SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0001 $01.00/0

VORTEX MODELS AND BOUNDARY LAYER INSTABILITY*

ALEXANDRE JOEL CHORIN†

Abstract. Random vortex methods are applied to the analysis of boundary layer instability in two and three space dimensions. A thorough discussion of boundary conditions is given. In two dimensions, the results are in good agreement with known facts. In three dimensions, a new version of the method is introduced, in which the computational elements are vortex segments. The numerical results afford new insight into the effects of the third dimension on the stability of a boundary layer over a flat plate.

Key words. *vortex, boundary layers, random walk, three-dimensional instability*

Introduction. The random vortex method as described in Chorin [7] is intended for the approximation of flows at high Reynolds number R. Its main features are as follows: (i) the nonlinear terms in the Navier–Stokes equation are taken into account by a detailed analysis of the inviscid interactions between vortices of small but finite core ("vortex blobs"), (ii) viscous diffusion is taken into account by adding to the motion of the vortices a small random Gaussian component of appropriate variance, and (iii) no-slip boundary conditions are approximated by a vorticity creation algorithm. Fuller details are given below. Developments, modifications, and applications of the method can be found e.g. in Ashurst [1], Chorin [10], [11], Leonard [26], [27], McCracken and Peskin [30], Shestakov [36]. Theoretical analysis can be found in Hald [18], Hald and Del Prete [19], and Chorin et al. [12].

This grid-free method is suitable for the analysis of flow at high Reynolds number because it has no obvious intrinsic source of diffusion. Most approximation methods solve equations which are close to the equations one wants to solve; the difference consists of higher order terms multiplied by small parameters. This is also the form of the diffusion term, and as a result, in most methods, the effects of a small R^{-1} are dominated by numerical effects and the physics of high Reynolds number flow are suppressed. In vortex methods, the misrepresentation of the higher harmonics which occurs in the usual discretization methods (which usually has a diffusive effect among other effects) is replaced by the misrepresentation of the interaction of neighboring vortices (an essentially inviscid phenomenon which is a source of error, but not of diffusive error). In the absence of the nonlinear terms, the diffusion is approximated on the average exactly. Thus one may hope that the results of the calculation approximate the flow at whatever Reynolds number was intended, albeit with a statistical error, rather than at some other lower Reynolds number intrinsic to the algorithm. A good guess at the solution of the problem one wants to solve is better than an unambiguous solution of the wrong problem.

The method produces a flow field which is random. The error in the calculation is the sum of two parts: the expected value of the computed solution differs from the true solution, and any realization of the computed solution (or more accurately, any functional thereof) differs from the expected value by a random amount which can be estimated by its standard deviation (see e.g., Lamperti, [25]). The expressions for these quantities will be given below, when the appropriate notation will be available.

In the present paper we apply random vortex methods to the analysis of the boundary layer over a flat plate in two and three space dimensions. The calculations

* Received by the editors May 15, 1979. This work was supported in part by the Engineering, Mathematical, and Geosciences Division of the U.S. Department of Energy through the Lawrence Berkeley Laboratory.

† Department of Mathematics, University of California, Berkeley, California 94720.

have two main objects. In the two dimensional case we shall show that the vortex method exhibits a physical instability at an appropriate Reynolds number. The ability to do so is of course a basic requirement for any method which claims to have some use at high Reynolds number. The specific problem we apply our method to has a simplifying feature, inasmuch as the location of the sharp gradients is known in advance to be near the wall, and thus the equations of motion can be solved in two dimensions by finite difference or other non-statistical methods in appropriately scaled variables. The interesting fact about our calculation is that it does not require such preliminary scaling of the variables, i.e., the random walk can be relied upon to create the appropriate diffusive length scale.

The second main goal of our calculation is to use the method to investigate the much harder problem of boundary layer instability in three dimensions, and in particular, two of the striking features of its solution: The formation of streamwise vortices and the creation of active spots. The three dimensional calculation requires a generalization of our method, and both the two dimensional and three dimensional problems afford the opportunity to use an improved algorithm for imposing the boundary conditions accurately.

In the next four sections we present the calculation in two dimensions. In later sections we present the three dimensional calculations.

The physical problem in two dimensions. Consider a semi-infinite flat plate placed on the positive half-axis, with an incompressible fluid of density 1 occupying the half space $y \geqq 0$. At time $t<0$ the fluid is at rest. At $t=0$, the fluid is impulsively set into motion with velocity U_∞. The flow is described by the Navier–Stokes equations,

$$\partial_t \xi + (\mathbf{u}\cdot\nabla)\xi = R^{-1}\Delta\xi, \tag{1a}$$

$$\Delta\psi = -\xi, \tag{1b}$$

$$u = \partial_y\psi, \qquad v = -\partial_x\psi, \tag{1c}$$

where $\mathbf{u}=(u, v)$ is the velocity vector, $\mathbf{r}=(x, y)$ is the position vector, ξ is the vorticity, ψ is the stream function, $\Delta \equiv \nabla^2$ is the Laplace operator, and R is the Reynolds number, $R = U_\infty L/\nu$, where L is a length scale typical of the flow. The boundary conditions are

$$\mathbf{u} = (U_\infty, 0) \quad \text{at } y=\infty,\ t>0, \tag{1d}$$

$$u = v = 0 \quad \text{for } y=0,\ x>0, \tag{1e}$$

$$\frac{\partial v}{\partial y} = 0 \quad \text{for } y=0,\ x<0. \tag{1f}$$

Initially, $\mathbf{u}=(U_\infty, 0)$ everywhere.

If R is large, the Prandtl boundary layer equations should provide a reasonable description of the flow near the plate and away from the leading edge. These equations can be written in the form [Schlichting [35], Chorin [10], [11]],

$$\partial_t \xi + (\mathbf{u}\cdot\nabla)\xi = \nu\partial_y^2\xi, \tag{2a}$$

$$\xi = -\partial_y u, \tag{2b}$$

$$\partial_x u + \partial_y v = 0. \tag{2c}$$

where ξ, u, v, x, y have the same meaning as in equations (1), and ν is the viscosity. If $U_\infty = 1$ and $L=1$, $R=\nu^{-1}$. The boundary conditions for equations (2) are: $u=U_\infty$ for $y=\infty$, $\mathbf{u}=0$ for $y=0$. Equations (2) have a stationary solution, the Blasius solution, which is a function of the similarity variable $\mu = