For example, if

$$B = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad g = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

then $\lambda_1 = -1$, and if $B + \alpha I$ is positive definite then $\| p_\alpha \|^2 \leqslant \frac{1}{2}$. In the hard case, a solution to (1.1) can be obtained by solving

$$(B - \lambda_1 I)p = -g$$

for p with $\| p \| \leqslant \Delta$ and by determining an eigenvector $z \in S_1$. Then

$$(B - \lambda_1 I)(p + \tau z) = -g$$

and $\| p + \tau z \| = \Delta$ for some τ. Lemma (2.3) now shows that $p + \tau z$ solves (1.1).

3. The algorithm. Consider the solution of $\| p_\alpha \| = \Delta$. The rational structure of $\| p_\alpha \|^2$ may be exploited by applying Newton's method to the zero finding problem

$$\phi(\alpha) \equiv \frac{1}{\Delta} - \frac{1}{\| p_\alpha \|} = 0. \tag{3.1}$$

Newton's method is very efficient when applied to (3.1) since ϕ is almost linear on $(-\lambda_1, \infty)$. Moreover, the computation of the Cholesky factorization of $B + \alpha I$ makes it possible to compute the necessary derivative whenever $\alpha \in (-\lambda_1, \infty)$. There is no need to compute the eigensystem of B as suggested by Goldfeld, Quandt, and Trotter [1966]. The following algorithm updates λ by Newton's method applied to (3.1).

ALGORITHM (3.2). *Let $\lambda \geqslant 0$ with $B + \lambda I$ positive definite and $\Delta > 0$ be given.*

1) *Factor* $B + \lambda I = R^T R$;

2) *Solve* $R^T R p = -g$;

3) *Solve* $R^T q = p$;

4) *Let* $\lambda := \lambda + \left(\dfrac{\| p \|}{\| q \|} \right)^2 \left(\dfrac{\| p \| - \Delta}{\Delta} \right)$;

In this algorithm $R^T R$ is the Cholesky factorization of $B + \lambda I$ with $R \in R^{n \times n}$ upper triangular. Although not represented in this simplified version, it is necessary to safeguard λ in order to obtain a positive definite $B + \lambda I$ and guarantee convergence. These safeguards, and the convergence criteria for this algorithm are discussed later on in this section.

If properly safeguarded, the iteration produced by Algorithm (3.2) is sufficiently rapid to solve most problems of type (1.1) expected in practice. However, in the hard case this scheme may require a large number of iterations to converge; in particular, if $g = 0$ then Algorithm (3.2) breaks down. In the hard case it is necessary to supply a vector z that is an approximate eigenvector of B corresponding to λ_1. Indeed, as pointed out at the end of Section 2, in the hard case a solution to (1.1) is

$$p = -(B - \lambda_1 I)^{\dagger} g + \tau z, \tag{3.3}$$

where the superscript $\dagger$ denotes the Moore-Penrose generalized inverse, $z \in S_1$, and τ is chosen so that $\| p \| = \Delta$. Note that actual computation of the two orthogonal components of p indicated in (3.3) may require more computational effort than is reasonable in the context of an optimization algorithm. Moreover, recognizing that a solution of the form (3.3) is required may also be time consuming. Fortunately, there is a completely acceptable alternative. The following result provides a foundation for this technique.

LEMMA (3.4). *Let* $0 < \sigma < 1$ *be given and suppose that*

$$B + \lambda I = R^T R \ , \ (B + \lambda I)p = -g \ , \lambda \geqslant 0.$$

Let $z \in R^n$ *satisfy*

$$\| p + z \| = \Delta \ , \ \| Rz \|^2 \leqslant \sigma (\| Rp \|^2 + \lambda \Delta^2). \tag{3.5}$$

Then

$$-\psi (p+z) \geqslant \tfrac{1}{2}(1-\sigma)(\| Rp \|^2 + \lambda \Delta^2) \geqslant (1-\sigma)|\psi^*| \ ,$$

where ψ^* *is the optimal value of* (1.1).

Proof. First note that for any $z \in R^n$,

$$\psi (p+z) = -\tfrac{1}{2}(\| Rp \|^2 + \lambda \| p+z \|^2) + \tfrac{1}{2} \| Rz \|^2. \tag{3.6}$$

Then for any z which satisfies (3.5),

$$-\psi (p+z) \geqslant \tfrac{1}{2}(1-\sigma)(\| Rp \|^2 + \lambda \Delta^2).$$

Moreover, if $\psi^* = \psi (p+z^*)$ where $\| p + z^* \| \leqslant \Delta$, then (3.6) implies

$$-\psi (p+z^*) \leqslant \tfrac{1}{2}\left[\| Rp \|^2 + \lambda \Delta^2\right].$$

The last two inequalities yield Lemma (3.4). ■

A consequence of Lemma (3.4) is that $|\psi (p+z) - \psi^*| \leqslant \sigma |\psi^*|$. This shows that if (3.5) holds then $p + z$ is a nearly optimal solution to problem (1.1).

A consequence of the proof of Lemma (3.4) is equation (3.6). This expression is quite useful and will be used throughout this section.

Gay [1981] has a result similar to Lemma (3.4) but the assumptions are stronger than those in Lemma (3.4). Instead of (3.5), Gay's assumptions imply that $\max\{\| p \|, \| z \|\} < \Delta$ and that

$$\| p + z \| = \Delta \ , \ \| (B + \lambda I)z \| \leqslant \left[\frac{\sigma}{\Delta}\right](\| Rp \|^2 + \lambda \| p \|^2). \tag{3.7}$$

Since $\| p \| < \Delta$ and $\| z \| < \Delta$, it is not difficult to show that (3.7) implies (3.5). A weakness of (3.7) is that in the hard case it may be a severe restriction on λ. This claim can be made precise by first noting that (3.7) implies that

$$|\lambda + \lambda_1| \| z \| \leqslant \left[\frac{\sigma}{\Delta}\right](\| R \|^2 + \lambda) \| p \|^2 \ .$$

Now, when $\| p \| \leqslant \epsilon \Delta$ for some ϵ in (0,1) then $\| z \| \geqslant (1-\epsilon)\Delta$ and thus

$$|\lambda + \lambda_1| \leqslant \left[\frac{\sigma}{1-\epsilon}\right](\| R \|^2 + \lambda)\epsilon^2 \ .$$

Since ϵ can be quite small (specially if g is small) in the hard case, this estimate shows that if ϵ is small then λ must be very close to $-\lambda_1$ before Gay's assumptions are satisfied. As a consequence, an algorithm based on (3.7) may require a large number of iterations in the hard case. Note that this weakness is not shared by (3.5).

The main use of Lemma (3.4) is in the hard case. In this situation we have $\lambda \geq 0$ with $B + \lambda I$ positive definite and the solution p of (2.2) satisfies $\| p \| < \Delta$. We can then attempt to satisfy (3.5) with $z = \tau \hat{z}$ by letting τ satisfy $\| p + \tau \hat{z} \| = \Delta$ and by choosing $\hat{z}$ with $\| \hat{z} \| = 1$ such that $\| R\hat{z} \|$ is as small as possible.

Given $\hat{z}$ with $\| \hat{z} \| = 1$ and p with $\| p \| < \Delta$ there are usually two choices of τ which satisfy $\| p + \tau \hat{z} \| = \Delta$, and equation (3.6) implies that the choice with the smaller magnitude minimizes $\psi(p + \tau \hat{z})$. This choice is

$$\tau = \frac{\Delta^2 - \| p \|^2}{p^T \hat{z} + \operatorname{sgn}(p^T \hat{z}) [(p^T \hat{z})^2 + (\Delta^2 - \| p \|^2)]^{1/2}} .$$

A vector $\hat{z}$ with $\| \hat{z} \| = 1$ such that $\| R\hat{z} \|$ is as small as possible can be obtained with the LINPACK technique (Cline, Moler, Stewart, and Wilkinson [1979]) for estimating the smallest singular value of a triangular matrix R. In this technique a vector e with components ± 1 is selected so that the solution w to the system $R^T w = e$ is large. Essentially the idea is to select the sign of the components of e to cause maximum local growth of w as the forward substitution proceeds. Then the system $Rv = w$ is solved for v. The vector v has the property that if

$$\hat{z} \equiv \frac{v}{\| v \|},$$

then $\| R\hat{z} \|$ is usually close to the smallest singular value of R. In the appendix to this paper we show that with the LINPACK technique, $\| R\hat{z} \|$ is close to zero if λ is near $-\lambda_1$. This property guarantees that (3.5) is satisfied by $z = \tau \hat{z}$ when λ is sufficiently close to $-\lambda_1$. The LINPACK technique is attractive because it is computationally inexpensive (roughly n^2 arithmetic operations) and quite reliable, but there are other possibilities, for example, the algorithms of Cline, Conn, and Van Loan [1982]. We emphasize that the only property of $\hat{z}$ required by our algorithm is that $\| R\hat{z} \|$ approach zero as λ approaches $-\lambda_1$.

An important ingredient of the iteration is the safeguarding required to ensure that a solution is found. The safeguarding depends on the fact that ϕ is convex and strictly decreasing on $(-\lambda_1, \infty)$. This fact was discovered by Reinsch [1971] and follows from (2.4). It implies that Newton's method started from $\lambda \in (-\lambda_1, \infty)$ with $\phi(\lambda) > 0$ produces a monotonically increasing sequence converging to the solution of $\phi(\alpha) = 0$. In addition, if $\lambda \in (-\lambda_1, \infty)$ and $\phi(\lambda) < 0$ then the next Newton iterate λ_+ is such that either $\lambda_+ \leq -\lambda_1$, or $\phi(\lambda_+) \geq 0$.

The safeguarding scheme uses parameters λ_L, λ_U, and λ_S such that $[\lambda_L, \lambda_U]$ is an interval of uncertainty which contains the desired λ, and λ_S is a lower bound on $-\lambda_1$.

Safeguard λ :

1) $\lambda := \max(\lambda, \lambda_L)$;

2) $\lambda := \min(\lambda, \lambda_U)$;

3) *If* $\lambda \leqslant \lambda_S$ *then* $\lambda := \max(0.001\lambda_U, (\lambda_L\lambda_U)^{1/2})$;

Safeguarding schemes of this type have been used by Moré [1978] for the case in which B is positive semidefinite and by Gay [1981] for the general case. The first two steps of the safeguarding scheme ensure that $\lambda \in [\lambda_L, \lambda_U]$ while the third step guarantees that the length of the interval of uncertainty is reduced. The third step is crucial; if the length of the interval of uncertainty remains bounded away from zero then the third step can only be executed a finite number of times. This last point will become clear once we set down the rules for updating the safeguarding parameters.

Given λ_S and a trial λ, the rules for revising λ_S are as follows. If $\lambda \in (-\lambda_1, \infty)$ and $\phi(\lambda) < 0$ then $\|p_\lambda\| < \Delta$ and we can compute τ and $\hat{z}$ as described above and set

$$\lambda_S := \max(\lambda_S, \lambda - \|R\hat{z}\|^2). \tag{3.8}$$

Since R is the Cholesky factor of $B + \lambda I$, it follows that for any $\hat{z}$ such that $\|\hat{z}\| = 1$ we have

$$\lambda - \|R\hat{z}\|^2 \leqslant -\lambda_1.$$

Thus, if λ_S is a lower bound on $-\lambda_1$ then (3.8) guarantees that the updated λ_S is also a lower bound on $-\lambda_1$. If $\lambda \leqslant -\lambda_1$ then we can update λ_S by noting that during the Cholesky decomposition of $B + \lambda I$ it is possible to find $\delta \geqslant 0$ such that the leading submatrix of order $l \leqslant n$ of

$$B + \lambda I + \delta e_l e_l^T$$

is singular. In addition, it is possible to determine $u \in R^n$ such that

$$(B + \lambda I + \delta e_l e_l^T)u = 0$$

with $u_l = 1$ and $u_i = 0$ for $i > l$. It follows that

$$\lambda_S := \max\left(\lambda_S, \lambda + \frac{\delta}{\|u\|^2}\right) \tag{3.9}$$

is a lower bound on $-\lambda_1$.

Gay [1981] proposed updating λ_S via (3.9) but (3.8) is new. Since $\|R\hat{z}\|^2$ is usually close to $\lambda + \lambda_1$, updating λ_S via (3.8) tends to avoid trial λ for which $B + \lambda I$ is not positive definite and thus reduces the number of iterations required for convergence.

The rules for updating λ_L and λ_U are fairly simple; they are presented in the following summary of the updating rules for the safeguarding parameters.

Update $\lambda_L, \lambda_U, \lambda_S$:

1) *If* $\lambda \in (-\lambda_1, \infty)$ *and* $\phi(\lambda) < 0$ *then*

$\lambda_U := \min(\lambda_U, \lambda)$;

else

$\lambda_L := \max(\lambda_L, \lambda)$;

2) *Update* λ_S *using* (3.8) *and* (3.9);

3) *Let* $\lambda_L := \max(\lambda_L, \lambda_S)$;

Initial values for the safeguarding parameters are

$$\lambda_S = \max\{-\beta_{ii}\}$$

where β_{ii} is the ith diagonal element of B, and

$$\lambda_L = \max\left(0, \lambda_S, \frac{\|g\|}{\Delta} - \|B\|_1\right),$$

$$\lambda_U = \frac{\|g\|}{\Delta} + \|B\|_1 \ .$$

These choices of λ_L and λ_U are similar to those used by Gay [1981]. They are based upon the observation that (2.4) implies that

$$\frac{\|g\|}{|\lambda + \lambda_n|} \leqslant \|p_\lambda\| \leqslant \frac{\|g\|}{|\lambda + \lambda_1|}, \quad -\lambda_1 < \lambda,$$

where λ_1 and λ_n are, respectively, the smallest and largest eigenvalues of B. Other choices of initial values are possible, but these are simple and are particularly effective for large values of $\|g\|/\Delta$.

The final ingredient of the iteration is the convergence criteria. The idea is to terminate the iteration with a nearly optimal solution of problem (1.1). Given σ_1 and σ_2 in (0,1), and a trial $\lambda \geqslant 0$ such that $B + \lambda I$ is positive definite, a vector p is computed as in Algorithm (3.2). If

$$|\Delta - \|p\|| \leqslant \sigma_1\Delta, \text{ or } \|p\| \leqslant \Delta, \ \lambda = 0, \tag{3.10}$$

then the algorithm terminates with $s = p$ as an approximate solution. The hard case is taken into account by computing p and $\hat{z}$ whenever $\|p\| < \Delta$, and if

$$\|R(\tau\hat{z})\|^2 \leqslant \sigma_1(2-\sigma_1)\max\{\sigma_2, \|Rp\|^2 + \lambda\Delta^2\} \tag{3.11}$$

then the algorithm terminates with $s = p + \tau\hat{z}$ as the approximate solution.

An additional subtlety of the convergence tests is that if both (3.10) and (3.11) are satisfied then we choose the approximate solution s for which $\psi(s)$ is least. This is easy to do because (3.6) shows that $\psi(p+\tau\hat{z}) \leqslant \psi(p)$ if and only if

$$\|R(\tau\hat{z})\|^2 \leqslant \lambda(\Delta^2 - \|p\|^2).$$

This subtlety is not theoretically necessary but is nice to have from a computational point of view. Also note that the factor of $\sigma_1(2-\sigma_1)$ in (3.11) is needed so that in each case $\psi(s)$ satisfies a bound of the same form. This is made clear in the discussion that follows.

We now show that (3.10) and (3.11) guarantee that if the algorithm terminates then the approximate solution s satisfies

$$\psi(s) - \psi^* \leqslant \sigma_1(2-\sigma_1)\max\{|\psi^*|, \sigma_2\}, \ \|s\| \leqslant (1+\sigma_1)\Delta, \tag{3.12}$$

and thus s is a nearly optimal solution of (1.1). First consider (3.11). If $\|Rp\|^2 + \lambda\Delta^2 > \sigma_2$ then the assumptions of Lemma (3.4) are satisfied when σ is replaced by $\sigma_1(2-\sigma_1)$ and hence (3.12) holds for $s = p + \tau\hat{z}$. Now suppose that $\|Rp\|^2 + \lambda\Delta^2 \leqslant \sigma_2$. To establish that (3.12) holds in this case, first note that if $\psi^* = \psi(p+z^*)$ where $\|p+z^*\| \leqslant \Delta$ then (3.6) implies that

$$|\psi^*| = -\psi^* \leqslant \tfrac{1}{2}(\|Rp\|^2 + \lambda\Delta^2) \leqslant \tfrac{1}{2}\sigma_2 \ .$$

We now use this result and (3.6) to obtain that

$$\psi(p+\tau\hat{z}) = -\tfrac{1}{2}(\| Rp\|^2 + \lambda\Delta^2) + \tfrac{1}{2}\| R(\tau\hat{z})\|^2 \leqslant \psi^* + \tfrac{1}{2}\sigma_1(2-\sigma_1)\sigma_2.$$

Hence, (3.12) also holds in this case.

The next result shows that if the algorithm terminates when (3.10) is satisfied then (3.12) holds with $s = p$.

LEMMA(3.13). *Let* $0 < \sigma < 1$ *be given and suppose that*

$$B + \lambda I = R^T R\ ,\ (B+\lambda I)p = -g\ ,\ \lambda \geqslant 0.$$

If ψ^* *is the optimal value of* (1.1) *and if* $\| p\| \geqslant (1-\sigma)\Delta$ *then*

$$-\psi(p) \geqslant \tfrac{1}{2}(1-\sigma)^2(\| Rp\|^2 + \lambda\Delta^2) \geqslant (1-\sigma)^2|\psi^*|.$$

Proof. Just as in the proof of Lemma (3.4), note that (3.6) holds for any $z \in R^n$ and hence,

$$-\psi^* \leqslant \tfrac{1}{2}\left(\| Rp\|^2 + \lambda\Delta^2\right).$$

Moreover, (3.6) with $z = 0$ also implies that

$$-\psi(p) \geqslant \tfrac{1}{2}(1-\sigma)^2(\| Rp\|^2 + \lambda\Delta^2).$$

The last two inequalities yield Lemma (3.13). ■

We have now discussed all the ingredients of the iterative scheme for solving problem (1.1). The following algorithm summarizes these ingredients and defines a typical iteration.

ALGORITHM(3.14).

1) *Safeguard* λ;
2) *If* $B + \lambda I$ *is positive definite then factor* $B + \lambda I = R^T R$; *otherwise go to* 5;
3) *Solve* $R^T R\, p = -g$;
4) *If* $\| p\| < \Delta$, *compute* τ *and* $\hat{z}$;
5) *Update* $\lambda_L, \lambda_U, \lambda_S$;
6) *Check the convergence criteria;*
7) *If* $B + \lambda I$ *is positive definite and* $g \neq 0$ *then update* λ *via steps* 3 *and* 4 *of Algorithm* (3.2); *otherwise update* λ *via* $\lambda := \lambda_S$;

The last step of Algorithm (3.14) deserves some explanation. If $B + \lambda I$ is positive definite and $g \neq 0$ then the Newton iterate of Algorithm (3.2) tends to be a lower bound on $-\lambda_1$ for $\| g\|$ sufficiently small and thus updating λ via λ_S is a reasonable choice when $g = 0$. Also note that setting λ to λ_S forces a safeguarded choice of λ in the next iteration, and that this is a desirable strategy whenever the Newton iteration cannot be used.

We now claim that after a finite number of iterations Algorithm (3.14) produces a $\lambda \in (-\lambda_1, \infty)$ with $\phi(\lambda) \geqslant 0$ or an arbitrarily small interval of uncertainty. If we assume that the length of the interval of uncertainty remains bounded away from zero then the third step of the safeguarding scheme guarantees that $\lambda \leqslant \lambda_S$ only happens a finite number of times. Now, if $\lambda \leqslant -\lambda_1$ then $\lambda \leqslant \lambda_S$ holds on the next iteration. Finally, if $\lambda \in (-\lambda_1, \infty)$ and $\phi(\lambda) < 0$ then recall that the next

Newton iterate λ_+ is such that either $\lambda_+ \leqslant -\lambda_1$, or $\phi(\lambda_+) \geqslant 0$. The above argument shows that $\lambda_+ \leqslant -\lambda_1$ can only happen a finite number of times, so eventually $\lambda_+ \in (-\lambda_1, \infty)$ and $\phi(\lambda_+) \geqslant 0$. This establishes our claim.

The importance of the above claim should be evident; given $\lambda \in (-\lambda_1, \infty)$ with $\phi(\lambda) \geqslant 0$ then Algorithm (3.14) eventually satisfies (3.10), while if the interval of uncertainty is small then R is nearly singular and it is then possible to satisfy (3.11). Thus Algorithm (3.14) terminates in a finite number of iterations with an approximate solution s which satisfies (3.12).

A frequent application of Algorithm (3.14) is to the solution of a sequence of problems of the form (1.1) in which only Δ is changing. In particular, in trust region methods we need to solve a sequence of problems for decreasing values of Δ and then it is possible to improve the initial choice of λ_L. Assume that λ and λ_L are the final values of these parameters for a specific value of Δ. Given a new value $\Delta_+ < \Delta$ then λ_L is still a lower bound for the new problem. Moreover, the convexity and monotonicity of ϕ shows that an update of λ based on a Newton step for

$$\phi_+(\alpha) \equiv \frac{1}{\Delta_+} - \frac{1}{\| p_\alpha \|}$$

is also a lower bound for the new problem. This improvement on the initial choice of λ_L follows a suggestion of Ron Dembo.

One of the differences between Gay's [1981] algorithm and Algorithm (3.14) is that in Gay's algorithm $\lambda = 0$ is always tried first. It is not at all clear that this is a desirable strategy, and it seems preferable to try $\lambda = 0$ first only if the safeguarded λ is zero. Note that if B is positive definite and $\| B^{-1} g \| \leqslant \Delta$ then Algorithm (3.14) terminates in at most two iterations. In fact, if initially $\lambda > 0$ then the convexity and monotonicity of ϕ and the positive definiteness of B guarantee that the next trial λ is zero.

We have already mentioned that another difference is the updating of λ_S via (3.8). A final difference occurs when $g = 0$; Gay's algorithm is not defined in this situation, but Algorithm (3.14) handles this case appropriately.

4. Trust region methods in unconstrained minimization. We now consider the use of Algorithm (3.14) in the context of trust region methods for unconstrained minimization and show how Algorithm (3.14) can be used to produce an efficient and reliable version of Newton's method.

Let $f : R^n \to R$ be a twice continuously differentiable function with gradient ∇f and Hessian $\nabla^2 f$. In Newton's method with a trust region strategy, each iterate x_k has a bound Δ_k such that

$$f(x_k + w) \approx f(x_k) + \psi_k(w), \quad \| w \| \leqslant \Delta_k \ ,$$

where

$$\psi_k(w) = \nabla f(x_k)^T w + \tfrac{1}{2} w^T \nabla^2 f(x_k) w \ .$$

In other words, ψ_k is a model of the reduction in f within a neighborhood of the iterate x_k. This suggests that it may be desirable to compute a step s_k which approximately solves the problem

$$\min\{\psi_k(w) : \| w \| \leqslant \Delta_k\}. \tag{4.1}$$

If the step is satisfactory in the sense that $x_k + s_k$ produces a sufficient reduction in f, then Δ_k can be increased; if the step is unsatisfactory then Δ_k should be decreased.

ALGORITHM (4.2). *Let* $0 < \mu < \eta < 1$ and $0 < \gamma_1 < \gamma_2 < 1 < \gamma_3$ *be specified constants.*

1) *Let* $x_0 \in R^n$ *and* $\Delta_0 > 0$ *be given.*
2) *For* $k = 0,1,2, \cdots$ *until "convergence"*
   a) *Compute* $\nabla f(x_k)$ *and* $\nabla^2 f(x_k)$.
   b) *Determine an approximate solution* s_k *to problem (4.1).*
   c) *Compute* $\rho_k = (f(x_k + s_k) - f(x_k))/\psi_k(s_k)$.
   d) *If* $\rho_k \leqslant \mu$ *then* $\Delta_k := \Delta \in [\gamma_1\Delta_k, \gamma_2\Delta_k]$ *and go to b).*
   e) $x_{k+1} = x_k + s_k$.
   f) *If* $\rho_k \leqslant \eta$ *then* $\Delta_{k+1} \in [\gamma_2\Delta_k, \Delta_k]$ else $\Delta_{k+1} \in [\Delta_k, \gamma_3\Delta_k]$.

This is a basic form of Newton's method which does not include a scaling matrix for the variables. To include a scaling matrix, subproblem (4.1) is replaced by

$$\min\{\psi_k(w) : \| D_k w \| \leqslant \Delta_k\}$$

where D_k is a nonsingular matrix. We shall not discuss this generalization here; however, it is important to note that our results hold if $\{D_k\}$ has uniformly bounded condition numbers.

In this section we are mainly interested in conditions on the approximate solution s_k of problem (4.1) which guarantee that the sequence $\{x_k\}$ generated by Algorithm (4.2) is convergent to a point x^* with $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ positive semidefinite. A minimal requirement on s_k is that there is a $\beta \in (0,1)$ such that

$$-\psi(s_k) \geqslant \beta \max\{-\psi(w) : w = \alpha \nabla f(x_k), \| w \| \leqslant \Delta_k\}, \quad \| s_k \| \leqslant \Delta_k \ .$$

Under this assumption, Powell [1975] proved that if $\mu = 0$ then some subsequence of $\{\nabla f(x_k)\}$ converges to zero, while Thomas [1975] showed that if $\mu > 0$ then the whole sequence $\{\nabla f(x_k)\}$ converges to zero. These results indicate that we can expect $\{x_k\}$ to converge to a point x^* with $\nabla f(x^*) = 0$. Sorensen [1982] proved that we can also expect to have $\nabla^2 f(x^*)$ positive semidefinite provided there is a constant $\sigma \in (0,1)$ such that

$$\psi_k(s_k) = \min\{\psi_k(w) : \| w \| \leqslant \delta_k\}$$

with

$$\| s_k \| \leqslant \delta_k \in [(1-\sigma)\Delta_k, (1+\sigma)\Delta_k] \, .$$

Unfortunately the termination criterion (3.11) is not necessarily consistent with these conditions and thus this result does not allow the choice of s_k provided by Algorithm (3.14). An appropriate generalization of Sorensen's results can be obtained by assuming that there are constants $\beta_1 > 0$ and $\beta_2 > 0$ such that

$$-\psi_k(s_k) \geqslant \beta_1 |\psi_k^*| \quad \text{with} \quad \| s_k \| \leqslant \beta_2 \Delta_k \, . \tag{4.3}$$

An immediate consequence of (3.12) is that if $\psi_k^* \neq 0$ then the approximate solution s_k provided by Algorithm (3.14) with $\sigma_2 = 0$ satisfies (4.3). Of course, if

$\psi_k^* = 0$ then $\nabla f(x_k) = 0$ and $\nabla^2 f(x_k)$ is positive semidefinite and thus Algorithm (4.2) terminates at x_k. Lemma (3.13) shows that Sorensen's assumptions imply that (4.3) holds.

Assumption (4.3) can be expressed in an alternate form which is more convenient for proofs of convergence. If $p_k \in R^n$ is a solution to problem (4.1) then Lemma (2.1) implies that there is a parameter λ_k such that

$$\nabla^2 f(x_k) + \lambda_k I = R_k^T R_k \ , \ (\nabla^2 f(x_k) + \lambda_k I) p_k = -\nabla f(x_k) \ , \ \lambda_k \geqslant 0$$

and with $\lambda_k(\Delta_k - \| p_k \|) = 0$. A calculation now shows that

$$|\psi_k^*| = \tfrac{1}{2}(\| R_k p_k \|^2 + \lambda_k \Delta_k^2). \tag{4.4}$$

This expression for ψ_k^* shows that if (4.3) holds then

$$-\psi_k(s_k) \geqslant \tfrac{1}{2}\beta_1(\| R_k p_k \|^2 + \lambda_k \Delta_k^2), \tag{4.5}$$

and thus the iterates $\{x_k\}$ generated by Algorithm (4.2) satisfy

$$f(x_k) - f(x_{k+1}) \geqslant \tfrac{1}{2}\mu\beta_1(\| R_k p_k \|^2 + \lambda_k \Delta_k^2). \tag{4.6}$$

These two inequalities are essential to the proof of our next result.

THEOREM (4.7). *Let $f : R^n \to R$ be twice continuously differentiable on the level set $\Omega = \{x : f(x) \leqslant f(x_0)\}$ and consider the sequence $\{x_k\}$ produced by Algorithm (4.2) where s_k satisfies (4.3). If Ω is a compact set then either the algorithm terminates at $x_l \in \Omega$ because $\nabla f(x_l) = 0$ and $\nabla^2 f(x_l)$ is positive semidefinite , or $\{x_k\}$ has a limit point $x^* \in \Omega$ with $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ positive semidefinite.*

Proof. If $\nabla f(x_l) = 0$ and $\nabla^2 f(x_l)$ is positive semidefinite for some iterate $x_l \in \Omega$ then the algorithm terminates; otherwise (4.3) implies that $\psi_k(s_k) < 0$ for $k \geqslant 0$ and thus $\{x_k\}$ is well defined and lies in Ω.

Let us now prove the result under the assumption that $\{\lambda_k\}$ is not bounded away from zero. If some subsequence of $\{\lambda_k\}$ converges to zero then since Ω is compact we can assume, without loss of generality, that the same subsequence of $\{x_k\}$ converges to some x^* in the level set Ω. Since $\nabla^2 f(x_k) + \lambda_k I$ is positive semidefinite, $\nabla^2 f(x^*)$ is also positive semidefinite , and $\nabla f(x^*) = 0$ follows by noting that

$$\| R_k p_k \|^2 \geqslant \frac{\| \nabla f(x_k) \|^2}{\| \nabla^2 f(x_k) \| + \lambda_k}$$

and that (4.6) implies that $\{\| R_k p_k \|\}$ converges to zero.

We can show that $\{\lambda_k\}$ is not bounded away from zero by contradiction. If $\lambda_k \geqslant \epsilon > 0$ then (4.3) and (4.5) yield that

$$-\psi_k(s_k) \geqslant \tfrac{1}{2}\beta_1 \lambda_k \Delta_k^2 \geqslant \tfrac{1}{2}\left(\frac{\beta_1}{\beta_2^2}\right)\epsilon \| s_k \|^2.$$

Now, a standard estimate is that

$$|f(x_k + s_k) - f(x_k) - \psi_k(s_k)| \leqslant \tfrac{1}{2}\| s_k \|^2 \max_{0 \leqslant \xi \leqslant 1} \| \nabla^2 f(x_k + \xi s_k) - \nabla^2 f(x_k) \|, \tag{4.8}$$

and thus the last two inequalities show that

$$|\rho_k - 1| \leqslant \left(\frac{\beta_2^2}{\beta_1 \epsilon}\right) \max_{0 \leqslant \xi \leqslant 1} \| \nabla^2 f(x_k + \xi s_k) - \nabla^2 f(x_k) \| \quad . \tag{4.9}$$

Inequality (4.6) implies that $\{\Delta_k\}$ converges to zero and hence $\{\| s_k \|\}$ also converges to zero. Thus the uniform continuity of $\nabla^2 f$ on Ω together with (4.9) implies that $\rho_k > \eta$ for all k sufficiently large and then the updating rules for Δ_k yield that $\{\Delta_k\}$ is bounded away from zero. This is in contradiction of the fact that $\{\Delta_k\}$ converges to zero. ■

The result we have just established is only a sample of the available results for Algorithm (4.7) under assumption (4.3) for s_k. All of the results of Sorensen [1982] hold, and in particular, it can be shown that if f has a finite number of critical points in the level set Ω then every limit point of the sequence $\{x_k\}$ satisfies the second order necessary conditions. We now prove a stronger version of this result.

LEMMA(4.10). *Let x^* be an isolated limit point of a sequence $\{x_k\}$ in R^n. If $\{x_k\}$ does not converge then there is a subsequence $\{x_{l_j}\}$ which converges to x^* and an $\epsilon > 0$ such that $\| x_{l_j+1} - x_{l_j} \| \geqslant \epsilon$.*

Proof. Choose $\epsilon > 0$ so that if $\| x - x^* \| \leqslant \epsilon$ and x is a limit point of $\{x_k\}$ then $x = x^*$. If $\| x_{k_j} - x^* \| \leqslant \epsilon$ then define l_j by

$$l_j = \max\{l : \| x_i - x^* \| \leqslant \epsilon,\ i = k_j, \ldots, l\}.$$

In this manner, a subsequence $\{x_{l_j}\}$ is defined such that

$$\| x_{l_j} - x^* \| \leqslant \epsilon, \quad \| x_{l_j+1} - x^* \| > \epsilon,$$

It follows that $\{x_{l_j}\}$ converges to x^* and thus $\| x_{l_j} - x^* \| \leqslant \frac{1}{2}\epsilon$ for all l_j sufficiently large. Hence,

$$\| x_{l_j+1} - x_{l_j} \| \geqslant \| x_{l_j+1} - x^* \| - \| x_{l_j} - x^* \| \geqslant \tfrac{1}{2}\epsilon$$

as desired. ■

THEOREM(4.11). *Let $f : R^n \to R$ be twice continuously differentiable on the level set $\Omega = \{x : f(x) \leqslant f(x_0)\}$ and consider the sequence $\{x_k\}$ produced by Algorithm (4.2) where s_k satisfies (4.3). If x^* is an isolated limit point of $\{x_k\}$ then $\nabla^2 f(x^*)$ is positive semidefinite.*

Proof. If $\{x_k\}$ converges to x^* then Theorem (4.7) shows (the compactness of Ω is only used to guarantee that $\{x_k\}$ is bounded) that $\nabla^2 f(x^*)$ is positive semidefinite. If $\{x_k\}$ does not converge then Lemma (4.10) applies. Thus, if $\{x_{l_j}\}$ is the subsequence guaranteed by Lemma (4.10) then $\Delta_{l_j} \geqslant \beta_2 \epsilon$, and since (4.6) shows that $\{\lambda_k \Delta_k^2\}$ converges to zero, $\{\lambda_{l_j}\}$ must then converge to zero. We can now use the positive semidefiniteness of $\nabla^2 f(x_k) + \lambda_k I$ to conclude that $\nabla^2 f(x^*)$ is positive semidefinite. ■

We have already noted that Thomas [1975] proved that $\{\nabla f(x_k)\}$ converges to zero. Hence, if $\nabla^2 f(x^*)$ is nonsingular at a limit point x^* of $\{x_k\}$ then x^* is an isolated limit point, and Theorem (4.11) shows that $\nabla^2 f(x^*)$ is positive definite.

Since $\psi_k(s_k) \leqslant 0$ we have that

$$\tfrac{1}{2}\| s_k \| \leqslant \| \nabla^2 f(x_k)^{-1} \| \, \| \nabla f(x_k) \| \tag{4.12}$$

whenever $\nabla^2 f(x_k)$ is positive definite, and thus Lemma (4.10) shows that $\{x_k\}$ converges to x^*. This establishes the following result.

THEOREM (4.13). *Let $f : R^n \to R$ be twice continuously differentiable on the level set $\Omega = \{x : f(x) \leqslant f(x_0)\}$ and consider the sequence $\{x_k\}$ produced by Algorithm (4.2) where s_k satisfies (4.3). If x^* is a limit point of $\{x_k\}$ and $\nabla^2 f(x^*)$ is nonsingular then $\{x_k\}$ converges to x^* and $\nabla^2 f(x^*)$ is positive definite.*

An additional result which is helpful in establishing rate of convergence results is that under the assumptions of Theorem (4.13) the sequence $\{\Delta_k\}$ is bounded away from zero. To prove this first note that if $\epsilon_0 > 0$ is a lower bound on the eigenvalues of $\nabla^2 f(x_k)$ then (4.4) shows that

$$|\psi_k^*| \geqslant \tfrac{1}{2}\epsilon_0 \min\{\Delta_k^2, \| s_k^N \|^2\}$$

where

$$s_k^N \equiv -\nabla^2 f(x_k)^{-1} \nabla f(x_k).$$

Now, (4.12) implies that $\tfrac{1}{2}\| s_k \| \leqslant \kappa \| s_k^N \|$ where κ is an upper bound on the condition number of $\nabla^2 f(x_k)$, and hence assumption (4.3) shows that there is a constant $\epsilon_1 > 0$ with

$$-\psi_k(s_k) \geqslant \tfrac{1}{2}\epsilon_1 \| s_k \|^2.$$

This estimate and (4.8) then yield that

$$\left| \rho_k - 1 \right| \leqslant \left(\frac{1}{\epsilon_1} \right) \max_{0 \leqslant \xi \leqslant 1} \| \nabla^2 f(x_k + \xi s_k) - \nabla^2 f(x_k) \| ,$$

and thus $\rho_k > \eta$ for all k sufficiently large. It follows that $\{\Delta_k\}$ is bounded away from zero as desired.

Rate of convergence results can be obtained with the additional - but mild - assumption that there is a constant $\beta_3 > 0$ such that if $\| s_k \| \leqslant \beta_3 \Delta_k$ then $s_k = s_k^N$. For example, Algorithm (3.14) satisfies this assumption because if we have $\| s_k \| < (1-\sigma_1)\Delta_k$ then $\lambda_k = 0$. In particular, note that if $\| s_k \| < \Delta_k$ then (3.11) cannot be satisfied because then Algorithm (3.14) would set $\| s_k \| = \Delta_k$.

Under the above assumption, Theorem (4.13) can be extended to show that $\{x_k\}$ converges to x^* with a Q-superlinear rate of convergence and that if $\nabla^2 f$ is Lipschitz continuous then the rate of convergence is quadratic. The proof is not difficult; since $\tfrac{1}{2}\| s_k \| \leqslant \kappa \| s_k^N \|$ where κ is an upper bound on the condition number of the Hessian matrix at x_k, eventually $\| s_k \| \leqslant \beta_3 \Delta_k$, and then $\{x_k\}$ becomes an unmodified Newton's method. The rate of convergence results are then standard.

5. Computational results. Algorithm (3.14) has been implemented in a Fortran subroutine GQTPAR, and in this section we present the results of GQTPAR on two sets of test problems. Since our main concern here is the performance of GQTPAR in a trust region method, we used $\sigma_1 = 0.1$ and $\sigma_2 = 0$ in the convergence criteria (3.10) and (3.11). The reason for setting $\sigma_2 = 0$ is that σ_2 is only required to deal with the case where $g = 0$ and B is positive semidefinite and

singular, and in this situation, a trust region method terminates. The initial choice of λ depends on the test and is described below. All tests were performed in double precision on a VAX 11/780. This provides an accuracy of about 17 significant decimal digits.

The first set of tests is concerned with the performance of GQTPAR in the context of a trust region Newton's method. The test problems used are the 18 unconstrained minimization problems described in Moré, Garbow, and Hillstrom [1981a]. For each problem it is possible to specify a set of starting points, and in 8 of the problems it is also possible to specify the dimension. A particular set of test cases is defined by the data provided to the test drivers. We used the sample data provided in Moré, Garbow, and Hillstrom [1981b] which defines 52 test cases. The dimensions of the problems specified in this set of tests range from 2 to 12.

The trust region Newton's method used follows Algorithm (4.2) and proved to be quite successful on these problems. Details of the Newton method will appear elsewhere. For the purposes of this paper it suffices to remark that on the first call to GQTPAR the initial λ is zero, but on succeeding calls the initial λ is the same as the final λ from the previous call of GQTPAR. At the end of Section 3 we pointed out that it is possible to obtain a more educated guess for the initial λ, but this choice provides a stringent test of GQTPAR.

The performance of GQTPAR on these problems was very satisfactory. There were 2580 calls to GQTPAR and the average number of iterations per call was 1.63; the largest number of iterations was 10. In about 20% of the calls convergence criterion (3.11) was satisfied.

The second set of tests is designed to exercise the various features of GQTPAR as an individual algorithm on problems of type (1.1). For these problems we decided to use

$$\lambda = \frac{\| g \|}{\Delta} \tag{5.1}$$

as the initial λ. Unless other information is available, this is a reasonable automatic choice. In these problems we generated sequences of uniformly distributed random numbers with the RAND function of Schrage [1979]. Given an integer seed, RAND generates a random number in (0,1) and changes the seed. Thus a sequence of random numbers can be generated by repeated calls to RAND.

A convenient way to define a problem of type (1.1) is to set $B = QDQ^T$ for some orthogonal matrix Q and diagonal matrix D, and to then let $g = Q\hat{g}$ for some vector $\hat{g}$. This makes it possible to generate a (potentially) hard case by setting to zero the component of $\hat{g}$ corresponding to the smallest element of D. The structure of B is scrambled by choosing the orthogonal matrix Q of the form $Q_1Q_2Q_3$ where

$$Q_j = I - 2\frac{w_j w_j^T}{\| w_j \|^2} \quad , \quad j = 1,2,3,$$

and the components of w_j are random numbers uniformly distributed in $(-1,1)$. A problem of type (1.1) can be generated be specifying Δ, $\hat{g}$, and D; different choices lead to problems with various characteristics.

We consider four different ways of specifying $\hat{g}$ and D. In all four cases, the elements of $\hat{g}$ and D are initially chosen as uniformly distributed random numbers in $(-1,1)$. This choice leads to the general case; as mentioned above, a hard case can then be obtained by setting to zero the component of $\hat{g}$ corresponding to the smallest element of D. A positive definite case is obtained by replacing D by $|D|$, and in the saddle point case all the components of g are set to zero.

The choice of Δ is critical; if Δ is chosen from $(0,1)$ then the tests are easy because (5.1) is almost always an excellent choice. A harder test is obtained if Δ is chosen as uniformly distributed from $(0,100)$, and this choice is made in our tests. We have observed that a wider distribution in the choice of Δ does not affect the results significantly, and that the range $(0,100)$ appears to be the hardest choice for these problems.

We now present the results of tests in each of the above four cases and for dimensions 10, 20, 40, 60, 80, and 100. For each case and each dimension we generated 5 problems and recorded both the average and the maximum number of iterations required for convergence. The results are presented in the tables below.

TABLE 1
The general case.

	Number of iterations	
Dimension	Average	Maximum
10	2.0	4
20	2.6	5
40	3.2	4
60	3.0	4
80	3.2	4
100	4.0	5

TABLE 2
The hard case.

	Number of iterations	
Dimension	Average	Maximum
10	1.6	3
20	2.2	3
40	3.0	3
60	2.8	3
80	3.2	4
100	3.2	4

TABLE 3
The saddle point case.

	Number of iterations	
Dimension	Average	Maximum
10	1.6	3
20	2.0	2
40	2.6	3
60	3.0	4
80	3.6	4
100	3.2	4

TABLE 4
The positive definite case.

	Number of iterations	
Dimension	Average	Maximum
10	2.4	4
20	2.0	2
40	2.4	3
60	2.4	3
80	2.4	3
100	3.0	4

An interesting aspect of the results for the general case is that Algorithm (3.14) terminated on condition (3.11) in 26 out of the 30 cases. This shows that (3.11) is powerful enough to terminate the algorithm even on nonhard cases. For smaller values of σ_1, however, it is more difficult to satisfy (3.11) and this gives GQTPAR a chance to produce an iterate $\lambda > -\lambda_1$ with $\phi(\lambda) > 0$. Once this occurs, the Newton iteration converges quadratically and (3.10) is eventually satisfied. As noted above, the results improve for smaller choices of Δ, and for example, if Δ is chosen from (0,1) then the maximum number of iterations is 2.

The results of Table 2 show that the hard case, once recognized and treated properly, can be handled with the same computational effort as the general case. In contrast to the general case, the results for the hard case are sensitive to the choice of σ_1 since in this case it is necessary to determine λ_1 and Algorithm (3.14) determines λ_1 with a bisection-type process. Another interesting point is that for these problems Algorithm (3.14) does not always terminate on condition (3.11) since the hard case only occurs if $\Delta > \| p_\alpha \|$ for $\alpha \geqslant -\lambda_1$. This situation is avoided in the saddle point case by choosing $g = 0$.

The saddle point case is unusual because the algorithm and the results are independent of the choice of Δ, and termination always occurs on condition

(3.11). Although setting $g = 0$ is an extreme choice, the numerical results are insensitive to the choice of g provided the components of g are sufficiently small. For example, if the components of g are chosen from $(-10^{-8},10^{-8})$ then the number of iterations increases by 1 in two of the problems, but otherwise the results are unchanged.

In the positive definite case, the choice of Δ as a uniformly distributed random number from (0,100) resulted in exits with $\lambda = 0$ in about half the problems, and this explains why the average number of iterations is close to 2. On the other hand, if Δ is chosen from (0,1) then (5.1) leads to termination on the first iteration.

These results show that GQTPAR performs adequately in all cases. As expected, a smaller value of σ_1 requires more iterations, but the increase is surprisingly small in most cases. The choice $\sigma_1 = 0.1$ is very satisfactory in many cases since it does not require a large number of iterations and produces a nearly optimal approximate solution as predicted by the theory.

6. Concluding remarks. We have presented an algorithm for the constrained quadratic minimization problem (1.1) and reported the computational results of the implementation GQTPAR. This implementation uses the Cholesky factorization to solve systems of the form

$$(B + \lambda I)u = v ,$$

but it is possible to use other factorizations. For example, the decomposition

$$B = QTQ^T \tag{6.1}$$

where Q is orthogonal and T is tridiagonal leads to systems of the form

$$(T + \lambda I)w = Q^T v , \quad u = Qw,$$

and since Algorithm (3.14) is invariant with respect to orthogonal transformations, it is possible to produce an implementation which only requires on the order of n arithmetic operations per iteration. We have not used this factorization because we expect GQTPAR to be used in a trust region method , and in this case our numerical results show that a call to GQTPAR requires less than two Cholesky factorizations on the average.

Another argument against the use of factorization (6.1) is that it usually ignores the structure of B. In particular, for sparse systems the Cholesky factorization offers many advantages. Good software based on the Cholesky factorization currently exists for the solution of positive definite linear systems, and this together with an estimator of the smallest singular value of a sparse upper triangular matrix is all that is required to provide a trust region Newton's method for optimization problems with a sparse Hessian matrix.

It would be of interest to develop a method for large scale problems of type (1.1) which does not require the solution of linear systems. Iterative approaches along the lines of conjugate directions or Lanczos type methods have been considered, but a complete solution is not known to us.

Appendix. The purpose of this appendix is to prove that if $\hat{z}$ is the LINPACK estimate of a right singular vector of R corresponding to the smallest singular value, then $\| R\hat{z} \|$ is near zero when R is nearly singular.

As mentioned in Section 3, the LINPACK technique constructs a vector e with components ± 1 such that the solution w to the system $R^T w = e$ is large. Then the system $Rv = w$ is solved for v, and

$$\hat{z} = \frac{v}{\| v \|}$$

is the LINPACK estimate of the singular vector. If R is exactly singular then the following description must be modified, but in this case $R\hat{z} = 0$ is obtained so there is nothing to prove. Therefore, we now assume that R is nonsingular and show that if ρ_{ij} is the (i,j) element of R and

$$\sum_{j=1}^{n} \sum_{i=1}^{j} |\rho_{ij}| \leqslant \rho , \tag{A.1}$$

then

$$\| R\hat{z} \| \leqslant n^{1/2}(1+\rho)\min\{ |\rho_{kk}| : 1 \leqslant k \leqslant n \} . \tag{A.2}$$

Thus, if $B + \lambda I = R^T R$ and λ is near $-\lambda_1$ then some diagonal element of R is near zero and (A.2) implies that $\| R\hat{z} \|$ is near zero.

To establish (A.2) we first note that for any vectors v and w such that $R^T w = e$ and $Rv = w$ we have

$$\| w \|^2 \leqslant n^{1/2} \| v \| . \tag{A.3}$$

In fact, the Cauchy-Schwarz inequality shows that

$$\| w \|^2 = \| R^{-T} e \|^2 = e^T R^{-1} R^{-T} e \leqslant \| e \| \| v \| ,$$

and thus (A.3) follows because each of the components of e are ± 1 . Also note that (A.3) implies that

$$\| R\hat{z} \| = \frac{\| w \|}{\| v \|} \leqslant \frac{n^{1/2}}{\| w \|} . \tag{A.4}$$

We now show that the LINPACK technique for selecting w does cause $\| w \|$ to grow if R is nearly singular. To prove this, recall (Cline, Moler, Stewart, and Wilkinson [1979]) that if $\omega_1, \ldots, \omega_{k-1}$ have been specified then we compute

$$p_i^{(k-1)} = \sum_{l=1}^{k-1} \rho_{li} w_l , \quad k \leqslant i \leqslant n ,$$

and consider two possible choices of w_k:

$$\omega_k^+ = \frac{1 - p_k^{(k-1)}}{\rho_{kk}} , \qquad \omega_k^- = \frac{-1 - p_k^{(k-1)}}{\rho_{kk}} .$$

If

$$|\omega_k^+| + \sum_{i=k+1}^{n} |p_i^{(k-1)} + \rho_{ki}\omega_k^+| \geqslant |\omega_k^-| + \sum_{i=k+1}^{n} |p_i^{(k-1)} + \rho_{ki}\omega_k^-|$$

then $\omega_k = \omega_k^+$ is chosen, and otherwise $\omega_k = \omega_k^-$ is chosen. Since

$$\max\{|\omega_k^+|, |\omega_k^-|\} \geq \frac{1}{|\rho_{kk}|},$$

it follows that

$$\frac{1}{|\rho_{kk}|} \leq |\omega_k| + \sum_{i=k+1}^{n} |p_i^{(k-1)} + \rho_{ki}\omega_k| .$$

Thus, in view of (A.1) we have that

$$\frac{1}{|\rho_{kk}|} \leq (1+\rho)\|w\|, \quad 1 \leq k \leq n.$$

This inequality together with (A.4) shows that (A.2) holds as desired.

Estimate (A.2) is quite crude, and it certainly is not being offered as an indication of the accuracy of the LINPACK estimate. The only purpose of (A.2) is to validate the use of the LINPACK estimate in Algorithm (3.14). It is of interest to note that the same proof techniques show that the look-behind algorithm (with unit weights) of Cline, Conn, and Van Loan [1982] also satisfies an estimate of the form (A.2).

Acknowledgment. We would like to thank David Gay for his comments on a preliminary version of this manuscript and for sharing his UNIX expertise with us.

REFERENCES

A. K. CLINE, A. R. CONN, and C. VAN LOAN, [1982], *Generalizing the LINPACK condition estimator,* Numerical Analysis, J. P. Hennart, ed., Lecture Notes in Mathematics 909, Springer-Verlag, Berlin.

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

R. FLETCHER [1980], *Practical Methods of Optimization, Volume* 1: *Unconstrained Optimization,* John Wiley, New York.

D. M. GAY, [1981], *Computing optimal locally constrained steps,* SIAM J. Sci. Stat. Comput., 2, pp. 186–197.

S. GOLDFELD, R. QUANDT and H. TROTTER, [1966], *Maximization by quadratic hill climbing,* Econometrica, 34, pp. 541–551.

M. D. HEBDEN [1973], *An algorithm for minimization using exact second derivatives,* Atomic Energy Research Establishment report T.P. 515, Harwell, England.

J. J. MORÉ [1978], *The Levenberg-Marquardt algorithm: Implementation and theory,* Proceedings of the Dundee Conference on Numerical Analysis, G. A. Watson, ed., Springer-Verlag, Berlin.

J. J. MORÉ, B. S. GARBOW and K. E. HILLSTROM [1981a], *Testing unconstrained optimization software,* ACM Trans. Math. Software, 7, pp. 17–41.

J. J. MORÉ, B. S. GARBOW, and K. E. HILLSTROM, [1981b], *Fortran subroutines for testing unconstrained optimization software,* ACM Trans. Math. Software, 7, pp. 136–140.

M. J. D. POWELL [1975], *Convergence properties of a class of minimization algorithms,* Nonlinear Programming 2, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, New York.

C. H. REINSCH [1967], *Smoothing by spline functions,* Numer. Math., 10, pp. 177–183.

——— [1971], *Smoothing by spline functions* II, Numer. Math., 16, pp. 451–454.

L. SCHRAGE [1979], *A more portable Fortran random number generator,* ACM Trans. Math. Software, 5, pp. 132–138.

D. C. SORENSEN, [1982], *Newton's method with a model trust-region modification,* SIAM J. Numer. Anal., 19, pp. 409–426.

——— [1982], *Trust region methods for unconstrained minimization,* in Nonlinear Optimization 1981, M. J. D. Powell, ed., Academic Press, New York.

S. W. THOMAS [1975], *Sequential estimation techniques for quasi-Newton algorithms,* Ph.D. dissertation, Cornell University, Ithaca, NY.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0001 $01.25/0

THE BORDERING ALGORITHM AND PATH FOLLOWING NEAR SINGULAR POINTS OF HIGHER NULLITY*

HERBERT B. KELLER†

Abstract. We study the behavior of the bordering algorithm (a form of block elimination) for solving nonsingular linear systems with coefficient matrices in the partitioned form $\begin{pmatrix} A & B \\ C^* & D \end{pmatrix}$ when $\dim \mathcal{N}(A) \geqq 1$. Systems with this structure naturally occur in path following procedures. We show that under appropriate assumptions, the algorithm, which is based on solving systems with coefficient matrix A, works as A varies along a path and goes through singular points. The required assumptions are justified for a large class of problems coming from discretizations of boundary value problems for differential equations.

Key words. path following, bordering algorithm, block elimination, singular systems

1. Introduction. We study here some specific procedures for solving linear systems of the special form

$$\mathcal{A}\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\xi} \end{pmatrix} \equiv \begin{pmatrix} A & B \\ C^T & D \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\xi} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \boldsymbol{\gamma} \end{pmatrix}. \tag{1.1}$$

The matrix $\mathcal{A}$ is of order $N+\nu$ with the indicated submatrices: A of order N, D of order ν, B is $N \times \nu$ and C is $\nu \times N$. The vectors $\mathbf{x}, \mathbf{g} \in \mathbb{R}^N$ and $\boldsymbol{\xi}, \boldsymbol{\gamma} \in \mathbb{R}^\nu$. Such linear systems arise when using Newton's method in some specific path following algorithms to be discussed later. Indeed it is *families* of systems of the form (1.1) that we solve in applications and invariably $N \gg \nu$. In fact a major part of our current interest stems from the fact that in the course of this process the matrix A becomes singular or near singular, while $\mathcal{A}$ remains nonsingular. The so-called bordering algorithm that we use to solve (1.1) when both A and $\mathcal{A}$ are nonsingular, which is based on solving systems of the form

$$A\mathbf{v} = \mathbf{b}, \qquad \mathbf{v}, \mathbf{b} \in \mathbb{R}^N, \tag{1.2}$$

can be shown to be valid for our applications when A is singular. (The bordering algorithm is but a special case of block Gaussian elimination.) Our analysis has been presented in [8] for the case $\nu = 1$. However this is not generally available and it is somewhat surprising that the case $\nu \geqq 1$, treated here, can be done so simply. Thus the present study includes the results of [8] but is independent of that reference.

In § 2 we formulate the bordering algorithm for solving (1.1) when A and $\mathcal{A}$ are nonsingular. We also show how the Woodbury formula [4] can be used in this case but under more restrictive conditions.

In § 3 we examine the solution of (1.1) when A has nullity ν and $\mathcal{A}$ is nonsingular. We use here bases for the right and left null spaces of A. In § 4 we show how these bases are obtained from an exact LU-decomposition (with pivoting) of A. Then in § 5 we show how the bordering algorithm can be applicable to this singular case when a finite precision factorization is employed.

The LU-factorization of singular matrices must, in general employ full pivoting. However, for a large class of "banded" linear systems arising from discretizations of boundary value problems for differential equations, we show that partial pivoting can

* Received by the editors March 15, 1982, and in revised form July 26, 1982. This work was supported by the U.S. Department of Energy under contract EY-76-S-03-0767, Project agreement #12, and the U.S. Army Research Office under contract DAAG 29-78-C-0011.

† Applied Mathematics, California Institute of Technology 217-50, Pasadena, California 91125.

be used until half a bandwidth from the end; only then is full pivoting required. This is done in § 6 where we also briefly describe the path following applications.

2. Bordering algorithms: nonsingular A and $\mathscr{A}$. Suppose A and $\mathscr{A}$ are nonsingular. Then we can solve (1.1) as follows. Determine the $N \times \nu$ matrix V and the vector $\mathbf{w} \in \mathbb{R}^N$ from

$$AV = B, \qquad A\mathbf{w} = \mathbf{g}. \tag{2.1a, b}$$

Find $\boldsymbol{\xi} \in \mathbb{R}^\nu$ by solving

$$(D - C^T V)\boldsymbol{\xi} = \boldsymbol{\gamma} - C^T \mathbf{w}. \tag{2.2}$$

Then evaluate $\mathbf{x} \in \mathbb{R}^N$ as

$$\mathbf{x} = \mathbf{w} - V\boldsymbol{\xi}. \tag{2.3}$$

This procedure is simply a form of block Gaussian elimination that results from the factorization

$$\begin{pmatrix} A & B \\ C^T & D \end{pmatrix} = \begin{pmatrix} A & 0 \\ C^T & I \end{pmatrix} \begin{pmatrix} I & V \\ 0 & D - C^T V \end{pmatrix}.$$

Since A is nonsingular V and $\mathbf{w}$ are uniquely defined by (2.1a, b). Since in addition $\mathscr{A}$ is nonsingular the Schür complement of A in $\mathscr{A}$ must be nonsingular and this is just

$$D - C^T A^{-1} B = D - C^T V. \tag{2.4}$$

Thus $\boldsymbol{\xi}$ is uniquely defined by (2.2). Using these results in (1.1) we see that the solution is obtained.

After an LU-factorization of A we need only $\nu + 1$ backsolves to obtain V and $\mathbf{w}$. Then we must solve the νth order system (2.2). The inhomogeneous term in this system and the formation of $\mathbf{x}$ in (2.3) requires the equivalent of 2ν inner products of N-vectors. Thus for $N \gg \nu$, our main case of interest, the bulk of the work is in the factorization of A.

Another interesting procedure for solving (1.1) is to use the last ν equations to eliminate $\boldsymbol{\xi}$ from the first N equations. This can be done if D is nonsingular—a requirement not imposed above. When this holds we obtain

$$\boldsymbol{\xi} = D^{-1}(\boldsymbol{\gamma} - C^T \mathbf{x}), \qquad (A - BD^{-1}C^T)\mathbf{x} = \mathbf{g} - BD^{-1}\boldsymbol{\gamma}. \tag{2.5a, b}$$

Note that the coefficient matrix in (2.5b) is just the Schür complement of D in $\mathscr{A}$. Thus if, as we have just assumed, D is nonsingular this Schür complement is also nonsingular. But we also note that this coefficient matrix is an at most rank ν modification of A; that is $BD^{-1}C^T$ has the structure required to apply the Woodbury formula [4, p. 124] to solve (2.5b). Specifically if A is nonsingular then

$$(A - BD^{-1}C^T)^{-1} = A^{-1} + A^{-1}BTC^T A^{-1}, \tag{2.6a}$$

where

$$T = (D - C^T A^{-1} B)^{-1}. \tag{2.6b}$$

The inverse in (2.6b) exists since T^{-1} is just the Schür complement of A in $\mathscr{A}$. However the application of (2.5a, b) and (2.6a, b) is more restrictive and more costly than the bordering algorithm and so we do not consider it further here.

3. Singular A, nonsingular $\mathscr{A}$. We now turn to the case of singular A with nullity ν. Equivalently we can assume that

$$\begin{aligned} &\dim \mathcal{N}(A) = \nu, \\ &\mathcal{N}(A) = \operatorname{span}\{\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_\nu\}, \qquad \Phi \equiv (\boldsymbol{\phi}_1 \cdots \boldsymbol{\phi}_\nu), \\ &\mathcal{N}(A^T) = \operatorname{span}\{\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_\nu\}, \qquad \Psi \equiv (\boldsymbol{\psi}_1 \cdots \boldsymbol{\psi}_\nu). \end{aligned} \tag{3.1a, b, c}$$

Under condition (3.1a) the matrix $\mathscr{A}$ of (1.1) is nonsingular if and only if (see [1, Appendix II])

$$\begin{aligned} &c_0)\ \dim \mathcal{R}(B) = \nu, \qquad c_1)\ \mathcal{R}(B) \cap \mathcal{R}(A) = 0, \\ &c_2)\ \dim \mathcal{R}(C^T) = \nu, \qquad c_3)\ \mathcal{N}(C^T) \cap \mathcal{N}(A) = 0. \end{aligned} \tag{3.2}$$

Using the $N \times \nu$ matrices Φ and Ψ of basis vectors introduced in (3.1b, c) we can reformulate (3.2) in the equivalent form:

$$\begin{aligned} &\Psi^T B \text{ is nonsingular} \Leftrightarrow (3.2c_0, c_1), \\ &C^T \Phi \text{ is nonsingular} \Leftrightarrow (3.2c_2, c_3). \end{aligned} \tag{3.3a, b}$$

The proofs of the indicated equivalences are exercises in basic linear algebra.

To solve (1.1) in this case we rewrite the system as

$$A\mathbf{x}_0 + B\boldsymbol{\xi}_0 = \mathbf{g}, \qquad C^T\mathbf{x}_0 + D\boldsymbol{\xi}_0 = \boldsymbol{\gamma}. \tag{3.4a, b}$$

Multiply (3.4a) by Ψ^T and use (3.3a) to get

$$\boldsymbol{\xi}_0 = (\Psi^T B)^{-1}(\Psi^T \mathbf{g}). \tag{3.5}$$

With this value of $\boldsymbol{\xi}_0$ in (3.4a) we obtain

$$A\mathbf{x}_0 = \mathbf{g} - B(\Psi^T B)^{-1}\Psi^T \mathbf{g}. \tag{3.6}$$

This clearly has a solution since the right-hand side is in $\mathcal{R}(A)$. The general solution of (3.6) is

$$\mathbf{x}_0 = \mathbf{x}^p + \Phi\boldsymbol{\zeta}_0, \tag{3.7a}$$

where $\mathbf{x}^p$ is any particular solution and $\boldsymbol{\zeta}_0 \in \mathbb{R}^\nu$ is arbitrary. Using this in (3.4b) we obtain on recalling (3.3b) the unique value for $\boldsymbol{\zeta}_0$:

$$\boldsymbol{\zeta}_0 = (C^T\Phi)^{-1}[\boldsymbol{\gamma} - D\boldsymbol{\xi}_0 - C^T\mathbf{x}^p]. \tag{3.7b}$$

The unique solution of (3.4a, b) is thus given in (3.5) and (3.7a, b).

To evaluate this solution representation we need Φ, Ψ, $\mathbf{x}^p$, $\boldsymbol{\xi}_0$ and $\boldsymbol{\zeta}_0$. Again for $N \gg \nu$ the work in solving for the ν-vectors $\boldsymbol{\xi}_0$ and $\boldsymbol{\zeta}_0$ is negligible compared to that in solving for the $2\nu + 1$ vectors in $\mathbb{R}^N$. We turn next to the determination of the null vectors.

4. Right and left null vectors. To compute the right and left null vectors of A when (3.1) holds we must use some form of full pivoting. Thus with appropriate permutation matrices P and Q, corresponding to row and column interchanges in A, we must work with

$$\bar{A} \equiv PAQ.$$

To avoid notational complexity we will, as usual, assume that the indicated interchanges have already been made in the systems (1.1) and (2.1a, b) and thus use A rather than

$\bar{A}$. Then we may assume that A can be factored, via Gauss elimination, in the form

$$A = LU \equiv \begin{pmatrix} L_r & 0_{r\nu} \\ L_{\nu r} & I_\nu \end{pmatrix} \begin{pmatrix} U_r & U_{r\nu} \\ 0_{\nu r} & \varepsilon_\nu \end{pmatrix}. \tag{4.1}$$

Here, $r+\nu = N$, L_r and U_r are nonsingular $r \times r$ matrices, I_ν and ε_ν are $\nu \times \nu$ matrices (I_ν the identity), $0_{r\nu}$ and $0_{\nu r}^T$ are $r \times \nu$ matrices of zeros, $U_{r\nu}$ is $r \times \nu$ and $L_{\nu r}$ is $\nu \times r$. For exact calculations, which we assume for this section

$$\varepsilon_\nu \equiv 0_\nu. \tag{4.2}$$

The matrices L_r and U_r are triangular with forms

$$L_r \equiv \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ X & & 1 \end{pmatrix}, \qquad U_r \equiv \begin{pmatrix} u_{11} & & X \\ & \ddots & \\ 0 & & u_{rr} \end{pmatrix}. \tag{4.3}$$

In addition $u_{11} \cdots u_{rr} \neq 0$.

To find the right null vectors we note that

$$A\boldsymbol{\phi} = \mathbf{0} \quad \text{if and only if} \quad U\boldsymbol{\phi} = \mathbf{0}.$$

With the ν unit vectors $\mathbf{e}_j \in \mathbb{R}^\nu$, $1 \leqq j \leqq \nu$, we seek vectors $\hat{\boldsymbol{\phi}}_j \in \mathbb{R}^r$ such that

$$\boldsymbol{\phi}_j = a_j \begin{pmatrix} \hat{\boldsymbol{\phi}}_j \\ -\mathbf{e}_j \end{pmatrix}, \qquad 1 \leqq j \leqq \nu,$$

are right null vectors of A. We find that this is the case for $a_j \neq 0$ provided the $\hat{\boldsymbol{\phi}}_j$ satisfy

$$U_r \hat{\boldsymbol{\phi}}_j = U_{r\nu} \mathbf{e}_j, \qquad 1 \leqq j \leqq \nu.$$

Thus a set of ν independent right null vectors of A is given by

$$\Phi \equiv \begin{pmatrix} U_r^{-1} U_{r\nu} \\ -I_\nu \end{pmatrix}. \tag{4.4a}$$

Similarly the left null vectors of A satisfy

$$A^T \boldsymbol{\psi} = \mathbf{0} \quad \text{if and only if} \quad L^T \boldsymbol{\psi} = \begin{pmatrix} \mathbf{0} \\ \mathbf{e} \end{pmatrix},$$

for arbitrary nontrivial $\mathbf{e} \in \mathbb{R}^\nu$. Thus if we choose for $\mathbf{e}$ the basis $\{\mathbf{e}_j\}$ for $\mathbb{R}^\nu$ we obtain the set of ν independent left null vectors

$$\Psi \equiv \begin{pmatrix} (L_r^T)^{-1} L_{\nu r}^T \\ -I_\nu \end{pmatrix}. \tag{4.4b}$$

Note that Φ and Ψ are each determined by solving only triangular systems. Thus both nullspaces are obtained using only ν backsolves for the factorization $L_r U_r$.

5. Practical calculations: almost singular A. In finite precision calculations, with A as in § 4, we do not get $\varepsilon_\nu \equiv 0_\nu$ as assumed in (4.2). If A is appropriately scaled, then with full pivoting in t-digit floating arithmetic we assume an estimate of the form [10]

$$\|\varepsilon_\nu\| \leqq \|A\| \times 10^{-t} \equiv \varepsilon_0. \tag{5.1}$$

Indeed all the nonidentically zero elements (i.e., $0_{\nu r}$ and $0_{r\nu}$) and nondiagonal elements of L_r and I_ν have errors that can be bounded by the same quantity.

To assure that our procedures and analysis are applicable we must assume that the factorization procedure can, at least, correctly determine the nullity. A reasonably simple test that we have found works well in many applications (see § 6) is to set $\nu = n$ when

$$\max_{\substack{i>N-n \\ j>N-n}} |a_{ij}^{(N-n)}| < \delta \frac{|a_{N-n,N-n}^{(N-n)}|^2}{|a_{N-n-1,N-n-1}^{(N-n-1)}|}. \tag{5.2}$$

Here we employ the usual Gaussian elimination notation $a_{ij}^{(k)}$ for the kth stage of the elimination. We have used δ in $[10^{-3}, 10^{-2}]$ for many calculations. There are more robust procedures for rank determination than Gauss elimination, such as QR- and SV-decompositions. However for the differential equations applications these other procedures are prohibitively expensive while Gaussian elimination does not suffer from the pathological cases designed to show that it does not always work.

Further we shall require that after the numerically factored form (4.1) is obtained it satisfies

$$\min_{j \leqq r} |u_{jj}| \gg \varepsilon_0, \tag{5.3}$$

say, for example, $|\varepsilon_0/u_{jj}| \leqq \delta$.

Finally we require for the analysis of this section that the matrix

$$\varepsilon_\nu \equiv (a_{ij}^{(r)}), \qquad r < i \leqq N, \quad r < j \leqq N, \tag{5.4a}$$

is nonsingular. Of course this would be the generic case if the elements $a_{ij}^{(r)}$ were truly random roundoff. But we can insure this in the actual calculations by setting

$$\varepsilon I_\nu \Rightarrow \varepsilon_\nu, \tag{5.4b}$$

where, say,

$$\varepsilon = \frac{1}{\nu} \|(a_{ij}^{(r)})\|_\infty.$$

With the above assumptions and the assurance that (5.4a) holds, we proceed to examine the bordering algorithm (2.1a, b)–(2.3) using the factorization (4.1) and shall see how it relates to the solution in § 3. We neglect for the moment all errors in the factorization save those of ε_ν. At the end of our analysis we easily include the effects of all the errors. To conform to the partitioning in (4.1) it is useful to introduce the notation

$$V \equiv \begin{pmatrix} V_{r\nu} \\ V_\nu \end{pmatrix}, \quad B \equiv \begin{pmatrix} B_{r\nu} \\ B_\nu \end{pmatrix}, \quad \mathbf{g} \equiv \begin{pmatrix} \mathbf{g}_r \\ \mathbf{g}_\nu \end{pmatrix}, \quad \mathbf{w} \equiv \begin{pmatrix} \mathbf{w}_r \\ \mathbf{w}_\nu \end{pmatrix}. \tag{5.5}$$

Specifically, $V_{r\nu}$ and $B_{r\nu}$ are $r \times \nu$, V_ν and B_ν are $\nu \times \nu$, $\mathbf{g}_r$ and $\mathbf{w}_r \in \mathbb{R}^r$, $\mathbf{g}_\nu$ and $\mathbf{w}_\nu \in \mathbb{R}^\nu$. With (5.5), (4.1) and (4.4a, b) we obtain from (2.1a):

$$V = \begin{pmatrix} (L_r U_r)^{-1} B_{r\nu} \\ 0_\nu \end{pmatrix} - \Phi V_\nu \equiv V^p - \Phi V_\nu, \qquad \varepsilon_\nu V_\nu = -\Psi^T B; \tag{5.6a, b}$$

and from (2.1b)

$$\mathbf{w} = \begin{pmatrix} (L_r U_r)^{-1} \mathbf{g}_r \\ \mathbf{0}_\nu \end{pmatrix} - \Phi \mathbf{w}_\nu \equiv \mathbf{w}^p - \Phi \mathbf{w}_\nu, \qquad \varepsilon_\nu \mathbf{w}_\nu = -\Psi^T \mathbf{g}. \tag{5.7a, b}$$

Since ε_ν is assumed nonsingular V_ν and $\mathbf{w}_\nu$ are uniquely determined. We use (5.6a) and (5.7a) in (2.2) to get

$$(D - C^T V^P + C^T \Phi V_\nu)\boldsymbol{\xi} = \boldsymbol{\gamma} - C^T \mathbf{w}^P + C^T \Phi \mathbf{w}_\nu.$$

Multiply this by $(\Psi^T B)^{-1}\varepsilon_\nu (C^T \Phi)^{-1}$, whose existence is assured by (3.3a, b), and use (5.6b) and (5.7b) to write the result as

$$\boldsymbol{\xi} = (\Psi^T B)^{-1}\{(\Psi^T \mathbf{g}) - \varepsilon_\nu (C^T \Phi)^{-1}[\boldsymbol{\gamma} - C^T \mathbf{w}^P - (D - C^T V^P)\boldsymbol{\xi}]\}. \tag{5.8}$$

Recalling (3.5) and the fact that $\|\varepsilon_\nu\| \ll 1$, the above implies

$$\boldsymbol{\xi} = \boldsymbol{\xi}_0 + \mathcal{O}(\varepsilon_\nu). \tag{5.9}$$

From (5.6b) and (5.7b) we get with the aid of (5.8)

$$\varepsilon_\nu(\mathbf{w}_\nu - V_\nu \boldsymbol{\xi}) = -\varepsilon_\nu (C^T \Phi)^{-1}[\boldsymbol{\gamma} - D\boldsymbol{\xi} - C^T(\mathbf{w}^P - V^P \boldsymbol{\xi})].$$

After cancelling the ε_ν factor above, we use (5.6a) and (5.7a) to get from (2.3)

$$\mathbf{x} = \mathbf{w} - V\boldsymbol{\xi} = (\mathbf{w}^P - V^P \boldsymbol{\xi}) + \Phi(C^T \Phi)^{-1}[\boldsymbol{\gamma} - D\boldsymbol{\xi} - C^T(\mathbf{w}^P - V^P \boldsymbol{\xi})]. \tag{5.10a}$$

Setting

$$\mathbf{x}^P \equiv \mathbf{w}^P - V^P \boldsymbol{\xi}_0 \tag{5.10b}$$

and using (5.9) we find on comparison with (3.7a, b) that

$$\mathbf{x} = \mathbf{x}_0 + \mathcal{O}(\varepsilon_\nu). \tag{5.11}$$

Thus we conclude from (5.9) and (5.11), that the bordering algorithm (2.1a, b)–(2.3) applied to solve (1.1) with singular A satisfying (3.1) yields an $\mathcal{O}(\varepsilon_\nu)$ accurate solution *provided* an LU-factorization is used to yield (4.1) with ε_ν nonsingular and satisfying (5.1) and (5.3). We must observe that since (5.3) holds the errors in computing Φ, Ψ, V and $\mathbf{w}$ are also at most $\mathcal{O}(\varepsilon_0)$ and so the estimates in (5.9) and (5.11) remain valid when these inexact values are used in the above derivation.

When A is singular or near singular and we use the bordering algorithm, even with a good pivot strategy, some loss in accuracy is to be expected. This occurs when forming $\mathbf{x}$ as in (2.3) and is due to cancellation of leading digits. One way to circumvent these errors is to use the "singular A algorithm" given by (3.5) and (3.7). This can be quite practical when A is sensed to be near singular since the bases of null vectors Φ and Ψ are easily determined as in (4.4a,b).

6. Path following applications. The computational linear algebra problems discussed above were in fact motivated by the path following applications we now describe. We shall also show that when these applications come from consistent, stable, discrete approximations to a broad class of differential equation problems then very efficient *partial* pivoting procedures can be used for the Gaussian elimination to determine the nullity and the LU-factorization of (4.1). Full pivoting need only be invoked in processing the final "block" in the special banded or block tridiagonal systems that arise.

We assume that some nonlinear operator equation has been discretized and that the resulting finite dimensional problem has the form

$$\mathbf{G}(\mathbf{u}, \lambda) = \mathbf{0}. \tag{6.1}$$

Here $\mathbf{G}: \mathbb{R}^{N+1} \to \mathbb{R}^N$ is an appropriately smooth function, $\mathbf{u} \in \mathbb{R}^N$ and $\lambda \in \mathbb{R}$. We are concerned with computing families or "paths" of solutions, $(\mathbf{u}, \lambda)$, of (6.1). One of

the most effective ways to do this is known as Euler–Newton continuation. Thus if $(\mathbf{u}(\lambda), \lambda)$ represents an arc of solutions of (6.1) the tangent to this arc is in the direction $(\mathbf{u}_\lambda(\lambda), 1)$, where $\mathbf{u}_\lambda$ satisfies

$$\mathbf{G}_\mathbf{u}(\mathbf{u}(\lambda), \lambda)\mathbf{u}_\lambda = -\mathbf{G}_\lambda(\mathbf{u}(\lambda), \lambda), \tag{6.2}$$

then a good approximation to the solution $(\mathbf{u}(\lambda + \Delta\lambda), \lambda + \Delta\lambda)$ is given by

$$\mathbf{u}^0(\lambda + \Delta\lambda) = \mathbf{u}(\lambda) + \Delta\lambda\, \mathbf{u}_\lambda(\lambda). \tag{6.3}$$

This is the approximation obtained by using one "Euler step" to solve (6.2) numerically. Now we use this approximation as the first iterate in Newton's method to solve (6.1) at $\lambda + \Delta\lambda$:

$$\mathbf{G}_\mathbf{u}^\nu \Delta\mathbf{u}^\nu = -\mathbf{G}^\nu, \qquad \mathbf{u}^{\nu+1} = \mathbf{u}^\nu + \Delta\mathbf{u}^\nu. \tag{6.4a, b}$$

Here we have used $\mathbf{G}_\mathbf{u}^\nu \equiv \mathbf{G}_\mathbf{u}(\mathbf{u}^\nu(\lambda + \Delta\lambda), \lambda + \Delta\lambda)$, $\mathbf{G}_\lambda^\nu \equiv \mathbf{G}_\lambda(\mathbf{u}^\nu(\lambda + \Delta\lambda), \lambda + \Delta\lambda)$.

This procedure generally works extremely well. But during the course of the continuation or path following, difficulties occur if the $N \times N$ Jacobian matrix $\mathbf{G}_\mathbf{u}$ becomes singular. Such singular points are not uncommon and indeed they can be extremely important in the applications. Their most frequent occurrence is at *limit points*, say $(\mathbf{u}_0, \lambda_0)$ where

$$\dim \mathcal{N}(\mathbf{G}_\mathbf{u}(\mathbf{u}_0, \lambda_0)) = 1, \qquad \mathbf{G}_\lambda(\mathbf{u}_0, \lambda_0) \notin \text{Range}\,[\mathbf{G}_\mathbf{u}(\mathbf{u}_0, \lambda_0)]. \tag{6.5a, b}$$

Geometrically this occurs at points where the tangent to the path becomes vertical on a $\mathbf{u}$ versus λ graph. At bifurcation points we also have (6.5a) but not (6.5b). In a number of other important cases the null space dimension is greater than one. These include multiple limit points [1], Hopf bifurcation and period doubling bifurcations in the study of periodic solution branches [2], fold following [3], and critical boundary paths [9]. In all of these cases there are additional parameters in the problem formulation (for example: the period, T, of the periodic solution, etc.) or there are natural parameters that can be introduced. Indeed the idea of introducing additional parameters leads to our current study and the application of bordering.

At limit points the difficulties are easily eliminated, in principal, by simply using some arclength-like parameter to describe the path. Thus we imagine a family or arc of solutions of (6.1) given by $(\mathbf{u}(s), \lambda(s))$ for $s \in \mathcal{I} \subset \mathbb{R}$. Let $(\dot{\mathbf{u}}_0, \dot{\lambda}_0)$ be the tangent to the solution arc for $s = s_0$. Then we consider the scalar constraint

$$N(\mathbf{u}, \lambda, s) \equiv \dot{\mathbf{u}}_0 \cdot (\mathbf{u} - \mathbf{u}_0) + \dot{\lambda}_0(\lambda - \lambda_0) - (s - s_0) = 0. \tag{6.6}$$

Now we seek to solve (6.1) and (6.6) simultaneously for $|s - s_0|$ not too large. We call this procedure pseudo-arclength continuation [7], since if we let $s \to s_0$, then (6.6) implies

$$\|\dot{\mathbf{u}}_0\|_2^2 + |\dot{\lambda}_0|^2 = 1.$$

Thus $s - s_0$ in (6.6) is a local approximation to arclength.

If Newton's method is used to solve (6.1) and (6.6) simultaneously for $(\mathbf{u}(s), \lambda(s))$ we get for the Newton corrections $(\Delta u^\nu, \Delta\lambda^\nu)$ a linear system of the form (1.1) with

$$\mathcal{A}^\nu \equiv \begin{pmatrix} \mathbf{G}_\mathbf{u}^\nu & \mathbf{G}_\lambda^\nu \\ \dot{\mathbf{u}}_0^T & \dot{\lambda}_0 \end{pmatrix}. \tag{6.7}$$

When (6.5a, b) holds we can easily show that $\mathcal{A}^\nu$ evaluated at $(\mathbf{u}_0, \lambda_0)$ is nonsingular. Thus for $(\mathbf{u}^\nu, \lambda^\nu)$ close to $(\mathbf{u}_0, \lambda_0)$, $\mathcal{A}^\nu$ is nonsingular while $\mathbf{G}^\nu$ is close to the singular matrix $\mathbf{G}_\mathbf{u}(\mathbf{u}_0, \lambda_0)$. In this situation our analysis in § 5 with $\nu = 1$, is applicable to the

solution of the Newton equations. As previously mentioned this $\nu = 1$ case is also discussed in [8]. However in the higher dimensional periodic bifurcation and fold following cases [2], [3], [9], where $\nu = 2$, our current analysis is also applicable. The inflation procedures used to get systems of order $N+2$ are along the same lines as the above but considerably more complicated so we do not sketch them here.

To show how partial pivoting can be justified for an important class of singular $A = \mathbf{G}_{\mathbf{u}}$ we consider the linear boundary value problem

$$\begin{aligned} &B_a(\lambda)\mathbf{y}(a) = \boldsymbol{\beta}_a, \\ &\mathbf{y}' - A(\lambda, t)\mathbf{y} = \mathbf{f}(t), \qquad a < t < b, \\ &B_b(\lambda)\mathbf{y}(b) = \boldsymbol{\beta}_b. \end{aligned} \tag{6.8a, b, c}$$

Here $\mathbf{y}, \mathbf{f} \in \mathbb{R}^m$, B_a is $p \times m$, B_b is $q \times m$, $p+q=m$, $\boldsymbol{\beta}_a \in \mathbb{R}^p$ and $\boldsymbol{\beta}_b \in \mathbb{R}^q$. Of course in applications this may represent the linearization of some nonlinear problem but that is not important for our current discussion. If $Y(\lambda\,;t)$ is a fundamental solution matrix for (6.8b) then it is well known [6] that (6.8a, b, c) has a unique solution for each $(\mathbf{f}, \boldsymbol{\beta}_a, \boldsymbol{\beta}_b)$ if and only if

$$F(\lambda) \equiv \det \begin{pmatrix} B_a(\lambda)Y(\lambda\,;a) \\ B_b(\lambda)Y(\lambda\,;b) \end{pmatrix} \neq 0. \tag{6.9a}$$

Indeed the "eigenvalues," λ, of the homogeneous case of (6.8a, b, c) are just the roots of

$$F(\lambda) = 0. \tag{6.9b}$$

If $A(\lambda, t)$, $B_a(\lambda)$ and $B_b(\lambda)$ are analytic in λ then the same is true of $F(\lambda)$ and we have that the eigenvalues are isolated and accumulate only at ∞ (or else all values are eigenvalues). We make one further assumption:

(6.10) The eigenvalues are nonconstant functions of the coefficients in $B_b(\lambda)$.

Suppose we approximate the solution of (6.8a, b, c) by using the Box scheme:

$$\begin{aligned} &B_a(\lambda)\mathbf{y}_0 = \boldsymbol{\beta}_a, \\ &\frac{\mathbf{y}_j - \mathbf{y}_{j-1}}{h} - A(\lambda, t_{j-1/2})\frac{\mathbf{y}_j + \mathbf{y}_{j-1}}{2} = \mathbf{f}(t_{j-1/2}), \qquad 1 \leqq j \leqq J, \\ &B_b(\lambda)\mathbf{y}_J = \boldsymbol{\beta}_b. \end{aligned} \tag{6.11a, b, c}$$

If we order the difference equations as indicated and denote the $N = m(J+1)$ unknowns as $\mathbf{y}^h \equiv (\mathbf{y}_0^T, \cdots, \mathbf{y}_J^T)^T$, then (6.11a, b, c) can be written as

$$\mathbb{A}_h(\lambda)\mathbf{y}^h = \mathbf{f}^h, \tag{6.12}$$

where $\mathbb{A}_h(\lambda)$ has the block tridiagonal structure

$$\begin{aligned} &\mathbb{A}_h(\lambda) \equiv [B_j, A_j, C_j] \\ &B_j \equiv \begin{pmatrix} \mathrm{X} \\ 0 \end{pmatrix} \begin{matrix} \}p \\ \}q \end{matrix}, \qquad C_j \equiv \begin{pmatrix} 0 \\ \mathrm{X} \end{pmatrix} \begin{matrix} \}p \\ \}q \end{matrix}. \end{aligned} \tag{6.13a}$$

In particular,

$$A_0(\lambda) \equiv \begin{pmatrix} B_a(\lambda) \\ \mathrm{X} \end{pmatrix} \}q, \qquad A_J(\lambda) \equiv \begin{pmatrix} \mathrm{X} \\ B_b(\lambda) \end{pmatrix} \}p. \tag{6.13b}$$

Now from [6, Thm. 2.9] we conclude that: *if* λ *is not an eigenvalue* (i.e., root of (6.9b)) *then for some* $h_0 > 0$ *and all* $h \leqq h_0$ *the matrices* $\{\mathbb{A}_h(\lambda)\}$ *are nonsingular with*

$$\|\mathbb{A}_h^{-1}(\lambda)\| < K. \tag{6.14}$$

It is further shown in [5, § 5], that when (6.14) holds a *restricted form* of pivoting yields LU-factorizations of the form:

$$\mathbb{A}_h(\lambda) = [\beta_j, \delta_j, 0][0, \alpha_j, \gamma_j], \qquad 0 \leqq j \leqq J, \tag{6.15}$$

where α_j, β_j, δ_j and γ_j are $m \times m$ matrices. The restriction on the pivoting is such that it allows interchanges *within* the set of m equations (6.11b) at any fixed net point (a restricted form of row-pivoting for $\mathbb{A}_h$) and it allows interchanges within the m variables $\mathbf{y}_j$ associated at each net point (a restricted form of column-pivoting). Both of these pivoting strategies insure that the zero structure of the B_j and C_j are preserved in the β_j and γ_j, respectively. To obtain factorizations of the form (4.1), (4.3) we use these techniques with δ_j and α_j in the triangular forms

$$\delta_j \equiv \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ X & & 1 \end{pmatrix}, \qquad \alpha_j \equiv \begin{pmatrix} x & & X \\ & \ddots & \\ 0 & & x \end{pmatrix}. \tag{6.16}$$

We note, by (6.13b) that the coefficients defining $B_b(\lambda)$ do not enter into the elimination procedure until the final $m \times m$ block is to be factored as

$$\delta_J \alpha_J = A_J(\lambda) - \beta_J \gamma_{J-1} \equiv \begin{pmatrix} X \\ \hline B_b(\lambda) \end{pmatrix}. \tag{6.17}$$

Furthermore since β_J contains zeros in the last q rows (as in all the B_j of (6.13a)), the last q rows on the right-hand side of (6.17) are just those of $B_b(\lambda)$.

Now if λ is an eigenvalue we could invoke (6.10) to ensure that the elimination does not fail until all but the last block has been processed. This is simply done by changing the data in $B_b(\lambda)$ so that the current value of λ is no longer an eigenvalue. Then the complete factorization is valid. But this uses data from $B_b(\lambda)$ only in the final block. Thus in the singular case (i.e., at an eigenvalue) we need not actually change any data—we merely use full pivoting in the final factorization of (6.17). A similar argument can be used to justify these techniques on discrete approximations for many other classes of functional equations including elliptic boundary value problems. The crucial requirement is some analogue of (6.10) which insures that changing the boundary conditions changes the eigenvalues.

Acknowledgments. I wish to thank the referees for helpful comments on the original version of this paper.

REFERENCES

[1] D. W. DECKER AND H. B. KELLER, *Multiple limit point bifurcation*, J. Math. Anal. Appl., 75 (1980), pp. 417–430.

[2] E. DOEDEL, A. JEPSON AND H. B. KELLER, *Paths of periodic solutions and their computation*, in preparation.

[3] J. FIER AND H. B. KELLER, *Follow the folds*, in preparation.

[4] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1975.

[5] H. B. KELLER, *Accurate difference methods for nonlinear two point boundary value problems*, SIAM J. Numer. Anal., 11 (1974), pp. 305–320.

[6] ———, *Numerical Solution of Two Point Boundary Value Problems*, CBMS Regional Conference Series in Applied Mathematics 24, Society for Industrial and Applied Mathematics, Philadelphia, 1976.
[7] ———, *Numerical Solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359–384.
[8] ———, *Practical procedures in path following near limit points*, to appear in Computing Methods in Applied Sciences and Engineering, R. Glowinski and J. L. Lions, eds., North-Holland Pub. Co., Amsterdam, 1982.
[9] W. RHEINBOLDT, *Computation of critical boundaries on equilibrium manifolds*, SIAM J. Numer. Anal., 19 (1982), pp. 653–669.
[10] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0002 $01.25/0

ITERATIVE SOLUTION OF LINEAR EQUATIONS IN ODE CODES*

C. W. GEAR† AND Y. SAAD‡

Abstract. Each integration step of a stiff equation involves the solution of a nonlinear equation, usually by a quasi-Newton method which leads to a set of linear problems involving the Jacobian, J, of the differential equation. Iterative methods for these linear equations are studied. Of particular interest are methods which do not require an explicit Jacobian but can work directly with differences of function values using $J\delta \cong f(x+\delta)-f(x)$. Some numerical experiments using a modification of LSODE are reported.

Key words. Newton, nonlinear equations, stiff equations, initial value problems, ordinary differential equations

1. Introduction. The solution of the large sparse nonlinear equations arising at each integration step constitutes an important part of the cost of numerical integrators for large, stiff systems of ordinary differential equations (ODEs). To handle stiffness, an integrator must be implicit (or, equivalently, involve an approximation to the inverse of the Jacobian system). We will discuss predictor-corrector linear multistep schemes in which an explicit predictor is used to get a first approximation for an implicit corrector, although the techniques to be discussed are applicable to other schemes such as implicit Runge–Kutta.

For illustration we will frequently use the forward Euler/backward Euler pair, as they introduce all of the difficulties associated with the linear systems, although in practice higher-order methods are used to achieve typical accuracy of three to five digits. The forward Euler predictor for the differential equation $y'=f(y,t)$, where f is a Lipschitz continuous mapping from $\mathbb{R}^{N+1}$ to $\mathbb{R}^N$, is

$$y_n^{(0)} = p_n = y_{n-1} + h_{n-1} y'_{n-1}, \tag{1.1}$$

where y_n is an approximation to $y(t_n)$, y'_n an approximation to $f(y_n, t_n)$ and $p_n = y_n^{(0)}$ is the predicted or first approximation to y_n. The stepsize $h_{n-1} = t_n - t_{n-1}$ may change from step to step, so we use a subscript in (1.1) to emphasize this fact, although we will drop it from now on. The backward Euler corrector to be solved at each step is

$$y_n = y_{n-1} + hf(y_n, t_n) \tag{1.2}$$

which is, in general, a nonlinear equation in y_n. If solved by Newton's method, we get the iteration

$$y_n^{(q+1)} = y_n^{(q)} + x^{(q)}, \tag{1.3}$$

where $x^{(q)}$ is the solution of

$$(I - hJ)\, x^{(q)} = y_{n-1} + hf(y_n^{(q)}, t_n) - y_n^{(q)} = b^{(q)}$$

or

$$Ax^{(q)} = b^{(q)}. \tag{1.4}$$

* Received by the editors February 10, 1981, and in final revised form August 12, 1982. This research was supported in part by the U.S. Department of Energy, under grant DOE AC0276ER02383.

† Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

‡Currently at the Department of Computer Science, Yale University, New Haven, Connecticut 06520. This work was done while the author was at the University of Illinois.

Here, J is the Jacobian $\partial f/\partial y$ evaluated at a suitable point. Note that A is just a shift and scale of this matrix. (In typical problems, J is slowly varying from step to step, so it is not important that J be reevaluated frequently in practical codes because iteration (1.3) converges at a reasonable rate when J is near the Jacobian.)

Systems (1.4) are very often sparse and nonsymmetric, and they are usually solved by direct methods. Little has been done so far on treating these linear systems by iterative methods. The symmetric case has been recently considered by Miranker and Chern [10] using conjugate gradient methods. Iterative techniques are advantageous in many cases, particularly when the number of equations is very large and when the Jacobian is not easily computable.

Although the use of iterative techniques might seem costly at first compared to direct methods, because only the right-hand side of (1.4) changes frequently, there are several reasons why it is not so. One reason is that the solutions of the linear systems need not be very accurate—the predicted value p_n is fairly accurate so the Newton corrections $x^{(q)}, q = 0, 1, \cdots$ made by (1.3) are small compared to $y_n^{(q)}$ and relatively few digits of accuracy are needed in the $x^{(q)}$. In fact, the aim of the corrector step is to attempt to gain stability by eliminating the components of the error vector, corresponding to the large eigenvalues of the Jacobian which have been amplified by the predictor.

This can best be seen by considering the simple test ODE $y' = Jy$ where $J = \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$. If y and λ are the corresponding scalar values of any one component of this problem, we have (assuming that $y_n' = \lambda y_n$)

$$y_n^{(0)} = p_n = (1 + h\lambda) y_{n-1}, \tag{1.5}$$

$$y_n^{(q+1)} = y_n^{(q)} + x^{(q)}, \tag{1.6}$$

$$(1 - h\lambda) x^{(q)} = y_{n-1} - (1 - h\lambda) y_n^{(q)}. \tag{1.7}$$

In an iterative scheme, (1.7) is solved only approximately. If $h\lambda$ is small, $y_n^{(0)}$ is an accurate approximation to y_n; the right-hand side of (1.7) is small for $q = 0$ (it is $(h\lambda)^2 y_{n-1}$). For these cases it is necessary only that the $x^{(q)}$ be small. If $h\lambda$ is large and negative, y_{n-1}, which is approximately $y_0 e^{\lambda t_{n-1}}$, is small for $t_{n-1} > 0$. However, $y_n^{(0)}$ will be amplified by $O(h\lambda)$. In this case, the right-hand side of (1.7) will be $O(h\lambda)$ for $q = 0$, so (1.7) must be solved more accurately to reduce y_n to its correct value in a small number of Newton iterations.

The preceding paragraph raises the problem of choosing an iterative technique more suitable for stiff ODE methods. More precisely, the question is: which iterative techniques possess the property of being more accurate in the eigenspace corresponding to the large eigenvalues? The class of projection methods on the Krylov subspaces $K_m = \operatorname{span}[r_0, Ar_0, \cdots, A^{m-1} r_0]$ possesses the above property. The best example of a projection method on the Krylov subspace is the conjugate gradient algorithm for symmetric positive definite matrices. This method can be extended to unsymmetric matrices by requiring that the residual vectors form an orthogonal sequence, a property which is known to be true in the symmetric case.

In the context of stiff ODEs, the projection method must be used at each step. In each projection process a basis of the Krylov subspace is generated along with a representation of (the restriction of) the matrix A in this subspace K_m. Here we obtain further simplifications in regard to the integration of stiff ODEs:

(i) The basis of the Krylov subspace and the $m \times m$ representation of A need not be generated at each integration step; once they are computed, they can be saved and used in several subsequent steps as long as J is slowly changing.

(ii) When the stepsize is changed by the integrator the shift of the Jacobian matrix changes, but this does not change the Krylov subspace so it need not be recomputed, only the factorization of the $m \times m$ representation of A in the subspace must be recomputed and, as will be seen, this is not costly.

(iii) There is no need to store the Jacobian in any fashion; all that is needed is to be able to compute $z = Jv$ for any given v and this may be easily realized by the approximation $Jv = [f(y + \delta v) - f(v)]/\delta$.

In this paper we examine the application of iterative methods based on the Krylov subspaces to ODEs and analyze the linear constant coefficient case. We report the results of some numerical experiments carried out by including some of the techniques discussed in a widely available code, LSODE. This problem may be a fruitful area for future work, and some additional modifications are discussed which could lead to further improvements in ODE codes.

2. The Krylov subspace method for solving linear systems.

2.1. The Krylov subspace projection method. This subsection summarizes the earlier paper Saad [11]. Consider the system

$$Ax = b, \tag{2.1}$$

where A is $N \times N$ (nonsymmetric). Suppose that $V_m = [v_1, \cdots, v_m]$ is an orthonormal system in $\mathbb{R}^N$ and let x_0 be an initial guess of the solution. By definition, a projection method on the subspace $K_m = \text{span}\,[V_m]$ is a method that obtains a solution of the form $x_m = x_0 + V_m y_m$ such that

$$V_m^T(b - Ax_m) = 0 \tag{2.2}$$

therefore

$$x_m = x_0 + V_m(V_m^T A V_m)^{-1} V_m^T r_0, \tag{2.3}$$

where $r_0 = b - Ax_0$ is the initial residual. Note that the $m \times m$ matrix $H_m = V_m^T A V_m$ is the representation of the linear application $\Pi_m A|_{K_m}$ where Π_m is the orthogonal projector onto K_m.

Of particular interest is the Krylov subspace projection method where K_m is the Krylov subspace $K_m = \text{span}\,[r_0, Ar_0, \cdots, A^{m-1}r_0]$. The following algorithm, for the solution of eigenvalue problems, described first by Arnoldi, provides a simple way of building an orthogonal basis $v_1, \cdots, v_m$ of K_m along with the representation $H_m = V_m^T A V_m$:

Algorithm.

1. Compute $r_0 = b - Ax_0$ and take $v_1 = r_0/\|r_0\|$.
2. For $j = 1, 2, \cdots, m$ do

$$\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i, \tag{2.4}$$

where

$$h_{ij} = (Av_j, v_i) \tag{2.5}$$

and

$$v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}},$$

where

$$h_{j+1,j} = \|\hat{v}_{j+1}\|. \tag{2.6}$$

An important fact is that the matrix $V_m^T A V_m$ is the $m \times m$ upper Hessenberg matrix whose nonzero elements are given by (2.5) and (2.6). Furthermore the vector $V_m^T r_0$ in (2.3) is equal to $\|r_0\| e_1$ where $e_1 = (1, 0, 0, \cdots, 0)^T$.

The approximate solution x_m is therefore easy to compute and (2.3) becomes

$$x_m = x_0 + \|r_0\| V_m H_m^{-1} e_1. \tag{2.7}$$

The fact that H_m is Hessenberg means that the operator H_m^{-1} is easy to compute via an LU decomposition of H_m. However, if the dimension m is small, an LU decomposition is inexpensive even if H_m is not Hessenberg. This is imporant for the technique discussed in the next paragraph.

The jth step of Step 2 in the above algorithm represents a Schmidt orthogonalization of Av_j against all the previous vectors $v_1, v_2, \cdots, v_j$. When j increases, this becomes intolerably time consuming and above all requires too much memory. A natural idea to escape this difficulty is to orthogonalize Av_j against the previous p vectors where p is some small integer. The algorithm which is derived from this produces a sequence of vectors $v_1, v_2, \cdots, v_m$ which is no longer orthonormal but satisfies the incomplete orthogonality property

$$(v_i, v_j) = 0 \quad \text{for } 0 < |i-j| < p. \tag{2.8}$$

The first algorithm will be referred to as Arnoldi's algorithm and the second as the incomplete orthogonalization method (IOM). The only difference between the two algorithms lies in (2.4) which for IOM becomes

$$\hat{v}_{j+1} = Av_j - \sum_{i=i_j}^{j} h_{ij} v_i, \tag{2.9}$$

where $i_j = \max\{1, j-p\}$.

The approximate solution x_m is again defined by (2.7). This second method can be interpreted as an oblique projection method on the Krylov subspace K_m and is analyzed in Saad [13]. It is important to mention that although all vectors v_i are needed for forming the approximate solution x_m by formula (2.7), only the most recent $p+1$ are necessary at step j to form v_{j+1}. This means that one may store the vectors v_j in secondary storage as soon as they are computed, and read them back at the final stage when the combination $x_m = x_0 + V_m y_m$ is formed (with $y_m = \|r_0\| H_m^{-1} e_1$).

We point out that a number of similar algorithms have also been developed by Jea and Young [8]. One of their methods (ORTHORES) is theoretically equivalent to ours but no secondary storage is used. Note that their methods may break down if the symmetric part of A is not positive definite. It is also possible in our case to derive an algorithm which does not require secondary storage preserving the benefits of forming the approximation x_m in a stable way, thanks to an implicit stable factorization of H_m. We omit the details which can be found in Saad [14].

Thus far nothing has been said about how to choose the two parameters m and p. The dimension m of K_m must be, ideally, equal to N in which case the approximate solution x_m is exactly $A^{-1}b$. However, good accuracy is achieved for m far less than N. In practice an upper bound $m_{\max}$ is fixed a priori and the residual norms are estimated periodically by using the formula (see [11])

$$\|b - Ax_m\| = h_{m+1,m} |y_m^T e_m|. \tag{2.10}$$

As soon as the residual norm estimate is small enough or the dimension $m_{\max}$ is reached, the algorithm stops. After that one could restart by taking x_0 equal to the last approximate solution. In stiff ODEs, however, we shall use the algorithm without restarting because the outer Newton iteration will be repeated if the residual is insufficiently small. The choice of the parameter p, the number of vectors against which the vector Av_j is orthogonalized at each step, is more difficult. The optimal p for a given problem is related, in a nonobvious way, to the degree of symmetry of A. When A is symmetric, then, with $p = 2$, the algorithm is equivalent to the conjugate gradient method. Thus, when A is almost symmetric, the choice $p = 2$ will be the most efficient. In any case, p should not exceed 10 or 15 and should be taken smaller if A is nearly symmetric and larger if not. Note that there is also a possibility of reducing p during the integration by examining the matrix H_m: p should be such that the h_{ij}'s, $i < j - p$ can be considered small in comparison with the h_{ij}'s for $j - p \leqq i \leqq j$. (See Saad [14].)

2.2. Application to stiff ODEs. All methods for solving stiff ODEs give rise to one or more linear systems involving the Jacobian. If the backward differentiation formulas are used, namely,

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h\beta_0 f(y_n) \tag{2.11}$$

the linear system is

$$(I - h\beta_0 J)x = b, \tag{2.12}$$

where b is the residual of (2.11) and x is the increment to the current approximation to y_n.

The matrix J appearing in (2.12) might be difficult to compute and above all to store. However, a look at the algorithm of § 2.1 shows that the matrix $A = I - h\beta_0 J$ is not really needed explicitly. All that is needed is to be able to compute the vector $u = Av$ for any given v, and this can be achieved through the formula

$$Jv \simeq \frac{1}{\varepsilon}[f(t, y + \varepsilon v) - f(t, y)]$$

which will necessitate only one function evaluation if $f(t, y)$ is already available from the previous computations.

Suppose that at a certain integration step t_n we use the Krylov subspace method for the system $Ax = b$, where $A = I - \beta_0 hJ$. This means that we generate a sequence of vectors $V_m = [v_1, v_2, \cdots, v_m]$ and a Hessenberg matrix H_m, representing A in the subspace $K_m = \text{span}\,[V_m]$, and that we solve the problem $Ax = b$ at the step t_n by (2.7). The important question is whether we can use the same subspace K_m and the representation H_m to solve the following problems arising at steps $t_{n+1}, t_{n+2}, \cdots$. There are two separate problems here. The first arises if the Jacobian changes significantly from step to step; the second arises because the shift $h\beta_0$ does change from step to step. In typical stiff equations the stepsize has to be sufficiently small for integration accuracy so that the significant ("large") components of the Jacobian do not change much from step to step (that is not to say that equations cannot be constructed in which J changes rapidly, but they do not occur too often). This fact is used successfully in most current codes that only update the Jacobian periodically and can be used equally well if iterative techniques are used to solve the linear system. Hence, we are

concerned at subsequent steps with solving

$$A'x = b', \tag{2.13}$$

where

$$A' = I - h\beta_0 J. \tag{2.14}$$

Note that the stepsize h' and the coefficient β_0' of the method can be very different from h and β_0. According to (2.3), if we apply Arnoldi's algorithm on K_m to solve the new system (2.13), we obtain the approximate solution

$$x_m = x_0 + V_m(V_m^T A' V_m)^{-1} V_m^T r_0, \tag{2.15}$$

where $r_0 = b' - A'x_0$ is the initial residual (x_0 is a initial guess to the solution of (2.13)). The matrix $V_m^T A' V_m$ may, at first, look difficult to form, but it is easy to compute $V_m^T A' V_m$ using the technique used by Enright [3]. Let us consider $(\beta_0 h/\beta_0' h')V_m^T A' V_m$ instead of $V_m^T A' V_m$. If we set $\alpha = \beta_0 h/\beta_0' h'$ we have

$$\alpha V_m^T A' V_m = \alpha V_m^T [I - \beta_0' h' J] V_m.$$

Hence

$$\alpha V_m^T A' V_m = V_m^T (I - \beta_0 h J) V_m + (\alpha - 1) V_m^T V_m. \tag{2.16}$$

Since $V_m^T V_m = I$, we obtain

$$\alpha V_m^T A' V_m = H_m + (\alpha - 1)I$$

and (2.15) becomes

$$x_m = x_0 + \alpha V_m (H_m + (\alpha - 1)I)^{-1} q_m, \tag{2.17}$$

where

$$q_m = V_m^T r_0. \tag{2.18}$$

Note that this technique is not dependent on H_m being Hessenberg. If the dimension m is small, the factorization of the operator $(H_m + (\alpha - 1)I)$ is inexpensive.

A few remarks are in order. First, notice that no function evaluation is required to build the approximate solution x_m. In fact, all that is needed is to compute $\alpha V_m^T r_0$, to solve an $m \times m$ system having a Hessenberg form, and to accumulate $V_m y_m$ with $y_m = \alpha(H_m + (\alpha - 1)I)^{-1} q_m$. Secondly, the derivation of (2.17) assumes that V_m is an orthonormal system, but this will not be true in the case of the IOM method. However, on multiplying (2.16) on the left by $(V_m^T V_m)^{-1}$, we see that (2.17) is still valid provided we replace q_m in (2.18) by $\tilde{q}_m = (V_m^T V_m)^{-1} V_m^T r_0$, and H_m by $(V_m^T V_m)^{-1} V_m^T (I - \beta_0 h J) V_m$. This final matrix can be shown to be a rank one perturbation of H_m, (see Saad [11]), and is the matrix representation of the section of A in K_m, that is, of $\Pi_m A|_{K_m}$ where Π_m is the orthogonal projector onto K_m. It is usually well approximated by H_m (Saad [13]). Note that it would be needlessly expensive to compute $\tilde{q}_m$ because we only need a rough approximation to the solution of (2.13) and this can be achieved by taking $\tilde{q}_m \simeq V_m^T r_0$. Practically, however, $V_m^T r_0$ is not the best choice. If we notice that at the earlier step t, when V_m has been generated, the residual r_0 is, apart from a multiplicative constant, equal to v_1, we may consider replacing $V_m^T r_0$ by $v_1^T r_0 e_1$, or even better, taking $\tilde{q}_m = \sum_{i=1}^{l} (r_0^T v_i) e_i$ where l is some integer of the same order as p, e.g., $l = \max(5, p+1)$.

3. The problem of stability.

3.1. Accuracy of the solution in the dominant subspace.

3.1.1. Basic property. An important property of the Krylov subspace method is that it provides an approximate solution $\tilde{x}$ to $Ax = b$, which is more accurate along the eigenvectors of A corresponding to the eigenvalues lying in the outermost part of the spectrum. In the case of stiff equations we are particularly interested in the dominant subspace which corresponds to the right part of the shaded spectrum in Fig. 1 since $A = I - h\beta_0 J$.

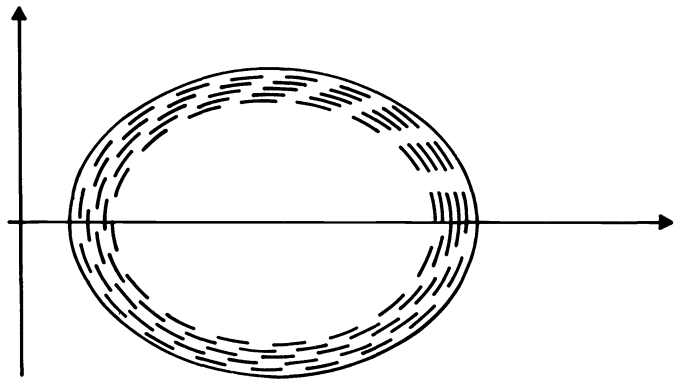


FIG. 1. *The shaded region indicates the part of the spectrum where the solution is accurate.*

Suppose that the basic algorithm of Arnoldi is used to solve the system $Ax = b$, and let x_m be the approximate solution obtained at the mth step. Let us denote by Π_m the orthogonal projector onto the subspace K_m, and by u_i, an eigenvector of A associated with λ_i. It is known that the distance $\|(I - \Pi_m)u_i\|$ between u_i and the subspace K_m decreases rapidly to zero with m, for the eigenvectors u_i associated with the eigenvalues situated in the outermost part of the spectrum (see Saad [12]). (Throughout this paper $\|\cdot\|$ denotes the Euclidean norm.) The following lemma gives an explanation of the above-mentioned property.

LEMMA 1. *The residual $r_m = b - Ax_m$ is such that*

$$|(r_m, u_i)| \leqq \|r_m\| \|(I - \Pi_m)u_i\|. \tag{3.1}$$

Proof. $(r_m, u_i) = (r_m, u_i - \Pi_m u_i)$, because $\Pi_m u_i \in K_m$ and the residual r_m is orthogonal to K_m. The result (3.1) follows immediately. Q.E.D.

3.1.2. Symmetric case. We shall first consider the consequences of Lemma 1 when A is a symmetric matrix.

PROPOSITION 1. *Let $\lambda_1 > \lambda_2 > \cdots > \lambda_N$ be the eigenvalues of A, and let θ_i be the acute angle between the initial vector v_1 and the eigenvector u_i, $i < N$. Let*

$$\gamma_i = 1 + 2\frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_N}, \quad K_i = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_N}{\lambda_j - \lambda_i}, \quad K_1 = 1.$$

Then

$$|(r_m, u_i)| \leqq \|r_m\| \frac{K_i}{T_{m-i}(\gamma_i)} \tan \theta_i, \tag{3.2}$$

where T_k represents the kth degree Chebyshev polynomial of the first kind.

Proof. The result follows from (3.1) and an inequality for $\|(I-\Pi_m)u_i\|$ established in [12]. Q.E.D.

The above inequalities are of interest only when i is small compared with N because then the quantity $K_i/T_{m-i}(\gamma_i)$ decays rapidly as m increases, i being fixed. The same result can be shown when the eigenvalues are labelled in increasing order. As a consequence, $|(r_m, u_i)|/\|r_m\|$ is very small when λ_i belongs to either extremity of the spectrum and becomes larger when λ_i is in the middle of the spectrum.

Since we have $\|r_m\| = |\sum_{i=1}^{N} (r_m, u_i)^2|^{1/2}$, the above proposition means that for λ_i belonging to the outermost part of the spectrum, the component (r_m, u_i) of the residual along the eigenvector u_i is much smaller than the average $\|r_m\|$.

It is natural to ask whether a similar result holds for the error vector (x^*-x_m), where x^* is the exact solution of $Ax-b$. The answer is given by the following corollary.

COROLLARY 1. *The error vector x^*-x_m is such that*

$$|(x^*-x_m, u_i)| \leqq \frac{\|A\|}{|\lambda_i|}\|x^*-x_m\|\frac{K_i}{T_{m-i}(\lambda_i)}\tan\theta_i. \tag{3.3}$$

Proof. Using $r_m = A(x^*-x_m)$ and (3.2), we get

$$|(A(x^*-x_m), u_i)| \leqq \|A(x^*-x_m)\|K_i\frac{\tan\theta_i}{T_{m-i}(\gamma_i)}.$$

Hence,

$$|(x^*-x_m, Au_i)| \leqq \|A\|\,\|x^*-x_m\|K_i\frac{\tan\theta_i}{T_{m-i}(\gamma_i)}.$$

But $Au_i = \lambda_i u_i$, which gives the desired inequality. Q.E.D.

Note that when A is positive definite, the coefficient $\|A\|/|\lambda_i|$ is just λ_1/λ_i.

The above result can be interpreted in the same way as the result for the residual vector.

3.1.3. Unsymmetric case. When A is nonsymmetric, the exploitation of Lemma 1 is more complicated because the distance $\|(I-\Pi_m)u_i\|$ is more difficult to estimate. A result as simple as (3.2) is not available in the general case where A admits complex eigenvalues. However, it can still be shown, in a way similar to that used in [12], that $\|(I-\Pi_m)u_i\|$ decreases rapidly to zero as m increases when u_i is one of the eigenvectors associated with the extreme eigenvalues, thus showing that we have essentially the same phenomenon as above, namely that the components of the residual corresponding to the outermost part of the spectrum will be damped out more rapidly than the others.

In the particular case where all the eigenvalues of A are real and simple, a result similar to that of the symmetric case can still be shown. Suppose that $\lambda_1 > \lambda_2 > \cdots > \lambda_N$ are the eigenvalues of A and let $v_1 = \sum_{i=1}^{N} \alpha_i u_i$. Then we can formulate the following.

PROPOSITION 2. *Suppose that $\alpha_1 \neq 0$ and let $\beta_i = \sum_{j=i+1}^{N} |\alpha_j/\alpha_i|$. Let γ_i and K_i be defined as in Proposition* 1. *Then the residual vector r_m satisfies*

$$|(r_m, u_i)| \leqq \|r_m\|\frac{K_i}{T_{m-i}(\gamma_i)}\beta_i. \tag{3.4}$$

Proof. Inequality (3.4) is a consequence of Lemma 1 and the following inequality, which can be derived by arguments very similar to those used in [12],

$$\|(I-\Pi_m)u_i\| \leqq \frac{K_i}{T_{m-i}(\gamma_i)}\beta_i. \qquad \text{Q.E.D.}$$

The above inequality does not show, however, that the component of the residual or of the error is small. Suppose $u_1, u_2, \cdots, u_l$ are eigenvectors corresponding to the largest eigenvalues and let $\sum_{i=1}^{l} \gamma_i u_i$ be the orthogonal projection of r_m onto the invariant subspace span $[u_1, u_2, \cdots, u_l]$, that is, the γ_i's, $i = 1, 2, \cdots, m$, minimize the norm

$$\left\| r_m - \sum_{i=1}^{l} \gamma_i u_i \right\|.$$

If we set $U_l = [u_1, u_2, \cdots, u_l]$, $g = (\gamma_1, \gamma_2, \cdots, \gamma_l)^H$ then we have $g = (U_l^H U_l)^{-1} U_l^H r_m$ and hence

$$\|g\| \leqq \tau \|U_l^H r_m\|, \tag{3.5}$$

where τ is the norm of the inverse of the Gramian of the system U_l. Therefore, the components γ_i, $i = 1, l$ of the residual in the subspace span $[U_l]$ will be small because the inner products $U_i^H r_m$ are small in general, as inequality (3.4) shows. This obviously depends on the norm of $[U_l^H U_l]^{-1}$ which is also equal to the inverse of the smallest singular value of the system U_l.

The above arguments concern the case of full orthogonalization (Arnoldi) only. For IOM a result similar to Lemma 1 is needed. Consider the subspace $L_m =$ span $[w_1, w_2, \cdots, w_m]$ where

$$w_i = v_i - (v_i, v_{m+1}) v_{m+1}$$

and denote by Q_m the orthogonal projector onto L_m. Then, as was shown in Saad [13], the residual vector r_m is orthogonal to L_m and we have

$$(r_m, u_i) = (r_m, Q_m u_i) + (r_m, (I - Q_m) u_i) = (r_m, (I - Q_m) u_i),$$

which shows the following analogue of Lemma 1:

$$|(r_m, u_i)| \leqq \|r_m\| \, \|(I - Q_m) u_i\|.$$

Although it is difficult to obtain any quantitative results for the distance $\|(I - Q_m) u_i\|$ between u_i and the subspace K_m, in general K_m and L_m are not too different and $\|(I - Q_m) u_i\|$ is again a small quantity.

3.2. The problem of stability in the presence of an approximate inverse of the Jacobian.

3.2.1. The quasi-Newton corrector. A P(EC) predictor-quasi-Newton-corrector method can be written as follows

P: $$y_{n,0} = \sum_{i=1}^{k} \bar{\alpha}_i y_{n-1} + \bar{\beta}_i d_{n-i}.$$

C: For $j = 1, 2, \cdots, M-1$ do

$$\begin{aligned} y_{n,j+1} &= y_{n,j} + B^{-1}\left[h\beta_0 f(y_{n,j}) + \sum_{i=1}^{k} (\alpha_i y_{n-i} + \beta_i d_{n-i}) - y_{n,j} \right] \\ y_n &= y_{n,M} \qquad d_n = \frac{1}{\beta_0}\left[y_n - \sum_{i=1}^{k} (\alpha_i y_{n-i} + \beta_i d_{n-i}) \right], \end{aligned} \tag{3.6}$$

where B^{-1} is an approximation to the inverse of $(I - h\beta_0 J)$ and J is the Jacobian of f relative to the variable y. (Note that d_n is the approximation to hy'_n that satisfies the corrector equation. It is *not* $hf(y_n)$ unless the corrector has been solved exactly by the iteration. This is important for stability.)

We shall simplify the notation by setting

$$\Sigma_c = \sum_{i=1}^{k} (\alpha_i y_{n-i} + \beta_i d_{n-i}). \tag{3.7}$$

The corrector iteration (3.6) corresponds to M steps of the Newton method for solving the equation

$$y_n = h\beta_0 f(y_n) + \Sigma_c \tag{3.8}$$

starting with the predicted value $y_{n,0}$ and using the approximation B^{-1} for the inverse of $I - \beta_0 hJ$.

3.2.2. Convergence of the corrector iteration. The first question that arises is under what conditions on B would the corrector iteration converge if it were iterated for an infinite number of steps. Let us assume that (3.8) has an exact solution $y_{n,*}$.

Consider the error $y_{n,j+1} - y_{n,*}$

$$y_{n,j+1} - y_{n,*} = y_{n,j} - y_{n,*} + B^{-1}[h\beta_0 f(y_{n,j}) + \Sigma_c - y_{n,j}]. \tag{3.9}$$

$f(y_{n,j})$ can be expressed as

$$f(y_{n,j}) = f(y_{n,*}) + J_j(y_{n,j} - y_{n,*}), \tag{3.10}$$

where the (i, k) entry of the matrix J_j is the partial derivative of f_i relative to the kth variable $y_{(k)}$, evaluated at a point ξ between $(y_{n,*})_{(k)}$ and $(y_{n,j})_{(k)}$. Substituting (3.10) into (3.9) and using the fact that $y_{n,*} = h\beta_0 f(y_{n,*}) + \Sigma_c$ we get

$$y_{n,j+1} - y_{n,*} = y_{n,j} - y_{n,*} + B^{-1}[h\beta_0 J_j(y_{n,j} - y_{n,*}) + y_{n,*} - y_{n,j}]$$

or

$$y_{n,j+1} - y_{n,*} = [I - B^{-1}(I - h\beta_0 J_j)](y_{n,*} - y_{n,j}). \tag{3.11}$$

As a result of (3.11) we can derive sufficient conditions for convergence of the corrector iteration such as $\|I - B^{-1}(I - h\beta_0 J_j)\| \leqq \alpha < 1 \ \forall j$, for a certain norm. This means, however, infinitely many assumptions. If we make the simplifying assumption that the Jacobian is constant and nondefective, then the above condition becomes

$$\|I - B^{-1}(I - h\beta_0 J)\| < 1. \tag{3.12}$$

Furthermore, we have a choice for the norm used in (3.12). Since we would like to interpret the result (3.12) in terms of the eigenvetors of J, it seems natural to consider the following norm

$$\|A\|_1 = \max_{i=1,\cdots,N} \|Au_i\|_1,$$

where $\{u_i\}_{i=1,\cdots,N}$ is a basis of $\mathbb{C}^N$ consisting of eigenvectors of J such that $\|u_i\| = 1$ and where we define[1] $\|x\|_1 = \sum_{i=1}^{N} |\alpha_i|$ for $x = \sum_{i=1}^{N} \alpha_i u_i$. $\|A\|_1$ is the corresponding matrix norm. Equation (3.12) then reads

$$\max_{i=1,\cdots,N} \|(I - (1 - h\beta_0\lambda_i)B^{-1})u_i\|_1 < 1$$

or

$$\left\| \left[B^{-1} - \frac{1}{1 - h\beta_0\lambda_i} I \right] u_i \right\|_1 < |1 - h\beta_0\lambda_i|^{-1}, \qquad i = 1, \cdots, N. \tag{3.13}$$

[1] Note that this norm is just the classical norm $\|\cdot\|_1$ relative to the eigenbasis.

The left-hand side of (3.13) is just the *residual norm of the eigenpair* u_i, $(1-h\beta_0\lambda_i)^{-1}$ for B^{-1}.

The interpretation of (3.13) is as follows. Since B^{-1} is an approximation to $(I-h\beta_0 J)^{-1}$, we can consider that the eigenvectors u_i of $(I-h\beta_0 J)^{-1}$ associated with $(1-h\beta_0\lambda_i)^{-1}$ are approximate eigenvectors of B^{-1}. Inequality (3.13) tells us that a sufficient condition for convergence of the corrector is that the residuals of the eigenpairs u_i, $(1-h\beta_0\lambda_i)^{-1}$ for B^{-1} have a norm smaller than $|1-h\beta_0\lambda_i|^{-1}$. Such a condition will be more difficult to satisfy for the most negative λ_i for which $|1-h\beta_0\lambda_i|^{-1}$ is small, while it will not cause any difficulty for the small eigenvalues for which $|1-h\beta_0\lambda_i|^{-1}\simeq 1$. In other words, B^{-1} should be a good approximation to $(I-h\beta_0 J)^{-1}$ in the dominant space.

3.2.3. The error equation. We now assume that the sufficient condition (3.12) is fulfilled and we will analyze the stability of the method for a linear equation of the form $y'=Jy$. Since there are difficulties in estimating the global error $y(t_n)-y_{n,M}$, we proceed in a different way by estimating $\tilde{e}_n = y_{n,M}-y_n$ where y_n is the approximation at t_n that would be obtained if the corrector were solved exactly *at each* integration step.

Since we assume that the underlying integration method is stable and accurate, it is sufficient to estimate the additional error introduced by the iterative scheme.

We will distinguish between the two sequences by denoting by $\tilde{y}_n$ the sequence obtained as described in § 3.2.1 when an approximate inverse, B^{-1}, of $A=(I-h\beta_0 J)$ is used with M iterations, that is, $\tilde{y}_n = y_{n,M}$. These two sequences satisfy

$$y_n = h\beta_0 J y_n + \Sigma_c, \tag{3.14}$$

$$\tilde{y}_n = h\beta_0 J\tilde{y}_n + \tilde{\Sigma}_c - r_n, \tag{3.15}$$

where

$$\Sigma_c = \sum_{j=1}^{k}(\alpha_j y_{n-j}+\beta_j h y'_{n-j}), \qquad \tilde{\Sigma}_c = \sum_{j=1}^{k}(\alpha_j \tilde{y}_{n-j}+\beta_j \tilde{d}_{n-j})$$

and r_n is a residual term which takes into account the fact that the corrector is not solved exactly.

Subtracting (3.14) from (3.15) we get

$$\tilde{e}_n = h\beta_0 J\tilde{e}_n + \sum_{j=1}^{k}(\alpha_j\tilde{e}_{n-j}+\beta_j\tilde{e}'_{n-i})-r_n, \tag{3.16}$$

where $\tilde{e}_n = \tilde{d}_n - hy'_n$.

Let $\beta_j = 0$, $j=1,\cdots,k$ (BDF formulas); then

$$\tilde{e}_n = A^{-1}\sum_{j=1}^{k}\alpha_j\tilde{e}_{n-j} - A^{-1}r_n. \tag{3.17}$$

If we expand $\tilde{e}_n$ and r_n in the eigenbasis $\{u_i\}_{i=1,N}$ as

$$\tilde{e}_n = \sum_{i=1}^{N}\delta_n^{(i)}u_i, \qquad r_n = \sum_{i=1}^{N}\varepsilon_n^{(i)}u_i,$$

we obtain for each component the equation

$$\delta_n^{(i)} = \left(\frac{1}{1-\beta_0 h\lambda_i}\right)\sum_{j=1}^{k}\alpha_j\delta_{n-j}^{(i)} - \frac{\varepsilon_n^{(i)}}{1-\beta_0 h\lambda_i}. \tag{3.18}$$

The homogeneous part of (3.18) is the difference formula that arises from applying the k-step BDF method to $\delta'^{(i)} = \lambda_i \delta^{(i)}$ (with stepsize h). We assume that the stepsize is such that this is either an absolutely stable process, or an accurate representation of a slowly growing solution in the case of positive $\lambda_i h$. If the process is absolutely stable, there exists a constant K_i such that the solution of (3.18) can be bounded by

$$|\delta_n^{(i)}| \leqq K_i \max_{j=k,n} \left| \frac{\varepsilon_j^{(u)}}{1-\beta_0 h\lambda_i} \right|. \tag{3.19}$$

If the solution is slowly growing, a bound of the form

$$|\delta_n^{(i)}| \leqq K_i e^{L_i t_n} \max_{j=k,n} \frac{1}{h} \left| \frac{\varepsilon_j^{(i)}}{1-\beta_0 h\lambda_i} \right| \tag{3.20}$$

can be obtained by standard techniques.

The error terms $\varepsilon_n^{(i)}$ are nothing but the components in the eigendirections of the residual in solving the corrector. In the case of small $h\lambda_i$, the residuals of the predictor are of order $O(h^{p+1})$ for a method of order p, so it is necessary only that they not be amplified by the corrector process. In the case of large $h\lambda_i$ the results of § 3.2.2. show that $\varepsilon_n^{(i)}$ can be made as small as desired provided that B^{-1} is a fairly good representation of $(I-h\beta_0 J)^{-1}$ in the dominant subspace. Hence we conclude that if B has the properties described in § 3.2.2, the corrector can be iterated until the residual is suitably small and the effect is that the additional errors due to the iteration will be bounded when the underlying integration technique is stable.

3.2.4. Practical considerations. There are a number of difficult-to-analyze techniques that can be used to get the greatest advantage from iterative methods. Some of these will be discussed in this section.

Reducing h. As the stepsize is reduced, fewer eigenvalues are part of the dominant subspace, so the dimension m needed for V_m can be reduced. In some cases it will be worth restricting h to be smaller than that dictated by integration accuracy in order to reduce m and reduce the work per step. For example, if halving h reduces m by more than a factor of two, the integrator will probably be faster with the smaller h.

Recomputing V_m. If the problem has a few isolated large eigenvalues, then it is likely that V_m can be kept constant over several steps. However, if there are several clusters of large eigenvalues, it is probably more efficient to recompute V_m frequently. This is because if we have a small number, μ, of isolated large eigenvalues, and m is of order μ (e.g., $m=\mu$ or $m=\mu+1$), the projection process will have the effect of correcting the errors anywhere in the dominant subspace. The process then can be regarded as a method of correction in the dominant space similar to the one described in [1]. If on the other hand there are several clusters, reasonable accuracy can be obtained with m equal to the number of clusters, if V_m is recomputed anew for each step starting with the corresponding residual and if all eigenvalues are simple. In addition to the effect of damping the components of the error in some of the large eigenvectors, recomputing V_m frequently will have the effect of solving the linear systems to greater accuracy. Finally, it is worth mentioning that in practice the need for recomputing a new V_m can be detected in a way similar to the way that the need for a new Jacobian is detected in the codes which implicitly use the Jacobian matrix—that is, by examining the convergence (or otherwise) of the iterative steps in the Newton solution of the corrector.

Representing the dominant space. We have seen that B should be a good representation of $A=(I-h\beta_0 J)$ in the dominant subspace. Because B is based on the Krylov subspace $\{A^q r_0, q=0, 1, \cdots, m-1\}$ it is a good representation for the extreme eigenvalues *including* the small ones. If, instead, we use the space $\{A^q r_0, q = s, s+1, \cdots, s+m-1\}$, we will emphasize the large eigenvalues. This effect can be increased by using J instead of A since the unimportant eigenvalues of J are those near zero. The use of $A^q r_0$ as the starting vector for the Krylov space reduces the significance of the component corresponding to the smallest eigenvalue λ_N by $|(1-h\beta\lambda_N)/(1-h\beta_0\lambda_1)|^q$ compared to the component corresponding to the largest. For small $|h\beta_0\lambda_N|$ and large $|h\beta_0\lambda_1|$ this is approximately $|h\beta_0\lambda_1|^{-q}$. Use of $J^q r_0$ results in a reduction of $|\lambda_N/\lambda_1|^q$. The ratio of these is $|h\beta_0\lambda_N|^q$, and for the nonstiff eigenvalue λ_N, $|h\beta_0\lambda_N|$ is less than one.

If we use $J^q r_0$ as a starting vector, the initial approximation x_0 must be chosen carefully. In a typical code we first approximate y_n *with an explicit predictor* p_n, and then solve (3.8) for the correction by

$$\tilde{y}_n = p_n + x,$$

where

$$Ax = \Sigma_c + h\beta_0 f(p_n) - p_n = r_0.$$

If we solve this *only* in the dominant space, then the component of $\tilde{y}_n$ in the subdominant space is simply the predicted value. Therefore, we should set x_0 to be r_0, which is equivalent to treating the nonstiff components with a simple PEC method.

Starting with $J^q r_0$ suffers from two serious drawbacks: it is expensive for $q>0$ and we must compute $[V_m^T V_m]^{-1} V_m^T r_0$ and cannot use the approximations suggested in § 2.2. However, we will see that one outcome of another modification suggested below is to be able to use $q=1$ without additional difficulty.

If a single method is used for all components, it has to be a method for stiff equations such as BDF. This is not the best choice for the nonstiff components, so it is natural to ask if it is possible to use a nonstiff method such as Adams for the nonstiff components. The answer is shown to be affirmative below. Suppose that method (A) (e.g. Adams) is a nonstiff method and method (B) (e.g. BDF) is a stiff method. Superscripts will be used to distinguish the methods. We get a PEC method (A) by executing

$$\tilde{y}_n^{(A)} = p_n + x_1$$

with

$$x_1 = r_0 = \Sigma_c^{(A)} + h\beta_0^{(A)} f(p_n) - p_n. \tag{3.21}$$

Now we consider solving (3.8) for method (B) setting

$$\begin{aligned} &\tilde{y}_n = \tilde{y}_n^{(A)} + x_2, \\ &(I - h\beta_0^{(B)} J)x_2 = A^{(B)} x_2 = r_1 = \Sigma_c^{(B)} + h\beta_0^{(B)} f(\tilde{y}_n^{(A)}) - \tilde{y}_n^{(A)}. \end{aligned} \tag{3.22}$$

If this is solved only in the dominant subspace, this causes components in that subspace to be solved by method (B). We can prove the following:

PROPOSITION 3. *Suppose that J has a complete set of eigenvectors and that the left and right invariant subspaces corresponding to the eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_m$ of J are spanned by the rows of U^T and columns of V, respectively, where U and V are $N \times m$ matrices. Then, if the operator $(A^{(B)})^{-1}$ involved in solving* (3.22) *is represented by*

$V[U^TA^{(B)}V]^{-1}U^T$, the solution to $y'=Jy$ by (3.21) *and* (3.22) *is that obtained by treating the components corresponding to $\lambda_1, \cdots, \lambda_m$ by method* (*B*) *and the remainder by method* (*A*).

Proof. The proof follows by diagonalizing the system. Q.E.D.

The significance of this result is that if V_m and V_m^T approximate V and U^T, we can expect to approximate this behavior which is desirable if λ_1 to λ_m are the only dominant eigenvalues. (On the surface it appears that the cost is higher because it is necessary to save d_{n-i} values for Adams and y_{n-i} values for BDF formulas. There are ways to avoid this. See, for example, Gear [7].)

When (3.22) is solved iteratively, we should build the Krylov space $K_m = \text{span}[r_1, Jr_1, \cdots, J^{m-1}r]$ and a basis V_m to solve $Ax_2 = r_1$. Let us examine r_1 in terms of r_0. Combine (3.21) and (3.22) and assume $f(y) = Jy$ to get

$$\begin{aligned} r_1 &= \Sigma_c^{(B)} + h\beta_0^{(B)} f(p_n + r_0) - p_n - r_0 \\ &= \Sigma_c^{(B)} + h\beta_0^{(B)} f(p_n) + h\beta_0^{(B)} Jr_0 - p_n - \Sigma_c^{(A)} - h\beta_0^{(A)} f(p_n) + p_n \\ &= [\Sigma_c^{(B)} + h\beta_0^{(B)} f(p_n) - \Sigma_c^{(A)} - h\beta_0^{(A)} f(p_n)] + h\beta_0^{(B)} Jr_0. \end{aligned}$$

The first term is the difference between the solutions by a PEC method using methods (*B*) and (*A*), respectively. For the nonstiff component it is $O(h^{p+1})$ where p is the minimum of the orders of the two methods and one plus the order of the predictor. Hence, it is small for nonstiff components. Hence, r_1 has properties very similar to that of $h\beta_0^{(B)}Jr_0$: smaller in the nonstiff components by a factor proportional to $1/|\lambda_1|$. Thus, the Krylov subspace $\{J^q r_1, q = 0, \cdots, m-1\}$ has similar properties to $\{J^q r_0, q = 1, \cdots, m\}$.

4. Numerical experiments. The numerical experiments described below were performed on a Prime 650 (which has a 48 bit mantissa in double precision giving about 14 decimal digits) and a DEC 20. The first series of tests treated has been adapted from Krogh's example given in [3, p. 218].

First, n functions z_i, $i = 1, 2, \cdots, n$ are defined by

$$\frac{dz_i}{dt} = \beta_i z_i + z_i^2, \qquad i = 1, 2, \cdots, n, \tag{4.1}$$

where the β_i are negative constants. Then a vector y is set equal to

$$y = Uz, \tag{4.2}$$

where

$$U = I - \frac{2}{v^T u} uv^T. \tag{4.3}$$

Obviously y satisfies a differential equation of the form

$$y' = Ay + g(y), \tag{4.4}$$

where

$$A = U \text{ diag}(\beta_i) U, \qquad [g(y)]_i = \sum_j U_{ij} \Big(\sum_k U_{jk} y_k \Big)^2.$$

(Notice that $U^{-1} = U$.) The solution of each equation (4.1) is

$$z_i(t) = \frac{-\beta_i}{1 + c_i e^{-\beta_i t}}, \tag{4.5}$$

where c_i is adjusted according to the initial value $z_i(0)$. The exact solution of (4.4) is therefore immediately obtainable by use of (4.5) and (4.2).

Another interesting feature of this class of examples is that one can easily choose u and v such that the Jacobian is more or less close to a symmetric (or a skew symmetric) matrix. Also, the degree of nonlinearity can be controlled by adding a multiplicative constant γ before the term z_i^2 in (4.1):

$$\frac{dz_i}{dt} = \beta_i z_i + \gamma z_i^2.$$

(In that case the solution (4.5) must be replaced by: $-\beta_i/(\gamma + c_i e^{-\beta_i t})$.) The following expression of the Jacobian of (4.4)

$$J(t) = U[\text{diag}\,(\beta_i + 2z_i)]U \tag{4.6}$$

can be used for comparisons with methods using explicit Jacobians.

The dimension N has been taken relatively small so as to allow comparisons with other methods. (Note that the storage of the Jacobian increases as N^2.) The following distribution has been taken for the elements β_i: $\beta_1 = -1000$, $\beta_2 = -800$, $\beta_3 = -500$, $\beta_4 = -300$, $\beta_i = -100(N-i+1)/(N-5)$ for $i \geqq 5$. The vectors u and v have been chosen as follows: $u = (0, 1, 1, \cdots, 1, 1)^T$, $v = (1, 1, 1, \cdots, 1)^T$ and the initial values are such that $y^{(0)} = Uz^{(0)}$ with

$$z^{(0)} = (-1, -1, \cdots, -1)^T.$$

Three different methods have been run on this example. They all use BDF schemes with automatic determination of stepsize and order as implemented by Hindmarsh in the code LSODE. LSODE was modified to include the IOM method and projection process described in § 2 as an additional option. The first method uses (full) Gaussian elimination to solve the linear systems and the Jacobian is directly provided by the user. The corresponding method flag in LSODE is MF = 21. The second method, as the first, uses the Gaussian algorithm but the Jacobian is internally generated rather than provided by the user, the method flag is MF = 22. The third method is a combination of the IOM method and the projection process. The method flag is MF = 26 for this new method. For the last case, the parameter p (the number of vectors against which the v_i is orthogonalized) has been tested with two values $p = 4$ and $p = 9$, while $m_{\max}$ (the maximum number of vectors v_i generated) is set to $3 \times (p+1)$. The value of the parameter l used for approximating $V_m^T r_0$, is, as suggested at the end of § 2.2, defined by $l = \max(5, p+1)$. Table 1 compares, for $N = 50$, the results obtained when (4.4) is integrated from $t_0 = 0$ to $t = 1$ with an integration tolerance EPS of 10^{-6}.

TABLE 1

	MF = 21	MF = 22	MF = 26 $p = 4$	MF = 26 $p = 9$
Total number of function evaluations	182	1382	720	643
Total number of steps	151	151	184	180
Core memory required	2972	2972	1035	1750
Run times (minutes on PRIME 650)	1'08	1'09	1'19	1'05

The run times are not very different from those of the first two methods (MF = 21 and MF = 22) because the dimension is relatively small. When $N = 80$, the results shown in Table 2 are obtained.

TABLE 2

	MF = 21	MF = 22	MF = 26 $p = 4$	MF = 26 $p = 9$
Total number of function evaluations	175	1935	733	640
Total number of steps	149	149	186	172
Core memory required	7142	7142	1545	2410
Run times (minutes on PRIME 650)	2'57	3'03	1'55	1'38

Another class of examples for test purposes is provided by the heat equation on a domain Ω of $\mathbb{R}^2$.

$$\Delta u = \frac{\partial u}{\partial t} \quad \forall x \in \Omega, \quad \forall t \in [t_0, t_1],$$

$$u(t, x) = 0 \quad \forall x \in \partial\Omega, \quad \forall t \in [t_0, t_1],$$

$$u(t_0, x) = u_0, \quad x \in \Omega.$$

Suppose that Ω is the square $(0, 1) \times (0, 1)$ and that the five point discretization is employed for the x variable. Then the above problem yields the following linear system of ODEs:

$$y'(t) = -Ay(t), \qquad t \in [t_0, t_1],$$

$$y_i(t_0) = u(t_0, x_i), \qquad i = 1, 2, \cdots, N,$$

where $x_1, x_2, \cdots, x_N$ are the N interior discretization points of Ω with $N = n^2$ if each side of the square is discretized into n interior points. Here A is the Laplacian block tridiagonal matrix

$$A = \begin{vmatrix} B & -I & & \\ -I & B & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & B \end{vmatrix},$$

where B is the tridiagonal matrix with diagonal elements +4 and off-diagonal elements −1.

Since the matrix A is symmetric, the parameter p can be taken equal to 2 in which case IOM is equivalent to the conjugate gradient method. Another important simplification here is that the Jacobian is A itself because of linearity, so we do not need to use the numerical differentiation formula.

The following experiment was performed on a DEC 20 computer using double precision with a machine unit roundoff $\cong 10^{-19}$. The initial distribution of temperature (u_0) was taken to be uniformly equal to one in the interior. When $n = 10$, $t_0 = 0$, $t_1 = 1.0$ we obtain the following results:

total number of steps: 183;

total number of function evaluations: 724;

total number of reevaluations of V_m and H_m: 98;

work space required: 1560;

CPU time on the DEC 20: 52.31 sec.

Note that the function evaluations include the operations of the form Jx since the Jacobian and A are identical.

LSODE can treat this problem more efficiently by using the LINPACK band solvers because A is of banded form with half bandwidth 10. The corresponding method flag is MF = 25 and the results are as follows:

total number of steps: 131;
total number of function evaluations: 530;
total number of re-evaluations of the Jacobian: 18;
work space required: 4022;
CPU time on the DEC 20: 21.48 sec.

Note that although the direct solver (MF = 25) is to be preferred in this simple case, large work space use is in favor of the iterative method. Space usage can become prohibitively large for finer discretization if MF = 25 is used.

5. Practical considerations. In the light of observations made during various numerical experiments we are in a position to make a few remarks. First we wish to give a few more practical indications on the choice of some of the parameters which are important for the performance of the IOM. It is possible to determine dynamically the parameter p by observing the elements of the Hessenberg matrix generated. The algorithm is first started with some initial p which is then reduced to p_{new} if it is observed that the elements outside the p_{new}th diagonal of H_m become small. More details on this may be found in Saad [14]. There is no upper limit restriction on the parameter m except for the physical limitation due to the storage. As explained earlier, the actual m used at a given step is determined by convergence criteria and does not exceed some fixed value $m_{\max}$. In most of our tests $m_{\max}$ was taken to be 20 or 30.

There are two important factors for the performance of the algorithm which are the accuracy, TOL, in the solutions of the linear systems and the parameter ε in the difference formula used to approximate Jx. For the solutions of the systems, various other papers and our analysis in § 3 shows that we should not require a high accuracy. In fact our experience is that satisfactory results are obtained by stopping the process as soon as the residual norm is reduced by a factor of order 0.01. Selecting an optimal parameter ε in the difference formula for approximating Jx is a more difficult problem. If ε is too small then the rounding errors made in the numerator are amplified by a factor of order $1/\varepsilon$ which leads to an inaccurate result. If on the other hand ε is too large then the approximation of Jx will be poor. Any reasonable choice of ε should attempt to reach a compromise between these two difficulties.

We now wish to discuss briefly the use of other possible iterative methods. Among the simplest iterative methods that might be used in stiff ODE methods are the relaxation methods. These require the Jacobian explicitly and so they do not offer the same generality of usage as those methods in which only operations of the form $x \to Jx$ are needed. A case in which the SOR method can be applied is the use of the method of lines for solving parabolic equations. In these cases the Jacobians are simple to compute and have a convenient structure which makes any iterative method easy to apply. For more general problems in which the Jacobian does not have any particular structure and/or is not easily available, the relaxation methods are no longer of interest. Relaxation methods also have the disadvantage of requiring an optimal parameter ω which is often difficult to estimate. One might argue, however, that when the system varies slowly it is not unreasonable to expect the optimal parameter to have a similar smooth behavior which would facilitate the computation of ω_{opt}. But when we deal with stiff systems we may have large variations in the matrices A of the system due to common large stepsizes, which means that the parameter actually has to be estimated anew for every step after the transient phase. Finally we should

point out that the rate of convergence of SOR is known to be sensitive to the accuracy of the optimal parameter.

Another class of methods uses the transpose of A as well as A by solving implicitly or explicitly the normal equations $A^TAx = A^Tb$. There are several arguments against this approach, the most commonly given in the literature being that the normal equations increase the conditioning of the original equations. In our case it is also not desirable to work with the transpose of A which may not be available.

There are a few other methods, among them ORTHOMIN(k) (Eisenstat et al. [2] and Vinsome [16]), which are similar to ours in that the solution belongs to the Krylov subspace, but in which the solution is produced directly in a conjugate gradient-like algorithm instead of using secondary storage. It has been observed in some experiments discussed in [11] that the two classes of methods behave in a similar way in terms of amount of work. A version using secondary storage can also be derived from the ORTHOMIN (k) algorithms although the storage would then be double that required by IOM (k). The unsymmetric Lanczos algorithm, which also belongs to the class of Krylov subspace methods is an interesting alternative but again requires the use of the transpose of A.

An important aspect of the iterative methods which we have not considered so far is the possibility of preconditioning the original system, i.e., of modifying the system into an equivalent one which is easier to solve. There are several ways of accomplishing this but most methods amount to approximating the inverse of A by some matrix M and then solving the system $MAx = Mb$ instead of $Ax = b$. The matrix M may be obtained for example from an approximate LU factorization of A. The advantage of using preconditionings is the faster rate of convergence that may result since MA will hopefully be closer to the identity than A. However there seems to be more difficulties than there are advantages when dealing with the solution of stiff ODEs. The main difficulty is that we need a technique having a wide range of application because the systems encountered during the ODE integration may pass from systems having a positive definite symmetric part $(A+A^T)/2$ to systems having an indefinite symmetric part. There does not seem to exist any such "universal" preconditioning technique so far. An exception to this may be found when using the method of lines for parabolic equations because then the Jacobian is known to have a structure which lends itself naturally to preconditioning. Preconditioning may then be helpful for the later steps of integration where h is large and the symmetric part of the matrix $(I-hJ)$ becomes close to a definite negative matrix (assuming J positive definite).

A second drawback with preconditionings is that we no longer obtain a solution which is more accurate in the dominant subspace since the solution will now be accurate in the dominant subspace of $M(I-hJ)$. Thus the advantage gained from a faster convergence might well be annihilated by this new undesirable feature. More experience is needed however before reaching any definitive conclusion.

6. Conclusions. We believe that there are two essential features which make the Krylov subspace methods attractive. First there is no need for storing the Jacobian in any form and this is important for the generality of the integration codes. In most of the available codes different system solvers are called according to the matrix structure and the method of storage. In methods using Krylov subspaces, on the other hand, this is avoided thanks to the differentiation formula.

The second important feature of Krylov subspace methods is that we do not need high accuracy when solving the systems because even when the solution is not accurate,

the residual is still likely to be small in the dominant space. This is all that is needed for stability, as our analysis in § 3 indicates.

REFERENCES

[1] P. ALFELD AND J. D. LAMBERT, *Corrections in the dominant space: a numerical technique for a certain class of stiff initial value problems*, Math. Comp. 31 (1977), pp. 922–938.

[2] S. C. EISENSTAT, H. C. ELMAN AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, Tech. Rept. 209, Yale University, New Haven, CT 1980.

[3] W. H. ENRIGHT, *Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations*, ACM Trans. Math. Software, 4 (3) (1981), pp. 127–136.

[4] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[5] ———, *Future developments in stiff integration techniques: stability methods that do not use an exact Jacobian*, UIUCDCS-R-76-839, Dept. Computer Sci., Univ. Illinois, Urbana, 1976.

[6] C. W. GEAR AND L. F. SHAMPINE, *A user's view of solving stiff ordinary differential equations*, SIAM Review, 21 (1974), pp. 1–17.

[7] C. W. GEAR, *Unified modified divided difference implementation of Adams and* BDF *formulas*, Dept. Rept. UIUCDCS-R-80-1014, Univ. Illinois, Urbana, 1980.

[8] K. C. JEA AND D. M. YOUNG, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Alg. Appl., 34 (1980), pp. 159–194.

[9] R. W. KLOPFENSTEIN, *Numerical differentiation formulas for stiff systems of ordinary differential equations*, RCA Rev., 32 (1971), pp. 447–462.

[10] W. L. MIRANKER AND I. L. CHERN, *Dichotomy and conjugate gradients in the stiff initial value problem*, IBM Research Rept. 8032-34917, January 1980.

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

[12] ———, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Alg. Appl, 34 (1980), pp. 269–295.

[13] ———, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal, 19 (3) (1982), pp. 485–506.

[14] ———, *Practical use of Krylov subspace methods for solving indefinite and unsymmetric linear systems*, Tech. Rept. #214, Dept. Computer Science, Yale University, New Haven, CT, 1982.

[15] L. F. SHAMPINE, *Implementation of implicit formulas for the solution of* ODEs, this Journal, 1 (1980), pp. 103–118.

[16] P. K. W. VINSOME, ORTHOMIN, *an iterative method for solving sparse sets of simultaneous linear equations*, In Society of Petroleum Engineers of AIME, Proc. Fourth Symposium on Reservoir Simulation, 1976, pp. 149–159.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0003 $01.25/0

A SET OF COUNTER-EXAMPLES TO THREE CONDITION NUMBER ESTIMATORS*

A. K. CLINE† AND R. K. REW‡

Abstract. The estimation of the condition number of a matrix relative to inversion is important for bounding the errors in computer solutions of linear systems of equations. Three methods of estimation are shown to underestimate the true condition number of particular matrices by arbitrarily large factors. One of the methods is that employed in the LINPACK software package.

Key words. condition number, LINPACK

1. Introduction. In Cline, Moler, Stewart and Wilkinson [1], three algorithms were proposed for estimating the l_1 norm condition number of a matrix. Although none of the algorithms was proved to produce estimates with small errors, matrices for which previous estimation algorithms produced large underestimates were handled well by the second of these estimators. Furthermore, in thousands of tests with random matrices of various distributions and dimensions, rarely has the estimate produced by the second algorithm been less than $\frac{1}{10}$ of the true condition number. On the basis of this evidence, the second estimation algorithm was incorporated into LINPACK [3], a popular package of software for solving systems of linear equations.

It will be shown here that all three algorithms can produce arbitrarily large underestimates for certain matrices. Two new matrices and two previously mentioned in [1] will be examined. For three of these matrices, it will be shown that not only is the condition number severely underestimated for the particular matrix, but that in fact entire neighborhoods of matrices exist for which underestimates are made.

Finally, some remarks are made upon the larger question of selection of condition number estimators. In particular, the role of counter-examples is mentioned.

2. The estimators. The value of the condition number of a matrix with respect to inversion is described in a large selection of numerical analysis texts. The quantity, which may be defined as

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

(for a fixed matrix norm $\|\cdot\|$), is employed in several inequalities relating the error in the solution of a linear system to perturbations in the matrix or in the right-hand side. Two such inequalities are:

1. if $Ax = b$ and $A(x + \Delta x) = b + \Delta b$ then

$$\frac{\|\Delta x\|}{\|x\|} \leqq \kappa(A) \frac{\|\Delta b\|}{\|b\|},$$

2. if $Ax = b$ and $(A + \Delta A)(x + \Delta x) = b$, then

$$\frac{\|\Delta x\|}{\|x\|} \leqq \frac{\kappa(A)\|\Delta A\|/\|A\|}{1 - \kappa(A) \cdot \|\Delta A\|/\|A\|},$$

provided $\|\Delta A\| < 1/\|A^{-1}\|$.

* Received by the editors March 30, 1981, and in final revised form August 16, 1982.

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

‡ National Center for Atmospheric Research, Boulder, Colorado 80307. The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research and is sponsored by the National Science Foundation.

The vector norm in these inequalities may be any vector norm consistent with the matrix norm.

Given a matrix norm such as the l_1 norm (which equals the maximum of the sums of absolute values of components in matrix columns) or the l_∞ norm (which equals the maximum of the sums of absolute values of components in matrix rows), then $\|A\|$ may be computed with a moderate number of arithmetic operations (in either case about n^2 additions and n^2 absolute value evaluations for an $n \times n$ matrix A) compared to a factorization of the matrix (about $n^3/3$ multiplications and $n^3/3$ additions). Unfortunately, if $\|A^{-1}\|$ is to be obtained by first computing A^{-1} and then taking its norm, about n^3 multiplications and n^3 additions are required. Since the solution to a linear system $Ax = b$ can be obtained in about n^2 multiplications and n^2 additions after a factorization has been determined, we see that perhaps three times as much computational effort is required to determine $\|A^{-1}\|$ (and hence $\kappa(A)$) as is required to solve a linear system. We would like to avoid this extra computational effort. This can be done if there exists an inexpensive algorithm for estimating $\|A^{-1}\|$ (and hence estimating $\kappa(A)$), instead of computing it exactly. An algorithm requiring only $O(n^2)$ operations is desirable.

Several approaches to characterizing the condition number of a matrix without evaluating or estimating $\|A^{+1}\|$ are simply folklore and need to be discussed here for no other reason than to show that they are worthless and thus, we hope, to give them much deserved burials. The first is that a moderately sized determinant of a matrix somehow indicates moderate condition. The second is that the ratio of largest to smallest eigenvalues in magnitude is an indicator of condition. The third is that if a matrix is poorly conditioned, a small element will appear on the diagonal of the upper triangular factor during Gaussian elimination with partial pivoting.

The single $n \times n$ matrix

$$\begin{bmatrix} 1 & & & & \\ -1 & 1 & & 0 & \\ -1 & -1 & 1 & & \\ \vdots & \vdots & \ddots & 1 & \\ -1 & -1 & \cdots & -1 & 1 \end{bmatrix}$$

shows the fallacy in all of these notions. This matrix has a determinant of 1, a ratio of largest to smallest eigenvalue of 1, and a smallest diagonal element in its upper triangular factor of 1. None of these indicators would suggest ill-conditioning, and yet the $n \times n$ matrix has an l_1 and l_∞ condition number of $n2^{n-1}$. Furthermore, equality is possible in the two conditioning inequalities, and linear systems can be constructed for which slight perturbations could cause arbitrarily poor solutions. The presence of the poor solutions is invisible to all three of these indicators.

It is helpful to briefly examine the actual relationships between these indicators and conditioning. The determinant of a singular matrix is, of course, zero, but determinants of near-singular matrices can be large, and determinants of well conditioned matrices can be small. Eigenvalues which differ greatly in magnitude do indicate ill-conditioning. In fact, the ratio of largest to smallest eigenvalue does provide a lower bound on the condition number of a matrix. Furthermore, for symmetric matrices, this ratio is exactly the l_2 condition number. However, for nonsymmetric matrices this ratio can severely underestimate a condition number. The diagonal elements in the upper triangular factor also can be used to underestimate the condition number. In fact, of these three approaches, this one can occasionally be useful:

empirical evidence suggests that there is a large class of matrices whose conditioning is predicted by examination of the diagonal of the upper triangular factor. As with the eigenvalues, though, the underestimate can be severe.

Several approaches which are more sound theoretically as well as empirically estimate $\|A^{-1}\|$ by using the fact that

$$\|A^{-1}\| = \max_{x \neq 0} \frac{\|A^{-1}x\|}{\|x\|}$$

(where the matrix norm is now assumed to be subordinate to the vector norm), and attempt to find vectors x in which the ratio $\|A^{-1}x\|/\|x\|$ is close to maximal. The three estimators of $\|A^{-1}\|$ in [1] are such approaches. They attempt to find a vector x which maximizes $\|A^{-1}\|/\|x\|$, where the l_1 norm is used. Each begins by constructing a certain vector b and then letting $x = A^{-T}b$. The first estimator from [1] is also described in Forsythe, Malcolm, and Moler [4]. The subroutine DECOMP found there intends to implement the algorithm, but contains a typographical error.

Let us assume that A has been factored. (Either of the forms $PA = LU$ or $LPA = U$ is adequate if P is a permutation matrix, L is unit lower triangular, and U is upper triangular.) We begin by solving $U^T z = b$ for z, where the elements of b are chosen during the solution steps in such a way that $x = A^{-T}b$ will be (we hope) a near-maximizer of $\|A^{-1}x\|/\|x\|$. The components of b are ± 1. Notice that at the intermediate stage where $b_1, \cdots, b_{s-1}$ have been specified, and thus $z_1, \cdots, z_{s-1}$ have been determined, we have

$$u_{1,s}z_1 + \cdots + u_{s-1,s}z_{s-1} + u_{s,s}z_s = b_s.$$

This yields two possibilities for z_s, depending upon the choice of b_s:

$$z_s^+ = \left(1 - \sum_{i=1}^{s-1} u_{i,s}z_i\right) \Big/ u_{ss}$$

or

$$z_s^- = \left(-1 - \sum_{i=1}^{s-1} u_{i,s}z_i\right) \Big/ u_{ss}.$$

In the first algorithm of [1], the sign of b_s is selected simply to maximize z_s. Thus:

ALGORITHM 1. Choose $b_s = +1$ if $|z_s^+| \geqq |z_s^-|$, and $b_s = -1$, otherwise.

The second and third algorithms attempt to consider more the effect of the selection of the sign of b_s, not only on the size of z_s, but on the ensuing components of z as well. In fact, for components $j = s, \cdots, n$, we have

$$(u_{1,j}z_1 + \cdots + u_{s-1,j}z_{s-1}) + u_{s,j}z_s + \sum_{i=s+1}^{j} u_{i,j}z_i = b_j.$$

The parenthesized expression above is denoted by $p_j^{(s-1)}$. Notice that it is fixed since $z_1, \cdots, z_{s-1}$ are. In an attempt to achieve a greater growth in the components of z, the second algorithm considers the sum of $|u_{s,s}z_s|$ and all the expressions $|p_j^{(s-1)} + u_{s,j}z_j|$ for $j = s+1, \cdots, n$. Thus:

ALGORITHM 2. Choose $b_s = +1$ if

$$|u_{s,s}z_s^+| + \sum_{j=s+1}^{n} |p_j^{(s-1)} + u_{s,j}z_s^+| \geqq |u_{s,s}z_s^-| + \sum_{j=s+1}^{n} |p_j^{(s-1)} + u_{s,j}z_s^-|,$$

and $b_s = -1$, otherwise.

The third algorithm modifies this only slightly to reflect the fact that when $p_j^{(s-1)} + u_{s,j} z_s$ is employed in the calculation of z_j, the expression is divided by $u_{j,j}$. Thus:

ALGORITHM 3. Choose $b_s = +1$ if

$$|z_s^+| + \sum_{j=s+1}^{n} |p_j^{(s-1)} + u_{s,j} z_s^+| / |u_{j,j}| \geqq |z_s^-| + \sum_{j=s+1}^{n} |p_j^{(s-1)} + u_{s,j} z_s^-| / |u_{j,j}|,$$

and $b_s = -1$, otherwise.

All algorithms begin with $b_1 = +1$. If the vectors b (and hence the vectors z) differ in the three algorithms because of a different choice of sign at some step, then different estimates of $\|A^{-1}\|$ may result. The final steps of estimation are identical, however. For all three, having obtained b, solve $A^T x = b$ for x (this actually only requires the use of z with the matrixes P and L), and finally solve $Ay = x$ for y. The estimate of $\|A^{-1}\|$ is then $\|y\|/\|x\|$, and $\kappa(A)$ is estimated by multiplying the estimated $\|A^{-1}\|$ by the computed $\|A\|$. The vector norm used in these calculations is the l_1 norm and the matrix norm is that subordinate to the l_1 vector norm. (Henceforth, these are denoted by $\|\cdot\|_1$.)

All of these approaches require $O(n^2)$ operations. The second algorithm has been used in LINPACK. As implemented there, (using a scratch array of length n), the second estimation requires about $2\frac{1}{2}n^2$ multiplications, $4\frac{1}{2}n^2$ additions, and $2n^2$ evaluations of absolute value. There may be an additional $\frac{1}{2}n^2$ multiplications and $\frac{1}{2}n^2$ additions required depending upon the $b_s = \pm 1$ decisions. Furthermore, in order to minimize the likelihood of computations overflowing, rescaling is done, involving as many as $4n^2$ additional multiplications. These figures represent the work in excess of the approximately $n^3/3$ multiplications and $n^3/3$ additions necessary to obtain the factorization. Although $O(n^2)$, this work may be as large as $14n^2$ total multiplications, additions, and absolute value evaluations, and hence comparable to the factorization effort for small matrices (i.e. $n \leqq 21$).

The description of the third algorithm in [1] is followed by the statement, "However, this modification increases the volume of computation appreciably." This statement should be clarified. As mentioned, the current LINPACK condition number estimator requires between $2\frac{1}{2}n^2$ and $7n^2$ multiplications, between $4\frac{1}{2}n^2$ and $5n^2$ additions, and $2n^2$ evaluations of absolute value. To implement the third algorithm (even allowing for rescaling) would require an additional $\frac{1}{2}n^2$ evaluations of absolute value and $\frac{1}{2}n^2$ divisions. Assuming all operations require the same computational time, the third algorithm requires between 7% and 11% additional time.

3. The counter-examples. Four counter-examples are presented. The first one was included in [1] and shows that the condition number may be seriously underestimated by the first algorithm. The second and third algorithms perform well on this matrix. Both the first and second algorithms perform poorly with the second counter-example; however, the third algorithm is satisfactory. A more severe difficulty with the second algorithm is displayed with the third counter-example. This matrix can be arbitrarily poorly conditioned, yet the second algorithm estimates a condition number of about 2.5. The first and third algorithms perform well on this matrix. Finally, the fourth counter-example shows that all three algorithms may make poor estimates on the same matrix.

As mentioned in [1], the matrix

$$A = \begin{bmatrix} 1 & 0 & k & -k \\ 0 & 1 & -k & k \\ 1 & -1 & 2k+1 & -2k \\ -1 & 1 & -2k & 2k+1 \end{bmatrix}$$

has condition number $24k^2+22k+3$. The first algorithm estimates only $6k+1$, and thus the ratio of the estimated condition number to the true value is $1/(4k+3)$, which can be arbitrarily small. The second and third algorithms both yield estimates of $20k^2+18k+O(1)$, which are quite acceptable. In summary:

Example A. For $k \geqq \frac{1}{3}$, the l_1 condition number of A is $24k^2+22k+3$; the estimated l_1 condition number (using the first algorithm of § 2) is $6k+1$. The ratio of estimated condition number to true condition number is $(6k+1)/(24k^2+22k+3) = 25/k+O(k^{-2})$.

The second counter-example is

$$B = \begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 1 & k+1 & -(k+1) \\ 0 & 0 & 0 & k \end{bmatrix}.$$

The construction of this matrix began with the determination of an upper triangular matrix U so that U^{-1} had large elements, yet the b and z selected by the first or second algorithms (which satisfy $U^T z = b$) would have z of moderate size. Using a U so that

$$U^{-T}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ k & -k & 1 & 0 \\ 2 & 0 & k^{-1} & k^{-1} \end{bmatrix}$$

with large values of k, we would have this property if $b_2 = 1$. (The large values of k and $-k$ would not affect z, since both b_1 and $b_2 = 1$.) The rest of the construction of a counter-example simply required choosing a unit lower triangular matrix L so that little increase would be made in the solution of $By = x$. For this purpose L was chosen as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Since

$$U = \begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & k \end{bmatrix},$$

the resultant

$$B = LU = \begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 1 & k+1 & -(k+1) \\ 0 & 0 & 0 & k \end{bmatrix},$$

and this is our matrix whose condition number is seriously underestimated.

A simple computation yields $\|B\|_1 = 4k+1$ and $\|B^{-1}\|_1 = 2k+1$ for $k \geqq 2$. The second estimation algorithm proceeds as follows:

Step 1. $b_1 = 1, z_1 = 1$.

Step 2. $p_2^{(1)} = -1 \qquad p_3^{(1)} = -2k \qquad p_4^{(1)} = 0$

$z_2^+ = 2$ and $z_2^- = 0$, thus since

$|2|+|-2k+2k|+|-2k| = 2k+2 > 2k = |0|+|-2k+0|+|0|,$

$b_2 = 1$ and $z_2 = z_2^+ = 2$.

Step 3. $p_3^{(2)} = 0 \qquad p_4^{(2)} = -2k$

$z_3^+ = 1$ and $z_3^- = -1$, thus since

$|1|+|-2k-1| = 2k+2 > 2k = |-1|+|-2k+1|,$

$b_3 = 1$ and $z_3 = z_3^+ = 1$.

Step 4. $p_4^{(3)} = -2k-1$

$z_4^+ = 2+2k^{-1}$ and $z_4^- = 2$, thus since

$|2k+2| = 2k+2 > 2k = |2k|,$

$b_4 = 1$ and $z_4 = z_4^+ = 1$.

(The choice of $z_2 = z_2^+$ rather than z_2^- causes the algorithm to overlook the ill-conditioning of B.)

It is easy to confirm that

$$x = B^{-T}b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2+2k^{-1} \end{bmatrix} \quad \text{and} \quad y = B^{-1}x = \begin{bmatrix} 6+4k^{-1} \\ 1 \\ 2k^{-1}+2k^{-2} \\ 2k^{-1}+2k^{-2} \end{bmatrix}.$$

Thus the estimated norm of B^{-1} is

$$\begin{aligned} \|y\|/\|x\| &= (7+8k^{-1}+4k^{-2})/(5+2k^{-1}) \\ &= (7k^2+8k+4)/(5k^2+2k), \end{aligned}$$

and the estimated condition number is this multiplied by $\|B\|_1 = 4k+1$ or $(28k^3+39k^2+24k+4)/(5k^2+2k)$, whereas the true condition number is $(2k+1)(4k+1) = 8k^2+6k+1$.

Example B. For $k \geqq 2$, the l_1 condition number of B is $8k^2+6k+1$; the estimated l_1 condition number (using the second algorithm of § 2) is $(28k^3+39k^2+24k+4)/(5k^2+2k) = 5.6k+5.56+0(1/k)$. The ratio of estimated condition number to true condition number is $(7k^2+8k+4)/(10k^3+9k^2+2k) = .7/k+1.7/k^2+O(k^{-3})$.

Experiments with the LINPACK subroutine SGECO using values of $k = 2, 4, 8, \cdots, 1024$ performed exactly as predicted above.

By examining the steps in the estimation, it is easy to see that Algorithm 1 will produce the same estimate as Algorithm 2 for this matrix. However Algorithm 3 chooses $b_2 = -1$ instead of $+1$ and thereby yields an estimated condition number of $8k^2+5+O(k^{-1})$. The ratio of estimated to true condition number is about $1-\frac{3}{4}k^{-1}+O(k^{-2})$.

A perturbation analysis shows that this matrix is not an isolated counter-example for Algorithms 1 and 2. In fact, it can be shown that if the matrix B is replaced by $B+E$ where $|e_{ij}| \leqq \varepsilon$ and $(e_{21}-e_{31})(1-e_{12}) > (e_{32}-e_{22})(1+e_{11})$ (a condition which guarantees that no interchanges are done when performing Gaussian elimination with partial pivoting on $B+E$), then the ratio of the estimated to the true condition number is less than about $.7k^{-1}+(15+5k)\varepsilon$. An example with $\varepsilon = 10^{-5}$ and $k = 100$ showed the ratio smaller than .00806 in 10,000 tests with random perturbations satisfying the above conditions. We may conclude that the matrix B actually sits within an open set of counter-examples.

The third counter-example is

$$C = \begin{bmatrix} 1 & 1-2k^{-2} & -2 \\ 0 & k^{-1} & -k^{-1} \\ 0 & 0 & 1 \end{bmatrix}.$$

For the second algorithm, computations similar to those above result in:

$$x = z = \begin{bmatrix} 1 \\ 2k^{-2} \\ 3+2k^{-2} \end{bmatrix} \quad \text{and} \quad y = C^{-1}x = \begin{bmatrix} 2+12k^{-2}+4k^{-4} \\ 5+2k^{-2} \\ 3+2k^{-2} \end{bmatrix}.$$

(As in example B, the choice of $z_2 = z_2^+$ rather than z_2^- causes the algorithm to overlook the ill-conditioning of C.) This yields the conclusion:

Example C. For $k \geqq 2$, the l_1 condition number of C is $6k+2+O(k^{-1})$; the estimated l_1 condition number (using the second algorithm) is $2.5+1.25k^{-1}+O(k^{-2})$. The ratio of estimated condition number to true condition number is $1.25k^{-1}-.625k^{-2}+O(k^{-3})$.

This matrix shows that the second algorithm can produce a bounded condition number estimate for an arbitrarily poorly conditioned matrix. In this situation both Algorithm 1 and Algorithm 3 perform well. Each estimates the condition number as $6k-4+O(k^{-1})$, and thus the ratio of estimated to true condition number is $1-k^{-1}+O(k^{-2})$.

By perturbations we can show that this matrix is not an isolated counter-example for Algorithm 2. In this case, if C is replaced by $C+E$ where the components e_{ij} of E are bounded by ε in magnitude and $\varepsilon \leqq \frac{1}{4}k^{-3}$, then the ratio of estimated condition number to true is less than about $\frac{5}{4}k^{-1}+.45/2k\varepsilon$. An example with $\varepsilon = 6\cdot 10^{-8}$ and $k = 125$ showed the ratio smaller than .009981 in 10,000 tests in which the perturbations were selected randomly.

A counter-example to all three algorithms appeared in [1]. Notice that if the upper triangular factor is a diagonal matrix with all components ± 1, then all of the algorithms result in $b_s = +1$, $s = 1, \cdots, n$. Thus the z vector is exactly the diagonal of U. If a unit lower triangular matrix is selected which is ill-conditioned but for which both $x = L^{-T}z$ and $y = U^{-1}L^{-1}x$ are moderately sized, then the ratio of $\|y\|$ to $\|x\|$ fails to display the ill-conditioning. Essentially we have placed all the ill-conditioning into the lower triangular factors, yet all the algorithms use the upper triangular factor to find ill-conditioning. Such a counter-example is the $n \times n$ matrix

$$D = \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & 0 & \\ -1 & -1 & 1 & & & \\ \vdots & \vdots & & \ddots & 1 & \\ -1 & -1 & \cdots & & -1 & -1 \end{bmatrix}.$$

Notice this is identical to the matrix of the second section except for the change in sign in the (n, n) element. Its l_1 condition number is $n2^{n-1}$, yet as the following example shows, all of the algorithms severely underestimate this.

Example D. For $n \geqq 2$, the l_1 condition number of D is $n2^{n-1}$; the estimated l_1 condition number (using the first, second, or third algorithms) is n. The ratio of estimated condition number to true condition number is 2^{1-n}.

This happens since the upper triangular factor of D is

$$U = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \\ 0 & & 1 & \\ & & & -1 \end{bmatrix},$$

and with all three algorithms, the choices for b_s are $+1$. Thus

$$x = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

so the estimate for $\|D^{-1}\|$ is 1 (yet the true value is 2^{n-1}). Since $\|D\| = n$, the estimated condition number is n, and the true value is $n2^{n-1}$.

This counter-example can be perturbed to yield other counter-examples, but care must be taken so that the perturbations do not cause interchanges in the elimination process. For this purpose it is easier to consider perturbing the upper and lower triangular factors and then obtaining counter-examples by multiplying these factors. If the upper triangular factor is perturbed in a positive fashion along the diagonal and in a negative fashion above the diagonal, then all algorithms still produce the decisions of $b_s = +1$ for all s. If the lower triangular factor is perturbed in a positive fashion below the diagonal, then a unit triangular matrix still results, and hence no interchanges will be performed if the product matrix is factored using Gaussian elimination with partial pivoting. With $n = 11$ and the perturbations selected as above but bounded by 10^{-5} in magnitude, in over 6,000 experiments, the condition number of the resulting matrix was always underestimated by a factor of less than .01467 using all three algorithms.

Table 1 summarizes the results of this section. The notation κ_i refers to the condition number estimated with the ith algorithm.

4. Conclusions. The algorithms found in [1] are not the only ones that have been suggested. Several extensions to these are found in Cline, Conn and Van Loan [2]. The first extension incorporates a "look-back" strategy to update the selection of b, to complement the "look-ahead" strategy already present in the second and third algorithms. The other extension applies a "divide and conquer" approach to the problem by examining certain principal submatrices. Experimental results with both extensions are comparable to the LINPACK estimator, yet the first extension performs poorly on examples B and D and the second extension performs poorly on examples A and D. Another proposal is to perform several additional steps of inverse iteration to the y resulting from the algorithms discussed in [1]. O'Leary [5] proposes computing the l_∞ condition number using the algorithms from [1] applied to the transpose of the original matrix. Although the counter-examples mentioned here are not counter-examples for this approach, it is clear that the transposes of the counter-examples here are such.

TABLE 1

Matrix	κ	κ_1	κ_1/κ	κ_2	κ_2/κ	κ_3	κ_3/κ
$\begin{bmatrix} 1 & 0 & k & -k \\ 0 & 1 & -k & k \\ 1 & -1 & 2k+1 & -2k \\ -1 & 1 & -2k & 2k+1 \end{bmatrix}$	$24k^2+22k+3$	$6k+1$	$\frac{1}{4}k^{-1}+O(k^{-2})$	$20k^2+18k+O(1)$	$\frac{5}{6}+O(k^{-1})$	$20k^2+18k+O(1)$	$\frac{5}{6}+O(k^{-1})$
$\begin{bmatrix} 1 & -1 & -2k & 0 \\ 0 & 1 & k & -k \\ 0 & 1 & k+1 & -(k+1) \\ 0 & 0 & 0 & k \end{bmatrix}$	$8k^2+6k+1$	$\frac{28}{5}k+O(1)$	$\frac{7}{10}k^{-1}+O(k^{-2})$	$\frac{28}{5}k+O(1)$	$\frac{7}{10}k^{-1}+O(k^{-2})$	$8k^2+5+O(k^{-1})$	$1-\frac{3}{4}k^{-1}+O(k^{-2})$
$\begin{bmatrix} 1 & 1-2k^{-2} & -2 \\ 0 & k^{-1} & -k^{-1} \\ 0 & 0 & 1 \end{bmatrix}$	$6k+2+O(k^{-1})$	$6k-4+O(k^{-1})$	$1-k^{-1}+O(k^{-2})$	$\frac{5}{2}+O(k^{-1})$	$\frac{5}{4}k^{-1}+O(k^{-2})$	$6k-4+O(k^{-1})$	$1-k^{-1}+O(k^{-2})$
$\begin{bmatrix} 1 & & & \\ -1 & 1 & 0 & \\ \vdots & & 1 & \\ -1 & \cdots & -1 & -1 \end{bmatrix}$	$n2^{n-1}$	n	2^{1-n}	n	2^{1-n}	n	2^{1-n}

The construction of counter-examples is insufficient motivation by itself for discarding condition number estimation algorithms, particularly those that have performed well in practice. It is the case that for any deterministic algorithm for interpolation, quadrature, or numerically solving differential equations, problems exist for which the algorithms perform arbitrarily poorly. This is due to the nature of the problem areas; the solution depends upon an infinite set of input information, and yet any algorithm must sample only a finite subset of this. In these areas, practical performance and soundly based heuristics are considered evidence of quality. The dilemma in judging condition number estimators is different: the opposition of efficiency and reliability. We *can* compute a condition number, although at a relatively great expense. The more difficult algorithms which only purport to estimate the condition number may be unrealiable.

The importance of the counter-examples is that they make clear that any effort toward proving that the algorithms always produce useful estimations is fruitless. It may be possible to prove that the algorithms provide useful estimations in certain situations, however, and this should be pursued. An effort simply to construct more complex algorithms is dangerous. A philosophical question should be answered before any highly complex algorithm is adopted: given that many condition number estimators perform well on random tests and in practice, yet fail for particular matrices, is the design of more and more complex algorithms yielding an improvement in estimation or is it simply erecting a higher barrier which makes the determination of counter-examples more difficult and also makes any theoretical analysis close to impossible? If more and more complex algorithms are constructed simply to escape from counter-examples, the process may terminate when counter-examples are not to be found only because of the limitations of human minds, not because of their nonexistence.

REFERENCES

[1] 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.

[2] A. K. CLINE, A. R. CONN AND C. VAN LOAN, *Generalizing the* LINPACK *condition estimator*, Technical Report TR81-462, Department of Computer Science, Cornell University, Ithaca, NY, 1981.

[3] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[4] G. E. FORSYTHE, M. A. MALCOLM AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[5] D. P. O'LEARY, *Estimating matrix condition number*, this Journal, 1 (1980), pp. 205–209.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0004 $01.25/0

AN EXTRAPOLATION METHOD FOR THE NUMERICAL SOLUTION OF SINGULAR PERTURBATION PROBLEMS*

F. C. HOPPENSTEADT† AND W. L. MIRANKER‡

Abstract. We show how the form of the perturbation approximation for the solution of stiff systems of ordinary differential equations with an identifiable small parameter can be used to generate associated nonstiff or relaxed equations. Solutions of these relaxed equations are easily calculated and appropriate combinations of these solutions furnish numerical approximations to the original stiff problem. Variations of this method are applied to two classes of initial value problems; those with highly oscillatory solutions and those with rapidly equilibrating solutions.

Key words. singular perturbations, highly oscillatory differential equations, stiff equations, averaging methods, matched asymptotic expansions

Introduction. Perturbation methods can be used to derive numerical schemes for solving ordinary differential equations whose solutions change on widely different time scales. Such schemes actually improve as the difference between the time scales increases. On the other hand, numerical implementation of the perturbation methods can involve costly evaluations of integrals which arise in averaging and matching conditions, and this approach usually requires some preliminary analysis of the problem.

Two methods are presented and compared here. These are described in detail for initial value problems to which the methods of averaging and matched asymptotic expansions can be applied. First, the perturbation solution is described along with its direct numerical evaluation. This frequently entails lengthy computations. Next, it is shown how the perturbation solution's form can be used to calculate the stiff solution by combining solutions of associated nonstiff, or relaxed, equations.

In many cases stiff differential equations involve a collection of identifiable small parameters, say $\vec{\varepsilon}$, which may be ratios of small to large eigenvalues or ratios of natural parameters arising in applications, such as reaction rates. Although we consider only the case where ε is a scalar (i.e., a single small parameter) the methods generalize to cases of several small parameters. First, problems having highly oscillatory solutions are studied by averaging methods. Then initial value problems having rapidly equilibrating solutions are studied using matched asymptotic expansions.

Problems having rapidly oscillating solutions are considered in § 1. The averaging procedure is reviewed, and then numerical methods based on it are presented. First, we consider direct evaluation of averages, and then an alternative method based on the almost periodic structure of the problem is described. This accelerates computation of averages by finding appropriate translation numbers of the function being averaged. Finally, a third averaging method is described which works in some special cases to which a form of the ergodic theorem applies. Last, an extrapolation method based on the already determined translation numbers, along with some illustrative computations, is presented.

In § 2, we study initial value problems having matched asymptotic expansion (MAE) solutions. First, the matching method is reviewed, then numerical schemes based on direct calculations of the MAE approximation and on an extrapolation method are described and sample calculations are presented.

* Received by the editors March 26, 1980, and in revised form April 7, 1982.
† Department of Mathematics, University of Utah, Salt Lake City, Utah 84112.
‡ IBM T. J. Watson Research Center, Yorktown Heights, New York 10598.

The novelty of this work lies in the introduction of extrapolation formulas, (7) in the averaging case, (12) in the MAE case. The extrapolation procedures introduced here are based on perturbation schemes which are appropriate to the two classes of problems studied here. They are distinct from straightforward Richardson extrapolation formulas which are based on Taylor's formula and which are not usually appropriate for stiff problems. It must be emphasized that the extrapolation formulas (7) and (12) break through the typical limitation of the customary numerical evaluation of one or more terms in the asymptotic expansions supplied by that methodology.

1. Averaging procedures. Many problems to which the Bogolyubov averaging method and various multi-time schemes are applied reduce to systems of the form

$$\frac{dx}{dt} = f\left(\frac{t}{\varepsilon}, x\right), \qquad x(0) = \xi, \tag{1}$$

where $x, f, \xi \in E^n$, and $f(\tau, \cdot)$ is an almost periodic function of τ. ε is a small positive parameter, and so in this case, the right-hand side involves highly oscillatory behavior. Multi-scale perturbation methods lead to the approximation

$$x(t, \varepsilon) = x_0(t) + \varepsilon x_1\left(t, \frac{t}{\varepsilon}\right) + O(\varepsilon^2), \tag{2}$$

where x_0 is determined from the initial value problem

$$\frac{dx_0}{dt} = \bar{f}(x_0), \qquad x_0(0) = \xi. \tag{3}$$

The approximation (2) is the sum of the solution's mean change (x_0), a small correction to account for variation of $\int_0^T f$ from $T\bar{f}$, (εx_1), and higher order corrections (see [7]). Here $\bar{f}$ is the mean value of f, defined by

$$\bar{f}(x_0) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f(\tau, x_0)\, d\tau.$$

The coefficient x_1 is determined from the formula

$$x_1\left(t, \frac{t}{\varepsilon}\right) = \tilde{x}_1(t) + \int_0^{t/\varepsilon} [f(\tau, x_0) - \bar{f}(x_0)]\, d\tau. \tag{4}$$

This expansion method is valid only if x_1 is a bounded function of t, ε. Thus, we assume that the average of $f(\bar{f}(x))$ exists as a smooth function of x and that the integral

$$\int_0^T [f(\tau, x) - \bar{f}(x)]\, d\tau$$

is bounded for all T.

In formula (4) $\tilde{x}_1$ is determined at a later step in the perturbation scheme. Since it will not be needed here, it is not discussed further (see [3]). Thus,

$$x(t, \varepsilon) = x_0(t) + \varepsilon\left\{\tilde{x}_1(t) + \int_0^{t/\varepsilon} [f(\tau, x_0) - \bar{f}(x_0)]\, d\tau\right\} + O(\varepsilon^2).$$

This approximation suggests several numerical schemes for determining $x(h, \varepsilon)$:

A. Computation of $\bar{f}$. We wish to determine numerical approximations of $x(h, \varepsilon)$ for a given step size $h \gg \varepsilon$. The customary method for approximating $x(h, \varepsilon)$ is to

calculate $x_0(h)$ from (3). This requires the calculation of $\bar{f}$ and in this section we propose several methods for doing this.

(i) *Direct evaluation of* $\lim_{T\to\infty}(1/T)\int_0^T f$. A convergence criterion is first set, and then the integral $(1/T)\int_0^T f$ is calculated for increasing values of T until the criterion is met: Given a tolerance δ, there is a value $T(\delta, x)$ such that

$$\left|\frac{1}{T_1}\int_0^{T_1} f(\tau, x)\,d\tau - \frac{1}{T_2}\int_0^{T_2} f(\tau, x)\,dt\right| < \delta \quad \text{and} \quad \left|\frac{1}{T_1}\int_0^{T_1} f(\tau, x)\,d\tau - \bar{f}(x)\right| < \delta$$

for all $T_1, T_2 \geqq T(\delta, x)$. Thus we can write

$$\bar{f}(x) = \frac{1}{T(\delta, x)}\int_0^{T(\delta,x)} f(\tau, x)\,d\tau + O(\delta)$$

and proceed to solve (3). Unfortunately, there is no certain way of finding $T(\delta, x)$, and its calculation may depend delicately on the frequencies of $f(\tau, x)$.

In order to find a candidate for $T(\delta, x)$, we calculate

$$V(T, x) = \int_0^T f(\tau, x)\,d\tau$$

for $0 \leqq T \leqq 2T^*$, and keep increasing T^* until the condition

$$\sup_{0\leqq T'\leqq T^*}\left|\frac{1}{T^*}V(T^*, x) - \frac{1}{(T^*+T')}V(T^*+T', x)\right| \leqq \delta$$

is met. Then we take $T(\delta, x) = 2T^*$. Usually $2T^*$ is of order $O(1/\delta)$. For example, if $f(\tau, x) = \sin\tau$, then $\bar{f}(x) = 0$ and $V(T, x) = \cos T - 1$. Thus in this case

$$0 \leqq \left|\frac{1}{T}V(T, x)\right| \leqq \frac{2}{T},$$

the maximum being attained in each interval of length 2π.

(ii) *Second difference method.* In most applications, the integral of the almost periodic function f has the form

$$V(T, x) = \bar{f}(x)T + p(T, x),$$

where p is an almost periodic function. Thus, given a tolerance δ, there is a δ-translation number $\mathcal{F}(\delta, x)$ such that

$$|p(T + \mathcal{F}(\delta, x), x) - p(T, x)| < \delta$$

for all $T \geqq 0$; in particular, since $p(0, x) = 0$, then $|p(\mathcal{F}(\delta, x), x)| < \delta$.

If the frequencies of V, hence of p, are known, then $\mathcal{F}(\delta, x)$ can be determined by a Diophantine approximation procedure [4]. Thus, Fourier transform methods can be used to determine the spectrum, and a Diophantine algorithm used to find $\mathcal{F}(\delta, x)$. We do not carry this procedure out here. Instead we find candidates for $\mathcal{F}$ by an alternate method.

Note that

$$V(2T, x) - 2V(T, x) = p(2T, x) - 2p(T, x).$$

In particular for $T = \mathcal{F}(\delta, x)$, we have that

$$V(2\mathcal{F}, x) - 2V(\mathcal{F}, x) = p(2\mathcal{F}, x) - p(\mathcal{F}, x) - p(\mathcal{F}, x) + p(0, x) = O(\delta). \tag{5}$$

Thus, any δ-translation number of p makes (5) of order δ. Unfortunately, the converse

does not hold; in particular, $|V(2T, x)-2V(T, x)|$ may be small while $|p(T, x)|$ is not small.

Still, by tabulating

$$V(2T, x)-2V(T, x),$$

candidates for $\mathscr{F}(\delta, x)$ can be found and tested by comparing the value of $V(T, x)/T$ for several of them, since these should all approximate $\bar{f}(x)$. In practice, this method is not worse than the direct calculation of f, and in periodic cases, it reliably gives $\bar{f}(x)$ after calculation over one period.

Thus, from either i or ii, we use $(1/\mathscr{F}(\delta, x))\int_0^{\mathscr{F}(\delta,x)} f(\tau, x)\,d\tau$ as an approximation to $\bar{f}$ and proceed to integrate (3) using this approximation.

(iii) *An ergodic method for quasi-periodic systems.* In many cases, f is given to be, or can be transformed into, or can be closely approximated by quasi-periodic functions of the form

$$f(\tau, x)\sim F(w_1\tau, \cdots, w_L\tau, x)$$

where $w_1, \cdots, w_L$ are rationally independent numbers, and the function $F(u_1, \cdots, u_L, x)$ has period 2π in each of the L variables $u_1, \cdots, u_L$. In this case,

$$\bar{f}(x)\equiv \lim_{T\to\infty}\frac{1}{T}\int_0^T f(\tau, x)\,d\tau = \frac{1}{(2\pi)^L}\int_0^{2\pi} F(\theta_1, \cdots, \theta_L, x)\,d\theta_1\cdots d\theta_L.$$

The computational method in this case can proceed by using the formula for $\bar{f}(x)$ to integrate (3) directly.

Thus, the time average can be replaced by an average over an L-torus, reminiscent of an ergodic theorem. This fact is derived rigorously in [5]; note however, that the result follows intuitively from the generalized Fourier series expansion of F since only the constant term survives either operation. In this case, calculation of $\bar{f}$ requires evaluation of an L-fold integral over a finite domain. This of course can be costly if $L>4$. However, in many problems from nonlinear vibrations, angle variables over which averaging must be carried out are "sparse" in the system, making this method practical.

B. The extrapolation method. In this section we use the form of the perturbation approximation of $x(h, \varepsilon)$ to generate an extrapolation formula including a remainder term for approximating $x(h, \varepsilon)$. The formula employs quantities like $x(h, \varepsilon')$ which are themselves easily determined since $\varepsilon' \gg \varepsilon$.

To begin, recall that

$$V(t, x)\equiv\int_0^t f(\tau, x)\,d\tau = \bar{f}(x)t + p(t, x).$$

We pick T to be a δ-translation number of p by one of the methods mentioned earlier in this section, or by additional knowledge of the specific problems being studied. Then

$$\begin{aligned} p(2T, x)-2p(T, x) &= V(2T, x)-2T\bar{f}(x)-2V(T, x)+2T\bar{f}(x)\\ &= V(2T, x)-V(T, x)-V(T, x)\\ &= \int_T^{2T} f(\tau, x)\,d\tau-\bar{f}(x)T-p(T, x)\\ &= \int_T^{2T}[f(\tau, x)-\bar{f}(x)]\,d\tau+O(\delta). \end{aligned}$$

It follows that

$$\frac{1}{T}\int_T^{2T} [f(\tau, x)-\bar{f}(x)]\,d\tau = O(\delta). \tag{6}$$

We will combine this observation with (2) and (4).

Once a T value is found for which (6) is satisfied, we define

$$\varepsilon' = \frac{h}{T},$$

and calculate $x(h, \varepsilon'/2)$ and $x(h, \varepsilon')$ from (1) by a rth-order numerical method. It follows from (2) and (4) that

$$\begin{aligned} 2x\left(h, \frac{\varepsilon'}{2}\right) - x(h, \varepsilon') &= x_0(h) + \varepsilon' \int_T^{2T} [f(\tau, x_0(h)) - \bar{f}(x_0(h))]\,d\tau + O((\varepsilon')^2) \\ &= x_0(h) + O(\delta h) + O\left(\left(\frac{h}{T}\right)^2\right). \end{aligned}$$

On the other hand,

$$\begin{aligned} x(h, \varepsilon) &= x_0(h) + \varepsilon \tilde{x}_1 + \varepsilon \int_0^{h/\varepsilon} [f(\tau, x_0(h)) - \bar{f}(x_0(h))]\,d\tau + O(\varepsilon^2) \\ &= x_0(h) + O(\varepsilon), \end{aligned}$$

since

$$\int_0^{h/\varepsilon} [f(\tau, x_0(h)) - \bar{f}(x_0(h))]\,d\tau = O(1).$$

Therefore,

$$x(h, \varepsilon) = 2x\left(h, \frac{\varepsilon'}{2}\right) - x(h, \varepsilon') + O(\varepsilon) + O(\delta h) + O\left(\left(\frac{h}{T}\right)^2\right). \tag{7}$$

This formula gives the extrapolation method for calculating $x(h, \varepsilon)$. It and the previous methods are compared for an example in the next subsection.

In (7) $2x(h, \varepsilon'/2) - x(h, \varepsilon')$ is used as an approximate replacement for $x(h, \varepsilon)$; it is in fact also an approximate replacement for $x_0(h)$. Thus, while calculating $x(h, \varepsilon')$ from (1) is usually much easier than calculating $x(h, \varepsilon)$ since $\varepsilon' \gg \varepsilon$, it is possible to calculate $x_0(h)$ directly. There are several reasons why this might not be desirable. Primarily equation (1) with ε replaced by ε' is not a stiff equation, and it can be reliably solved by simple explicit numerical methods. Equation (3) for x_0 is not stiff but its formulation requires the evaluation of the average, $\bar{f}$, at several meshpoints. As discussed earlier, this time consuming step is bypassed by the extrapolation method.

Note that if T differs by the quantity Δ from the approximation $\mathscr{F}(\delta, x)$ of the δ-translation number, then from (7), we see that the corresponding difference in the associated values of $x(h, \varepsilon)$ is

$$O\left(\frac{h^2}{T^3}\right)\Delta. \tag{8}$$

The dependence of this estimate on h is illustrated in Table 1.

C. Sample calculations: a linear system. In this subsection we present simple calculations performed with the various methods derived here for the linear initial value problem

$$\frac{du}{dt} = \left(\left(\frac{1}{\varepsilon}\right)A + B\right)u, \qquad u(0) = \xi.$$

This is transformed into the problem

$$\frac{dv}{dt} = e^{-At/\varepsilon} B e^{At/\varepsilon} v, \qquad v(0) = \xi$$

by the change of variables $u = e^{At/\varepsilon}v$. We take $v \in E^4$ and $\xi = (1, 1, 1, 1)^{\mathrm{Tr}}$.

Running values of $V(T, \xi)$ are computed using a quadrature increment of $\Delta\tau = .01$. The tolerances for each of the methods (i) and (ii) are denoted by δ_{i} and δ_{ii}, respectively, and the corresponding values of T at which the associated computations halt are denoted T_{i} and T_{ii}, respectively. The calculations are carried out for two different matrices:

$$A(1, w) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & w \\ 0 & 0 & -w & 0 \end{pmatrix}, \qquad A(2, w) = \begin{pmatrix} 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & 3 \\ 0 & 0 & 0 & w \\ 0 & 0 & -w & 0 \end{pmatrix},$$

where w is a parameter specified below.

B is taken to be

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 \end{pmatrix}.$$

The average $\bar{B}$

$$\bar{B} = \lim_{T\to\infty} \frac{1}{T} \int_0^T e^{-At} B\, e^{At}\, dt$$

can be determined in closed form, which we denote by $\bar{B}(1, w)$ and $\bar{B}(2, w)$, respectively. These are compared with $\bar{B}$'s obtained by methods i and ii, e.g., $\bar{B}(1, w)$ is compared with $\bar{B}_{\mathrm{i}}(1, w)$ and $\bar{B}_{\mathrm{ii}}(1, w)$ in Tables 1–3.

The averaging method gives $e^{\bar{B}h}\xi$ as an approximation to $x(h, \varepsilon)$. Tables 1–3 compare these approximations for the three ways of determining $\bar{B}$. The extrapolation formula (7) gives an approximation to $x(h, \varepsilon)$, which is presented in the tables as well. These are denoted by Ex $(\varepsilon'_{\mathrm{i}})$ and Ex $(\varepsilon'_{\mathrm{ii}})$, respectively, where $\varepsilon'_{\mathrm{i}} = h/T_{\mathrm{i}}$ and $\varepsilon'_{\mathrm{ii}} = h/T_{\mathrm{ii}}$.

Observe that a value of ε is not prescribed for these computations. This demonstrates the characteristic of numerical methods based on perturbation procedures in that they supply approximations which are uniformly good for all ε smaller in modulus than some prescribed value ε_0, say. ε_0 depends only on the accuracy desired of the leading term of the expansion as an approximation to the full solution. Calculations are given for values of ε' conveniently near the appropriate values of h/T since ε' is only approximately determined by the method.

TABLE 1
Case 1. $A = A(1, 1/2)$.
a. Approximations to $\bar{B}$

				Method (i)				Method (ii)			
Exact average				$\delta_i = .001$		$T_i = 163.36$		$\delta_{ii} = .001$		$T_{ii} = 25.12$	
$\bar{B}(1, 1/2)$				$\bar{B}_i(1, 1/2)$				$\bar{B}_{ii}(1, 1/2)$			
3.5	−1.5	0.0	0.0	3.5	−1.5	−.001	−.001	3.501	−1.502	−.002	−.002
1.5	3.5	0.0	0.0	1.5	3.5	0.0	0.0	1.498	3.499	−.004	−.004
0.0	0.0	3.5	1.5	0.0	0.0	3.5	1.5	−.004	−.004	3.499	1.498
0.0	0.0	−1.5	3.5	0.0	0.0	−1.5	3.5	−.002	−.002	−1.502	3.501

b. Approximations to $x(h, \varepsilon)$, $h = .1$.

Approximation	Approximating vector			
$e^{\bar{B}h}\xi$	1.19	1.61	1.61	1.19
$e^{\bar{B}_i h}\xi$	1.20	1.60	1.60	1.20
$e^{\bar{B}_{ii} h}\xi$	1.20	1.60	1.60	1.20
Ex (.0006)	1.19	1.62	1.62	1.19
Ex (.004)	1.19	1.62	1.61	1.19

c. Approximations to $x(h, \varepsilon)$, $h = .05$

Approximation	Approximating vector			
$e^{\bar{B}h}\xi$	1.10	1.28	1.28	1.10
$e^{\bar{B}_i h}\xi$	1.10	1.28	1.28	1.10
$e^{\bar{B}_{ii} h}\xi$	1.10	1.28	1.28	1.10
Ex (.0003)	1.10	1.28	1.28	1.10
Ex (.002)	1.10	1.28	1.28	1.10

D. Discussion of these numerical methods. Existing stiff differential equation solving routines, such as Gear's [6], can degrade markedly when applied to problems having highly oscillatory solutions since computations must continually be made using very small increments. On the other hand, methods like those presented here can require extensive a priori preparation of the system to be solved.

Numerical implementation of the averaging procedure requires the determination of T_i (§ 1A(i)) which is used in direct approximation of $\bar{f}$ or determination of T_{ii} (§ 1A(ii)) as an approximate translation number for $p(T, x)$. The ergodic method (§ 1A(iii)) can involve computation of high dimensional integrals.

The ratio h/ε is a measure of the system's stiffness, while $1/(\varepsilon h)$ measures the work involved in direct computation of solutions. On the other hand, $T_i/\Delta T$ (ΔT is the increment used in the averaging quadrature) measures the work needed to calculate $\bar{f}$, and $T_{ii}/\Delta T$ gives the work involved in calculating the approximate translation number. While the method based on approximation of $\bar{f}$ (§ 1A(i)) is reliable, it is costly even for a periodic function f. The work involved in the computation of translation numbers varies from minimal (e.g., for a periodic function) to no improvement when resonances occur in the integral V. Computations illustrating the ergodic method (§ 1A(iii)) were not carried out here since the process reduces to a well known quadrature problem.

TABLE 2
Case 2. $A = A(1, 1/4)$.
a. Approximations to $\bar{B}$

Exact average				Method (i) $\delta_i = .001$ $T_i = 158.34$				Method (ii) $\delta_{ii} = .001$ $T_{ii} = 50.28$			
$\bar{B}(1, 1/4)$				$\bar{B}_i(1, 1/4)$				$\bar{B}_{ii}(1, 1/4)$			
3.5	−1.5	0.0	0.0	3.48	−1.51	−.085	−.027	3.50	−1.50	.001	.001
1.5	3.5	0.0	0.0	1.40	3.52	−.024	−.026	1.501	3.501	.002	.002
0.0	0.0	3.5	1.5	−.085	.026	3.40	1.53	.002	.002	3.50	1.50
0.0	0.0	−1.5	3.5	.024	0.026	−1.47	3.60	.001	.001	−1.50	3.50

b. Approximations to $x(h, \varepsilon)$, $h = .1$.

Approximation	Approximating vector			
$e^{\bar{B}h}\xi$	1.19	1.61	1.61	1.19
$e^{\bar{B}_i h}\xi$	1.18	1.60	1.59	1.22
$e^{\bar{B}_{ii} h}\xi$	1.28	1.60	1.60	1.20
Ex (.0006)	1.20	1.61	1.64	1.17
Ex (.002)	1.19	1.62	1.62	1.19

c. Approximations to $x(h, \varepsilon)$, $h = .05$.

Approximation	Approximating vector			
$e^{\bar{B}h}\xi$	1.10	1.28	1.28	1.10
$e^{\bar{B}_i h}\xi$	1.09	1.28	1.27	1.11
$e^{\bar{B}_{ii} h}\xi$	1.10	1.28	1.28	1.10
Ex (.0003)	1.10	1.27	1.29	1.09
Ex (.001)	1.10	1.27	1.29	1.09

The value of ε' used in this method is typically much larger than ε, but still small. Therefore, the relaxed problem may still be stiff, but it will be significantly less so than the original problem.

Finally, we note that formula (7) shows the error arising in the extrapolation procedure decreases as a power of h; e.g., replacing h by $h/2$ implies the error changes from h^2/T to $h^2/(4T)$. Consequences of this fact are illustrated in Table 1 where computations are carried out for two or three values of h. For example, from Case 3, Table 3, we see that for the three values of h, $h = .1$, $h = .05$ and $h = .025$, $\|e^{.1\bar{B}}\xi - \text{Ex}(.001)\| = 59.5$, $\|e^{.05\bar{B}}\xi - \text{Ex}(.0005)\| = 8.1$, $\|e^{.025\bar{B}}\xi - \text{Ex}(2.5\text{E}-4)\| = 1.1$. Here $\|\cdot\|$ denotes the Euclidean norm.

2. Initial value problems having matched asymptotic expansion solutions. Consider an initial value problem

$$\frac{dx}{dt} = f(t, x, y, \varepsilon), \qquad x(0) = \xi(\varepsilon) \sim \xi(0) + O(\varepsilon),$$

(9)

$$\varepsilon \frac{dy}{dt} = g(t, x, y, \varepsilon), \qquad y(0) = \eta(\varepsilon) \sim \eta(0) + O(\varepsilon),$$

TABLE 3
Case 3. $A = A(2, 1/2)$.
a. Approximations to $\bar{B}$

Exact average				Method (i) $\delta_i = 0.1$ $T_i = 404.28$				Method (ii) $\delta_{ii} = .0001$ $T_{ii} = 75.4$			
$B(2, 1/2)$				$B_i(2, 1/2)$				$B_{ii}(2, 1/2)$			
9.17	−28.17	−290.56	−60.48	9.10	−28.03	−289.05	−60.12	9.17	−28.17	−290.6	−60.44
28.17	−9.17	−52.89	289.44	28.09	9.03	52.81	−288.48	28.17	9.17	52.89	−289.44
0.0	0.0	−2.17	29.83	0.02	0.028	−2.15	29.77	0.0	0.0	−2.17	29.39
0.0	0.0	−29.83	2.17	−0.013	0.039	−29.71	−1.98	0.0	0.0	−29.83	−2.17

b. Approximations to $x(h, \varepsilon)$, $h = .1$.

Approximations	Approximating vector			
$e^{\bar{B}h}\xi$	−55.4	−26.5	−1.31	−5.98
$e^{\bar{B}_i h}\xi$	−55.2	−26.6	−1.29	−5.97
$e^{\bar{B}_{ii} h}\xi$	−55.4	−26.5	−1.31	−5.98
Ex (.0002)	1.09	−13.9	−.701	−.873
Ex (.001)	1.546	−13.6	−.761	−1.01

c. Approximations to $x(h, \varepsilon)$, $h = .05$.

Approximation	Approximating vector			
$e^{\bar{B}h}\xi$	−22.4	−10.9	1.12	−1.55
$e^{\bar{B}_i h}\xi$	−22.3	−10.9	1.12	1.54
$e^{\bar{B}_{ii} h}\xi$	−22.4	−10.9	1.12	−1.55
Ex (.0001)	−14.2	−9.82	1.12	−1.54
Ex (.0005)	−14.3	−9.77	.97	.81

d. Approximations to $x(h, \varepsilon)$, $h = .025$.

Approximation	Approximating vector			
$e^{\bar{B}h}$	−9.46	−4.46	1.38	−.036
Ex (.00025)	−8.46	−4.35	1.34	.056

where the real parameter ε is near zero. Here, x, f, $\xi \in E^m$ and y, g, $\eta \in E^n$. The reduced problem ($\varepsilon = 0$ in (9) with the initial y condition cancelled),

$$\frac{dx}{dt} = f(t, x, y, 0), \quad x(0) = \xi(0), \quad 0 = g(t, x, y, 0), \tag{10}$$

is assumed to have a solution, $x = x_0(t)$, $y = y_0(t)$, on some interval $0 \leqq t \leqq T$, the data f, g, ξ and η are assumed to be smooth functions of their arguments and the Jacobian matrix

$$\left(\frac{\partial g}{\partial y}\right)(t, x_0(t), y_0(t), 0)$$

is assumed to be stable (i.e., all eigenvalues lie in the left half plane, bounded away from the imaginary axis uniformly for $0 \leqq t \leqq T$).

Under these conditions it is known [1] that the solution of (9) has the form

$$
\begin{aligned}
x(t,\varepsilon) &= x_0(t)+x_1(t)\varepsilon + X\left(\frac{t}{\varepsilon},\varepsilon\right)+O(\varepsilon^2),\\
y(t,\varepsilon) &= y_0(t)+y_1(t)\varepsilon + Y\left(\frac{t}{\varepsilon},\varepsilon\right)+O(\varepsilon^2),
\end{aligned}
\tag{11}
$$

where X, Y satisfy

$$
\left|X\left(\frac{t}{\varepsilon},\varepsilon\right)\right|+\left|Y\left(\frac{t}{\varepsilon},\varepsilon\right)\right|\leqq K e^{-\mu t/\varepsilon},
$$

for some positive constants K and μ. These estimates hold uniformly for $0\leqq t\leqq T$ and for all small positive ε.

We wish to determine numerical approximations of $x(h,\varepsilon)$, $y(h,\varepsilon)$, for a given step size $h\gg\varepsilon$. In § 2A we do this by direct calculation of the leading terms $x_0(h)$, $y_0(h)$ of the asymptotic expansion; this is the customary approach. In § 2B we derive the extrapolation method for this approximation. In § 2C we present and compare sample calculations.

A. Direct calculation of $x_0(h)$, $y_0(h)$. Since $h/\varepsilon\gg 1$, we can ignore X, Y-terms in (11), and to simplify the computations we will be satisfied with $x_0(h)$, $y_0(h)$, as an approximation to $x(h,\varepsilon)$, $y(h,\varepsilon)$. The numerical calculation of higher order terms in the perturbation expansion is discussed elsewhere (see [2]).

$x_0(h)$, $y_0(h)$ can be calculated from (10) in a number of ways. One widely used method combines Newton iteration and a pth-order numerical method for solving ordinary differential equations. The result is that

$$
\begin{aligned}
x(h,\varepsilon) &= x_0(h)+O(\varepsilon)+O(h^{p+1}),\\
y(h,\varepsilon) &= y_0(h)+O(\varepsilon)+O(h^{p+1}).
\end{aligned}
$$

B. The extrapolation method. The extrapolation method begins by identifying a value of ε, say ε', which is substantially larger than ε in magnitude, but for which the solution of (9) with ε replaced by ε' can be used to approximate $x(h,\varepsilon)$, $y(h,\varepsilon)$. Thus, (9) is solved for larger values of ε, and so it can be solved more accurately with less effort. In many applications the number of operations used in these computations varies with $1/\varepsilon$ and $1/\varepsilon'$, respectively. Therefore we use the ratio ε'/ε to indicate the relative number of operations of direct solution compared to the extrapolation method.

First, a value T is determined so that $Ke^{-\mu T}=O(h^{p+1})$. μ is usually of the order of the smallest (modulus) eigenvalue of g_y, and K depends on Lipschitz constants of g; and determination of K typically requires special knowledge of the probem. Next, a value $\varepsilon'=h/T$ is defined, and system (9) is solved by a standard (order p) integration method for $x(h,\varepsilon'/2)$, $y(h,\varepsilon'/2)$, and $x(h,\varepsilon')$, $y(h,\varepsilon')$. It follows that

$$
\begin{aligned}
x(h,\varepsilon) &= 2x\left(h,\frac{\varepsilon'}{2}\right)-x(h,\varepsilon')+O(h^{p+1})+O((\varepsilon')^2)+O(\varepsilon),\\
y(h,\varepsilon) &= 2y\left(h,\frac{\varepsilon'}{2}\right)-y(h,\varepsilon')+O(h^{p+1})+O((\varepsilon')^2)+O(\varepsilon).
\end{aligned}
$$

These formulas can be derived in the following way: Since

$$2x\left(h, \frac{\varepsilon'}{2}\right) = 2x_0(h) + x_1(h)\varepsilon' + 2X\left(2T, \frac{\varepsilon'}{2}\right) + O((\varepsilon')^2),$$

and

$$x(h, \varepsilon') = x_0(h) + x_1(h)\varepsilon' + X(T, \varepsilon') + O((\varepsilon')^2),$$

we get by subtraction that

$$2x\left(h, \frac{\varepsilon'}{2}\right) - x(h, \varepsilon') = x_0(h) + O(h^{p+1}) + O((\varepsilon')^2).$$

On the other hand,

$$x(h, \varepsilon) = x_0(h) + O(\varepsilon) + O(h^{p+1}),$$

and similarly, for $y(h, \varepsilon)$. The final result is that

$$\begin{aligned} x(h, \varepsilon) &= 2x\left(h, \frac{\varepsilon'}{2}\right) - x(h, \varepsilon') + O(h^{p+1}) + O(\varepsilon) + O\left(\left(\frac{h}{T}\right)^2\right), \\ y(h, \varepsilon) &= 2y\left(h, \frac{\varepsilon'}{2}\right) - y(h, \varepsilon') + O(h^{p+1}) + O(\varepsilon) + O\left(\left(\frac{h}{T}\right)^2\right). \end{aligned} \tag{12}$$

This and the direct evaluation methods are compared for two examples in the following section.

Again, the approximations given in (12), using the extrapolation method, provide a possible alternative method to direct evaluation of $x_0(h)$, $y_0(h)$. Both approaches give the same numerical accuracy. However, analytical preprocessing might be required to pose the problem in the form (9). Once this is done, the equation $0 = g(t, x, y, 0)$ must be solved for y at each meshpoint required by the solving method to be used in the first equations in (10). Both the preprocessing and the solution for the quasi-static state y can involve a substantial investment of time. These are avoided by the extrapolation method.

C. Sample calculations.

(i) *A linear system.* We consider first a linear example of (9) (see [2]):

$$\frac{dx}{dt} = y - x, \quad x(0) = \xi, \qquad \frac{dy}{dt} = -\frac{1}{\varepsilon}y + 1, \quad y(0) = \eta. \tag{13}$$

The eigenvalues of this system are -1 and $-1/\varepsilon \ll -1$. The exact solution is given by

$$x(t) = e^{-t}\xi + (1 - e^{-t})\varepsilon - \left(\frac{\varepsilon}{1-\varepsilon}\right)(\eta - \varepsilon)(e^{-t/\varepsilon} - e^{-t}), \qquad y(t) = \varepsilon + e^{-t/\varepsilon}(\eta - \varepsilon),$$

and the leading terms of the matched asymptotic expansion solutions are

$$x(t) = e^{-t}\xi + \varepsilon\{(\eta - 1)e^{-t} + 1\} + \cdots, \qquad y(t) = \varepsilon + \cdots.$$

Table 4 summarizes our calculations of the solutions for typical ε ($\varepsilon = 10^{-5}$) and h ($h = .1$ and $.01$, respectively) values. Evaluation of the matched asymptotic expansion solution to leading order and the extrapolation method are presented in Table 4. Since $K = 1 + |\eta|$ and $\mu = 1$ in this case, we take $T = -\ln[h^{p+1}/(1 + |\eta|)]$. The calculated values are for $\varepsilon' = h/T$ ($\varepsilon' = .008$ and $\varepsilon' = .003$, respectively). Calculations are also presented for nearby values of ε', since ε' is only approximately determined by the

TABLE 4
$\xi = \eta = 1$, $p = 4$,

$h = .1$, $\varepsilon' = .008$				
Extrapolation method	ε'	$x(h)$	$y(h)$	ε'/ε
	.005	.863	.002	500
	.008	.864	.004	819
	.01	.865	.005	1,000
	.02	.870	.010	2,000
Matched solution	—	.905	0.0	
$h = .01$ $\varepsilon' = .003$				
Extrapolation method	ε'	$x(h)$	$y(h)$	ε'/ε
	.003	.986	.0015	300
	.004	.987	.0025	400
Matched solution	—	.990	0.0	

method. A Runge–Kutta method of order four has been used to calculate the values of $x(h)$ and $y(h)$ furnished by the extrapolation method in all of the examples presented in Tables 4 and 5. In order to assure high accuracy and stability these Runge–Kutta calculations were performed on a submesh of $[0, h]$ with submesh increment $k = h\varepsilon'$.

Observe that although $\varepsilon = 10^{-5}$, the computations displayed in Table 4 are independent of ε and are valid to the accuracy indicated uniformly for all ε smaller in modulus than some prescribed value ε_0, say. ε_0 depends only on the accuracy desired of the leading term of the perturbation expansion as an approximation to the full solution. This same observation may be made for the example (ii) and Table 5 below.

Notice that the extrapolation method gives an answer to 4% accuracy for $h = .1$, but it gives better than a 1% accuracy for $h = .01$. Of course, in each case, a submesh of points, appropriate to ε' is employed to reach these values $h = .1$ and $h = .01$, respectively. For example, in the case $h = .01$ a submesh increment of 10^{-3} is appropriate.

To compare our extrapolation method with the package of C. W. Gear which is frequently used for integrating stiff differential equations, we employed the latter to solve (13) with $\varepsilon = 10^{-5}$ for the case $h = .01$ in Table 4. The package calculates the solution at $h = .01$ by employing a variable submesh of points which it determines adaptively. The package requires an average step size of 1.96×10^{-4} to produce a 1% answer. The extrapolation method with $\varepsilon' = .003$ produces a 1% answer with a fixed step size of 9.75×10^{-4}. As ε decreases, the latter remains invariant. There are so many details of variation in each of these methods that this comparison should only be interpreted qualitatively.

(ii) *A model enzyme reaction.* A simple enzyme reaction involves an enzyme E, substrate S, complex C and product P. Schematically, the reaction is

$$E + S \rightleftharpoons C, \qquad C \rightleftharpoons E + P.$$

After some preliminary scaling, this reaction can be described by a system of differential equations for the substrate concentration (x) and the complex concentration (y) as

$$\frac{dx}{dt} = -x + (x+k)y, \qquad x(0) = 1,$$
$$\varepsilon \frac{dy}{dt} = x - (x+k')y, \qquad y(0) = 0, \tag{14}$$

where ε measures a typical ratio of enzyme to substrate ($O(10^{-5})$), and k and k' ($k < k'$) denote ratios of rate constants suitably normalized ($O(1)$).

Table 5 summarizes the result of these numerical calculations for $\varepsilon = 10^{-5}$, $h = .1$ and $.01$, $k = 1$, $k' = 2$. In this case, $K = 1$, $\mu = k'$, so we take $T = -((p+1)/2) \ln h$.

TABLE 5
$x(0) = 1$, $y(0) = 0$, $p = 4$

$h = 1$, $\varepsilon' = .04$				
Extrapolation method	ε'	$x(h)$	$y(h)$	ε'/ε
	.01	.953	.323	1,000
	.04	.960	.325	4,000
	.05	.961	.325	5,000
	.1	.973	.328	10,000
	.15	.988	.335	15,000
	.2	.994	.341	20,000
Matched solution	—	.989	.331	

$h = .01$, $\varepsilon' = .0009$				
Extrapolation method	ε'	$x(h)$	$y(h)$	ε'/ε
	.0004	.995	.332	40
	.0008	.995	.332	80
	.0009	.995	.332	90
	.001	.995	.332	100
	.0016	.995	.332	160
Matched solution	—	.992	.332	

The calculated values are for $\varepsilon' = h/T$ ($\varepsilon' = .04$ and $\varepsilon' = .0009$, respectively). Calculations are also presented for some nearby values of ε'.

The extrapolation method gives a 3% accurate solution for $h = .1$, but it gives better than a 1% solution for $h = .01$. As in the case of Table 4, a comparison here produces an average step size of 2.4×10^{-4} for a 1% answer for Gear's package as opposed to a fixed step size of 2.7×10^{-3} for a 1% answer for the extrapolation method with $\varepsilon' = .0009$. (See the comments following Tables 1–3 above.)

REFERENCES

[1] F. C. HOPPENSTEADT, *Properties of solutions of ordinary differential equations with small parameters*, Comm. Pure Appl. Math., XXIV (1971). pp. 807–840.

[2] W. L. MIRANKER, *The computational theory of stiff differential equations*, Lecture Notes, No. 219–7667, V. Paris XI, U.E.R. Mathematique, 91405 Orsay, France.

[3] S. C. PERSEK AND F. C. HOPPENSTEADT, *Iterated averaging methods for systems of ordinary differential equations with a small parameter*, Comm. Pure Appl. Math., XXXI (1978), pp. 133–156.

[4] W. LEVEQUE, *Topics in Number Theory*, Addison-Wesley, Reading, MA, 1977.

[5] F. HOPPENSTEADT AND H. ROSSI, *Remarks on the method of averaging*, Tech. Rep., Dept. Mathematics, Univ. of Utah, Salt Lake City, 1979.

[6] A. C. HINDMARSH, *Gear's ordinary differential equation solver*, UCID-30001 (Rev. 3), Lawrence Livermore Lab., Livermore, CA, Dec., 1974.

[7] F. HOPPENSTEADT AND W. MIRANKER, *Differential equations having rapidly changing solutions: Analytic methods for weakly nonlinear systems*, J. Differential Equations, 22 (1976), pp. 237–249.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/82/0404-0005 $01.00/0

AN APPLICATION OF MIXED FINITE ELEMENT METHODS TO THE STABILITY OF THE INCOMPRESSIBLE NAVIER–STOKES EQUATION*

JANET S. PETERSON†

Abstract. Following the work of Serrin [Arch. Rational Mech. Anal., 3 (1958), pp. 1–13], [Encyclopedia of Physics, Springer-Verlag, Berlin, 1959], the problem of determining the energy stability of incompressible viscous flows is cast in a setting where one is required to determine the largest eigenvalue of a partial differential equation problem. A finite element algorithm for the approximation of this eigenvalue is presented. Error estimates are derived for the eigenvalue approximation. In addition, some remarks are given concerning the solution of the algebraic eigenvalue problem resulting from the finite element discretization. Finally, the use of the finite element algorithm is illustrated by numerical examples.

Key words. finite element, stability, Navier–Stokes, eigenvalue

1. Introduction.

We consider the problem of determining the stability of viscous incompressible flows. Let a fluid motion occupy a bounded region $\mathcal{V}(t)$ and be subject to a prescribed velocity distribution on the boundary $\mathcal{S}$ of $\mathcal{V}(t)$. If the velocity field is perturbed at the instant $t = 0$, then the question of stability of the flow is to determine if the subsequent motion, subjected to the same boundary conditions, will change radically in character or will vary only slightly. This problem can be addressed by either standard linear perturbation techniques which require the perturbations to be infinitesimal in magnitude or by an energy method. The latter approach is considered here.

Specifically, we want to determine sufficient conditions for the flow of an incompressible viscous fluid to be stable under arbitrary disturbances. Let $\mathbf{v}$ be the velocity of the given flow whose stability we are investigating; let $\mathbf{v}^*$ be the velocity of the flow resulting from the perturbation of the given flow and let $\mathbf{u}$ be the velocity of the difference flow, i.e., $\mathbf{u} = \mathbf{v}^* - \mathbf{v}$. The energy method is based on the observation that if the kinetic energy of the difference flow $\mathbf{u}$ tends to zero then $\mathbf{u}$ must also tend to zero almost everywhere. Thus, we define the basic flow to be stable provided the kinetic energy K of the disturbance $\mathbf{u}$ tends to zero as t increases. Using the energy method, Serrin [1] was able to formulate this stability question as a linear eigenvalue problem. We present a brief derivation of this problem.

The initial step in formulating the eigenvalue problem is to obtain an expression for dK/dt. We first note that since $\mathbf{v}$ and $\mathbf{v}^*$ satisfy the same boundary conditions, $\mathbf{u}$ is zero on $\mathcal{S}$. Now using this fact, the incompressibility condition and the fact that $\mathbf{v}$ and $\mathbf{v}^*$ satisfy the nonstationary (nonlinear) Navier–Stokes equations, we obtain the following expression [1]:

$$\frac{dK}{dt} = -\int_{\mathcal{V}(t)} \{\nu \operatorname{grad} \mathbf{u} : \operatorname{grad} \mathbf{u} + \mathbf{u} \cdot \mathbb{D} \cdot \mathbf{u}\}\, d\mathcal{V}. \tag{1.1}$$

Here ν is the kinematic viscosity and $\mathbb{D}$ is the deformation tensor of the given flow $\mathbf{v}$. The (i, j) component of $\mathbb{D}$ is

$$D_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right).$$

* Received by the editors August 4, 1981, and in revised form June 14, 1982. This work was sponsored in part by the Air Force Office of Scientific Research under grant AFOSR-80-0083.

† Department of Mathematics, University of Pittsburgh, Pittsburgh, Pennsylvania 15261.

The colon in (1.1) represents the scalar product of the tensor grad $\mathbf{u}$ with itself. It is clear that if the right-hand side of (1.1) is negative for arbitrary $\mathbf{u}$ satisfying div $\mathbf{u} = 0$ in $\mathcal{V}$ and $\mathbf{u} = 0$ on $\mathcal{S}$, then the flow is stable. Note, however that the first term on the right-hand side of (1.1) is always negative, whereas the second term may be large and positive due to a high rate of shear of the given flow. Restating these conditions, we obtain the following variational problem [1]. Seek $\mathbf{u}$ such that

$$\int_{\mathcal{V}} \mathbf{u} \cdot \mathbb{D} \cdot \mathbf{u}\, d\mathcal{V} \tag{1.2}$$

is minimized subject to the conditions

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \mathcal{V}, \tag{1.3}$$

$$\int_{\mathcal{V}} \operatorname{grad} \mathbf{u} : \operatorname{grad} \mathbf{u}\, d\mathcal{V} = 1, \tag{1.4}$$

$$\mathbf{u} = 0 \quad \text{on } \mathcal{S}. \tag{1.5}$$

Letting $-\nu^*$ be the minimum of (1.2) subject to the conditions (1.3)–(1.5), we see that the flow is stable when $\nu^* < \nu$. Using the standard techniques of the calculus of variations, we reformulate equations (1.2)–(1.5) as a partial differential equation for the extremal function $\mathbf{u}$. Introducing Lagrange multipliers p and $\tilde{\nu}$ for the constraints (1.3) and (1.4), respectively, the Euler–Lagrange equations are given by

$$\mathbf{u} \cdot \mathbb{D} = -\operatorname{grad} p + \tilde{\nu}\, \Delta \mathbf{u}, \quad \text{in } \mathcal{V}(t), \tag{1.6}$$

$$\operatorname{div} \mathbf{u} = 0 \tag{1.7}$$

$$\mathbf{u} = 0 \quad \text{on } \mathcal{S}. \tag{1.8}$$

The flow is stable if the kinematic viscosity ν is greater than the maximum eigenvalue ν^* of (1.6)–(1.8). Thus, given the deformation tensor $\mathbb{D}$ of the given flow, we must calculate the maximum eigenvalue of (1.6)–(1.8) in order to determine sufficient conditions for the flow to be stable. However, since most flows are too complex to solve for this eigenvalue exactly, we propose to obtain an approximation to ν^* by mixed finite element methods. In the analysis that follows, it is advantageous to rewrite (1.6)–(1.8) in the form

$$\Delta \mathbf{u} - \operatorname{grad} \phi = \lambda\, \mathbf{u} \cdot \mathbb{D} \quad \text{in } \mathcal{V}(t), \tag{1.9}$$

$$\operatorname{div} \mathbf{u} = 0 \tag{1.10}$$

$$\mathbf{u} = 0 \quad \text{on } \mathcal{S} \tag{1.11}$$

where $\phi = p/\tilde{\nu}$ and $\lambda = 1/\tilde{\nu}$, $\tilde{\nu} \neq 0$.

We note that the resulting eigenvalue problem (1.6)–(1.8) or equivalently (1.9)–(1.11) is linear despite the fact that the perturbation $\mathbf{u}$ to the given flow may be of finite magnitude.

2. Notation. Standard Sobolev space notation is used throughout this work [2]. The rth order Sobolev space associated with a bounded region Ω is denoted $H^r(\Omega)$ with $\|\cdot\|_r$ indicating the usual norm on $H^r(\Omega)$. $L_2(\Omega)$ is the space of all functions which are square integrable and is equal to $H^0(\Omega)$. The inner product on L_2 is indicated by $(\cdot, \cdot)$. Also, $H_0^1(\Omega)$ denotes the subspace of $H^1(\Omega)$ of functions which vanish on the boundary while $\mathbf{H}_0^1(\Omega)$ denotes the space of vector fields whose components have weak derivatives up to order one in $L^2(\Omega)$ in each variable and which vanish on the

boundary of Ω. For functions $u \in H_0^1$, we will use the norm

$$\|u\|_1^2 \equiv (\nabla u, \nabla u),$$

while for functions $u \in H^1$ we will use the norm

$$\|u\|_1^2 \equiv (\nabla u, \nabla u) + (u, u).$$

The operator norm for $A: X \to Y$ is given by

$$\|A\|_{XY} = \sup_{\substack{x \in X \\ x \neq 0}} \frac{\|Ax\|_Y}{\|x\|_X}.$$

3. Abstract results. For the eigenvalue problem (1.9)–(1.11) we consider the following variational formulation which is obtained by the standard techniques of the Galerkin method; that is, we multiply (1.9) and (1.10) by appropriate test functions and then integrate by parts. The weak formulation of (1.9)–(1.11) is to find nonzero $\mathbf{u} \in \mathbf{H}_0^1(\mathcal{V})$, $\phi \in L_2(\mathcal{V})$ and $\lambda \in R$ such that

$$a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, \phi) = \lambda(\mathbb{D}\mathbf{u}, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \tag{3.1}$$

$$b(\mathbf{u}, \psi) = 0 \quad \text{for all } \psi \in L_2(\mathcal{V}) \tag{3.2}$$

where $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$ are bilinear forms defined by

$$a(\mathbf{u}, \mathbf{v}) = \int_{\mathcal{V}} \nabla \mathbf{u} : \nabla \mathbf{v} \, d\mathcal{V} \quad \text{for all } \mathbf{u}, \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \tag{3.3}$$

$$b(\mathbf{v}, \phi) = \int_{\mathcal{V}} \phi \operatorname{div} \mathbf{v} \, d\mathcal{V} \quad \text{for all } \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \quad \phi \in L_2(\mathcal{V}). \tag{3.4}$$

It is easily verified that $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$ are bounded bilinear forms; that is, there exist constants c_1, c_2 such that

$$|a(\mathbf{u}, \mathbf{v})| \leqq c_1 \|\mathbf{u}\|_1 \|\mathbf{v}\|_1 \quad \text{for all } \mathbf{u}, \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \tag{3.5}$$

$$|b(\mathbf{v}, \phi)| \leqq c_2 \|\mathbf{v}\|_1 \|\phi\|_0 \quad \text{for all } \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \quad \phi \in L_2(\mathcal{V}). \tag{3.6}$$

In addition, $a(\cdot, \cdot)$ is coercive on $\mathbf{H}_0^1(\mathcal{V}) \times \mathbf{H}_0^1(\mathcal{V})$; i.e., there exists $\alpha > 0$ such that

$$a(\mathbf{u}, \mathbf{u}) \geqq \alpha \|\mathbf{u}\|_1^2. \tag{3.7}$$

Since the goal is to approximate the solution of (3.1)–(3.2), we introduce finite dimensional subspaces $\mathbf{V}^h \subset \mathbf{H}_0^1(\mathcal{V})$ and $W^h \subset L_2(\mathcal{V})$ which depend upon a parameter $0 < h < 1$ tending to zero. The approximate problem is to find nonzero $(\mathbf{u}^h, \phi^h) \in \mathbf{V}^h \times W^h$ and $\lambda^h \in \mathbb{R}$ such that

$$a(\mathbf{u}^h, \mathbf{v}^h) + b(\mathbf{v}^h, \phi^h) = \lambda^h(\mathbb{D}\mathbf{u}^h, \mathbf{v}^h) \quad \text{for all } \mathbf{v}^h \in \mathbf{V}^h, \tag{3.8}$$

$$b(\mathbf{u}^h, \psi^h) = 0 \quad \text{for all } \psi^h \in W^h. \tag{3.9}$$

The objective of the subsequent analysis is to estimate the difference between λ and λ^h as h approaches zero. This error estimate is derived in Theorem 3.4.

The source problems associated with the continuous and the discrete eigenvalue problems play an important role in the error analysis. For the continuous source problem, $f \in L_2$ is given and we seek $(\mathbf{u}, \phi) \in \mathbf{H}_0^1(\mathcal{V}) \times L_2(\mathcal{V})$ such that

$$a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, \phi) = (f, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{H}_0^1(\mathcal{V}), \tag{3.10}$$

$$b(u, \psi) = 0 \quad \text{for all } \psi \in L_2(\mathscr{V}). \tag{3.11}$$

The approximate source problem associated with (3.8), (3.9) is to find $(\mathbf{u}^h, \phi^h) \in \mathbf{V}^h \times W^h$ such that

$$a(\mathbf{u}^h, \mathbf{v}^h) + b(\mathbf{v}^h, \phi^h) = (f, \mathbf{v}^h) \quad \text{for all } \mathbf{v}^h \in \mathbf{V}^h, \tag{3.12}$$

$$b(\mathbf{u}^h, \psi^h) = 0 \quad \text{for all } \psi^h \in W^h \tag{3.13}$$

where $f \in L_2$ is given. Brezzi [3] investigated the existence and uniqueness of source problems of this type. In order to guarantee the existence and uniqueness of the solutions of (3.10), (3.11) and (3.12), (3.13), $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$ must be continuous as well as satisfy certain stability conditions on the appropriate spaces. The continuity of $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$ on $\mathbf{H}_0^1 \times \mathbf{H}_0^1$ and $\mathbf{H}_0^1 \times L_2$ as well as on $\mathbf{V}^h \times \mathbf{V}^h$ and $\mathbf{V}^h \times W^h$ is guaranteed by (3.5) and (3.6). The coercivity of $a(\cdot, \cdot)$ is a sufficient condition for its stability. The stability condition for $b(\cdot, \cdot)$ is proved in [4] on $\mathbf{H}_0^1 \times L_2$ and in [5], [6] for the particular choice of subspaces used in the examples.

Next we introduce the components of the solution operator. For the continuous source problem, we define solution operators S and T such that

$$S: L_2 \to \mathbf{H}_0^1 \quad \text{and} \quad Sf = \mathbf{u}, \tag{3.14}$$

$$T: L_2 \to L_2 \quad \text{and} \quad Tf = \phi \tag{3.15}$$

where $\mathbf{u}$ and ϕ are defined by (3.10), (3.11). Similarly for the discrete source problem we define the operators S^h and T^h such that

$$S^h: L_2 \to \mathbf{V}^h \quad \text{and} \quad S^h f = \mathbf{u}^h, \tag{3.16}$$

$$T^h: L_2 \to W^h \quad \text{and} \quad T^h f = \phi^h \tag{3.17}$$

where $\mathbf{u}^h$ and ϕ^h are determined by (3.12), (3.13). The S^h's form a sequence of compact operators since they are maps into finite dimensional subspaces. We now note that if $(\lambda, \mathbf{u})$ and ϕ satisfy (3.1), (3.2) a comparison with (3.10), (3.11) with $f = \lambda \mathbb{D}\mathbf{u}$ and the definition of S imply that $\lambda S \mathbb{D}\mathbf{u} = \mathbf{u}$. Similarly if $\lambda S \mathbb{D}\mathbf{u} = \mathbf{u}$ then there exists a ϕ such that λ, $(\mathbf{u}, \phi)$ satisfies (3.1), (3.2). This gives the following result.

LEMMA 3.1. $(\lambda, \mathbf{u})$ *satisfies* (3.1), (3.2) *if and only if* $\lambda S \mathbb{D}\mathbf{u} = \mathbf{u}$, $\mathbf{u} \neq 0$.

A similar result holds for the discrete case, which we give for completeness.

LEMMA 3.2. $(\lambda^h, \mathbf{u}^h)$ *satisfies* (3.3), (3.4) *if and only if* $\lambda^h S^h \mathbb{D}\mathbf{u}^h = \mathbf{u}^h$, $\mathbf{u}^h \neq 0$.

We assume that S^h converges to S in the operator norm and thus S is compact. For the particular choice of the space $\mathbf{V}^h$ in the examples in the next section, we will see that, indeed, this is the case. This assumption guarantees convergence of the eigenvalues; that is, given an eigenvalue λ of S of multiplicity m, then there are exactly m eigenvalues of S^h counted according to multiplicity which converge to λ [7]. We are now able to compare the eigenvalues λ of (3.1)–(3.2) with their Galerkin approximations λ^h by comparing the eigenvalues of the compact operator $S\mathbb{D}$ with those of $S^h\mathbb{D}$.

Using the machinery that has been developed thus far, we are now ready to obtain the eigenvalue error estimate. As a starting point for the estimate, we state the following result of Osborn [8] for the spectral approximation of compact operators.

THEOREM 3.3. *Let* λ *be an eigenvalue of* $S\mathbb{D}$ *of multiplicity* m *and* λ_l^h, $l = 1, 2, \cdots, m$ *be the* m *eigenvalues of* $S^h\mathbb{D}$ *which converge to* λ. *Let* $\psi_1, \cdots, \psi_m$ *be an orthonormal basis for the eigenspace of* $S\mathbb{D}$ *corresponding to* (λ^{-1}), *denoted by* $M(\lambda^{-1})$.

Then there exists a constant c independent of h such that

$$|\lambda-\lambda_l^h| \leqq c\left\{\sum_{i,j=1}^{m}((S-S^h)\mathbb{D}\psi_i,\psi_j)+\|(S-S^h)\mathbb{D}|_{M(\lambda^{-1})}\|_0^2\right\}, \quad l=1,2,\cdots,m, \tag{3.18}$$

where $|_{M(\lambda^{-1})}$ denotes the restriction to $M(\lambda^{-1})$.

The eigenvalue error estimate is obtained by estimating the first term on the right-hand side of (3.18). With λ, λ_l^h defined as in Theorem 3.3 and S, S^h, T and T^h defined by (3.14)–(3.17), we have the following result:

THEOREM 3.4. *There exists a constant c independent of h such that*

$$|\lambda-\lambda_l^h| \leqq c\{\|S-S^h\|_{L_2H^1}^2+\|S-S^h\|_{L_2H^1}\|T-T^h\|_{L_2L_2}+\|S-S^h\|_{L_2L_2}^2\}. \tag{3.19}$$

Proof. To obtain an estimate for the first term on the right-hand side of (3.18) we estimate $(f,(S-S^h)g)$ where f, $g\in L_2$. Adding equations (3.10), (3.11) with $\mathbf{v}=(S-S^h)g\in\mathbf{H}_0^1$, $\psi=(T-T^h)g\in L_2$ and using (3.14), (3.15) yield

$$a(Sf,(S-S^h)g)+b((S-S^h)g,Tf)+b(Sf,(T-T^h)g)=(f,(S-S^h)g). \tag{3.20}$$

We now combine the continuous and discrete source problems (3.10)–(3.13) with f equal to g to obtain

$$a(\mathbf{u}-\mathbf{u}^h,\mathbf{v}^h)+b(\mathbf{v}^h,\phi-\phi^h)+b(\mathbf{u}-\mathbf{u}^h,\psi^h)=0,$$

which holds for all $\mathbf{v}^h\in\mathbf{V}^h$ and $\psi^h\in W^h$. Choosing $\mathbf{v}^h=S^hf$, $\psi^h=T^hf$ and using the definition of the solution operators (3.14)–(3.15) lead to

$$a((S-S^h)g,S^hf)+b(S^hf,(T-T^h)g)+b((S-S^h)g,T^hf)=0.$$

We now subtract this result from (3.20) and use the symmetry of $a(\cdot,\cdot)$ to obtain

$$\begin{aligned}(f,(S-S^h)g)=a((S-S^h)f,(S-S^h)g)+b((S-S^h)g,(T-T^h)f)\\ +b((S-S^h)f,(T-T^h)g).\end{aligned}$$

Using the boundedness of the bilinear forms (3.5), (3.6) yields

$$\begin{aligned}|(f,(S-S^h)g)| \leqq c\{&\|(S-S^h)f\|_1\|(S-S^h)g\|_1\\ &+\|(S-S^h)g\|_1\|(T-T^h)f\|_0+\|(S-S^h)f\|_1\|(T-T^h)g\|_0\}.\end{aligned}$$

Choosing $f=\psi_j$ and $g=\mathbb{D}\psi_i$ produces

$$\begin{aligned}|(\psi_j,(S-S^h)\mathbb{D}\psi_i)| \leqq c\{&\|(S-S^h)\psi_j\|_1\|(S-S^h)\mathbb{D}\psi_i\|_1\\ &+\|(S-S^h)\mathbb{D}\psi_i\|_1\|(T-T^h)\psi_j\|_0+\|(S-S^h)\psi_j\|_1\|(T-T^h)\mathbb{D}\psi_i\|_0\}.\end{aligned} \tag{3.21}$$

We now substitute (3.21) into (3.18) and use the fact that $\mathbb{D}$ is bounded to obtain the desired result.

4. Examples. The approximate eigenvalue problem (3.8), (3.9) is equivalent to an algebraic generalized eigenvalue problem. This is shown in the usual manner by choosing bases $\{\boldsymbol{\mu}_i\}$ and $\{\omega_i\}$ for $\mathbf{V}^h$ and W^h, respectively. We are then led to

$$\begin{pmatrix}A & B^T\\ B & 0\end{pmatrix}\begin{pmatrix}X_1\\ X_2\end{pmatrix}=\lambda^h\begin{pmatrix}D & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}X_1\\ X_2\end{pmatrix} \tag{4.1}$$

where A, B, D are matrices whose elements are given by $(A)_{ij}=a(\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)$, $(B)_{ij}=b(\boldsymbol{\mu}_i,\omega_i)$, $(D)_{ij}=(\mathbb{D}\boldsymbol{\mu}_i,\boldsymbol{\mu}_j)$ and X_1, X_2 are vectors whose elements consist of coefficients in the expansion of $\mathbf{u}^h$ and ϕ^h in terms of the bases for $\mathbf{V}^h$ and W^h respectively.

For most problems the system (4.1) is large and sparse. If A is $n \times n$, B, $m \times n$, $n > m$, then we consider a method for reducing the $(n+m)\times(n+m)$ system (4.1) to an $(n-m)\times(n-m)$ system. To do this we obtain an orthogonal basis for the null space B by performing a QR decomposition of B^T. That is,

$$B^T = (V_1 V_2)\begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

where $Q = (V_1 V_2)$ is an orthogonal matrix and R_1 is upper triangular. From (4.1) $Bx_1 = 0$, and thus using the QR decomposition we have that $LV_1^T x_1 = 0$ where $L = R_1^T$. If B is of full rank, L is invertible and thus $V_1^T x_1 = 0$. Using this fact we can reduce the first equation in (4.1) to

$$V_2^T D V_2 y = \tilde{\nu}^h V_2^T A V_2 y \tag{4.2}$$

where $y = V_2^T x_1$ and $\tilde{\nu}^h$ is $(\lambda^h)^{-1}$. We are interested in finding the maximum eigenvalue of (4.2).

We consider two computational examples. The first is a plane flow in a rectangle defined by the deformation tensor

$$\mathbb{D} = \begin{pmatrix} 0.0 & 0.5 \\ 0.5 & 0.0 \end{pmatrix}.$$

This is a simple example to illustrate the feasibility of our approximation algorithm for two-dimensional problems.

Care must be exercised in choosing the finite element spaces $\mathbf{V}^h$ and W^h. For example, if we triangulate as in Fig. 1A and let $\mathbf{V}^h$ be the space of continuous vector fields which vanish on the boundary and are linear in each triangle and W^h be the

A. *Box grid.* B. *Criss-cross grid.*

FIG. 1. *Grids.*

space of functions which are piecewise constant on each triangle, then the numerical approximation is unstable [9]. Stable approximations are defined by each of the two following finite element pairs based on the grids of Figs. 1A and 1B. For both grids we define $\mathbf{V}^h$ to be the space of continuous vector fields which vanish on the boundary and are linear on each triangle. For the *box grid* we let W^h be the space of piecewise constant functions on each *rectangle* while for the *criss-cross grid* we define $W^h = \operatorname{div} \mathbf{V}^h$. We note that in the latter case W^h is a subspace of the space of functions which are piecewise constant on each triangle. For these specific choices of $\mathbf{V}^h$ and W^h, the error estimates for the source problem are given by [9]

$$\|(S - S^h)f\|_0 \leqq ch^2 \|f\|_0, \tag{4.3}$$

$$\|(S - S^h)f\|_1 \leqq ch \|f\|_0, \tag{4.4}$$

$$\|(T - T^h)f\|_0 \leqq ch \|f\|_0. \tag{4.5}$$

These error estimates do not depend on the uniformity of the grids but are valid for polygonal regions with nonuniform grids. Note that (4.3) guarantees the norm convergence of S^h to S. Using these estimates in Theorem 3.4 produces the following eigenvalue error estimate:

$$|\lambda - \lambda_l^h| \leqq ch^2, \qquad l = 1, 2, \cdots, m. \tag{4.6}$$

The maximum eigenvalue for this example was computed for both the criss-cross and box grids using the reduction discussed at the beginning of this section. The results are given in Table 1. The extrapolated value was computed using Richardson's extrapolation and assuming the convergence is of order h^2.

TABLE 1

Eigenvalue approximations for two-dimensional flow.

	Criss-cross grid		Box grid	
h	Maximum eigenvalue	Extrapolated value	Maximum eigenvalue	Extrapolated value
$\frac{1}{5}$	.002450		.002437	
$\frac{1}{6}$	.002744	.003412	.002731	.003392
$\frac{1}{7}$	.002927	.003434	.002915	.003425
$\frac{1}{8}$			.003039	.003444
$\frac{1}{9}$			.003124	.003444
$\frac{1}{10}$			.003186	.003451
$\frac{1}{11}$			.003233	.003453
$\frac{1}{12}$			.003268	.003454
$\frac{1}{13}$			.003296	.003455

The second example that we consider is Couette flow, which is flow between two rotating cylinders. Let R_2 denote the radius of the outer cylinder which is rotating with angular speed Ω_2 and let R_1 denote the radius of the inner cylinder rotating with angular speed Ω_1. The deformation tensor for this flow is given by [10]:

$$\mathbb{D} = -\frac{\beta}{r^2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\beta = -\frac{(R_1R_2)^2(\Omega_2 - \Omega_1)}{R_2^2 - R_1^2}.$$

First, we write (1.6) and (1.7) in cylindrical coordinates. We then assume that the flow is independent of θ, the polar angle, and that the solution is periodic in the axial variable z; that is, the kth axial mode of the flow variables are only functions of r, the cylindrical radius. In this case the weak form of the equations for the kth mode are

$$\int_{R_1}^{R_2} \frac{\beta}{r^2} vur\, dr + \int_{R_1}^{R_2} p\left(u + r\frac{\partial u}{\partial r}\right) dr = \tilde{\nu}\left[\int_{R_1}^{R_2} \left(\frac{\partial u}{\partial r}\right)^2 r\, dr + \left(k^2 + \frac{1}{r^2}\right)\int_{R_1}^{R_2} u^2 r\, dr\right], \tag{4.7}$$

$$\int_{R_1}^{R_2} \frac{\beta}{r^2} uvr\,dr = \tilde{\nu}\int_{R_1}^{R_2} \left(\frac{\partial v}{\partial r}\right)^2 r\,dr + \left(k^2 + \frac{1}{r^2}\right)\int_{R_1}^{R_2} v^2 r\,dr, \tag{4.8}$$

$$k\int_{R_1}^{R_2} pwr\,dr = \tilde{\nu}\left[\int_{R_1}^{R_2} \left(\frac{\partial w}{\partial r}\right)^2 r\,dr + k^2\int_{R_1}^{R_2} w^2 r\,dr\right], \tag{4.9}$$

$$\int_{R_1}^{R_2} p\left(u + r\frac{\partial u}{\partial r}\right) dr + k\int_{R_1}^{R_2} pwr\,dr = 0, \tag{4.10}$$

where the first equation holds for all $u \in H_0^1$, the second for all $v \in H_0^1$, the third for all $w \in H_0^1$ and the last for all $p \in L_2$. Here u, v and w are the components of the velocity and k is the axial wave number [10].

Equations (4.7)–(4.10) were solved numerically. The velocity was approximated by piecewise linear functions on each interval, and p, by piecewise constant functions on each interval. The error estimates (4.3)–(4.5) still apply, and thus (4.6) gives the theoretical rate of convergence.

Computations were performed for various values of Ω_1 and Ω_2, each verifying the predicted estimate (4.6). We give here only the case when $\Omega_1 = 1.292$ and $\Omega_2 = 1.0$. The radii were chosen as $R_1 = 3.55$ and $R_2 = 4.03$ so that comparisons with known experiments and previous approximate calculations could be made. Table 2 gives the computed results for this particular example. The eigenvalue is a function of the axial

TABLE 2

Eigenvalue approximations for couette flow.

Intervals	Maximum eigenvalue	Extrapolated value
10	.0062169	
11	.0062415	.0063585
12	.0062602	.0063582
13	.0062746	.0063581
14	.0062861	.0063579
15	.0062954	.0063578
16	.0063029	.0063578
17	.0063092	.0063577
18	.0063144	.0063577
19	.0063188	.0063577
20	.0063226	.0063576

wave number k. The given intervals are of equal length h and the extrapolated values are obtained by using Richardson's extrapolation and assuming the convergence is $O(h^2)$.

In [10] Serrin reports calculations which provide the stability criterion

$$\left|\frac{\Omega_2 - \Omega_1}{\nu}\right| \leqq 45.5 \tag{4.11}$$

for the particular pair of radii 3.55 and 4.03. Serrin does concede that this number is, at best, an estimate, but hydrodynamic experiments do support it. According to the numerical calculations reported in Table 2, we find that

$$\left|\frac{\Omega_2 - \Omega_1}{\nu^*}\right| \leqq 45.93,$$

which is in close agreement with (4.11). Here ν^* is the maximum positive eigenvalue of (4.7)–(4.10).

Acknowledgment. The author would like to thank Professor M. D. Gunzburger for many helpful discussions.

REFERENCES

[1] J. SERRIN, *Mathematical Principles of Classical Fluid Mechanics*, Encyclopedia of Physics, Springer-Verlag, Berlin, 1959.
[2] R. A. ADAMS, *Sobolev Spaces*, Academic Press, New York, 1975.
[3] F. BREZZI, *On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers*, RAIRO, 8-R2 (1974), pp. 129–151.
[4] R. A. NICOLAIDES, *Existence, uniqueness and approximation for generalized saddle point problems*, Report 79-22, ICASE, Hampton, VA, 1979.
[5] G. FIX AND M. SURI, *The construction of stable conforming finite elements for the Navier–Stokes equations*, to appear.
[6] J. BOLAND AND R. A. NICOLAIDES, *Stability of finite elements for the Navier–Stokes equations*, to appear.
[7] J. E. OSBORN, *Spectral approximation for compact operators*, Math. Comp., 29 (1975), pp. 712–725.
[8] ———, *Eigenvalue approximation by mixed methods*, in Advances in Computer Methods for Partial Differential Equations, III, I.M.A.C., 1979, pp. 158–161.
[9] M. D. GUNZBURGER, R. A. NICOLAIDES AND J. PETERSON, *On conforming mixed finite element methods for incompressible viscous flow problems*, Comput. Math. Appl., 8 (1982), pp. 167–179.
[10] J. SERRIN, *On the stability of viscous fluid motion*, Arch. Rational Mech. Anal., 3 (1958), pp. 1–13.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0006 $01.25/0

TYPE-INSENSITIVE ODE CODES BASED ON EXTRAPOLATION METHODS*

LAWRENCE F. SHAMPINE†

Abstract. It is shown how to construct a code for the initial value problem for a system of ordinary differential equations (ODEs) which at every step recognizes and responds efficiently to stiffness and so is insensitive to the (possibly changing) type of the problem. Extrapolation of the explicit midpoint rule is used when this is feasible and otherwise extrapolation of a semi-implicit midpoint rule is used.

Key words. ODE, type-insensitive, stiff, extrapolation, midpoint rule

1. Introduction. For a long time, extrapolation of the (explicit) midpoint rule [4] has been a popular way to solve nonstiff initial value problems for systems of ordinary differential equations (ODEs). In the last few years Bader and Deuflhard have been studying the theory and practice of extrapolation of a semi-impliciti midpoint rule for the solution of stiff problems. They have developed an effective code, METAN1 [1], which will be the object of our attention in this paper.

To describe our objectives, it will first be necessary to sketch how the extrapolation methods work. Suppose y_0 approximates $y(x_0)$, where $y(x)$ is a solution of

$$\frac{d}{dx}y = f(y). \tag{1}$$

To advance one step H to an approximation of $y(x_0+H)$, extrapolation methods repeatedly integrate from x_0 to x_0+H with a low order method and subsequently combine the results to obtain a high order result at x_0+H. The midpoint rules can be expressed uniformly in terms of a matrix J. The semi-implicit rule forms the Jacobian matrix at y_0, $f_y(y_0)$, and takes J to be this matrix. The explicit midpoint rule takes J to be 0. In the notation

$$\bar{f}(y) = f(y) - Jy,$$

each subintegration proceeds as follows. An integer $2m$ is selected, and then with $h = H/2m$,

$$\eta_0 = y_0, \qquad \eta_1 = (I - hJ)^{-1}[y_0 + h\bar{f}(y_0)],$$

and for $k = 1, \cdots, 2m$,

$$\eta_{k+1} = (I - hJ)^{-1}[(I + hJ)\eta_{k-1} + 2h\bar{f}(\eta_k)].$$

Finally,

$$S_{2m} = \tfrac{1}{2}(\eta_{2m+1} + \eta_{2m-1})$$

is a smoothed approximation to the solution at x_0+H. Polynomial extrapolation in h^2 is now done on the S_{2m} to obtain a high order approximation for $y(x_0+H)$.

If $J = 0$ is used in the scheme just sketched, one obtains a variant of the usual extrapolated explicit midpoint rule for solving nonstiff problems. The only difference

* Received by the editors June 8, 1982, and in revised form August 20, 1982. This work was performed at Sandia National Laboratories, supported by the U.S. Department of Energy under contract DC-AC04-76DP00789.

† Numerical Mathematics Division 1642, Sandia National Laboratories, Albuquerque, New Mexico 87185.

is in the final smoothing. The one described was devised by Bader to improve the stability when solving stiff problems. It is not essentially different, *when applied to nonstiff problems*, from the usual smoothing of Gragg [4]. It is clear that the principal role of the Jacobian is to stabilize the computations when solving stiff problems, although it does affect the accuracy to some degree. There are major costs for the semi-implicit scheme which are unnecessary for nonstiff problems, viz. the formation of the Jacobian J for each step H, Gaussian decomposition of $I-hJ$ for each subintegration, and the solution of $2m+1$ linear systems for each subintegration.

This outline of the situation suggests the possibility of devising an algorithm based on METAN1 which would recognize at every step when an explicit method is feasible, and so produce a code which is effective regardless of the changing problem type—stiff or nonstiff. This amounts to deciding whether we must use $J=f_y(y_0)$, or can use $J=0$ for the step. There is already some evidence that an effective code is possible. In his development of a step size and order selection algorithm for extrapolation methods, Deuflhard [2] compared METAN1 to some of the very best codes based on extrapolation of the explicit midpoint rule when applied to a standard set of nonstiff test problems. Despite the fact that METAN1 computes Jacobians by differences of f values, it is quite acceptable when compared in terms of the number of evaluations of f (function evaluations). However, the total computing time was about three times larger due to the linear algebra costs. Our principal goal is to reduce these costs to a minimum by recognizing automatically when they are unnecessary. Another major goal is to avoid the unnecessary formation of Jacobians and so improve the efficiency of the code for nonstiff problems. This is more difficult because the information carried in the Jacobian is invaluable for recognizing stiffness, and because it is not necessarily true that it is better to take $J=0$ for a nonstiff problem. The tests reported by Deuflhard somewhat understate the importance of this goal because the test set consists mainly of small sets of equations.

It is so convenient for users not to have to worry about what stiffness is, or about the type changing from stiff to nonstiff and back, that some inefficiency is quite acceptable. Basically we must avoid degrading the efficiency of METAN1 significantly when the type is stiff, while providing it with the efficiency of the explicit formula when the type is nonstiff.

It is becoming common for ODE codes to provide the user with assistance in the selection of the initial step size. There are actually two goals. One is to select a step size which reveals to the code the scale of the problem, and the other is to provide an efficient step size. The first is by far the more important, and the more difficult, especially for stiff problems. In our investigation we see how to restrict the user's guess for a suitable step size so as to be confident that the scale of the problem is revealed. Deuflhard's step size and order selection algorithm is then able to proceed efficiently.

A recognized weakness of extrapolation methods is that they produce answers infrequently as compared to alternative methods. This is because a relatively large amount of work is done at each step, so that the step size must be large for the sake of efficiency. In our opinion the matter is much more troublesome for nonstiff problems than for stiff. A by-product of our ideas is a reduction of the costs induced by frequent, specific output points.

2. Changes to METAN1—a new norm. Our intention was to alter METAN1 as little as possible so that the effect of our ideas would be clear. Some obvious changes were necessary. A comparatively subtle matter concerns the error control. METAN1

has a scalar relative error control in which the error in each solution component is measured relative to the largest magnitude observed for that component so far in the integration. To use METAN1, one calls the code with initial values at $x = T$ and asks for an answer (output) at $x =$ TEND. A flag can be set if one wants to print out the answers computed at all the steps on the way from T to TEND. For reasons we take up in § 7, this is the most efficient way to monitor the solution. If one does not realize this (the prologue of this *research* code does not point this out), or if one wants output at *specific* arguments, it is natural to repeatedly call METAN1 with TEND taken to be the desired output points. METAN1 restarts itself at each call. This is not particularly inefficient, because a one-step method is implemented, but it has the serious consequence of reinitializing the weights of the error control. Because we intended to ask the code to produce answers at specific points, we added a logical variable to indicate the first call and so avoid reinitializing the weights on subsequent calls.

One important change was made: a different norm was used. Only a couple of norms are at all popular in ODE codes. One is the max norm, $\|v\|_\infty$. (All the norms used weight the individual components, but we suppress these weights to avoid notational complication.) In [2, p. 23] Deuflhard objects to this norm because using it "... may sometimes cause a zigzagging behavior when the index of the maximum component of the vector varies fast along the trajectory." For this reason some authors use the Euclidean norm, $\|v\|_2$, and others use the constant multiple called the *root mean square norm*:

$$\|v\|_{RMS} = \left(\frac{1}{N}\sum_{i=1}^{N} v_i^2\right)^{1/2}.$$

These two possibilities do yield smoother behavior, but they are not completely satisfactory either. One difficulty is that not infrequently they lead to underflow when solving stiff ODE problems, and this can be a nuisance on some computing systems. Also, when applied to an error test, $\|\text{error}\| \leqq \varepsilon$, they require only an average of the error components to be less than the tolerance ε. A lot of small errors allows a relatively large error in some solution component. In this sense the max norm gives a more uniform behavior.

Our developments require matrix norms, too. It is most natural to use the subordinate matrix norm. This is practical with $\|v\|_\infty$, but not with $\|v\|_2$. In the latter case the compatible Frobenius matrix norm could be used, but the fact that it can be much larger than $\|M\|_2$ is a disadvantage in our application.

An interesting possibility here is the norm $\|v\|_1$ or, as we prefer, the constant multiple we shall call the *mean magnitude norm*:

$$\|v\|_{MM} = \frac{1}{N}\sum_{i=1}^{N} |v_i|.$$

It avoids the scaling problems of the Euclidean norm and the lack of smoothness of the max norm. The subordinate matrix norm,

$$\|M\|_1 = \max_j \sum_{i=1}^{N} |M_{ij}|,$$

is readily computed and is particularly suited to the task at hand: The matrices of METAN1 which interest us are Jacobian matrices of the function f of (1). They are formed by differences a column at a time, so that it is convenient to compute the norm as the matrix is constructed.

The biggest disadvantage of the mean magnitude norm is that it is even more liberal in the error control than is the RMS norm. This must be realized in selecting error tolerances. Presumably users could get accustomed to it just as they have with RMS, but we confess that for nonstiff problems we prefer the uniformity of the max norm. Anyway, we needed to change the RMS norm of METAN1 and decided to try the mean magnitude norm. It proved convenient to reprogram the Jacobian evaluation as a subroutine and to compute the norm as the matrix was constructed.

The code resulting from the changes described was called METAN2. The only significant change from METAN1 is the change of norm to accommodate the computation of norms of Jacobians. Because this change means that results from the modified code cannot be compared with published results for METAN1, we gave the code a different name for this paper to avoid confusion. The effects due to our ideas for recognizing stiffness are all demonstrated by comparison to METAN2.

3. Starting step size. The starting step size plays several roles in any ODE solver. A crucial one for robustness is that it indicate to the code the scale of the problem. The fundamental algorithms all presume that the step size varies relatively slowly so that if the initial step size is much too large, completely erroneous results might be accepted. Because the step size is allowed to change by only a limited amount at each step, an initial step size much too small can also impair the efficiency of integration.

METAN1 asks the user to provide the initial step size. Here we propose to reduce this step size, if necessary, to provide assurance that the code begins on scale. Our proposal is not intended to provide an *efficient* choice, only to enhance the code's reliability. Deuflhard's algorithm for step size and order selection [2] differs on the first step from subsequent ones. On the first step it monitors the computation rather more carefully, with the consequence that the efficiency of the code is insensitive to the initial guess, provided that it is on scale.

Our idea is very simple. We insist that the first step be small enough to be regarded as nonstiff, hence resolving the change in any integral curve. We computed the stability regions of the various formulas of METAN1 when the approximate Jacobian is taken to be zero. The Jacobian J is always formed at the initial point and its (weighted) norm computed. For later use we state that the explicit method of k extrapolations is stable, provided that $|H|\,\|J\|_1 \leqq C_k$, where $C_k = 3, 4, 5, 6, 8, 10$, respectively.

On the first step we do not know how many extrapolations will be done and, anyway, we want to be conservative. For this reason we shorten the initial guess as necessary so that $|H|\,\|J\|_1 \leqq 3$. This test on the step size is convenient in this kind of ODE solver and provides considerable safety. We note that Enright et al. provide initial step sizes of this general size for their test set [3] to get codes on scale, but they do not suggest an automatic way to achieve it.

One difficulty in the evaluation of ODE solvers is the provision of an initial step size. In all our tests the initial guess was taken to be the distance from initial point to the first output point. This was subsequently shortened as needed by our type-insensitive code METAN3 in the way described. For consistency in testing, METAN2 was provided the initial step size selected by METAN3. This removed one source of variability in the testing but, of course, removed the effects of our addition. In view of the facts that our test is practically free and that the kind of restriction proposed is generally accepted, we provide here no numerical evidence of its utility.

4. Test sets. For the development of our ideas and for the illustration here of the results, we need some test problems. There is a widely used set of nonstiff test problems [5] and a corresponding set of stiff test problems [3]. In [7] we have pointed

out certain difficulties with the latter set. Here these difficulties are mostly unobtrusive. A crucial point is that some kind of relative error control must be used. As noted, METAN1 has available only a relative error control. Another important point is that some of the problems in the stiff test set are not very stiff. In our use of the problems this does not matter, because we aggregate the sets and expect the code to do well regardless of the type.

There is one genuine difficulty. As the description of the semi-implicit midpoint rule makes clear, linear problems with constant Jacobian are peculiarly related to the method. Unfortunately, quite a few of the test problems of [3] and [5] are of this kind. It is even worse. All but one of these problems is homogeneous, and the sole exception has a constant forcing function. We have separated them out so as not to distort our understanding of the numerical results. Thus the set NSCJ consists of the linear problems with a constant Jacobian from the nonstiff test set [5], namely A1, B2, C1, C2, C3, C4. The set NS consists of the remaining problems from this nonstiff test set.

There was a complication when working with the test set of stiff problems [3]. Namely, the codes would not solve E4 for all the tolerances considered. We simply dropped it from our comparisons. Thus, the set SCJ consists of the linear problems with a constant Jacobian from the stiff test set [3], namely A1, A2, A3, A4, B1, B2, B3, B4, B5. The set S consists of the remaining problems from this test set except for E4.

5. Recognizing stiffness. The principal reason for introducing the Jacobian and the solution of linear systems into the midpoint rule is to stabilize the integration. If the explicit midpoint rule is stable at the desired step size H, it is usually to our advantage to use it, thereby avoiding the expensive formation of the approximate Jacobian J and the expensive linear algebra. We say "usually" because the Jacobian does affect the accuracy of the formula, so that it is not clear just when one formula is to be preferred.

It is crucial that steps taken with the explicit midpoint rule be stable. This is because the error estimation procedure must *recognize* when the smoothness of the solution permits a step size much larger than the stability of the explicit method would permit. Our approach is very simple. If the H predicted by the code as suitable for the current step to be taken with k extrapolations satisfies

$$|H|\,\|J\|_1 \leqq C_k, \tag{2}$$

where C_k is the constant given in § 3 as guaranteeing stability, we use the explicit midpoint rule, and otherwise, the semi-implicit midpoint rule. This is quite conservative, because $\|J\|_1$ may be a lot larger than the eigenvalues of J; hence the restriction on the step size is much more severe than that of the usual stability theory. Being conservative here is not inappropriate because of the limitations of the usual stability theory and because using the Jacobian is not as inefficient as one might think.

It is worth remarking that if convergence is not obtained at k extrapolations, Deuflhard's order selection scheme will consider $k+1$ extrapolations. The resulting formula is more stable than that of k extrapolations, so that we can still reliably estimate the next step size. It might seem that the step prediction would be very poor on changing from the use of a Jacobian $J \neq 0$ in one step to $J = 0$ in the next, or vice versa. In point of fact, this is little different from the normal use of the code for stiff problems. The Jacobian is evaluated at every step and hence normally changes. We

use $J = 0$ only when this approximate Jacobian should give results very similar to the ones obtained with the approximation normally used by the semi-implicit method.

Having said how we decide which formula to use, we need only specify when we evaluate an approximate Jacobian. This is an essential matter, for we do not want to evaluate it any more than necessary. In particular, if the whole problem is clearly nonstiff, i.e., the explicit rule is appropriate, we want to reduce these evaluations to an absolute minimum. As explained in § 4, we choose to evaluate the Jacobian at the initial point and reduce H as necessary to force the explicit rule to be used for the first step. We consider this desirable even for a nonstiff problem.

METAN1 stores a copy of the Jacobian. This fact is of obvious value for the extrapolation process, but it also implies that it is never necessary to form the Jacobian more than once at the same point. In METAN1 a scaled Jacobian is stored. We had to alter this because the scaling changes from step to step.

Suppose we have an approximate Jacobian evaluated at TJAC, we are at T and we wish to step to $T+H$. If TJAC $= T$, there is nothing to decide, so suppose TJAC $\neq T$. The first case is that we just attempted a step from T (necessarily with the explicit formula), and the step failed, so that we are trying again with a reduced step size. Failed steps are unusual with Deuflhard's step and order selection scheme, so they indicate that the character of the problem might have changed from that seen in the preceding step. We choose to form a Jacobian at T then, so that our decision about the formula is based on current information. Multiple failures are possible, but, of course, no more than one Jacobian evaluation will be made.

If the rest (2) using the norm of an old Jacobian says that we should switch to the semi-implicit formula, we form a new Jacobian and repeat the test. If the test (2) based on current data says to switch, we then have available the Jacobian required by the semi-implicit formula.

We do not want to evaluate the Jacobian any more often than necessary, but from time to time we must do it to account for the evolution of the character of the problem. The "classical situation" is that one solves a problem with Lipschitz constant L on an interval $[a, b]$ such that $L|b-a|$ is not "large". Such problems are not stiff. We approximate L by $\|J\|_1$. If we get no other signal, as described above, to form a new Jacobian, we shall do so whenever $|(T+H)-\text{TJAC}| \cdot \|J\|_1 > 50$. The idea is that based on an approximation to L at TJAC, we declare an interval of appropriate length to be nonstiff, unless we are later contradicted by, say, a step failure. If we should want to leave the interval either because we have integrated all of it as a nonstiff problem or because the smoothness of the solution suggests that a "large" step is possible, we form a new Jacobian and define a new interval of nonstiffness.

Notice that if the problem is nonstiff and difficult to solve because of, say, a stringent tolerance, we could do many steps without forming a Jacobian. Indeed, if the problem is clearly nonstiff, the Jacobian formed at the initial point will be the only one ever formed, and the semi-implicit method with its high linear algebra costs will never be used. The conservative nature of our algorithm does mean that Jacobians are formed when it is not necessary to switch, and that the semi-implicit method is used when it is not necessary for stability. This is a price we pay to ensure that stiff problems are solved efficiently.

In all our numerical examples, the tolerance EPS is the scalar relative error criterion of METAN1 in the (weighted) L_1 norm as described in § 2. The number of times the function f of (1) was evaluated is NFEV. All Jacobians are formed by numerical differentiation. This is the standard option in METAN1 and seems appropriate to a type-insensitive code. The number of matrix decompositions for the Gaussian

elimination is NDEC. In these codes no account is taken of the structure of the matrices. The number of linear systems solved is NSOL. The standard for judging our ideas is METAN2, which is essentially METAN1 with a different norm, as explained in § 2, and a special choice of starting step size, as described in § 3. The code with automatic selection of semi-implicit and explicit methods is called METAN3.

First we present in Table 1 the results for both stiff and nonstiff test sets, without the problem of the constant Jacobian and of E4 of the stiff set, i.e., sets NS and S

TABLE 1
Solution of test sets S *and* NS *with one output point.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	11,688	1,573	11,172	9,442	555	3,967
10^{-6}	25,100	2,340	25,265	22,349	587	5,784
10^{-9}	51,954	3,789	52,911	50,791	1,264	15,512

of § 4. Here we have asked for an answer only at the end of the specified interval of integration. In § 6 we shall take up the effect of additional output. The results show that we have had considerable success at reducing the linear algebra costs. In addition, the reduced number of Jacobians formed in METAN3 has somewhat reduced the total number of function evaluations.

The results of Table 1 illustrate what we conceive to be normal use of a type-insensitive code. Naturally compromises were made. The effects of these compromises are clearer in the results of Tables 2 and 3. Table 2 shows that METAN2 is doing a great deal of linear algebra that is not really necessary. We have reduced it very substantially in METAN3, but we have been conservative enough that we still must do a significant number of solutions of linear systems. It is necessary to be conservative in order to solve the stiff problems efficiently, as displayed in Table 3. Notice that

TABLE 2
Solution of test set NS *with one output point.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	9,717	1,292	9,412	7,195	316	2,460
10^{-6}	20,201	1,807	20,457	16,427	178	2,177
10^{-9}	36,152	2,484	36,963	32,238	123	1,905

TABLE 3
Solution of test set S *with one output point.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	1,971	281	1,760	2,247	239	1,507
10^{-6}	4,899	533	4,808	5,922	409	3,607
10^{-9}	15,802	1,305	15,948	18,553	1,141	13,607

METAN3 is doing less linear algebra than METAN2 but is making more function evaluations. The work has been balanced pretty well for this particular set of test problems, because many of the problems have function evaluations no more expensive than the solution of a linear system.

Table 4 displays the results of the problems peculiarly correlated to the semi-implicit midpoint rule. Most of them are from the stiff test set which also favors the semi-implicit midpoint rule. Despite the special nature of these problems, METAN3 gives comparable results. It does more function evaluations but less linear algebra. This performance seems quite acceptable.

TABLE 4
Solution of test sets SCJ *and* NSCJ *with one output point.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	2,744	312	2,140	3,082	273	1,891
10^{-6}	7,854	663	7,273	8,947	514	5,522
10^{-9}	18,506	1,179	18,017	21,732	758	11,106

6. Stability. In the codes known to us which extrapolate the explicit midpoint rule, there is the possibility of overflow due to instability. If the various subintegrations with the midpoint rule succeed in reaching $x_n + H$, the convergence monitors in use respond to stability difficulties. However, it is possible that a subintegration will not reach $x_n + H$ if the step size is considerably too large. Years ago we observed this in tests [9], but to our knowledge, no special consideration has been given the matter in codes. The ideas of Hussels [6] and Deuflhard [2], in effect, adjust the step size cheaply so as to get on scale at the first step. In conjunction with their convergence monitors on subsequent steps, overflows due to instability are likely to be rare.

In the present context, difficulties with stability are quite possible. In the development of the ideas of § 5, we naturally experimented with many variations. On several occasions we experienced overflow, even though the computers used have a very large exponent range. In the final version presented in § 5 we are certain that no instability can occur. However, in this work we made an observation which is worth noting.

In their report [1, p. 24] Bader and Deuflhard describe a device for recognizing a nearly singular linear system arising in the use of the semi-implicit midpoint rule. It may be useful for this purpose, but it has another use as well. It tests whether $\|\eta_{k+1} - \eta_k\| > 10$ for each step of each subintegration. The norm of the code is a relative one, so the test here is whether solution components grow more than an order of magnitude in a single step of the subintegration. If they are growing too fast, the step size is reduced. This is quite a reasonable way to recognize and respond to instability. The overflows we experienced came about because we implemented the explicit midpoint rule as a separate computation. When we introduced the equivalent test on the growth of solution components, overflow was prevented and the step size was reduced as needed to secure stability.

7. Output. A characteristic of extrapolation methods is that they do a considerable amount of work before producing a result. As a consequence they must take "large" steps to be efficient. When the step size is restricted, as for example by stability, lack of smoothness, or output, such codes may become inefficient. Deuflhard's step

and order selection algorithm reduces the order, hence cost, when the step size is restricted. Generally speaking, it is very helpful at controlling the effects of step size restrictions, but there is no doubt that users should try to get along with results produced where the code finds it convenient, rather than impose their will. In our experience, this is acceptable for stiff problems, because solutions often change radically, and users do not expect to follow these changes in great detail. On the other hand, users often want to follow the solutions of nonstiff problems in great detail and their output requirements may seriously impact extrapolation codes.

Restrictions due to output are especially serious for the solution of nonstiff problems with the semi-implicit midpoint rule, because a new Jacobian is formed at every step. The algorithm we presented in § 5 helps with this difficulty. Reducing the step size makes it more likely that the explicit formula will be used. The algorithm for deciding when to form a Jacobian is not related to the number of steps (or output points), so the number of Jacobians formed is insensitive to the number of output points. Indeed this number is likely to decrease as the number of output points is increased, because ambiguous decisions requiring a Jacobian evaluation may no longer be ambiguous. There is the possibility of improved performance for stiff problems in this situation, but there is no reason to expect dramatic changes.

We computed results like those of § 5 with ten equally spaced output points. For the nonstiff problems, this is modest output, and the cost was not greatly affected. METAN3 considerably reduced its linear algebra as ambiguous cases became clearer. This is a highly desirable result in this situation, which we regard as more typical for nonstiff problems than for the one output point problems of § 5. The cost of solving the stiff problems went up significantly as the possible large step sizes were restricted. The relative behavior of METAN2 and METAN3 with stiff and nonstiff problems aggregated was similar to the case of one output point, already presented in § 5.

To show the effects of output strongly, we asked for 100 equally spaced output points. The results for stiff and nonstiff problems are displayed in Tables 5 and 6. Comparison of Table 6 to Table 3 shows that frequent output when solving stiff

TABLE 5
Solution of test set NS *with* 100 *output points.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	27,938	4,170	22,533	17,877	5	59
10^{-6}	46,817	5,929	43,186	32,925	66	647
10^{-9}	67,682	7,324	65,436	58,293	63	885

TABLE 6
Solution of test set S *with* 100 *output points.*

	METAN2			METAN3		
EPS	NFEV	NDEC	NSOL	NFEV	NDEC	NSOL
10^{-3}	19,598	3,148	15,857	19,614	2,904	14,628
10^{-6}	25,999	3,760	22,833	26,621	3,428	20,508
10^{-9}	41,848	4,981	39,832	43,381	4,469	34,843

problems is very harmful. The software ought to recognize this and inform the user, much as we did with RFK45 [8, p. 109]. Even for nonstiff problems, there is a serious effect due to the output. Our ideas for controlling the linear algebra and the formation of Jacobians obviously do very well, but the software ought to recognize and report the serious impact. One hundred output points is a lot for the nonstiff problems, but it is by no means uncommon. Users should be directed by the software to codes based on, say, backward differentiation or Adams formulas if they must have so much output at *specific* points.

8. Conclusion. Our investigation has shown that it is possible to base a type-insensitive ODE solver on extrapolation of the explicit and semi-implicit midpoint rules. It is a great convenience for a user to be able to ignore the question of stiffness. Also, when the solver is a module in an applications package, it may not be convenient for a user to supply information about stiffness. For these reasons some inefficiency is acceptable in a type-insensitive code. Nevertheless, the developments presented in this paper show that a type-insensitive code can be quite effective, even compared to excellent codes intended for a specified type. It also proved possible to increase robustness by helping the codes get on scale reliably. A by-product of the algorithms for achieving insensitivity to type is a reduction of the sensitivity of the extrapolated semi-implicit midpoint rule to frequent output.

9. Acknowledgment. The author acknowledges with gratitude the computing support provided by L. S. Baca and her valuable comments which influenced the evolution of the approach to type-insensitivity presented here.

REFERENCES

[1] G. BADER AND P. DEUFLHARD, *A semi-implicit mid-point rule for stiff systems of ordinary differential equations*, University of Heidelberg, SFB 123, Rept. 114, 1981.

[2] P. DEUFLHARD, *Order and stepsize control in extrapolation methods*, University of Heidelberg, SFB 123, Rept. 93, 1980.

[3] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of* O.D.E.'*s*, BIT, 15 (1975), pp. 10–48.

[4] W. B. GRAGG, *On extrapolation algorithms for ordinary initial value problems*, SIAM J. Numer. Anal., 2 (1965) pp. 384–404.

[5] T. E. HULL, W. H. ENRIGHT, B. M. FELLEN AND A. E. SEDGWICK, *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal., 9 (1972), pp. 603–637.

[6] H. G. HUSSELS, *Schrittweitensteuerung bei der Integration gewöhnlicher Differentialgleichungen mit Extrapolationsverfahren*, Univ. Köln, Math. Inst. Diplomarbeit (1973).

[7] L. F. SHAMPINE, *Evaluation of a test set for stiff* ODE *solvers*, ACM Trans. Math. Software, 7 (1981), pp. 409–420.

[8] L. F. SHAMPINE AND H. A. WATTS, *The art of writing a Runge-Kutta code*, II, Appl. Math. Comp., 5 (1979), pp. 93–121.

[9] L. F. SHAMPINE, H. A. WATTS AND S. M. DAVENPORT, *Solving nonstiff ordinary differential equations—The state of the art*, SIAM Rev., 18 (1976), pp. 376–411.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0007 $01.25/0

ACCURATE MONOTONICITY PRESERVING CUBIC INTERPOLATION*

JAMES M. HYMAN†

Abstract. A simple and effective algorithm to construct a monotonicity preserving cubic Hermite interpolant for data with rapid variations is presented. Constraining the derivatives of the interpolant according to geometric considerations makes the interpolant consistent with local monotonicity properties of the data. Numerical examples are given that compare the quality and accuracy of the proposed interpolation method with other standard interpolants.

Key words. approximation theory, interpolation, monotonicity, numerical analysis, shape preservation, spline

1. Introduction. Piecewise polynomial interpolation is used to deduce probable values for an implied function defined at a discrete set of points. For an accurate interpolation, we must carefully retain crucial properties of the data (such as monotonicity or convexity), and we must not introduce details or artifacts that cannot be ascertained from the data. This requires that the interpolant be designed according to both geometric and algebraic considerations.

The geometric qualities of an interpolant are based on how well the interpolated curve reflects the intrinsic shape inferred by the data points. A good geometric interpolant will produce a curve similar to the visually pleasing one of a draftsman. Although some mathematical properties, such as those that preserve monotonicity and convexity, can describe the goodness of fit in a geometric sense, choosing the better curve often is a heuristic decision based on human judgment rather than on firm mathematical theory.

A good geometric interpolant is most important when the data arise from a physical experiment and an underlying mathematical structure does not exist. For these data sets, geometric considerations such as preventing spurious behavior near rapid changes in the data may be more important than the method's asymptotic accuracy. In fact, maintaining monotonicity or convexity in the interpolation process may be necessary to represent physical reality. For example, if the data come from an equation-of-state table for density versus pressure, then a nonmonotone interpolant will have a negative derivative and will imply an imaginary sound speed for the material [10]. This error can destroy the accuracy of any calculation based on the interpolated data.

The functional or algebraic properties of a good interpolant in classic approximation theory are defined more precisely [2]. They include the order of accuracy as the mesh spacing becomes arbitrarily small, continuity or smoothness of some derivative of the interpolant, and invariance or linearity properties such as

$$P(\alpha f+g)=\alpha Pf+Pg, \tag{1.1}$$

where P is the interpolation operator, α is a scalar and f and g are functions. Note that none of these properties guarantees a good geometric interpolant.

Another consideration is a method's practicality, as evidenced by its simplicity, efficiency, and storage requirements. In this report, we restrict our analysis to local

* Received by the editors March 26, 1981, and in revised form September 20, 1982.

† Center for Nonlinear Studies, Theoretical Division, MS B284, Los Alamos National Laboratory, Los Alamos, New Mexico 87545.

piecewise polynomial interpolants that, when compared to other methods, rate high in these three categories.

We have developed and tested a practical algorithm that is excellent both geometrically and algebraically and is highly accurate when monotonicity properties of the interpolant do not intervene. Loss of accuracy in favor of shape preservation occurs only at isolated points where the grid is rough compared to the solution variation.

First, we briefly describe the piecewise polynomial cubic Hermite interpolant and the restrictions sufficient to guarantee monotonicity. We then describe some possible algorithms that compute the derivatives needed by the interpolant at the mesh points, and we give numerical examples that compare our method to similar methods.

2. Cubic Hermite interpolation. Let the mesh $\{x_i\}_{i=1}^n$ be a partition $x_1 < x_2 < \cdots < x_n$ of the interval $[x_1, x_n]$, and let $\{f_i\}$, $f_i = f(x_i)$, be the corresponding data points. The local mesh spacing is $\Delta x_{i+1/2} = x_{i+1} - x_i$, and the slope of the piecewise linear interpolant between the data points is $S_{i+1/2} = \Delta f_{i+1/2}/\Delta x_{i+1/2}$. The data are *locally monotone at* x_i if $S_{i+1/2}S_{i-1/2} > 0$. The interpolant is *piecewise monotone* if $(Pf)(x)$ is monotone between f_i and f_{i+1} for x between x_i and x_{i+1}. The interpolant Pf is *class* C^k if $(Pf)(x)$ is continuous and has continuous derivatives for all orders less than or equal to k.

A. The interpolation formula. Given the data points $\{f_i\}$, a numerical approximation of the slope $\dot{f}_i$ at x_i is calculated for $1 \leqq i \leqq n$. The cubic Hermite interpolant then is defined for $1 \leqq i < n$ as

$$P(x) = c_1 + (x - x_i)c_2 + (x - x_i)^2 c_3 + (x - x_i)^3 c_4, \tag{2.1}$$

where $x_i \leqq x \leqq x_{i+1}$,

$$c_1 = f_i, \qquad c_2 = \dot{f}_i,$$

$$c_3 = \frac{3S_{i+1/2} - \dot{f}_{i+1} - 2\dot{f}_i}{\Delta x_{i+1/2}}, \qquad c_4 = -\frac{2S_{i+1/2} - \dot{f}_{i+1} - \dot{f}_i}{\Delta x_{i+1/2}^2}.$$

The interpolant (2.1) has a continuous first derivative, $p(x) \in C^1$, and possibly, but not necessarily, a continuous second derivative. The continuity of the second derivative and the order of accuracy depend on how $\{\dot{f}_i\}$ are calculated.

Note that once $\{\dot{f}_i\}$ are given, (2.1) becomes a local interpolation formula. By changing the value of f_i or $\dot{f}_i$ at a data point, the interpolant changes only in the region $[x_{i-1}, x_{i+1}]$. If the calculation of $\dot{f}$ also is local, only nearby data points need be available when interpolating between x_i and x_{i+1}. This localness is important when storage requirements are critical as is the case for very large data sets or multidimensional interpolation.

Localness of the interpolant also is desirable when data are being readjusted a few points at a time. This occurs in interactive graphics routines to avoid recalculating the interpolation function at all data points.

The numerical approximation of $\{\dot{f}_i\}$ which makes (2.1) a C^2 interpolant (for example, the complete spline interpolant[2]), is not local. Thus, to gain total localness for (2.1), we must sacrifice global continuity in the second derivative.

B. Monotonicity. Even when $\{\dot{f}_i\}$ are defined accurately, additional constraints may be necessary because (2.1) may fail to produce an acceptable interpolant in the geometric sense for certain data sets. A simple generalization of what was recognized by de Boor and Swartz [3] is that if the data are locally monotonically increasing at

x_i, and if

$$0 \leqq \dot{f}_j \leqq 3 \min (S_{j-1/2}, S_{j+1/2}) \tag{2.2}$$

for $j = i$ or $i+1$, then the resulting interpolant is monotone in $[x_i, x_{i+1}]$. Fritsch and Carlson [5] independently found an extension of this criteria giving a *necessary* and sufficient condition for (2.1) to be monotone. The de Boor–Swartz criterion is a square inscribed within the Fritsch–Carlson monotonicity region.

Note that if $\{\dot{f}_i\}$ are calculated to make the resulting interpolant C^2, then (2.2) may not be satisfied. That is, there are monotone data sets for which there is no C^2 piecewise cubic Hermite interpolant.

When the data are locally monotone, we restrict $\{\dot{f}_i\}$ to the de Boor–Swartz piecewise monotonicity range of (2.2) as follows. After calculating an accurate approximation of $\dot{f}_i$ (for instance, finite differences or by the complete spline formula), we project it to the allowed monotonicity region according to

$$\dot{f}_i \leftarrow \begin{cases} \min [\max (0, \dot{f}_i), 3S^i_{\min}] & \text{if } 0 < S^i_{\min}, \\ \max [\min (0, \dot{f}_i), 3S^i_{\max}] & \text{if } 0 > S^i_{\max}, \\ 0 & \text{if } 0 \geqq S_{i-1/2} S_{i+1/2}, \end{cases} \tag{2.3}$$

where

$$S^i_{\min} = \min (S_{i-1/2}, S_{i+1/2}), \qquad S^i_{\max} = \max (S_{i-1/2}, S_{i+1/2}).$$

Near the boundary, the de Boor–Swartz constraint can be used by letting $S_{-1/2} = S_{1/2}$ and $S_{n+1/2} = S_{n-1/2}$.

When an $\dot{f}_i$ associated with a complete spline interpolant falls outside the range of (2.2), as it inevitably will when the variation between the data points is large, resetting $\dot{f}_i$ according to (2.3) will cause the second derivative of the interpolant to jump where $\dot{f}_i$ was reset and at the two nearest mesh points.

If the underlying function is strictly monotone and sufficiently smooth, and $\dot{f}_i$ is an accurate approximation to the derivative at x_i, then as the mesh is refined, (2.2) will be satisfied in the limit, because

$$\left.\frac{df}{dx}\right|_{x=x_i} = \dot{f}_i + O(\Delta x^?) = S_{i+1/2} + O(\Delta x) = S_{i-1/2} + O(\Delta x). \tag{2.4}$$

Thus, the interpolant is restricted by geometric considerations only when the mesh is coarse and the asymptotic accuracy in $\dot{f}$ is meaningless. When the mesh accurately resolves the function implied by the data, the accuracy in $\dot{f}$ is retained because (2.2) will be satisfied.

When the data are not locally monotone, the interpolant also must have an extrema. Retaining piecewise monotonicity would require that $\dot{f}_i = 0$ and would "clip" the interpolant by forcing inter-interval monotonicity on nonmonotone data. However, the piecewise monotonicity constraint can be relaxed in the interval pair next to the extrema to produce (in the author's opinion) a more visually pleasing curve. But if a new constraint is imposed at extrema, the change in decision algorithms must still produce a stable interpolant. That is, a small change in the data should not create a large change in the interpolant. If we remove all constraints on the interpolant near locally nonmonotone data while retaining (2.3) elsewhere, the resulting interpolant will be unstable.

We chose to extend (2.2) by requiring that $\dot{f}_i$ have the same sign as originally calculated, and that

$$|\dot{f}_i| \leqq 3 \min (|S_{i-1/2}|, |S_{i+1/2}|). \tag{2.5}$$

The constraining function extending (2.3) is

$$\dot{f}_i \leftarrow \begin{cases} \min [\max (0, \dot{f}_i), 3 \min (|S_{i-1/2}|, |S_{i+1/2}|)], & \sigma > 0, \\ \max [\min (0, \dot{f}_i), -3 \min (|S_{i-1/2}|, |S_{i+1/2}|)], & \sigma < 0, \end{cases} \tag{2.6}$$

where $\sigma = \text{sign}\,(\dot{f}_i)$. The sign function sign $(S) = 1$ if $S \geqq 0$ and -1 otherwise.

Often the monotonicity of the interpolant's derivative is an important quality that can be incorporated into the interpolant; if the data are convex, a good geometric interpolant should preserve this convexity. However, a C^1 convexity preserving cubic Hermite interpolant does not exist for all data sets [11]. For example, a C^1 convex interpolant does not exist for $f = x + |x|$ when $x = 0$ is a data point. If the C^1 constraint is dropped, restrictions similar to (2.6) can be incorporated in (2.1) to preserve convexity [7].

A simple, necessary but not sufficient, and often effective convexity preserving constraint involves limiting the $\{\dot{f}_i\}$ so that

$$\min (S_{i-1/2}, S_{i+1/2}) \leqq \dot{f}_i \leqq \max (S_{i-1/2}, S_{i+1/2}) \tag{2.7}$$

by using

$$\dot{f}_i \leftarrow \max \{\min [\dot{f}_i, \max (S_{i-1/2}, S_{i+1/2})], \min (S_{i-1/2}, S_{i+1/2})\}. \tag{2.8}$$

3. Derivative approximation. The order of accuracy of (2.1) can be, at best, one order higher than the order of accuracy of $\dot{f}_i$. Therefore, it is prudent to calculate $\dot{f}_i$ accurately whenever possible. The difference approximations can be divided into two classes; local and nonlocal. The local schemes use only f values near x_i to calculate $\dot{f}_i$. The nonlocal schemes use all $\{f_i\}$ values and obtain $\{\dot{f}_i\}$ by solving a linear system of equations.

A. Local methods. De Boor and Swartz have shown that there are no linear algorithms yielding derivative approximations above first-order that also automatically satisfy (2.2). There are, however, many nonlinear formulas that do. The Butland [1] algorithm, for example, yields $\{\dot{f}_i\}$ which automatically satisfies (2.2) and is second-order on a uniform grid. The Fritsch–Butland [4] algorithm listed in Table 1 is a slight modification of this formula.

The parabolic interpolation method in Table 1 is linear, and the resulting $\{\dot{f}_i\}$ do not automatically satisfy (2.2). This formula can be multiplied by a nonlinear factor with magnitude $1 + O(\Delta x^2)$ to give the monotonicity preserving formula

$$\dot{f}_i = \frac{(2+\theta) S_{i+1/2} S_{i-1/2}}{S_{i+1/2}^2 + S_{i-1/2}^2 + \theta S_{i+1/2} S_{i-1/2}} \cdot \frac{\Delta x_{i-1/2} S_{i+1/2} + \Delta x_{i+1/2} S_{i-1/2}}{\Delta x_{i-1/2} + \Delta x_{i-1/2}}. \tag{3.1}$$

This formula is second-order for monotone data when

$$-2 < \theta \leqq 1 + 3 \min \left(\frac{\Delta x_{i+1/2}}{\Delta x_{i-1/2}}, \frac{\Delta x_{i-1/2}}{\Delta x_{i+1/2}} \right).$$

The tests for the resulting interpolant with $\theta = 1$ are not included in this report, but they are very similar to and only slightly less accurate than the monotonicity constrained parabolic interpolant.

TABLE 1
Local formulas for $\dot{f}_i$ and their order of accuracy for smooth functions and mesh variations.

Method	Formula for $\dot{f}_i$	Order of accuracy
Akima[a]	$\dfrac{\lvert S_{i+3/2}-S_{i+1/2}\rvert S_{i-1/2}+\lvert S_{i-1/2}-S_{i-3/2}\rvert S_{i+1/2}}{\lvert S_{i+3/2}-S_{i+1/2}\rvert+\lvert S_{i-1/2}-S_{i-3/2}\rvert}$	$O(\Delta x^2)$
Fritsch–Butland[b]	$\dfrac{3S^i_{\min}S^i_{\max}}{S^i_{\max}+2S^i_{\min}}$	$O(\Delta x)$
Parabolic[c]	$\dfrac{\Delta x_{i-1/2}S_{i+1/2}+\Delta x_{i+1/2}S_{i-1/2}}{x_{i+1}-x_{i-1}}$	$O(\Delta x^2)$
Fourth-order finite difference[d]	$\dfrac{-f_{i+2}+8f_{i+1}-8f_{i-1}+f_{i-2}}{-x_{i+2}+8x_{i+1}-8x_{i-1}+x_{i-2}}$	$O(\Delta x^4)$

[a] See [2]. [b] See [4]. [c] See [9]. [d] See [9].

The fourth-order finite difference method [9] in Table 1 is based on first mapping $\{x_i\}$ to a standard equally spaced reference grid $\{r_i\}$, and then approximating each term in the identity

$$\frac{df}{dx}=\frac{df}{dr}\left(\frac{dx}{dr}\right)^{-1}, \tag{3.2}$$

with centered fourth-order finite differences for an equally spaced grid. If the mapping from x into r is not sufficiently smooth (that is, certain high derivatives of the map are not bounded independent of n), the order of accuracy of the method will be reduced accordingly. Thus, the finite difference formula is fourth-order on a smoothly varying mesh, but only first-order on a rougher mesh and, in fact, could become singular on a mesh having a local mesh ratio greater than 3.5. When this is the case, the parabolic method is to be preferred.

The Akima [2] and Fritsch–Butland formulas are nonlinear algorithms for each $\dot{f}_i$. Consequently, the sum of the interpolants for data sets (x, f) and (x, g) is different from the interpolant for the sum of the data sets $(x, f+z)$. The other formulas do not have this defect in their initial approximations of $\dot{f}_i$. However, after filtering according to (2.6), none of the interpolants are additive in the sense of (1.1) for $f=x+|x|$, $g=x-|x|$ when $x=0$ is a data point.

B. Nonlocal methods. The most common nonlocal method for computing $\dot{f}_i$ is the "not-a-knot" C^2 spline interpolation method [2], where $\dot{f}_i$ is $O(\Delta x^3)$ on an unequally spaced mesh or $O(\Delta x^4)$ away from the boundary on an equally spaced mesh. The $\dot{f}_i$'s are calculated so that the resulting interpolant has a continuous second derivative at the knots. As mentioned in § 2 the C^2 spline interpolant will not necessarily satisfy (2.2) for monotone data. By allowing isolated discontinuities in the second derivative, a slightly deficient C^1 monotone spline interpolant can be constructed in various ways. A possible solution is to filter the C^2 spline $\{\dot{f}_i\}$ according to (2.6). This is the approach taken in the numerical examples presented here.

An algorithm that keeps the number of jumps in the second derivative small involves first computing $\{\dot{f}_i\}$ for the complete spline interpolant in the interval $[x_1, x_n]$. If the interpolant is not locally monotone, we locate point x_j where $\dot{f}_j$ is farthest outside the monotonicity region. We redefine $\dot{f}_j$ according to (2.6) and solve for the

complete spline interpolant in $[x_1, x_j]$ and $[x_j, x_n]$, using f_j and $\dot{f}_i$ as boundary conditions. The resulting interpolant will have a break in the second derivative only at x_j. If none of the resulting $\{\dot{f}_i\}$ violate (2.6), we are finished. If some do violate (2.6), we repeat the process, break $[x_1, x_j]$ or $[x_j, x_n]$ into smaller subregions and continue. This algorithm will always terminate if $\dot{f}_1$ and $\dot{f}_n$ are given at the boundaries and satisfy (2.6).

Another nonlocal approximation to $\dot{f}$ is the Fritsch–Carlson algorithm [5]. This method provides an approximation of $\dot{f}$ that preserves piecewise monotonicity in (2.1) with curves geometrically similar to those produced by the Fritsch–Butland method. Although we have not compared the Fritsch–Carlson method to the other methods in this report, we have included an example of specific data from their paper [5]. From this example and other similar ones from their paper, we have found that the monotonicity constrained algorithms using (2.3) or (2.6) perform very similarly to the Fritsch–Carlson algorithm. The major difference is that the constraints (2.3) and (2.6) are much easier to implement. Also the behavior of the (2.6) constrained interpolant at extrema in nonmonotone data sets is different. The Fritsch–Carlson algorithm clips the interpolant like the constraint (2.3) does.

C. Boundaries. At the boundaries we will use either the not-a-knot option for splines [2] or an uncentered difference approximation. The second-order uncentered parabolic method used with the Akima, Fritsch–Butland, and parabolic algorithms is

$$\dot{f}_i = \frac{(2\,\Delta x_{i+1/2} + \Delta x_{i+3/2})S_{i+1/2} - \Delta x_{i+1/2} S_{i+3/2}}{\Delta x_{i+1/2} + \Delta x_{i+3/2}}$$

or

$$\dot{f}_i = \frac{(2\,\Delta x_{i-1/2} + \Delta x_{i-3/2})S_{i-1/2} - \Delta x_{i-1/2} S_{i-3/2}}{\Delta x_{i-1/2} + \Delta x_{i-3/2}}.$$

The third-order uncentered finite difference approximations used with the fourth-order interior formula are

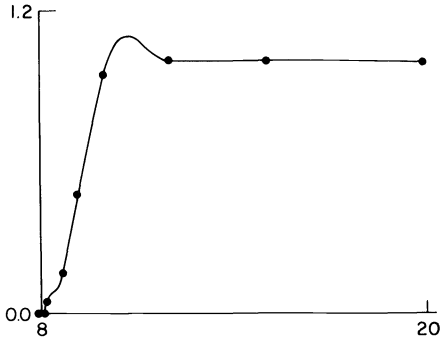
$$\dot{f}_i = \frac{-22f_i + 36f_{i+1} - 18f_{i+2} + 4f_{i+3}}{-22x_i + 36x_{i+1} - 18x_{i+2} + 4x_{i+3}},$$

$$\dot{f}_i = \frac{-2f_{i-1} - 3f_i + 6f_{i+1} - f_{i+2}}{-2x_{i-1} - 3x_i + 6x_{i+1} - x_{i+2}},$$

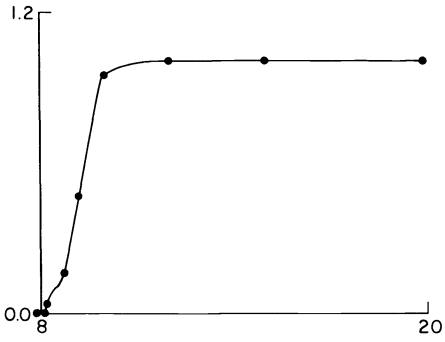
$$\dot{f}_i = \frac{22f_i - 36f_{i-1} + 18f_{i-2} - 4f_{i-3}}{22x_i - 36x_{i-1} + 18x_{i-2} - 4x_{i-3}},$$

$$\dot{f}_i = \frac{2f_{i+1} + 3f_i - 6f_{i-1} + f_{i-2}}{2x_{i+1} + 3x_i - 6x_{i-1} + x_{i-2}}.$$

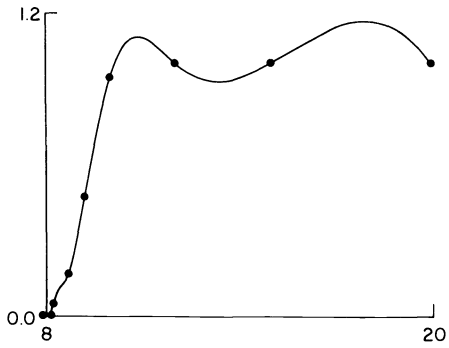
4. Numerical examples. The geometric and accuracy properties of the interpolants are compared on both smoothly varying and rough data sets. When the derivatives are constrained by the extended de Boor–Swartz monotonicity limit (2.6), we call the resulting interpolant monotonically constrained (MC).



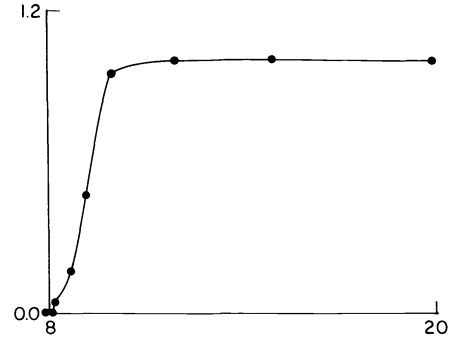
(a) Akima



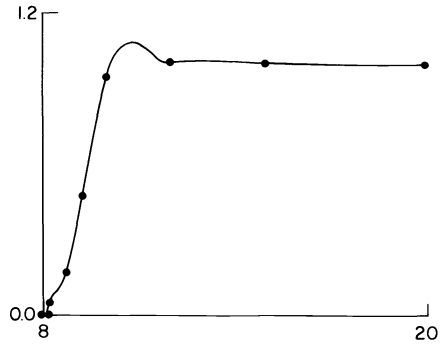
(b) Fritsch–Butland



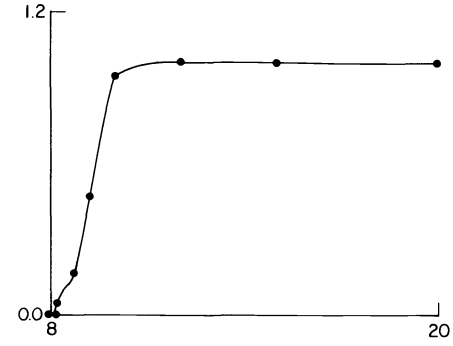
(c) Complete spline



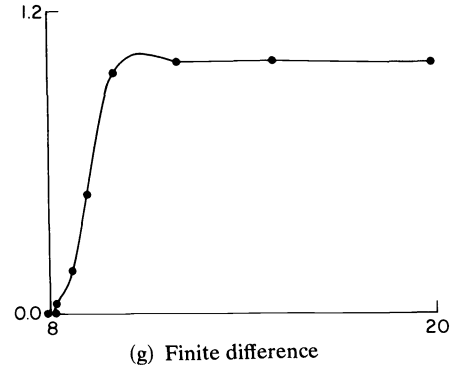
(d) MC spline



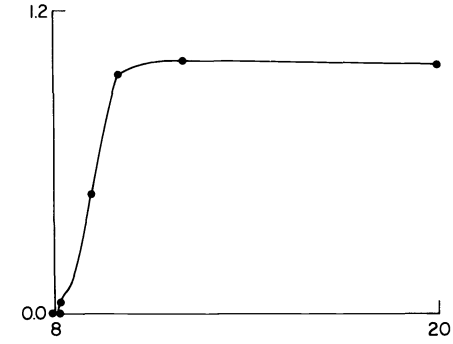
(e) Parabolic



(f) MC parabolic

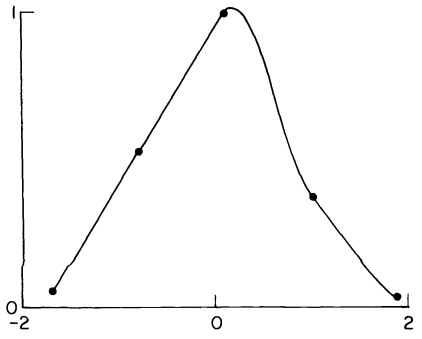


(g) Finite difference

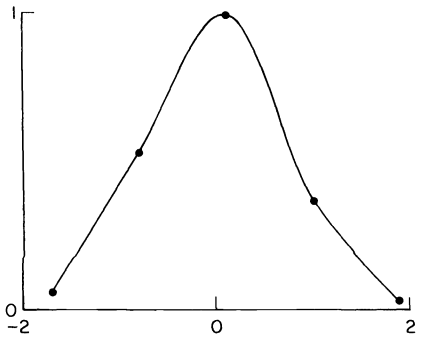


(h) MC finite difference

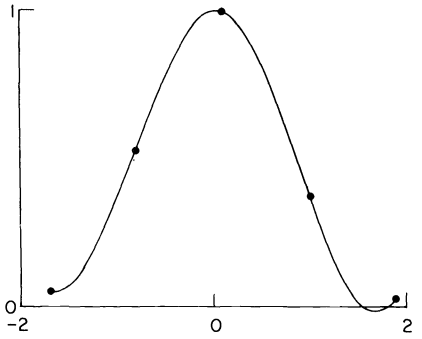
FIG. 1. *Interpolation curves for the* RPN 15A *data.*



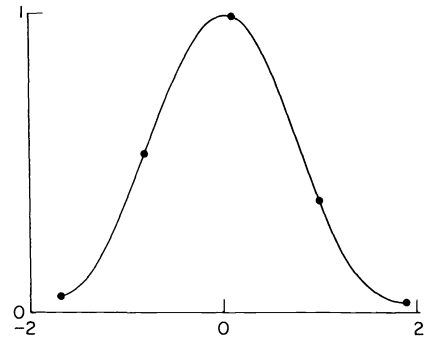
(a) Akima



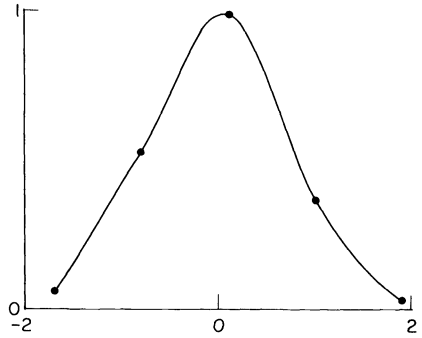
(b) Fritsch–Butland



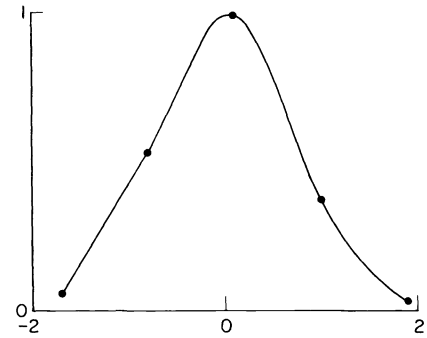
(c) Complete spline



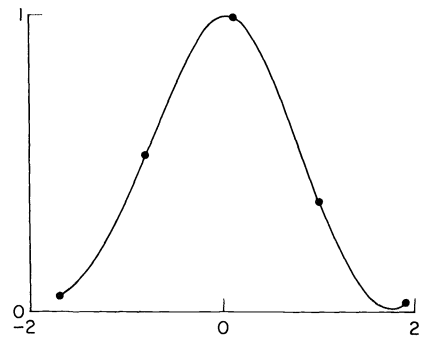
(d) MC spline



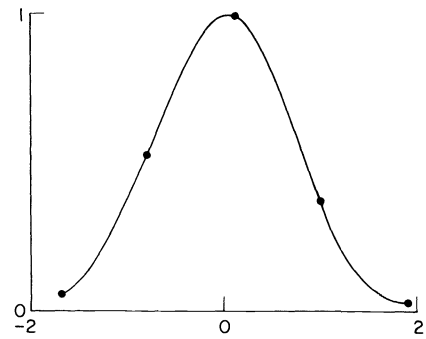
(e) Parabolic



(f) MC parabolic



(g) Finite difference



(h) MC finite difference

FIG. 2. *Interpolation curves for* $f(x) = e^{-x^2}$.

A. Monotone example. The Fritsch–Carlson RPN 15A data have been used to compare many different algorithms [4], [5], [8], some not included in this report. The data points are

x	f
7.99	0
8.09	2.76429E−5
8.19	4.37498E−2
8.7	0.169183
9.2	0.469428
10	0.943740
12	0.998636
15	0.999919
20	0.999994

Figure 1 shows interpolation curves of the Akima, Fritsch–Butland, parabolic, fourth-order finite difference, complete spline, MC parabolic, MC fourth-order finite difference, and MC spline methods. These data show clearly that the Fritsch–Butland and MC methods are geometrically superior to the unconstrained methods. The simple constraint of (2.6) can convert an unacceptable geometric interpolant, such as the complete spline, into an excellent one. Note that the Akima algorithm, which was designed as a good geometric interpolant, fails to preserve monotonicity in this relatively simple example.

B. Nonmonotone example. To interpolate the monotone function $f(x)=e^{-x^2}$, $x\varepsilon[-1.7, 1.9]$, the mesh was equally spaced with $\Delta x = 3.6/(n-1)$. This domain was chosen so the mesh points would not be symmetrical about the point of symmetry for the function.

Figure 2 shows that the higher order MC interpolants can be geometrically superior to the Akima, Fritsch–Butland, parabolic and the unconstrained interpolants when $n = 5$.

TABLE 2
Comparison of the methods for $f(x)=e^{-x^2}$.

Method	L_2 error			
	$n=5$	$n=9$	$n=17$	$n=33$
Akima	6.0E−2	6.4E−3	1.0E−3	1.3E−4
Fritsch–Butland	4.4E−2	7.3E−3	2.4E−3	1.6E−4
Parabolic method	3.9E−2	4.1E−3	4.1E−5	4.3E−5
Finite difference	2.2E−2	3.4E−3	7.4E−5	2.3E−6
Complete spline	3.5E−2	2.0E−3	4.0E−5	1.8E−6
MC parabolic	3.9E−2	4.1E−3	1.9E−3	4.3E−5
MC finite difference	1.5E−2	3.4E−3	1.9E−3	2.3E−6
MC spline	1.7E−2	2.0E−3	1.9E−3	1.8E−6

In Table 2, the errors,

$$L_2 \text{ error} = \left(\int_{-1.7}^{1.9} [(Pf)(x) - e^{-x^2}]^2 \right)^{1/2},$$

of the interpolants are compared as the mesh is refined. Note that the higher order MC finite difference and spline methods are more accurate than the other methods both on the fine and coarse grids.

The MC methods agree with the corresponding unconstrained methods when $n = 9$ and 33, but not when $n = 17$, because the nonmonotonicity of the underlying function is being interpolated.

5. Summary and conclusions. When only geometric considerations are important, any interpolant constrained to stay within the de Boor–Swartz monotonicity limits using algorithm (2.6) is acceptable. The Fritsch–Butland does this automatically (that is, there are no conditional statements) but the approximating derivatives in other procedures must be filtered.

When both geometric and accuracy considerations are important, the lower order methods (Akima, Fritsch–Butland, and parabolic) have larger truncation errors than the higher order constrained methods.

Therefore, we recommend first computing an approximation $\dot{f}_i$ to df/dx at the mesh points, using either the local fourth-order finite difference method (Table 1) or the nonlocal, but smoother, complete spline approximation. Before interpolating using (2.1), we filter $\{\dot{f}_i\}$ with (2.6), so the interpolant will retain the important local monotonicity properties of the data.

The simplicity of the filtering approach and the dramatic improvements in the interpolation curve far outweigh the cost of the extra few lines of code. Analysis of our numerical examples indicates that most cubic Hermite interpolation programs would be more versatile, robust, and often more accurate, if a monotonicity constraint such as (2.6) were an option.

Acknowledgment. I am grateful to Blair Swartz for providing me with much welcome advice in our many discussions during this work.

REFERENCES

[1] J. BUTLAND, *A method of interpolating reasonable-shaped curves through any data*, Proc. Computer Graphics 80, Online Publications Ltd., Northwood Hills, Middlesex, England, 1980, pp. 409–422.

[2] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[3] C. DE BOOR AND B. SWARTZ, *Piecewise monotone interpolation*, J. Approx. Theory, 21 (1977), pp. 411–416.

[4] F. N. FRITSCH AND J. BUTLAND, *An improved monotone piecewise cubic interpolation algorithm*, Lawrence Livermore National Laboratory preprint UCRL-85104, 1980.

[5] F. N. FRITSCH AND R. E. CARLSON, *Monotone piecewise cubic interpolation*, SIAM J. Numer. Anal., 17 (1980), pp. 238–246.

[6] F. N. FRITSCH AND R. E. CARLSON, *Piecewise cubic interpolation methods*, Lawrence Livermore National Laboratory preprint UCRL-81230, 1978.

[7] J. M. HYMAN, *Accurate convexity preserving cubic interpolation*, informal report, Los Alamos Scientific Laboratory, Los Alamos, NM, November 1980.

[8] ———, *Accurate cubic piecewise monotone interpolation*, Los Alamos Scientific Laboratory, Los Alamos, NM, LA-UR-80-3700, 1980.

[9] ———, *The numerical solution of time dependent* PDEs *on an adaptive mesh*, Los Alamos Scientific Laboratory, Los Alamos, NM, LA-UR-80-3702, 1980 to be published.

[10] G. I. KERLEY, *Rational function method of interpolation*, Los Alamos National Laboratory report LA-6903-MS, Los Alamos, NM, 1977.

[11] D. F. MCALLISTER, E. PASSOW AND J. A. ROULIER, *Algorithms for computing shape preserving spline interpolations to data*, Math. Comput., 31 (1977), pp. 717–725.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0008 $01.25/0

MULTIGRID ALGORITHMS FOR THE SOLUTION OF LINEAR COMPLEMENTARITY PROBLEMS ARISING FROM FREE BOUNDARY PROBLEMS*

ACHI BRANDT† AND COLIN W. CRYER‡

Abstract. Several free boundary problems (including saturated-unsaturated flow through porous dams, elastic-plastic torsion and cavitating journal bearings) can be formulated as linear complementarity problems of the following type: Find a nonnegative function u which satisfies prescribed boundary conditions on a given domain and which, furthermore, satisfies a linear elliptic equation at each point of the domain where u is greater than zero. We show that the multigrid FAS algorithm, which was developed by Brandt to solve boundary value problems for elliptic partial differential equations, can easily be adapted to handle linear complementarity problems. For large problems, the resulting algorithm, PFAS (projected full approximation scheme) is significantly faster than previous single-grid algorithms, since the computation time is proportional to the number of grid points on the finest grid.

We then introduce two further multigrid algorithms, PFASMD and PFMG. PFASMD is a modification of PFAS which is considerably faster than PFAS. Using PFMG (projected full multigrid) it is possible to solve a linear complementarity problem to within truncation error using less work than the equivalent of seven Gauss–Seidel sweeps on the finest grid.

AMS (MOS) subject classifications. 35J65, 35R35, 65N99, 90C33

Key words. multigrid algorithms, free boundary problems, linear complementarity problems

1. Introduction. Several free boundary problems can be reformulated as an (infinite-dimensional) LCP (linear complementarity problem): Given a domain $\Omega \subset R^n$ with boundary $\partial\Omega$, and given functions f and g, find u (defined on Ω) such that (in an appropriate weak sense)

$$
\begin{aligned}
&\text{(a)} && \mathcal{L}u(x) \leqq f(x), && x \in \Omega,\\
&\text{(b)} && u(x) \geqq 0, && x \in \Omega,\\
&\text{(c)} && u(x)[\mathcal{L}u(x) - f(x)] = 0, && x \in \Omega,\\
&\text{(d)} && u(x) = g(x), && x \in \partial\Omega,
\end{aligned}
\tag{1.1}
$$

where $\mathcal{L}$ is a given second order elliptic operator. We do not write (1.1a) in the more usual form $-\mathcal{L}u(x) + f(x) \geqq 0$ because we wish to maintain compatibility with the notation in previous papers by Brandt.

Well-known examples of free boundary problems which can be written in the form (1.1) include porous flow through dams (a recent reference is Baiocchi [1978]), journal bearing lubrication (Cryer [1971a], Cimatti [1977]) and elastic-plastic torsion (Cea, Glowinski and Nedelec [1974], Lanchon [1974], Cryer [1980]). General references include: Duvaut and Lions [1976], Glowinski, Lions and Tremolieres [1976], Cryer [1977], Glowinski [1978], Cottle, Giannessi and Lions [1980], and Kinderlehrer and Stampacchia [1980].

* Received by the editors January 16, 1981, and in revised form July 15, 1982.

† Department of Applied Mathematics, the Weizmann Institute of Science, Rehovot, Israel. The research of this author was sponsored by the United States Army under contract DAAG29-80-C-0041 and contract DAJ37-79-C-0504.

‡ Computer Sciences Department and Mathematics Research Center, University of Wisconsin–Madison, Madison, Wisconsin 53706. Current address: Department of Applied Mathematics and Theoretical Physics, University of Cambridge. The research of this author was sponsored by the National Science Foundation under grant MCS77-26732 and by the United States Army under contract DAAG29-80-C-0041.

When (1.1) is approximated using finite differences on a grid G, one obtains a (finite-dimensional) LCP:

$$
\begin{aligned}
&\text{(a)} && LU(x) \leqq f(x), && x \in G,\\
&\text{(b)} && U(x) \geqq 0, && x \in G,\\
&\text{(c)} && U(x)[LU(x) - f(x)] = 0, && x \in G,\\
&\text{(d)} && U(x) = g(x) && x \in \partial G,
\end{aligned}
\tag{1.2}
$$

where $U(x)$ is an approximation to $u(x)$ at the grid points $x \in G \cup \partial G$ and where L is a difference operator which approximates $\mathscr{L}$. The coefficients of L are $O(h^{-2})$, where h denotes the grid length.

By multiplying (1.2) by h^2 and eliminating the known values of $U(x)$ on ∂G, the LCP (1.2) may be written in matrix form:

$$
\begin{aligned}
&\text{(a)} && AU \leqq b,\\
&\text{(b)} && U \geqq 0,\\
&\text{(c)} && U^T(AU - b) = 0,
\end{aligned}
\tag{1.3}
$$

where U is the N-vector of values of $U(x)$ on G, and A is an $N \times N$ matrix with coefficients which are $O(1)$. We will assume that A is symmetric and negative definite.

For example, if $\mathscr{L}$ is the Laplace operator in R^2, then a possible choice for L would be the classical five-point difference operator, in which case A would be a matrix with diagonal elements -4 and off-diagonal elements either 0 or 1.

There is an extensive literature on the (finite-dimensional) LCP (see Balinski and Cottle [1978]). In particular, if A is negative definite, as we assume, then there exists a unique solution to (1.2) and (1.3).

Since the LCP (1.3) arises from a free boundary problem, the matrix A has special properties which make it possible to use specialized algorithms which are particularly efficient. Such algorithms include projected SOR (Cryer [1971], Glowinski [1971]) the method of Cottle and Sacher [1977], and the modified block SOR (MBSOR) method of Cottle, Golub and Sacher [1978]; Cryer [1980a] summarizes these algorithms, and Cottle [1974] gives numerical comparisons between them.

Recently, it has been found (Brandt [1977], Brandt and Dinar [1979]) that multigrid algorithms are an effective tool for solving linear equations of the form

$$AX = b. \tag{1.4}$$

The basic idea of these multigrid algorithms is to compute on a sequence of nested grids. The computation proceeds on a particular grid until the error becomes smooth and the rate of convergence slows, at which point the computation is transferred to a coarser grid. When the error has been reduced on the coarser grid, the solution on the finer grid is corrected using interpolated values from the coarser grid.

In this paper, we show how the multigrid algorithm FAS of Brandt can be modified to solve the LCP (1.3). We find that the modified multigrid algorithm, PFAS, is substantially faster than previous single-grid algorithms.

The paper is organized as follows. In § 2, we describe PFAS, the projected full approximation scheme for solving (1.3): PFAS combines the concepts of multigrid algorithms with those of projected SOR. In § 3, we discuss the implementation of PFAS, and in § 4, we give numerical results obtained using PFAS. In § 5, we discuss

alternative implementations of PFAS, the last of which (PFASMD) leads to substantially improved convergence. We also include several less successful implementations because they are instructive. In § 6, we describe results obtained using PFMG, the projected full multigrid algorithm. The basic idea of PFMG is to compute the initial approximation on each grid by interpolating an accurate solution on the next coarsest grid. Using PFMG we are able to solve a problem to within truncation error using less work than the equivalent of seven Gauss–Seidel sweeps on the finest grid. Our results are summarized in § 7, and some possible extensions are mentioned.

Listings of the programs used in this paper are given in Brandt and Cryer [1980].

2. PFAS (projected full approximation scheme). Brandt [1977], [1980], and Brandt and Dinar [1979] give a detailed exposition of multigrid methods and their philosophy, and the reader is referred to these papers for background information. The algorithm described below, PFAS, is a modification of the FAS (full approximation scheme) which is considered in Brandt [1977, § 5] and in Brandt and Dinar [1979, § 2.2].

The domain $\Omega \subset R^n$ is approximated by a sequence of grids

$$G^1 \subset G^2 \subset \cdots \subset G^M \subset R^n,$$

with corresponding grid sizes

$$h_1 = 2h_2 = 4h_3 = \cdots = 2^{M-1}h_M.$$

Let F^k be the restriction of f to G^k,

$$F^k(x) = f(x), \qquad x \in G^k. \tag{2.1}$$

Then, on G^k the difference equations (1.2) approximating (1.1) take the form

$$\begin{aligned}
&\text{(a)} && L^k U^k(x) \leqq F^k(x) && \text{in } G^k,\\
&\text{(b)} && U^k(x) \geqq 0 && \text{in } G^k,\\
&\text{(c)} && U^k(x)[L^k U^k(x) - F^k(x)] = 0 && \text{in } G^k,\\
&\text{(d)} && U^k(x) = g(x) && \text{in } \partial G^k.
\end{aligned} \tag{2.2}$$

Let the points of G^k be ordered: $x_1^k, x_2^k, \cdots, x_{N_k}^k \in G^k$, and let U^k be the vector

$$U^k = \{U_j^k : 1 \leqq j \leqq N_k\} \equiv \{U^k(x_j^k) : 1 \leqq j \leqq N_k\}.$$

Then, (1.3) takes the form

$$\begin{aligned}
&\text{(a)} && A^k U^k \leqq b^k,\\
&\text{(b)} && U^k \geqq 0,\\
&\text{(c)} && (U^k)^T [A^k U^k - b^k] = 0
\end{aligned} \tag{2.3}$$

where

$$A^k = \{a_{ij}^k : 1 \leqq i, j \leqq N_k\} \tag{2.4}$$

is a known sparse symmetric negative definite matrix and $b^k = \{b_j^k\}$ is a known vector with components $b_j^k = h_k^2 F^k(x_j^k)$ except at points x_j^k adjacent to ∂G^k.

2.1. The projected Gauss–Seidel algorithm. It is possible to solve the LCP's (2.2) and (2.3) using the *projected Gauss–Seidel algorithm* which we now describe.

Let $u^{k,0}(x)$ be an approximate solution of (2.2) and (2.3). We compute recursively a sequence of approximations $u^{k,1}(x), u^{k,2}(x), \cdots$. Let $u^{k,s-1}(x)$ be given. From (2.2d),

the boundary values of $u^{k,s}(x)$ are equal to $g(x)$. The interior values of $u^{k,s}(x)$, which together comprise the vector

$$u^{k,s} = \{u_j^{k,s} : 1 \leqq j \leqq N_k\} = \{u^{k,s}(x_j^k) : 1 \leqq j \leqq N_k\}, \tag{2.5}$$

are obtained, point by point, by first applying the classical Gauss–Seidel method to (2.3) to obtain

$$u_j^{k,s-1/2} = u_j^{k,s-1} + \left[b_j^k - \sum_{l<j} a_{jl}^k u_l^{k,s} - \sum_{l \geqq j} a_{jl}^k u_l^{k,s-1} \right] \Big/ a_{jj}^k = u_j^{k,s-1} + \tilde{r}_j^{k,s} / a_{jj}^k, \tag{2.6}$$

say, and then *projecting*:

$$u_j^{k,s} = \max\{0, u_j^{k,s-1/2}\}. \tag{2.7}$$

The process of applying (2.6) and (2.7) for $1 \leqq j \leqq N_k$ to obtain $u^{k,s}$ from $u^{k,s-1}$ will be called a *G^k-projected Gauss–Seidel sweep*, or a *G^k-projected sweep*. The quantities $\tilde{r}_j^{k,s}$ will be called the *dynamic residuals*.

It is known (Cryer [1971], Glowinski [1971]) that $u^{k,s} \to U^k$ as $s \to \infty$.

When implementing the projected Gauss–Seidel method only the latest values of the solution are stored. We will, therefore, often suppress the iteration counter s and denote one projected Gauss–Seidel sweep applied to (2.2) and (2.3) by

$$u^k \leftarrow \text{projected Gauss–Seidel } [u^k : L^k, F^k]. \tag{2.8}$$

Similarly,

$$\nabla u^k = u^{k,s} - u^{k,s-1} \tag{2.9}$$

will denote the difference between the latest approximation u^k and its predecessor, while

$$\nabla u_{\text{old}}^k = u^{k,s-1} - u^{k,s-2} \tag{2.10}$$

denotes the previous difference.

2.2. Error estimates for the projected Gauss–Seidel algorithm. When implementing the projected Gauss–Seidel algorithm as part of a multigrid process, it is important to be able to estimate the error. In order to do so, we note that since, by assumption, $-A^k$ is symmetric and positive definite, there exists a coercitivity constant $\alpha_k > 0$ such that

$$w^T(-A^k)w \geqq \alpha_k w^T w, \tag{2.11}$$

for all $w \in R^{N_k}$.

LEMMA 2.1. *Let U^k be the solution of the* LCP (2.3), *and let $u^k \geqq 0$ be an approximate solution. Let*

$$r^k = (r_j^k) = b^k - A^k u^k, \tag{2.12}$$

and $r_+^k = (r_{+j}^k)$, where

$$r_{+j}^k = \begin{cases} r_j^k & \text{if } u_j^k > 0, \\ \min\{0, r_j^k\} & \text{if } u_j^k = 0. \end{cases} \tag{2.13}$$

Then

$$(U^k - u^k)^T(-A^k)(U^k - u^k) \leqq (U^k - u^k)^T(-r_+^k). \tag{2.14}$$

Hence,

$$\|U^k - u^k\|_2 \leqq \alpha_k^{-1} \|r_+^k\|_2. \tag{2.15}$$

Proof. With r_+^k defined as above, we see that u^k satisfies the LCP:

$$\begin{aligned} &\text{(a)} && A^k u^k \leqq b^k - r_+^k, \\ &\text{(b)} && u^k \geqq 0, \\ &\text{(c)} && (u^k)^T (A^k u^k - b^k + r_+^k) = 0. \end{aligned} \tag{2.16}$$

Following Falk [1974], we multiply (2.3a) by the nonnegative vector $(u^k)^T$ and use the complementarity condition (2.3c) to obtain

$$(u^k - U^k)^T A^k U^k \leqq (u^k - U^k)^T b^k. \tag{*}$$

Similarly, multiplying (2.16a) by $(U^k)^T$ we obtain

$$(U^k - u^k)^T A^k u^k \leqq (U^k - u^k)^T (b^k - r_+^k). \tag{**}$$

Adding (*) and (**) and combining terms we obtain (2.14) and hence (2.15). □

LEMMA 2.2. *Let U^k be the solution of the* LCP (2.3), *and let $u^k \geqq 0$ be an approximate solution obtained after one or more G^k projected sweeps. Let*

$$A^k = (D^k - L^k - P^k) \tag{2.17}$$

where D^k is diagonal, and L^k and P^k are strictly lower and upper triangular matrices, respectively.

Then u^k satisfies the LCP

$$\begin{aligned} &A^k u^k \leqq b^k - P^k \nabla u^k, \\ &u^k \geqq 0, \\ &(u^k)^T (A^k u^k - b^k + P^k \nabla u^k) = 0. \end{aligned} \tag{2.18}$$

Hence,

$$\|U^k - u^k\|_2 \leqq \alpha_k^{-1} \|P^k\|_2 \|\nabla u^k\|_2. \tag{2.19}$$

Proof. Consider the projected Gauss–Seidel method defined by (2.6) and (2.7). For each point x_j^k we first compute the dynamic residual $\tilde{r}_j^{k,s}$. The new value of $u_j^{k,s}$ is chosen so as to reduce the residual. Denote the residual at the point x_j^k immediately after step (2.7) by $\hat{r}_j^{k,s}$, so that

$$\hat{r}_j^{k,s} = \bar{r}_j^{k,s} - a_{jj}^k (u_j^{k,s} - u_j^{k,s-1}). \tag{2.20}$$

Remembering that A^k is negative definite, and hence $a_{jj}^k < 0$, we see that there are two possibilities:

$$\begin{aligned} \text{either} \quad & u_j^{k,s} > 0 \text{ and } \hat{r}_j^{k,s} = 0, \\ \text{or} \quad & u_j^{k,s} = 0 \text{ and } \hat{r}_j^{k,s} \geqq 0. \end{aligned}$$

Thus, dropping the superscript s, and setting $\hat{r}^k = \{\hat{r}_j^k : 1 \leqq j \leqq N_k\}$,

$$u^k \geqq 0, \quad \hat{r}^k \geqq 0, \quad (u^k)^T \hat{r}^k = 0. \tag{2.21}$$

Let

$$r^k = b^k - A^k u^k.$$

It is readily seen from (2.17) that

$$r^k = \hat{r}^k + P^k(u^{k,s} - u^{k,s-1}) = \hat{r}^k + P^k \nabla u^k. \tag{2.22}$$

Combining (2.21) and (2.22) we obtain (2.18). Comparing (2.16) and (2.18) we see that the arguments which led to (2.15) from (2.16) may be applied to (2.18), with r_+^k replaced by $P^k \nabla u^k$, to obtain (2.19). □

As Lemmas 2.1 and 2.2 show, we can estimate the error in an approximate solution u^k in terms of the residual r^k or the difference ∇u^k; we will usually use ∇u^k to estimate the error, since this quantity is readily available during a G^k-projected sweep.

Remark 2.1. The reader may wonder why we bothered to introduce r_+^k in Lemma 2.1, since (2.15) holds with r_+^k replaced by r^k. The reason is that for the LCP (2.3) there may be large positive residuals at points x_j^k where $U^k(x_j^k) = 0$, but this does not mean that the error is large.

In multigrid algorithms it is necessary to compare norms on different grids. We, therefore, wish to introduce a norm which is not grid dependent. To do so, we proceed as follows.

We first note that, to a good approximation, the coercivity constant α_k for $-A^k$ satisfies

$$\alpha_k \doteq \alpha h^2,$$

where α is the smallest eigenvalue of $\mathscr{L}$.

Next, assume that the approximate grid function u^k has been extended to a function $u^k(x)$ on Ω approximating the solution $u(x)$ of (1.1). Then

$$\begin{aligned} \|u(x) - u^k(x)\|_{2,\Omega} = \left| \int_\Omega |u(x) - u^k(x)|^2 \, dx \right|^{1/2} &\doteq \left| \sum_{j=1}^{N_k} h_k^n |U_j^k - u_j^k|^2 \right|^{1/2} \\ &= h_k^{n/2} \|U^k - u^k\|_2 \\ &\leqq \frac{h_k^{n/2}}{\alpha_k} \|P^k\|_2 \|\nabla u^k\|_2 \\ &\doteq \frac{\|P^k\|_2}{\alpha} h_k^{n/2-2} \|\nabla u^k\|_2. \end{aligned}$$

The norms $\|P^k\|_2$ are essentially independent of k; for example, for the five-point formula, $\|P^k\|_2 \leqq 2$. Thus a measure for the error $\|u(x) - u^k(x)\|_{2,\Omega}$ is provided by

$$\|\nabla u^k\|_G \equiv h_k^{n/2-2} \|\nabla u^k\|_2, \tag{2.23}$$

and this norm will be used in the computations.

2.3. PFAS (projected full approximation scheme). PFAS (projected full approximation scheme) obtains an approximation $\bar{u}^M$ to the solution U^M on the finest grid G^M by recursively generating a sequence of approximations $\bar{u}^k$ on the grids G^k.

Each $\bar{u}^k$ is an approximate solution to an LCP of the form (2.2) with F^k replaced by a function $\bar{F}^k$ which is defined later. In general, $\bar{F}^k$ is different from F^k so that $\bar{u}^k$ is not an approximation to U^k. However, $\bar{F}^M = F^M$ and so $\bar{u}^M$ is an approximation to U^M.

We begin by initializing $\bar{u}^M$ to some suitable value. For example, we might set

$$\bar{u}^M(x) = \begin{cases} g(x) & \text{on } \partial G^M, \\ 0 & \text{in } G^M. \end{cases} \tag{2.24}$$

We also set

$$\|\nabla \bar{u}^M\|_G = 10^{30}, \qquad \varepsilon^M = \varepsilon \tag{2.25}$$

(where ε is the desired accuracy on the finest grid, and where the astronomical number 10^{30} ensures that at least two G^M projected sweeps are carried out),

$$\begin{aligned} \bar{F}^M(x) &= F^M(x) \quad \text{for } x \in G^M, \\ \bar{U}^M(x) &= U^M(x) \quad \text{for } x \in G^M. \end{aligned} \tag{2.26}$$

We now make a number of G^M projected sweeps,

$$\bar{u}^M \leftarrow \text{projected Gauss–Seidel } [\bar{u}^M : L^M, \bar{F}^M]. \tag{2.27}$$

After each sweep we test whether

$$\|\nabla \bar{u}^M\|_G \leqq \varepsilon^M. \tag{2.28}$$

If so, the accuracy criterion is satisfied, and we accept $\bar{u}^M$ as an accurate approximation to $U^M \equiv \bar{U}^M$ on G^M.

It is known that Gauss–Seidel iteration is a smoothing process: the error $\bar{U}^M(x) - \bar{u}^M(x)$ becomes smoother as the number of sweeps increases, while, at the same time, the rate of convergence slows down. We, therefore, carry out only a few G^M projected sweeps, stopping when either (2.28) is satisfied or convergence is slow:

$$\|\nabla \bar{u}^M\|_G \geqq \eta \|\nabla \bar{u}^M_{\text{old}}\|_G. \tag{2.29}$$

Here, η is a fixed parameter; in our work we have taken $\eta = .5$.

Suppose that (2.28) is not satisfied but that (2.29) is satisfied. This means on the one hand that the accuracy of $\bar{u}^M$ must be improved, and on the other hand that it is inefficient to continue iterating on G^M. The slow rate of convergence on G^M indicates that the error is smooth, so that the error can be represented satisfactorily on the next coarsest grid, G^{M-1}. We therefore move to G^{M-1}.

Since $\bar{U}^M(x)$ satisfies (2.2), with $k = M$ and $F^M = \bar{F}^M$, the error

$$V^M(x) = \bar{U}^M(x) - \bar{u}^M(x) \tag{2.30}$$

satisfies the LCP

$$\begin{aligned} & L^M V^M(x) \leqq \bar{r}^M(x) && \text{on } G^M, \\ & V^M(x) + \bar{u}^M(x) \geqq 0 && \text{on } G^M, \\ & [V^M(x) + \bar{u}^M(x)][L^M V^M(x) - \bar{r}^M(x)] = 0 && \text{on } G^M, \\ & V^M(x) = 0 && \text{on } \partial G^M, \end{aligned} \tag{2.31}$$

where the residual $\bar{r}^M$ is given by

$$\bar{r}^M(x) = \bar{F}^M(x) - L^M \bar{u}^M(x), \qquad x \in G^M. \tag{2.32}$$

As already observed, $V^M(x)$ is a smooth function and may, therefore, be accurately represented on G^{M-1}. Furthermore, comparing (2.31) and (1.1) we see that $V^M(x)$

is an approximation to the continuous solution $v(x)$ of the LCP

$$
\begin{aligned}
&\mathcal{L}v(x) \leqq \bar{r}^M(x), && x \in \Omega,\\
&v(x) + \bar{u}^M(x) \geqq 0, && x \in \Omega,\\
&[v(x) + \bar{u}^M(x)][\mathcal{L}v(x) - \bar{r}^M(x)] = 0, && x \in \Omega,\\
&v(x) = 0 && \text{on } \partial\Omega
\end{aligned} \tag{2.33}
$$

(where, by abuse of notation, $\bar{r}^M(x)$ and $\bar{u}^M(x)$ are defined on Ω by appropriate interpolation between the values of $\bar{r}^M$ and $\bar{u}^M$ on the gridpoints of G^M). Thus, a good approximation to $V^M(x)$ may be obtained by solving the finite difference approximation to (2.33) on G^{M-1}. That is, $V^M(x)$ is closely approximated on G^{M-1} by the solution $W^{M-1}(x)$ of the LCP,

$$
\begin{aligned}
&\text{(a)} && L^{M-1}W^{M-1}(x) \leqq S_M^{M-1}\bar{r}^M(x), && \text{on } G^{M-1},\\
&\text{(b)} && W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x) \geqq 0, && \text{on } G^{M-1},\\
&\text{(c)} && [W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x)][L^{M-1}W^{M-1}(x) - S_M^{M-1}\bar{r}^M(x)] = 0, && \text{on } G^{M-1},\\
&\text{(d)} && W^{M-1}(x) = 0, && \text{on } \partial G^{M-1}.
\end{aligned} \tag{2.34}
$$

Here I_M^{M-1} and S_M^{M-1} are operators taking grid functions on G^M into grid functions on G^{M-1}. (As an aid in memorization, note that in $I_M^{M-1}\bar{u}^M$ the subscript M and superscript M "cancel".)

The operators I_M^{M-1} and S_M^{M-1} can be defined in many ways. One way is to choose both I_M^{M-1} and S_{M-1}^M to be the injection operator:

$$
\mathrm{Inj}_M^{M-1} w(x) = w(x), \qquad x \in G^{M-1}. \tag{2.35}
$$

If we were solving a linear boundary value problem, then condition (2.34b) would not apply, and it would be most efficient to solve for the correction W^{M-1} on G^{M-1}. Since we are solving inequalities the problem is nonlinear, and it is necessary to solve for a "full approximation" $\bar{U}^{M-1}$ on G^{M-1}.

Setting

$$
\bar{U}^{M-1}(x) = W^{M-1}(x) + I_M^{M-1}\bar{u}^M(x), \tag{2.36}
$$

it follows that $\bar{U}^{M-1}(x)$ satisfies the LCP

$$
\begin{aligned}
&\text{(a)} && L^{M-1}\bar{U}^{M-1}(x) \leqq \bar{F}^{M-1}(x) && \text{in } G^{M-1},\\
&\text{(b)} && \bar{U}^{M-1}(x) \geqq 0 && \text{in } G^{M-1},\\
&\text{(c)} && \bar{U}^{M-1}(x)[L^{M-1}\bar{U}^{M-1}(x) - \bar{F}^{M-1}(x)] = 0 && \text{in } G^{M-1},\\
&\text{(d)} && \bar{U}^{M-1}(x) = g(x) && \text{on } \partial G^{M-1},
\end{aligned} \tag{2.37}
$$

where

$$
\begin{aligned}
\bar{F}^{M-1}(x) &= S_M^{M-1}\bar{r}^M(x) + L^{M-1}I_M^{M-1}\bar{u}^M(x)\\
&= S_M^{M-1}[\bar{F}^M(x) - L^M\bar{u}^M(x)] + L^{M-1}I_M^{M-1}\bar{u}^M(x).
\end{aligned} \tag{2.38}
$$

Finally, we set

$$
\varepsilon^{M-1} = \delta\|\nabla\bar{u}^M\|_G, \tag{2.39}
$$

and

$$\bar{u}^{M-1} = I_M^{M-1}\bar{u}^M, \tag{2.40}$$

where δ is a constant; in our computations δ has been set equal to .15.

To recapitulate, starting with initial values of $\bar{u}^M$, ε^M, and $\bar{F}^M$, we first carry out G^M projected sweeps until convergence slows down. We then introduce a subsidiary problem on G^{M-1} with known $\bar{F}^{M-1}$ and ε^{M-1} and initial approximation $\bar{u}^{M-1}$. The process can be repeated, so that at any one stage of the computation we have a sequence of grid approximations $\bar{u}^M, \bar{u}^{M-1}, \cdots, \bar{u}^{k-1}$ (approximating $\bar{U}^M, \bar{U}^{M-1}, \cdots, \bar{U}^{k-1}$, respectively), tolerances $\varepsilon^M, \varepsilon^{M-1}, \cdots, \varepsilon^{k-1}$, and right-hand sides $\bar{F}^M, \bar{F}^{M-1}, \cdots, \bar{F}^{k-1}$.

In the general case, $\bar{U}^k$ is the solution of the LCP

$$\begin{aligned}
&\text{(a)} && L^k\bar{U}^k(x) \leqq \bar{F}^k(x) && \text{in } G^k,\\
&\text{(b)} && \bar{U}^k(x) \geqq 0 && \text{in } G^k,\\
&\text{(c)} && \bar{U}^k(x)(L^k\bar{U}^k(x) - \bar{F}^k(x)) = 0 && \text{in } G^k,\\
&\text{(d)} && \bar{U}^k(x) = g(x) && \text{on } \partial G^k,
\end{aligned} \tag{2.41}$$

or equivalently,

$$\begin{aligned}
&\text{(a)} && A^k\bar{U}^k \leqq \bar{b}^k,\\
&\text{(b)} && \bar{U}^k \geqq 0,\\
&\text{(c)} && (\bar{U}^k)^T(A^k\bar{U}^k - b^k) = 0.
\end{aligned} \tag{2.42}$$

This LCP is solved approximately using G^k projected sweeps until the latest approximation $\bar{u}^k$ satisfies either

$$\|\nabla\bar{u}^k\|_G \leqq \varepsilon^k \tag{2.43}$$

or

$$\|\nabla\bar{u}^k\|_G \geqq \eta\|\nabla\bar{u}^k_{\text{old}}\|_G. \tag{2.44}$$

If (2.44) holds but (2.43) does not, then a new problem on G^{k-1} is defined by setting

$$\bar{F}^{k-1} = S_k^{k-1}[\bar{F}^k - L^k\bar{u}^k] + L^{k-1}I_k^{k-1}\bar{u}^k, \tag{2.45}$$

$$\varepsilon^{k-1} = \delta\|\nabla\bar{u}^k\|_G, \tag{2.46}$$

$$\bar{u}^{k-1} = I_k^{k-1}\bar{u}^k, \tag{2.47}$$

$$\bar{U}^{k-1} = W^{k-1} + I_k^{k-1}\bar{u}^k, \tag{2.48}$$

$$V^k = \bar{U}^k - \bar{u}^k, \tag{2.49}$$

where W^{k-1} is an approximation to V^k on G^{k-1}. Unless otherwise indicated, I_k^{k-1} and S_k^{k-1} will be taken to be the injection operator Inj_k^{k-1}.

At some stage the latest approximation $\bar{u}^{k-1}$ must satisfy (2.43):

$$\|\nabla\bar{u}^{k-1}\|_G \leqq \varepsilon^{k-1}, \tag{2.50}$$

if for no other reason than that when $k-1=1$ we cannot introduce any more subsidiary problems and must iterate until (2.50) is satisfied. Having found an approximation $\bar{u}^{k-1}$ of sufficient accuracy, we return to G^k. To do so, we first determine an

approximation w^{k-1} to W^{k-1} from (2.48), namely,

$$w^{k-1} = \bar{u}^{k-1} - I_k^{k-1}\bar{u}^k. \tag{2.51}$$

Next, let I_{k-1}^k be an interpolation operator taking grid functions on G^{k-1} into grid functions on G^k. One choice for I_{k-1}^k is the bilinear interpolation operator L_{k-1}^k defined as follows. If P_1, P_2, P_3 and P_4 are the corners of a square in G^{k-1} (see Fig. 2.1), then

$$L_{k-1}^k w^{k-1}(P_i) = \begin{cases} w^{k-1}(P_i), & 1 \leqq i \leqq 4, \\ (w^{k-1}(P_1) + w^{k-1}(P_2))/2, & i = 5, \\ (w^{k-1}(P_1) + w^{k-1}(P_4))/2, & i = 6, \\ \left(\sum_{i=1}^{4} w^{k-1}(P_i)\right)\Big/4, & i = 7. \end{cases} \tag{2.52}$$

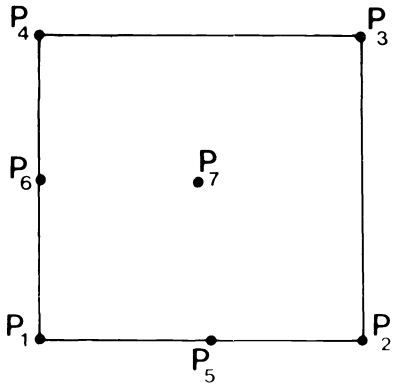


FIG. 2.1. *Bilinear interpolation from* G^{k-1} *to* G^k.

Since W^{k-1} is an approximation to V^k on G^{k-1},

$$I_{k-1}^k w^{k-1} = I_{k-1}^k [\bar{u}^{k-1} - I_k^{k-1}\bar{u}^k] \tag{2.53}$$

is an approximation to V^k, and, noting (2.49),

$$\tilde{u}^k = \bar{u}^k + I_{k-1}^k w^{k-1} \tag{2.54}$$

is an improved approximation to $\bar{U}^k$. However, because of the nonnegativity constraint upon $\bar{U}^k$, we allow somewhat greater generality, and replace $\bar{u}^k$ as follows:

$$\bar{u}^k \leftarrow \varphi(\tilde{u}^k; \bar{u}^k) = \varphi(\bar{u}^k + I_{k-1}^k w^{k-1}; \bar{u}^k). \tag{2.55}$$

Initially we set

$$\varphi(\tilde{u}^k; \bar{u}^k) = \tilde{u}^k, \tag{2.56}$$

but other choices will be considered later.

PFAS is described by (2.24) through (2.56). A flowchart is given in Fig. 3.1, and the implementation is discussed in § 3. If the algorithm converges, we will eventually obtain an approximation $\bar{u}^M$ satisfying the required accuracy condition (2.28), and the algorithm will terminate.

3. Implementation of PFAS. The flowchart for PFAS is given in Fig. 3.1. PFAS has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2, $\mathscr{L}$ is the Laplacian operator, L is the five-point difference operator, I_k^{k-1} and S_k^{k-1} are injections (see (2.35)), and I_{k-1}^k is bilinear interpolation (see (2.52)). The subroutine PFAS, which is listed in Brandt and Cryer [1980] as part of the program for solving the porous flow free boundary problem described in § 4, is a modification of an earlier program, FAS Cycle C, of Brandt.

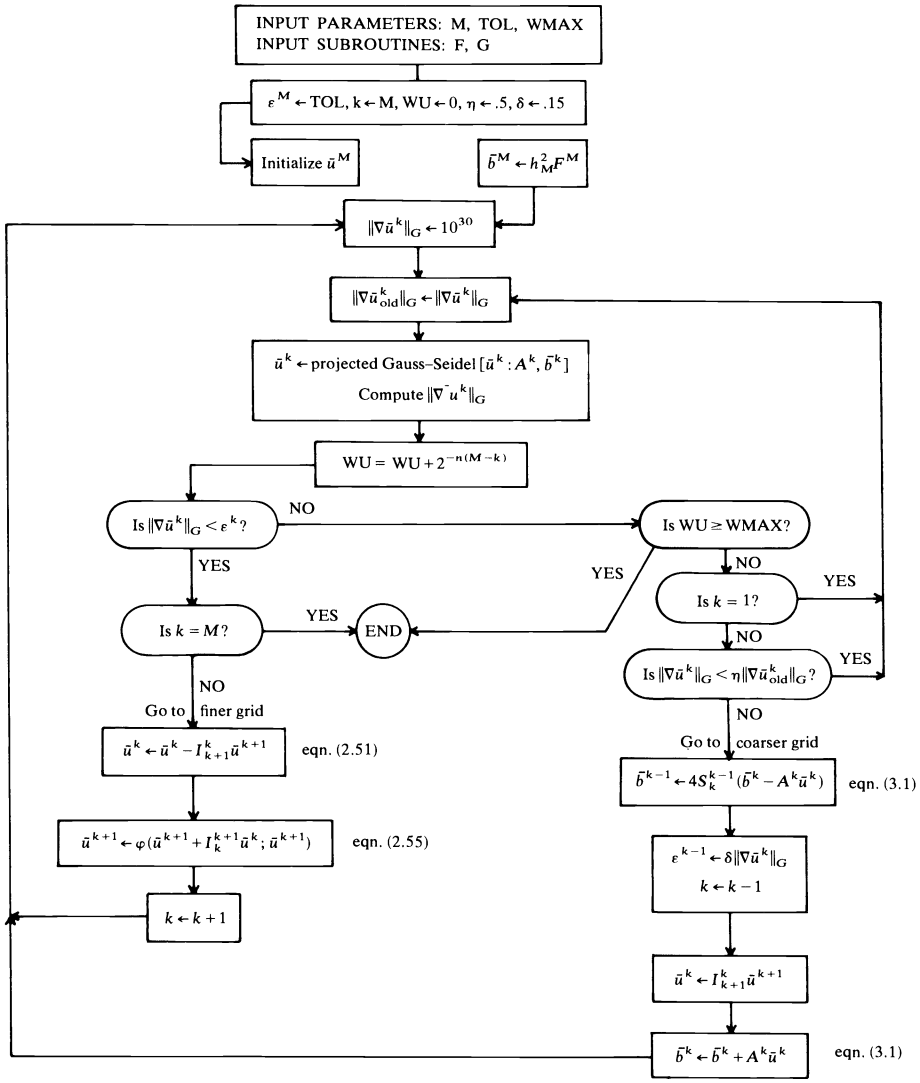


FIG. 3.1. *Flow chart for* PFAS.

PFAS is very easy to implement: the subroutine, with profuse comment cards, requires only 280 FORTRAN statements. It may also be remarked that many other interesting free boundary problems (for example, elastic-plastic torsion problems and cavitating journal bearing problems) are formulated on simple polygonal regions, and the program could easily be modified to handle these problems.

The following comments arise.

1. In PFAS, the LCP for $\bar{U}^k$ is solved in the form (2.42) rather than (2.41), but the values of $\bar{u}^k$ on ∂G^k are also stored. Thus, $\bar{b}^k = h_k^2 \bar{F}^k$ is stored instead of $\bar{F}^k$. In going from G^k to G^{k-1} we have, from (2.45), since $h_{k-1} = 2h_k$,

$$
\begin{aligned}
\bar{b}^{k-1} &= h_{k-1}^2 \bar{F}^{k-1} \\
&= h_{k-1}^2 (S_k^{k-1}[\bar{F}^k - L^k \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k) \\
&= h_{k-1}^2 (S_k^{k-1} h_k^{-2}[\bar{b}^k - A^k \bar{u}^k] + L^{k-1} I_k^{k-1} \bar{u}^k) \\
&= 4 S_k^{k-1}[\bar{b}^k - A^k \bar{u}^k] + A^{k-1} I_k^{k-1} \bar{u}^k.
\end{aligned} \tag{3.1}
$$

2. A G^k-*work-unit* is the work required for one G^k-projected sweep. The work for one G^k-projected sweep is approximately $2^{-n(M-k)} G^M$-work-units, and WU denotes the total number of G^M-work-units. When no confusion is possible we write "work unit" instead of "G^M-work-unit".

3. The asymptotic speed of convergence is measured by the *asymptotic convergence factor* $\mathring{\mu}$, which is defined by

$$
\mathring{\mu} = \lim_{WU \to \infty} [\|\nabla \bar{u}^M\|_G]^{1/WU}. \tag{3.2}
$$

4. All the numerical computations were performed on the Univac 1180 at the University of Wisconsin–Madison. The programs were written in ASCII FORTRAN and compiled and executed using full optimization.

The Univac 1180 single-precision arithmetic has approximately eight decimals. The residuals usually decrease quite rapidly at the beginning of a computation so the round-off threshold is quickly reached. For example, for the problem considered in § 4 with $M = 5$, $\|U^M\|_G$ is about 2×10^3, and the single precision algorithm went into a loop when $\|\nabla \bar{u}^M\|_G$ reached 5×10^{-6} after a mere 50 work units.

In the numerical experiments we were particularly interested in measuring the asymptotic convergence factor $\mathring{\mu}$. To eliminate round-off effects, all the computations reported on here used double precision arithmetic. Of course, this is not normally necessary. Furthermore, even if very accurate solutions of the discrete problem (2.2) were required, it would suffice to store $\bar{u}^M$ in double precision and all other quantities in single precision.

The execution times quoted are those provided by the Univac 1180 Exec. System. As is often the case on timesharing systems, the times are only reproducible to within about 10%.

Because of its word length, the UNIVAC 1180 can only directly access 64K words of storage. When $M \geqq 7$, more than 64K words of storage are needed by PFAS, and there is a significant degradation in performance.

5. To measure $\mathring{\mu}$, the iterations were continued for the first 100 work units, unless the residuals vanished before. In practice, one usually iterates only for about 30 work units.

We also used several values of M in order to measure the dependence of $\mathring{\mu}$ upon M.

The computations starting at a level-M–level-$(M-1)$ junction and continuing until the next level-M–level-$(M-1)$ junction are called a *cycle*.

While minor variations do arise, a cycle often consists of a sequence of 2 sweeps at each of levels $M-1, M-2, \cdots, 1$, followed by 2 sweeps at each of levels $2, \cdots, M-1$, terminating with 2 or 3 sweeps at level M. If this pattern is followed

with 3 sweeps at level M, then the average number of work units per cycle is

$$3+4[2^{-n}+2^{-2n}+\cdots]=3+4/(2^n-1), \tag{3.3}$$

and the average number of work units per G^M projected sweep is $1+4/(3(2^n-1))$.

Of course, very irregular patterns are observed when the round-off threshold is reached.

6. It is usually found that $\|\nabla \bar{u}^M\|_G$ decreases steadily but not very regularly, in part because of slight variations in the number of sweeps at each level. To evaluate the algorithm, we have used two quantities:

$$r_f = \|\nabla \bar{u}^M_{\text{final}}\|_G = \text{ the value of } \|\nabla \bar{u}^M\|_G \text{ at the end of the last complete cycle before 100 work units,} \tag{3.4}$$

$$\mathring{\mu}_f = [\|\nabla \bar{u}^M_{\text{final}}\|_G / \|\nabla \bar{u}^M_{\text{initial}}\|_G]^{1/[\text{WU}_{\text{final}}-\text{WU}_{\text{initial}}]}. \tag{3.5}$$

where $\|\nabla \bar{u}^M_{\text{initial}}\|_G$ is the value of $\|\nabla \bar{u}^M\|_G$ after the first G^M sweep; $\mathring{\mu}_f$ is an estimate for the asymptotic convergence factor $\mathring{\mu}$.

We usually only quote r_f to one decimal place and $\mathring{\mu}_f$ to two decimal places, since this is quite adequate for our purposes.

7. In all the experiments reported here, the parameters δ and η (see (2.29) and (2.39)) were given by $\delta = .5$ and $\eta = .15$. According to Brandt [1977], the rate of convergence is not very sensitive to changes in these parameters, and this was confirmed in a few experiments.

In a few cases, but never for $\delta = .5$ and $\eta = .15$, the program "hunted": that is, the program went down from G^M to G^1, up to G^k for $k < M$, and then down again to G^1 instead of continuing up to G^M. This might happen several times before G^M was reached again.

4. Numerical results for porous flow through a dam. Calculations were performed on the well-known free boundary problem describing the flow of water through a porous dam. The geometry is shown in Fig. 4.1. Water seeps from a reservoir of height y_1 through a rectangular dam of width a to a reservoir of height y_2. Part of the dam is saturated and the remainder of the dam is dry. The wet and dry regions are separated by an unknown free boundary which must be found as part of the solution. For an introduction to the problem see Bear [1972] or Cryer [1976].

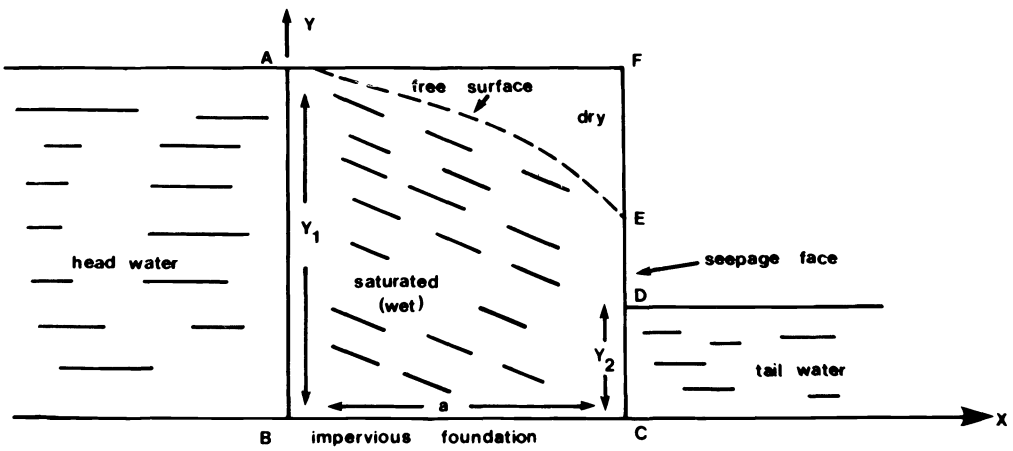


FIG. 4.1. *Seepage through a simple rectangular dam.*

As shown by Baiocchi [1971], the problem can be formulated as follows: Find u on the rectangle $\Omega = ABCF$ such that

$$
\begin{aligned}
&u_{xx} + u_{yy} \leqq 1 && \text{in } \Omega, \\
&u \geqq 0 && \text{in } \Omega, \\
&u(u_{xx} + u_{yy} - 1) = 0 && \text{in } \Omega,
\end{aligned} \tag{4.1}
$$

$$
u = g = \begin{cases} (y_1 - y)^2/2 & \text{on } AB, \\ (y_2 - y)^2/2 & \text{on } CD, \\ [y_1^2(a-x) + y_2^2(x)]/2a & \text{on } BC, \\ 0 & \text{on } DFA, \end{cases} \tag{4.2}
$$

which is in the form (1.1).

This problem was solved using PFAS. The initial values of $\bar{u}^M$ were obtained by interpolating the boundary values of u linearly in the x direction. A listing of the program is given in Brandt and Cryer [1980].

We considered the well-known case, $y_1 = 24$, $y_2 = 4$ and $a = 16$. In all computations G^1 was a $(2+1)\times(3+1)$ grid with $h_1 = 8$. The finest grid used was G^7 with $(128+1)\times(192+1) = 24897$ grid points.

To give the reader an idea of the solution, the solution U^2 of (2.2) is given to four decimal places in Table 4.1.

TABLE 4.1

U^2 for the dam problem.

y \ x	0	4	8	12	16
24	0	0	0	0	0
20	8	2.5371	0	0	0
16	32	18.1486	6.7841	0	0
12	72	47.2732	24.9879	7.9120	0
8	128	89.9564	53.9823	22.6601	0
4	200	146.5702	94.3247	44.7462	0
0	288	218.0000	148.0000	78.0000	8

TABLE 4.2

Solution of the dam problem using PFAS.

M	2	3	4	5	6	7
G^M	5×7	9×13	17×25	33×49	65×97	129×193
r_f	0*	4(−17)*	1(−13)	1(−8)	1(−10)	1(−7)
$\mathring{\mu}_f$	.404	.607	.726	.813	.778	.81
Execution time for 100 work units (seconds)	.114	.428	1.04	3.55	13.39	†
ρ_{SORopt}	.18	.49	.71	.84	.92	.96

* Reached round-off level before 100 work units.

† Required 70K workspace so extended storage facility invoked, and timing not compatible.

The numerical results, for different values of M and $\varepsilon^M = \text{TOL} = 0$, are given in Table 4.2. The most important conclusions are that convergence always occurs and that the convergence factor $\mathring{\mu}_f$ is always less than .81.

We now compare the convergence factors $\mathring{\mu}_f$ in Table 4.2 with those for single-grid methods of solving the LCP (2.2).

A popular single-grid method of solving the LCP (1.3) is G^M-projected SOR (point SOR with projection) which has also been called "modified SOR" by Cottle.

When using G^M-projected SOR, it is observed experimentally that the values of $\bar{u}^M$ settle down quite quickly into positive values and zero values. Thereafter G^M-projected SOR is equivalent to using point SOR on the subset $G_+^M = \{x \in G^M : U^M(x) > 0\}$. Thus the asymptotic convergence factor for G^M projected SOR is in general equal to the asymptotic convergence factor for point SOR on G_+^M. It is known (Varga [1962, p. 294]) that for a region of area A and for the finite difference equations corresponding to the five-point difference approximation to Laplace's equation with stepsize h, the convergence factor for the optimum choice of overrelaxation parameter ω is approximated quite well by

$$\rho_1(h) = \frac{2}{1+3.015[h^2/A]^{1/2}} - 1. \tag{4.3}$$

In the present case we do not know the area of G_+^M, but, as a rough guide, the area of G_+^M is approximately equal to the area of Ω, which is about 80% of the area of the rectangle $ABCF$. Therefore, for our present purposes the asymptotic convergence factor for G^M-projected SOR with optimum choice of ω may be taken to be

$$\rho_{\text{SORopt}} \doteq \frac{2}{1+3.015[h^2/(.8\times 16\times 24)]^{1/2}} - 1 \doteq \frac{2}{1+.172h} - 1; \tag{4.4}$$

these values are given at the bottom of Table 4.2.

As Table 4.2 shows, for large problems, PFAS is faster than G^M-projected SOR. On G^7, for example, the increase in speed (measured in work units) is $\ln .81/\ln .96 \doteq 5.2$. Against this, two factors must be borne in mind: (1) PFAS is more complicated and requires more overhead per work unit; (2) PFAS requires somewhat more storage. We discuss these two factors below, but before doing so we wish to emphasize that although these factors reduce the advantage in speed of PFAS, the measured execution times for PFAS are much smaller than those for G^M-projected SOR.

1. *Overhead.* To obtain an indication of the additional overhead required by PFAS, we compared execution times for $M = 5$. We first used PFAS with $\varepsilon^M = 2 \cdot 10^{-8}$. This required 96.156 work units and took 3.40 seconds. We then modified PFAS so that only the grid $k = M$ was used and so that overrelaxation was used with the overrelaxation parameter ω given by equation (4.4). We were thus using G^M-projected SOR with a nearly optimum ω. To reduce $\|\nabla \bar{u}^M\|_G$ to $\varepsilon^M = 2 \cdot 10^{-8}$ required 146 work units and took 4.82 seconds. Since

$$(3.40/96.156)/(4.82/146) \doteq 1.07,$$

we conclude that, in this application, the additional overhead required by PFAS only increases the computation time per G^M-work-unit by about 10%.

2. *Storage.* As implemented here, PFAS keeps the solutions and residuals on all the grids, and therefore requires storage for $2[1+4^{-1}+4^{-2}+\cdots] = 8/3G^M$ grids. In contrast, G^M-projected SOR requires storage for only one G^M grid.

If storage is at a premium, the residuals on G^M need not be stored, and PFAS requires only 5/3 times as much storage as G^M-projected SOR. If $\bar{u}^M$ is stored to double precision, but $\bar{u}^k$ and $\bar{b}^k$ are stored to single precision for $k < M$, only 4/3 times as much storage is needed. If $F(x)$ were not the constant 1, but a complicated function, then either the function values or the residuals would have to be stored for G^M-projected SOR, and PFAS would require at most 33% more storage.

Another possible single-grid algorithm for solving the LCP (1.3) is the MBSOR (Modified Block SOR) algorithm of Cottle, Golub and Sacher [1978]. This algorithm is based upon the solution of a sequence of "one-dimensional" LCP's in much the same way as line SOR is based upon solving a sequence of "one-dimensional" equations. We used MBSOR to solve the dam problem (4.1), (4.2) for the case $M = 5$. The program was kindly provided by Professor Sacher. We tried a few values of the overrelaxation parameter ω, and found that 1.8 gave the best results. With $\omega = 1.8$, MBSOR required 114 iterations to reduce $\|\nabla u^M\|_G$ to below $2 \cdot 10^{-8}$ and took 13.13 seconds. The following comments arise.

(i) In numerical experiments on the dam problem, Cottle [1974] found that MBSOR was about 20% faster than "modified point SOR", that is, G^M-projected SOR. This is consistent with the fact that, for equations, the convergence ratio for line SOR is only faster by a factor of $\sqrt{2}$ than point SOR, while there is more computation per iteration. This is also consistent with the present results, since G^M-projected SOR required 146 iterations to reduce the residual to $2 \cdot 10^{-8}$ while MBSOR required only 114.

(ii) The poor execution time of MBSOR (13.13 seconds) compared to PFAS (3.40 seconds) can be explained in part by two factors: (a) MBSOR requires more computation per iteration than is needed by PFAS for a single work unit; (b) the MBSOR program was written for the case of general coefficients, while the PFAS program takes advantage of the properties of the five-point difference operator.

(iii) It must also be borne in mind that Cottle, Golub and Sacher [1978] found that MBSOR was three times as fast as G^M-projected SOR for the journal bearing problem where the solution is zero at a high percentage of the gridpoints.

We conclude from Table 4.2 and from the above discussion, that for the dam problem (4.1), (4.2), PFAS is faster than G^M-projected SOR and modified block SOR for $M \geqq 5$, that is, for grids of dimension 33×49 or greater. Furthermore, we also conclude that the values of $\mathring{\mu}_f$ and ρ_{SORopt} in Table 4.2 provide a reasonably accurate guide to the relative performance of PFAS and G^M-projected SOR. We believe that PFAS will be faster than both G^M-projected SOR and MBSOR for a wide range of problems. Indeed, as shown by Table 4.2, the asymptotic convergence factor $\mathring{\mu}$ for PFAS is approximately equal to .8 for all values of M. Consequently, the amount of work required to reduce the residuals on G^M to below a given threshold ε is $O(N)$, where N is the number of gridpoints in G^M. In contrast, both G^M-projected SOR and modified block SOR have computation times which are $O(N^{3/2})$.

5. Alternative implementations of PFAS. In this section we discuss alternative implementations of PFAS, the best of which achieves substantially improved performance.

The improvement in PFAS which might be possible is suggested by considering the asymptotic convergence ratio, $\mathring{\mu}_{\text{FAS}}$ say, for FAS for Poisson's equation. For FAS, the error reduction per G^M-sweep is .5. If each G^M-sweep is accompanied by, on average, one G^k-sweep for $1 \leqq k \leqq M-1$, then the number of work units per

G^M-sweep is

$$1+2^{-2}+2^{-4}+\cdots=4/3,$$

and the convergence ratio is $(.5)^{3/4}=.595$, as stated by Brandt [1977, p. 351]. In the present case, as observed in § 3, the average number of work units per G^M-sweep is

$$1+4/[3(2^n-1)]=13/9,$$

so that

$$\mathring{\mu}_{\mathrm{FAS}}=(.5)^{9/13}=.6188. \tag{5.1}$$

This value of $\mathring{\mu}_{\mathrm{FAS}}$ is observed experimentally. The worst observed value of $\mathring{\mu}_f$ for the PFAS results quoted in § 3 was $\mathring{\mu}_f=.81$. Thus, FAS (for equations) is faster than PFAS (for LCP's) by a factor of $\ln .6188/\ln .81=2.28$.

Plausible reasons why PFAS is slower than FAS include the following difficulties:

D1: *Negative components of* $\bar{u}^k$. The inequality (2.41b) requires that $\bar{U}^k$ be nonnegative. In each G^k-projected sweep the step (2.7) ensures that $\bar{u}^k$ is nonnegative. Furthermore, if I_k^{k-1} is the injection operator, the initial approximation $\bar{u}^{k-1}$ defined by (2.47) is also nonnegative. However, (2.54) does not preserve nonnegativity: in returning to G^k from G^{k-1}, the initial approximation $\bar{u}^k$ may have negative components, and this is often observed. Of course, any negative components are removed in the first subsequent G^k-projected sweep, but nevertheless the introduction of negative components must retard convergence.

D2: *Large residuals near the free boundary.* At a point $x\in G^k$ where $\bar{U}^k(x)=0$ the corresponding residual

$$\bar{R}^k(x)=\bar{F}^k(x)-L^k\bar{U}^k(x) \tag{5.2}$$

must be nonnegative because of the inequality (2.41a) but need not be small.

D3: *Influence of the discrete interface.* The *discrete interface* $\Gamma^k\subset R^2$ is the interface between the set of points where $\bar{U}^k>0$ and the set of points where $\bar{U}^k=0$. Γ^k approximates the *continuous interface*, or *free boundary*, Γ separating the points where the solution $u(x)$ is positive from the points where $u(x)$ is zero.

In special cases it may happen that $\Gamma^k=\Gamma$ for all k, in which case PFAS converges as fast as FAS. An example is given by problem (5.3), (5.4) below with $R=2$, for which Γ is the line $y=5-2x$; it is found experimentally that $\Gamma^k=\Gamma$ for $k\leqq 6$.

In general, Γ^k and Γ differ by $O(h_k)$, and Γ^k and Γ^{k-1} differ by $O(h_k)$. In particular, it may happen that $\bar{U}^k(x)>0$ while $\bar{U}^{k-1}(x)=0$. Furthermore, near Γ^{k-1} the residuals may be less smooth because of the projection (2.7) and because of the irregular shape of Γ^k and Γ^{k-1}. This introduces errors in the coarse grid corrections (2.55), thereby slowing the rate of convergence. Finally, the injection operator (2.35) is not adequate if the data to which it is applied is not smooth.

To test the influence of the relationship between Γ and Γ^k on the convergence of PFAS, computations were made not only for the dam problem (4.1), (4.2) but also for the LCP:

$$\begin{aligned}
&\text{(a)} && u_{xx}+u_{yy}\leqq f(x,y) && \text{in } \Omega,\\
&\text{(b)} && u\geqq 0 && \text{in } \Omega,\\
&\text{(c)} && u=g && \text{on } \partial\Omega,\\
&\text{(d)} && u(u_{xx}+u_{yy}-f)=0 && \text{in } \Omega,
\end{aligned} \tag{5.3}$$

where $\Omega = [0, 3] \times [0, 2]$, and f and g are chosen so that the exact solution is

$$u = [\cos(x+y)+2][\max\{0; 2.5R - Rx - y\}]^2. \tag{5.4}$$

Here, R is a parameter which is chosen close to the value 2. Note that $u \in C^2(\Omega)$ and $u = 0$ above the line $y = R(2.5 - x)$. By changing the value of R we can force gridpoints to lie very close to the exact free boundary; this may be expected to cause PFAS difficulty, because if $\bar{U}^k(x)$ is positive but very small for some $x \in G^k$ then it will take PFAS a large number of iterations to determine whether $\bar{U}^k(x)$ is zero or positive.

Multigrid algorithms can often be speeded up by modifying the operators I_k^{k-1}, S_k^{k-1} and I_{k-1}^k. We have tried a number of modifications of the corresponding PFAS subroutines which were intended to address the difficulties D1 to D3 mentioned above.

Our first modifications to the auxiliary subroutines of PFAS were not very successful, but they were very instructive and we briefly summarize them. In all cases, the results are for the dam problem with $M = 5$.

M1. PFAS was modified so as to enforce nonnegativity of $\bar{u}^k$ immediately after returning from G^{k-1}. This was done by defining φ in (2.55) by

$$\varphi(\tilde{u}^k; \bar{u}^k) = \max\{0, \tilde{u}^k\}. \tag{5.5}$$

This modification converged slightly faster than PFAS with $\mathring{\mu}_f = .803$.

M2. The usual situation in which the nonnegativity of $\bar{u}^k$ is violated is as follows. Let $\bar{u}^k(x) = 0$, where $x \in G^k$ but $x \notin G^{k-1}$. Let $y \in G^{k-1}$ be a neighbor of x, such that $\bar{u}^k(y) > 0$. It may then happen that $W^{k-1}(y) < 0$. As a result, $(I_{k-1}^k w^{k-1})(x)$ may be negative, and if so the updated value of $\bar{u}^k(x)$ will be negative.

To avoid this, PFAS was modified so that the operator I_k^{k-1} became:

$$I_k^{k-1}\bar{u}^k(y) = \begin{cases} \bar{u}^k(y) & \text{if } \bar{u}^k(x) > 0 \text{ for all eight} \\ & \text{neighbors } x \text{ of } y \text{ in } G^k, \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

Remembering from (2.48) that

$$\bar{U}^{k-1} = W^{k-1} + I_k^{k-1}\bar{u}^k,$$

we see from (5.6) and (2.41b) that the restraint $W^{k-1}(y) \geqq 0$ is enforced for every point $y \in G^{k-1}$ with a neighbor $x \in G^k$ such that $\bar{u}^k(x) = 0$.

This modification converged slightly more slowly than PFAS, with $\mathring{\mu}_f = .817$.

M3. PFAS was modified so that if the current value of $\bar{u}^M(x)$ was zero, then $\bar{u}^k(x)$ was forced to be zero for $k < M$. In effect, (2.7) was followed by a further operation:

$$\text{If } k < M \quad \text{and} \quad \bar{u}^M(x_j^k) = 0, \quad \text{then } \bar{u}_j^{k,s} = 0. \tag{5.7}$$

This modification converged, but much more slowly than PFAS, with $\mathring{\mu}_f = .887$.

M4. Brandt [1977, p. 378] has found residual weighting useful when the coefficients of the differential equation are changing rapidly. We, therefore, changed the algorithm so that S_k^{k-1} became:

$$4S_k^{k-1}r^k(x) = \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k), \tag{5.8}$$

where $\Delta = (\Delta_1, \Delta_2)$ for integers Δ_1, Δ_2 and the only nonzero $\rho(\Delta)$ are

$$\begin{aligned} &\rho(0,0) = 1 \\ &\rho(0,1) = \rho(1,0) = \rho(0,-1) = \rho(-1,0) = \tfrac{1}{2}, \\ &\rho(1,1) = \rho(1,-1) = \rho(-1,1) = \rho(-1,-1) = \tfrac{1}{4}. \end{aligned} \tag{5.9}$$

This modification cycled between G^1 and G^2, as did the further modification for which I_k^{k-1} was also defined by (5.8), (5.9).

The nonconvergence of Modification M4 requires explanation, and this is provided by

LEMMA 5.1. *Let φ be defined by* (2.56). *For $1 \leqq k \leqq M$ let $\bar{U}^k$ be the solution of the* LCP (2.41), *where $\bar{F}^k$ satisfies* (2.45). *Finally, let I_{k-1}^k satisfy*

$$(I_{k-1}^k(z^{k-1}) = 0) \Rightarrow (z^{k-1} = 0) \quad \textit{for all } z^{k-1} \in R^{N_{k-1}}. \tag{5.10}$$

Then for PFAS *to converge it is necessary that*

$$S_k^{k-1}[\bar{F}^k - L^k\bar{U}^k] \geqq 0, \tag{5.11}$$

$$I_k^{k-1}\bar{U}^k \geqq 0, \tag{5.12}$$

$$[I_k^{k-1}\bar{U}^k]^T S_k^{k-1}[\bar{F}^k - L^k\bar{U}^k] = 0. \tag{5.13}$$

Proof. We apply PFAS by setting $\bar{u}^k = \bar{U}^k$, and forming the LCP (2.41) on G^{k-1}:

$$\begin{aligned} &L^{k-1}\bar{U}^{k-1} \leqq \bar{F}^{k-1}, \\ &\bar{U}^{k-1} \geqq 0, \\ &(\bar{U}^{k-1})^T(L^{k-1}\bar{U}^{k-1} - \bar{F}^{k-1}) = 0. \end{aligned} \tag{***}$$

Solving this exactly so that $\bar{u}^{k-1} = \bar{U}^{k-1}$, we then return to G^k. Since PFAS converges, the new value of $\bar{u}^k$ given by (2.55) must be equal to $\bar{U}^k$. That is,

$$I_{k-1}^k w^{k-1} = I_{k-1}^k[\bar{U}^{k-1} - I_k^{k-1}\bar{U}^k] = 0,$$

which, from (5.10), implies that

$$\bar{U}^{k-1} = I_k^{k-1}\bar{U}^k.$$

Substituting into (***) and noting (2.45), we obtain (5.11) through (5.13). □

The following remarks follow from Lemma 5.1.

1. Lemma 5.1 brings out an interesting difference between multigrid methods for equations and for inequalities. For equations, $\bar{F}^k - L^k\bar{U}^k = 0$ and conditions (5.11)–(5.13) are satisfied for any reasonable choice of S_k^{k-1} and I_k^{k-1}, but this is not true for inequalities.

2. Since $\bar{U}^k$ solves (2.41), inequalities (5.11) and (5.12) will certainly hold if S_k^{k-1} and I_k^{k-1} map nonnegative vectors into nonnegative vectors. In particular, this will be the case if S_k^{k-1} and I_k^{k-1} take linear combinations of values with nonnegative weights.

3. If S_k^{k-1} and I_k^{k-1} are injections, then (5.13) is implied by (2.41c).

4. If S_k^{k-1} is defined by (5.8) and (5.9) while I_k^{k-1} is injection, then (5.13) does not hold in general. This is because in general there will be points $x, y \in G^k$ such that $x \in G^{k-1}$, $\bar{U}^k(x) > 0$, $\bar{U}^k(y) = 0$, y is a neighbor of x in G^k and $(\bar{F}^k - L^k\bar{U}^k)(y) > 0$. Then

$$I_k^{k-1}\bar{U}^k(x) = \bar{U}^k(x) > 0$$

and

$$(S_k^{k-1}(\bar{F}^k - L^k\bar{U}^k))(x) \geqq \tfrac{1}{4}(\bar{F}^k - L^k\bar{U}^k)(y) > 0,$$

so that (5.13) does not hold. This explains why Modification M4 of PFAS did not converge.

We now describe two further modifications of PFAS which were tried:

M5. Bearing Lemma 5.1 in mind, it is possible to introduce weighted sums for which (5.13) does hold. One choice uses weighted residuals only near the boundary:

$$4S_k^{k-1}r^k(x) = \begin{cases} 4r^k(x) & \text{if } \bar{u}^k(x) = 0 \text{ or if } \bar{u}^k(y) > 0 \\ & \text{for all eight neighbors } y \in G^k \text{ of } x, \\ \sum_{\Delta} \rho(\Delta) r^k(x + \Delta h_k) \text{ signum } [\bar{u}^k(x + \Delta h_k)] & \text{otherwise,} \end{cases} \tag{5.14}$$

where

$$\text{signum } \alpha = \begin{cases} 1 & \text{if } \alpha > 0, \\ 0 & \text{if } \alpha = 0, \end{cases}$$

and where the weights $\rho(\Delta)$ are as in (5.9). This modification converged more slowly than with PFAS, and it was found that $\mathring{\mu}_f = .854$.

M6. As mentioned in D1 and D3 above, if $\bar{u}^k(x) = 0$ then it may happen that $\bar{u}^k(x) = \bar{u}^k(x) + I_{k-1}^k w^{k-1}(x)$ is not zero. It can be argued that changes of $\bar{u}^k(x)$ from or to zero should only be done on G^k. We, therefore, modified PFAS so that in (2.55) φ was defined by

$$\varphi(\tilde{u}^k(x); \bar{u}^k(x)) = \begin{cases} \tilde{u}^k(x) & \text{if } \bar{u}^k(x) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

I_k^{k-1} and S_k^{k-1} were injections. This program was called PFASMD.

We solved (4.1), (4.2) with $M = 5$ using PFAS and PFASMD. In each case, the computations were terminated when $\|\nabla \bar{u}^M\|_G \leqq 2 \cdot 10^{-8}$. The results are summarized in Table 5.1.

In Table 5.2 we compare PFAS and PFASMD for the problem (5.3), (5.4). As in Table 5.1 we iterated until $\|\nabla u^{(k)}\|_G \leqq 2 \cdot 10^{-8}$ on G^5.

We conclude from the results given in Tables 5.1 and 5.2 that PFASMD is substantially faster than PFAS.

Finally, in Table 5.3 we extend Table 4.2 by comparing the measured execution times for the projected SOR method and PFASMD for the dam problem for various values of M. In each case, the iterations were continued until $\|\nabla \bar{u}\|_G \leqq 2 \cdot 10^{-8}$.

TABLE 5.1

Solution of (4.1), (4.2) *with* $M = 5$ *and* $\varepsilon^M = 2.10^{-8}$ *using* PFAS *and* PFASMD.

	Method	
	PFAS	PFASMD
Work units	96.15	42.81
Execution time (seconds)	3.40	1.63
$\mathring{\mu}_f$	.815	.623

TABLE 5.2

Solution of (5.3), (5.4) *with* $M=5$, $R=32/15$ *and* $\varepsilon^M=2\cdot 10^{-8}$ *for* PFAS *and* PFASMD.

	Method	
	PFAS	PFASMD
Work units	73.62	56.96
Execution time (seconds)	3.09	2.58
$\mathring{\mu}_f$	.731	.669

TABLE 5.3

Comparison of G^M *projected* SOR *and* PFASMD *for the dam problem with* $\varepsilon^M=2\cdot 10^{-8}$.

	$M=$	2	3	4	5	6
	$G^M=$	5×7	9×13	17×25	33×49	65×97
G^M projected SOR	G^M iterations	19	34	69	146	295
	Execution time (seconds)	.02	.09	.60	4.88	39.37
PFASMD	G^M work units	23	30.5	38.7	42.8	45.7
	Execution time (seconds)	.04	.12	.41	1.64	6.57

As can be seen from Table 5.3, PFASMD is better than projected SOR except for very small grids.

6. PFMG (projected full multigrid algorithm). In this section we describe PFMG (projected full multigrid algorithm), which is a modification of the full multigrid algorithm of Brandt. The flowchart for PFMG is given in Fig. 6.1. PFMG has been implemented as a FORTRAN subroutine for the case when Ω is a rectangle in R^2, and $\mathscr{L}$ is the Laplacian operator. This subroutine is listed in Brandt and Cryer [1980] as part of a program for solving the porous flow free boundary problem of § 4, and the problem (5.3), (5.4).[1]

PFMG differs from PFAS in the following respects:

I. Instead of beginning on G^M, one begins on a coarser grid G^{LIN} and gradually works up to G^M. The computations begin on the initial grid G^l, $l=\mathrm{LIN}$, with an initial approximation $\bar{u}^l$. $\bar{u}^l$ is computed to the required accuracy using grids G^1 through G^l as in PFAS, except that, as will be discussed below, the decision to move to a different grid is based on slightly different criteria.

Once $\bar{u}^l$ has been found to sufficient accuracy, the initial approximation $\bar{u}^{l+1}$ is obtained from

$$\bar{u}^{l+1}=J_l^{l+1}\bar{u}^l, \tag{6.1}$$

where J_l^{l+1} is an interpolation operator taking grid functions on G^l into grid functions on G^{l+1}. It is known (Brandt [1977, p. 377]) that J_l^{l+1} should be more accurate than I_l^{l+1} in order to preserve the smoothness of $\bar{u}^l$.

[1] There are two errors in the program as listed in Brandt and Cryer [1980]. On line 1127 change (ITAU.EQ.1) to (ITAU.EQ.1 · AND · T.NE.0). Card 1123 (TAUGNM = TAUGNM + T*T) should be placed after card 1126 (*Q(IP+JK) . EQ.0) $T=0$).

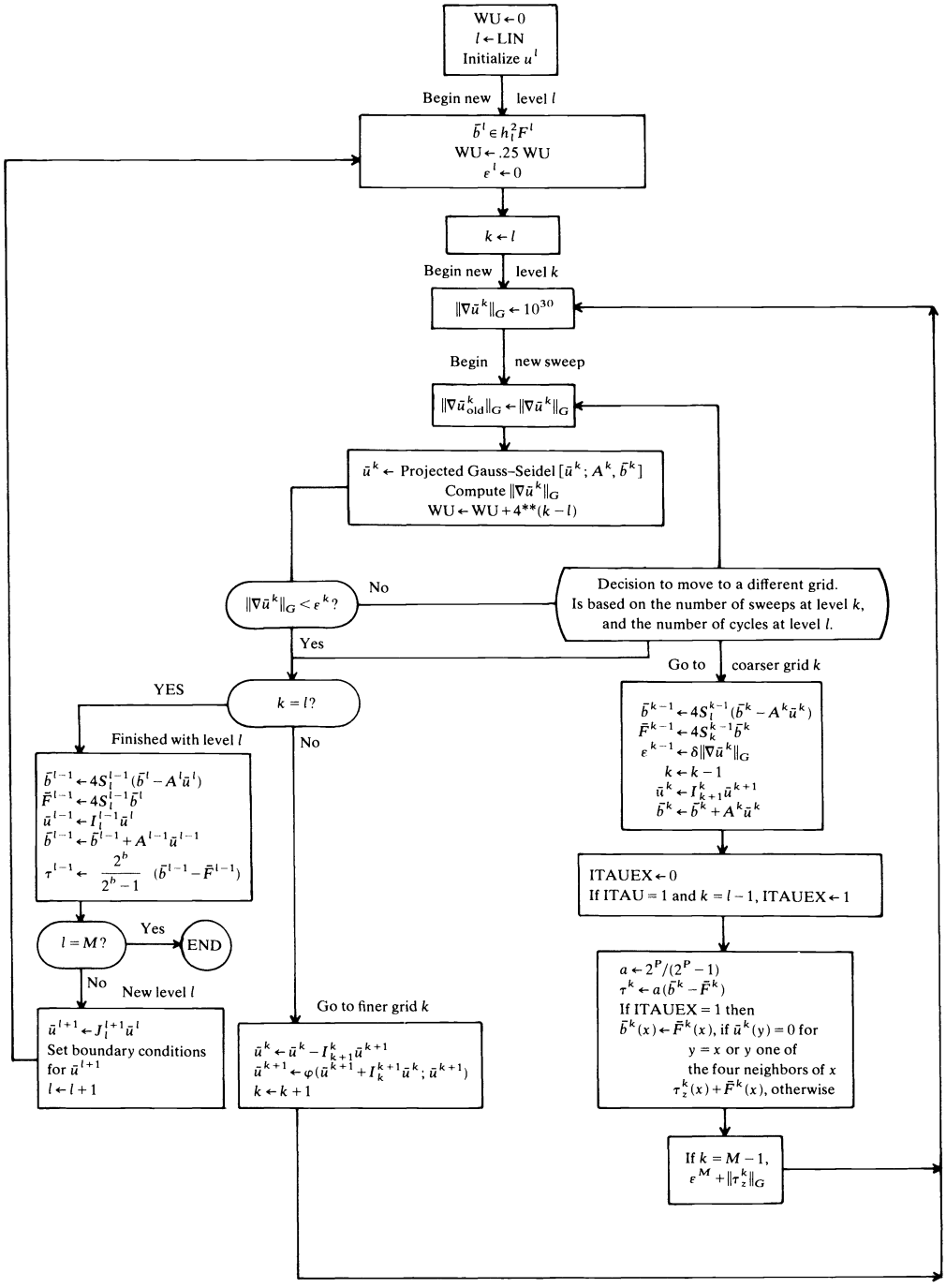


FIG. 6.1. *Flow chart for* PFMG.

In PFMG, J_l^{l+1} is based upon repeated use of the cubic interpolation formulas

$$f(\tfrac{1}{2}) = [-f(-1) + 9f(0) + 9f(1) - f(2)]/16, \tag{6.2}$$

$$f(\tfrac{3}{2}) = [f(-1) - 5f(0) + 15f(1) + 5f(2)]/16. \tag{6.3}$$

Repeating this process, we finally obtain an initial approximation $\bar{u}^M$ on G^M. Thereafter, the computation proceeds essentially as in PFAS.

II. $\bar{u}^k$ is used to estimate the local truncation error on G^{k-1}. Suppose that the difference approximations are of order p and that $\bar{u}^k$ can be extended to a smooth function on Ω. Then on G^{k-1},

$$A^{k-1} I_k^{k-1} \bar{u}^k \doteq h_{k-1}^2 \mathscr{L} \bar{u}^k + \tau^{k-1} \tag{6.4}$$

and

$$S_k^{k-1} A^k \bar{u}^k \doteq h_k^2 \mathscr{L} \bar{u}^k + 2^{-(p+2)} \tau^{k-1}, \tag{6.5}$$

where the *local truncation error* τ^{k-1} depends upon the derivatives of $\bar{u}^k$. Eliminating the unknown $\mathscr{L}\bar{u}^k$ we obtain

$$\tau^{k-1} \doteq \frac{2^p}{2^p - 1} [A^{k-1} I_k^{k-1} \bar{u}^k - 4 S_k^{k-1} A^k \bar{u}^k] \tag{6.6}$$

$$= \frac{2^p}{2^p - 1} [\{4 S_k^{k-1} (\bar{b}^k - A^k \bar{u}^k)\} + \{A^{k-1} I_k^{k-1} \bar{u}^k\} - \{4 S_k^{k-1} \bar{b}^k\}]. \tag{6.7}$$

The estimate (6.7) is not accurate near the discrete interface, and so PFMG computes τ_z^{k-1} where

$$\tau_z^{k-1}(x) = \begin{cases} \tau^{k-1}(x), & \text{if } \bar{u}^{k-1}(x) > 0, \\ 0 & \text{if } \bar{u}^{k-1}(x) = 0. \end{cases} \tag{6.8}$$

Because of the lack of smoothness of the solution near the free boundary, it is not entirely clear what the value of p should be. It is known (Brezzi and Sacchi [1976]) that the convergence of the finite difference approximations is probably only $O(h^1)$ in the $W^{1,2}(\Omega)$ norm, and Nitsche [1975] has proved $O(h^2 \ln h)$ convergence in the infinity norm. However, these are global error bounds, while we are concerned with the asymptotic behavior of the local truncation error τ. Except in a neighborhood of the discrete interface Γ^l, p is clearly equal to 2. Since the choice of p may vary over Ω, we could perhaps set $p = 1$ near Γ^l, but the values of τ near Γ^l are not very accurate and so, for simplicity, we have taken $p = 2$ everywhere.

III. As usual in numerical analysis, the estimate (6.7) for τ^{k-1} can be used in two ways:

(a) To estimate the error $\bar{u}^k - u$. Since $\tau^k \doteq 2^{-2-p} \tau^{k-1}$, and remembering that G^k has four times as many points as G^{k-1} but $h_{k-1} = 2h_k$, we see from (2.23) that

$$\|\tau_z^k\|_G \doteq \|\tau_z^{k-1}\|_G / 2^p. \tag{6.9}$$

Combining (6.7), (6.8) and (6.9) we obtain an estimate for $\|\tau_z^k\|_G$.

In the previous sections we were concerned with asymptotic convergence. That is, we were concerned with the rate of convergence of $\bar{u}^k$ to $\bar{U}^k$ over a very large number of iterations. However, if we want an approximation to the solution u of (1.1), it is only necessary to iterate until the residual on G is small compared with the truncation error, that is, until

$$\|\nabla \bar{u}^k\|_G = O(\|\tau_z^k\|_G). \tag{6.10}$$

Once (6.10) holds, further computation will improve the accuracy of $\bar{u}^k$ as a solution of the finite difference equations, but will not improve its accuracy as an approximation to u. Noting (6.9), we see that (6.10) will certainly be true if

$$\|\nabla \bar{u}^k\|_G \leqq \|\tau_z^{k-1}\|_G. \tag{6.11}$$

(b) Improvement of accuracy of $\bar{u}^{k-1}$. Once an estimate for the truncation error τ_z^{k-1} is available, it can be used to improve the accuracy of the difference approximation on G^{k-1} by replacing $F^{k-1}(x)$ by $F^{k-1}(x)+\tau_z^{k-1}(x)$ (see (6.4)). This is only done at points $x \in G^{k-1}$ such that $\bar{u}^{k-1}(y)>0$ for all four neighbors $y \in G^{k-1}$ of x since the value of τ_z^{k-1} is not accurate elsewhere.

Of course, this is only meaningful when $\|\tau_z^{k-1}\|_G$ is small compared to $\|\nabla \bar{u}^k\|_G$: if the iterations are continued for a long time, then convergence will not occur because the conditions of Lemma 5.1 will be violated; but PFMG is never used in this way. In fact, experience with equalities indicates that when τ-extrapolation is used, the best procedure is to avoid relaxation after returning for the last time to the finest grid.

IV. As already mentioned, the logic of PFMG is more complicated than that of PFAS. Several parameters are introduced, and this enables one to control explicitly the number of G^k-projected sweeps at any level k, and the number of cycles at level l. In the computations reported on here, in each cycle on grid G^l two G^k-projected sweeps are carried out for $1<k \leqq l$ as we descend from G^l to G^1, and one G^k-projected sweep is carried out as we ascend from G^1 to G^l. For $l=\mathrm{LIN}$, up to three G^l cycles are allowed, so that a good initial approximation can be obtained. For $\mathrm{LIN}<l<M$, only one G^l cycle is allowed, while up to 10 G^M cycles are allowed.

We now describe numerical results obtained using PFMG to solve the dam problem (4.1), (4.2). In all cases, G^1 is a $(2+1)\times(3+1)$ grid and $\mathrm{LIN}=2$.

PFMG includes the option of computing $\|\bar{u}^l-u\|_\infty$ and $\|\bar{u}^l-u\|_G$, where u is the exact solution. For the dam problem, it is possible to compute u analytically using elliptic integrals (Cryer [1976]), but this has not yet been done: we therefore took u to be the most accurate approximation known to us, namely the approximation $\bar{u}^7$ computed in double precision on a $(128+1)\times(192+1)$ grid as described in § 4. For problem (5.3), (5.4) the exact solution is given by (5.4).

We first performed a number of experiments with $M=2$, 3, 4, and 5:

1. τ-extrapolation (with $p=2$) gave slightly worse results for the dam problem and slightly better results for problem (5.3), (5.4). (This is explained in part by the observed behavior of τ as a function of h, as discussed below.)

2. In contrast to our experience with PFASMD, the use of equation (5.15) (modification M6) had only a slight effect. (This may be explained by the observation that the slow convergence of PFAS is caused by the existence of gridpoints near the discrete interface at which (without modification M6) the computed values of $\bar{u}^k$ fluctuate between zero and small positive quantities. This is a delicate asymptotic matter below the level of the truncation error, and hence does not trouble FMG.)

3. It was thought that convergence might be improved by multiplying the difference $\nabla \bar{u}^k(x)$ by h for points x near the free boundary before computing $\|\nabla \bar{u}^k(x)\|_G$. This was found to have negligible effect.

All the results given below are for the case of no τ-extrapolation and no modification.

The results for the dam problem for different values of M are shown in Table 6.1.

Since we only have estimates for τ^{M-1}, it is not possible to obtain rigorous error bounds. Nevertheless, it is interesting to apply the error bounds of § 2.

TABLE 6.1

Solution of the dam problem using PFMG.

	M		
	3	4	5
G^M work units	8.75	6.67	6.41
Execution time (seconds)	.0675	.145	.404
$\|\bar{u}^M - \bar{u}^7\|_\infty / \|u\|_\infty$	.000665	.000168	.0000532
$\|\bar{u}^M - \bar{u}^7\|_G / \|\bar{U}^M\|_G$	.000810	.000145	.0000388
$\|\nabla \bar{u}^M\|_G$	.0666	.0557	.0339
$\|\tau_z^{M-1}\|_G$	0.0771	0.0795	0.0383

Let $\bar{U}^M$ denote the vector obtained by evaluating the solution $u(x)$ on G^M. Then, from (6.4), (1.1), (2.2), (2.3), (2.13) and (3.1),

$$A^M \bar{U}^M \leqq b^M + \tau^M, \tag{6.12}$$

so that, from Lemma 2.1,

$$\|\bar{U}^M - U^M\|_2 \leqq \frac{1}{\alpha_M} \|\tau_+^M\|_2. \tag{6.13}$$

On the other hand, from Lemma 2.2,

$$\|U^M - \bar{u}^M\|_2 \leqq \frac{1}{\alpha_M} \|P^M\|_2 \|\nabla \bar{u}^M\|_2.$$

For the dam problem, P is an upper triangular matrix with at most two nonzero elements per row, and $\|P^M\|_2 \leqq 2$. Thus,

$$\|U^M - \bar{u}^M\|_2 \leqq \frac{2}{\alpha_M} \|\nabla \bar{u}^M\|_2. \tag{6.14}$$

Combining these inequalities we obtain

$$\|\bar{U}^M - \bar{u}^M\|_2 \leqq \frac{1}{\alpha_M} [\|\tau_+^M\|_2 + 2\|\nabla \bar{u}^M\|_2],$$

or, equivalently,

$$\|\bar{U}^M - \bar{u}^M\|_G \leqq \frac{1}{\alpha_M} [\|\tau_+^M\|_G + 2\|\nabla \bar{u}^M\|_G]. \tag{6.15}$$

Using (6.8) and (6.9), we conclude that

$$\|\bar{U}^M - \bar{u}^M\|_G \doteqdot \frac{1}{\alpha_M} \left[\frac{1}{2^p} \|\tau_z^{M-1}\|_G + 2\|\nabla \bar{u}^M\|_G\right]. \tag{6.16}$$

Next, we note that for the dam problem

$$\alpha_M \doteqdot \alpha h_M^2, \tag{6.17}$$

where

$$\alpha = \left(\frac{\pi}{16}\right)^2 + \left(\frac{\pi}{24}\right)^2 \doteqdot .055 > 14/256 \tag{6.18}$$

and

$$h_M = 16 \cdot 2^{-M}.$$

Thus, finally, for the dam problem,

$$\|\bar{U}^M - \bar{u}^M\|_G \doteqdot \frac{2^{2M}}{14}\left[\frac{1}{2^p}\|\tau_z^{M-1}\|_G + 2\|\nabla \bar{u}^M\|_G\right]. \tag{6.19}$$

For example, for $M = 5$ we obtain, using Table 6.1, that

$$\|\bar{U}^5 - \bar{u}^5\|_G / \|\bar{U}^5\|_G \doteqdot \frac{2^{10}}{14}[\tfrac{1}{4}(0.0383) + 2(.0339)]/(5.9\ 10^3) \doteq .000959; \tag{6.20}$$

the observed value quoted in Table 6.1 is .0000388.

In Table 6.2 we repeat the computations of Table 6.1 for the problem (5.3), (5.4).

TABLE 6.2

Solution of problem (5.3), (5.4) *using* PFMG.

	M		
	3	4	5
G^M work units	6.75	5.672	5.414
Execution time (seconds)	.101	.271	.861
$\|\bar{u}^M - \bar{U}^M\|_\infty / \|u\|_\infty$	.000985	.000266	.0000645
$\|\bar{u}^M - \bar{U}^M\|_G / \|\bar{U}^M\|_G$	.00122	.000376	.0000956
$\|\nabla \bar{u}^M\|_G$	.241	.121	.0764
$\|\tau_z^{M-1}\|_G$	1.56	.509	.147

The error estimate (6.19) also holds for the problem (5.3), (5.4), since we are using the Laplace operator on a rectangle with sides in the ratio 2:3. Applying (6.19) we obtain

$$\|\bar{U}^5 - \bar{u}^5\|_G / \|\bar{U}^5\|_G \doteqdot \frac{2^{10}}{14}|\tfrac{1}{4}(.147) + 2(.076)|/(1 \cdot 2\ 10^4) \doteq .00115$$

the observed value quoted in Table 6.2 is .0000645.

The behavior of the global error $\bar{u}^M - u$ can be checked using Tables 6.1 and 6.2. From Table 6.1 we have

$$\left[\frac{\|\bar{u}^5 - \bar{u}^7\|_\infty}{\|\bar{u}^3 - \bar{u}^7\|_\infty}\right]^{1/2} = \left[\frac{.0000532}{.000665}\right]^{1/2} \doteq \frac{1}{2^{1.82}}.$$

From Table 6.2,

$$\left[\frac{\|\bar{u}^5 - u\|_\infty}{\|\bar{u}^3 - u\|_\infty}\right]^{1/2} = \left[\frac{.0000645}{.000985}\right]^{1/2} \doteq \frac{1}{2^{1.96}}.$$

These results strongly suggest that the global error is $O(h^2)$.

The behavior of the local error τ can also be checked using Tables 6.1 and 6.2. From Table 6.1,

$$[\|\tau_z^4\|_G / \|\tau_z^2\|_G]^{1/2} = [.0383/.0771]^{1/2} \doteq 1/2^{.50},$$

while, from Table 6.2,

$$[\|\tau_z^4\|_G/\|\tau_z^2\|_G]^{1/2} = [.147/1.56]^{1/2} \doteq 1/2^{1.7},$$

so that $\tau = O(h^q)$ with $q \in (.50, 1.7)$. This explains why τ-extrapolation with $p = 2$ did not reduce the computational effort for the dam problem. The essential difficulty is of course that the irregularity of the discrete interface makes it difficult to obtain accurate estimates for τ.

Finally, in Table 6.3 we repeat the computations of Table 5.3 for a tolerance $\varepsilon^M = .0339$, the value of $\|\nabla \bar{u}^5\|_G$ in Table 6.1. We are thus comparing the performance

TABLE 6.3

Solution of the dam problem for $M = 5$ and $\varepsilon^M = .0339$ using PFASMD (*modification* M6), PFMG, *and projected* SOR.

	Method		
	PFMG	PFASMD	Projected SOR
Work units	6.41	9.64	60.0
$\|\nabla \bar{u}^M\|_G$	.0339	.0239	.0296
Execution time (seconds)	.404	.447	2.07

of PFAS (with Modification M6), PFMG and projected SOR for comparable errors. From Table 6.3, we see that PFMG is faster than projected SOR even when only low accuracy is required. PFAS and PFMG require comparable times, but PFMG gives much more information and is, therefore, preferable. PFMG also uses fewer work units than PFAS. This is significant because the number of work units used is independent of the computer. Furthermore, on the basis of experience with many problems, it can be said that the number of work units used does not vary greatly with the problem: for most operators $\mathscr{L}$, FMG requires only 5.4 work units.

We conclude this section with some remarks on the implementation of PFMG.

1. From Table 6.3 we see that the execution time per work unit of PFMG is greater than the comparable quantity for PFAS by a factor

$$\frac{.404}{6.41} \Big/ \frac{.447}{9.64} = 1.36.$$

This additional overhead is probably due to the cubic interpolation used by J_{k-1}^k, and could perhaps be reduced by better programming. When $\mathscr{L}$ is complicated, the additional overhead required by PFMG is relatively much less significant: it is only with a very simple operator like the 5-point Laplacian that the additional overhead is so expensive.

2. In PFMG one often need not have *any* storage for the finest grid G^M—not even external storage. The algorithm visits G^M only twice: at the beginning of the last cycle and at the end of the last cycle.

At the beginning of the cycle, the following operations are performed: interpolate (J_{M-1}^M); two G^M-projected sweeps; and residual transfer (I_M^{M-1} and S_M^{M-1}). All these operations can be made in one passage over G^M in such a way that only four columns of G^M are held in memory at one time. Each time a new column, say column i, is

created (by interpolation), a relaxation can be made in column $i-1$; then the second relaxation can already be made in column $i-2$ and the residuals from column $i-3$ can be transferred back to the coarse grid. Column $i-4$ can simultaneously be discarded (i.e., replaced by column i). After this visit to G^M, all the information is available (in $\bar{F}^{M-1}$ and $\bar{u}^{M-1}$) to solve the G^{M-1} problem to the truncation level of G^M.

The final return to G^M (which would require the storage of the previous values of U^M) is made in order to obtain the solution on G^M rather than on G^{M-1}, but it does not improve its pointwise accuracy. If one is interested only in knowing some functionals of the solution, these can be calculated without having the final solution on G^M. To approximate a functional $\mathscr{H}(U)$, for example, one computes $\mathscr{H}(\bar{u}^{M-1})+\sigma_M^{M-1}$, where $\sigma_M^{M-1}=\mathscr{H}(\bar{u}^M)-\mathscr{H}(I_M^{M-1}\bar{u}^M)$, $\bar{u}^{M-1}$ is the final solution on G^{M-1}, and $\bar{u}^M$ is the last solution on G^M before switching back to G^{M-1}. Clearly, σ_M^{M-1} can be calculated during the above-mentioned passage on G^M. Note that σ_M^{M-1} is a "relative truncation correction", similar to τ_M^{M-1}. It makes the approximation $\mathscr{H}(\bar{u}^{M-1})+\sigma_M^{M-1}$ correct to the G^M truncation level. $\mathscr{H}$ need not be a linear functional.

7. Conclusions and recommendations.

1. Multigrid methods can easily be adapted to handle linear complementarity problems arising from free boundary problems.

2. Multigrid methods are superior to projected SOR and modified block SOR (see Tables 5.3 and 6.3).

3. For high accuracy solutions of the discrete LCP, one should use PFASMD (see Tables 5.1 and 5.2).

4. For solutions which are accurate to within truncation error, one should use PFMG with no modifications (see Tables 6.1, 6.2, and 6.3).

Finally, we conclude with some comments suggesting possible future applications of multigrid methods to complementarity problems:

1. For equalities, experience has shown that multigrid methods are as efficient for problems where $\mathscr{L}$ is nonlinear as for problems where $\mathscr{L}$ is linear.

2. Experience from equalities indicates that with similar efficiency (just a few more work units), one can solve much more difficult problems, such as problems in which the coefficients of $\mathscr{L}$ vary by orders of magnitude (e.g., large variations in the diffusivity of the dam). In such cases SOR and other methods converge very slowly. See Alcouffe et al. [1980].

3. The truncation error near a discrete interface cannot be reduced by using higher order approximations, because the second derivatives are usually discontinuous. A good way to improve the approximation would be to use finer mesh sizes near the discrete interface. This can be combined very effectively effectively with the multigrid process (see Brandt [1979, § 3]). In fact, a vast improvement is to be expected if τ-extrapolation is used *together* with local refinements. Fine levels will then be used only near the interface.

4. It would be possible to use a parallel processor, in which case the Gauss–Seidel iterations would be performed using the red–black ordering of the grid points (Brandt [1980a], Foerster et al. [1980], Cryer et al. [1981]).

5. Although the numerical results for PFAS and PFMG are convincing, it would be desirable to obtain a rigorous proof of convergence, such as is available for the projected SOR method.

Acknowledgment. We thank Professor R. Sacher for making available a copy of his program for solving LCP's using the modified block SOR algorithm of Cottle and Sacher, and for his comments on an early version of this report.

REFERENCES

R. ALCOUFFE, A. BRANDT, J. E. DENDY, JR. AND J. W. PAINTER (1980), *The multi-grid methods for the diffusion equation with strongly discontinuous coefficients*, Report LA-UR-80-1463, Los Alamos Scientific Laboratory, Los Alamos, NM.

C. BAIOCCHI (1971), *Sur un problème à frontière libre traduisant le filtrage des liquides à travers des milieux poreux*, Comptes Rendus Acad. Sci. Paris Ser. A, 273, pp. 1215–1217.

——— (1978), *Free boundary problems and variational inequalities*, Technical Summary Report 1883, Mathematics Research Center, University of Wisconsin, Madison.

M. L. BALINSKI AND R. W. COTTLE (1978), *Complementarity and Fixed Point Problems*, North-Holland, Amsterdam.

J. BEAR (1972), *Dynamics of Fluids in Porous Media*, American Elsevier, New York.

A. BRANDT (1977), *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31, pp. 333–390.

——— (1979), *Multilevel adaptive techniques* (MLAT) *for singular-perturbation problems*, in Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. H. Miller, eds., Academic Press, New York, pp. 53–142.

——— (1980), *Stages in developing multigrid solutions*, in Numerical Methods for Engineering, E. Absi, R. Glowinski, P. Lascaux and H. Veysseyre, eds., Dunod, Paris, pp. 23–44.

———(1980a), *Multigrid solvers on parallel computers*, in Elliptic Problem Solvers, M. Schulz, ed., Academic Press, New York.

A. BRANDT AND C. W. CRYER (1980), *Multigrid algorithms for the solution of linear complementarity problems arising from free boundary problems*, Technical Summary Report 2131, Mathematics Research Center, University of Wisconsin, Madison.

A. BRANDT AND N. DINAR (1979), *Multi-grid solutions to elliptic flow problems*, in Symposium on Numerical Solution of Partial Differential Equations, S. V. Parter, ed., Academic Press, New York, pp. 53–147.

F. BREZZI AND G. SACCHI (1976), *A finite element approximation of variational inequalities related to hydraulics*, Calcolo, 13, pp. 259–273.

J. CEA, R. GLOWINSKI AND J. C. NEDELEC (1974), *Application des méthodes d'optimisation, de differences et d'éléments finis à l'analyse numérique de la torsion élasto-plastique d'une barre cylindrique*, in Approximation et méthodes iteratives de resolution d'inéquations variationelles et de problèmes non linéaires, Cahier de l'IRIA, 12, pp. 7–138.

G. CIMATTI (1977), *On a problem of the theory of lubrication governed by a variational inequality*, Appl. Math. Optim., 3, pp. 227–242.

R. W. COTTLE (1974), *Computational experience with large-scale linear complementarity problems*, Technical Report SOL 74-13, Systems Optimization Laboratory, Department of Operations Research, Stanford Univ., Stanford, CA.

R. W. COTTLE, F. GIANNESSI AND J. L. LIONS, eds. (1980), *Variational Inequalities and Complementarity Problems*, John Wiley, New York.

R. W. COTTLE, G. H. GOLUB AND R. S. SACHER (1978), *On the solution of large, structured linear complementarity problems: The block partitioned case*, Appl. Math. Optim., 4, pp. 347–363.

R. W. COTTLE AND R. S. SACHER (1977), *On the solution of large, structured linear complementarity problems: The tridiagonal case*, Appl. Math. Optim., 3, pp. 321–340.

C. W. CRYER (1971), *The solution of a quadratic programming problem using systematic overrelaxation*, SIAM J. Control, 9, pp. 385–392.

——— (1971a), *The method of Christopherson for solving free boundary problems for infinite journal bearings by means of finite differences*, Math. Comp., 25, pp. 435–444.

——— (1976), *A survey of steady state porous flow free boundary problems*, Technical Summary Report 1657, Mathematics Research Center, University of Wisconsin, Madison.

——— (1977), *A bibliography of free boundary problems*, Technical Summary Report 1793, Mathematics Research Center, University of Wisconsin, Madison.

——— (1980), *The solution of the axisymmetric elastic-plastic torsion of a shaft using variational inequalities*, J. Math. Anal. Appl., 76, pp. 535–570.

——— (1980a), *Successive overrelaxation methods for solving linear complementarity problems arising from free boundary problems*, in Proceedings, Seminar on Free Boundary Problems, Pavia, October 1979, E. Magenes, ed., Istituto Nazionale di Alta Matematica Francesco Severi, Rome, vol. 2, pp. 109–131.

C. W. CRYER, P. M. FLANDERS, D. J. HUNT, S. F. REDDAWAY AND J. STANSBURY (1981), *The solution of linear complementarity problems on an array processor*, Technical Summary Report 2170, Mathematics Research Center, University of Wisconsin, Madison; J. Comput. Phys., to appear.

G. DUVAUT AND J. L. LIONS (1976), *Inequalities in Mechanics and Physics.* Dunod, Paris.

R. S. FALK (1974), *Error estimates for the approximation of a class of variational inequalities*, Math. Comp., 28, pp. 963–971.

H. FOERSTER, K. STUEBEN AND U. TROTTENBERG (1980), *Non-standard multigrid techniques using checkered relaxation and intermediate grids*, in Elliptic Problem Solvers, M. Schulz, ed., Academic Press, New York.

R. GLOWINSKI (1971), *La méthode de rélaxation*, Rend. Mat., 14, pp. 1–56.

——— (1978), *Finite elements and variational inequalities*, Technical Summary Report 1885, Mathematics Research Center, University of Wisconsin, Madison.

R. GLOWINSKI, J. L. LIONS AND R. TREMOLIERES (1976), *Analyse numérique des inéquations variationnelles*, Dunod, Paris.

D. KINDERLEHRER AND G. STAMPACCHIA (1980), *An Introduction to Variational Inequalities and Their Applications*, Academic Press, New York.

H. LANCHON (1974), *Torsion élastoplastique d'un arbre cylindrique de section simplement ou multiplement connexé*, J. Mécanique, 13, pp. 267–320.

J. A. NITSCHE (1975), *L-infinity convergence of finite element approximations*, in Mathematical Aspects of Finite Element Methods, Lecture Notes in Mathematics 606, Springer, Berlin.

R. S. VARGA (1962), *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0009 $01.25/0

NUMERICAL STUDY OF INCOMPRESSIBLE SLIGHTLY VISCOUS FLOW PAST BLUNT BODIES AND AIRFOILS*

A. Y. CHEER†

Abstract. A grid-free numerical method is used to simulate incompressible flow at high Reynolds numbers. The numerical method simulates the flow inside the boundary layer by vortex sheets and the flow outside this layer by vortex blobs. The algorithm produces a smooth transition between the sheets and the blobs.

The accuracy of this hybrid numerical method is tested in several numerical experiments. In the first experiment, the algorithm is used to simulate slightly viscous flow past a circular cylinder. In the second experiment, the algorithm is used to simulate flow past a Joukowski airfoil at various angles of attack. In the latter case, the computations simulating the flow at the airfoil's trailing edge do not "blow-up". In both experiments, the calculated flow and its functionals (such as lift and drag coefficients) are in good agreement with both theoretical results and wind tunnel experiments.

Key words. Reynolds number, vortex sheets, vortex blobs, boundary layer, random walk, Joukowski airfoils, lift coefficient, drag coefficient

1. Introduction. The Navier–Stokes equations, which describe viscous fluid flows, are difficult to solve numerically, especially at large Reynolds numbers. For example, a method based on a grid has the disadvantage that the mesh width must decrease as the Reynolds number R increases. Consequently, at very large Reynolds numbers, a very fine grid must be imposed or else the numerical viscosity due to the grid will swamp the effects of the true viscosity as represented by the Reynolds number. In this paper we use a grid-free numerical method to obtain numerical solutions to the problems of flow past a circular cylinder and flow past airfoils at varying angles of attack.

Consider a flow of a fluid of small viscosity ν past a flat plate started impulsively from rest. Initially, the flow is irrotational and without circulation. The initial impulse produces a vortex sheet coincident with the solid surface. This vorticity immediately begins to diffuse, and is eventually convected downstream. After a short time t there is a boundary layer whose thickness is of order $(\nu t)^{1/2}$. In our study, the solution to the flow inside and outside this boundary layer is considered separately. The two solutions are then patched together at the edge of the boundary layer.

In this paper, flow past an obstacle started impulsively from rest is simulated by a hybrid numerical method which couples the method of the random vortex sheets with the method of the random vortex blobs. The random vortex blobs method presented by Chorin in 1973 [5] is a grid-free numerical method where the nonlinear terms of the equations are studied through inviscid interactions between vortex blobs; the effects of viscosity are studied through use of the relationship between diffusion and random walks; and the no-slip condition is satisfied via a vorticity generation algorithm. In the random vortex sheet method presented by Chorin in 1978 [6], vortex sheet elements are used near the boundary to solve the Prandtl boundary layer equations. This method is also grid-free and has the advantage that the interaction between vortex sheet elements are not singular.

In the hybrid algorithm introduced by this paper the flow inside the boundary layer is approximated by the vortex sheet method, while the flow outside the boundary

* Received by the editors July 15, 1981, and in final revised form September 27, 1982. This work was partially supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Engineering, Mathematical and Geosciences Division of the U.S. Dept. of Energy under contract W-7404-ENG-48, and in part by the Office of Naval Research, under contract N00014-76-C-0316.

† Lawrence Berkeley Laboratory and Department of Mathematics, University of California, Berkeley, California 94720.

layer is approximated by the vortex blob method. These two methods are coupled at the edge of the boundary layer. In this coupling, we replace the vortex blobs with vortex sheets near the boundary, thereby eliminating the problem of convergence of the vortex blobs near the boundary of the obstacle. The interaction between the blobs and sheets is not singular, and thus poses no additional complications.

To test the accuracy of this hybrid algorithm, we applied it to two problems: the problem of flow past a circular cylinder, and the problem of flow past an airfoil at varying angles of attack. Comparison of our results with those obtained by physical experiments indicate that we have a good model for simulating viscous fluid flow in two-dimensions. For our hybrid method, new computational elements are introduced when we satisfy the boundary conditions, and the total number of computations at each time step is of order $O(n^2)$, where n is the total number of computational elements.

2. Problem I: The cylinder problem. Consider a circular cylinder of radius 1 and immersed in an incompressible fluid of density 1. Consider two frames of reference, frame A (Fig. 1A) with the origin fixed at the center of the cylinder and frame B with the origin fixed relative to the fluid (Fig. 1B) and coincident with frame A prior to

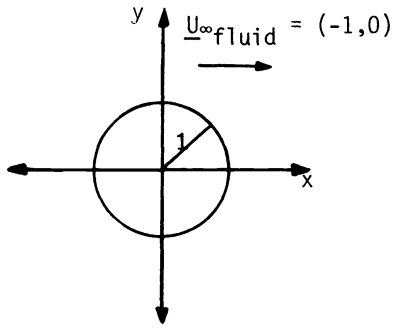


FIG. 1A.

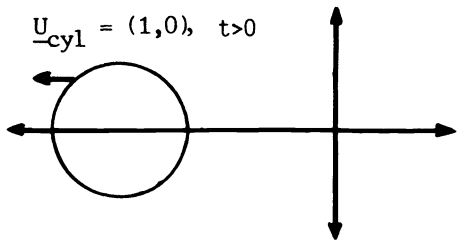


FIG. 1B.

time $t = 0$. At time $t = 0$, the cylinder is started impulsively from rest with a velocity of 1 (Fig. 1B). The direction of the flow of the fluid at infinity in frame A is therefore $(-1, 0)$. We shall consider the development of the flow of the two-dimensional cross section of this problem.

The equations of motion written in vorticity form are

$$\xi_t + (\mathbf{U} \cdot \nabla)\xi = R^{-1}\Delta\xi, \tag{1}$$

$$\Delta\psi = -\xi, \tag{2}$$

$$u = \psi_y, \qquad v = -\psi_x, \tag{3}$$

where $\mathbf{U} = (u, v)$ is the velocity field, $\xi = \operatorname{curl} \mathbf{U}$ is the vorticity, t represents the time, ψ is the stream function, $\Delta = \nabla^2$ is the Laplace operator, $z = (x, y)$ is the position vector, and R is the Reynolds number. The no-slip boundary conditions we need to satisfy are:

$$\begin{aligned} &\mathbf{U} \cdot \boldsymbol{\tau} = 0 \quad \text{on the boundary } \partial D, \ \boldsymbol{\tau} \text{ tangent to } \partial D, \\ &\mathbf{U} \cdot \mathbf{n} = 0 \quad \text{on } \partial D, \text{ where } n \text{ is normal to } \partial D. \end{aligned} \tag{4}$$

Consider the two-dimensional flow streaming steadily past a circular cylinder with no slip at the solid surface of the cylinder. Due to the no-slip boundary condition, vorticity is created at the solid surface. This vorticity diffuses and is convected downstream, producing a layer of large vorticity adjacent to the solid surface. Through this (boundary) layer the tangential velocity falls from its value in the main stream to zero at the solid surface.

In the case when the curvature of the boundary of the obstacle does not change abruptly,[1] the motion of the fluid in the layer adjacent to the boundary is described by the boundary layer equations (for theoretical analysis, see Chorin, Hughes, McCracken, and Marsden [7]). The Prandtl boundary layer equations for two-dimensional incompressible fluid flow written in vorticity form are

$$\xi_t + (\mathbf{U} \cdot \nabla)\xi = \nu\xi_{yy}, \tag{5}$$

$$\xi = -u_y, \tag{6}$$

$$u_x + v_y = 0, \tag{7}$$

with the following boundary conditions:

$$\mathbf{U} = (u, v) = (0, 0) \quad \text{at } y = 0, \tag{8}$$

$$u(x, y = \infty) = \mathbf{U}_\infty(x), \tag{9}$$

where $\mathbf{U} = (u, v)$ is the velocity vector with u tangential and v normal to the boundary, ξ is the vorticity, ν is the viscosity, and ∇ is the gradient operator.

3. The random vortex methods in brief.

A. The following is an algorithm for using the random vortex blob method to approximate the solution to (1), (2), and (3), and boundary conditions (4). For details of this numerical method see Chorin [5]. For theoretical analysis see Hald [11] and [12], Hald and DelPrete [13], and Majda and Beale [2].

Step (i). The vortices in the flow move according to the discrete approximation to the Euler equations.

Euler equations.

$$\xi_t + (\mathbf{U} \cdot \nabla)\xi = 0,$$

$$\Delta\psi = -\xi,$$

$$u = \psi_y, v = -\psi_x,$$

[1] that is, when the boundary layer thickness is much smaller than the radius of curvature of the obstacle.

Discrete approximation.

$$x_i^{n+1} = x_i^n + \Delta t \cdot u_i^n,$$

$$y_i^{n+1} = y_i^n + \Delta t \cdot v_i^n,$$

where

$$-u_i^n = (2\pi)^{-1} \sum_{r_{ij}>\sigma} k_j \left(\frac{y_i^n - y_j^n}{r_{ij}^2} \right) + (2\pi)^{-1} \sum_{r_{ij} \leqq \sigma} k_j \left(\frac{y_i^n - y_j^n}{\sigma r_{ij}} \right), \tag{10}$$

$$v_i^n = (2\pi)^{-1} \sum_{r_{ij}>\sigma} k_j \left(\frac{x_i^n - x_j^n}{r_{ij}^2} \right) + (2\pi)^{-1} \sum_{r_{ij} \leqq \sigma} k_j \left(\frac{x_i^n - x_j^n}{\sigma r_{ij}} \right), \tag{11}$$

where

Δt = time step,

(x_i^n, y_i^n) = position of the ith vortex at time $t = n \cdot \Delta t$,

$r_{ij} = \sqrt{(x_i^n - x_j^n)^2 + (y_i^n - y_j^n)^2}$,

$r_{ij}^2 = (x_i^n - x_j^n)^2 + (y_i^n - y_j^n)^2$,

σ = cut off value to be determined in § 5 (the value of σ will be chosen so that the transition between vortex blobs and vortex sheets in the hybrid algorithm is smooth),

k_j = strength of the vortex blobs whose center is (x_j^n, y_j^n).

Step (ii). Viscosity is included by adding a random walk component to the discrete solution of Euler's equation (above). Random walks are used to approximate the solution to the diffusion equation

$$\xi_t = R^{-1} \Delta \xi, \qquad \xi = \xi(x, y, t),$$

where ξ is the vorticity and R the Reynolds number.

Thus the discrete approximations to (1), (2) and (3) are:

$$x_i^{n+1} = x_i^n + \Delta t u_i^n + \eta_1, \tag{12}$$

$$y_i^{n+1} = y_i^n + \Delta t v_i^n + \eta_2 \tag{13}$$

where η_1 and η_2 are independent random variables with a Gaussian distribution of mean zero and variance $2\Delta t/R$. For more discussion on random walk solutions to the diffusion equation see references [14] and [21].

Step (iii). The tangential boundary condition is satisfied by using a vorticity generation algorithm. In this algorithm, the amount of vorticity created on the boundary is exactly the amount that will satisfy the tangential boundary condition.

Step (iv). The normal boundary condition is satisfied by using the method of images which in the case of a circular cylinder is particularly simple. Consider a vortex of strength k situated at a point z_1 in the x, y-plane, and also outside a circular cylinder of radius a, centered at the origin. The image system consists of a vortex $-k$ at the inverse point $1/\bar{z}_1$ and a vortex k at the origin. The vortex k at z_1 together with its inverse vortex cancels exactly on the boundary, thus giving zero normal velocity. The vortex k at the origin is needed to satisfy conservation of circulation. (See Fig. 2.) For the hybrid algorithm outlined in § 4, the vortex in the center should be omitted to satisfy the condition of zero circulation at infinity.[2] For a discussion on this point see § 4.

[2] A. Leonard, personal communication August, 1982.

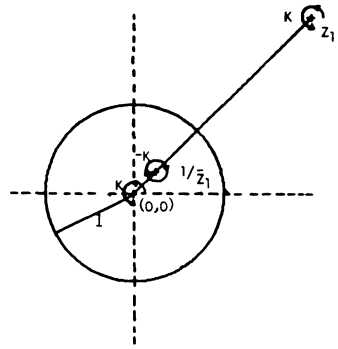


FIG. 2

Step (v). Time is advanced by Δt. The whole process, step (i)–(v) is repeated until the desired number of times is reached.

B. We now present the numerical algorithm using the random vortex sheet method to approximate the solution to (5), (6) and (7) satisfying boundary conditions (8) and (9). For details of this method, see Chorin [6] and Chorin and Marsden [4]. For documentation on the computer program implementing this method see Cheer [3].

Step (i). As in the previous case, first let the vortex sheets move according to the discrete approximation to the Euler equations.

Euler equations.

$$\xi_t + (\mathbf{U} \cdot \nabla)\xi = 0,$$

$$\xi = -u_y,$$

$$u_x + v_y = 0.$$

Discrete approximation.

$$x_i^{n+1} = x_i^n + \Delta t u_i^n,$$

$$y_i^{n+1} = y_i^n + \Delta t v_i^n,$$

where

$u_i^n = U_\infty(x_i^n) - \frac{1}{2}\xi_i - \sum_j \xi_j d_j$,
$d_j = 1 - |x_i^n - x_j^n|/h$,
$\sum$ is over all vortex sheets S_j such that $y_j^n > y_i^n$,
h = length of the sheet,
$v_i^n = -(I_1 - I_2)/h$,
$I_1 \cong U_\infty(x_i^n + h/2)y_i^n - \sum_j + \xi_j d_j^+ y_j^{*n}$,
$I_2 \cong U_\infty(x_i^n - h/2)y_i^n - \sum_j - \xi_j d_j^- y_j^{*n}$,

$$d_j^+ = 1 - \left(\left|x_i^n + \frac{h}{2} - x_j^n\right| \Big/ h\right),$$

$$d_j^- = 1 - \left(\left|x_i^n - \frac{h}{2} - x_j^n\right| \Big/ h\right),$$

$y_j^{*n} = \min(y_i^n, y_j^n)$,
$\sum_+$ is the sum over all vortex sheets S_j such that $0 \leqq d_j^+ \leqq 1$,
$\sum_-$ is the sum over all vortex sheets S_j such that $0 \leqq d_j^- \leqq 1$.

Step (ii). The effects of viscosity are included by adding to the y-component of the solution an independent random variable η_i drawn from a Gaussian distribution

of mean zero and variance $2\nu\Delta t$. Thus

$$x_i^{n+1} = x_i^n + \Delta t \cdot u_i^n, \tag{14}$$

$$y_i^{n+1} = y_i^n + \Delta t \cdot v_i^n + \eta_i. \tag{15}$$

Step (iii). First note that the boundary conditions $u(x, y) = \mathbf{U}_\infty(x)$ at the edge of the boundary layer and $v(x, y) = 0$ on the boundary are automatically satisfied. The remaining boundary condition $u(x, y) = 0$ on the boundary is satisfied by a vorticity creation algorithm. Vortex sheets are created on the boundary so that the resultant flow has $(u, v) = (0, 0)$. In this way, the transition from zero on the boundary to $\mathbf{U}_\infty(x)$ on the edge of the boundary layer is achieved.

Step (iv). Time is advanced by Δt. The procedure (i)–(iv) is repeated until the desired time is reached.

4. Hybrid numerical method applied to the circular cylinder problem.

1. First we divide the circle into M segments each of length $2\pi/M$. Let each segment be represented by its midpoint. For each of the M points on the body, there corresponds M-points on the edge of the boundary layer. The boundary layer is assumed to be of thickness $O(R^{-1/2})$. We call the M points on the edge of the boundary layer M'.

2. On each of the M' points we calculate the velocity contribution from both the freestream velocity and the vortex blobs already in the flow. Note that initially at time zero there are no vortex blobs in the flow, so that the only contribution is from the freestream velocity. The amount of vorticity at each of these M' points is $\xi_i = \text{curl}\,(u_i, v_i)$.

3. The amount of vorticity created at each point on the boundary is negative twice the difference between the amount of vorticity at the edge of the boundary layer and the amount already existing inside the boundary layer. Initially there are no vortex sheets inside the boundary layer, so the correct amount is twice the amount calculated on the edge of the boundary layer, i.e. $-2\xi_i$.

4. These newly created vortex sheets, and the vortex sheets already in the flow, are convected and diffused according to formulae (14) and (15). At time $t = 0$ the only sheets in the flow are the ones created in step 3 above. Since these and every subsequent newly created sheet are on the boundary, they have velocity (0, 0) and it follows that the only contribution to their motion initially is the random walk component of the flow. After applying formulae (14) and (15), we see that on the average one half of the newly created sheets will move away from the cylinder and the other half will move into the cylinder. By doing this, we satisfy the tangential boundary condition exactly, and achieve a smooth transition from zero on the boundary to the correct amount at the edge of the boundary layer. Note that at this point all of the boundary conditions for the boundary layer equations are satisfied.

5. Next we check to see if any of the sheets have moved or diffused outside of the boundary layer, and if so we turn them into vortex blobs. Since sheets and blobs are determined by the same parameters, this change is simple and straightforward, i.e., (x_i, y_i, ξ_i) becomes (x_i, y_i, k_i), where $k_i = h \cdot \xi_i$. Thus, when a vortex sheet becomes a vortex blob, the only change is in the intensity of the point to satisfy conservation of circulation. To preserve antisymmetry, the sheets in the boundary layer (but not on the boundary) that flow into the object are reflected to their image points. If their images are inside the boundary layer, they remain sheets; otherwise, they are changed into blobs. Since we know a priori that the sheets with sharp gradients are close to the point of separation, one can test to see if sheets have velocity gradient u/v greater than some chosen parameter determined by the geometry of the obstacle in the flow,

and if so, turn them into blobs. Note that the newly created vortex sheets do not follow any of the above rules.

6. We now calculate the velocity of the vortex blobs in the flow. We can, at the same time, take care of the normal boundary condition by modifying (10) and (11) to include the image points. With this modification, the complex potential for the circular cylinder problem satisfying $\mathbf{u} \cdot \mathbf{n} = 0$ is

$$w(z_i) = -\mathbf{U}_\infty\left(z_i - \frac{a^2}{z_i}\right) + \frac{1}{2\pi}\sum_{j=1}^{NN} ik_j\left[\log(z_i - \bar{z}_j) - \log\left(z_i - \frac{a^2}{\bar{z}_j}\right) + \log z_i\right],$$

where $a = 1$ is the radius of the circle and $\mathbf{U}_\infty = (-1, 0)$. Evaluating this expression, we get

$$\begin{aligned} u_i = 1 &- \frac{(x_i^2 - y_i^2)}{(r_i^2)^2} + \frac{1}{2\pi}\sum_{j1} k_j \frac{(y_i - y_j)}{(r_{ij})^2} + \frac{1}{2\pi}\sum_{j2} k_j \frac{(y_i - y_j)}{\sigma(r_{ij})} \\ &- \frac{1}{2\pi}\sum_{j1} k_j \frac{(y_i - y_j/r_j^2)}{(r_{ij}^*)^2} - \frac{1}{2\pi}\sum_{j2} k_j \frac{(y_i - y_j/r_j^2)}{\sigma(r_{ij}^*)} \\ &+ \frac{1}{2\pi}\sum_{j1} k_j \frac{y_i}{(r_i)^2} + \frac{1}{2\pi}\sum_{j2} k_j \frac{y_i}{\sigma(r_i)}, \end{aligned} \tag{16}$$

$$\begin{aligned} v_i = &- \frac{2x_i y_i}{(r_i^2)^2} - \frac{1}{2\pi}\sum_{j1} k_j \frac{(x_i - x_j)}{(r_{ij})^2} - \frac{1}{2\pi}\sum_{j2} k_j \frac{(x_i - x_j)}{\sigma(r_{ij})} \\ &+ \frac{1}{2\pi}\sum_{j1} k_j \frac{(x_i - x_j/r_j^2)}{(r_{ij}^*)^2} + \frac{1}{2\pi}\sum_{j2} k_j \frac{(x_i - x_j/r_j^2)}{\sigma(r_{ij}^*)} \\ &- \frac{1}{2\pi}\sum_{j1} k_j \frac{x_i}{(r_i)^2} - \frac{1}{2\pi}\sum_{j2} k_j \frac{x_i}{\sigma(r_i)}, \end{aligned} \tag{17}$$

where

$r_i = \sqrt{x_i^2 + y_i^2}$,
$(r_i)^2 = (x_i^2 + y_i^2)$,
$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$,
$r_{ij}^* = \sqrt{(x_i - x_j/r_j^2)^2 + (y_i - y_j/r_j^2)^2}$,
$(r_j)^2 = x_j^2 + y_j^2$,
σ = cut-off to be determined later,
$\sum_1$ is taken over all vortices such that $r_i, r_{ij}, r_{ij}^* > \sigma$,
$\sum_2$ is taken over all vortices such that $r_i, r_{ij}, r_{ij}^* \leqq \sigma$.

7. Vortex blobs move according to formulae (12) and (13) where the u and v velocities are modified in step 6 above. Transition from vortex blobs to vortex sheets follows a procedure similar to that used in the transition from sheets to blobs. A blob can become a sheet either if it flows back into the boundary layer or if it flows into the cylinder and its image point lies in the boundary layer. In the latter case, the point is reflected as a sheet. One can argue that if Δt were more refined, this point would end up in the boundary layer as it follows its trajectory. Thus, when it flows into the object as a sheet, it will automatically be reflected in step 5 to preserve antisymmetry. Note that this cannot be done if the image point is not in the boundary layer for we only have this antisymmetry property for the boundary layer equations. See Chorin [8].

8. Advance the time step and repeat steps 2–8 until the desired time is reached.

Recall from § 3 that in order to satisfy the normal boundary condition on the circular cylinder we add to our solution the solution of the potential equation with

boundary condition $-\mathbf{u}\cdot\mathbf{n}=0$. Since the solution to the Neumann problem is unique up to a constant, we choose the constant that will minimize the total number of computational elements introduced into the flow. For the cylinder problem, the solution where the image system includes the vortex in the center is the one that minimizes computation. In the random vortex blob method this choice does not violate the condition at infinity because, if the circulation at infinity is nonzero, the numerical algorithm will immediately create vortex blobs on the boundary with the appropriate strengths so that the condition at infinity is satisfied.

In the hybrid algorithm vortex sheet elements, rather than vortex blobs, are created on the boundary. When calculating the motion of the vortex blobs, if we include the contribution of the vortex sheet elements in the boundary layer to their velocities then the above choice for the solution to the Neumann problem (with the vortex in the center) would be the best choice. However, if we choose to view each vortex sheet element as having an image sheet so that the error introduced by ignoring the contributions of sheets and their images is of the order of the displacement thickness, then the above choice is not valid. In this case, in order to save on computation by neglecting the contribution of the sheets and their images to the velocity of the vortex blobs, we must choose the Neumann solution to be the one that not only satisfy the Neumann boundary problem, but also fixes zero circulation at infinity. This corresponds to choosing the vortex at the center to have zero strength.

5. How to choose the cut-off value σ. We approach the problem of determining the value of σ in the following way. Consider a collection of vortex blobs. If these blobs are close to the boundary of the object, the effects of their interaction with the boundary should be the same as the effects of the interaction of the vortex sheets with the boundary. In other words, the cut-off should have the value that will give not only fast convergence, but also smooth transition from vortex sheets to vortex blobs.

Let us consider the following example in determining the cut-off. Let the line $y=0$ be a wall, and consider the flow to be in the upper half plane. The boundary layer thickness is $O(R^{-1/2})$. Let there be a vortex sheet of intensity ξ situated at point z_1 at the edge of the boundary layer. Also, let a vortex blob with the same circulation (i.e., of intensity ξh) be situated at the same point z_1. The image vortex is thus at $\bar{z}_1$. Let z_2 be the point on the boundary such that the segment $\overline{z_1 z_2}$ is normal to the boundary. If we choose $\sigma = h/\pi$, we can verify from formula (16) that the velocity at z_2 induced by the blob at z_1 and its image at $\bar{z}_1$ is

$$u_2 = 2\left(\frac{1}{2\pi}\xi h\frac{y-y_1}{\sigma r}\right) = \left(\frac{1}{\pi}\xi h\frac{1}{h/\pi}\right) = \xi, \qquad v_2 = 0.$$

Thus, as vortex blobs approach the edge of the boundary layer, their effects on the boundary coincide with the effects of the sheets on the edge of the boundary layer. We can, therefore, view the computational elements as sheets near the boundary and as blobs far away from the boundary. Hence, $\sigma = h/\pi$ is the value for the cut-off which is consistent with our hybrid numerical method.

6. Problem II: The airfoil problem. Let

$$\tilde{z} = f(z) = \frac{1}{(1+a)}\left(z+\frac{1}{z}\right)e^{-i\alpha}, \tag{18}$$

where $z = -a+(1+a)\,e^{-i\theta}$, $0 \leqq \theta \leqq 2\pi$, $a \in \mathbb{R}$. (See Fig. 3.) The parameter a controls the shape of the airfoil, and the parameter α controls the angle of attack. We partition the airfoil into M segments each of length h_i, $i = 1, 2, \cdots, M$. As in the case of the

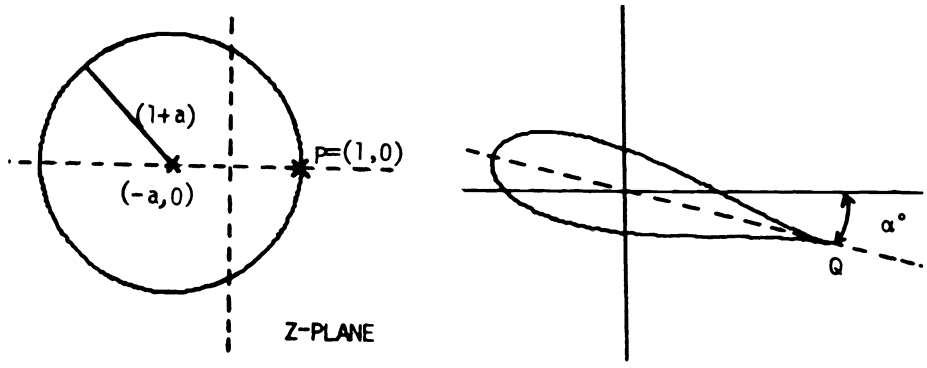


FIG. 3

cylinder, the airfoil is immersed in a fluid of density 1. At time $t = 0$, the airfoil is started impulsively from rest with a velocity magnitude of 1 (see Fig. 1B). The velocity of the fluid at infinity relative to the airfoil is $(-1, 0)$ (see Fig. 1A).

A slight modification of the hybrid algorithm presented in § 4 is used to simulate the flow past the airfoil. The modifications are:

1. Numerical calculations for the potential component of the velocity are carried out in the circle plane, and Routh's theorem is applied to obtain the corresponding velocity in the airfoil plane. For more information on Routh's theorem see Lin [18]. If the point whose velocity is required is also a vortex position the following correction term must be added:

$$\frac{-k_0}{2\pi i}\left(\frac{\frac{1}{2}W''(z_0)}{W'(z_0)}\right),$$

where k_0 is the strength of the vortex point situated at z_0, and $W(z)$ is the conformal mapping from the airfoil plane into the circle plane. For more information on the derivation of this correction term see Clements [9].

2. Treatment of the sheets at the trailing edge of the airfoil is given special consideration. We assume that vortex sheets lie in the direction of the streamlines. Since the streamlines of the flow do not bend over the trailing edge of the airfoil, vortex sheets lying in the direction of the streamlines should not be allowed to bend over the trailing edge. To calculate the contribution of a vortex sheet to the boundary condition, consider the situation in Fig. 4 below. We project the sheet situated at $f(z_i)$ onto the boundary of the object in the direction normal to the boundary. Because this sheet casts no shadow onto the lower part of the boundary the contribution of this sheet to the boundary condition is restricted to the top part of the airfoil. Also, there is a portion of the sheet that does not cast a shadow onto the boundary of the

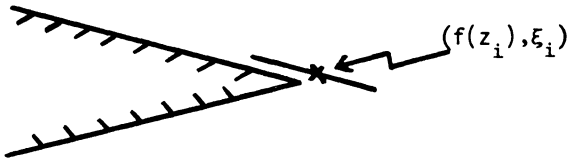


FIG. 4

object and consequently this portion of the sheet has no effect on the boundary condition of the airfoil. The only contribution is therefore restricted to the portion of the sheet that when projected normally casts a shadow onto the boundary.

3. The creation of vorticity and random walk components of the hybrid algorithm were calculated in the airfoil plane.

7. Numerical experiments for flow past a circular cylinder.

A. Numerical parameters. In all of the numerical calculations done on the circular cylinder problem, the boundary of the circle was divided into $M = 20$ pieces, each of length $h = 2\pi/M$. The radius r of the circle was equal to 1.

The cylinder was impulsively set into motion at time $t = 0$, where t was measured in nondimensional units ($t = t^* \cdot (U/r)$, U = freestream velocity, r = radius). Δt was chosen to be 0.2. Numerical experiments were done with $\Delta t = 0.2$ and $\Delta t = 0.1$ and with all the other parameters kept fixed. Refining Δt did not improve the calculations significantly, i.e., when we compared the calculated lift and drag coefficients with these choices of Δt, the average value of the lift and drag coefficient was changed by less than one percent.

For reasons of economy, we chose $\xi_{\max}$ as large as possible, but not so large that we would lose the main features of the problem. After some experimentation $\xi_{\max} = 1.0$ was chosen. Note that a sheet of length h and strength $\xi_{\max}$ will eventually become a blob of strength $= h \cdot \xi_{\max}$. For the values of h and $\xi_{\max}$ chosen as above, this means that the value 0.31415 is the maximum strength for each blob.

Physical experiments show that a flow started impulsively from rest goes through a long transitional period before it becomes fully developed. In our numerical experiments we found that at time $t = 8$, the flow was leaving the transitional period and entering into the stage of fully developed flow. For Reynolds number 1,000, the numerical calculations were carried out from time $t = 0$ to time $t = 8$. For Reynolds number 2,000, the calculations were carried out from $t = 0$ to $t = 11$. In this case, the calculations were refined by taking $\xi_{\max}$ to be 0.2 instead of 1.0 at time $t = 8$. This refinement corresponds to creating more computational elements of smaller strength. Changing $\xi_{\max}$ to an even smaller quantity, say $\xi_{\max} = 0.1$, after time $t = 8$ did not improve the results significantly. Again, the results corresponding to $\xi_{\max} = 0.1$ and $\xi_{\max} = 0.2$ after time $t = 8$ differ from each other by less than one percent.

In all of the numerical calculations presented in this paper, the vortex in the center was included when using the method of images. The circulation at infinity due to this inclusion was small. For example, the circulation at infinity for Reynolds number $\mathrm{Re} = 2{,}000$ was bounded by ± 0.5 except for a few cases. The average circulation at infinity due to this inclusion was -0.13. For this reason, the flows were not recalculated.

B. Development of the flow. Numerical experiments for flow past a circular cylinder started impulsively from rest were performed for Reynolds numbers 1,000 and 2,000. The flow development at these two Reynolds numbers differs only slightly. Figures 5 through 8 show different stages of the development of the flow with $\xi_{\max} = 1.0$ until time $t = 8$ then $\xi_{\max} = 0.2$; $h = 2\pi/M$, $M = 20$, $\sigma = h/\pi$, $R = 2{,}000$, and $\Delta t = 0.2$. This development compares well with physical experiments (see [1] and [23]).

Initially after impulsive start, diffusion outweighs convection. However, after a short time, the convection of vorticity becomes more significant than the diffusion of vorticity, especially in the directon of the flow parallel to the cylinder. The vorticity created to satisfy the boundary condition is carried by this convection to the rear of the cylinder. This vorticity created is negative in sign on the upper surface and positive

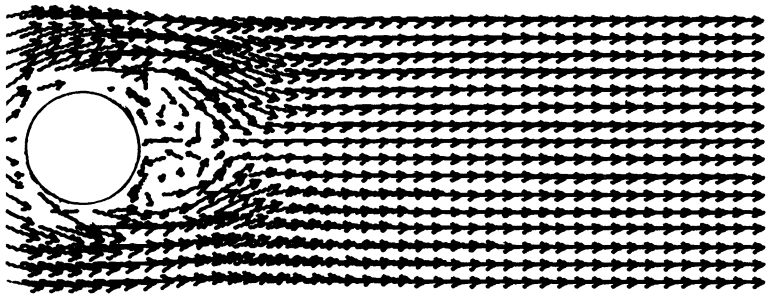


FIG. 5. *Time* $t = 2$.

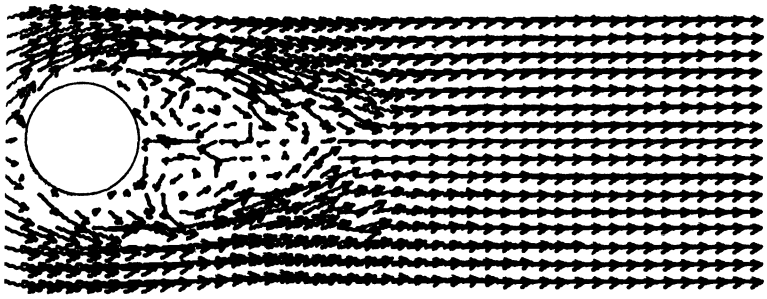


FIG. 6. *Time* $t = 5$.

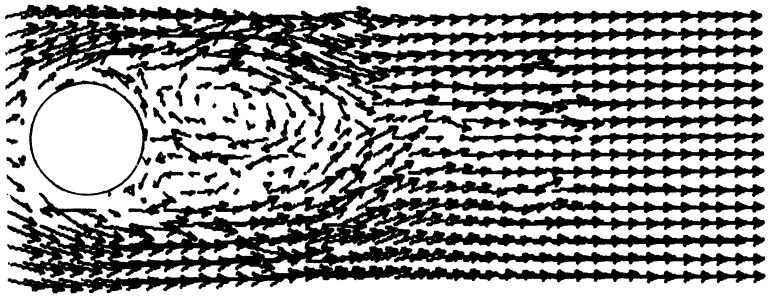


FIG 7. *Time* $t = 8$.

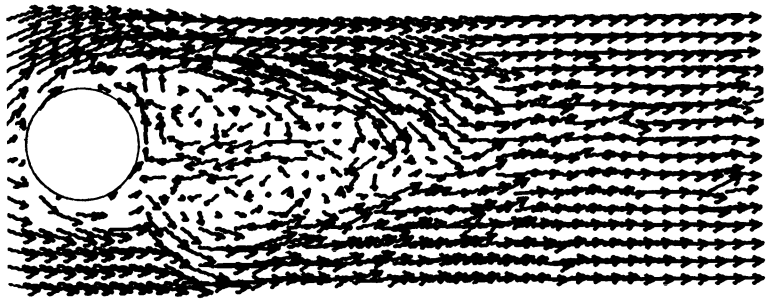


FIG. 8. *Time* $t = 11$.

on the lower surface. Ultimately there is more vorticity of each sign at the rear of the cylinder than is needed to satisfy the no-slip condition there and backflow is induced near the surface. The backflow counters the forward-moving fluid and deflects it away from the rear of the cylinder. Most of the fluid passing close to the cylinder appears to gather itself into two discrete lumps, or eddies, at the rear of the cylinder. (See Fig. 5 corresponding to time $t = 2$.) At time $t = 2$, we see that the eddies in the wake of the cylinder are just beginning to take form. By time $t = 5$, we see that the eddies are well-developed and the points of separation are about 85° from the forward stagnation point on both sides. The streamlines leave the body tangentially at the points of separation. (See Fig. 6.) Furthermore, new eddies are being formed while the original ones have convected downstream. The region enclosed by the two separating streamlines grows larger and becomes even larger than the cylinder itself. By time $t = 8$, the eddies created earlier are merging due to diffusion. (See Fig. 7.) At time $t = 11$, the flow is asymmetric and the points of separation can be estimated from the graph to be around 78° and 115° from the forward stagnation point.

Ultimately, one of the eddies at the rear of the cylinder will break loose from the cylinder and move downstream. The remaining eddy of the opposite rotation will consequently become larger, and eventually this eddy will also shed. The shedding of the eddies causes asymmetry in the flow pattern and in the two points of separation, and causes dips and rises in the lift and drag coefficients. The numerical experiments we ran were stopped on or before time $t = 11$. This is not enough time for the flow to develop a vortex street with more than one oscillation. Hence, Strouhal numbers were not calculated in this study.

C. Lift and drag coefficients. The lift and drag coefficients are calculated using the formula

$$L - iD = \rho \frac{\partial}{\partial t} \sum_{j=1}^{N} \Gamma_j \left(z_j - \frac{1}{z_j^*} \right) \tag{19}$$

with

$$D = C_D \left(\frac{\rho U^2}{2} \right) A, \qquad L = C_L \left(\frac{\rho U^2}{2} \right) A.$$

Here L is the value of the lift, D is the drag, Γ_j is the strength of the vortex point situated at $z_j = (x_j, y_j)$, z_j^* is the image vortex, A is the characteristic length, and ρ = density = 1. The derivation of this formula can be found in [10].

The drag coefficients for Re = 1,000 and Re = 2,000 are presented in Figs. 10 and 9 respectively. The drag coefficient for Reynolds number 2,000 starts at 1.017, drops down to 0.947, then climbs sharply up to 1.407. This corresponds to the impulsive start of the cylinder. Averaging over time from $t = 0$ to $t = 11$, we get an average drag coefficient of $C_D = 1.01$. This is slightly higher than the average drag given by experiments. If the values of the first peak in the graph of the drag coefficient (corresponding to impulsive start) is excluded, we get after averaging from $t = 3$ to $t = 11$, $C_D = 0.95$. This value is in excellent agreement with experimental values. (See Fig. 11.) Figure 11 is a plot of the measured drag coefficient for a circular cylinder as a function of Reynolds number. See Batchelor [1, p. 341], Prandtl and Tietjens [23, p. 97] and Schlichting [25, p. 17]. For Reynolds number 1,000, we get a similar effect. The drag coefficient starts at 1.158, drops down to 0.946, then climbs up to a maximum of 1.419. The average drag coefficient averaging from $t = 0$ to $t = 8$ is 1.10. Averaging from $t = 3$ to $t = 8$ we get an average of 1.04 which again is in excellent agreement

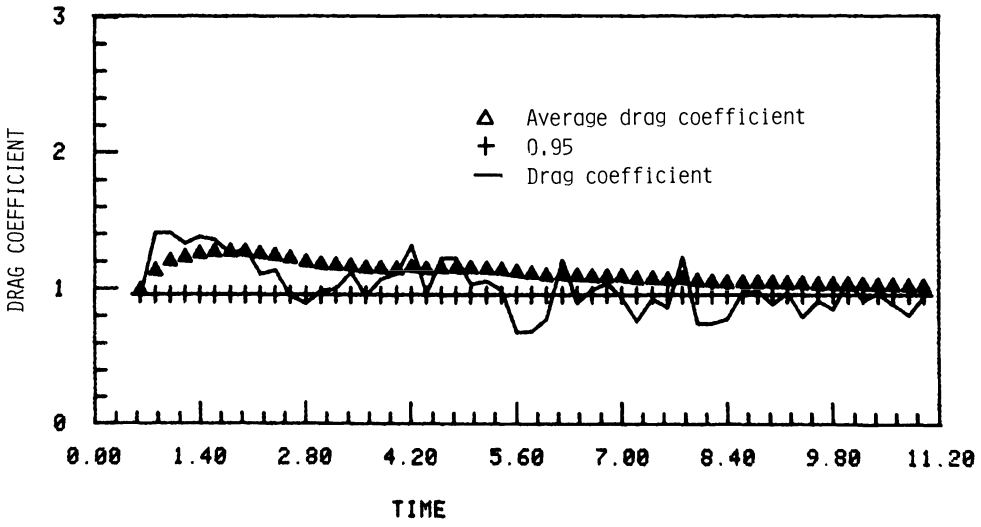


FIG. 9.

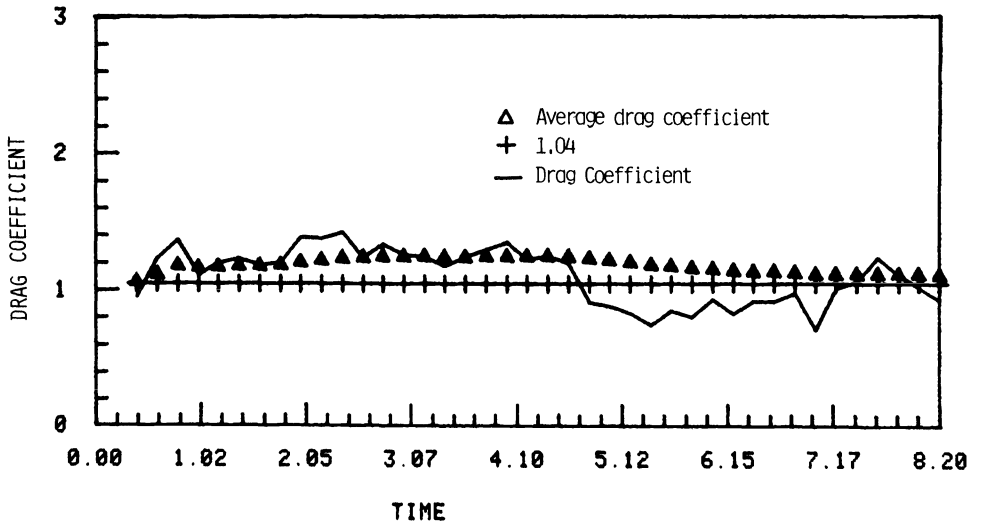


FIG. 10.

with experimental data. (See Fig. 11.) The lift coefficient for Re = 2,000 starts at zero, rises to 0.1, and then becomes slightly negative for most of the time until $t = 5.6$. It then oscillates with varying amplitude staying almost always between ± 1, with an average which is slightly below zero at time $t = 11$, and with a variance of 0.100. The last dip in the lift coefficient corresponds to the changing flow pattern around the cylinder, and to the shedding of an eddy at the rear of the cylinder. (See Fig. 12.)

It takes less than three minutes of CPU-time on the CDC-7600 to model the development of the flow past a circular cylinder from rest to four diameter movements away. This includes the calculations of the numerical functionals. At time $t = 8$, the flow is modeled by approximately 500 computational elements. A study which

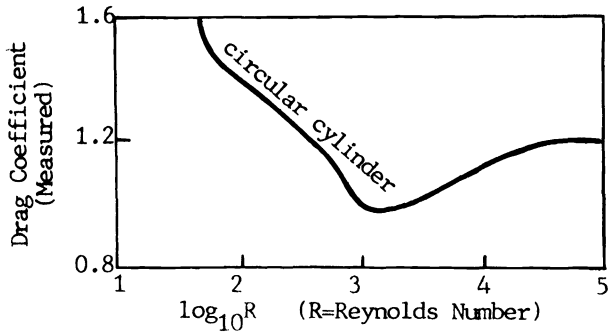


FIG. 11.

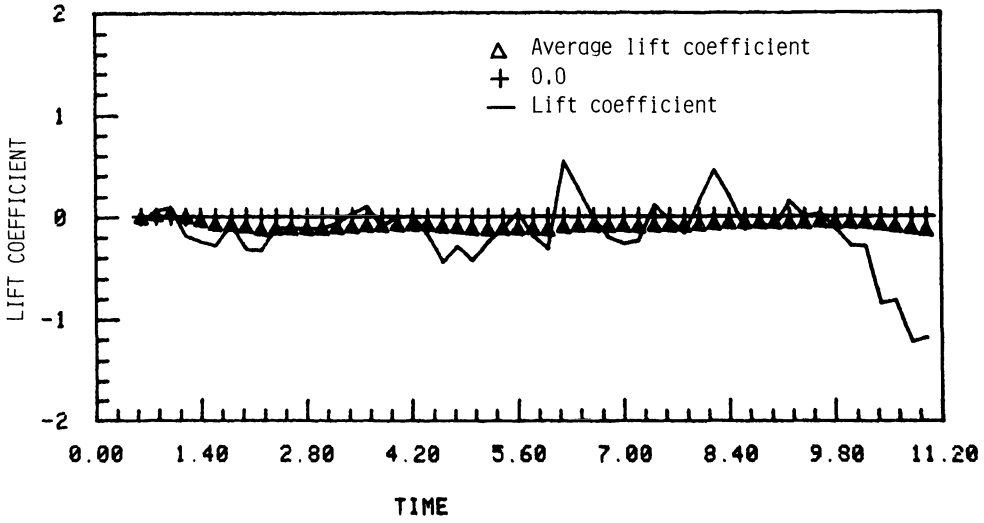


FIG. 12.

compares numerical methods other than ours to model flow past a circular cylinder can be found in Nahavandi and Chen [22].

8. Numerical experiments for flow past airfoils.

A. Numerical parameters. In calculating flow past an airfoil, the conformal transformation (18) was used. Note that the solution to the irrotational flow is not uniquely determined and therefore in calculating the flow past an airfoil, we can choose the potential flow solution to be the one that minimizes the work we must do. Suppose we choose an arbitrary solution, for example the one that gives large velocities at the trailing edge. To satisfy the no-slip condition on the boundary, large amounts of vorticity of the opposite sign must immediately be created in the vicinity of the trailing edge. As this vorticity is convected downstream, new amounts will be created to replace it. Eventually, we will reach a state where the amount of vorticity in the vicinity of the trailing edge cancels out the large contribution from the potential component of the flow. All this work can be avoided if we choose the potential flow solution to be the one that gives zero velocity at the trailing edge. This is done by adding the appropriate circulation term to the fundamental solution of the potential

equation. For any given orientation of the airfoil, this circulation should have the value such that the rear stagnation point is located at the sharp trailing edge. By choosing the circulation to have the value that will make the trailing edge a stagnation point, we have minimized the amount of work required to achieve steady flow, and have also enforced the Kutta condition at the trailing edge. For information on how to determine the value of this circulation term see Milne-Thomson [20].

Recall that numerical calculations for the potential component of the velocity were carried out in the circle plane, and Routh's theorem was applied to obtain the corresponding velocity in the airfoil plane. The creation of vorticity and the random walk components of the algorithm were calculated in the airfoil plane.

The numerical parameters for the airfoil problem were chosen to be the same as for the case of the problem of flow past a circular cylinder. The variable parameters in the problem of flow past an airfoil are a and α; a changes the thickness of the airfoil and α changes the angle of attack. In each case, the boundary of the cylinder was divided into $M = 20$ pieces, each of length $h = 2\pi/M$. Under the conformal mapping, the airfoil was therefore also divided into $M = 20$ pieces each of length $h_i = h \cdot |f'(z_i)|$. When a vortex sheet flowed out of the boundary layer, it became a vortex blob of strength equal to $h_i\xi_i$. $\xi_{\max}$ has value 1.0, $R = 1{,}000$ and $\Delta t = 0.2$.

B. Development of the flow.

Case $a = 0.25$, $\alpha = -\pi/12$, $\mathrm{Re} = 1{,}000$. We will only present the case where an airfoil tilted at an angle of $-\pi/12$ radians is started impulsively from rest.

Immediately after the impulsive start, the top point of separation separates at the trailing edge. The velocity on the average is greater above the body than below it. Therefore, different amounts of vorticity are created and shed from the top and from the bottom.

Backflow develops almost immediately and consequently the point of separation is pushed back up towards the front of the airfoil. At time $t = 1$, the top separation point is half way up the airfoil (See Fig. 13.) The bottom point of separation is very close to the trailing edge, and essentially remains there. The streamlines leave the points of separation tangentially. The flow at the trailing edge behaves very smoothly and leaves the body tangentially there.

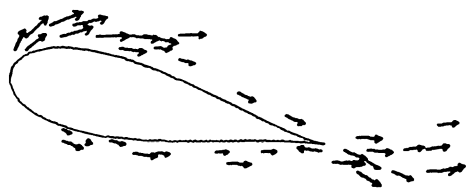


FIG. 13. *Time* $t = 1$.

The vorticity created at the boundary first diffuses, then is carried downstream by convection and finally is shed from the edge by the fluid. So, continual generation of vorticity at the boundary is necessary to satisfy the no-slip boundary condition. The deceleration of the fluid close to the body, and the acceleration of the fluid above, cause the fluid to collect into circular patterns in the area downstream from the top separation point. These circular patterns get larger and stronger as they roll down the

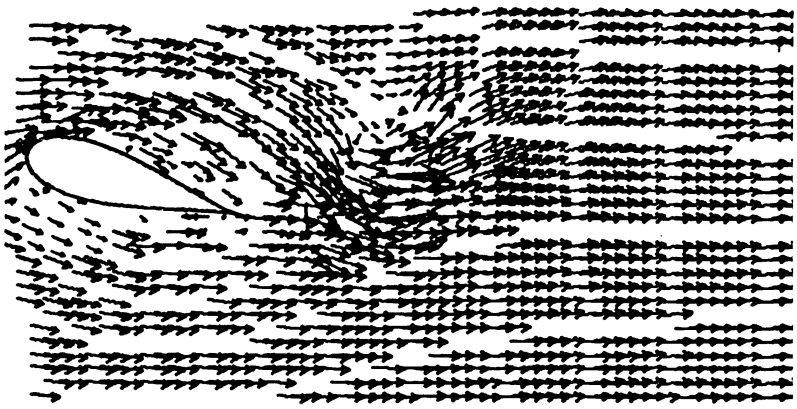


FIG. 14. *Time* $t = 2$.

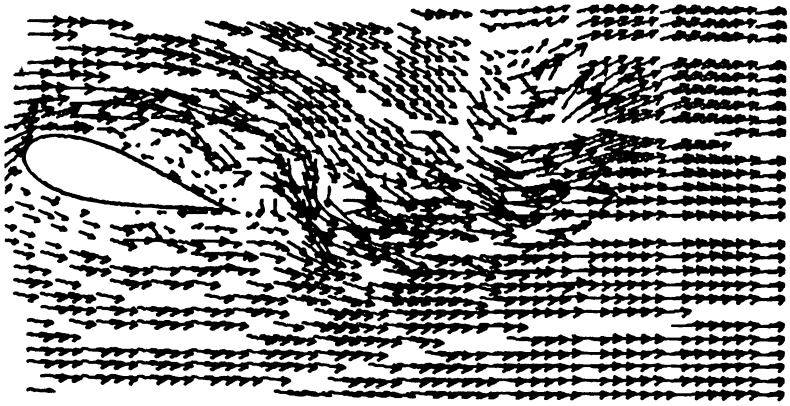


FIG. 15. *Time* $t = 4$.

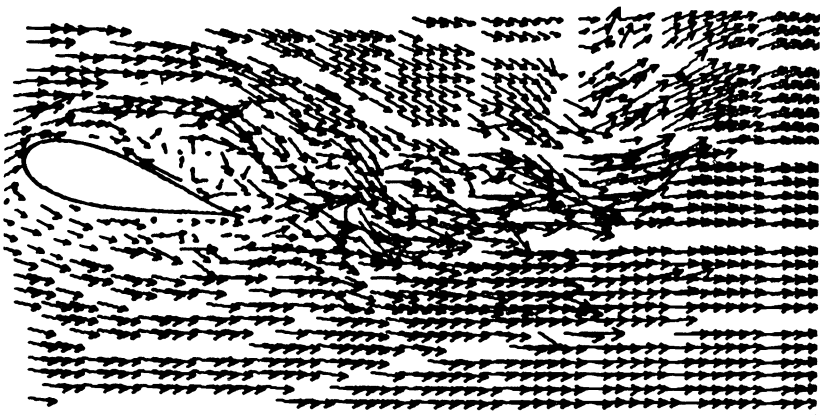


FIG. 16. *Time* $t = 5$.

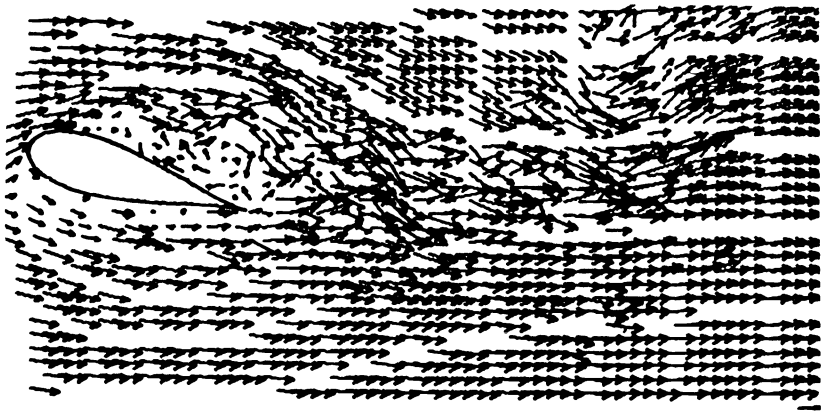


FIG. 17. *Time* $t = 5.4$.

airfoil. Eventually the vorticity in these circular eddies is shed off the tail end of the airfoil into the wake. There is not a continuous shedding of these eddies and therefore the concentration of the vorticity in the wake is not uniform. Since the visualization of this process is more pronounced as the angle of attack is increased, I have included the plots for the case $a = 0.25$ and $\alpha = -\pi/6$. (See Figs. 20–25.)

At time $t = 5.4$ we see that the flow on the top of the airfoil separates very close to the leading edge of the airfoil. The vorticity in the flow downstream from this point causes the formation of circular eddies which become larger farther downstream. The flow in the bottom part of the airfoil separates at the trailing edge. The vorticity in the wake is not uniform and generates an oscillatory pattern (see Fig. 17).

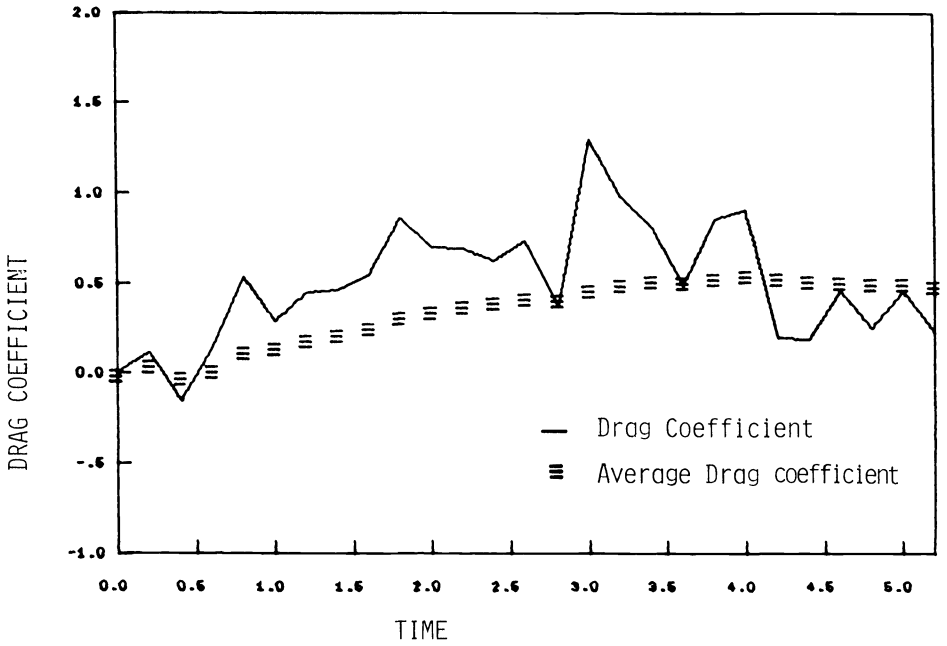


FIG. 18.

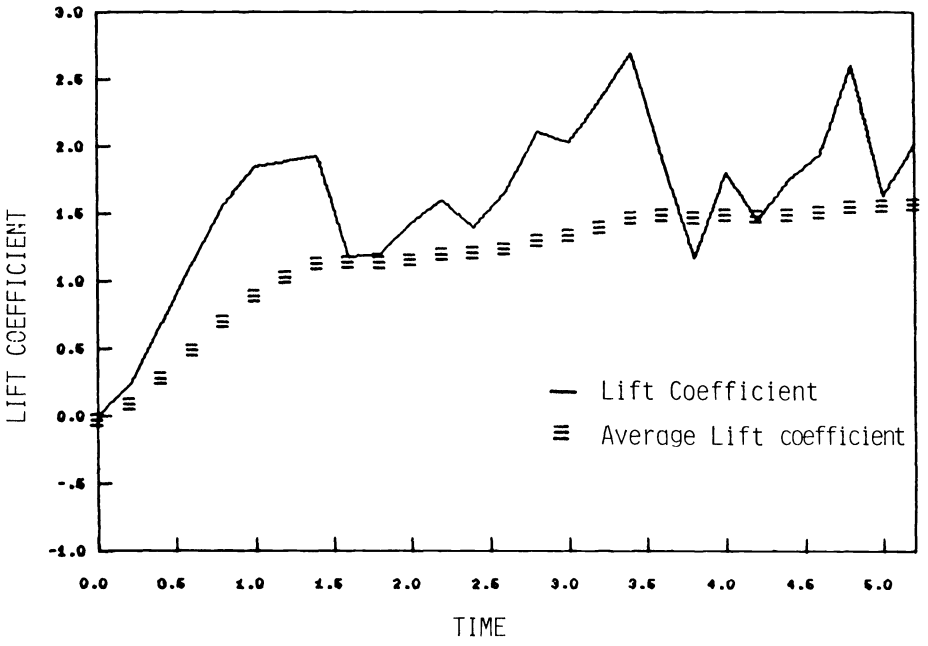


FIG. 19.

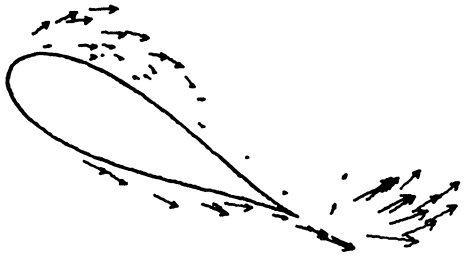


FIG. 20. *Time* $t = 1$.

FIG. 21. *Time* $t = 2$.

FIG. 22. *Time* $t = 3$.

FIG. 23. *Time* $t = 4$.

FIG. 24. *Time* $t = 5$.

FIG. 25. *Time* $t = 5.2$.

The lift and drag coefficients in this study were calculated using (19) and the results are plotted in Figs. 18 and 19. The drag coefficient starts at zero, oscillated up and down giving an average of 0.49 after time $t = 5.2$. Similarly the lift coefficient oscillates up and down. The average lift coefficient approaches the value 1.5 after time $t = 5.2$.

This run took 68 CP-seconds on the CDC-7600 at Lawrence Berkeley Laboratory. At time $t = 5$, the flow was modeled by approximately 70 sheets and 300 points.

Conclusions. In the previous sections, we have outlined both the vortex-blob and the vortex-sheet methods. We then developed a hybrid algorithm by coupling these two methods. This hybrid algorithm was used to simulate flow past a circular cylinder, and flow past a Joukowski airfoil at varying angles of attack. We have seen that random-vortex methods for modeling the fluid flow past a circular cylinder yield results that are very accurate when compared to experimental data. Similarly, good results are obtained for flow past conformal transformation airfoils. This hybrid method is grid-free and can easily be adapted to give flow past objects of arbitrary shape.

Note that in our calculations no special assumption is made about the point of separation, no lower limit on the thickness of the boundary layer is imposed, there is no evidence of blow-up at the cusp end of the airfoil, and new computational elements are introduced only to satisfy the tangential boundary conditions. For discussion of other vortex methods used for flow simulation, see A. Leonard [17], Kuwahara [15], and Sarpkaya and Schoaff [24].

There is hope that the algorithm presented in this paper can be expanded to study oscillating airfoils, three-dimensional flows and turbulence.

Acknowledgments. The author extends special thanks to Professor A. J. Chorin not only for suggesting the problem, but also for his helpful discussions, and to Dr. A. Leonard for his discussions on the center vortex.

REFERENCES

[1] G. K. BATCHELOR, *An Introduction to Fluid Dynamics*, Cambridge Univ. Press, Cambridge, 1967.

[2] J. T. BEAL AND A. MAJDA, *Rates of convergence for viscous splitting of the Navier–Stokes equations*, Math. Comp., 39 (1982), pp. 1–27.

[3] A. Y. CHEER, *Program* BOUNDL, LBL-6443 Suppl. Report, Lawrence Berkeley Laboratory, Berkeley, CA, 1978.
[4] A. J. CHORIN AND J. E. MARSDEN, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, 1979.
[5] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.
[6] ———, *Vortex sheet approximation of boundary layers*, J. Comp. Physics, 27 (1978), pp. 428–442.
[7] A. J. CHORIN, T. J. HUGHES, M. F. MCCRACKEN AND J. E. MARSDEN, *Product formulas and numerical algorithms*, Comm. Pure Appl. Math., 31 (1978), pp. 205–256.
[8] A. J. CHORIN, *Vortex models and boundary layer instability*, this Journal, 1 (1980), pp. 1–22.
[9] R. R. CLEMENTS, *An inviscid model of two-dimensional vortex shedding*, J. Fluid Mech., 57 (1973), pp. 321–336.
[10] J. M. R. GRAHAM, *The forces on sharp-edged cylinders in oscillatory flow at low Keulegan–Carpenter numbers*, J. Fluid Mech., 97, part 1 (1980), pp. 331–346.
[11] O. H. HALD, *Convergence of vortex methods for Euler equations:* II, SIAM J. Numer. Anal., 16 (1979), pp. 726–755.
[12] ———, *Convergence of random methods for reaction diffusion equations*, this Journal, 2 (1981), pp. 85–94.
[13] O. H. HALD AND V. M. DELPRETE, *Convergence of vortex methods for Euler equations*, Math. Comp., 32 (1978), pp. 791–809.
[14] J. M. HAMMERSLEY AND D. C. HAMDSCOMB, *Monte Carlo Methods*, Methuen, London, 1964.
[15] K. KUWAHARA, *Study of flow past a circular cylinder by an inviscid model*, J. Phys. Soc. Japan, 45 (1978), pp. 292–297.
[16] L. D. LANDAU AND E. M. LIFSCHITZ, *Fluid Mechanics*, Pergamon Press, New York, 1975.
[17] A. LEONARD, *Vortex methods for flow simulation*, J. Comp. Physics, 37 (1980), pp. 289–335.
[18] C. C. LIN, *On the motion of vortices in two dimensions*, Univ. Toronto Studies, Applied Math. Series, No. 5, 1943.
[19] MILNE-THOMSON, *Theoretical Hydrodynamics*, 3rd ed., Macmillan, New York, 1955.
[20] ———, *Theoretical Aerodynamics*, 4th ed., Dover, New York, 1966.
[21] P. A. P. MORAN, *Introduction to Probability Theory*, Oxford Univ. Press, London 1968.
[22] A. N. NAHAVANDI AND S. S. CHEN, *A review on fluid forces on circular cylinders in cross flow*, Argonne National Laboratory Technical Memorandum, Argonne, IL, January 1979.
[23] L. PRANDTL AND O. G. TIETJENS, *Applied Hydro and Aeromechanics*, Dover, New York, 1957.
[24] T. SARPKAYA AND R. L. SCHOAFF, *Inviscid model of two-dimensional vortex shedding by a circular cylinder*, AIAA J., 17 (1979), pp. 1193–1200.
[25] H. SCHLICHTING, *Boundary Layer Theory*, 7th ed., McGraw-Hill, New York, 1979.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0010 $01.25/0

A SPACE-EFFICIENT RECURSIVE PROCEDURE FOR ESTIMATING A QUANTILE OF AN UNKNOWN DISTRIBUTION*

LUKE TIERNEY†

Abstract. Consider the problem of computing an estimate of a percentile or quantile of an unknown population based on a random sample of n observations. By viewing this problem as a problem in stochastic approximation, we obtain an estimator that requires only a small amount of direct access storage space that does not increase with the sample size. We show that a modified version of the simple stochastic approximation estimator has the same large-sample behavior as the sample quantile, which has the smallest asymptotic variance among all reasonable estimators. The modified procedure also yields an estimate of the asymptotic variance of the estimator. Some simulation results are presented to show that the proposed estimator performs well in samples of moderate size.

Key words. quantile estimation, recursive estimation, stochastic approximation

1. Introduction. Suppose we are interested in estimating the αth quantile (or 100 αth percentile) of an unknown distribution function F using a random sample of size n. We assume that this quantile is unique and denote it by $\xi = \xi(\alpha)$. Thus ξ is the unique value of x such that $F(x) = \alpha$. The estimator that immediately comes to mind is the αth sample quantile, i.e., the $\lfloor \alpha n \rfloor$th smallest of the n observations. Under mild restrictions on F the sample quantile is consistent and asymptotically normal with mean ξ and variance $n^{-1}\alpha(1-\alpha)f(\xi)^{-2}$, where $f(\xi) = F'(\xi)$. In fact, Pfanzagl (1974) has shown that it is not possible to find a translation invariant estimator of ξ that has smaller asymptotic variance unless more is known about F, e.g., that F belongs to a simple parametric family.

Unfortunately, the sample quantile has some computational disadvantages. All algorithms for computing the αth sample quantile, even the recently developed time-efficient algorithms with computation time that is linear in n (see Knuth (1973)), require an amount of direct access storage space that is linear in the sample size. If n is large then the cost of this storage requirement may be prohibitive. Furthermore, in most computing environments existing programs have to be modified to allow for storage space management once storage requirements exceed a certain level.

These problems raise the question whether it is possible to find an estimator that requires a fixed amount of storage space but retains the asymptotic properties of the sample percentile.[1] The answer is yes.

In this paper we show that such an estimator can be obtained by viewing the problem at hand as a stochastic approximation problem. The simple stochastic approximation estimator, which is described at the beginning of the next section, is modified slightly to obtain the desired asymptotic behavior. As a by-product of this modification we also obtain an estimate of the standard deviation of the estimator. The resulting estimator is recursive and requires only a small, fixed amount of direct access storage space. Its recursive nature also makes it suitable for use in situations where data is

* Received by the editors January 15, 1981.

† Department of Statistics, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

[1] Weide (1977) studied this problem and suggested that one break the sample up into blocks of size r, compute the αth sample quantile within each block and then average across the block estimates. The storage space required by this procedure is proportional to r. However, Weide's claim that the asymptotic distribution of the sample quantile is attained for any sequence $r(n)$ increasing to infinity is incorrect; $r(n)$ must increase faster than $\sqrt{n}$.

collected or processed sequentially. The only restriction that we need to impose on F is the requirement that it have a bounded derivative that is continuous at ξ.

The next section of this paper introduces the stochastic approximation estimator and formalizes the statement on its asymptotic behavior in Theorem 1. The third section presents some simulation results comparing the finite sample behavior of the sample quantile and the stochastic approximation estimator. Concluding remarks are given in § 4 and a proof of Theorem 1 is given in the appendix.

2. The stochastic approximation estimator. Stochastic approximation was introduced by Robbins and Monro (1951) as a method for finding the solution y_0 to an equation $M(y)=c$ in which $M(\cdot)$ is an increasing function that can only be evaluated with error. The method is recursive. To update Y_n, the current estimate of the root, one obtains $Z_{n+1}=M(Y_n)+\varepsilon_{n+1}$, an approximation to $M(\cdot)$ at Y_n. Here ε_{n+1} is a random error with zero mean that is independent of the previous errors. The updated estimate of the root is then given by $Y_{n+1}=Y_n-(d/(n+1))(Z_{n+1}-c)$ for some constant d. Under mild assumptions on the behavior of M the estimates Y_n converge almost surely to y_0. Furthermore, if $M'(y_0)$ exists and $2\,dm'(y_0)>1$, then the distribution of $\sqrt{n}(Y_n-y_0)$ converges to a normal distribution with mean zero and variance $d^2\sigma^2(2\,dm'(y_0)-1)^{-1}$ (see Sacks (1958)). Here σ^2 is the common variance of the ε_i.

A simple calculation shows that this variance is minimized by setting $d=M'(y_0)^{-1}$, and the resulting variance is $\sigma^2M'(y_0)^{-2}$. Typically $M'(y_0)$ is not known. However, one can often construct estimates m_n of $M'(y_0)$ based on $X_0,\cdots,X_n$ and use these to obtain the modified estimators $\hat{Y}_{n+1}=\hat{Y}_n-(d_n/(n+1))(Z_{n+1}-c)$, where $d_n=m_n^{-1}$. Under some additional regularity conditions the distribution of $\sqrt{n}(\hat{Y}_n-y_0)$ converges to a normal distribution with mean zero and variance $\sigma^2M'(y_0)^{-2}$ (see Ventner (1967) and Fabian (1968)).

In the years since its introduction, the method of stochastic approximation has been adapated for use in a variety of estimation problems. For example, Albert and Gardner (1967) study the problem of estimating the parameters of a nonlinear regression model by stochastic approximation, and Martin and Masreliez (1975) use stochastic approximation to obtain robust estimators of a location parameter that are asymptotically equivalent to Huber's M-estimators.

To adapt this procedure to the present setting, note that we are interested in estimating the root of the equation $F(x)=\alpha$. We denote the solution by ξ and assume that it is unique. Suppose $X_1,X_2,\cdots$ is a random sample from F and define the functions $Z(\cdot,\cdot)$ and $I(x,y,z)$ by

$$Z(x,y)=\begin{cases}1 & \text{if } x\leqq y,\\ 0 & \text{if } x>y,\end{cases} \tag{1}$$

and

$$I(x,y,z)=\begin{cases}1 & \text{if } |x-y|\leqq z,\\ 0 & \text{if } |x-y|>z.\end{cases} \tag{2}$$

Now define the estimators $\hat{\xi}_n$ of ξ recursively by

$$\hat{\xi}_{n+1}=\hat{\xi}_n-\frac{d_n}{n+1}(Z(X_{n+1},\hat{\xi}_n)-\alpha), \tag{3}$$

with

$$d_n=\min(\hat{f}_n(\xi)^{-1},d_0n^a). \tag{4}$$

Here $0<a<\frac{1}{2}$, $d_0>0$, and $\hat{f}_n(\xi)$ is an estimator of $f(\xi)$ defined by

$$\hat{f}_n(\xi)=\frac{1}{n}\sum_{i=0}^{n-1} I(\hat{\xi}_i, X_{i+1}, h_{i+1})/(2h_{i+1}) \tag{5}$$

for a sequence $\{h_n\}$ that tends to zero at an appropriate rate. The initial estimate ξ_0 of ξ and d_0, an initial estimate of $f(\xi)^{-1}$, are treated as fixed; in practice they can be obtained from a small preliminary sample. Note that $\hat{f}_n(\xi)$ can be calculated recursively by setting $\hat{f}_0(\xi)=0$ and using the identity

$$\hat{f}_{n+1}(\xi)=\frac{1}{n+1}(n\hat{f}_n(\xi)+I(\hat{\xi}_n, X_{n+1}, h_{n+1})/(2h_{n+1})). \tag{6}$$

Since $E[Z(X_{n+1}, \hat{\xi}_n)|X_0, \cdots, X_n]=F(\hat{\xi}_n)$, the estimators defined by (3) fit the framework described above. Furthermore, since $E[I(\hat{\xi}_n, X_{n+1}, h_{n+1})|X_0, \cdots, X_n]=F(\hat{\xi}_n+h_{n+1})-F(\hat{\xi}_n-h_{n+1})$, which is approximately equal to $2h_{n+1}\hat{f}_n(\xi)$ if n is large, $\hat{f}_n(\xi)$ will converge to $f(\xi)$.[2] The d_n are defined by (4) to guard against possible severe fluctuations of $\hat{f}_n(\xi)$ for small values of n. The resulting estimators $\hat{\xi}_n$ converge almost surely to ξ and are asymptotically normal with mean ξ and variance $n^{-1}\alpha(1-\alpha)f(\xi)^{-2}$, which is the asymptotic variance of the sample quantile. This result is formalized in the following theorem.

THEOREM 1. *Let $\alpha\in(0,1)$, $d_0>0$, ξ_0, $a\in(0,\frac{1}{2})$, and a sequence $\{h_n\}$ of nonnegative numbers that tend to zero and satisfy $\sum_{n=1}^{\infty}(n^2h_n)^{-1}<\infty$ be given. Let F be a distribution function for which $f(x)=F'(x)$ exists for all x and is uniformly bounded. Furthermore assume that $f(\cdot)$ is continuous and positive at ξ, the unique value of x such that $F(x)=\alpha$. Let $X_1, X_2, \cdots$ be a random sample from F and define $\hat{\xi}_n$, d_n and $\hat{f}_n(\xi)$ by* (3), (4) *and* (5), *respectively. Then $\hat{\xi}_n\to\xi$ and $\hat{f}_n(\xi)\to f(\xi)$ almost surely, and the distribution of $\sqrt{n}(\hat{\xi}_n-\xi)$ converges to a normal distribution with mean zero and variance $\alpha(1-\alpha)f(\xi)^{-2}$.*

Due to the structure of this estimator, the amount of direct access storage required is independent of the sample size; the particular implementation described in the next section requires only six direct access storage locations. Furthermore, since $\hat{f}_n(\xi)$ converges to $f(\xi)$, the variance of the asymptotic distribution of the estimator can be estimated by $\alpha(1-\alpha)\hat{f}_n(\xi)^{-2}$. A proof of Theorem 1 is given in the appendix.

3. Some simulation results. To determine whether the asymptotic properties of the estimator derived in the previous section are applicable to samples of several thousand observations, in which space considerations become important, we conducted some small simulation experiments using smaller sample sizes. Specifically, we used samples of size 100, 200 and 500 to estimate the .75th and .90th quantiles of a standard normal distribution using the sample quantile and the stochastic approximation (SA) estimator. The simulation experiment was replicated 100 times.

To compute the SA estimator we used the first ten observations in each series to obtain the initial values ξ_0 and d_0, setting ξ_0 equal to the $\lfloor 10\alpha \rfloor$th smallest of the first ten observations and d_0^{-1} equal to the interquantile range of the first ten observations (specifically, to the difference between the eighth and third smallest observations). For h_n we used the sequence $\{n^{-1/2}\}$ (if we restrict our choice of h_n to sequences of the form $\{n^{-\beta}\}$, then the restrictions on h_n imply that $0<\beta<1$; $\frac{1}{2}$ is the center of this

[2] The slope estimator proposed by Ventner (1967) and Fabian (1968) for the general stochastic approximation problem can be applied as well. However, we believe that (5) produces estimators of ξ that are more efficient (in terms of second order efficiency).

interval). Uniform random numbers were generated using a version of the generator contained in the McGill Random Number Package "Super Duper," and these were transformed into standard normal random variables using the Box-Muller transformation.

The results of the simulations are summarized in Tables 1 and 2. These tables list the estimates of the standardized mean squared errors, i.e., the mean squared

TABLE 1

Standardized mean squared errors and mean squared difference between estimators for the .75th quantile of the standard normal distribution based on 100 *replications of the simulation experiment.*

Sample size	SA estimator	Sample percentile	Standard MS difference
100	2.3771	1.9480	.7034
200	2.1577	2.0005	.3675
500	1.9240	1.7599	.2948
∞	1.8580	1.8580	0

TABLE 2

Standardized mean squared errors and mean squared difference between estimators for the .90th quantile of the standard normal distribution based on 100 *replications of the simulation experiment.*

Sample size	SA estimator	Sample percentile	Standard MS difference
100	3.6887	2.9817	1.0525
200	4.0743	3.6134	.9012
500	3.0450	3.0262	.9598
∞	2.9231	2.9231	0

errors of the estimators multiplied by the sample size, along with the asymptotic values of these quantities. In addition, we have listed the standardized mean squared differences between the two estimators. Even for these small sample sizes the standard errors of the SA estimator and the sample quantile are quite close. Thus it is reasonable to expect that the statistical properties of the sample quantile will be virtually identical in samples of several thousand observations, where the considerations of efficient storage space use that motivated our discussion become relevant.

4. Concluding remarks. When using the stochastic approximation estimator in practice it is advisable to use a preliminary sample of size on the order of 100 to obtain initial values for ξ_0 and d_0. Furthermore, to obtain a scale invariant estimation procedure, the constants h_n should be multiplied by an estimate of scale based on the preliminary sample. Since the variance of $\hat{f}_n(\xi)$ is approximately proportional to $f(\xi)$, the reciprocal of an estimate of the density at ξ based on the preliminary sample might be used for this purpose.

An important area of applications for recursive estimation techniques is the analysis of output from system simulations. Computer simulations of dynamic systems produce large sets of data that are generated sequentially. In most simulations the generated output series have a stationary distribution, at least in the limit, but the

individual observations are not independent. For this reason it would be of interest to determine whether the stochastic approximation estimator still performs well when applied to dependent data; this is a subject of current research.

Appendix. Proof of Theorem 1. In this appendix we consider estimators of the form

$$\hat{\xi}_{n+1} = \hat{\xi}_n - \frac{d_n}{n+1}(Z(X_{n+1}, \hat{\xi}_n) - F(\xi)) \tag{A1}$$

for $n \geqq 0$, where $F(\xi) = \alpha$, the d_n's are nonnegative and determined by $X_1, \cdots, X_n$, and $Z(x, y)$ is given by (1). We will prove Theorem 1 using the following two propositions.

PROPOSITION 1. *Suppose that $f(\xi) = F'(\xi)$ exists and is positive. Suppose furthermore that*

$$E\left[\sum_{n=1}^{\infty}\left(\frac{d_n}{n+1}\right)^2\right] < \infty \quad \text{and} \quad \sum_{n=1}^{\infty}\frac{d_n}{n+1} = \infty \quad \text{almost surely.}$$

Then $\hat{\xi}_n$ almost surely converges to ξ.

PROPOSITION 2. *Assume, in addition to the hypotheses of Proposition 1, that d_n converges almost surely to a constant d with $2df(\xi) > 1$. Then the distribution of $\sqrt{n}(\hat{\xi}_n - \xi)$ converges to a normal distribution with zero mean and variance $d^2(2df(\xi) - 1)^{-1}\alpha(1-\alpha)$.*

Proposition 2 is a special case of a very general theorem of Fabian (1968). The proof of Theorem 1 in terms of these propositions is as follows:

Proof of Theorem 1. Since $d_n \leqq n^a$, we have

$$\sum_{n=1}^{\infty}\left(\frac{d_n}{n+1}\right)^2 \leqq \sum_{n=1}^{\infty} n^{2a-2} < \infty.$$

So

$$E\left[\sum_{n=1}^{\infty}\left(\frac{d_n}{n+1}\right)^2\right] < \infty.$$

Now write

$$\hat{f}_n(\xi) = \frac{1}{n}\sum_{j=0}^{n-1}(I(\hat{\xi}_j, X_{j+1}, h_{j+1}) - 2h_{j+1}f(\theta_j))/(2h_{j+1}) + \frac{1}{n}\sum_{j=0}^{n-1} f(\theta_j), \tag{A2}$$

where θ_j is such that $|\theta_j - \hat{\xi}_j| \leqq h_{j+1}$ and

$$P\{\hat{\xi}_j - h_{j+1} \leqq X_{j+1} \leqq \hat{\xi}_j + h_{j+1}\} = 2h_{j+1}f(\theta_j).$$

Such quantities exist by the mean value theorem. Since $f(\cdot)$ is bounded, the second term in (A.2) is bounded. The boundedness of f and the assumptions on $\{h_n\}$ imply that

$$\sum_{j=0}^{n-1}(I(\hat{\xi}_j, X_{j+1}, h_{j+1}) - 2h_{j+1}f(\theta_j))/(2jh_j)$$

is an $\mathscr{L}^2$-bounded martingale and therefore converges almost surely to some limit. Kronecker's lemma (see Chung (1974, p. 123)) then implies that the first term in (A2) tends to zero. Hence d_n is bounded away from zero and thus $\sum_{n=1}^{\infty} d_n/(n+1) = \infty$ almost surely. So the assumptions of Proposition 1 hold and thus $\hat{\xi}_n$ converges almost surely to ξ. This, in turn, implies that the θ_j's in (A.2) converge almost surely to ξ.

Since f is continuous at ξ, this means that d_n^{-1} converges almost surely to $f(\xi)$. Therefore, by Proposition 2, $\sqrt{n}(\hat{\xi}_n - \xi)$ converges in distribution to the desired limit. □

To complete our argument, we must prove Proposition 1.

Proof of Proposition 1. Define the following quantities:

$$A_n = -\frac{d_n}{n+1}(Z(\hat{\xi}_n, X_{n+1}) - F(\hat{\xi}_n))$$

and

$$B_n = \frac{d_n}{n+1}(F(\hat{\xi}_n) - F(\xi))/(\hat{\xi}_n - \xi).$$

Note that $B_n \geqq 0$. In terms of these quantities the error $\hat{\xi}_{n+1} - \xi$ can be written as

$$(\hat{\xi}_{n+1} - \xi) = (\hat{\xi}_n - \xi)(1 - B_n) + A_n. \tag{A3}$$

This identity can be iterated to obtain

$$(\hat{\xi}_{n+1} - \xi) = (\hat{\xi}_0 - \xi) \prod_{j=0}^{n} (1 - B_j) + \sum_{k=0}^{n} A_k \prod_{j=k+1}^{n} (1 - B_j). \tag{A4}$$

The sums $\sum_{k=0}^{n} A_k$ converge almost surely since the assumptions on d_n imply that they form an $\mathcal{L}^2$-bounded martingale. Furthermore, $B_n \geqq 0$ for all n and $B_n \to 0$ almost surely since $f(\xi)$ exists and $d_n/(n+1) \to 0$ almost surely. These facts, along with a modification of the argument used to prove Lemma 2 in the appendix to Albert and Gardner (1967) imply that $\hat{\xi}_n$ is almost surely bounded. This, along with the assumptions that $f(\xi) > 0$ and that $\sum_{n=1}^{\infty} (d_n/(n+1)) = \infty$ almost surely, implies that $\sum_{n=1}^{\infty} B_n = \infty$ and therefore $\prod_{n=1}^{\infty} (1 - B_n) = 0$ almost surely. Now the lemma mentioned above may be applied directly to obtain the desired result. □

Acknowledgments. The author would like to thank Lee W. Schruben for suggesting this problem, William F. Eddy for his valuable comments on a preliminary draft of this paper, and Diane Lambert for many helpful discussions.

REFERENCES

A. E. ALBERT AND L. A. GARDNER (1967), *Stochastic Approximation and Nonlinear Regression*, MIT Press, Cambridge, MA.

K. L. CHUNG (1974), *A Course in Probability Theory*, Academic Press, New York.

V. FABIAN (1968), *On asymptotic normality in stochastic approximation*, Ann. Math. Statist., 39, pp. 1327–1332.

D. E. KNUTH (1973), *The Art of Computer Programming, Vol.* III: *Sorting and Searching*, Addison-Wesley, Reading, MA.

R. D. MARTIN AND C. J. MASRELIEZ (1975), *Robust estimation via stochastic approximation*, IEEE Trans. Inform. Theory, IT-21, pp. 263–271.

J. PFANZAGL (1974), *Investigating the quantile of an unknown distribution*, in Contributions to Applied Statistics (dedicated to Arthur Linder), W. J. Ziegler, ed., Birkhauser Verlag, Basel, pp. 111–126.

H. ROBBINS AND S. MONRO (1951), *A stochastic approximation method*, Ann. Math. Statist., 22, pp. 400–427.

J. H. VENTNER (1967), *An extension of the Robbins–Monro procedure*, Ann. Math. Statist., 38, pp. 181–190.

B. WEIDE (1978), *Space-efficient on-line selection algorithms*, Proc. Computer Science and Statistics: Eleventh Annual Symposium on the Interface, Institute of Statistics, North Carolina State University, Raleigh.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0011 $01.25/0

IMPROVING THE ACCURACY OF COMPUTED SINGULAR VALUES*

J. J. DONGARRA†

Abstract. This paper describes a computational method for improving the accuracy of a given singular value and its associated left and right singular vectors. The method is analogous to iterative improvement for the solution of linear systems. That is, by means of a low-precision computation, an iterative algorithm is applied to increase the accuracy of the singular value and vectors; extended precision computations are used in the residual calculation. The method is related to Newton's method applied to the singular value problem and inverse iteration for the eigenvalue problem.

Key words. singular values improvement, iterative method, singular values

1. The basic algorithm. In a recent paper, Dongarra, Moler and Wilkinson [1] described an algorithm for improving an approximation to a simple eigenvalue and the corresponding eigenvector. In this paper we extend and modify the algorithm to cover the singular value problem. We know that the singular values of a matrix are well conditioned in the sense that small changes in the matrix result in small changes in the singular values. The singular vectors may not be well determined and may vary drastically with small changes in the matrix. In [3], Stewart describes a somewhat analogous procedure for determining error bounds and obtaining corrections to the singular values and vectors associated with invariant subspaces. Here we describe a procedure for improving a single or arbitrary singular value and singular vectors using the previously computed factorization.

We begin with a brief description of the basic algorithm.

Given an $m \times n$ rectangular matrix A, we are interested in the decomposition

$$A = U \Sigma V^T, \tag{1.1}$$

where U and V are unitary matrices and Σ is a rectangular diagonal matrix of the same dimension as A with real nonnegative diagonal entries. The equations can also be written as

$$Av_i = \sigma_i u_i \tag{1.2}$$

and

$$A^T u_i = \sigma_i v_i \quad \text{for each singular value } \sigma_i. \tag{1.3}$$

If σ, u, and v have been derived from some computation on a computer with finite precision or by some insight into the problem, they are generally not the true singular value and vectors, but approximations. We know, however, that there exist μ_1, μ_2, y, and z such that

$$A(v+y) = (\sigma+\mu_1)(u+z) \tag{1.4}$$

and

$$A^T(u+z) = (\sigma+\mu_2)(v+y), \tag{1.5}$$

where μ_1, μ_2, y, and z, when added to computed σ, u, and v, give the exact left and

* Received by the editors February 5, 1982. This research was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under contract W-31-109-Eng-38.

† Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439.

right singular vectors and the exact singular value. The corrections μ_1 and μ_2 come about by the separate nature of (1.2) and (1.3). We compute the correction to σ as $\mu = (\mu_1 + \mu_2)/2$.

The above equations can be expanded to obtain

$$Ay - \sigma z - \mu_1 u = \sigma u - Av + \mu_1 z$$

and

$$A^T z - \sigma y - \mu_2 v = \sigma v - A^T u + \mu_2 y. \tag{1.6}$$

If the orthogonality conditions

$$(v+y)^T(v+y) = 1 \tag{1.7}$$

and

$$(u+z)^T(u+z) = 1$$

are included, we then have $m+n+2$ equations in $m+n+2$ unknowns. We can now rewrite the equations in matrix notation to obtain

$$\begin{pmatrix} -\sigma I & A & -u & 0 \\ A^T & -\sigma I & 0 & -v \\ 2u^T & 0 & 0 & 0 \\ 0 & 2v^T & 0 & 0 \end{pmatrix} \begin{pmatrix} z \\ y \\ \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} \sigma u - Av + \mu_1 z \\ \sigma v - A^T u + \mu_2 y \\ 1 - u^T u - z^T z \\ 1 - v^T v - y^T y \end{pmatrix}. \tag{1.8}$$

Note that this is a mildly nonlinear matrix equation. We can determine the unknowns $(z, y, \mu_1, \mu_2)^T$ iteratively by solving

$$\begin{pmatrix} -\sigma^{(p)} I & A & -u^{(p)} & 0 \\ A^T & -\sigma^{(p)} I & 0 & -v^{(p)} \\ 2u^{(p)T} & 0 & 0 & 0 \\ 0 & 2v^{(p)T} & 0 & 0 \end{pmatrix} \begin{pmatrix} z^{(p+1)} \\ y^{(p+1)} \\ \mu_1^{(p+1)} \\ \mu_2^{(p+1)} \end{pmatrix} = \begin{pmatrix} \sigma^{(p)} u^{(p)} - Av^{(p)} + \mu_1^{(p)} z^{(p)} \\ \sigma^{(p)} v^{(p)} - A^T u^{(p)} + \mu_2^{(p)} y^{(p)} \\ 1 - u^{(p)T} u^{(p)} - z^{(p)T} z^{(p)} \\ 1 - v^{(p)T} v^{(p)} - y^{(p)T} y^{(p)} \end{pmatrix}, \tag{1.9}$$

to obtain corrections to $u^{(p)}$, $v^{(p)}$, and $\mu^{(p)}$ by the updates

$$u^{(p+1)} = u^{(p)} + z^{(p+1)},$$
$$v^{(p+1)} = v^{(p)} + y^{(p+1)},$$
$$\sigma^{(p+1)} = \sigma^{(p)} + (\mu_1^{(p+1)} + \mu_2^{(p+1)})/2.$$

If A is $m \times n$, then this is an $(m+n+2) \times (m+n+2)$ system to be solved. If this system is solved, we can compute corrections μ, y, and z to the singular value and the singular vectors, thereby obtaining a more accurate value for the singular value and singular vectors.

If we handle this as we do in the eigenvalue case [1], we will improve the accuracy of σ, u, and v. The accuracy obtained by the algorithm will be full working precision, with only the residual calculations (the right-hand side of (1.9)) done in extended precision.

2. Relationship to Newton's method. The algorithm as described above can be derived by the use of Newton's method applied to (1.2) and (1.3). We define functions f_i and f as follows:

$$\begin{aligned} f_1(u, v, \sigma_1, \sigma_2) &= Av - \sigma_1 u, & f_3(u, v, \sigma_1, \sigma_2) &= u^T u - 1, \\ f_2(u, v, \sigma_1, \sigma_2) &= A^T u - \sigma_2 v, & f_4(u, v, \sigma_1, \sigma_2) &= v^T v - 1, \end{aligned} \tag{2.1}$$

and

$$f(x) = (f_1(x), f_2(x), f_3(x), f_4(x)),$$

where

$$x = \begin{pmatrix} u \\ v \\ \sigma_1 \\ \sigma_2 \end{pmatrix}.$$

The approach is to find the zeros of $f(x)$. Newton's method applied to this problem is

$$f'(x_i)(x_{i+1} - x_i) = -f(x_i), \tag{2.2}$$

where

$$x_i = \begin{pmatrix} u^{(i)} \\ v^{(i)} \\ \sigma_1^{(i)} \\ \sigma_2^{(i)} \end{pmatrix}.$$

The derivative of $f(x)$ is

$$f'(x) = \begin{pmatrix} -\sigma_1 I & A & -u & 0 \\ A^T & -\sigma_2 I & 0 & -v \\ 2u^T & 0 & 0 & 0 \\ 0 & 2v^T & 0 & 0 \end{pmatrix}. \tag{2.3}$$

The above method expressed in matrix notation is then just a restatement of (1.8), ignoring the second order terms in the right-hand side.

Notice that since the method is equivalent to Newton's method, we could compute the left and right singular vectors, given a close approximation to the singular value.

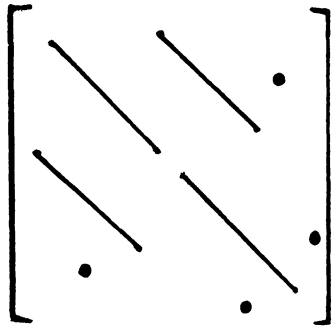
3. Effects of various factorizations. If we have computed the singular value decomposition and retained the matrices produced during the factorization, each singular value and the corresponding singular vectors can be improved in $O(mn)$ operations. We will assume that the matrices U, Σ, and V are available such that $A \approx U\Sigma V^T$. Then the coefficient matrix in (1.8) can be decomposed into the form

$$\begin{pmatrix} U & 0 & 0 & 0 \\ 0 & V & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -\sigma_s I & \Sigma & -e_s & 0 \\ \Sigma & -\sigma_s I & 0 & -e_s \\ e_s^T & 0 & 0 & 0 \\ 0 & e_s^T & 0 & 0 \end{pmatrix} \begin{pmatrix} U^T & 0 & 0 & 0 \\ 0 & V^T & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.1}$$

where e_s is the sth column of the identity matrix and σ_s is the approximation being improved.

This factored form can be used to simplify (1.8). Since U and V are orthogonal, systems of equations involving the left and the right matrices of (3.1) can be easily solved by simply multiplying by the transposes. Systems of equations involving the matrix in the center can be handled by solving 2×2 or 4×4 subsystems of equations as can be seen from the nonzero structure of the matrix:

(3.2)



If we have a bidiagonal factorization of A, say $A = UBV^T$, where B is bidiagonal, then we can improve the accuracy in $O(mn)$ operations. Let us assume we have the matrices V and B from the bidiagonalization procedure. We will concentrate only on the matrix

$$\begin{pmatrix} -\sigma I & A \\ A^T & -\sigma I \end{pmatrix}. \tag{3.3}$$

This matrix is the interesting part of the one in (1.8) and can be thought of as a rank 2 modification of that equation. The matrix can then be written as

$$\begin{pmatrix} I & 0 \\ -\frac{1}{\sigma}A^T & V^T \end{pmatrix} \begin{pmatrix} -\sigma I & 0 \\ 0 & -\sigma I + \frac{1}{\sigma}B^T B \end{pmatrix} \begin{pmatrix} I & -\frac{1}{\sigma}A \\ 0 & V \end{pmatrix}. \tag{3.4}$$

Note that solving systems based on this factored form is a simple task since $V^{-1} = V^T$. The only actual need for an equation solver comes from

$$-\sigma I + \frac{1}{\sigma}B^T B, \tag{3.5}$$

and this matrix is tridiagonal. Thus, given the bidiagonal matrix and the V matrix of the transformation, we can improve the accuracy of the singular values.

If we have instead the QR factorization of A, namely $A = QR$, where R is upper triangular and Q is orthogonal, then we can improve the accuracy of the singular value in $O(mn + n^3)$ operations, provided we have some approximation to it. We will concentrate on the matrix in (3.3). This matrix can be rewritten in factored form as

$$\begin{pmatrix} I & 0 \\ -\frac{1}{\sigma}A^T & Q^T \end{pmatrix} \begin{pmatrix} -\sigma I & 0 \\ 0 & -\sigma I + \frac{1}{\sigma}R^T R \end{pmatrix} \begin{pmatrix} I & -\frac{1}{\sigma}A \\ 0 & Q \end{pmatrix}. \tag{3.6}$$

As in (3.4) it becomes a matter of solving equations with a matrix of the form

$$-\sigma I + \frac{1}{\sigma}R^T R. \tag{3.7}$$

Unlike (3.5), this matrix is full and, unfortunately, the factor R cannot be used to simplify the process since the matrix $R^T R$ is being modified by a rank n matrix, σI. Equation (3.7) requires a further factorization to solve systems based upon it.

4. Convergence of the update process. The convergence results for this method are the same as for the eigenvalue case. We state the results here but omit the proof which can be found in [1].

In the presence of round-off error, if the initial error in the singular value is small enough in some sense and the singular value is an isolated one, the iterative process will converge.

If working precision is used in computing the approximate singular values and extended precision is used in the residual calculation, then when the method converges, it produces results that are accurate to at least full working precision.

The method is equivalent to Newton's method; therefore, the convergence is quadratic.

The method just described has a deficiency: When there are multiple singular values, the matrix in (1.8) becomes ill-conditioned. The degree of ill-conditioning is related to the separation between the singular value being improved and its closest neighbor. Existence of close or multiple singular values can be monitored by examining the condition number of the matrix in (1.8). If the matrix of (1.8) has a large condition number, then the iteration will converge with a less than quadratic rate. For identical singular values, the matrix involved is exactly singular.

This deficiency can be illustrated by an example. For a 2×2 system the matrix has the form

$$\begin{pmatrix} -\sigma & \sigma_i \\ \sigma_i & -\sigma \end{pmatrix},$$

where σ is an approximation to σ_s. If any σ_i is close to σ_s, then this system will be ill-conditioned, and the conditioning depends upon $1/(\sigma-\sigma_i)$. In this situation one cannot improve just one singular value but must work with a cluster of them, as well as the invariant subspace of singular vectors.

5. Results. The following numerical tests were run on a VAX 11/780. The initial reduction was performed in single precision; double precision was used only to compute the residuals and to add the correction to the previous result. In single precision, the working accuracy is 2^{-28}; in double precision, the accuracy is 2^{-56}.

The matrices used here come from the original paper by Golub and Reinsch [2]. The first matrix has the form

$$A=\begin{pmatrix} 22 & 10 & 2 & 3 & 7 \\ 14 & 7 & 10 & 0 & 8 \\ -1 & 13 & -1 & -11 & 3 \\ -3 & -2 & 13 & -2 & 4 \\ 9 & 8 & 1 & -2 & 4 \\ 9 & 1 & -7 & 5 & -1 \\ 2 & -6 & 6 & 5 & 1 \\ 4 & 5 & 0 & -2 & 2 \end{pmatrix},$$

with singular values

$$\sigma_1=\sqrt{1248},\quad \sigma_2=20,\quad \sigma_3=\sqrt{384},\quad \sigma_4=\sigma_5=0.$$

The results from the improvement algorithm on this problem are given in Table 1. All results were achieved using single precision computations except to accumulate the residuals. The method used was based on the factored form of (3.1).

TABLE 1

Iteration	σ	$u^T u$	$v^T v$
0	35.3270149	0.999999718	0.999999683
1	35.327043465315658	1.000000000000101	1.000000000000304
2	35.327043465311387	1.000000000000000	1.000000000000000
true	35.327043465311387419		
0	19.9999790	0.999999520	0.999999326
1	20.000000000006048	1.000000000003621	1.000000000003431
2	20.000000000000000	1.000000000000000	1.000000000000000
true	20.0		
0	19.5958881	0.999999043	0.999999379
1	19.595917942277176	1.000000000003258	1.000000000003183
2	19.595917942265425	1.000000000000000	1.000000000000000
true	19.595917942265424785		
0	0.00000718535284	0.999998454	0.999999228
1	−0.000000000004162	1.000000000000745	1.000000000000306
2	0.000000000000000	1.000000000533098	1.000000000281307
true	0.0		
0	0.00000120505399	0.999998900	0.999999509
1	−0.000000000000479	1.000000000000304	1.000000000000061
2	0.000000000000000	1.000000018476308	1.000000001164373
true	0.0		

The results in Table 1 show the iteration converging very rapidly. The singular values are initially correct to working precision, and two iterations have gained full extended precision.

For the next example we use a standard symmetric eigenvalue problem. The matrix, W^+_{2k+1} [4], is symmetric tridiagonal, and has some pathologically close eigenvalues and singular values. It is defined by the relations

$$\alpha_i = k+1-i, \qquad i = 1, \cdots, k+1,$$

$$\alpha_i = i-k-1, \qquad i = k+2, \cdots, 2k+1,$$

$$\beta_i = 1, \qquad i = 2, \cdots, 2k+1,$$

where $k = 5$, α_i is the ith diagonal element, and β_i is the ith subdiagonal element. See Table 2.

TABLE 2

Iteration	σ	$u^T u$	$v^T v$
0	5.7462210	0.999998079	0.999997771
1	5.746231847961203	1.001536038280316	1.001536023270078
2	5.746231833605774	1.000033093486984	1.000033093488725
3	5.746231833805267	1.000000000729813	1.000000000729813
4	5.746231833809865	1.000000000000009	1.000000000000002
5	5.746231833809865	1.000000000000000	1.000000000000000
0	5.7461471	0.999998012	0.999997719
1	5.746157555822260	1.000863731344646	1.000863745222875
2	5.746157545424549	1.000016916083231	1.000016916084818
3	5.746157545577390	1.000000000495525	1.000000000495525
4	5.746157545580572	1.000000000000011	1.000000000000011
5	5.746157545580572	1.000000000000000	1.000000000000000

The singular values displayed in Table 2 are the largest ones of W_{11}^{+} and happen to be the closest. The matrix has a condition number of 10^5, as a result, each iteration makes an improvement of approximately three digits. Note the contrast to the previous case where the matrix was well conditioned and each iteration gained a full seven digits.

6. Multiple singular values. We are interested in improving more than one singular value at a time. We are motivated to do so since the approach for improving one singular value breaks down when there are multiple singular values with close numerical values. For simplicity we will restrict the discussion initially to two singular values σ_1 and σ_2 and the corresponding vectors u_1, v_1 and u_2, v_2.

We know that the two-space in which u_1 and u_2 and v_1 and v_2 lie is numerical well determined. Hence we have

$$\begin{aligned} A(v_1+y_1) &= (\sigma_1+\mu_{11})(u_1+z_1)+\mu_{21}(u_2+z_2), \\ A(v_2+y_2) &= \mu_{12}(u_1+z_1)+(\sigma_2+\mu_{22})(u_2+z_2) \end{aligned} \tag{6.1}$$

and

$$\begin{aligned} A^T(u_1+z_1) &= (\sigma_1+\mu_{11})(v_1+y_1)+\mu_{21}(v_2+y_2), \\ A^T(u_2+z_2) &= \mu_{12}(v_1+y_1)+(\sigma_2+\mu_{22})(v_2+y_2), \end{aligned}$$

where the corrections y_i, z_i, and μ_{ij} are expected to be small.

From above we form

$$\begin{aligned} A(v_1+y_1|v_2+y_2) &= (u_1+z_1|u_2+z_2)\begin{pmatrix}\sigma_1+\mu_{11} & \mu_{12}\\ \mu_{21} & \sigma_2+\mu_{22}\end{pmatrix}, \\ A^T(u_1+z_1|u_2+z_2) &= (v_1+y_1|v_2+y_2)\begin{pmatrix}\sigma_1+\mu_{11} & \mu_{12}\\ \mu_{21} & \sigma_2+\mu_{22}\end{pmatrix}. \end{aligned} \tag{6.2}$$

We will require in addition that the orthogonality conditions

$$(v_i+y_i)^T(v_j+y_j)=\begin{cases}0 & \text{if } i=j,\\ 1 & \text{if } i\neq j\end{cases} \tag{6.3}$$

and

$$(u_i+z_i)^T(u_j+z_j)=\begin{cases}0 & \text{if } i=j,\\ 1 & \text{if } i\neq j\end{cases}$$

be satisfied. For simplicity we will assume that the 2×2 matrix of (6.2) which contains the correction to the singular values is symmetric, therefore, $\mu_{12}=\mu_{21}$. In order to produce the improved singular values this 2×2 matrix must be diagonalized.

Equation (6.2) together with (6.3) give rise to $2n+2m+6$ equations in $2n+2m+6$ unknowns. This matrix equation has the form

$$\begin{pmatrix}
-\sigma_1 I & A & & & -u_2 & & -u_1 & & \\
A^T & -\sigma_1 & & & & -v_2 & & -v_1 & \\
 & & -\sigma_2 I & A & -u_1 & & & & -u_2 \\
 & & A^T & -\sigma_2 I & & -v_1 & & & -v_2 \\
u_2^T & & u_1^T & & & & & & \\
 & v_2^T & & v_1^T & & & & & \\
2u_1^T & & & & & & & & \\
 & 2v_1^T & & & & & & & \\
 & & 2u_2^T & & & & & & \\
 & & & 2v_2^T & & & & &
\end{pmatrix}
\begin{pmatrix} z_1\\ y_1\\ z_2\\ y_2\\ \mu_{12}^1\\ \mu_{12}^2\\ \mu_{11}^1\\ \mu_{11}^2\\ \mu_{22}^1\\ \mu_{22}^2 \end{pmatrix}
=
\begin{pmatrix}
\sigma_1u_1-Av_1+\mu_{11}^1z_1+\mu_{12}^1z_2\\
\sigma_2u_2-Av_2+\mu_{12}^1z_1+\mu_{22}^1z_2\\
\sigma_1v_1-A^Tu_1+\mu_{11}^2y_1+\mu_{12}^2y_2\\
\sigma_2v_2-A^Tu_2+\mu_{12}^2y_1+\mu_{22}^2y_2\\
-\mu_1^Tu_2-z_1^Tz_2\\
-v_1^Tv_2-y_1^Ty_2\\
1-u_2^Tu_2-z_2^Tz_2\\
1-v_1^Tv_1-y_1^Ty_1\\
1-u_1^Tu_1-z_1^Tz_1\\
1-v_2^Tv_2-y_2^Ty_2
\end{pmatrix}.$$

As in the case of a single singular value, if one has access to the matrix factorization then the matrix problem can easily be solved.

In general, if we extend the procedure to handle k close singular values we have

$$A[v_1+y_1, \cdots, v_k+y_k]=[u_1+z_1, \cdots, u_k+z_k][\text{diag}\,(\sigma_i)+M]$$

and

$$A^T[u_1+z_1, \cdots, u_k+z_k]=[v_1+y_1, \cdots, v_k+y_k][\text{diag}\,(\sigma_i)+M],$$

where $m_{ij}=\mu_{ij}$ and $M=M^T$ and it is expected that y_i, z_i, and μ_{ij} will be small. These equations together with (6.3) lead to a system of equations of order $k(m+n)+k(k+1)$ and an eigenvalue problem of order k.

REFERENCES

[1] J. J. DONGARRA, C. B. MOLER AND J. H. WILKINSON, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM J. Numer. Anal., 20 (1983), pp. 23–45.

[2] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

[3] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[4] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, London, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0012 $01.25/0

APPROXIMATING CONDITIONAL MOMENTS OF THE MULTIVARIATE NORMAL DISTRIBUTION*

JOSEPH G. DEKEN†

Abstract. An approach to approximating the conditional moments of the multivariate normal distribution is presented. Symbolic manipulation (via MACSYMA) is used to derive a closed system of polynomial approximations to the conditional normal moments (higher-order Shepard's corrections) whch can be iterated rapidly to obtain higher dimensional results from elementary forms. The motivation of this technique is to make the *E-M* algorithm computationally feasible for explicitly treating interval (or rounding-error) measurements in regression problems. An example with an ill-conditioned regression problem shows the method's effectiveness.

Key words. *E-M* algorithm, Longley data, MACSYMA, multivariate normal distribution, regression, rounding error, symbolic manipulation

1. Introduction. If $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n$ are random $p \times 1$ vectors independently distributed according to a common normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the maximum likelihood estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are well known and easily computed:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i, \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} (X_i - \hat{\mu})(X_i - \hat{\mu})'.$$

In particular, if we wish to estimate the regression of X_{1p} on $X_{11}, \cdots, X_{1p-1}$, the vector $\boldsymbol{\beta}$ of regression parameters defined by

$$E(X_{1p} | X_{11}, X_{12}, \cdots, X_{1p-1}) - \mu_p =: (X_{11} - \mu_1, X_{12} - \mu_2, \cdots, X_{1p-1} - \mu_{p-1})\boldsymbol{\beta}$$

has a maximum likelihood estimate $\hat{\beta}$ which is a function of $\hat{\Sigma}$. If we partition $\hat{\boldsymbol{\Sigma}}$ as

$$\hat{\boldsymbol{\Sigma}} = \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}' & s^2 \end{bmatrix}$$

where $\mathbf{A}$ is $p-1 \times p-1$, $\mathbf{C}$ is $p-1 \times 1$ and s^2 is a scalar, the estimate is $\hat{\boldsymbol{\beta}} = \mathbf{A}^{-1}\mathbf{C}$.

If the components of the vectors $\mathbf{X}_i$ are *censored*, however, no such simple maximum likelihood estimates exist. For example, some values X_{ij} may not be observed exactly, but only specified within a certain range. As one example, measuring instruments for environmental or other contaminants may be able to record values below a certain level only as "trace." Such observations are then only known to lie in the range of values between the lowest level detectable by the instrument and an upper limit at which exact measurement becomes possible.

On a much wider scale, censoring is inevitable with numerical procedures, since any observation cannot be recovered exactly from its finite representation. (For example, an individual whose height is recorded as "189 cm" may be presumed to have height more precisely measured somewhere in the interval [188.5, 189.5) cm. Such censoring due to rounding error is often inconsequential, but in "ill-conditioned" regression problems, when the matrix $\hat{\Sigma}$ based on the rounded values is nearly singular, the estimate $\hat{\beta}$ may vary substantially when various values, all consistent with the rounded observations given, are taken as the "true" observations. A classic example of such a problem is the data presented by Longley (1967), in which the rounding error present leads to significant instability of the estimate $\hat{\beta}$ (Beaton, Rubin, and

* Received by the editors November 5, 1979.
† Department of General Business, University of Texas, Austin, Texas 78712.

Barone (1976), Dent and Cavender (1977)). If the recorded values in cases such as this may be considered as obtained by rounding the "true" continuous observations, then the correct maximum likelihood estimate $\hat{\beta}$, incorporating this rounding explicitly, exists and may be substantially different from that obtained by substituting the midpoints of the rounding intervals (the rounded values) for the true values.

The kinds of censoring described above may be viewed in the general framework of "incomplete data". That is, our rounded observations X_{ij} are used only to define intervals within which the actual values Y_{ij} fall. The *E-M* method (Dempster, Laird and Rubin (1977)) of finding the maximum likelihood estimates based on $\mathbf{X}_1, \cdots, \mathbf{X}_n$ is attractive here because if the actual values $\mathbf{Y}_1, \cdots, \mathbf{Y}_n$ were known, these regression estimates would be readily computed. The *E-M* method proceeds iteratively, each stage involving an expectation (*E*) step followed by a maximization (*M*) step. From starting values $\hat{\boldsymbol{\mu}}^{(0)}$, $\hat{\boldsymbol{\Sigma}}^{(0)}$, one finds the expected values $\hat{\mathbf{Y}}_i^{(j+1)} := E^{(j)}(\mathbf{Y}_i|\mathbf{X}_i)$, $\hat{\mathbf{Z}}_i^{(j+1)} := E^{(j)}(\mathbf{Y}_i\mathbf{Y}_i'|\mathbf{X}_i)$ (the *E*-step), and computes updated estimates $\hat{\boldsymbol{\mu}}^{(j+1)}$, $\hat{\boldsymbol{\Sigma}}^{(j+1)}$ as the maximum likelihood estimates from $\mathbf{Y}^{(j+1)}$ and $\mathbf{Z}^{(j+1)}$:

$$\hat{\mu}^{(j+1)} := \frac{1}{n}\sum_{i=1}^{n} \hat{\mathbf{Y}}_i^{(j+1)}, \qquad \hat{\boldsymbol{\Sigma}}^{(j+1)} := \frac{1}{n}\sum_{i=1}^{n} \hat{\mathbf{Z}}_i^{(j+1)}$$

(the *M*-step), repeating the procedure until the estimates converge.

In the *E*-step above the expectation $E^{(j)}$ required is the expected value (either of $\mathbf{Y}$ or of $\mathbf{YY}'$) when $\mathbf{Y}$ has a multivariate normal distribution with mean $\hat{\boldsymbol{\mu}}^{(j)}$ and variance $\hat{\boldsymbol{\Sigma}}^{(j)}$ and is *restricted to the interval determined by* $\mathbf{X}_i$. The necessity of calculating this restricted (i.e. conditional) expectation, particularly in multidimensional regression problems, motivated the work described here.

2. Approximate conditional moments. The conditional moments such as $E(X_j^k)$ of a multivariate normal random variable $X = (X_1, X_2, \cdots, X_p)$, when X is restricted to a subset $A \subset \mathbb{R}^p$, are not readily obtained numerically, since the required integration in p-dimensions is time-consuming except for very small p. We present here an efficient approxiation scheme for these moments, which makes the computation practical for moderately large p.

For convenience of description, we restrict attention to sets A of the form $I_1 \times I_2 \times \cdots \times I_p$, where all the I_j are intervals, but the approach is more general, as indicated below. We start by observing that the approximations to the ratio

$$E_k = \frac{\displaystyle\int_{s-t}^{s+t} x^k \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx}{\displaystyle\int_{s-t}^{s+t} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx}$$

obtained by the first terms in a Taylor series around $t = 0$ may be written as polynomials in μ:

$$E_k \approx \sum_{\alpha=0}^{m} c_{k\alpha}(s, t, \sigma^2)\mu^{\alpha}.$$

That is, the sum of the first terms of a Taylor series for this ratio about $t = 0$ is of the form

$$\sum_{\alpha=0}^{q} c_{k\alpha}^{*}(\sigma^2, \mu, s)t^{2\alpha}, \tag{1}$$

but also may be represented by identifying the coefficients of μ^{α} in (1) as

$$\sum_{\alpha=0}^{2q} c_{k\alpha}(\sigma^2, s, t)\mu^{\alpha}.$$

For example, (lettting $v=\sigma^2$), and expanding to order t^4 gives

$$\begin{aligned} E_1 &= s+\frac{(\mu-s)t^2}{3v}+\frac{2v(s-\mu)+(s-\mu)^3}{45v^3}t^4+\cdots \\ &= s\left(1+\frac{-1st^2v^2+2t^4v+s^2t^4}{45v^3}\right) \\ &\quad +\left(\frac{15t^2v^2-2t^4v-3s^2t^4}{45v^3}\right)\mu+\left(\frac{st^4}{15v^3}\right)^2-\left(\frac{t^4}{45v^3}\right)\mu^3+\cdots \end{aligned}$$

(e.g. $c_{12}(v, s, t)=st^4/15v^3$, $c_{13}(v, s, t)=-t^4/45v^3, \cdots$).

Since E_k, the conditional expectation of X^k, is approximated by a polynomial in μ, for any k, it follows that the conditional expectation of any polynomial in X, $E(p_0+p_1X+p_2X^2+\cdots+p_nX^n)=\sum_{\alpha=0}^{n} p_\alpha E_\alpha$ is also approximated by a polynomial $\sum_{j=0}^{m}(\sum_{\alpha=0}^{n} p_\alpha c_{\alpha j})\mu^j$ in μ, the mean of X. As will be shown below, this polynomial scheme gives a "closed system" when we consider multivariate normal $\mathbf{X}\in\mathbb{R}^p$ and the conditioning $\{X_j\in I_j, j=1, 2, \cdots, p\}$, $I_j=(s_j-t_j, s_j+t_j)$. That is, we guarantee that if $g(X)$ is a polynomial, the approximation

$$\begin{aligned} &E(g(X_p)|x_1\in I_1, X_2\in I_2, \cdots, X_p\in I_p) \\ &\quad\approx E(h_1(X_1, X_2, \cdots, X_{p-1})|X_1\in I_1, \cdots, X_{p-1}\in I_{p-1}) \\ &\quad\approx E(h_2(X_1, X_2, \cdots, X_{p-2})|X_1\in I_1, \cdots, X_{p-2}\in I_{p-2})\cdots \\ &\quad\approx E(h_{p-1}(X_1)|X_1\in I_1) \end{aligned}$$

involves functions $h_1, h_2, \cdots, h_{p-1}$ at each stage which are always polynomials, and hence can be approximated by linear combinations of approximations of E_k.

As an illustration, consider (X_1, X_2) bivariate normal, with means μ_1, μ_2, variances σ_1^2, σ_2^2, and regression such that $(X_2|X_1)$ is normally distributed with mean $\mu_2+\beta_{21}(X_1-\mu_1)$ and variance σ_{21}^2. To approximate $E(X_2|X_1\in I_1, X_2\in I_2)$, we have:

$$E(X_2|X_1\in I_1, X_2\in I_2)=E(E(X_2|X_1, X_2\in I_2)X_1\in I_1).$$

Using the fact that $X_2|X_1$ has a normal distribution, the innermost expectation above is replaced by

$$\begin{aligned} E(X_2|X_1, X_2\in I_2) &\approx \sum_{j=0}^{m} c_{1j}(\sigma_{21}^2, s_2, t_2)\mu_{X_2|X_1}^{j} \\ &= \sum_{j=0}^{m} c_{1j}(\sigma_{21}^2, s_2, t_2)(\mu_2+\beta_{21}(X_1-\mu_1))^j. \end{aligned}$$

This last sum is a polynomial $Q(X_1)=:\sum_{j=0}^{m} q_jX_1^j$, e.g. if $m=2$,

$$\begin{aligned} q_0 &= c_{10}(\sigma_{21}^2, s_2, t_2)+c_{11}(\sigma_{21}^2, s_2, t_2)(\mu_2-\beta_{21}\mu_1)+c_{12}(\mu_2-\beta_{21}\mu_1)^2, \\ q_1 &= c_{11}(\sigma_{21}^2, s_2, t_2)\beta_{21}+2c_{12}(\alpha_{21}^2, s_2, t_2)\beta_{21}(\mu_2-\mu_1\beta_{21}), \\ q_2 &= c_{12}(\sigma_{21}^2, s_2, t_2)\beta_{21}^2, \end{aligned}$$

and a final approximation gives

$$E(Q(X_1)|X_1 \in I_1) \approx \sum_{\alpha=0}^{m} \left(\sum_{j=0}^{m} q_j c_{j\alpha}(\sigma_1^2, s_1, t_1) \right) \mu_1^{\alpha}.$$

3. Multiple integration. The derivation of conditional moments for multivariate polynomials

$$Q(\mathbf{X}) = q_0 + \sum_{i=1}^{p} q_i X_i + \sum_{1 \leq i \leq j \leq p} q_{ij} X_i X_j + \sum_{1 \leq i \leq j \leq k \leq p} q_{ijk} X_i X_j X_k + \cdots$$

when $X \in \mathbb{R}^p$ is multivariate normal can be outlined simply.

1) If we fix $(X_1, \cdots, X_{p-1})$, X_p is multivariate normal and $Q(\mathbf{X})$ is a polynomial in X_p. Hence the conditional expectation of $Q(\mathbf{X})$, with $(X_1, \cdots, X_{p-1})$ fixed and $X_p \in I_p$, is approximated (using Taylor's series as outlined above) by a polynomial in the mean of $(X_p|X_1, \cdots, X_{p-1}) =: \mu^*$.

2) The coefficients of the polynomial obtained in step 1 above are themselves polynomials in $(X_1, \cdots, X_{p-1})$. The conditional mean μ^* (because of multivariate normality) is a linear function of $(X_1, \cdots, X_{p-1})$. Thus X_p has been eliminated and for $(X_1, \cdots, X_{p-1})$ fixed, we have a new polynomial $Q^*(X_1, \cdots, X_{p-1})$.

3) Letting $\mathbf{Y} \in \mathbb{R}^{p-1} := (X_1, \cdots, X_{p-1})$, $\mathbf{Y}$ is multivariate normal, and the problem is reduced to finding $E\{Q^*(\mathbf{Y})|\, Y_1 \in I_1, \cdots, Y_{p-1} \in I_{p-1}\}$. Successive applications of steps 1 and 2 eventually lead to the one-dimensional case.

Specifically, we approximate (A) the expected value of X_p^k first by a sum of terms of the form $\mu^j_{p|X_1,\cdots,X_{p-1}}$. This conditional mean is expanded (C) as $\mu^j_{p|X_1,\cdots,X_{p-1}} = (b_{p1}X_1 + b_{p2}X_2 + \cdots + b_{pp-1}X_{p-1})^j$, which is then reexpressed via the binomial theorem (B) as a sum of terms of the form $(b_{p1}X_1 + b_{p2}X_2 + \cdots + b_{pp-2}X_{p-2})^{\alpha}(b_{pp-1}X_{p-1})^{\beta}$, and the process A–C–B is then repeated, with $X_{p-1}^{\beta}, X_{p-2}^{\gamma}, \cdots, X_1^{\delta}$.

In six dimensions, for example, the process begins:

$$\begin{aligned}
X_6^j &\xrightarrow{A} \mu^k_{6|2345} \xrightarrow{C} (b_{61}X_1 + b_{62}X_2 + \cdots + b_{65}X_5)^k \\
&\xrightarrow{B} (b_{61}X_1 + \cdots + b_{64}X_4)^j (b_{65}X_5)^k \xrightarrow{A} (b_{61}X_1 + \cdots + b_{64}X_4)^j \mu^k_{5|234} \\
&\xrightarrow{C} (b_{61}X_1 + b_{62}X_2 + \cdots + b_{64}X_4)^j (b_{51}X_1 + X_2 + \cdots + b_{54}X_4)^k \\
&\xrightarrow{B} (b_{61}X_1 + b_{62}X_2 + b_{63}X_3)^j (b_{51}X_1 + b_{52}X_2 + b_{53}X_3)^k b_{64}^j b_{54}^k X_4^{j+k} \\
&\xrightarrow{A} (b_{61}X_1 + b_{62}X_2 + b_{63}X_3)^j (b_{51}X_1 + b_{52}X_2 + b_{53}X_3)^k \mu^{\alpha}_{4|321} \\
&\xrightarrow{C} (b_{61}X_1 + b_{62}X_2 + b_{63}X_3)^j (b_{51}X_1 + b_{52}X_2 + b_{53}X_3)^k (b_{41}X_1 + b_{42}X_2 + b_{43}X_3)^l \\
&\xrightarrow{B} \cdots
\end{aligned}$$

4. A numerical implementation. In three dimensions, the conditional expectation problem is defined in terms of the numerical quantities μ_1, μ_2, μ_3, b_{21}, b_{31}, b_{32}, v_1, v_2, v_3, s_1, s_2, s_3, t_1, t_2, t_3. That is, it is required to find the conditional expectation, in the region $(s_1 \pm t_1, s_2 \pm t_2, s_3 \pm t_3)$ of a polynomial in the random vector $\mathbf{X}$, when $\mathbf{X}$ has a multivariate normal distribution with mean (μ_1, μ_2, μ_3), regression coefficients b_{21}, b_{31}, b_{32}, and (conditional) variances $\mathrm{Var}\,(X_3|X_1, X_2) = v_3$, $\mathrm{Var}\,(X_2|X_1) = v_2$,

$\mathrm{Var}(X_1) = v_1$. Numerical functions suitable for use in a FORTRAN program ($\gamma_{\alpha\beta}$, s_α, $\cdots$ below) are derived which give the expectation explicitly as a function of the input quantities. The polynomial to be approximated is presumed to be of degree at most two, and perhaps bivariate in X_2 and X_3. (This case is general enough for all the means, squares, and products $EX_1, EX_2, \cdots, EX_1^2, EX_2^2, \cdots, EX_1X_2, \cdots, EX_2X_3$.) In fact, we describe here the approximate values of $E(X_3)$, $E(X_3^2)$, $E(X_2X_3)$, since other expectations reduce, by interchange of variables, to these three. The approximation used will include powers of t up to t^4 inclusive, i.e.

$$EX^j \approx \sum_{\alpha=0}^{3} c_{j\alpha}(s, t, v)\mu^\alpha.$$

Appendix A contains an extensive list of the functions c_{jk}, which have been computed using the symbolic computation system MACSYMA at MIT. Card versions of these functions containing FORTRAN assignment statements of the form "$C(J, K) = \cdots$" are available from the author.

To approximate $E(X_3)$, the integration on X_3 is first carried out, yielding

$$\sum_{\alpha=0}^{3} c_{1\alpha}(s_3, t_3, v_3)(\mu_3 - \mu_1 b_{31} - \mu_2 b_{32} + X_1 b_{31} + X_2 b_{32})^\alpha$$

$$= \sum_{\alpha=0}^{3} c_{1\alpha}(s_3, t_3, v_3) \sum_{\beta=0}^{\alpha} \sum_{\gamma=0}^{\alpha-\beta} \binom{\alpha}{\beta\gamma} (b_{31}X_1)^\beta (b_{32}X_2)^\gamma (\mu_3 - \mu_1 b_{31} - \mu_2 b_{32})^{\alpha-(\beta+\gamma)}$$

$$\cdot \sum_{\alpha=0}^{3} \sum_{\beta=0}^{3-\alpha} \left(\sum_{\gamma=\alpha+\beta}^{3} \binom{\gamma}{\alpha,\beta} c_{1\gamma}(s_3, t_3, v_3)(\mu_3 - \mu_1 b_{31} - \mu_2 b_{32})^{\gamma-(\alpha+\beta)} \right) b_{31}^\alpha b_{32}^\beta X_1^\alpha X_2^\beta.$$

The above expression is of the form

$$\sum_{\alpha=0}^{3} \sum_{\beta=0}^{3-\alpha} r_{\alpha\beta} X_1^\alpha X_2^\beta,$$

where

$$r_{\alpha\beta} := b_{31}^\alpha b_{32}^\beta \sum_{\gamma=\alpha+\beta}^{3} \binom{\gamma}{\alpha,\beta} c_{1\gamma}(s_3, t_3, v_3)(\mu_3 - \mu_1 b_{31} - \mu_2 b_{32})^{\gamma-(\alpha+\beta)}.$$

Integration on X_2 then yields

$$\sum_{\alpha=0}^{3} \sum_{\beta=0}^{3-\alpha} r_{\alpha\beta} X_1^\alpha \left(\sum_{\gamma=0}^{3} c_{\beta\gamma}(s_2, t_2, v_2)(\mu_2 - \mu_1 b_{21} + b_{21} X_1)^\gamma \right) = \sum_{\alpha=0}^{6} s_\alpha X_1^\alpha,$$

where

$$s_\alpha := \sum_{\delta=(\alpha-2)\vee 0}^{\alpha\wedge 3} \sum_{\beta=0}^{3-\delta} r_{\delta\beta} \sum_{\eta=\alpha-\delta}^{3} \binom{\gamma}{\alpha-\delta} b_{21}^{\alpha-\delta} c_{\beta\gamma}(s_2, t_2, v_2)(\mu_2 - \mu_1 b_{21})^{\gamma-(\alpha-\delta)},$$

and the final form is

$$\sum_{\alpha=0}^{6} s_\alpha \sum_{\beta=0}^{3} c_{\alpha\beta}(s_1, t_1, v_1)\mu_1^\beta.$$

The approximation of X_3^2 is carried out in the same fashion:

$$EX_3^2 = \sum_{\alpha=0}^{6} s'_\alpha \sum_{\beta=0}^{3} c_{\alpha\beta}(s_1, t_1, v_1)\mu_1^\beta,$$

$$s'_\alpha = \sum_{\delta=(\alpha-3)\vee 0}^{\alpha\wedge 3} \sum_{\beta=0}^{3-\delta} r'_{\delta\beta} \sum_{\gamma=\alpha-\delta}^{3} \binom{\gamma}{\alpha-\delta} b_{21}^{\alpha-\delta} c_{\beta\gamma}(s_2, t_2, v_2)(\mu_2 - \mu_1 b_{21})^{\gamma-(\alpha-\delta)},$$

$$r'_{\alpha\beta} = b_{31}^\alpha b_{32}^\beta \sum_{\gamma=\alpha+\beta}^{3} \binom{\gamma}{\alpha\beta} c_{2\gamma}(s_3, t_3, v_3)(\mu_3 - \mu_1 b_{31} - \mu_2 b_{32})^{\gamma-(\alpha+\beta)}.$$

(Note that the only change is the substitution of $c_{2\gamma}(s_3, t_3, v_3)$ for $c_{1\gamma}(s_3, t_3, v_3)$.)

To approximate X_2X_3, the first approximate integration is exactly as in approximating X_3, but the resulting sum is

$$X_2\Big(\sum_{\alpha=0}^{3} \sum_{\beta=0}^{3-\alpha} r''_{\alpha\beta} X_1^\alpha X_2^\beta\Big),$$

i.e. we have $r''_{\alpha\beta} = r_{\alpha(\beta-1)}$, $\beta = 1, \cdots, 4-\alpha$. A second integration gives

$$\sum_{\alpha=0}^{3} \sum_{\beta=1}^{4-\alpha} r''_{\alpha\beta} X_1^\alpha \Big(\sum_{\gamma=0}^{3} c_{\beta\gamma}(s_2, t_2, v_2)(\mu_2 - b_{21}\mu_1 + b_{21}X_1)^\gamma\Big),$$

so the final result is

$$\sum_{\alpha=0}^{6} s''_\alpha \sum_{\beta=0}^{3} c_{\alpha\beta}(s_1, t_1, v_1)\mu_1^\beta,$$

where

$$s''_\alpha = \sum_{\delta=(\alpha-3)\vee 0}^{\alpha\wedge 3} \sum_{\beta=1}^{4-\delta} r_{\delta\beta} \sum_{\eta=\alpha-\delta}^{3} \binom{\gamma}{\alpha-\delta} b_{21}^{\alpha-\delta} c_{\beta\gamma}(s_2, t_2, v_1)(\mu_2 - \mu_1 b_{21})^{\gamma-(\alpha-\delta)}.$$

5. Illustration: the Longley data. The effect of rounding can be shown using the Longley data (Longley (1967), Beaton, Rubin, Barone (1976)), which involves economics data from the years 1947–1962:

X_1: gross national product implicit price deflator,
X_2: gross national product,
X_3: unemployment (in thousands),
X_4: size of armed forces,
X_5: noninstitutional population,
X_6: year.

These regressands are used to predict

X_7: total derived employment.

The regressands are highly collinear and so the estimates $\hat{\mu}$ and $\hat{\Sigma}$ must be accurately computed.

For these data, the estimated means and covariances using the rounded values (our starting estimates $\hat{\mu}^{(0)}$, $\hat{\Sigma}^{(0)}$) are

$$\hat{\mu}^{(0)} = (101.68125, 387{,}698.4375, 3{,}193.3125, 2{,}606.6875, 117{,}424, 1{,}954.5, 65{,}317),$$

$$\hat{\Sigma}^{(0)} = \begin{bmatrix} 1.091790\text{E}4 & & & & & & \\ 9.971289\text{E}6 & 9.261894\text{E}9 & & & & & \\ 5.867500\text{E}4 & 5.261660\text{E}7 & 8.186470\text{E}5 & & & & \\ 3.272113\text{E}4 & 2.895040\text{E}7 & -1.081676\text{E}5 & 4.540351\text{E}5 & & & \\ 6.890906\text{E}5 & 6.424134\text{E}8 & 4.183820\text{E}6 & 1.653842\text{E}6 & 4.536314\text{E}7 & & \\ 4.774063\text{E}2 & 4.415418\text{E}5 & 2.787219\text{E}3 & 1.296031\text{E}3 & 3.086006\text{E}4 & 2.125000\text{E}1 & \\ 3.449687\text{E}5 & 3.218721\text{E}8 & 1.546034\text{E}6 & 1.047826\text{E}6 & 2.199559\text{E}7 & 1.522588\text{E}4 & 1.156305\text{E}7 \end{bmatrix}.$$

The vector of regression coefficients is:

$$\hat{\beta}^{(0)} = [1.506003\text{E}1, -3.581033\text{E}-2, -2.020214, -1.033223 \\ -5.111016\text{E}-2, 1.829138\text{E}3].$$

With the exception of the time variable X_6, these data may be reasonably modeled as multivariate normal. We compute here the maximum likelihood estimates in possible situations:

1) The "year" variable X_6 is considered exact, all other variables considered as being accurate only to the number of places given. That is, if Y_i denotes a "true" value, we have $Y_1 \in (X_1 - .05, X_1 + .05)$, $Y_i \in (X_i - .5, X_i + .5)$, $i = 2, 3, 4, 5, 7$; $Y_6 = X_6$.

2) All variables are considered accurate only to the number of places given. Identical to case 1 except that $Y_6 \in (X_6 - .5, X_6 + .5)$.

Case 1. Maximum likelihood estimates:

$$\hat{\mu} = (101.68125, 387{,}698.4375, 3{,}193.3125, 2{,}606.6875, 117{,}424, 1{,}954.5, 65{,}317),$$

$$\hat{\Sigma} = \begin{bmatrix} 1.091\underline{82}\text{E}4 & & & & & & \\ 9.971\underline{27}6\text{E}6 & 9.261894\text{E}9 & & & & & \\ 5.867\underline{46}8\text{E}4 & 5.261660\text{E}7 & 8.186\underline{46}9\text{B}5 & & & & \\ 3.272\underline{07}2\text{E}4 & 2.895040\text{E}7 & -1.081\underline{67}4\text{B}5 & 4.540\underline{03}5\text{E}5 & & & \\ 6.890\underline{92}7\text{E}5 & 6.424134\text{E}8 & 4.183820\text{B}6 & 1.653842\text{E}6 & 4.536314\text{E}7 & & \\ 4.774\underline{06}4\text{E}2 & 4.415418\text{E}5 & 2.787219\text{B}3 & 1.296031\text{E}3 & 3.086006\text{E}4 & 2.125000\text{E}1 & \\ 3.449\underline{68}6\text{E}5 & 3.218721\text{E}8 & 1.546034\text{B}6 & 1.047826\text{E}6 & 2.199559\text{E}7 & 1.522\underline{58}7\text{E}4 & 1.156305\text{E}7 \end{bmatrix},$$

$$\hat{\beta} = [1.\underline{483547}\text{E}1, -3.57\underline{5403}\text{E}-2, -2.01\underline{9400}, -1.03\underline{3005}, \\ -5.1\underline{53857}\text{E}-2, 1.82\underline{8820}\text{E}3].$$

In the covariance matrix and regression vector above, we have underlined digits which differ from the values in $\hat{\Sigma}^{(0)}$ and $\hat{\beta}^{(0)}$. On the one hand, the differences are not great in the regression vectors $\hat{\beta}$ and $\hat{\beta}^{(0)}$, although $\hat{\beta}_1^{(0)}$ and $\hat{\beta}_5^{(0)}$ have only two-digit accuracy. On the other hand, the uncertainty in the values is probably understated (Beaton, Rubin, and Barone judge the accuracy of the X'*s* at only two significant digits), and rounding in X_6 = time, the most severe case, has been eliminated.

Case 2. Maximum likelihood estimates:

$$\hat{\Sigma} = \begin{bmatrix} 1.091782\text{E}4 & & & & & & \\ 9.971248\text{E}6 & 9.261894\text{E}9 & & & & & \\ 5.867333\text{E}4 & 5.261660\text{E}7 & 8.186468\text{E}5 & & & & \\ 3.272049\text{E}4 & 2.895040\text{E}7 & -1.081675\text{E}5 & 4.540350\text{E}5 & & & \\ 6.890927\text{E}5 & 6.424134\text{E}8 & 4.183820\text{E}6 & 1.653842\text{E}6 & 4.536314\text{E}7 & & \\ 4.787684\text{E}2 & 4.427704\text{E}5 & 2.803387\text{E}3 & 1.291858\text{E}3 & 3.095919\text{E}4 & 2.141268\text{E}1 & \\ 3.449691\text{E}5 & 3.218721\text{E}8 & 1.546034\text{E}6 & 1.047826\text{E}6 & 2.199559\text{E}7 & 1.526929\text{E}4 & 1.156305\text{E}7 \end{bmatrix},$$

$$\hat{\beta} = [-2.864\text{E}1,\ 2.428\text{E}-2,\ -1.135,\ -7.672\text{E}-1,\ -2.671\text{E}-1,\ 8.561\text{E}2].$$

The regression coefficients $\hat{\beta}$ obtained are in complete disagreement with the values in $\hat{\beta}^{(0)}$, and the correlation matrix clearly shows that time (row six) is the culprit. The multivariate normality in this example may be suspect, but there is reasonable agreement with the coefficient values obtained by Beaton, Rubin, and Barone (1976) in a Monte Carlo study where an additional significant digit is generated at random and added to each of the observations. The mean and median values of $\hat{\beta}$ they obtained were:

mean $\hat{\beta}$: [−2.644E1, 3.44E−2, −9.637E−1, −7.209E−1, −2.804E−1, 6.370E2],

median $\hat{\beta}$: [−3.164E1, 3.63E−2, −9.384E−1, −6.978E−1, −3.065E−1, 6.526E2].

Appendix A. Values of c_{jk}

$$\text{Interval } (s-t, s+t), \quad X \sim \eta(\mu, \sigma^2), \quad w = \frac{1}{\sigma^2} = \frac{1}{v}.$$

$j = 1$

$k = 0$

$$-\frac{s(t^2w(t^2w(w(2t^2(s^2w(s^2w+4)+1)-21s^2)-42)+315)-945)}{945}$$

$k = 1$

$$\frac{t^2w(t^2w(w(2t^2(s^2w(5s^2w+12)+1)-63s^2)-42)+315)}{945}$$

$k = 2$

$$-\frac{st^4w^3(4t^2w(5s^2w+6)-63)}{945}$$

$k = 3$

$$\frac{t^4w^3(4t^2w(5s^2w+2)-21)}{945}$$

$k = 4$

$$-\frac{2st^6w^5}{189}$$

$k = 5$

$$\frac{2t^6w^5}{945}$$

$j=2$

$k=0$

$$-\frac{t^2(2w(t^2(w(t^2(2s^2w(s^2w(s^2w+5)+2)-1)-21s^2(s^2w+3))+21)+315s^2)-315)-945s^2}{945}$$

$k=1$

$$\frac{2st^2w(t^2w(w(2t^2(s^2w+3)(5s^2w+1)-63s^2)-84)+315)}{945}$$

$k=2$

$$-\frac{2t^4w^2(w(2t^2(2s^2w(5s^2w+9)+1)-63s^2)-21)}{945}$$

$k=3$

$$\frac{2st^4w^3(4t^2w(5s^2w+4)-21)}{945}$$

$k=4$

$$-\frac{4t^6w^4(5s^2w+1)}{945}$$

$k=5$

$$\frac{4st^6w^5}{945}$$

$j=3$

$k=0$

$$-\frac{s(t^2(w(t^2(w(t^2(s^2w(2s^2w(s^2w+6)+3)-14)-21s^2(s^2w+4))+105)+315s^2)-315)-315s^2)}{315}$$

$k=1$

$$\frac{t^2w(t^2(w(t^2(s^2w(10s^2w(s^2w+4)+1)-12)-63s^2(s^2w+2))+63)+315s^2)}{315}$$

$k=2$

$$-\frac{st^4w^2(w(t^2(2s^2w+5)(10s^2w-1)-63s^2)-42)}{315}$$

$k=3$

$$\frac{t^4w^3(t^2(4s^2w(5s^2w+6)-3)-21s^2)}{315}$$

$k=4$

$$-\frac{2st^6w^4(5s^2w+2)}{315}$$

$k=5$

$$\frac{2s^2t^6w^5}{315}$$

$j=4$

$k=0$

$$-\frac{\begin{array}{r}8s^8t^6w^5+56s^6t^6w^4-4s^4t^6w^3-84s^6t^4w^3-192s^2t^6w^2-420s^4t^4w^2+36t^6w+1008s^2t^4w\\+1260s^4t^2w-189t^4-1890s^2t^2-945s^4\end{array}}{945}$$

$k=1$

$$\frac{4st^2w(t^2(w(t^2(s^2w(2s^2w(5s^2+24)-13)-54)-21s^2(3s^2w+8))+189)+315s^2)}{945}$$

$k=2$

$$-\frac{4t^4w^2(t^2(s^2w(20s^2w(s^2w+3)-21)-9)-63s^2(s^2w+1))}{945}$$

$k=3$

$$\frac{4st^4w^3(t^2(4s^2w(5s^2w+8)-9)-21s^2)}{945}$$

$k=4$

$$-\frac{8s^2t^6w^4(5s^2w+3)}{945}$$

$k=5$

$$\frac{8s^3t^6w^5}{945}$$

$j=5$

$k=0$

$$-\frac{s(2s^8t^6w^5+16s^6t^6w^4-8s^4t^6w^3-21s^6t^4w^3-112s^2t^6w^2-126s^4t^4w^2+63t^6w+462s^2t^4w+315s^4t^2w-189t^4-630s^2t^2-189s^4)}{189}$$

$k=1$

$$\frac{t^2w(10s^8t^4w^4+56s^6t^4w^3-36s^4t^4w^2-63s^6t^2w^2-144s^2t^4w-210s^4t^2w+27t^4+378s^2t^2+315s^4)}{189}$$

$k=2$

$$-\frac{st^4w^2(2t^2(s^2w(2s^2w(5s^2w+18)-23)-18)-21s^2(3s^2w+4))}{189}$$

$k=3$

$$\frac{s^2t^4w^3(2t^2(10s^2w(s^2w+2)-9)-21s^2)}{189}$$

$k=4$

$$-\frac{2s^3t^6w^4(5s^2w+4)}{189}$$

$k=5$

$$\frac{2s^4t^6w^5}{189}$$

$j=6$

$k=0$

$$-\frac{(4s^{10}t^6w^5+36s^8t^6w^4-36s^6t^6w^3-42s^8t^4w^3-430s^4t^6w^2-294s^6t^4w^2+450s^2t^6w+1470s^4t^4w+630s^6t^2w-45t^6-945s^2t^4-1575s^4t^2-315s^6)}{315}$$

$k=1$

$$\frac{2st^2w(10s^8t^4w^4+64s^6t^4w^3-68s^4t^4w^2-63s^6t^2w^2-300s^2t^4w-252s^4t^2w+135t^4+630s^2t^2+315s^4)}{315}$$

$k=2$

$$-\frac{2s^2t^4w^2(2t^2(2s^2w(s^2w+5)(5s^2w-4)-45)-21s^2(3s^2w+5))}{315}$$

$k=3$

$$\frac{2s^3t^4w^3(2t^2(2s^2w(5s^2w+12)-15)-21s^2)}{315}$$

$k=4$

$$-\frac{4s^4t^6w^4(s^2w+1)}{63}$$

$k=5$

$$\frac{4s^5t^6w^5}{315}$$

$j=7$

$k=0$

$$\frac{-s(2s^{10}t^6w^5+20s^8t^6w^4-31s^6t^6w^3-21s^8t^4w^3-366s^4t^6w^2-168s^6t^4w^2+585s^2t^6w+1071s^4t^4w+315s^6t^2w-135t^6-945s^2t^4-945s^4t^2-135s^6)}{135}$$

$k=1$

$$\frac{s^2t^2w(10s^8t^4w^4+72s^6t^4w^3-109s^4t^4w^2-63s^6t^2w^2-540s^2t^4w-294s^4t^2w+405t^4+945s^2t^2+315s^4)}{315}$$

$k=2$

$$-\frac{s^3t^4w^2(t^2(s^2w(4s^2w(5s^2w+24)-123)-180(-63s^2)s^2w+2))}{135}$$

$k=3$

$$\frac{s^4t^4w^3(t^2(4s^2w(5s^2w+14)-45)-21s^2)}{135}$$

$k=3$

$$-\frac{2s^5t^6w^4(5s^2w+6)}{135}$$

$k=5$

$$\frac{2s^6t^6w^5}{135}$$

$j=8$

$k=0$

$$\frac{-s^2(16s^{10}t^6w^5+176s^8t^6w^4-376s^6t^6w^3-168s^8t^4w^3-4592s^4t^6w^2-1512s^6t^4w^2+10080s^2t^6w+11760s^4t^4w+2520s^6t^2w-3780t^6-13230s^2t^4-8820s^4t^2-945s^6)}{945}$$

$k=1$

$$\frac{8s^3t^2w(10s^8t^4w^4+80s^6t^4w^3-159s^4t^4w^2-68s^6t^2w^2-882s^2t^4w-336s^4t^2w+945t^4+1323s^2t^2+315s^4)}{945}$$

$k=2$

$$\frac{8s^4t^4w^2(t^2(s^2w(4s^2w(5s^2w+27)-175)-315)-21s^2(3s^2w+7))}{945}$$

$k = 3$

$$\frac{8s^2t^4w^3(t^2(4s^2w(5s^2w+16)-63)-21s^2)}{945}$$

$k = 4$

$$-\frac{16s^6t^6w^4(5s^2w+7)}{945}$$

$k = 5$

$$\frac{16s^7t^6w^5}{945}$$

$j = 9$

$k = 0$

$$\frac{-s^3(2s^{10}t^6w^5+24s^8t^6w^4-66s^6t^6w^3-21s^8t^4w^3-848s^4t^6w^2-210s^6t^4w^2+2394s^2t^6w+1932s^4t^4w+315s^6t^2w-1260t^2-2646s^2t^2-1260s^4t^2-105s^6)}{105}$$

$k = 1$

$$\frac{s^4t^2w(10s^8t^4w^4+88s^6t^4w^3-218s^4t^4w^2-63s^6t^2w^2-1344s^2t^4w-387s^4t^2w+1890t^4+1764s^2t^2+315s^4)}{105}$$

$k = 2$

$$\frac{s^5t^4w^2(4t^2(s^2w(5s^2w(s^2w+6)-59)-126)-21s^2(3s^2w+8))}{105}$$

$k = 3$

$$\frac{s^6t^4w^3(4t^2(s^2w(5s^2w+18)-21)-21s^2)}{105}$$

$k = 4$

$$-\frac{2s^7t^6w^4(5s^2w+8)}{105}$$

$k = 5;$

$$\frac{2s^8t^6w^5}{105}$$

$j = 10$

$k = 0$

$$\frac{-s^4(4s^{10}t^6w^5+52s^8t^6w^4-176s^6t^6w^3-42s^8t^4w^3-2394s^4t^6w^2-462s^6t^4w^2+8316s^2t^6w+4914s^4t^4w+630s^6t^2w-5670t^6-7938s^2t^4-2835s^4t^2-189s^6)}{189}$$

$k = 1$

$$\frac{2s^5t^2w(10s^8t^4w^4+96s^6t^4w^3-286s^4t^4w^2-63s^6t^2w^2-1944s^2t^4w-420s^4t^2w+3402t^4+2268s^2t^2+315s^4)}{189}$$

$k = 2$

$$\frac{2s^6t^4w^2(2t^2(s^2w(2s^2w(5s^2w+33)-153)-378)-63s^2(s^2w+3))}{189}$$

$k = 3$

$$\frac{2s^7t^4w^3(4t^2(5s^2w(s^2w+4)-27)-21s^2)}{189}$$

$k = 4$

$$-\frac{4s^8t^6w^4(5s^2w+9)}{189}$$

$k = 5$

$$\frac{4s^9t^6w^5}{189}$$

Acknowledgment. The author would like to thank the Mathlab Group at the Laboratory for Computer Science at MIT for the use of the MACSYMA symbolic computation system.

REFERENCES

ALBERT E. BEATON, DONALD B. RUBIN AND JOHN L. BARONE (1976), *The acceptability of regression solutions: another look at computational accuracy*, J. Amer. Statist. Assoc., 71, pp. 158–168.

A. P. DEMPSTER, N. M. LAIRD AND D. B. RUBIN (1977), *Maximum likelihood from incomplete data via the EM algorithm*, J. Roy. Statist. Soc. (B), 39, pp. 1–38.

WARREN T. DENT AND DAVID C. CAVENDER (1977), *More on computational accuracy in regression*, J. Amer. Statist. Assoc., 72, pp. 598–601.

JAMES W. LONGLEY (1967), *An appraisal of least squares programs for the electronic computer from the point of view of the user*, J. Amer. Statist. Assoc., 62, pp. 819–841.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0013 $01.25/0

A NEW CLASS OF CYCLIC MULTISTEP FORMULAE FOR STIFF SYSTEMS*

P. TISCHER† AND R. SACKS-DAVIS†

Abstract. A new class of two-stage cyclic multistep formulae is developed for the numerical integration of systems of ordinary differential equations. The formulae are suitable for the solution of both stiff and nonstiff systems. They are derived by matching the characteristic polynomials of two-stage cyclic formulae with those of a certain class of second derivative formulae.

Key words. cyclic multistep formulae, stiff ordinary differential equations

1. Introduction. In this paper we consider methods based on the cyclic use of linear multistep formulae for the numerical integration of the differential equation

$$y' = f(x, y), \qquad x \in [a, b],$$

with initial condition

$$y(x_0) = y_0, \qquad x_0 = a.$$

When M different multistep formulae are used cyclically to obtain an approximate solution at M successive values of the independent variable, the resulting method is called an M-stage cyclic composite method.

The properties of these methods are, in general, different from those of their constituent formulae. For example, Dahlquist [2] has shown that a k-step linear multistep formula of order p can be stable only if

$$p \leqq \begin{cases} k+1 & \text{if } k \text{ is odd,} \\ k+2 & \text{if } k \text{ is even.} \end{cases}$$

Yet, Donelson and Hansen [3] derived stable cyclic methods from k-step formulae of order $2k-1$ for $k=2$, 3 and 4. Furthermore, these methods were shown to be globally convergent to order $2k$. Donelson and Hansen's approach to order is based on the construction of an equivalent method or "auxiliary system." A different approach to order is taken by Albrecht [1], who expresses an M-stage cyclic method as a one-step method

$$w^*_{n+1} = \hat{A} w^*_n + h\hat{\phi}(x_n, w^*_n, w^*_{n+1}; h)$$

and considers the components of the local error in the directions of the eigenvectors of the matrix $\hat{A}$. Albrecht also derives higher order cyclic k-step methods that are stiffly stable.

A class of stiffly stable, cyclic composite formulae that has been implemented in a variable stepsize, variable order code called STINT is described in [6]. The formulae used in STINT exhibit improved stiff stability properties, order by order, over the backward differentiation formulae (BDF) and include formulae of orders up to 7.

In this paper we derive a new class of cyclic formulae. Some desirable properties of cyclic formulae suitable for solving both stiff and nonstiff equations are proposed below.

* Received by the editors August 26, 1981, and in revised form April 30, 1982.
† Department of Computer Science, Monash University, Clayton, Victoria, Australia 3168.

(i) The extraneous roots at the origin should be equal to zero (as with the Adams formulae).

(ii) The roots at infinity should be equal to zero (as with the BDF formulae).

(iii) The formulae should exhibit better stiff-stability properties, order by order, than the BDF formulae.

There are also implementation considerations which are often overlooked when deriving new methods. When solving stiff equations, it is common to use a Newton iteration to solve the corrector equation. The iteration matrix used is of the form $\alpha_k I - h\beta_k \partial f/\partial y$, where α_k and β_k are the leading coefficients of the corrector formulae. This matrix is expensive to reevaluate in factored form (see also [5]). In order to reduce the number of reevaluations necessary to ensure fast convergence of the Newton scheme over a cycle, it is desirable that

(iv) The leading coefficients of the multistep formulae should not vary within a cycle.

In current implementations, the stepsize is not allowed to vary within a cycle. This stepsize strategy best reflects theoretical studies of cyclic formulae. However, a disadvantage of this approach is that if a step is rejected during the final stages of a cycle, a stepsize reduction will be required and the work in calculating the solution during the previous stages will be wasted.

For this reason we require that

(v) The number of stages within the cycle is limited to two.

Finally, it is required that

(vi) The leading truncation error coefficients of the multistep formulae used in the cycle should not be "large".

We note that the formulae used in the code STINT fail to satisfy conditions (i), (iv) and (v).

In this paper we observe that the stiff-stability properties of two-stage cyclic formulae closely resemble those of second derivative formulae. By matching the characteristic polynomials of the two-stage cyclic formulae with those of a certain class of second derivative formulae we are able to find a class of cyclic formulae satisfying conditions (i)–(vi). The new formulae exhibit improved stiff stability properties over both the BDF and STINT formulae and include formulae of orders up to 8. These new methods are globally convergent to order k and are based on 2-stage cyclic formulae whose constituent multistep formulae are kth order. We also derive methods with stages of order k that are globally convergent to order $k+1$ in the sense of Albrecht.

2. Two-stage cyclic formulae. Let us consider a two-stage cyclic composite formula where each stage is a k-step linear multistep formula. We may write the first stage as

$$\begin{aligned}\alpha_k^{(1)} y_{2n+k}^* + \alpha_{k-1}^{(1)} y_{2n+k-1}^* + \cdots + \alpha_0^{(1)} y_{2n}^* \\ = h(\beta_k^{(1)} f_{2n+k}^* + \cdots + \beta_0^{(1)} f_{2n}^*), \qquad \alpha_k^{(1)} := 1,\end{aligned}$$

and similarly the second stage as

$$\begin{aligned}\alpha_k^{(2)} y_{2n+k+1}^* + \alpha_{k-1}^{(2)} y_{2n+k}^* + \cdots + \alpha_0^{(2)} y_{2n+1}^* \\ = h(\beta_k^{(2)} f_{2n+k+1}^* + \cdots + \beta_0^{(2)} f_{2n+1}^*), \qquad \alpha_k^{(2)} := 1.\end{aligned}$$

We will adopt the notation of Albrecht [1] and set

$$z^*_{2n+1} := (y^*_{2n+1}, y^*_{2n+2}, \cdots, y^*_{2n+k})^T,$$

$$A^{(1)} := \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_0^{(1)} & -\alpha_1^{(1)} & -\alpha_2^{(1)} & \cdots & -\alpha_{k-1}^{(1)} \end{pmatrix}$$

and

$$\phi^{(1)}(x_{2n}, z^*_{2n}, z^*_{2n+1}; h) := (0, 0, \cdots, 0, \beta_k^{(1)} f_{2n+k} + \cdots + \beta_0^{(1)} f_{2n})^T.$$

We may then express the first stage as the one-step procedure

$$z^*_{2n+1} = A^{(1)} z^*_{2n} + h\phi^{(1)}(x_{2n}, z^*_{2n}, z^*_{2n+1}; h).$$

If we denote by z_{2n+1} the vector composed of the exact solution:

$$z_{2n+1} = (y(x_{2n+1}), y(x_{2n+2}), \cdots, y(x_{2n+k}))^T$$

and if the stages of the cycle are of order p, then for sufficiently differentiable y we may write

$$z_{2n+1} = A^{(1)} z_{2n} + h\phi^{(1)}(x_{2n}, z_{2n}, z_{2n+1}; h) + h^{p+1} y^{(p+1)}(x_{2n}) t^{(1)} + O(h^{p+2})$$

with $t^{(1)} = (0, 0, \cdots, 0, C^{(1)}_{p+1})^T$.

Similarly, for the second stage we may write

$$z_{2n+2} = A^{(2)} z_{2n+1} + h\phi^{(2)}(x_{2n+1}, z_{2n+1}, z_{2n+2}; h) + h^{p+1} y^{(p+1)}(x_{2n}) t^{(2)} + O(h^{p+2})$$

with $t^{(2)} = (0, 0, \cdots, 0, C^{(2)}_{p+1})^T$.

Here, $C^{(i)}_{p+1}$, $i = 1, 2$, are the error constants of the stages. Finally, if we set $w_n := z_{2n}$ we obtain

$$w_{n+1} = \hat{A} w_n + h\hat{\phi}(x_n, w_n, w_{n+1}; h) + h^{p+1} y^{(p+1)}(x_n)\hat{t} + O(h^{p+2}),$$

where

$$\hat{A} := A^{(2)} A^{(1)}$$

and

$$\hat{t} := t^{(2)} + A^{(2)} t^{(1)} = (0, 0, \cdots, 0, C^{(1)}_{p+1}, C^{(2)}_{p+1} - \alpha^{(2)}_{k-1} C^{(1)}_{p+1})^T.$$

Thus we may express the two-stage cyclic composite method as a one-step method:

$$w^*_{n+1} = \hat{A} w^*_n + h\hat{\phi}(x_n, w^*_n, w^*_{n+1}; h), \tag{1}$$

where $w^*_n := z^*_{2n}$. The method has a local error of the form $h^{p+1} y^{(p+1)}(x_n)\hat{t} + O(h^{p+2})$ and the stability of the method is characterized by the eigenvalues of the matrix $\hat{A} = A^{(2)} A^{(1)}$.

DEFINITION. Let the eigenvalues of $\hat{A} := A^{(2)} A^{(1)}$ be μ_i, $i = 1, 2, \cdots, k$. Then the matrix $\hat{A}$ is strongly D-stable if $\mu_1 = 1$ and $|\mu_i| < 1$, $i = 2, 3, \cdots, k$.

Generally, if the component formulae of the cycle are of order p, then the method will converge with order p. However, Albrecht [1] has shown that it is possible for the method to converge with order $p+1$ if the vector $\hat{t}$ appearing in the expression for the local error has no component in the direction of the eigenvector of $\hat{A}$ corresponding to $\mu_1 = 1$. That is,

$$\hat{t} = \sum_{i=1}^{k} d_i u(\mu_i) \text{ with } d_1 = 0, \tag{2}$$

where $\{u(\mu_i),\ i = 1, 2, \cdots, k\}$ are the eigenvectors of $\hat{A}$. We assume that $\hat{A}$ has only linear elementary divisors.

THEOREM (Albrecht [1]). *If the following conditions hold*:

(a) $\hat{A}$ *is strongly D-stable,*

(b) $\hat{\phi}$ *satisfies the Lipschitz condition*

$$\|\hat{\phi}(x, y_1, z_1; h) - \hat{\phi}(x, y_2, z_2; h)\| \leqq k_1\|y_1 - y_2\| + k_2\|z_1 - z_2\|$$

for $x \in [a, b]$ *and* $h \in (0, h_0]$,

(c) *the starting values are of order* $p+1$,

(d) *the local error is of the form* $h^{p+1}y^{(p+1)}(x_n)\hat{t} + O(h^{p+2})$ *with* $\hat{t}$ *satisfying* (2),

then the method (1) *converges with order* $p+1$, *i.e.*,

$$\|w_n^* - w_n\| = O(h^{p+1}).$$

In order to simplify condition (d), Albrecht noted that an equivalent condition to (2) is to require that

$$p^T\hat{t} = 0 \quad \text{where } \hat{A}^T p = p, \quad p \neq 0.$$

Now, we may use the fact that the matrices $A^{(1)}$ and $A^{(2)}$ are Frobenius matrices to further simplify condition (d). In fact, for the 2-stage cyclic multistep formulae that we are considering, the condition for higher order convergence reduces to a simple relation involving the coefficients of the component multistep formulae.

LEMMA. *Let*

$$\alpha_{\text{even}}^{(i)} = \alpha_0^{(i)} + \alpha_2^{(i)} + \alpha_4^{(i)} + \cdots + \alpha_{2\lfloor k/2 \rfloor}^{(i)}.$$

Then, if $\hat{A}$ *is strongly D-stable, condition* (d) *is satisfied for 2-stage cyclic multistep formulae if either*

(i) $\alpha_{\text{even}}^{(1)} = 0$, $\alpha_{\text{even}}^{(2)} \neq 0$ *and* $C_{p+1}^{(1)} = 0$

or

(ii) $\alpha_{\text{even}}^{(1)} \neq 0$ *and* $C_{p+1}^{(2)} + (\alpha_{\text{even}}^{(2)}/\alpha_{\text{even}}^{(1)})C_{p+1}^{(1)} = 0$.

Proof. Let $p = (p_1, p_2, \cdots, p_k)^T$. Then we may write the equations $(\hat{A}^T - I)p = 0$ as

$$\begin{pmatrix}
-1 & 0 & 0 & 0 & \cdots & 0 & -\alpha_0^{(1)} & \alpha_{k-1}^{(2)}\alpha_0^{(1)} \\
0 & -1 & 0 & 0 & \cdots & 0 & -\alpha_1^{(1)} & \alpha_{k-1}^{(2)}\alpha_1^{(1)} - \alpha_0^{(2)} \\
1 & 0 & -1 & 0 & \cdots & 0 & -\alpha_2^{(1)} & \alpha_{k-1}^{(2)}\alpha_2^{(1)} - \alpha_1^{(2)} \\
0 & 1 & 0 & -1 & \cdots & 0 & -\alpha_3^{(1)} & \alpha_{k-1}^{(2)}\alpha_3^{(1)} - \alpha_2^{(2)} \\
 & \cdots & & & \cdots & & \cdots & \\
0 & 0 & 0 & 0 & \cdots & 0 & -1 - \alpha_{k-2}^{(1)} & \alpha_{k-1}^{(2)}\alpha_{k-2}^{(1)} - \alpha_{k-3}^{(2)} \\
0 & 0 & 0 & 0 & \cdots & 1 & -\alpha_{k-1}^{(1)} & \alpha_{k-1}^{(2)}\alpha_{k-1}^{(1)} - \alpha_{k-2}^{(2)} - 1
\end{pmatrix}
\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ \cdots \\ p_{k-1} \\ p_k \end{pmatrix} = 0.$$

If we label each of the above equations from 1 to k and sum either the even numbered

equations or the odd numbered equations, then using the facts that if

$$\alpha_{\text{odd}}^{(i)} = \alpha_1^{(i)} + \alpha_3^{(i)} + \cdots + a_{2\lfloor (k+1)/2 \rfloor - 1}$$

then

$$\alpha_{\text{even}}^{(i)} + \alpha_{\text{odd}}^{(i)} = 0, \qquad i = 1, 2,$$

(by consistency) and

$$\alpha_k^{(i)} = 1, \qquad i = 1, 2,$$

we obtain the following single equation:

$$\alpha_{\text{even}}^{(1)} p_{k-1} - (\alpha_{k-1}^{(2)} \alpha_{\text{even}}^{(1)} + \alpha_{\text{even}}^{(2)}) p_k = 0. \tag{3}$$

Now, if $\alpha_{\text{even}}^{(1)} = \alpha_{\text{even}}^{(2)} = 0$ then the coefficient matrix, $\hat{A}^T - I$, is of rank at most $k-2$. Thus $\hat{A}$ has multiple eigenvalues equal to one and this contradicts the requirement of strong D-stability. There are two cases to consider

Case 1. If $\alpha_{\text{even}}^{(1)} = 0$, then $\alpha_{\text{even}}^{(2)} \neq 0$ and the eigenvector p takes the form

$$(p_1, p_2, \cdots, p_{k-1}, 0)^T$$

with $p_{k-1} \neq 0$. Then the condition $p^T \hat{t} = 0$ takes the form $C_{p+1}^{(1)} = 0$.

Case 2. If $\alpha_{\text{even}}^{(1)} \neq 0$ then p takes the form

$$\left(p_1, p_2, \cdots, p_{k-2}, \frac{\alpha_{k-1}^{(2)} \alpha_{\text{even}}^{(1)} + \alpha_{\text{even}}^{(2)}}{\alpha_{\text{even}}^{(1)}} p_k, p_k \right)^T$$

with $p_k \neq 0$. Then the condition $p^T \hat{t} = 0$ becomes

$$\left(\frac{\alpha_{k-1}^{(2)} \alpha_{\text{even}}^{(1)} + \alpha_{\text{even}}^{(2)}}{\alpha_{\text{even}}^{(1)}} \right) C_{p+1}^{(1)} + C_{p+1}^{(2)} - \alpha_{k-1}^{(2)} C_{p+1}^{(1)} = 0$$

and we have condition (ii) of the lemma.

We note that by condition (i) of the lemma we have the interesting result, reported by Donelson and Hansen [3], that a strongly D-stable 2-stage cyclic method composed of Simpson's rule (which is 4th order) and any other 3rd order multistep formula will be globally convergent to order 4. Condition (ii) of the lemma may be used to find 2-stage cyclic methods which are globally convergent to order $p+1$ but whose constituent formulae are each of order p.

We now investigate the stiff-stability properties of 2-stage cyclic methods. For the linear problem $y' = \lambda y$, (1) takes the form

$$w_{n+1}^* = \hat{S}(h\lambda) w_n^*,$$

where

$$\hat{S}(h\lambda) = S^{(2)}(h\lambda) S^{(1)}(h\lambda)$$

and

$$S^{(i)}(h\lambda) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & 0 & \cdots & 1 \\ a_0^{(i)} & a_1^{(i)} & a_2^{(i)} & \cdots & a_{k-1}^{(i)} \end{pmatrix}, \qquad i = 1, 2,$$

$$a_j^{(i)} = \frac{-\alpha_j^{(i)} + h\lambda \beta_j^{(i)}}{\alpha_k^{(i)} - h\lambda \beta_k^{(i)}}, \qquad j = 0, 1, \cdots, k-1.$$

The stiff-stability properties of the cyclic method will be characterized by the eigenvalues of $\hat{S}(h\lambda)$. For $k=3$ the characteristic equation, $\phi_k(t) := \det(\hat{S}(h\lambda)-tI)$, is given by

$$\begin{aligned}\phi_3(t) = \lambda^2 h^2[&(-\beta_3^{(1)}\beta_3^{(2)})t^3+(-\beta_3^{(1)}\beta_1^{(2)}+\beta_2^{(1)}\beta_2^{(2)}-\beta_1^{(1)}\beta_3^{(2)})t^2\\ &+(\beta_2^{(1)}\beta_0^{(2)}-\beta_1^{(1)}\beta_1^{(2)}+\beta_0^{(1)}\beta_2^{(2)})t+(\beta_0^{(1)}\beta_0^{(2)})]\\ +\lambda h[&(\alpha_3^{(1)}\beta_3^{(2)}+\beta_3^{(1)}\alpha_3^{(2)})t^3\\ &+(\alpha_3^{(1)}\beta_1^{(2)}-\alpha_2^{(1)}\beta_2^{(2)}+\alpha_1^{(1)}\beta_3^{(2)}+\beta_3^{(1)}\alpha_1^{(2)}-\beta_2^{(1)}\alpha_2^{(2)}+\beta_1^{(1)}\alpha_3^{(2)})t^2\\ &+(-\alpha_2^{(1)}\beta_0^{(2)}+\alpha_1^{(1)}\beta_1^{(2)}-\alpha_0^{(1)}\beta_2^{(2)}-\beta_2^{(1)}\alpha_0^{(2)}+\beta_1^{(1)}\alpha_1^{(2)}-\beta_0^{(1)}\alpha_2^{(2)})t\\ &+(-\alpha_0^{(1)}\beta_0^{(2)}-\beta_0^{(1)}\alpha_0^{(2)})]\\ +[&(-\alpha_3^{(1)}\alpha_3^{(2)})t^3+(-\alpha_3^{(1)}\alpha_1^{(2)}+\alpha_2^{(1)}\alpha_2^{(2)}-\alpha_1^{(1)}\alpha_3^{(2)})t^2\\ &+(\alpha_2^{(1)}\alpha_0^{(2)}-\alpha_1^{(1)}\alpha_1^{(2)}+\alpha_0^{(1)}\alpha_2^{(2)})t].\end{aligned} \tag{4}$$

In general, when the component multistep formulae are k-step, the characteristic polynomial, $\phi_k(t)$, may be expressed as

$$\phi_k(t) = \sum_{j=0}^{k} (p_j - h\lambda q_j + h^2\lambda^2 m_j)t^j, \tag{5}$$

where

$$\begin{aligned} p_j &= \sum (-1)^r \alpha_r^{(1)}\alpha_s^{(2)},\\ q_j &= \sum (-1)^r (\alpha_r^{(1)}\beta_s^{(2)}+\beta_r^{(1)}\alpha_s^{(2)}),\\ m_j &= \sum (-1)^r (\beta_r^{(1)}\beta_s^{(2)}),\end{aligned}$$

the sums being taken over all r, s satisfying $r+s=2j$ & $r, s \in [0, \min(k, 2j)]$.

3. A class of second derivative formulae. Let us consider the following class of k-step, kth order second derivative formula

$$\sum_{j=0}^{k} \bar{\alpha}_j y_{n+j} = h \sum_{j=0}^{k} \bar{\beta}_j y'_{n+j} + h^2 \sum_{j=0}^{k} \bar{\gamma}_j y''_{n+j}.$$

When applied to the linear problem $y' = \lambda y$, the associated characteristic polynomial is

$$\psi_k(t) = \sum_{j=0}^{k} (\bar{\alpha}_j - h\lambda\bar{\beta}_j - h^2\lambda^2\bar{\gamma}_j)t^j.$$

The similarity between $\phi_k(t)$ and $\psi_k(t)$ suggests that we may be able to find 2-stage cyclic formulae with the same stiff-stability properties as second derivative formulae. Because we desire that the cyclic method exhibit properties (i), (ii) and (iv) of § 1, we restrict our attention to the class of second derivative formulae of the form

$$y_{n+k} = y_{n+k-1} + h \sum_{j=0}^{k} \bar{\beta}_j y'_{n+j} + h^2\bar{\gamma}_k y''_{n+k}, \tag{6}$$

where

$$\bar{\gamma}_k = -\left(\frac{\bar{\beta}_k}{2}\right)^2.$$

The requirement that the formula be of order k imposes k conditions on the $\bar{\beta}_j$, $j = 0, 1, \cdots, k$, so that formula (6) has one degree of freedom. The class of formulae (6) was considered by Enright [4] who was motivated by the fact that the Newton iteration matrix used to solve (6) could be expressed as a perfect square. Enright chose to use the one degree of freedom to raise the order of the formula to $k+1$. Alternatively, it is possible to use this degree of freedom to improve the stiff-stability properties of the formulae and we determined the formulae of order k which maximized the angle α in the definition of $A(\alpha)$-stability. The angle α for these formulae appear in Table 1 together with the values of the leading coefficient, $\bar{\beta}_k$ (calculated to 3 significant

TABLE 1
A comparison of α-values.

Order	BDF	STINT	New formulae
2	90.00	90.00	90.00 ($\bar{\beta}_2 = 0.575$)
3	86.03	89.43	90.00 ($\bar{\beta}_3 = 1.72$)
4	73.35	80.88	90.00 ($\bar{\beta}_4 = 1.72$)
5	51.84	77.48	86.64 ($\bar{\beta}_5 = 1.59$)
6	17.84	63.25	76.32 ($\bar{\beta}_6 = 1.40$)
7		33.53	57.66 ($\bar{\beta}_7 = 1.275$)
8			22.15 ($\bar{\beta}_8 = 1.18$)

figures). In cases where there is more than one kth order formula with the stability angle listed in Table 1, the smallest value of $\bar{\beta}_k$ was determined. (The computational advantages of formulae with small leading coefficients are discussed by Shampine [7]). For a comparison, we give the values of α for the BDF and STINT formulae in Table 1. It may be observed that the new formulae have far better $A(\alpha)$-stability properties than either the BDF or STINT formulae.

In the following section we will derive 2-stage cyclic multistep formulae with the same $A(\alpha)$-stability properties as the new second derivative formulae. Unlike a second derivative method, a code based on cyclic multistep formulae will not require a user-supplied analytic Jacobian.

4. The new formulae. A 2-stage cyclic method for which the component formulae are k-step linear multistep formulae contains $4k+4$ free parameters. Two of these are scaling factors. The requirement that each formula be of order k accounts for $2k+2$ parameters. The requirements that $k-1$ roots at the origin be zero, that k roots at infinity be zero and that the two linear multistep formulae have the same leading coefficients account for the remaining $2k$ free parameters. Thus conditions (i), (ii) and (iv) together with the requirement that the component formulae be of order k uniquely define a class of cyclic formulae. Unfortunately, it turns out that these cyclic formulae have poorer stiff-stability properties than the BDF formulae of the corresponding orders.

In order to derive cyclic methods with the same stiff-stability properties as the k-step, kth order second derivative formulae described in the previous section, it is necessary to introduce another free parameter.

Consider the cyclic use of a k-step kth order linear multistep formulae with a $(k+1)$-step kth order linear multistep formula as illustrated in Fig. 1. Although the $(k+1)$-step formula requires $(k+1)$ past values, one of these values is calculated within the cycle, so the cycle as a whole requires only k-past values. The characteristic

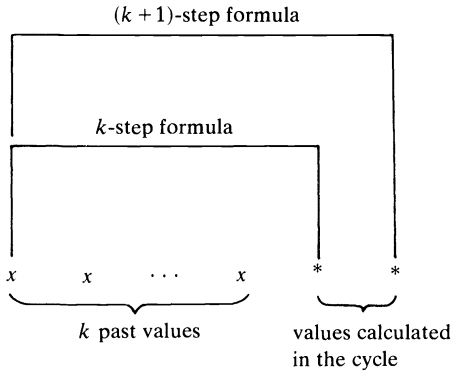


FIG. 1. *Two stage cyclic formulae.*

polynomial of the $(k-1)$-step, k-step cycle may be obtained from (5) by setting $\alpha_0^{(1)} = \beta_0^{(1)} = 0$.

To illustrate the derivation of the new class of cyclic formulae, let us consider the 2-step/3-step second order case. The characteristic polynomial for this method is given by $\phi_3(t)/t$ with $\alpha_0^{(1)} = \beta_0^{(1)} = 0$, where $\phi_3(t)$ is defined by (4). There are 14 coefficients to be determined: $\alpha_i^{(1)}, \beta_i^{(1)}$, $i = 1, 2, 3$, and $\alpha_i^{(2)}, \beta_i^{(2)}$, $i = 0, 1, 2, 3$. We have

(7)
(a) $\alpha_3^{(1)} = 1$,
(b) $\alpha_3^{(2)} = 1$,
(c) $\beta_3^{(1)} = \beta_3^{(2)}$,
(d) $\alpha_1^{(1)} + \alpha_2^{(1)} + \alpha_3^{(1)} = 0$,
(e) $\alpha_1^{(1)} + 2\alpha_2^{(1)} + 3\alpha_3^{(1)} - \beta_1^{(1)} - \beta_2^{(1)} - \beta_3^{(1)} = 0$,
(f) $1/2(\alpha_1^{(1)} + 4\alpha_2^{(1)} + 9\alpha_3^{(1)}) - \beta_1^{(1)} - 2\beta_2^{(1)} - 3\beta_3^{(1)} = 0$,
(g) $\alpha_0^{(2)} + \alpha_1^{(2)} + \alpha_2^{(2)} + \alpha_3^{(2)} = 0$,
(h) $\alpha_1^{(2)} + 2\alpha_2^{(2)} + 3\alpha_3^{(2)} - \beta_0^{(2)} - \beta_1^{(2)} - \beta_2^{(2)} - \beta_3^{(2)} = 0$,
(i) $1/2(\alpha_1^{(2)} + 4\alpha_2^{(2)} + 9\alpha_3^{(2)}) - \beta_1^{(2)} - 2\beta_2^{(2)} - 3\beta_3^{(2)} = 0$,
(j) $\alpha_2^{(1)}\alpha_0^{(2)} - \alpha_1^{(1)}\alpha_1^{(2)} = 0$,
(k) $-\beta_3^{(1)}\beta_1^{(2)} + \beta_2^{(1)}\beta_2^{(2)} - \beta_1^{(1)}\beta_3^{(2)} = 0$,
(l) $\beta_2^{(1)}\beta_0^{(2)} - \beta_1^{(1)}\beta_1^{(2)} = 0$,
(m) $\beta_3^{(1)} = \beta^*$.

Equations (7a) and (7b) result from our choice of scaling factor, (7c) is the requirement that the leading coefficients within the cycle do not vary, (7d)–(7i) are the conditions that the component multistep formulae are second order, (7j)–(7l) are the conditions that the extraneous root at the origin as well as the two roots at infinity are zero and (7m) is the condition that the cyclic method have the same characteristic polynomial as the 2nd order second derivative formulae (6) with stability angle given in Table 1. The value of $\beta^* = \bar{\beta}_2$ is also given in Table 1.

There are 14 coefficients to be determined and (7a)–(7m) consists of 13 equations. It is possible to parameterize the solutions to the nonlinear equations (7) in the following form.

Stage 1 *coefficients.*

$$\alpha_3^{(1)} = 1, \qquad \beta_3^{(1)} = \beta^*,$$
$$\alpha_2^{(1)} = \tilde{\alpha}_2^{(1)}, \qquad \beta_2^{(1)} = \tilde{\beta}_2^{(1)},$$
$$\alpha_1^{(1)} = \tilde{\alpha}_1^{(1)}, \qquad \beta_1^{(1)} = \tilde{\beta}_1^{(1)},$$

Stage 2 *coefficients.*

$$\alpha_3^{(2)} = 1, \qquad \beta_3^{(2)} = \beta^*,$$
$$\alpha_2^{(2)} = \tilde{\alpha}_2^{(2)} + s, \qquad \beta_2^{(2)} = \tilde{\beta}_2^{(2)} + s\beta^*,$$
$$\alpha_1^{(2)} = s\tilde{\alpha}_2^{(1)}, \qquad \beta_1^{(2)} = \tilde{\beta}_1^{(2)} + s\tilde{\beta}_2^{(1)},$$
$$\alpha_0^{(2)} = s\tilde{\alpha}_1^{(1)}, \qquad \beta_0^{(2)} = \tilde{\beta}_0^{(2)} + s\tilde{\beta}_1^{(1)}.$$

Here the $\tilde{\alpha}_i^{(j)}$ and the $\tilde{\beta}_i^{(j)}$ are constants and s is a free parameter. Note that the free parameter affects only the coefficients of the second stage. Now, the first stage is

$$y_{2n+3}^* + \tilde{\alpha}_2^{(1)} y_{2n+2}^* + \tilde{\alpha}_1^{(1)} y_{2n+1}^* = h(\beta^* f_{2n+3}^* + \tilde{\beta}_2^{(1)} f_{2n+2}^* + \tilde{\beta}_1^{(1)} f_{2n+1}^*),$$

while the second stage is

$$y_{2n+4}^* + \alpha_2^{(2)} y_{2n+3}^* + \alpha_1^{(2)} y_{2n+2}^* + \alpha_0^{(2)} y_{2n+1}^*$$
$$= h(\beta_3^{(2)} f_{2n+4}^* + \beta_2^{(2)} f_{2n+3}^* + \beta_1^{(2)} f_{2n+2}^* + \beta_0^{(2)} f_{2n+1}^*)$$

or

$$y_{2n+4}^* + \tilde{\alpha}_2^{(2)} y_{2n+3}^*$$
$$= h(\beta^* f_{2n+4}^* + \tilde{\beta}_2^{(2)} f_{2n+3} + \tilde{\beta}_1^{(2)} f_{2n+2} + \tilde{\beta}_0^{(2)} f_{2n+1}^*)$$
$$- s[y_{2n+3}^* + \tilde{\alpha}_2^{(1)} y_{2n+2}^* + \tilde{\alpha}_1^{(1)} y_{2n+1}^*$$
$$- h(\beta^* f_{2n+3}^* + \tilde{\beta}_2^{(1)} f_{2n+2}^* + \tilde{\beta}_1^{(1)} f_{2n+1}^*)].$$

But if the first stage is solved exactly then the term in square brackets is zero and the solution, y_{2n+4}^*, calculated from the second stage is invariant to the free parameter, s. Similarly, although the parameter s affects the error constant, $C_{p+1}^{(2)}$, of the second stage, the term $C_{p+1}^{(2)} - \alpha_{k-1}^{(2)} C_{p+1}^{(1)}$ is invariant to s so that the vector

$$\hat{t} = (0, 0, \cdots, 0, C_{p+1}^{(1)}, C_{p+1}^{(2)} - \alpha_{k-1}^{(2)} C_{p+1}^{(1)})^T$$

appearing in the expression for the local error does not depend on s. Of course, the characteristic polynomial is also invariant to s.

The higher order cases can be handled similarly and for orders $p = 2, 3, \cdots, 8$, the coefficients of p-step/$p+1$-step two stage cyclic multistep formulae were found by solving sets of nonlinear equations such as (7). The results are listed in Tables 2–8 using the same parameterization as was used in the second order case. All computations were done on a Burroughs 6700 in DOUBLE PRECISION. The nonlinear equation solver that was used was the IMSL routine ZSYSTM which is an implementation of Brown's algorithm. The coefficients in Tables 2–8 are listed to 16 significant figures.

The formulae listed in Tables 2–8 have the same characteristic polynomial as the kth order second derivative formulae:

$$y_{n+k} = y_{n+k-1} + h \sum_{j=0}^{k} \bar{\beta}_j y'_{n+j} - h^2 \left(\frac{\bar{\beta}_k}{2}\right)^2 y''_{n+k}, \tag{8}$$

with the $\bar{\beta}_k$ listed in Table 1. As discussed in § 3, Enright [4] derived a different set of formulae of form (8) for which $\bar{\beta}_k$ was chosen to raise the order of the formula to $k+1$ rather than to maximize the stability angle. It is interesting to note that it is possible to derive k-step/$(k+1)$-step cyclic formulae of order $k+1$ with the same characteristic polynomial as Enright's formulae if condition (7m) is replaced by the condition (ii) of the lemma (the condition for higher order convergence). In particular, there is a 3-step/4-step method whose constituent formulae are 3rd order, but which is globally convergent to 4th order and has a stability angle of 89.99°. The coefficients of this formula are listed in Table 9.

TABLE 2

2-step/3-step, 2nd order cyclic formulae.

$\alpha_1^{(1)} = 0.2713503499466539$	$\beta_1^{(1)} = -0.06067517497332695$
$\alpha_2^{(1)} = -1.271350349946654$	$\beta_2^{(1)} = 0.2143248250266731$
$\alpha_3^{(1)} = 1$	$\beta_3^{(1)} = 0.575$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = -0.04894502624599904 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = 0.1728900524919981 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = -1 + s$	$\beta_2^{(2)} = 0.3010549737540010 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = 1$	$\beta_3^{(2)} = 0.575$
$C_{p+1}^{(1)} = -0.1357208041711122$	$C_{p+1}^{(2)} = -0.1093883070873343 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_2^{(2)} C_{p+1}^{(1)} = -0.2451091112584465$	
stability angle = 90°	

TABLE 3

3-step/4-step, 3rd order cyclic formulae.

$\alpha_1^{(1)} = -0.7156809323463300$	$\beta_1^{(1)} = -0.8629888978651531$
$\alpha_2^{(1)} = 2.421409496232843$	$\beta_2^{(1)} = 3.166634912306544$
$\alpha_3^{(1)} = -2.705728563886513$	$\beta_3^{(1)} = -4.013693645981575$
$\alpha_4^{(1)} = 1$	$\beta_4^{(1)} = 1.72$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = 1.894838801355351 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = -6.952885160409694 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = 0.4195749238763092 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = 9.231466454815180 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = -1.419574923876309 + s$	$\beta_3^{(2)} = -5.312995019637147 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = 1$	$\beta_4^{(2)} = 1.72$
$C_{p+1}^{(1)} = -1.244107937656965$	$C_{p+1}^{(2)} = 0.5673210898501973 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_3^{(2)} C_{p+1}^{(1)} = -1.198783341043101$	
stability angle = 90°	

TABLE 4
4-step/5-step, 4th order cyclic formulae.

$\alpha_1^{(1)} = 0.6103461100271559$	$\beta_1^{(1)} = 0.9215366381763085$
$\alpha_2^{(1)} = -2.747685823192436$	$\beta_2^{(1)} = -4.136159579898865$
$\alpha_3^{(1)} = 4.670005693524193$	$\beta_3^{(1)} = 7.141520549769378$
$\alpha_4^{(1)} = -3.532665980358913$	$\beta_4^{(1)} = -5.652569985267611$
$\alpha_5^{(1)} = 1.0$	$\beta_5^{(1)} = 1.72$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = -2.649958179621567 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = 11.89388403771322 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = -0.2716147139534076 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = -20.89280076921304 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = 1.222768338553090 + s\alpha_4^{(1)}$	$\beta_3^{(2)} = 17.85562740240295 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = -1.951153624599682 + s$	$\beta_4^{(2)} = -7.606291401927840 + s\beta_5^{(1)}$
$\alpha_5^{(2)} = 1$	$\beta_5^{(2)} = 1.72$
$C_{p+1}^{(1)} = -1.255794490885898$	$C_{p+2}^{(1)} = 1.307818847289458 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_4^{(2)} C_{p+1}^{(1)} = -1.142429125354875$	
stability angle = 90°	

TABLE 5
5-step/6-step, 5th order cyclic formulae.

$\alpha_1^{(1)} = -0.5473524843400159$	$\beta_1^{(1)} = -0.8285692507199495$
$\alpha_2^{(1)} = 3.046981165805116$	$\beta_2^{(1)} = 4.523688780360865$
$\alpha_3^{(1)} = -6.842547445085438$	$\beta_3^{(1)} = -10.05367226203171$
$\alpha_4^{(1)} = 7.741845845969376$	$\beta_4^{(1)} = 11.42213178164391$
$\alpha_5^{(1)} = -4.398927082349038$	$\beta_5^{(1)} = -6.661863565106905$
$\alpha_6^{(1)} = 1$	$\beta_6^{(1)} = 1.59$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = 2.569479392269974 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = -14.02842923277731 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = 0.2023989966807180 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = 31.45494082222007 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = -1.126707101014822 + s\alpha_4^{(1)}$	$\beta_3^{(2)} = -36.93603791002108 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = 2.473981498420040 + s\alpha_5^{(1)}$	$\beta_4^{(2)} = 23.96920129089773 + s\beta_5^{(1)}$
$\alpha_5^{(2)} = -2.549673394085936 + s$	$\beta_5^{(2)} = -8.446918649021926 + s\beta_6^{(1)}$
$\alpha_6^{(2)} = 1$	$\beta_6^{(2)} = 1.59$
$C_{p+1}^{(1)} = -1.133876687363511$	$C_{p+1}^{(2)} = 1.347003669875171 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_5^{(2)} C_{p+1}^{(1)} = -1.544011552069870$	
stability angle = 86.64°	

TABLE 6

6-step/7-step, 6th order cyclic formulae.

$\alpha_1^{(1)} = 0.5065949241573893$	$\beta_1^{(1)} = 0.6632827080818050$
$\alpha_2^{(1)} = -3.347588901033099$	$\beta_2^{(1)} = -4.318145132370032$
$\alpha_3^{(1)} = 9.288264853658161$	$\beta_3^{(1)} = 11.87138330994840$
$\alpha_4^{(1)} = -13.85423667736953$	$\beta_4^{(1)} = -17.69684560038281$
$\alpha_5^{(1)} = 11.70833137691048$	$\beta_5^{(1)} = 15.14109874111104$
$\alpha_6^{(1)} = -5.301365576323402$	$\beta_6^{(1)} = -7.068045626188859$
$\alpha_7^{(1)} = 1$	$\beta_7^{(1)} = 1.4$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = -1.966557927018182 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = 12.80281007873031 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = -0.1622565792899431 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = -35.39751311755192 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = 1.072194564235031 + s\alpha_4^{(1)}$	$\beta_3^{(2)} = 53.77255375657074 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = -2.943247343299985 + s\alpha_5^{(1)}$	$\beta_4^{(2)} = -48.44295713982840 + s\beta_5^{(1)}$
$\alpha_5^{(2)} = 4.228025197181846 + s\alpha_6^{(1)}$	$\beta_5^{(2)} = 26.08926183188078 + s\beta_6^{(1)}$
$\alpha_6^{(2)} = -3.194715838826949 + s$	$\beta_6^{(2)} = -8.166685368910522 + s\beta_7^{(1)}$
$\alpha_7^{(2)} = 1$	$\beta_7^{(2)} = 1.4$
$C_{p+1}^{(1)} = -0.9495222263711510$	$C_{p+1}^{(2)} = 0.9261782360977777 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_6^{(2)} C_{p+1}^{(1)} = -2.107275459808366$	
stability angle = 76.32°	

TABLE 7

7-step/8-step, 7th order cyclic formulae.

$\alpha_1^{(1)} = -0.4730377247001383$	$\beta_1^{(1)} = -0.5593384312526297$
$\alpha_2^{(1)} = 3.614094099926330$	$\beta_2^{(1)} = 4.220709442337714$
$\alpha_3^{(1)} = -11.91361867598596$	$\beta_3^{(1)} = -13.79410919873106$
$\alpha_4^{(1)} = 21.97700424011387$	$\beta_4^{(1)} = 25.36722912840438$
$\alpha_5^{(1)} = -24.50710155920918$	$\beta_5^{(1)} = -28.42514847816480$
$\alpha_6^{(1)} = 16.51027888261980$	$\beta_6^{(1)} = 19.46252654187500$
$\alpha_7^{(1)} = -6.207619262764723$	$\beta_7^{(1)} = -7.551726936498641$
$\alpha_8^{(1)} = 1$	$\beta_8^{(1)} = 1.275$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = 1.627202025358370 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = -12.27869670539260 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = 0.1354428064536342 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = 40.28383410923075 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = -1.034807631868773 + s\alpha_4^{(1)}$	$\beta_3^{(2)} = -74.96406486441271 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = 3.390826065383960 + s\alpha_5^{(1)}$	$\beta_4^{(2)} = 86.48642725198855 + s\beta_5^{(1)}$
$\alpha_5^{(2)} = -6.137117459840426 + s\alpha_6^{(1)}$	$\beta_5^{(2)} = -63.47949128834042 + s\beta_6^{(1)}$
$\alpha_6^{(2)} = 6.516126407008639 + s\alpha_7^{(1)}$	$\beta_6^{(2)} = 29.33643618328776 + s\beta_7^{(1)}$
$\alpha_7^{(2)} = -3.870470187137034 + s$	$\beta_7^{(2)} = -8.238999899992440 + s\beta_8^{(1)}$
$\alpha_8^{(2)} = 1$	$\beta_8^{(2)} = 1.275$
$C_{p+1}^{(1)} = -0.8295861815319158$	$C_{p+1}^{(2)} = 0.7053824986060715 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_7^{(2)} C_{p+1}^{(1)} = -2.505506084674061$	
stability angle = 57.66°	

TABLE 8

8-step/9-step, 8th order cyclic formulae.

$\alpha_1^{(1)} = 0.4453348513468461$	$\beta_1^{(1)} = 0.4822252401396465$
$\alpha_2^{(1)} = -3.859180665772141$	$\beta_2^{(1)} = -4.136942315524837$
$\alpha_3^{(1)} = 14.71642002045547$	$\beta_3^{(1)} = 15.65983206216709$
$\alpha_4^{(1)} = -32.27324489323404$	$\beta_4^{(1)} = -34.21855612352479$
$\alpha_5^{(1)} = 44.53945382773937$	$\beta_5^{(1)} = 47.30113434219221$
$\alpha_6^{(1)} = -39.61766259292482$	$\beta_6^{(1)} = -42.45139314249457$
$\alpha_7^{(1)} = 22.16853315999479$	$\beta_7^{(1)} = 24.20929805095255$
$\alpha_8^{(1)} = -7.119653707605472$	$\beta_8^{(1)} = -8.028548065406828$
$\alpha_9^{(1)} = 1$	$\beta_9^{(1)} = 1.18$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = -1.384350259109908 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = 11.87614560523851 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = -0.1162654110659919 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = -45.07917311093609 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = 1.007532253824141 + s\alpha_4^{(1)}$	$\beta_3^{(2)} = 99.29376589604582 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = -3.827901007455769 + s\alpha_5^{(1)}$	$\beta_4^{(2)} = -139.7912993045485 + s\beta_5^{(1)}$
$\alpha_5^{(2)} = 8.302866483037190 + s\alpha_6^{(1)}$	$\beta_5^{(2)} = 130.5230230565463 + s\beta_6^{(1)}$
$\alpha_6^{(2)} = -11.16589138289488 + s\alpha_7^{(1)}$	$\beta_6^{(2)} = -81.18525412979233 + s\beta_7^{(1)}$
$\alpha_7^{(2)} = 9.369883714651345 + s\alpha_8^{(1)}$	$\beta_7^{(2)} = 33.00040235056584 + s\beta_8^{(1)}$
$\alpha_8^{(2)} = -4.570224650096031 + s$	$\beta_8^{(2)} = -8.408425274884485 + s\beta_9^{(1)}$
$\alpha_9^{(2)} = 1$	$\beta_9^{(2)} = 1.18$
$C_{p+1}^{(1)} = -0.7390717521435601$	$C_{p+1}^{(2)} = 0.5521149771269926 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_8^{(2)} C_{p+1}^{(1)} = -2.825608962709170$	
stability angle = 22.15°	

TABLE 9

3-step/4-step, 4th order cyclic formulae.

$\alpha_1^{(1)} = -0.7174260738375705$	$\beta_1^{(1)} = -0.8427538469096104$
$\alpha_2^{(1)} = 2.426199286279159$	$\beta_2^{(1)} = 3.108006010400179$
$\alpha_3^{(1)} = -2.708773212441589$	$\beta_3^{(1)} = -3.957345114139985$
$\alpha_4^{(1)} = 1$	$\beta_4^{(1)} = 1.700745812045397$
$\alpha_0^{(2)} = s\alpha_1^{(1)}$	$\beta_0^{(2)} = 1.815760117302048 + s\beta_1^{(1)}$
$\alpha_1^{(2)} = s\alpha_2^{(1)}$	$\beta_1^{(2)} = -6.696372112348223 + s\beta_2^{(1)}$
$\alpha_2^{(2)} = 0.4198486192398069 + s\alpha_3^{(1)}$	$\beta_2^{(2)} = 8.936285375215797 + s\beta_3^{(1)}$
$\alpha_3^{(2)} = -1.419848619239807 + s$	$\beta_3^{(2)} = -5.176267811454826 + s\beta_4^{(1)}$
$\alpha_4^{(2)} = 1$	$\beta_4^{(2)} = 1.700745812045397$
$C_{p+1}^{(1)} = -1.224654175117099$	$C_{p+1}^{(2)} = 0.5075079977249758 + sC_{p+1}^{(1)}$
$C_{p+1}^{(2)} - \alpha_3^{(2)} C_{p+1}^{(1)} = -1.231315541861302$	
stability angle = 89.99°	

There are other approaches to deriving 2-stage formulae with the same characteristic polynomials as the second derivative formulae. A different approach to the one we have presented is based on cyclic methods whose constituent formulae are each k-step multistep formulae of order $k-1$. Condition (ii) of the lemma is used to ensure that the overall method is kth order.

In the approach we have taken, the stepnumbers of the constituent multistep formulae increase within the cycle. We may consider the general case of an M-stage cyclic method for which the ith stage is a $(k+i-1)$-step formula. Note that the cycle as a whole still requires only k past values, but has more free parameters than would be the case if each constituent formula were k-step. These free parameters may be used to improve the properties of the cyclic method. For example, it is conjectured that if each constituent formula is of order $2k-1$ then the extra free parameters may be used to find M-stage cyclic methods of order $2k$ with better stability properties than the Donelson and Hansen formulae.

5. Numerical testing. To confirm numerically the orders of all the methods derived, we applied our formulae to the simple problem

$$y' = -2y \qquad y(0) = 1.$$

The relative errors $|y(x) - y^*(x)/y(x)|$ at $x = 5$ are listed in Table 10 for various values of h. They indicate that the methods have the requisite orders.

TABLE 10
Test results for the problem $y' = -2y$.

Formula	Order	Range of integration	$h = 0.5$	$h = 0.05$	$h = 0.1$	$h = 0.01$
2/3-step	2	0.–5.	$.64\times10^{0}$	$.14\times10^{-1}$	$.54\times10^{-1}$	$.55\times10^{-3}$
3/4-step	3	0.–5.	$.33\times10^{3}$	$.18\times10^{-2}$	$.37\times10^{-1}$	$.28\times10^{-5}$
4/5-step	4	0.–5.	$.87\times10^{3}$	$.27\times10^{-2}$	$.45\times10^{-1}$	$.43\times10^{-5}$
5/6-step	5	0.–5.	$.50\times10^{4}$	$.69\times10^{-3}$	$.27\times10^{-1}$	$.19\times10^{-6}$
6/7-step	6	0.–5.	$.89\times10^{4}$	$.99\times10^{-4}$	$.87\times10^{-2}$	$.50\times10^{-8}$
7/8-step	7	0.–5.	$.17\times10^{5}$	$.16\times10^{-4}$	$.31\times10^{-2}$	$.15\times10^{-9}$
8/9-step	8	0.–5.	$.60\times10^{5}$	$.26\times10^{-5}$	$.12\times10^{-2}$	$.50\times10^{-11}$
3/4-step	4	0.–5.	$.35\times10^{3}$	$.26\times10^{-2}$	$.44\times10^{-1}$	$.41\times10^{-5}$

To confirm the $A(0)$ stability of our formulae, we applied our formulae to the problem

$$y' = -10^{12}(y-x)+1, \quad y(0)=1, \quad y(x) = e^{-10^{12}x} + x.$$

The errors $|y(x) - y^*(x)|$ are listed in Table 11. All numerical tests were carried out in DOUBLE PRECISION on a VAX 11/780.

TABLE 11
Test results for the problem $y' = -10^{12}(y-x)+1$.

Formula	Order	Range of integration	$h = 0.5$	$h = 0.05$	$h = 0.1$	$h = 0.01$
2/3-step	2	2.−7.	0.	0.	$.11\times10^{-15}$	$.11\times10^{-15}$
3/4-step	3	2.−7.	0.	$.78\times10^{-15}$	$.22\times10^{-15}$	$.67\times10^{-15}$
4/5-step	4	3.−8.	0.	$.44\times10^{-15}$	$.56\times10^{-15}$	$.22\times10^{-15}$
5/6-step	5	3.−8.	0.	$.18\times10^{-14}$	$.33\times10^{-15}$	$.36\times10^{-14}$
6/7-step	6	3.−8.	$.64\times10^{-14}$	$.89\times10^{-15}$	$.43\times10^{-14}$	$.56\times10^{-14}$
7/8-step	7	3.5.−8.5	$.18\times10^{-14}$	$.18\times10^{-13}$	$.69\times10^{-14}$	$.12\times10^{-14}$
8/9-step	8	4.−9.	0.	$.89\times10^{-14}$	$.25\times10^{-13}$	$.33\times10^{-14}$
3/4-step	4	2.−7.	$.33\times10^{-15}$	$.11\times10^{-15}$	$.11\times10^{-15}$	$.22\times10^{-15}$

REFERENCES

[1] P. ALBRECHT, *On the order of composite multistep methods for ordinary differential equations*, Numer. Math., 29 (1978), pp. 381–396.

[2] G. DAHLQUIST, *Convergence and stability in the numerical integration of ordinary differential equations*, Math. Scand., 4 (1956), pp. 33–53.

[3] J. DONELSON AND E. HANSEN, *Cyclic composite multistep predictor-corrector methods*, SIAM J. Numer. Anal., 8 (1971), pp. 137–157.

[4] W. ENRIGHT, *Optimal second derivative methods for stiff systems* in Stiff Differential Systems, R. A. Willoughby, ed., Plenum, New York, 1974, pp. 95–111.

[5] K. R. JACKSON AND R. SACKS-DAVIS, *An alternative implementation of variable stepsize multistep formulae for stiff* ODE's, ACM Trans. Math. Software, 6 (1980), pp. 295–318.

[6] J. M. TENDLER, T. A. BICKART AND Z. PICEL, *A stiffly stable integration process using cyclic composite methods*, ACM Trans. Math. Software, 4 (1978), pp. 339–368.

[7] F. SHAMPINE, *Implementation of implicit formulas for the solution of* ODE's, this Journal, 1 (1980), pp. 103–118.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0014 $01.25/0

AN ALGORITHM FOR THE SINGLE FACILITY LOCATION PROBLEM USING THE JACCARD METRIC*

G. A. WATSON†

Abstract. The solution of the single facility location problem using the Jaccard metric is considered. Necessary conditions for a solution are derived, and a finite descent algorithm is developed and illustrated numerically.

Key words. single facility location problem, Jaccard metric

1. Introduction. A common type of data provided for statistical analysis is of binary form, where the presence or absence of a number of characteristics in a set of objects is recorded. For such data, it is appropriate to define a similarity coefficient and Jaccard's coefficient is widely used [3], [4], [5]: for example, as claimed in [2], it is used extensively by ecologists, in the comparison of plant communities in terms of the species present. If x and y are two binary vectors in R^n, then the Jaccard coefficient $s(x, y)$ is defined by

$$s(x, y) = \frac{\sum_{j=1}^{n} \min(x_j, y_j)}{\sum_{j=1}^{n} \max(x_j, y_j)} \qquad (x \neq 0, y \neq 0) \tag{1.1}$$

with $s(0, 0) = 1$. Clearly $0 \leqq s(x, y) \leqq 1$. Further,

$$d(x, y) = 1 - s(x, y)$$

defines a *dissimilarity* coefficient, which may be shown to be a metric on the set of binary vectors [5]. This so-called Jaccard (or sometimes Tanimoto) metric turns out to be of use in the more general situation when we merely have x and y nonnegative vectors in R^n. According to Späth [6], the Jaccard metric "has been successfully used for ordinal and other nonnegative data by the author of this article and other authors": for example, an application is given in [6] to cluster analysis.

The underlying problem is the following single location problem:

$$\begin{aligned} &\text{given } a_i \in R^n,\ a_i \geqq 0,\ i = 1, 2, \cdots, m, \\ &\text{find } z \in R^n,\ z \geqq 0, \text{ to minimize } \textstyle\sum_{i=1}^{m} d(z, a_i). \end{aligned} \tag{1.2}$$

Since, for all $b, c \in R$,

$$\min(b, c) = (b + c - |b - c|)/2,$$

$$\max(b, c) = (b + c + |b - c|)/2,$$

it is easy to see that if $z \geqq 0$, $a_i \geqq 0$, $i = 1, 2, \cdots, m$, then

$$d(z, a_i) = 2h(z, a_i),$$

where

$$h(z, a_i) = \frac{\|z - a_i\|}{\|z\| + \|a_i\| + \|z - a_i\|}, \qquad i = 1, 2, \cdots, m, \tag{1.3}$$

and the norm is the L_1 norm on R^n. Thus we may consider instead of (1.2) the

* Received by the editors April 30, 1982, and in revised form October 14, 1982.
† Department of Mathematical Sciences, University of Dundee, Dundee DD1 4HN, Scotland.

equivalent problem

$$\text{given } a_i \in R^n,\ a_i \geqq 0,\ i = 1, 2, \cdots, m$$
$$\text{find } z \in R^n,\ z \geqq 0 \text{ to minimize } f(z) = \sum_{i=1}^{m} h(z, a_i). \tag{1.4}$$

In view of the apparent usefulness of (1.4) in practical situations, it would be of value to have available a systematic method for its solution, and it is the purpose of this paper to develop an algorithm which may be used to solve the problem. The method is finite, and may be interpreted as a "vertex to vertex" descent method, similar in some senses to that of [1], which terminates at a point satisfying first order necessary conditions for a solution to (1.4): since $f(z)$ is not a convex function of z, we will in fact merely expect to obtain a local minimum. In the next section, we give appropriate necessary conditions; in § 3 we develop the proposed algorithm, and illustrate by applying it to some examples.

It will be assumed in what follows that $\|a_i\| \neq 0$, $i = 1, 2, \cdots, m$.

2. Necessary conditions. A crucial role is played by the one-sided directional derivative of $f(z)$ at a point z, in the direction s, given by

$$f'(z; s) = \lim_{\gamma \to 0+} \frac{f(z + \gamma s) - f(z)}{\gamma}.$$

This exists for all z, but we will only be concerned with z satisfying $z \geqq 0$, and begin by deriving an expression for the directional derivative in this case. Some notation is required, and we will denote by Z_i the set

$$Z_i = \{j\colon z_j = (a_i)_j\}, \qquad i = 1, 2, \cdots, m,$$

where $(a_i)_j$ denotes the jth component of a_i. It is also convenient to define the set

$$Z_0 = \{j\colon z_j = 0\},$$

and numbers $(\theta_i)_j$ such that

$$(\theta_i)_j = \operatorname{sign}(z_j - (a_i)_j), \qquad j \notin Z_i.$$

Because of the frequent occurrence of the following expressions, we will subsequently use the abbreviated notation

$$D_i(z) = \|z\| + \|a_i\| + \|z - a_i\|, \qquad i = 1, 2, \cdots, m.$$

THEOREM 1. *Let $z \in R^n$ satisfying $z \geqq 0$, and $s \in R^n$ be given. Then*

$$f'(z; s) = \sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|) v_i^T s - \|z - a_i\| v_0^T s}{D_i(z)^2} \tag{2.1}$$

where, for each i, $i = 1, 2, \cdots, m$,

$$(v_i)_j = (\theta_i)_j,\ j \notin Z_i,$$
$$(v_i)_j = \operatorname{sign}(s_j),\ j \in Z_i,$$
$$(v_0)_j = 1,\ j \notin Z_0,$$
$$(v_0)_j = \operatorname{sign}(s_j),\ j \in Z_0.$$

Proof. For any $a \in R^n$, $a \geqq 0$, let

$$Z = \{j\colon z_j = a_j\}, \qquad \theta_j = \operatorname{sign}(z_j - a_j), \quad j \notin Z.$$

Then for $\gamma > 0$ sufficiently small

$$\|z + \gamma s\| = \sum_{j \notin Z_0} (z_j + \gamma s_j) + \sum_{j \in Z_0} \gamma |s_j|, \tag{2.2}$$

and also

$$\|z + \gamma s - a\| = \sum_{j \notin Z} \theta_j (z_j + \gamma s_j - a_j) + \sum_{j \in Z} \gamma |s_j| = \|z - a\| + \gamma A, \tag{2.3}$$

where $A = \sum_{j \notin Z} \theta_j s_j + \sum_{j \in Z} |s_j|$.

Let $B = \sum_{j \notin Z_0} s_j + \sum_{j \in Z_0} |s_j| + A$. Then for $\gamma > 0$ small enough that (2.2) and (2.3) hold,

$$\begin{aligned} h(z + \gamma s, a) &= \frac{\|z - a\| + \gamma A}{\|z\| + \|a\| + \|z - a\| + \gamma B}, \\ &= h(z, a) + \gamma \frac{A(\|z\| + \|a\| + \|z - a\|) - B(\|z - a\|)}{(\|z\| + \|a\| + \|z - a\|)^2} \\ &\quad + \gamma^2 \frac{B(B\|z - a\| - A(\|z\| + \|a\| + \|z - a\|))}{(\|z\| + \|a\| + \|z - a\|)^3} + O(\gamma^3) \\ &= h(z, a) + \gamma \frac{A(\|z\| + \|a\|) - \|z - a\| v_0^T s}{(\|z\| + \|a\| + \|z - a\|)^2} + O(\gamma^2). \end{aligned} \tag{2.4}$$

Setting $a = a_i$, summing over i, dividing by γ and letting $\gamma \to 0+$, the relation (2.1) is obtained. □

This result may be conveniently interpreted in terms of subgradients. Using the standard notation ∂ for subgradient, it is well known that for any $s \in R^n$

$$\partial \|s\| = \{t \in R^n : |t_j| \leqq 1, j = 1, 2, \cdots, n, t_j = \text{sign}(s_j), s_j \neq 0\}.$$

Then clearly

$$v_i \in \partial \|z - a_i\|, \quad i = 1, 2, \cdots, m, \qquad v_0 \in \partial \|z\|.$$

Further, for each i, $w_i^T s \leqq v_i^T s$ for all $w_i \in \partial \|z - a_i\|$, so that

$$v_i^T s = \max_{w_i \in \partial \|z - a_i\|} w_i^T s, \qquad i = 1, 2, \cdots, m.$$

Similarly

$$v_0^T s = \max_{w_0 \in \partial \|z\|} w_0^T s.$$

We have the following first order necessary condition for a solution to (1.4).

THEOREM 2. *Let $z \geqq 0$ solve (1.4). Then there exist $w_i \in \partial \|z - a_i\|$, $i = 1, 2, \cdots, m$ such that*

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)(w_i)_j - \|z - a_i\|}{D_i(z)^2} \begin{Bmatrix} = \\ \geqq \end{Bmatrix} 0, \quad \begin{cases} j \notin Z_0, \\ j \in Z_0. \end{cases} \tag{2.5}$$

Proof. Let $z \geqq 0$ solve (1.4). Then we must have

$$f'(z; s) \geqq 0$$

for all $s \in R^n$ with $s_j \geqq 0$, $j \in Z_0$. For such an s,

$$\max_{w_0 \in \partial \|z\|} w_0^T s = e^T s$$

where $e = (1, 1, \cdots, 1)^T$. Thus, by Theorem 1,

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)v_i^T s - \|z - a_i\| e^T s}{D_i(z)^2} \geqq 0, \tag{2.6}$$

where for $i = 1, 2, \cdots, m$,

$$(v_i)_j = (\theta_i)_j, \quad j \notin Z_i, \qquad (v_i)_j = \text{sign}\,(s_j), \quad j \in Z_i,$$

for all $s \in R^n$ with $s_j \geqq 0$, $j \in Z_0$. Now assume (with no loss of generality) that $Z_0 = \{r+1, \cdots, n\}$, and that any $w_i \in \partial\|z - a_i\|$ is partitioned so that $w_i = \begin{bmatrix} w_i^1 \\ w_i^2 \end{bmatrix}$ with $w_i^1 \in R^r$ and $w_i^2 \in R^{n-r}$. We first show that there exist vectors w_i^1, with $w_i \in \partial\|z - a_i\|$, $i = 1, 2, \cdots, m$, such that the equalities hold in (2.5). Suppose not. Then by a standard separation result in R^r (see, for example [7, p. 13]), there exists $t \in R^r$ such that

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)w_i^{1T} t - \|z - a_i\| e^T t}{D_i(z)^2} < 0 \tag{2.7}$$

for all $w_i^1 : w_i \in \partial\|z - a_i\|$, $i = 1, 2, \cdots, m$. Choosing $s \in R^n$ with $s_j = t_j$, $j = 1, 2, \cdots, r$, $s_j = 0$, $j = r+1, \cdots, n$, contradicts (2.6).

Finally $s = e_j$, $j = r+1, \cdots, n$ (the jth co-ordinate vector) satisfies (2.6) with $(v_i)_j = (\theta_i)_j, j \notin Z_i$, $(v_i)_j = 1, j \in Z_i$. Forming vectors w_i^2 from these values, and combining them with the w_i^1 whose existence was previously established, we obtain vectors w_i, $i = 1, 2, \cdots, m$ satisfying (2.5) and the proof is complete. □

COROLLARY. *Let $z > 0$ solve (1.4). Then there exist $w_i \in \partial\|z - a_i\|$, $i = 1, 2, \cdots, m$ such that*

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)w_i - \|z - a_i\| e}{D_i(z)^2} = 0.$$

In seeking a point $z \geqq 0$ satisfying the above conditions (2.5) it would appear that we must search over the continuum of possible values. However, second derivative information may be used to strengthen Theorem 2, and to eliminate all but a finite number of points from consideration. The following result is due to Späth [6]; we give an alternative proof which uses the expansion (2.4).

THEOREM 3. *Let $z \geqq 0$ solve (1.4). Then there exist n indices $\tau_1, \tau_2, \cdots, \tau_n \in \{1, 2, \cdots, m\}$ such that*

$$z_j = (a_{\tau_j})_j, \qquad j = 1, 2, \cdots, n.$$

Proof. Let $z \geqq 0$ solve (1.4), but there exist k, $1 \leqq k \leqq n$, such that $k \notin Z_i$, $i = 1, 2, \cdots, m$. If $z_k = 0$, then $(\theta_i)_k = -1$, $i = 1, 2, \cdots, m$, and it is easily verified that $f'(z; e_k) < 0$, a contradiction, so we may assume $z_k \neq 0$. Now, using (2.4), we have (for $\gamma > 0$ small enough)

$$f(z + \gamma s) = f(z) + \gamma f'(z; s) + \frac{\gamma^2}{2} f''(z; s) + O(\gamma^3), \tag{2.8}$$

say, where

$$\frac{1}{2} f''(z; s) = \sum_{i=1}^{m} \frac{(v_i^T s + v_0^T s)(v_0^T s\|z - a_i\| - v_i^T s(\|z\| + \|a_i\|))}{D_i(z)^3}.$$

Thus, letting $\sigma = \pm 1$, we have

$$\frac{1}{2}f''(z;\sigma e_k) = \sum_{i=1}^{m} \frac{(\sigma(\theta_i)_k + \sigma)(\sigma\|z-a_i\| - \sigma(\theta_i)_k(\|z\|+\|a_i\|))}{D_i(z)^3}$$

$$= \sum_{i:\,(\theta_i)_k=1} \frac{2(\|z-a_i\|-\|z\|-\|a_i\|)}{D_i(z)^3} \leqq 0,$$

with equality only if $(\theta_i)_k = -1$, $i = 1, 2, \cdots, m$, which leads to a contradiction as before. It follows that

$$f''(z;\sigma e_k) < 0,$$

and since $f'(z;\sigma e_k) = 0$, we have a further contradiction which concludes the proof. □

This result shows that we need only search over such "vertices" to find a point z such that (2.5) is satisfied. Thus one possible approach is simply to evaluate $f(z)$ at each of these m^n points. This is, however, inefficient in general, and in the next section we present an algorithm which consists of a systematic descent from vertex to vertex until a point satisfying (2.5) is reached.

3. A descent algorithm. Let $z^{(l)} \in R^n$ be such that

$$z_j^{(l)} = (a_{\tau_j})_j, \qquad j = 1, 2, \cdots, n,$$

for some set of n indices $\tau_1, \tau_2, \cdots, \tau_n \in \{1, 2, \cdots, m\}$. Unless $z^{(l)}$ satisfies (2.5), we will exchange one of these indices for another outside the set so as to achieve a decrease in the value of $f(z)$. Thus we consider a step from $z^{(l)}$ to a new point $z^{(l+1)} = z^{(l)} + \gamma^{(l)} s^{(l)}$, where $s^{(l)} = +e_k$ or $-e_k$, for some $k \in \{1, 2, \cdots, n\}$, and where $\gamma^{(l)} > 0$ is chosen so that we reach the nearest new vertex in the direction $s^{(l)}$. It is sufficient for this to achieve a reduction in the objective function that $s^{(l)}$ be a descent direction for $f(z)$ at $z^{(l)}$: for $z = z^{(l)} + \gamma s^{(l)}$, $0 < \gamma < \gamma^{(l)}$, it follows by the argument used in the proof of Theorem 3 that if $f'(z^{(l)}; s^{(l)}) = 0$ then $f''(z^{(l)}; s^{(l)}) < 0$, and so $f(z)$ continues to decrease at least as far as $f(z^{(l+1)})$. We now show how a suitable direction $s^{(l)}$ may be obtained, and consider two separate cases. Dropping the superscript (l), let z be the current point, and define

$$W_j = \{i : z_j - (a_i)_j = 0\}, \qquad j = 1, 2, \cdots, n.$$

(a) *Nondegenerate case.* W_j is a singleton i_j, for every j, $j = 1, 2, \cdots, n$. Consider the system of equations

$$\sum_{i=1}^{m} \frac{(\|z\|+\|a_i\|)(\lambda_i)_j - \|z-a_i\|}{D_i(z)^2} = 0, \qquad j = 1, 2, \cdots, n \tag{3.1}$$

where $(\lambda_i)_j = (\theta_i)_j$, $i = 1, 2, \cdots, m$, $i \neq i_j$, $j = 1, 2, \cdots, n$. Then

$$(\lambda_{i_j})_j = \left[\sum_{i=1}^{m} \frac{\|z-a_i\|}{D_i(z)^2} - \sum_{\substack{i=1 \\ i\neq i_j}}^{m} \frac{(\|z\|+\|a_i\|)(\theta_i)_j}{D_i(z)^2}\right] * \left[\frac{D_{i_j}(z)^2}{\|z\|+\|a_{i_j}\|}\right], \qquad j = 1, 2, \cdots, n. \tag{3.2}$$

THEOREM 4. *For the nondegenerate case, let* $(\lambda_{i_j})_j$, $j = 1, 2, \cdots, n$ *be given by* (3.2), *and let* k *be such that*

$$|(\lambda_{i_k})_k| \geqq |(\lambda_{i_j})_j|, \qquad j = 1, 2, \cdots, n.$$

Then if $|(\lambda_{i_k})_k| \leqq 1$, z *satisfies* (2.5); *otherwise the direction* $s = \sigma e_k$, *with* $\sigma = \text{sign}\,((\lambda_{i_k})_k)$, *leads to another vertex with a lower function value.*

Proof. If $|(\lambda_{i_k})_k| \leqq 1$, then $\lambda_i \in \partial\|z - a_i\|$, $i = 1, 2, \cdots, m$, and (2.5) is satisfied, so assume not. Let $z_k = \min_i (a_i)_k$. Then $(\theta_i)_k = -1, i = 1, 2, \cdots, m, i \neq i_k$, and it is immediately obvious from (3.2) that $(\lambda_{i_k})_k > 0$, so that $\sigma = +1$. Let $z_k = \max_i (a_i)_k$. Then $(\theta_i)_k = +1$, $i = 1, 2, \cdots, m$, $i \neq i_k$ and so

$$(\lambda_{i_k})_k = \frac{\|z - a_{i_k}\|}{\|z\| + \|a_{i_k}\|} + \text{nonpositive terms.}$$

If $(\lambda_{i_k})_k > 1$, then $\|z - a_{i_k}\| > \|z\| + \|a_{i_k}\|$, a contradiction which shows that $\sigma = -1$. We have shown that we will step to a new vertex; it remains to show that $f(z)$ will be decreased, or equivalently that s is a descent direction. Now with v_i, $i = 1, 2, \cdots, m$ defined as in Theorem 1,

$$\begin{aligned} f'(z; s) &= \sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|) v_i^T s - \|z - a_i\| e^T s}{D_i(z)^2} \\ &= \sigma \sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)(v_i)_k - \|z - a_i\|}{D_i(z)^2} \\ &= \sigma \sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)((v_i)_k - (\lambda_i)_k)}{D_i(z)^2} \quad \text{using (3.1)} \\ &= \sigma \frac{(\|z\| + \|a_{i_k}\|)(\sigma - (\lambda_{i_k})_k)}{D_{i_k}(z)^2} \\ &< 0. \end{aligned} \tag{3.3}$$

(b) *Degenerate case.* W_j is not a singleton, for some j, $j = 1, 2, \cdots, n$. In this case, there are more than n unknowns in the system of equations (3.1), and so no unique solution. Of course if W_j is a singleton for *some* values of j, we may compute values of $(\lambda_{i_j})_j$ from (3.2), and if any of these values exceed one in modulus, we can obtain a descent direction as before, choosing k by finding the maximum over this subset of values. Thus we consider only the situation where such values of $(\lambda_{i_j})_j$ have modulus no greater than one, or where W_j is not a singleton, for any j.

In this case, perhaps the simplest way to proceed is to directly compute the directional derivative in the directions $+e_j$ and $-e_j$ (using (3.3)) for each j for which W_j is not a singleton. If $z_j = \min_i (a_i)_j$, $s = -e_j$ or $z_j = \max_i (a_i)_j$, $s = +e_j$, it is easily seen that (3.3) is nonnegative. If one of the remaining directions gives a negative value, then progress can be made to another vertex as before. Otherwise, it is readily shown that z satisfies the first order necessary conditions. For, let $s \in R^n$ be arbitrary except that $s_j \geqq 0, j \in Z_0$. We will assume, without loss of generality, that $s_j \geqq 0$, $j = 1, 2, \cdots, r$, $s_j < 0$, $j = r+1, \cdots, n$. Now the directions e_j, $j = 1, 2, \cdots, r$, $-e_j$, $j = r+1, \cdots, n$ are, by assumption, not descent directions, so that

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)(v_i)_j - \|z - a_i\|}{D_i(z)^2} \geqq 0, \qquad j = 1, 2, \cdots, r,$$

$$\sum_{i=1}^{m} \frac{(\|z\| + \|a_i\|)(-(v_i)_j) + \|z - a_i\|}{D_i(z)^2} \geqq 0, \qquad j = r+1, r+2, \cdots, n,$$

where, for each $i = 1, 2, \cdots, m$,

$$(v_i)_j = \begin{cases} (\theta_i)_j, & j \notin Z_i, \\ \text{sign } (s_j), & j \in Z_i. \end{cases}$$

Multiplying in turn by $s_j, j = 1, 2, \cdots, r, (-s_j), j = r+1, \cdots, n$, and summing gives (2.6).

THEOREM 5. *An algorithm based on the rules developed above will converge to a point satisfying* (2.5) *in a finite number of steps.*

Proof. We have defined a vertex to vertex descent process which can only terminate at a point satisfying (2.5). In addition, the number of vertices is finite. □

Considerable improvement in the efficiency of the algorithm, at least for larger problems, can be achieved by letting the step length exceed $\gamma^{(l)}$, if possible. Thus we test to see if $s^{(l)}$ continues to be a descent direction as we encounter new vertices, and run through such vertices until one is reached at which $s^{(l)}$ is no longer downhill. If an iteration is redefined in this way, then the total number can often be substantially reduced. For example, a problem with $m = 20$, $n = 10$ which required 75 iterations of the basic method was solved (from the same starting point) in 14 iterations by the modified algorithm.

The progress of the algorithm is illustrated by giving details of its performance on a simple example. In the implementation used, the direction of descent was the co-ordinate direction defined by the largest (in modulus) value of $(\lambda_{ij})_j$ defined by (3.2), if this value exceeded one, otherwise by the largest negative directional derivative in the remaining co-ordinate directions. Running through vertices was permitted, so that the maximum possible step was taken in each of the calculated directions $s^{(l)}$.

Let $m = 8$, $n = 4$ and the vectors a_i, $i = 1, 2, \cdots, 8$ be the rows of the matrix

$$\begin{bmatrix} 1 & 5 & 3 & 7 \\ 1 & 4 & 6 & 5 \\ 3 & 7 & 11 & 2 \\ 4 & 7 & 2 & 5 \\ 2 & 1 & 5 & 8 \\ 6 & 4 & 1 & 2 \\ 5 & 1 & 1 & 9 \\ 7 & 7 & 5 & 3 \end{bmatrix}$$

In Tables 1–3 we show the progress of the method for 3 different starting points. The integer l gives the iteration number, k is the index to be changed, $\lambda_{\max}$ is the value

TABLE 1

l	z	$f(z)$	k	$\lambda_{\max}$	der
0	1 5 11 9	2.045579	4	−4.842	—
1	1 5 11 7	1.977331	3	−4.013	—
2	1 5 5 7	1.644590	1	−0.453	−0.081
3	4 5 5 7	1.493896		−0.768	0.043

TABLE 2

l	z	$f(z)$	k	$\lambda_{\max}$	der
0	7 7 5 3	1.694596	4	3.175	—
1	7 7 5 5	1.588243	1	−3.727	—
2	4 7 5 5	1.494021		0.535	0.013

TABLE 3

l	z	$f(z)$	k	$\lambda_{\max}$	der
0	7 1 1 2	2.452621	1	−3.797	—
1	6 1 1 2	2.400695	2	−0.659	−0.130
2	6 7 1 2	1.956758	1	−1.054	—
3	5 7 1 2	1.952666	4	0.577	−0.102
4	5 7 1 5	1.706781	3	−0.542	−0.100
5	5 7 5 5	1.501278	1	−1.310	—
6	4 7 5 5	1.494021		0.535	0.013

found using (3.2) whose modulus is maximum, and der is the minimum directional derivative in the other co-ordinate directions (if this has to be calculated). From the results it is clear that in this case (1.4) has (at least) 2 local minima.

Finally we give an indication of how the algorithm copes with some larger problems. Table 4 shows the number of iterations for data matrices of varying dimension obtained by generating random positive numbers between 0 and 1. In all cases the initial approximation was chosen by setting z_j to the element $(a_i)_j$ closest to the average of all the $(a_i)_j$, $i = 1, 2, \cdots, m$. As might be expected, the number of iterations depends primarily on the value of n.

TABLE 4

m	n	iterations
20	10	9
40	10	11
60	10	11
100	10	11
200	10	17
60	20	26
100	20	25
200	20	32
60	40	49

4. Concluding remarks. A finite descent method has been developed for the minisum location problem for the Jaccard metric. The method is based on descent from vertex to vertex along co-ordinate directions, a strategy which seems a natural one for the problem. Any method operating outside this framework would require some increase in complexity, and this could well offset any gain in efficiency which might otherwise appear possible. This remains to be seen, but in any event, a new method is now available which it is hoped will prove useful for empirical sciences.

REFERENCES

[1] R. H. BARTELS, A. R. CONN AND J. W. SINCLAIR (1978), *Minimization techniques for piecewise differentiable functions: the l_1 solution to an overdetermined linear system*, SIAM J. Numer. Anal., 15, pp. 224–241.

[2] C. CHATFIELD AND A. J. COLLINS (1980), *Introduction to Multivariate Analysis*, Chapman and Hall, London and New York.

[3] A. H. CHEETHAM AND J. E. HAZEL (1969), *Binary (presence-absence) similarity coefficients*, J. Paleontol., 43, pp. 1130–1136.
[4] P. H. A. SNEATH (1957), *The application of computers to taxonomy*, J. Gen. Microbiol., 17, pp. 201–226.
[5] H. SPÄTH (1980), *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Wiley, Chichester.
[6] ——— (1981), *The minisum location problem for the Jaccard metric*, O. R. Spektrum, 3, pp. 91–94.
[7] G. A. WATSON (1980), *Approximation Theory and Numerical Methods*, Wiley, Chichester.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0015 $01.25/0

A DESCENT ALGORITHM FOR MINIMIZING POLYHEDRAL CONVEX FUNCTIONS*

D. I. CLARK† AND M. R. OSBORNE‡

Abstract. The idea of continuation applied to the proximal transform of a polyhedral convex function is used to develop a general procedure for the construction of descent algorithms which has the property that it specializes to variants of the projected gradient method when applied to such problems as l_1 and l_∞ curve fitting and linear programming. The novelty in this approach lies in the application of a generic form for the subdifferential of a polyhedral convex function which provides an explicit representation in terms of a small number of parameters. This is illustrated by applications to l_∞ curve fitting and the minimization of piecewise linear functions, and these examples serve to establish the feasibility of a unified approach. The power of the method is demonstrated by deriving an effective algorithm for the rank regression problem (the existence of such an algorithm makes practical the application of nonparametric procedures based on rank in robust estimation). The new approach also opens up the possibility of common implementation strategies, and a tableau like scheme is described based on the use of orthogonal matrix factorizations.

Key words. descent methods, projected gradient algorithm, nondifferentiable functions, subdifferential, convex functions, proximal transform, polyhedral convexity, maximum norm approximation, piecewise linear functions, penalty functions, rank regression, finite algorithms, tableau form, orthogonal transformations, line search, problem generation

1. Introduction. A polyhedral convex function (PCF) $F(\mathbf{x})$ assuming bounded values on a polyhedral convex subset of R^p is the pointwise maximum of a finite number of affine functions. That is $F: X \subseteq R^p \to R^1$ is given by

$$F(\mathbf{x}) = \begin{cases} \max\limits_{1 \leqq i \leqq N} \{\mathbf{f}_i^T \mathbf{x} - \alpha_i\}, & \mathbf{x} \in X, \\ +\infty, & \mathbf{x} \in X^c \end{cases} \tag{1.1}$$

where X is the polyhedral convex set

$$X = \{\mathbf{x};\ \mathbf{c}_i^T \mathbf{x} \geqq \beta_i,\ i = 1, 2, \cdots, M\}, \tag{1.2}$$

and the superscript c denotes the set complement. It is assumed that X has a proper interior in R^p. An explicit form for $F(\mathbf{x})$ can be given using the indicator function for X to write

$$F(\mathbf{x}) = \max_{1 \leqq i \leqq N} \{\mathbf{f}_i^T \mathbf{x} - \alpha_i\} + \delta(\mathbf{x}|X) \tag{1.3}$$

where

$$\delta(\mathbf{x}|X) = \begin{cases} 0, & \mathbf{x} \in X, \\ +\infty, & \mathbf{x} \in X^c. \end{cases} \tag{1.4}$$

Introducing the elementary PCF

$$\delta(t) = \begin{cases} 0, & t \leqq 0, \\ +\infty, & t > 0, \end{cases} \tag{1.5}$$

* Received by the editors June 2, 1981, and in final revised form July 23, 1982.
† Mathematics Department, University of Kentucky, Lexington, Kentucky 40506.
‡ Statistics Department, Institute of Advanced Studies, Australian National University, Canberra ACT 2601, Australia.

we have

$$\delta(\mathbf{x}|X) = \sum_{i=1}^{M} \delta(\beta_i - \mathbf{c}_i^T\mathbf{x}). \tag{1.6}$$

The problem of minimizing a PCF is an important example of a linear optimization problem, by which we mean that the first order necessary conditions for a minimum lead to a linear complementarity problem. From this it follows that the solution can be found by a finite process, at least in principle. Thus the algorithmic interest lies in developing efficient solution procedures. In this paper we give a descent algorithm for minimizing a PCF and develop our approach by considering in detail three examples which provide both specific contexts for treating the main varieties of problem behaviour, and important applications of the new algorithm. It is convenient to define an auxiliary vector $\mathbf{r}$ by

$$\mathbf{r} = A\mathbf{x} - \mathbf{b} \tag{1.7}$$

where $A: R^p \to R^n$ is assumed to have full column rank and $n > p$.

Example 1.1. *Parameter estimation in the maximum norm* (MAX). Let

$$\mathbf{f}_i^T = \rho_i(D)A, \qquad \alpha_i = \rho_i(D)\mathbf{b} \tag{1.8}$$

where $\rho_i(\cdot)$ should be read "row i of" (the corresponding column operator is $\kappa_i(\cdot)$),

$$D = \begin{bmatrix} I \\ -I \end{bmatrix}, \tag{1.9}$$

and $N = 2n$. In this case

$$F(\mathbf{x}) = \max_{1 \leqq i \leqq n} |r_i|, \tag{1.10}$$

and $X = R^p$.

Example 1.2. *Minimizing piecewise linear functions.* Here we consider the objective function

$$F(\mathbf{x}) = \chi \mathbf{c}^T\mathbf{x} + \sum_{i=1}^{n} \psi(\eta_i, \zeta_i, r_i) \tag{1.11}$$

where

$$\psi(\eta, \zeta, t) = \begin{cases} \eta|t|, & t < 0, \\ \zeta|t|, & t > 0, \end{cases} \tag{1.12}$$

and $\eta, \zeta \geqq 0$. Two important special cases are

(a) $\chi = 0$, $\eta = \zeta = 1$ corresponding to the l_1 approximation problem or least absolute deviations (LAD) curve fitting, and

(b) $\chi > 0$, $\eta = 1$, $\zeta = 0$ corresponding to the use of a penalty function to solve the linear programming (LP) problem

$$\min_{\mathbf{x} \in S} \mathbf{c}^T\mathbf{x}: S = \{\mathbf{x}; A\mathbf{x} \geqq \mathbf{b}\}. \tag{1.13}$$

Here χ is the penalty parameter which must be chosen small enough. An algorithm must be able to modify χ if the initial value is too large. The PCF corresponding to (1.11) is obtained by setting

$$\mathbf{f}_i^T = \chi \mathbf{c}^T + \rho_i(D)A, \qquad \alpha_i = \rho_i(D)\mathbf{b}, \qquad i = 1, 2, \cdots, N \tag{1.14}$$

where the rows of D correspond to the $N=2^n$ different ways of putting either $-\eta_i$ or ζ_i into the ith of n locations, $i=1,2,\cdots,n$.

Example 1.3. *Rank regression* (RR). Consider (1.7) written in the modified form

$$\mathbf{r}-x_0\mathbf{e}^{(n)}=A\mathbf{x}-\mathbf{b} \tag{1.7'}$$

where it is usual to refer to x_0 as the intercept term, and $\mathbf{e}^{(n)}$ is the vector of dimension n each component of which is 1. The method to be described eliminates x_0 from the problem and provides an estimate for the remaining parameters $\mathbf{x}$. Let *scores* d_i, $i=1,2,\cdots,n$ satisfy

(i) $d_1 \leqq d_2 \leqq \cdots \leqq d_n$,

(ii) $d_i=-d_{n-i+1}$, $i=1,2,\cdots,n$, and

(iii) at least one $d_i \neq 0$ so that $d_1<d_n$.

We define the *dispersion* of $\mathbf{r}$ by

$$\Delta(\mathbf{x})=\sum_{i=1}^{n} d_i r_{\mu(i)} \tag{1.15}$$

where μ is an index set with elements pointing to the components of $\mathbf{r}$ sorted into increasing order. Note that if $\mathbf{r}=\mathbf{e}^{(n)}$ then the dispersion of $\mathbf{r}$ is zero by (ii) above so that $\Delta(\mathbf{x})$ is independent of x_0. The dispersion is a nonnegative PCF with

$$\mathbf{f}_i^T=\rho_i(D)A, \qquad \alpha_i=\rho_i(D)\mathbf{b}, \qquad i=1,2,\cdots,N \tag{1.16}$$

where $D:R^n \to R^N$ has rows consisting of the N distinct permutations of the d_i, $i=1,2,\cdots,n$. If the d_i are distinct then $N=n!$. The suggestion that $\mathbf{x}$ be estimated by minimizing Δ is due to Jaeckel [12]. The resulting procedure is called rank regression.

It is assumed that F is bounded below so that the problem of minimizing $F(\mathbf{x})$ makes sense. However, it is not the case for any of the above examples that the minimizing $\mathbf{x}$ is necessarily unique unless further assumptions are made. One class of such assumptions frequently made to simplify exposition is that the problem is nondegenerate in a certain sense (the usage comes from linear programming), and it will be convenient to follow this practice. Neither the assumptions on the rank of A, nor the nondegeneracy assumptions are critical to the development of the basic algorithm. The weakening of these assumptions will be discussed subsequently [16]. Necessary and sufficient conditions for $\mathbf{x}$ to minimize F are stated below. They make use of properties of convex functions which are assumed. An appropriate reference to this material is [17].

LEMMA 1.1. *Let F be defined by* (1.1). *Then $\partial F(\mathbf{x})$, the subdifferential of F at $\mathbf{x}$, is given by*

$$\partial F(\mathbf{x})=\operatorname{conv}\{\mathbf{f}_i^T,\ i\in\pi\}-\sum_{i=1}^{M}\partial\delta(\beta_i-\mathbf{c}_i^T\mathbf{x})\mathbf{c}_i^T \tag{1.17}$$

where the index set π is defined by

$$\pi=\{i;\ \mathbf{f}_i^T\mathbf{x}-\alpha_i=F(\mathbf{x})\}, \tag{1.18}$$

and

$$\partial\delta(t)=\begin{cases}0, & t<0,\\ [0,\infty), & t=0,\\ \varnothing, & t>0.\end{cases} \tag{1.19}$$

THEOREM 1.1. *A necessary and sufficient condition for* $\mathbf{x} \in X$ *to minimize* $F(\mathbf{x})$ *is that there exist multipliers* $\mu_i \geqq 0$, $i = 1, 2, \cdots, M$ *such that*

$$\sum_{i=1}^{M} \mu_i \mathbf{c}_i^T \in \operatorname{conv}\{\mathbf{f}_i^T, i \in \pi\} \tag{1.20}$$

and

$$\mu_i(\mathbf{c}_i^T \mathbf{x} - \beta_i) = 0, \qquad i = 1, 2, \cdots, M. \tag{1.21}$$

Proof. This result follows by using Lemma 1.1 to restate the condition $0 \in \partial F(\mathbf{x})$ in the form of a multiplier relation.

For PCF's the question of uniqueness of the minimizer is intimately linked to the question of strong uniqueness. The properties of strong uniqueness required are stated in Theorem 1.2.

THEOREM 1.2. *Let* $\mathbf{x}^*$ *be a strong unique minimizer of the convex function* $g(\mathbf{x})$. *Then the following properties are equivalent* [19].

(a) $\exists \gamma > 0 \ni g(\mathbf{x}^* + \mathbf{t}) \geqq g(\mathbf{x}^*) + \gamma \|\mathbf{t}\|, \forall \mathbf{t} \in R^p$,

(b) $\exists \gamma > 0 \ni \forall \mathbf{t} \in R^p \ \exists \mathbf{v} \in \partial g(\mathbf{x}^*) \ni \mathbf{v}^T \mathbf{t} > \gamma \|\mathbf{t}\|$,

(c) $\exists \varepsilon > 0 \ni \forall \mathbf{t} \in R^p, \|\mathbf{t}\| < \varepsilon \Rightarrow \mathbf{t} \in \partial g(\mathbf{x}^*)$.

Remark 1.1. Property (c) can be stated as $0 \in \operatorname{int} \partial g(\mathbf{x}^*)$.

The link with PCF's can be made using the result [13].

THEOREM 1.3. *Uniqueness and strong uniqueness are equivalent for linear programming problems.*

To apply this result note that the minimization of the PCF $F(\mathbf{x})$ is equivalent to solving the linear programming problem

$$\min x_{p+1}$$

subject to

$$\mathbf{f}_i^T \mathbf{x} - \alpha_i \leqq x_{p+1}, \qquad i = 1, 2, \cdots, N,$$

and

$$C\mathbf{x} - \boldsymbol{\beta} \geqq 0, \tag{1.22}$$

and $F(\mathbf{x}) = x_{p+1}$ provided $\mathbf{x}$, x_{p+1} are feasible and equality holds in at least one equation $\mathbf{f}_i^T \mathbf{x} - \alpha_i = x_{p+1}$.

COROLLARY 1.1. *Uniqueness is equivalent to strong uniqueness for* PCF's.

Remark 1.2. The linear programming formulation provides a basis for a sensible approach only for the MAX example. The cause of difficulty here is not just the size of N because given $\mathbf{r}$ it is often possible to give rules for generating $\mathbf{f}_i$, α_i for $i \in \pi$ (see for example the discussion of a related problem in [1]). It can be explained by noting that when $X = R^p$ then linear programming theory ensures that at an extreme point of the feasible region $|\pi| \geqq p + 1$ and $\dim\{\mathbf{f}_i, i \in \pi\} = p$. Consider now the LAD problem. It is well known that these requirements force $\mathbf{r}$ to have at least p zero components so that $|\pi| \geqq 2^p$ as the rows of D defining the $\mathbf{f}_i$, $i \in \pi$, can have components ± 1 corresponding to each zero. A second example is provided by the RR problem. Assume the d_i are distinct. Then the linear programming conditions force a range of possibilities from p groups of pairs of tied residuals ($|\pi| = 2^p$) to one group of $p+1$ tied residuals ($|\pi| = (p+1)!$). In either case this implies that there are large numbers of redundant constraints active at an extreme point in this formulation of the problem, and this makes it difficult to determine which subsets of the constraints, adequate to

specify the extreme point, give directions consistent with all the active constraints. Thus it is necessary to seek out structural information (zero residuals, groups of tied residuals) and to develop descent directions and tests for optimality based on this.

The development of the basic ideas of the algorithm is given in the next section. We consider the use of continuation with the proximal transform of the PCF, and this proves attractive because

(i) rates of change of solution quantities are piecewise constant along the optimal trajectories, and

(ii) the minimizer of the proximal transform is also a minimizer of the PCF for sufficiently large but finite values of the continuation parameter.

Continuation could be used to follow a trajectory from the initial point to the solution, but this strategy depends on the whole past history of the process and on such possibly arbitrary decisions as the selection of the starting point. We prefer to use the defining equations of the continuation process, which are local in nature, to generate a descent vector. In fact the key step in our development is the determination of the conditions for a descent vector in a form which is directly identifiable with the basic equations of a projected gradient algorithm. However, the continuation process suggests naturally the development of further options.

(i) To test if any of the defining constraints on the current descent step can be relaxed. This is analogous to the process of deleting a constraint from the basis in linear programming, but we do not require a full basis to be established so that we allow explicitly nonsimplex steps.

(ii) To pass thorough points where the continuation information should be updated if it is possible to make further progress in the current descent step. In the LAD case one possibility is just the line search procedure of Barrodale and Roberts [2], and Bartels, Conn, and Sinclair [5] that has proved so successful in this application.

In presenting this information we have elected not to attempt a treatment in full generality, but rather to consider in detail each of Examples 1.1–1.3. This approach has the advantage that it permits us to isolate each of the different structural alternatives and different computing strategies in a specific environment.

Numerical implementation of the algorithms for each example is facilitated by the formal similarity of the resulting linear systems. A result is that the basic operations in each case can be carried out by updating a tableau much as in the classical implementations of the simplex algorithm for linear programming, but with the interesting difference that the updating procedure permits the use of orthogonal transformations. Results of numerical experiments show the new algorithm in a very favourable light. Taking this in conjunction with the experience with projected gradient algorithms reported by the other authors referenced in connection with the l_1 and MAX examples suggests that this general approach can be recommended for the minimization of polyhedral convex functions.

2. Development of the basic algorithm. We were lead to the basic form of the algorithm to be presented here as a result of developing a finite algorithm for minimizing Huber's M estimator using essentially a continuation method [8]. This suggested that we seek a form of objective function having the properties that

(i) an initial estimate can be found readily,

(ii) the rate of change of the solution is piecewise constant on the continuation trajectory, and

(iii) the solution is obtained after a finite change in the continuation parameters (implying a finite computation process).

Happily the first function tried met all requirements. Clearly, a further desirable property is that the function to be minimized is strictly convex, and the simplest way to make the PCF $F(\mathbf{x})$ strictly convex is to add a term $\frac{1}{2}\|\mathbf{x}-\mathbf{x}_0\|^2$, where the norm is the least squares norm. This suggests considering

$$P(\mathbf{x}, \mathbf{x}_0, \gamma) = \tfrac{1}{2}\|\mathbf{x}-\mathbf{x}_0\|^2 + \gamma F(\mathbf{x}). \tag{2.1}$$

Let $P^*(\mathbf{x}_0, \gamma)$ satisfy

$$P^*(\mathbf{x}_0, \gamma) = \min_{\mathbf{x}\in X} P(\mathbf{x}, \mathbf{x}_0, \gamma). \tag{2.2}$$

Then P^* is the proximal transform of F introduced by Moreau [15], and studied, in particular, by Rockafellar [18].

Given $\mathbf{x}_0$ let $\mathbf{x}(\gamma)$ minimize $P(\mathbf{x}, \mathbf{x}_0, \gamma)$. Then $\mathbf{x}(\gamma)$ is uniquely determined as P is strictly convex. Thus the continuation trajectory is well determined and $\mathbf{x}(0) = \mathbf{x}_0$. The key result fixes the behaviour of $\mathbf{x}(\gamma)$ for large γ.

THEOREM 2.1. *Let* $\mathbf{x}(\gamma)$ *minimize* $P(\mathbf{x}, \mathbf{x}_0, \gamma)$. *Then there exists* $\gamma_0 < \infty$ *such that* $\mathbf{x}(\gamma)$ *minimizes* $F(\mathbf{x})$ for $\gamma \geqq \gamma_0$.

Proof. If $\mathbf{x}^*$ minimizes $P(\mathbf{x}, \mathbf{x}_0, \gamma)$ then

$$0 \in \partial P(\mathbf{x}^*, \mathbf{x}_0, \gamma) = \mathbf{x}^* - \mathbf{x}_0 + \gamma\, \partial F(\mathbf{x}^*). \tag{2.3}$$

The proof of the theorem consists in constructing a suitable $\mathbf{x}^*$ from the set of minimizers of $F(\mathbf{x})$ when γ is large enough.

If the minimizer of F is unique then it is strongly unique by Theorem 1.3. By Theorem 1.2 $0 \in \operatorname{int} \partial F(\mathbf{x}^*)$ so that there exists $\zeta > 0$ such that

$$-\zeta(\mathbf{x}^* - \mathbf{x}_0) \in \partial F(\mathbf{x}^*).$$

Equation (2.3) now follows provided $\gamma\zeta \geqq 1$.

If the minimum of F is not unique then the set of minimizers forms a polyhedral convex set S defined by

$$\begin{aligned} S &= \{\mathbf{x} \in X;\ 0 \in \partial F(\mathbf{x})\} \\ &= \{\mathbf{x} \in X;\ \mathbf{f}_i^T\mathbf{x} - \alpha_i \leqq \min F,\ i \in \Pi\} \end{aligned} \tag{2.4}$$

where

$$\Pi = \{i;\ \exists \mathbf{x} \in S \ni \mathbf{f}_i^T\mathbf{x} - \alpha_i = \min F\}. \tag{2.5}$$

Now select $\mathbf{x}^*$ to minimize $\frac{1}{2}\|\mathbf{x}-\mathbf{x}_0\|^2$ for $\mathbf{x}$ in S. The necessary conditions for an optimum give multipliers $\lambda_i \geqq 0$, $i \in \Pi$, and $\nu_i \geqq 0$, $i = 1, 2, \cdots, M$ such that

$$\begin{aligned} &\text{(i)} && \mathbf{x}^* - \mathbf{x}_0 = -\sum_{i\in\Pi} \lambda_i \mathbf{f}_i^T + \sum_{i=1}^{M} \nu_i \mathbf{c}_i^T, \\ &\text{(ii)} && \lambda_i(\mathbf{f}_i^T\mathbf{x}^* - \alpha_i - \min F) = 0,\ i \in \Pi, \\ &\text{(iii)} && \nu_i(\mathbf{c}_i^T\mathbf{x}^* - \beta_i) = 0,\ i = 1, 2, \cdots, M. \end{aligned} \tag{2.6}$$

It follows from Lemma 1.1 and (2.6) that

$$-\frac{\mathbf{x}^* - \mathbf{x}_0}{\sum \lambda_i} \in \partial F(\mathbf{x}^*) \tag{2.7}$$

and by convexity as $0 \in \partial F(\mathbf{x}^*)$,

$$-\phi \frac{\mathbf{x}^* - \mathbf{x}_0}{\sum \lambda_i} \in \partial F(\mathbf{x}^*), \qquad 0 \leqq \phi \leqq 1.$$

Thus, $0 \in \partial P(\mathbf{x}^*, \mathbf{x}_0, \gamma)$ provided $\gamma > \sum_{i \in \Pi} \lambda_i$.

Remark 2.1. This theorem is implicit in [18, Prop. 8]. The above proof addresses the theorem directly and is given for completeness.

It remains to check that $d\mathbf{x}/d\gamma$ is piecewise constant on the optimal solution trajectory to verify that all the required properties hold. This is done first for the particular case in which the following assumptions are valid for each $\mathbf{x}$ in X. They will be specialized subsequently to the MAX problem. The index set π is defined in (1.18).

Assumption 2.1. $|\pi| \leqq p+1$.

Assumption 2.2. $\dim \{\mathbf{f}_i, i \in \pi\} = \min (|\pi|, p)$.

Assumption 2.3. $X = R^p$.

The first two assumptions are the most significant. The first is readily seen to be a nondegeneracy condition by considering the linear programming formulation of the minimization problem given in (1.22). By the rank assumption $\mathbf{x}$, $F(\mathbf{x})$ are determined completely when $|\pi| = p+1$. Thus if $|\pi| > p+1$ then there is the possibility of multiple specifications, and hence the chance of cycling between such reference sets without making progress. The second assumption is a numerical convenience, but it also serves in conjunction with the first assumption to exclude the cases in which zero or tied residuals give rise to large numbers of elements in $\partial F(\mathbf{x})$. If $X \neq R^p$ then a feasible initial point is required and active constraints must be taken into account using Lagrange multipliers.

To develop the continuation method consider the condition (2.3) for a minimum of $P(\mathbf{x}, \mathbf{x}_0, \gamma)$ which as a consequence of Assumption 2.3 reduces to

$$0 = \mathbf{x} - \mathbf{x}_0 + \gamma \sum_{i=1}^{|\pi|} \lambda_i \mathbf{f}_{\pi(i)} \tag{2.8}$$

where $\lambda_i \geqq 0$ and $\sum_{i=1}^{|\pi|} \lambda_i = 1$. Implicit in the idea of continuation is the requirement that (2.8) is (nearly always) differentiable as a function of γ. Differentiating gives

$$0 = \frac{d\mathbf{x}}{d\gamma} + \sum_{i=1}^{|\pi|} \xi_i \mathbf{f}_{\pi(i)} \tag{2.9}$$

where

$$\xi_i = \frac{d}{d\gamma}(\gamma \lambda_i), \qquad i = 1, 2, \cdots, |\pi|. \tag{2.10}$$

The feasibility of this step requires that the set of $\mathbf{f}_i$ pointed to by π does not change in an infinitesimal variation of γ, and, as we are interested in a minimizing trajectory, this requires that the subsidiary equations

$$\mathbf{f}_{\pi(i)}^T \frac{d\mathbf{x}}{d\gamma} = -\chi, \qquad i = 1, 2, \cdots, |\pi| \tag{2.11}$$

be satisfied for some $\chi > 0$. These equations can be arranged to give

$$0 = \frac{d\mathbf{x}}{d\gamma} + \mathbf{g} + \sum_{i=1}^{|\pi|-1} \xi_{i+1} \mathbf{v}_i \tag{2.12}$$

and

$$\mathbf{v}_i^T \frac{d\mathbf{x}}{d\gamma} = 0, \qquad i = 1, 2, \cdots, |\pi| - 1 \tag{2.13}$$

where

$$\mathbf{g} = \left(\sum_{i=1}^{|\pi|} \xi_i \right) \mathbf{f}_{\pi(1)}, \tag{2.14}$$

and

$$\mathbf{v}_i = \mathbf{f}_{\pi(i+1)} - \mathbf{f}_{\pi(1)}, \qquad i = 1, 2, \cdots, |\pi| - 1. \tag{2.15}$$

The scale factor χ can now be absorbed by setting

$$\sum_{i=1}^{|\pi|} \xi_i = 1. \tag{2.16}$$

Equations (2.12) and (2.13) are conveniently summarized in matrix form as

$$0 = \frac{d\mathbf{x}}{d\gamma} + \mathbf{g} + V\boldsymbol{\xi} \tag{2.12}$$

and

$$V^T \frac{d\mathbf{x}}{d\gamma} = 0. \tag{2.13}$$

These equations are fundamental to our method. In fact *each of Examples* 1.1–1.3 *will differ only in the specification and interpretation of the quantities appearing in* (2.12) *and* (2.13).

DEFINITION. V is the (current) *reference matrix*. The columns of V form the *reference*. $\mathbf{f}_{\pi(1)}$ is the *reference origin*.

Remark 2.2. It is interesting to compare (2.12) with the expression for an element in $\partial F(\mathbf{x})$. This gives

$$\mathbf{u} \in \partial F(\mathbf{x}) \Rightarrow \exists \lambda_i \geqq 0, \sum_{i=1}^{|\pi|} \lambda_i = 1 \ni \mathbf{u} = \sum_{i=1}^{|\pi|} \lambda_i \mathbf{f}_{\pi(i)}$$

$$\Rightarrow \mathbf{u} = \mathbf{g} + V \begin{bmatrix} \lambda_2 \\ \vdots \\ \lambda_{|\pi|} \end{bmatrix}$$

so that the current reference provides an explicit parametrization of $\partial F(\mathbf{x})$. Thus $d\mathbf{x}/d\gamma$ looks like an element of ∂F except that the ξ_i need not satisfy the convex hull conditions. If they do, and if $|\pi| = p + 1$ so that (2.13) forces $d\mathbf{x}/d\gamma = 0$, then optimality follows immediately. It is reasonable to expect that the ξ_i will evolve to give values satisfying the optimality conditions as the computation progresses.

It is easy to see that $d\mathbf{x}/d\gamma$, ξ_1, $\boldsymbol{\xi} = [\xi_2, \cdots, \xi_{|\pi|}]^T$ are uniquely determined by (2.12), (2.13) and hence by the index set π. We have

$$V^T V \boldsymbol{\xi} = -V^T \mathbf{g}$$

and this system is invertible by Assumption 2.2 so that $\boldsymbol{\xi}$ and also $\xi_1 = 1 - \sum_{i=2}^{|\pi|} \xi_i$ are independent of γ. Substituting for $\boldsymbol{\xi}$ in (2.12) gives $d\mathbf{x}/d\gamma$ which is also independent of γ, and this verifies that the rate of change of solution quantities is piecewise constant

on the solution trajectory. This suggests a continuation algorithm. However, there are two kinds of behaviour that interrupt the continuation process and cause revision of π and hence of $d\mathbf{x}/d\gamma$ and $\boldsymbol{\xi}$.

(i) A new i must be added to π if at any point $\mathbf{f}_i^T\mathbf{x}-\alpha_i = F(\mathbf{x})$. But (2.13) suggests that elements of V tend to stay around so that there is a tendency for the dimension of V to increase until it has rank p and the system (2.12), (2.13) consequently has only the zero solution for $d\mathbf{x}/d\gamma$. It follows that there must be a corresponding mechanism for deleting columns. This is provided by the second kind of behavior interrupting the continuation process.

(ii) $d\mathbf{x}/d\gamma$ determines an optimal trajectory only if $\lambda_i \geqq 0$ and $\sum \lambda_i = 1$ hold in (2.8). Assuming that the current values of ξ_i have been determined at γ_0 then integrating (2.10) gives

$$\lambda_i(\gamma) = \left(1-\frac{\gamma_0}{\gamma_1}\right)\xi_i + \frac{\gamma_0}{\gamma_1}\lambda_i(\gamma_0) \tag{2.17}$$

where the constant of integration is chosen to make $\mathbf{x}$ continuous as a function of γ. If at any stage $\lambda_i(\gamma) = 0$ and $\xi_i < 0$ then π must be updated by deleting i. A straightforward application of Lemma 2.4 (derived below) shows that optimality is maintained on the revised trajectory.

The criterion for deleting an element from π depends on the past history of the process through the term $\lambda_i(\gamma_0)$ in (2.17). Early numerical experiments tended to indicate that this could be less than satisfactory. This led us to consider an alternative method for using the system (2.12), (2.13) which exploits the local nature of its solution. This development is based on the following results.

LEMMA 2.1. *If* $|\pi| = p+1$, *and* $0 \leqq \xi_i \leqq 1$, $i = 1, 2, \cdots, p+1$ *then* $\mathbf{x}$ *minimizes* F.

Proof. If $|\pi| = p+1$ then (2.17) gives $d\mathbf{x}/d\gamma = 0$ so that (2.12) becomes

$$0 = \mathbf{g} + V\boldsymbol{\xi} = \sum_{i=1}^{|\pi|} \xi_i \mathbf{f}_{\pi(i)}$$

but this is equivalent to $0 \in \partial F(\mathbf{x})$.

COROLLARY 2.1. *The minimum of* F *is unique if and only if* $0 < \xi_i < 1$, $i = 1, 2, \cdots, p+1$.

Proof. By Theorems 1.2 and 1.3 $\mathbf{x}$ is a unique minimizer if and only if $0 \in \operatorname{int} \partial F(\mathbf{x})$. As V has full rank this requires

$$\mathbf{g} + V(\boldsymbol{\xi}+\boldsymbol{\eta}) \in \partial F(\mathbf{x})$$

for arbitrary $\boldsymbol{\eta}$ such that $\|\boldsymbol{\eta}\|$ small enough. This will be the case if and only if the components of $\boldsymbol{\xi}$ are not at their bounds.

LEMMA 2.2. *If the system* (2.12), (2.13) *has a nontrivial solution* $d\mathbf{x}/d\gamma$ *then for small enough* $\varepsilon > 0$

$$F\left(\mathbf{x} + \varepsilon \frac{d\mathbf{x}}{d\gamma}\right) < F(\mathbf{x}). \tag{2.18}$$

Proof. Let $\mathbf{u} \in \partial F(\mathbf{x})$. Then $\mathbf{u} = \mathbf{g} + V\boldsymbol{\lambda}$ for some $\boldsymbol{\lambda}$ satisfying $\lambda_i \geqq 0, \sum_{i=0}^{|\pi|} \lambda_i = 1$. Thus

$$\mathbf{u}^T \frac{d\mathbf{x}}{d\gamma} = (\mathbf{g} + V\boldsymbol{\lambda})^T \frac{d\mathbf{x}}{d\gamma} = \mathbf{g}^T \frac{d\mathbf{x}}{d\gamma} = -\left\|\frac{d\mathbf{x}}{d\lambda}\right\|^2. \tag{2.19}$$

The result now follows from the definition of the directional derivative of a convex function [17] as (2.19) ensures that the directional derivative is negative.

Remark 2.3. Equation (2.19) provides a generalization of the idea of a descent direction to convex nondifferentiable optimization problems.

Remark 2.4. To estimate the reduction that can be achieved note that for at least one $k \in \pi^c$ it must be true that

$$\mathbf{f}_k^T \frac{d\mathbf{x}}{d\gamma} > \mathbf{f}_i^T \frac{d\mathbf{x}}{d\gamma}, \qquad i \in \pi$$

as otherwise F is unbounded below. Progress can be made in the direction determined by $d\mathbf{x}/d\gamma$ until for some $k \in \pi^c$ (setting $r_j = \mathbf{f}_j^T\mathbf{x} - \alpha_j$, $j = 1, 2, \cdots, N$)

$$r_k + \varepsilon \mathbf{f}_k^T \frac{d\mathbf{x}}{d\gamma} = r_i + \varepsilon \mathbf{f}_i^T \frac{d\mathbf{x}}{d\gamma},$$

so that the maximum allowed value of ε is the smallest positive ε_k where

$$\varepsilon_k = -\frac{r_k - r_i}{(\mathbf{f}_k - \mathbf{f}_i)^T d\mathbf{x}/d\gamma}, \qquad k \in \pi^c, \tag{2.20}$$

and the actual reduction achieved is

$$\Delta r_i = -\frac{(r_i - r_k)\mathbf{f}_i^T d\mathbf{x}/d\gamma}{(\mathbf{f}_k - \mathbf{f}_i)^T d\mathbf{x}/d\gamma} > 0.$$

Remark 2.5. It should be noted that the nondegeneracy assumption is important in ensuring that residuals not in the current reference are strictly less than the current value of F.

As orthogonal factorization techniques will be used in the implementation of our algorithms it is convenient to use them at this stage to give an explicit form for the solution to (2.12), (2.13). *This result will be used frequently.*

LEMMA 2.3. *Let V be factorized into*

$$V = [Q_1 | Q_2] \begin{bmatrix} U \\ 0 \end{bmatrix} \tag{2.21}$$

where

$$Q = [Q_1 | Q_2]$$

is $p \times p$ orthogonal and U is $(|\pi| - 1) \times (|\pi| - 1)$ upper triangular. Then Q_1 is $p \times (|\pi| - 1)$ and provides an orthogonal basis for the column space of V, $d\mathbf{x}/d\gamma$ satisfies

$$\frac{d\mathbf{x}}{d\gamma} = -Q_2 Q_2^T \mathbf{g} \tag{2.22}$$

(hence projected gradient), and the multiplier vector $\boldsymbol{\xi}$ is given by

$$\boldsymbol{\xi} = -U^{-1} Q_1^T \mathbf{g}. \tag{2.23}$$

LEMMA 2.4. *Assume $\xi_{k+1} < 0$ for some k, $1 \leqq k \leqq |\pi| - 1$. If the corresponding column is removed from V to give a new reference matrix $\bar{V}$, and a new descent vector $d\bar{\mathbf{x}}/d\gamma$ then*

$$\mathbf{v}_k^T \frac{d\bar{\mathbf{x}}}{d\gamma} < 0 \tag{2.24}$$

so that $\pi(k+1)$ can be deleted from π. If $\xi_1 = 1 - \sum_{i=2}^{|\pi|} \xi_i < 0$ then the origin constraint can similarly be removed.

Proof. In the first part there is no restriction in assuming that $k = |\pi| - 1$ so that

$$V = [\bar{V}|\mathbf{v}_k]$$

and

$$\bar{Q}_2 = [\kappa_k(Q_1)|Q_2].$$

It follows from (2.14) and (2.16) that $\mathbf{g} = \bar{\mathbf{g}}$ as the origin is unchanged so that

$$-\mathbf{v}_k^T \frac{d\bar{\mathbf{x}}}{d\gamma} = \mathbf{v}_k^T \bar{Q}_2 \bar{Q}_2^T \mathbf{g} = \mathbf{v}_k^T \kappa_k(Q_1) \mathbf{e}_1^T \bar{Q}_2^T \mathbf{g}$$

as $\mathbf{v}_k^T Q_2 = 0$ by (2.21). Thus

$$-\mathbf{v}_k^T \frac{d\mathbf{x}}{d\gamma} = U_{kk} \kappa_k(Q_1)^T \mathbf{g} = -U_{kk}^2 \xi_{|\pi|}.$$

To prove the second part of the lemma assume that $\mathbf{f}_{\pi(2)}$ becomes the new origin. Now

$$\mathbf{f}_{\pi(i)} - \mathbf{f}_{\pi(1)} = (\mathbf{f}_{\pi(i)} - \mathbf{f}_{\pi(2)}) + (\mathbf{f}_{\pi(2)} - \mathbf{f}_{\pi(1)}),$$

so that

$$V = [0|\bar{V}] + \mathbf{v}_1 \mathbf{e}^{(k)T}, \qquad \mathbf{g} = \bar{\mathbf{g}} - \mathbf{v}_1$$

where $\mathbf{e}^{(k)}$ is the vector of dimension k each component of which is 1.

It is readily seen that

$$\bar{Q}_2 = [\mathbf{y}|Q_2]$$

where $\mathbf{y} \in \text{span}\,(\kappa_i(Q_1),\ i = 1, 2, \cdots, k)$ as $\rho_j(Q^T \bar{V}) = 0$ for $j = k+1, \cdots, p$ so that reduction of this matrix to upper triangular form need alter only the first k rows. We have

$$\begin{aligned} -\mathbf{v}_1^T \frac{d\bar{\mathbf{x}}}{d\gamma} &= \mathbf{v}_1^T [\mathbf{y}|Q_2] \begin{bmatrix} \mathbf{y}^T \\ Q_2^T \end{bmatrix} \{\mathbf{g} + \mathbf{v}_1\} \\ &= (\mathbf{v}_1^T \mathbf{y}) \mathbf{y}^T \{\mathbf{g} + \mathbf{v}_1\} \\ &= (\mathbf{v}_1^T \mathbf{y})^2 - (\mathbf{v}_1^T \mathbf{y}) \left\{ \mathbf{y}^T \frac{d\mathbf{x}}{d\gamma} + \mathbf{y}^T ([0|\bar{V}] + \mathbf{v}_1 \mathbf{e}^{(k)T}) \boldsymbol{\xi} \right\} \\ &= (\mathbf{v}_1^T \mathbf{y})^2 \left\{ 1 - \sum_{i=2}^{|\pi|} \xi_i \right\}. \end{aligned}$$

Remark 2.6. The striking feature of this result is that it does not require that $|\pi| = p + 1$, and this should be compared with the analogous situation in the simplex algorithm. Such a result is suggested by the connection with continuation where it is necessary if optimality is to be preserved on the continuation path. However, it also provides a way of weakening the constraints (2.13) restricting $d\mathbf{x}/d\gamma$ and so makes possible a more efficient direction of descent.

So far nothing has been said about the choice of origin constraint. The next result shows that $d\mathbf{x}/d\gamma$ is independent of the choice of origin so that (for example) a new origin can be selected to minimize the amount of work in updating the factorization of V.

LEMMA 2.5. *The current descent vector $d\mathbf{x}/d\gamma$ depends only on π and is independent of the choice of origin.*

Proof. Let V_j correspond to the choice of $\mathbf{f}_{\pi(j)}$ as origin. Then

$$V_j = [V + (\mathbf{f}_{\pi(1)} - \mathbf{f}_{\pi(j)})(\mathbf{e}^{(k)} + \mathbf{e}_j)^T]P$$
$$= [V - \kappa_j(V)(\mathbf{e}^{(k)} + \mathbf{e}_j)^T]P$$

where P is the permutation matrix interchanging columns 1 and j. But this gives immediately that

$$0 = Q_2^T V = Q_2^T V_j,$$

and

$$Q_2^T \mathbf{f}_{\pi(1)} = Q_2^T \mathbf{f}_{\pi(j)}.$$

The desired result is a consequence of these equations.

The form of descent algorithm we consider can now be described.

ALGORITHM 2.1.

(i) Set $\mathbf{x}_0$, determine initial $\mathbf{r}$, V. Factorize V if $|\pi| > 1$.
(ii) Compute $\boldsymbol{\xi}$.
(iii) Test for optimality. If optimal then stop.
(iv) Let k point to the most negative of the ξ_i, $i = 1, \cdots, |\pi|$. Delete corresponding column from V. $\pi := \pi/\{\pi(k)\}$.
(v) Update factorization of V to take account of column deletion.
(vi) Compute $d\mathbf{x}/d\gamma$.
(vii) Determine smallest $\varepsilon_k > 0$ from (2.24). $\pi := \pi \cup \{k\}$.
(viii) Update factorization of V to take account of variable addition.
(ix) Go to (ii).

Remark 2.7. The test (iv) to determine the column to be deleted from V is an "unnormalized" test in the sense that no attempt has been made to ensure that the problem scaling permits such comparisons to be made meaningfully. One form of normalized test looks at the rate of reduction in the PCF in each of the directions obtained by deleting a column of V corresponding to a negative ξ_i. Here we compare directional derivatives w_i given by

$$w_i = \frac{\mathbf{g}^T d\mathbf{x}^{(i)}/d\gamma}{\|d\mathbf{x}^{(i)}/d\gamma\|} = -\left\|\frac{d\mathbf{x}^{(i)}}{d\gamma}\right\|.$$

These are available provided the diagonal elements of $(U^TU)^{-1}$ are known, and these can be updated economically whenever the factorization of V is modified. For simplicity discussion here is limited to unnormalized tests and this corresponds to standard usage. However, our experiments have indicated some advantages in normalization, and there is some published evidence which supports this (for example the LAD algorithm given in [7]).

The nondegeneracy assumption ensures that this algorithm is finite.

THEOREM 2.2. *Provided the nondegeneracy assumption is satisfied then Algorithm* 2.1 *terminates at a minimum of F after at most a finite number of steps.*

Proof. Successive values of F are decreasing and bounded below so that the algorithm is convergent. Also any limit point $\mathbf{x}^*$ of the sequence of iterates $\{\mathbf{x}_i\}$ must minimize F as otherwise there is a nontrivial direction of descent for each $\mathbf{x}_i$ close enough to $\mathbf{x}^*$ and this will suffice to give a lower value than $F(\mathbf{x}^*)$ by the nondegeneracy assumption.

To show the algorithm is finite note that there are at most a finite number of possible index sets π. If $|\pi| = p+1$ then $\mathbf{x}$, $F(\mathbf{x})$ are determined uniquely so that once $|\pi| = p+1$ for the first time nondegeneracy and the descent property of $d\mathbf{x}/d\gamma$ ensure that index sets cannot be repeated. Thus the index set $\hat{\pi}$ can be repeated only if $|\hat{\pi}| \leqq p$ and at least one $\xi_i < 0$. Assume now that the number of repetitions of $\hat{\pi}$ is unbounded. There must be a corresponding (sub) sequence of points $\{\hat{\mathbf{x}}_k\}$ converging to $\mathbf{x}^*$ minimizing F as epi F is closed. Then $\hat{\pi} \subseteq \pi^*$, the extremal reference, and $|\pi^*| \leqq p+1$ by assumption 2.1. But this gives $\xi_i < 0$ for the optimal reference as the multiplier vector $\boldsymbol{\xi}$ is unique by Assumption 2.2.

Remark 2.8. It should be noted that the above argument does not require $\mathbf{x}^*$ to be unique. Nondegeneracy is used as a sufficient condition both to ensure that progress can be made in the direction defined by $d\mathbf{x}/d\gamma$, and to ensure a unique and therefore nonnegative multiplier vector at $\mathbf{x}^*$.

Example 2.1. This simple form of the algorithm is appropriate for the MAX problem, Example 1.1. The necessary identifications are easily made. Let $i \in \pi$. Then there is a $k = k(i)$, $1 \leqq k \leqq n$ such that (by (1.8), (1.9))

$$\mathbf{f}_i = \operatorname{sgn}(r_i)\mathbf{a}_k, \qquad \alpha_i = \operatorname{sgn}(r_i)b_k.$$

The nondegeneracy assumption is usual in discussions of this problem, while the rank assumption requires that

$$\operatorname{rank}(A_\pi) = \min(p, |\pi|)$$

where A_π is the matrix with rows given by the $\mathbf{a}_k$. This assumption also can be regarded as usual. Descent algorithms have been considered for the MAX problem by Cline [9], and by Bartels, Conn and Charalambous [4] who give references to earlier work. Bartels and Joe [14] consider carefully an algorithm which does not rely on our assumptions. The Bartels–Conn–Charalambous algorithm is similar to ours although it does not appear to exercise the option of relaxing columns from V before $|\pi| = p+1$, and it permits the use of a line search to extract the maximum benefit from each descent direction. To see how this is possible note that the minimum $\varepsilon_k > 0$ in (2.24) could occur for an $\mathbf{f}_k$ such that $\mathbf{f}_k^T d\mathbf{x}/d\gamma < 0$ so that there is scope to further reduce F in this direction. However, if this is done then the line search will eventually terminate with $|\pi| = 2$ in general as all the elements in the previous reference are now discarded at the first new tie. Thus at least another $p-1$ steps will be needed before an optimum can be attained. This is a sufficiently large number relative to the number of steps expected if the algorithm is to be competitive that this option was not investigated further. Supporting evidence for this decision is provided by Lemma 2.6 which shows that in the particular case under consideration a constraint entering the reference matrix cannot immediately be deleted so the structure of the problem seems to be opposed to moving past the current point. This will not be the case for Examples 1.2–1.3.

LEMMA 2.6. *Let k be the index to be added to π. If equation* (2.16) *for the updated quantities is written as*

$$0 = \frac{d\bar{\mathbf{x}}}{d\gamma} + \bar{\mathbf{g}} + [V|\mathbf{v}_k]\begin{bmatrix}\bar{\boldsymbol{\xi}}\\ \bar{\xi}_k\end{bmatrix}$$

then

$$\bar{\xi}_k = \frac{\mathbf{v}_k^T d\mathbf{x}/d\gamma}{\|Q_2^T \mathbf{v}_k\|^2} > 0. \tag{2.25}$$

Proof. In this case the updated factors give

$$\bar{Q}_1 = [Q_1 | \mathbf{y}]$$

where

$$\mathbf{y} = \frac{P_2 \mathbf{v}_k}{\|P_2 \mathbf{v}_k\|}$$

and P_2 is the projection onto the columns of Q_2, so that $P_2 = Q_2 Q_2^T$. Now the origin constraint is not changed when a new variable is added to the reference matrix so that $\mathbf{g} = \bar{\mathbf{g}}$. We have

$$0 = -\mathbf{y}^T \frac{d\bar{\mathbf{x}}}{d\gamma} = \mathbf{y}^T \mathbf{g} + \mathbf{y}^T \mathbf{v}_k \bar{\xi}_k = \frac{-\mathbf{v}_k^T \, d\mathbf{x}/d\gamma}{\|P_2 \mathbf{v}_k\|} + \|P_2 \mathbf{v}_k\| \bar{\xi}_k.$$

3. Piecewise linear functions. This is one of the important problem classes where the standard convex hull form of $\partial F(\mathbf{x})$ is impractical and where an alternative parametrization must be sought. Here the important structural information is summarized in the zero components of $\mathbf{r}$, and the calculation of $\partial F(\mathbf{x})$ is simplified by noting that

$$\partial\psi(\eta, \zeta, t) = \begin{cases} \psi(\eta, \zeta, t)/t, & t \neq 0, \\ [-\eta, \zeta], & t = 0. \end{cases} \tag{3.1}$$

Applying this formula to evaluate $\partial F(\mathbf{x})$ gives

LEMMA 3.1. *Let*

$$\sigma = \{i, r_i = 0\}, \tag{3.2}$$

$$\kappa_i(V) = \mathbf{a}_{\sigma(i)}, \qquad i = 1, 2, \cdots, |\sigma|, \tag{3.3}$$

and

$$\mathbf{g} = \chi \mathbf{c} + \sum_{i \in \sigma^c} \frac{\psi(\eta_i, \zeta_i, r_i)}{r_i} \mathbf{a}_i. \tag{3.4}$$

Then

$$\boldsymbol{u} \in \partial F(\mathbf{x}) \Leftrightarrow \mathbf{u} = \mathbf{g} + V\boldsymbol{\lambda} \tag{3.5}$$

where the components of $\boldsymbol{\lambda}$ *must satisfy the constraints*

$$-\eta_i \leqq \lambda_i \leqq \zeta_i, \qquad i = 1, 2, \cdots, |\sigma|. \tag{3.6}$$

In this case $P(\mathbf{x}, \mathbf{x}_0, \gamma)$ is given by

$$P(\mathbf{x}, \mathbf{x}_0, \gamma) = \tfrac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2 + \gamma F(\mathbf{x})$$

so the condition for a minimum is that there exist λ_i, $-\eta_i \leqq \lambda_i \leqq \zeta_i$, $i = 1, 2, \cdots, |\sigma|$, such that

$$0 = \mathbf{x} - \mathbf{x}_0 + \gamma\{\mathbf{g} + V\boldsymbol{\lambda}\}. \tag{3.7}$$

The equations for the continuation trajectory are found by differentiating (3.7) to obtain

$$0 = \frac{d\mathbf{x}}{d\gamma} + \mathbf{g} + V\boldsymbol{\xi} \tag{3.8}$$

where

$$\xi_i = \frac{d}{d\gamma}(\gamma\lambda_i), \qquad i = 1, 2, \cdots, |\sigma|, \tag{3.9}$$

and by noting that differentiability implies that the set of zero residuals must not change in a small enough variation in γ so that

$$V^T \frac{d\mathbf{x}}{d\gamma} = 0. \tag{3.10}$$

The similarity between these equations and equations (2.12), (2.13) is evident, and the further development of the continuation algorithm follows as in the previous section. For this reason we proceed directly to consider the use of (3.8), (3.10) for calculating a descent direction. It is convenient to make essentially the same simplifying assumptions as before. Note that they apply for each $\mathbf{x} \in X = R^p$.

Assumption 3.1. $|\sigma| \leqq p$.

Assumption 3.2. $\operatorname{rank}(V) = |\sigma|$.

The following results parallel Lemmas 2.1, 2.2, 2.4, 2.5.

LEMMA 3.2. *If* $|\sigma| = p$, *and* $-\eta_i \leqq \xi_i \leqq \zeta_i$, $i = 1, 2, \cdots, p$, *then* $\mathbf{x}$ *minimizes* F.

Proof. If $|\sigma| = p$ then it follows from Assumption 3.2 and equation (3.10) that $d\mathbf{x}/d\gamma = 0$. But then (3.8) and Lemma 3.1 give $0 \in \partial F(\mathbf{x})$.

The argument used to prove Corollary 2.1 now gives:

COROLLARY 3.1. *Let* $\mathbf{x}$ *minimize* $F(\mathbf{x})$. *Then* $\mathbf{x}$ *is unique if and only if* $-\eta_i < \xi_i < \zeta_i$, $i = 1, 2, \cdots, p$.

The proof of the next result is identical to the proof of Lemma 2.2

LEMMA 3.3. *If the system of equations* (3.8), (3.10) *has a nontrivial solution* $d\mathbf{x}/d\gamma$ *then this defines a direction of descent for minimizing* F *at* $\mathbf{x}$.

If $\mathbf{x}$ is not optimal then either $|\sigma| < p$ and/or there exists k, $1 \leqq k \leqq |\sigma|$, such that $\xi_k \notin [-\eta_k, \zeta_k]$. The next result shows that in the latter case $\kappa_k(V)$ can be deleted from the reference. If $|\sigma| = p$ then this operation permits the construction of a nontrivial direction of descent.

LEMMA 3.4. *If* $\xi_k \notin [-\eta_k, \zeta_k]$, *and* $\mathbf{a}_{\sigma(k)}$ *is deleted from the reference,*

$$\operatorname{sgn}\left(\mathbf{a}_{\sigma(k)}^T \frac{d\bar{\mathbf{x}}}{d\gamma}\right) = \operatorname{sgn}(\xi_k)$$

where quantities computed after the deletion are indicated by bars.

Proof. The solution of (3.8), (3.10) is given by Lemma 2.3. Identifying quantities before and after deletion gives

$$V = [\bar{V}|\mathbf{a}_{\sigma(k)}], \qquad \mathbf{g} = \bar{\mathbf{g}} - \tau_k \mathbf{a}_{\sigma(k)}$$

where $\tau_k = \psi(\eta_k, \zeta_k, \xi_k)/\xi_k$, and

$$\bar{Q}_2 = [\mathbf{y}|Q_2]$$

where $\mathbf{y} = \kappa_{|\sigma|}(Q_1)$. Thus

$$\begin{aligned} -\mathbf{a}_{\sigma(k)}^T \frac{d\bar{\mathbf{x}}}{d\gamma} &= \mathbf{a}_{\sigma(k)}^T \bar{Q}_2 \bar{Q}_2^T \bar{\mathbf{g}} = \mathbf{a}_{\sigma(k)}^T \bar{Q}_2 \bar{Q}_2^T (\mathbf{g} + \tau_k \mathbf{a}_{\sigma(k)}) \\ &= U_{kk} \mathbf{e}_1^T \bar{Q}_2^T (\mathbf{g} + \tau_k \mathbf{a}_{\sigma(k)}) = U_{kk}^2 (\tau_k - \xi_k) \end{aligned}$$

using (3.8). But by assumption $\operatorname{sgn}(\xi_k) = \operatorname{sgn}(\xi_k - \tau_k)$.

Remark 3.2. This result does not require $|\sigma| = p$. Thus a reference column associated with an infeasible multiplier can be relaxed at any stage. The reason for considering this strategy is that the resulting descent direction is less constrained in (3.10), but this is not a proof that this direction is more efficient in any other sense.

The final result shows that it is possible to predict the multiplier associated with a column added to the reference. This result parallels Lemma 2.5. A key feature is that all the information needed to compute the multiplier (component of $\boldsymbol{\xi}$!) is available before the reference is augmented.

LEMMA 3.5. *If* $\mathbf{a}_k$ *is the column to be added to the reference then*

$$\bar{\xi}_k = \tau_k + \mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma} \Big/ \|P_2 \mathbf{a}_k\|^2 \tag{3.11}$$

where $\bar{\xi}_k$ *is the resulting multiplier,*

$$\bar{\mathbf{g}} = \mathbf{g} - \tau_k \mathbf{a}_k,$$

P_2 *is the projection onto the columns of* Q_2, $\tau_k = \psi(\eta_k, \zeta_k, \theta_k)/\theta_k$, *and* $\theta_k = -1$ *if* r_k *goes from negative to positive and* $+1$ *otherwise.*

Proof. For the updated reference we have

$$\bar{V} = [V | \mathbf{a}_k] \quad \text{and} \quad \bar{Q}_1 = [Q_1 | \mathbf{y}]$$

where $\mathbf{y} = P_2 \mathbf{a}_k / \|P_2 \mathbf{a}_k\|$. Thus, taking the scalar product of (3.8) with $\mathbf{y}$, we have

$$0 = \mathbf{y}^T \frac{d\bar{\mathbf{x}}}{d\gamma} = \mathbf{y}^T \mathbf{g} - \tau_k \mathbf{y}^T \mathbf{a}_k + \bar{\xi}_k \mathbf{y}^T \mathbf{a}_k = \frac{-\mathbf{a}_k^T \, d\mathbf{x}/d\gamma}{\|P_2 \mathbf{a}_k\|} + (\bar{\xi}_k - \tau_k)\|P_2 \mathbf{a}_k\|.$$

Remark 3.3. To estimate the step in the direction determined by $d\mathbf{x}/d\gamma$ note that it is possible to move until a new zero residual is introduced. For the kth residual the distance ε_k to the next zero is given by

$$r_k + \varepsilon_k \mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma} = 0. \tag{3.12}$$

Thus the step is determined by $\hat{\varepsilon} = \min \varepsilon_k$ over values of $k \in \sigma^c$ such that $\varepsilon_k > 0$. That a positive step is possible is a consequence of the nondegeneracy assumption (Assumption 3.1). The change in F is given by

$$\sum_{k \in \sigma^c} \tau_k \left(r_k + \hat{\varepsilon} \mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma} \right) - \sum_{k \in \sigma^c} \tau_k r_k = \hat{\varepsilon} \left(\sum_{k \in \sigma^c} \tau_k \mathbf{a}_k^T \right) \frac{d\mathbf{x}}{d\gamma} = \hat{\varepsilon} \mathbf{g}^T \frac{d\mathbf{x}}{d\gamma} = -\hat{\varepsilon} \left\| \frac{d\mathbf{x}}{d\gamma} \right\|^2. \tag{3.13}$$

Remark 3.4. In contrast to the MAX problem several options are available at the new point $\mathbf{x} + \hat{\varepsilon}\, d\mathbf{x}/d\gamma$.

(a) The column corresponding to the new zero residual can be added to the reference and a new descent direction computed. This corresponds to the strategy used in the MAX problem.

(b) The multiplier associated with the column to be added can be computed using (3.11). If $\bar{\xi}_k \notin [-\eta_k, \zeta_k]$ then it is a candidate for immediate deletion, and this is an attractive strategy because the result is to leave the reference unchanged so that the relatively costly operation of updating the matrix factors is avoided. However, note

that the residual in question (say r_k) passes through zero so that $\mathbf{g}$ must be updated by

$$\mathbf{g} := \mathbf{g} - (\zeta_k + \eta_k)\theta_k \mathbf{a}_k. \tag{3.14}$$

It follows that, in general, the direction of $d\mathbf{x}/d\gamma$ is changed unless $|\sigma| = p-1$ when the direction is essentially defined uniquely by (3.10).

(c) The directional derivative of F in the direction determined by $d\mathbf{x}/d\gamma$ is equal to $\mathbf{g}^T(d\mathbf{x}/d\gamma)/\|d\mathbf{x}/d\gamma\|$. By (3.14) the change in directional derivative in passing through $r_k = 0$ is

$$\Delta F = -(\zeta_k + \eta_k)\theta_k \mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma} \Big/ \left\|\frac{d\mathbf{x}}{d\gamma}\right\| = (\zeta_k + \eta_k)\left|\mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma}\right| \Big/ \left\|\frac{d\mathbf{x}}{d\gamma}\right\|. \tag{3.15}$$

Thus $d\mathbf{x}/d\gamma$ is still downhill at $\mathbf{x} + \hat{\varepsilon}\, d\mathbf{x}/d\gamma$ provided

$$\mathbf{g}^T \frac{d\mathbf{x}}{d\gamma} + (\zeta_k + \eta_k)\left|\mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma}\right| < 0. \tag{3.16}$$

If $d\mathbf{x}/d\gamma$ is still downhill then more progress can be made in this direction without any further computation. The important feature of this case is that the ε_k already computed contain the information needed to locate the subsequent zero residuals in the direction $d\mathbf{x}/d\gamma$. This makes possible the provision of an efficient line search algorithm.

Remark 3.5. Options (b) and (c) coincide when $|\sigma| = p-1$. In this case Q_2 has a single column (say $\mathbf{q}$) so that

$$\mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma} = -\mathbf{a}_k^T \mathbf{q}\mathbf{q}^T \mathbf{g} = -\operatorname{sgn}(\mathbf{a}_k^T \mathbf{q})\|P_2 \mathbf{a}_k\| \mathbf{q}^T \mathbf{g},$$

and

$$\frac{\mathbf{a}_k^T\, d\mathbf{x}/d\gamma}{\|P_2 \mathbf{a}_k\|^2} = -\frac{\operatorname{sgn}(\mathbf{a}_k^T\, d\mathbf{x}/d\gamma)(d\mathbf{x}^T/d\gamma)\mathbf{g}}{|\mathbf{a}_k^T\, d\mathbf{x}/d\gamma|}.$$

Now $\operatorname{sgn}(\mathbf{a}_k^T\, d\mathbf{x}/d\gamma) = -\theta_k$, and substituting in (3.11) gives

$$-|\bar{\xi}_k| = \psi(\eta_k, \zeta_k, \theta_k) - \frac{|\mathbf{a}_k^T\, d\mathbf{x}/d\gamma|}{\|P_2 \mathbf{a}_k\|^2}$$

so that the constraint is deleted immediately if

$$\frac{d\mathbf{x}^T}{d\gamma}\mathbf{g} \Big/ \left|\mathbf{a}_k^T \frac{d\mathbf{x}}{d\gamma}\right| < -(\eta_k + \zeta_k)$$

and this is equivalent to (3.16). Fast implementation seems more difficult with option (b).

We now give a summary of the algorithm possibilities. The basic procedure is much the same as Algorithm 2.1, but the various options for adding to and deleting from the reference are indicated.

ALGORITHM 3.1.

(i) Set $\mathbf{x}_0$, determine initial $\mathbf{r}$, V. Factorize V if $|\sigma| > 0$.
(ii) Compute $\boldsymbol{\xi}$.
(iii) Test for optimality ($|\sigma| = p$, $-\eta_k \leqq \xi_k \leqq \zeta_k$, $k = 1, 2, \cdots, |\sigma|$).
(iv) Let k point to the component of $\boldsymbol{\xi}$ maximizing $\phi_j = \max(\xi_j - \zeta_j, -\eta_j - \xi_j)$, $j = 1, 2, \cdots, |\sigma|$. If $\phi_j > 0$ then the kth column can be deleted from the reference.

Option D(i). Delete kth column.
Option D(ii). Delete kth column only if $|\sigma| = p$.

(v) If column deletion update $\mathbf{g}$ and factors of V.

(vi) Compute $d\mathbf{x}/d\gamma$, $\Sigma = \{\varepsilon_k;\ k \in \sigma^c,\ \varepsilon_k > 0\}$, set $\varepsilon = 0$.

(vii) Determine column to enter reference.
Option A(i) Determine k corresponding to smallest ε_j in Σ, $\varepsilon = \varepsilon_k$.
Option A(ii) Determine k corresponding to smallest ε_j in Σ, $\varepsilon = \varepsilon_k$.
Compute $\bar{\xi}_k$ using (3.11).
If $\bar{\xi}_k \notin [-\eta_k, \zeta_k]$ then $\mathbf{g} := \mathbf{g} - (\zeta_k + \eta_k)\theta_k \mathbf{a}_k$
$\mathbf{x} := \mathbf{x} + \varepsilon\, d\mathbf{x}/d\gamma$
go to (vi).
Option A(iii) Determine k corresponding to smallest $\varepsilon_j > 0$ in Σ.

$$F' := F' + (\zeta_k + \eta_k)|\mathbf{a}_k^T\, d\mathbf{x}/d\gamma|;\ \varepsilon := \varepsilon + \varepsilon_k$$

$$\varepsilon_j := \varepsilon_j - \varepsilon_k\ \forall \varepsilon_j \in \Sigma$$

repeat while $F' < 0$.
(But don't do computations like this!)
$\mathbf{x} := \mathbf{x} + \varepsilon\, d\mathbf{x}/d\gamma$.

(viii) Update $\mathbf{g}$ and factorization of V to take account of variable addition.

(ix) Go to (ii).

THEOREM 3.1. *Provided the nondegeneracy assumption is satisfied then Algorithm* 3.1 *terminates at a minimum of the* LAD *problem after at most a finite number of steps.*

Proof. This follows by essentially the same argument as Theorem 2.1.

Remark 3.6. This algorithm specializes to known algorithms in certain cases. For example in the LAD problem the choice of options D(ii) and A(iii) gives the algorithm of Bartels, Conn and Sinclair [5]. The method of Barrodale and Roberts [2] fits into the same picture once $|\sigma| = p$ as then projected gradient and reduced gradient algorithms generate essentially the same descent directions. Conn [10] has considered a penalty function algorithm for linear programming which also corresponds closely to our procedure when $\eta_k = 1$, $\zeta_k = 0$, $k = 1, 2, \cdots, n$. This requires a procedure for modifying the penalty parameter χ. An appropriate strategy corresponds to that adopted by Bartels in [3] and replaces step (iv) of the algorithm by

(iv)′ Let k point to the arithmetically largest component of $\boldsymbol{\xi}$. If $\xi_k > 0$ then $\mathbf{a}_{\sigma(k)}$ is deleted from V, $\sigma := \sigma/\{\sigma(k)\}$. If $|\sigma| < p$ then go to (v).
(Otherwise $\xi_k < 0$ and $|\sigma| = p$ and the appropriate strategy is to reduce χ to obtain a downhill direction.)
$\chi := \alpha\chi$ $(0 < \alpha < 1)$
go to (ii).

4. Rank regression. The rank regression problem, Example 1.3, provides a second type of example in which the number of elements in $\partial F(\mathbf{x}) = \partial\Delta(\mathbf{x})$ is potentially extremely large. Here the cause is ties in the ranking of the components of $\mathbf{r}$ as permuting the corresponding scores does not change $\Delta(\mathbf{x})$. Thus each distinct permutation of the d_i associated with each different set of ties generate a new element in $\partial\Delta(\mathbf{x})$ by (1.16) and (1.17).

To specify the tied residuals we introduce an index set σ such that

$$\sigma = \sigma_1 \cup \sigma_2 \cdots \cup \sigma_q \tag{4.1}$$

where σ_i is the subset pointing to the ith group of ties, $i = 1, 2, \cdots, q$. It will be

convenient to set

$$|\sigma_i| = m_i + 1, \qquad i = 1, 2, \cdots, q \tag{4.2}$$

and

$$\sum_{i=1}^{q} m_i = m. \tag{4.3}$$

Associated with σ_i we define the *group origin* $a_{\sigma_i(1)}$, and the matrix V_i where

$$\mathbf{v}_j^{(i)} = \kappa_j(V_i) = \mathbf{a}_{\sigma_i(j+1)} - \mathbf{a}_{\sigma_i(1)}, \qquad j = 1, 2, \cdots, m_i. \tag{4.4}$$

The matrix V which appears in the parameterization of $\partial\Delta(\mathbf{x})$ is given by

$$V = [V_1 | V_2 | \cdots | V_q]. \tag{4.5}$$

The simplifying assumptions in this case become:

Assumption 4.1. $m \leqq p$.

Assumption 4.2. rank $(V) = m$.

As before they are assumed to hold for each $\mathbf{x} \in X = R^p$.

LEMMA 4.1. $\mathbf{u} \in \partial\Delta(\mathbf{x}) \Leftrightarrow \mathbf{u} = \mathbf{g} + V\boldsymbol{\lambda}$ *where*

$$\mathbf{g} = \sum_{i \in \sigma^c} d_{\nu(i)} \mathbf{a}_i + \sum_{i=1}^{q} \left\{ \sum_{j=1}^{m_i+1} d_{\nu(\sigma_i(j))} \right\} \mathbf{a}_{\sigma_i(1)}, \tag{4.6}$$

where ν is the index set such that

(a) *$d_{\nu(j)}$ is the score associated with r_j, and*

(b) *within each group $d_{\nu(\sigma_i(j))} \geqq d_{\nu(\sigma_i(k))}$ if $j \geqq k$, and where*

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(1)} \oplus \boldsymbol{\lambda}^{(2)} \oplus \cdots \boldsymbol{\lambda}^{(q)},$$

$\boldsymbol{\lambda}^{(i)}$ is the vector of multipliers associated with the ith group, and

$$\sum_{j=1}^{k} d_{\nu(\sigma_i(j))} \leqq \sum_{j=1}^{k} \lambda_{\omega(j)}^{(i)} \leqq \sum_{j=1}^{k} d_{\nu(\sigma_i(m_i - j + 2))} \tag{4.7}$$

for $k = 1, 2, \cdots, m_i$, and ω any permutation of $1, 2, \cdots, m_i$.

Remark 4.1. The inequalities (4.7) do not depend on the choice of group origin. To see this let the contribution to $\partial\Delta(\mathbf{x})$ from the ith group with $\sigma_i(k)$ pointing to the group origin be $\mathbf{g}_i^{(k)} + V_i^{(k)} \boldsymbol{\lambda}^{(i,k)}$. Then a change of origin to $\sigma_i(1)$ gives

$$\begin{aligned} \mathbf{g}_i^{(k)} + V_i^{(k)} \boldsymbol{\lambda}^{(i,k)} &= \mathbf{g}_i^{(1)} + \zeta \mathbf{v}_{k-1} + [V_i - \mathbf{v}_{k-1} \mathbf{e}^{(m_i)T} - \mathbf{v}_{k-1} \mathbf{e}_{k-1}^T] \boldsymbol{\lambda}^{(i,k)} \\ &= \mathbf{g}_i^{(1)} + V_i \left\{ \boldsymbol{\lambda}^{(i,k)} + \mathbf{e}_{k-1} \left(\zeta - \sum_{j=1}^{m_i} \lambda_j^{(i,k)} - \lambda_{k-1}^{(i,k)} \right) \right\} \\ &= \mathbf{g}_i^{(1)} + V_i \boldsymbol{\lambda}^{(i)} \end{aligned}$$

where $\zeta = \sum_{j \in \sigma_i} d_{\nu(j)}$ and

$$\boldsymbol{\lambda}^{(i)} = \begin{bmatrix} \lambda_1^{(i,k)} \\ \vdots \\ \lambda_{k-2}^{(i,k)} \\ \zeta - \sum_{j=1}^{m_i} \lambda_j^{(i,k)} \\ \vdots \\ \lambda_{m_i}^{(i,k)} \end{bmatrix}.$$

Direct substitution shows that the inequalities (4.7) hold for $\boldsymbol{\lambda}^{(i,k)}$ if and only if they hold for $\boldsymbol{\lambda}^{(i)}$.

Proof. It is readily seen that the elements in $\partial\Delta(\mathbf{x})$ must have the declared form by using (1.16), collecting terms, and summing explicitly. The more difficult part is determining the constraints on $\boldsymbol{\lambda}^{(i)}$, and we use directly the subgradient inequality

$$\mathbf{u}\in\partial\Delta(\mathbf{x}) \Leftrightarrow \Delta(\mathbf{x}+\mathbf{t})\geqq\Delta(\mathbf{x})+\mathbf{u}^T\mathbf{t} \quad \forall\mathbf{t}\in R^p. \tag{4.8}$$

Let $W=[V|E_\sigma]^{-1}$ where E_σ is made up of columns of the unit matrix chosen so that $[V|E_\sigma]$ has full rank. Then any $\mathbf{t}$ can be written in an obvious notation as

$$\mathbf{t}=\sum_{j=1}^{p}\alpha_j\rho_j(W)^T=\sum_{i=1}^{q}\sum_{j=1}^{m_i}\alpha_j^{(i)}\mathbf{w}_j^{(i)}+\sum_{j=p-m}^{p}\alpha_j\mathbf{w}_j. \tag{4.9}$$

Because the ordering of the columns of V_i is disposable, and because the system of inequalities (4.7) is invariant under change of group origin, there is no lack of generality in assuming that the $\alpha_j^{(i)}$ are ordered for each i such that

$$0\leqq\alpha_1^{(i)}\leqq\alpha_2^{(i)}\leqq\cdots\leqq\alpha_{m_i}^{(i)}.$$

The only point to note is that the group origin can always be chosen to ensure that $\alpha_1^{(i)}$ is nonnegative. To see this let $\mathbf{a}_{\sigma_i(s)}$ become the new origin. Then $\alpha_{s-1}^{(i)}:=-\alpha_{s-1}^{(i)}$ while for $j\neq s-1$

$$\mathbf{v}_j^{(i)}:=\mathbf{v}_j^{(i)}-\mathbf{v}_{s-1}^{(i)}, \qquad \alpha_j^{(i)}:=\alpha_j^{(i)}-\alpha_{s-1}^{(i)}.$$

Thus starting with any ordering it is necessary only to choose the origin corresponding to the most negative of the $\alpha_j^{(i)}$ and then to reorder the columns of the resulting V_i.

With these assumptions, and assuming also that $\|\mathbf{t}\|$ is sufficiently small that only rankings within groups are disturbed, then the subgradient inequality gives

$$\begin{aligned}\Delta(\mathbf{x}+\mathbf{t})&=\Delta(\mathbf{x})+\mathbf{g}^T\mathbf{t}+\sum_{i=1}^{q}\sum_{j\in\sigma_i}d_{\nu(j)}(\mathbf{a}_j-\mathbf{a}_{\sigma_i(1)})^T\mathbf{t}\\&=\Delta(\mathbf{x})+\mathbf{g}^T\mathbf{t}+\sum_{i=1}^{q}\sum_{j=1}^{m_i}\alpha_j^{(i)}d_{\nu(\sigma_i(1))+j}\\&\geqq\Delta(\mathbf{x})+\mathbf{g}^T\mathbf{t}+\sum_{i=1}^{q}\sum_{j=1}^{m_i}\alpha_j^{(i)}\lambda_j^{(i)}.\end{aligned}$$

In similar fashion, replacing $\mathbf{t}$ by $-\mathbf{t}$ in (4.8) gives

$$\begin{aligned}\Delta(\mathbf{x}-\mathbf{t})&=\Delta(\mathbf{x})-\mathbf{g}^T\mathbf{t}-\sum_{i=1}^{q}\sum_{j=1}^{m_i}\alpha_j^{(i)}d_{\nu(\sigma_i(m_i+1))-j}\\&\geqq\Delta(\mathbf{x})-\mathbf{g}^T\mathbf{t}-\sum_{i=1}^{q}\sum_{j=1}^{m_i}\alpha_j^{(i)}\lambda_j^{(i)}.\end{aligned}$$

Thus $\mathbf{u}\in\partial\Delta(\mathbf{x})$ if and only if

$$\sum_{j=1}^{m_i}\alpha_j^{(i)}d_{\nu(\sigma_i(m_i+1))-j}\leqq\sum_{j=1}^{m_i}\alpha_j^{(i)}\lambda_{\omega(j)}^{(i,k)}\leqq\sum_{j=1}^{m_i}\alpha^{(i)}d_{\nu(\sigma_i(1))+j},$$

as the $\alpha_j^{(i)}$ can be varied independently for each i. Note that these inequalities take account of the choice of origin and the reordering of the columns of V_i required to order the $\alpha_j^{(i)}$ to permit arbitrary $\mathbf{t}$. These inequalities can be rearranged to give the

equivalent set

$$\begin{aligned}
&\beta_1 \sum_{j=1}^{m_i} d_{\nu(\sigma_i(m_i+1))-j} + \sum_{s=2}^{m_i} \beta_s \sum_{j=s}^{m_i} d_{\nu(\sigma_i(m_i+1))-j} \\
&\leqq \beta_1 \sum_{j=1}^{m_i} \lambda^{(i,k)}_{\omega(j)} + \sum_{s=2}^{m_i} \beta_s \sum_{j=s}^{m_i} \lambda^{(i,k)}_{\omega(j)} \qquad (4.10) \\
&\leqq \beta_1 \sum_{j=1}^{m_i} d_{\nu(\sigma_i(1))+j} + \sum_{s=2}^{m_i} \beta_s \sum_{j=s}^{m_i} d_{\nu(\sigma_i(1))+j}
\end{aligned}$$

where $\beta_1 = \alpha_1^{(i)}$, $\beta_j = \alpha_j^{(i)} - \alpha_{j-1}^{(i)}$, $j = 2, \cdots, m_i$ are arbitrary nonnegative numbers. The inequalities (4.7) follow at once as a consequence of Remark 4.1.

Lemma 4.1 gives a parametrization of $\partial\Delta(\mathbf{x})$ in the form needed for the development of the descent algorithm. The equations for $d\mathbf{x}/d\gamma$ are

$$0 = \frac{d\mathbf{x}}{d\gamma} + \mathbf{g} + V\boldsymbol{\xi} \qquad (4.11)$$

and

$$V^T \frac{d\mathbf{x}}{d\gamma} = 0. \qquad (4.12)$$

If these equations have a nontrivial solution for $d\mathbf{x}/d\gamma$ then this defines a descent direction.

LEMMA 4.2.

$$\mathbf{u}^T \frac{d\mathbf{x}}{d\gamma} = -\left\| \frac{d\mathbf{x}}{d\gamma} \right\|^2 \quad \forall \mathbf{u} \in \partial\Delta(\mathbf{x}).$$

Progress can be made in the direction defined by $d\mathbf{x}/d\gamma$ until a new tie occurs. Equation (4.12) shows that ties tend to persist. Thus there must be a mechanism for breaking ties so that progress can be made when $\dim(V) = p$ and $0 \notin \partial\Delta(\mathbf{x})$. Deleting a column from the reference corresponds either

(i) to a single residual moving away from a group, or

(ii) to a group splitting into two nontrivial subgroups, as progress is made in the direction determined by $d\mathbf{x}/d\gamma$. To be specific note that there is no restriction in assuming that it is the qth group that divides. We define

$$s_i^{(q)} = \begin{cases} 0, & \sigma_q(i) \in \bar{\sigma}_q, \\ 1, & \sigma_q(i) \in \bar{\sigma}_{q+1}, \end{cases} \qquad (4.13)$$

and assume that $\mathbf{a}_{\sigma_q(k+1)}$ becomes the origin of the new group (including the case of the trivial group containing a single element). After possible reordering we can write

$$[\bar{V}|0] = V - \mathbf{v}_k^{(q)}[0|\mathbf{s}^{(q)T}] \qquad (4.14)$$

and

$$\bar{\mathbf{g}} = \mathbf{g} + \zeta \mathbf{v}_k^{(q)} \qquad (4.15)$$

where ζ is the sum of the scores associated with the elements in the new group so that

$$\zeta = \sum_{j \in \bar{\sigma}_{q+1}} d_{\bar{\nu}(j)}. \qquad (4.16)$$

LEMMA 4.3. *If*

$$\mathbf{s}^{(q)T}\boldsymbol{\xi}^{(q)} > \sum_{j=1}^{\bar{m}_{q+1}} d_{\nu(\sigma_q(1))+m_q-j+1} \tag{4.17}$$

then the residuals in the new group increase faster in the direction determined by $d\bar{\mathbf{x}}/d\gamma$ *than the residuals remaining in the old group.*

If

$$\sum_{j=1}^{\bar{m}_{q+1}} d_{\nu(\sigma_q(1))+j-1} > \mathbf{s}^{(q)T}\boldsymbol{\xi}^{(q)} \tag{4.18}$$

then the residuals in the new group decrease relative to those remaining in the old group in the direction determined by $d\bar{\mathbf{x}}/d\gamma$.

Proof. The general approach is familiar. Set

$$\bar{Q}_2 = [\mathbf{y}|Q_2]$$

where $\mathbf{y} \in \text{span}\,(\kappa_i(Q_1),\ i = 1, 2, \cdots, m)$. Then

$$-\bar{Q}_2^T \frac{d\mathbf{x}}{d\gamma} = \bar{Q}_2^T(\bar{\mathbf{g}} - \zeta \mathbf{v}_k^{(q)}) + \bar{Q}_2^T \mathbf{v}_k^{(q)} \mathbf{s}^{(q)T}\boldsymbol{\xi}^{(q)},$$

and

$$-\mathbf{v}_k^{(q)T}\bar{Q}_2\bar{Q}_2^T\frac{d\mathbf{x}}{d\gamma} = (\mathbf{v}_k^{(q)T}\mathbf{y})\mathbf{e}_1^T\bar{Q}_2^T\frac{d\mathbf{x}}{d\gamma} = (\mathbf{v}_k^{(q)T}\mathbf{y})\mathbf{y}^T\frac{d\mathbf{x}}{d\gamma} = 0,$$

so that

$$0 = -\mathbf{v}_k^{(q)T}\frac{d\bar{\mathbf{x}}}{d\gamma} + (-\zeta + \mathbf{s}^{(q)T}\boldsymbol{\xi}^{(q)})\|\bar{Q}_2^T\mathbf{v}_k^{(q)}\|^2,$$

and the result follows from this as $\mathbf{v}_k^{(q)T}d\bar{\mathbf{x}}/d\gamma$ determines the relative movement of the groups as it gives the difference in the rates of change of the residuals corresponding to the group origins.

Remark 4.2. The conditions (4.17) and (4.18) should be compared with the inequalities (4.7). They show that $\boldsymbol{\xi}^{(q)}$ fails at least one of the tests characterizing an element of $\partial\Delta(\mathbf{x})$, and this shows immediately how to generate a downhill direction by deleting $\mathbf{v}_k^{(q)}$ from the reference matrix by (5.14). To implement this test let the index set ω_q point to the components of $\boldsymbol{\xi}^{(q)}$ sorted in increasing rank. Then check for the most violated of the inequalities,

$$\sum_{j=1}^{k} d_{\nu(\sigma_q(j))} \leqq \sum_{j=1}^{k} \xi_{\omega_q(j)}^{(q)},$$

and

$$\sum_{j=m_q-k+1}^{m_q} \xi_{\omega_q(j)}^{(q)} \leqq \sum_{j=m_q-k+1}^{m_q} d_{\nu(\sigma_q(1))+j}$$

for $k = 1, 2, \cdots, m_q$. The ordering defined by ω_q and the value of k giving the most violated inequality then fix $\mathbf{s}^{(q)}$.

The final lemma in this section considers the case of a tie when two groups (each of which may be the trivial group consisting of a single residual) come together and asks if this new group is *stable* in the sense that the resulting multiplier vector does not immediately violate the inequality which permits the group to interchange and

move apart. If the new group is not stable then the relatively expensive computation required to update the matrix factors can be avoided. Assume that V_q, V_{q+1} coalesce to form potentially $\bar{V}_q$ then

$$\bar{V}_q = [V_q | V_{q+1} | 0] + \hat{\mathbf{v}}\mathbf{s}^{(q)T}$$

where

$$\mathbf{s}^{(q)T} = [0 | \mathbf{e}^{(m_{q+1})T} | 1],$$

and

$$\hat{\mathbf{v}} = \mathbf{a}_{\sigma_{q+1}(1)} - \mathbf{a}_{\sigma_q(1)}.$$

LEMMA 4.4.

$$\mathbf{s}^{(q)T}\bar{\boldsymbol{\xi}}^{(q)} = \zeta + \hat{\mathbf{v}}^T \frac{d\mathbf{x}}{d\gamma} \Big/ \|P_2\hat{\mathbf{v}}\|^2 \tag{4.19}$$

where

$$\zeta = \sum_{j \in \sigma_{q+1}} d_{\nu(j)}.$$

Proof. We have

$$\bar{\mathbf{g}} = \mathbf{g} - \zeta\hat{\mathbf{v}} \quad \text{and} \quad \bar{Q}_1 = [Q_1 | \mathbf{y}],$$

where $\mathbf{y} = P_2\hat{\mathbf{v}}/\|P_2\hat{\mathbf{v}}\|$. Thus

$$\begin{aligned} 0 = \mathbf{y}^T \frac{d\bar{\mathbf{x}}}{d\gamma} &= \mathbf{y}^T\{\mathbf{g} - \zeta\hat{\mathbf{v}} + ([V|0] + \hat{\mathbf{v}}[0|\mathbf{s}^{(q)T}])\bar{\boldsymbol{\xi}}\} \\ &= \mathbf{y}^T\mathbf{g} - (\zeta - \mathbf{s}^{(q)T}\bar{\boldsymbol{\xi}}^{(q)})\mathbf{y}^T\hat{\mathbf{v}}. \end{aligned}$$

The desired result follows on noting that

$$\mathbf{y}^T\mathbf{g} = -\hat{\mathbf{v}}^T \frac{d\mathbf{x}}{d\gamma} \Big/ \|P_2\hat{\mathbf{v}}\| \quad \text{and} \quad \mathbf{y}^T\hat{\mathbf{v}} = \|P_2\hat{\mathbf{v}}\|.$$

COROLLARY 4.1. *The groups V_q, V_{q+1} can interchange their rankings provided either*

$$\nu(\sigma_{q+1}(1)) < \nu(\sigma_q(1)) \quad \textit{and} \quad \sum_{j=1}^{m_{q+1}+1} d_{\nu(\sigma_q(1))+m_q-j+1} < \mathbf{s}^{(q)T}\bar{\boldsymbol{\xi}}^{(q)} \tag{4.20}$$

or

$$\nu(\sigma_q(1)) < \nu(\sigma_{q+1}(1)) \quad \textit{and} \quad \sum_{j=1}^{m_{q+1}+1} d_{\nu(\sigma_q(1))+j-1} > \mathbf{s}^{(q)T}\bar{\boldsymbol{\xi}}^{(q)}. \tag{4.21}$$

Remark 4.3. The two options of most interest in the variable addition part of the algorithm are

(i) interchanging groups and continuing if either of the tests (4.20) or (4.21) are satisfied, and

(ii) line searching in the direction determined by $d\mathbf{x}/d\gamma$ until the direction determined by $d\mathbf{x}/d\gamma$ is no longer downhill.

Both options are available, but the basis for comparison has changed somewhat. In particular, it is no longer so clear that a fast algorithm of the kind described for Example 1.2 in the next section can be effective. The problem is that in the previous

examples it was necessary to record for each residual the length of step required to reduce it to zero, while here it is the distance to the next tie which is important; and this depends on the rate of change of the other residuals. For each residual, distances to first, second, and subsequent ties are potentially needed. Recording and sorting these quantities is a much more substantial problem computationally and storage wise, and it may be that the naive algorithm sketched in Algorithm 3.1 is preferable.

The algorithm described here uses the option of interchanging groups on the basis of (4.20) and (4.21).

ALGORITHM 4.1.

(i) Set initial $\mathbf{x}$, compute and rank initial $\mathbf{r}$, determine ties and factorize initial V.
(ii) Compute $\boldsymbol{\xi}$.
(iii) Test for deletion of column from reference matrix. If possible go to (v).
(iv) If $m = p$ then $\mathbf{x}$ is optimal, otherwise go to (vi).
(v) Delete column from V, update $\mathbf{g}$ and factors of V to correspond to new ranking.
(vi) Determine $d\mathbf{x}/d\gamma$.
(vii) Compute first tie in the descent step in the direction of $d\mathbf{x}/d\gamma$.
(viii) Increment $\mathbf{r}$, $\mathbf{x}$.
(ix) While new grouping is unstable update $\mathbf{g}$ and go to (vi).
(x) Update $\mathbf{g}$ and factors of V.
(xi) Go to (ii).

The justification of the algorithm follows by the standard argument.

THEOREM 4.1. *Provided Assumptions* 4.1 *and* 4.2 *are satisfied then Algorithm* 4.1 *terminates at a minimum of* $\Delta(\mathbf{x})$ *in a finite number of steps.*

5. Implementation. In this section three questions relating to the implementation of Algorithms 2.1–4.1 are discussed. These are

(i) development of a computational scheme which exploits the generic form of the equations determining $d\mathbf{x}/d\gamma$, $\boldsymbol{\xi}$,
(ii) development of an efficient line search procedure in the variable addition phase for piecewise linear problems and
(iii) devising suitable test problems for evaluating the algorithms.

It proves to be possible to organize the computation around transforming the matrix.

$$M = [A^T | I], \tag{5.1}$$

and updating the vector

$$\mathbf{w}^T = [\mathbf{r}^T | \mathbf{x}^T]. \tag{5.2}$$

Assume that at the current step we have available the matrix

$$\hat{M} = \hat{Q}^T [A^T | I] \tag{5.3}$$

where $\hat{Q}$ is the matrix of the orthogonal factorization in Lemma 2.3. Then we can generate as a linear combination of the columns of $\hat{M}$

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} = \hat{Q}^T \mathbf{g}. \tag{5.4}$$

As V is a linear function of the columns of M, $\hat{U}$ can be read off as the same function

of the columns of $\hat{M}$ and this permits calculation of

$$\boldsymbol{\xi} = -\hat{U}^{-1}\mathbf{c}_1. \tag{5.5}$$

Deletion of a column from the current reference requires modification of the matrix factors, and this can be carried out using the now familiar techniques described in [11]. The key point is that the operations are carried out on $\hat{M}$ not on an explicit form for $\hat{U}$. After updating $\mathbf{c}$ if necessary, the rates of change of $\mathbf{r}$ and $\mathbf{x}$ can be determined from

$$dr_i = \mathbf{a}_i^T \frac{d\mathbf{x}}{d\gamma} = -\mathbf{a}_i^T \hat{Q}_2 \mathbf{c}_2 = -[0, \mathbf{c}_2^T]\kappa_i(\hat{M}), \qquad i = 1, 2, \cdots, n, \tag{5.6}$$

and

$$dx_i = -\rho_i(\hat{Q}_2)\hat{Q}_2^T \mathbf{g} = -\rho_i(\hat{Q})\begin{bmatrix} 0 \\ \mathbf{c}_2 \end{bmatrix} = -[0, \mathbf{c}_2^T]\kappa_{n+i}(\hat{M}), \qquad i = 1, 2, \cdots, p. \tag{5.7}$$

Thus updating the solution quantities is achieved by

$$\mathbf{w}^T := \mathbf{w}^T + \varepsilon[d\mathbf{r}^T | d\mathbf{x}^T] := \mathbf{w}^T - \varepsilon[0, \mathbf{c}_2^T]\hat{M}.$$

Again the transformations to update the matrix factors are applied to $\hat{M}$ and not to an explicit form for $\hat{U}$ in adding the new column to the reference.

This development shows that a major part of the computation can be carried out in an elegant and economical manner by applying transformations to a tableau in the manner of the simplex algorithm of linear programming. However, in contrast to the Jordan elimination steps used in the simplex algorithm, the transformations used here have the superior stability properties of orthogonal transformations.

To implement the line search algorithm for Example 1.2 we note that there are two main parameters relevant. The first of these is the number of elements in Σ, and the second is the position of the optimum ε in the sorted list. In the experiments we carried out $|\Sigma|$ has turned out to be close to n initially, and usually $|\Sigma|$ was at least as large as $n/2$. More striking has been the position of the optimum ε. Initially it turns out to be near the head or middle of the sorted list of ε_k, but it settles down quite rapidly until by the time $|\sigma|$ is about $p/2$ it is always in the bottom few elements of the sorted list. On the basis of this evidence we conclude that

(i) there are always enough elements in Σ to make it necessary to take the task of finding the optimum ε seriously, and

(ii) it is worthwhile adopting different search strategies in the initial steps of the computation and in a second phase in which behavior is more predictable (and progress is slower).

In the first phase we follow [7] in adopting a strategy based on Hoare's partitioning algorithm. The steps are as follows.

(i) Set directional derivative Dx at current point, Σ.
(ii) Set partition bound ε (we use the median of the first, last, and middle elements of the current list).
(iii) Partition Σ into two sets $L = \{\varepsilon_k;\ \varepsilon_k < \varepsilon\}$ and $R = \{\varepsilon_k;\ \varepsilon_k \geqq \varepsilon\}$.
(iv) Compute $G = Dx + \Sigma(\eta_k + \zeta_k)|\mathbf{a}_k^T\, d\mathbf{x}/d\gamma|$, $\forall k \ni \varepsilon_k \in L$.
(v) If $G > 0$ then go to (vii).
(vi) $Dx := G$, $\Sigma := R$, go to (ii).
(vii) $\Sigma := L$, go to (ii).

The process terminates when the number of elements in Σ is sufficiently small to permit the optimum ε to be read off. In the second phase there seems no point in undertaking anything more elaborate than a few steps of a standard comparison sort. Note that in the first phase we might expect to find the minimum in about $m \log_2 m$ comparisons, where m is the number of elements in the initial list, provided the successive choices of the partition bound are not unfortunate. Thus the changeover from phase 1 to phase 2 should occur as soon as the optimum ε appears consistently in the bottom $\log_2 m$ elements of the sorted list. The heuristic $|\sigma| \geqq p/2$ has proved satisfactory.

For Examples 1.1, 1.2 testing has been largely based on sequences of randomly generated test problems. The method used has followed closely that suggested by Bartels [3] for generating LP problems with known solution and modified subsequently by Bartels and Joe [6] to generate MAX problems. We describe here the corresponding method used to generate piecewise linear test problems. It uses two random number generators: a uniform generator RND, and a second generator FNRnd which will be discussed below. The main steps are as follows.

(i) Set up σ^c by using $J = \text{INT}(N * \text{RND} + 1)$ to generate $N - P$ distinct integers in $[1, N]$.
(ii) For $J \in \sigma^c$, $R(J) = \text{FNRnd}$, $Ir(J) = \psi(\eta_j, \zeta_j, R(J))/R(J)$. (In LP problems this must be modified to force $R(J) \geqq 0$.)
(iii) For $I = 1, 2, \cdots, P$ set $Y(I) = -\eta_{\sigma(I)} + (\zeta_{\sigma(I)}) + \eta_{\sigma(I)}) * \text{RND}$, $X(I) = \text{FNRnd}$, $C(I) = \text{FNRnd}$ and for $J = 1, 2, \cdots, N$ $A(I, J) = \text{FNRnd}$.
(iv) Let K be the smallest index in σ^c. Determine $\kappa_K(A)$ so that the optimality condition is satisfied. This gives $-Ir(K) * \kappa_K(A) = \sum_{\sigma^c/\{K\}} Ir(I) * \kappa_I(A) + \sum_{\sigma} Y(I) * \kappa_I(A) + \chi * C$. (If $\chi \neq 0$ and not too small this equation can be used to determine C. In the LP case put $\chi = 1$ to obtain the Kuhn–Tucker conditions).
(v) Generate the right-hand side for $I = 1, 2, \cdots, N$ $B(I) = \sum_{J=1}^{P} A(J, I) * X(J) - R(I)$.

Two different forms have been used for FNRnd:

(a) $\text{FNRnd} = (\text{RND} - .5) * S$

where S is a scale factor, and

(b) $\text{FNRnd} = (1 - 2\alpha)/(\alpha - 1) + (1 - \text{RND})^{-1/\alpha}$ for $\alpha = 1.2$.

The uniform generator appears to produce very well behaved problems on which the algorithms work very well and distinguishing between options is often difficult. The second generator is based on a Pareto distribution with a very long tail. It produces a wide spread of numbers, and experience indicates that the resulting problems offer a more serious challenge.

6. Numerical experiments.

(i) *Example.* MAX. Numerical results obtained using algorithm 2.1 are given in Table 6.1. The problems were generated using a procedure very similar to that employed in the piecewise linear generator sketched in § 5, and the figures quoted are for the average of five runs. The Pareto generator was used in all cases. The results appear excellent. Two interesting features are that for a wide range of parameters the number of iterations (number of addition steps) appears to be approximately $2p$ independent of n, and in every case $|\pi| < p + 1$ except in the last steps of the iteration. Here deletion from the reference when $|\pi| < p + 1$ appears a good strategy. The strategy of deletion only when $|\pi| = p + 1$ also performed well. It gave very similar results to

TABLE 6.1
Average number of iterations in the MAX *problem.*

p \ n	5	10	20	40	80	160
2	3.0	3.6	5.6	4	4.2	3.6
5		9.6	11.6	10	11.4	12.4
10			22.6	20.8	21.8	20.8
20				40	53.2	47.8

those in Table 6.1 for small values of n and p but required more iterations for larger values (19% more when $n = 160$, $p = 20$ in the worst case).

However, a note of caution must be added. Both the strategies performed badly on the problem of fitting polynomials of low degree to e^x at 50 equispaced points in $[0, 1]$. The difficulty is that most of the time the descent step terminates at a point adjacent to a point in the current reference which it then replaces in the next deletion. This results in very slow progress towards the optimum. What is lacking here is some way for making the sizeable corrections that the line search facility provides for the other examples. It appears that ascent algorithms [1] are favoured in this case when the problem data varies systematically and slowly. It also shows that some discretion is needed in interpreting the results of studies based on random data.

(ii) *Example. Piecewise linear problems.* Calculations have been carried out with Algorithm 3.1 in both LAD and penalty function special cases using both random number generators. In the LAD case the uniform random numbers did not provide much discrimination between the possible options. However, with the Pareto data the best algorithm was provided by the option choices corresponding to the method of Bartels, Conn and Sinclair [5]. It was disappointing that deletion from the reference when $|\sigma| < p$ did not prove a good strategy except in the apparently very regular problems produced by the uniform generators. It appeared that weakening the constraints on $d\mathbf{x}/d\gamma$ leads to a problem similar to hemstitching suggesting that the minimum in the Pareto case is frequently in a narrow valley. This strategy performed better in all the other problem classes considered.

For the penalty function problem we give numerical results of two kinds. We compare the performance of Algorithm 3.1 on data produced by both the uniform and Pareto distributions in Tables 6.2 and 6.3, and for reference we give in Tables 6.4 and 6.5 results obtained using the reduced gradient form of the penalty function algorithm described in [16]. Comparisons suggest that this latter algorithm performs in a similar manner to the penalty algorithm of Bartels and the careful simplex implementation of Hanson and Wisniewski [3] (although the tests were done using uniform random number generation only) so that it provides a useful means of assessing the new method. In all cases $\chi_0 = .025$, and this proved small enough. The results show clearly the way in which the difficulty of the problems changes when the random number generator is changed from uniform to Pareto, and also suggest a definite edge for the new algorithm. In this case both numbers of iterations and the cumulative total of the line search steps are given (again averaged over five replications). Other options, for example terminating the line search if proceeding would make a constraint infeasible, have been tested with generally negative conclusions. However, for small

TABLE 6.2
Algorithm 3.1, *uniform data.*

p \ n	10	20	40	80	160
5	8 15.6	7.2 18.6	7.6 28.8	8.2 5.13	7 73.6
10		17.4 48	18.6 65.4	15 85.4	15.4 121.8
20			44.4 157.4	49.2 240.1	

TABLE 6.3
Algorithm 3.1, *Pareto data.*

p \ n	10	20	40	80	160
5	6.8 15.4	8.4 30	8.4 42	8.8 90.4	8.8 156.4
10		17.8 44	17.2 65.6	22.6 119	29.2 239.2
20			37.8 119.4	50.2 208.6	56.2 324

TABLE 6.4
Penalty algorithm, uniform data.

p \ n	10	20	40	80	160
5	5.8 14.4	7.8 18.4	11.8 41.2	9.2 57	14.2 108
10		15.2 32.4	21.6 71.8	24 140.2	30.6 290.2
20			32.8 107.2	49.4 228	

TABLE 6.5
Penalty algorithm, Pareto data.

p \ n	10	20	40	80	160
5	7 15.2	10 37.2	12.8 59	13.4 155.4	14.8 326.4
10		15.4 44	22 109	27.6 216.2	34.4 447.2
20			38 109.6	53.2 267.6	63 549.8

p and n a mixed strategy which permitted column deletion early in the computation (say for the first $p/2$ iterations) but then forced the reference to build up to p columns proved excellent. This performance was not maintained for larger p (say $p = 20$).

(iii) *Example.* RR. For this example facilities for generating random problem data have not been implemented, and testing of the algorithm has been restricted to contrived examples designed to test particular features (such as a group splitting into two proper subgroups to generate a downhill direction), and to a standard problem used for experimenting with methods for robust estimation. The program details surrounding the core algorithm prove more complex in this case, but the general performance measured in terms of iterations and line search steps appears about the same.

Calculations have been carried out on an HP 9845B computer programmed in BASIC.

Acknowledgments. The authors wish to thank Bill Steiger for interesting them in the LAD problem, for providing the Pareto random number generator, and for pointing out the importance of doing the line search properly. Ian Barrodale, Richard Bartels, and Carl Lemke helped develop the background by asking the right questions and by providing valuable references. David Anderson took a lively interest and helped with supporting calculations. Alan Miller suggested the rank regression problem which plays such an important part in filling in the general picture. The referees made a number of valuable comments and suggestions.

REFERENCES

[1] D. H. ANDERSON AND M. R. OSBORNE, *Discrete, linear approximation problems in polyhedral norms*, Numer. Math., 26 (1976), pp. 179–189.

[2] 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.

[3] R. BARTELS, *A penalty linear programming method using reduced gradient basis exchange techniques*, Linear Algebra and Appl., 29 (1980), pp. 17–32.

[4] R. BARTELS, A. R. CONN AND C. CHARALAMBOUS, *On Cline's direct method for solving over determined linear systems in the l_∞ sense*, SIAM J. Numer. Anal., 15 (1978), pp. 255–270.

[5] 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.

[6] R. BARTELS AND B. JOE, *On generating l_∞ test problems*, to be published.

[7] P. BLOOMFIELD AND W. STEIGER, *Least absolute derivation curve fitting*, this Journal, 1 (1981), pp. 290–301.

[8] D. I. CLARK, *Finite algorithms for linear optimisation problems*, PhD thesis, Australian National University, Canberra, 1981.

[9] A. K. CLINE, *A descent method for the uniform solution to overdetermined systems for linear equations*, SIAM J. Numer. Anal., 13 (1976), pp. 293–309.

[10] A. R. CONN, *Linear programming via a nondifferentiable penalty function*, SIAM J. Numer. Anal., 13 (1976), pp. 145–154.

[11] P. E. GILL, G. H. GOLUB, W. MURRAY AND M. A. SAUNDERS, *Methods for modifying matrix factorizations*, Math. Comp., 29 (1974), pp. 505–535.

[12] L. A. JAECKEL, *Estimating regression coefficients by minimizing the dispersion of the residuals*, Ann. Math. Statist., 43 (1972), pp. 1449–1458.

[13] K. JITTORNTRUM AND M. R. OSBORNE, *Strong uniqueness and second order convergence in nonlinear discrete approximation*, Numer. Math., 34 (1980), pp. 439–455.

[14] B. JOE AND R. BARTELS, *An exact penalty method for constrained discrete, linear l_∞ data fitting*, this Journal, 4 (1983), pp. 69–84.

[15] J. J. MOREAU, *Proximité et dualité dans un espace Hilbertien*, Bull. Soc. Math. France, 93 (1965), pp. 273–299.

[16] M. R. OSBORNE, *Finite algorithms in optimization and data analysis*, in preparation.
[17] R. T. ROCKAFELLAR, *Convex Analysis*, Princeton Univ. Press, Princeton, NJ, 1970.
[18] ———, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898.
[19] S. R. SMITH, *A multivariate Remes algorithm for strongly unique best approximations*, J. Approx. Theory, to appear.

SIAM J. SCI. STAT. COMPUT.
Vol. 4, No. 4, December 1983

0196-5204/83/0404-0016 $01.25/0

ERRATUM: A GENERALIZED EIGENVALUE APPROACH FOR SOLVING RICCATI EQUATIONS*

P. VAN DOOREN†

In the above paper, some errors appeared in the treatment of the spectral factorization problems for both the continuous-time and the discrete-time case.

The corrections to be performed are the following. Formulas (46), (47), (50) and (51) have to be replaced by their "dual" forms which appear below:

$$Z(s)+Z'(-s)=R'(-s)\cdot R(s), \tag{46}$$

$$\begin{aligned} C'(D+D')^{-1}C+P[A-B(D+D')^{-1}C] \\ +[A-B(D+D')^{-1}C]'P+PB(D+D')^{-1}B'P=0, \end{aligned} \tag{47}$$

$$Z(z)+Z'(z^{-1})=R'(z^{-1})\cdot R(z), \tag{50}$$

$$P=F'PF+(H'-F'PG)(J+J'-G'PG)^{-1}(H-G'PF). \tag{51}$$

Acknowledgment. The author is grateful to Bert van Gent who pointed out these errors.

* This Journal, 2 (1981), pp. 121–135.

‡ Philips Research Laboratory, B-1170 Brussels, Belgium.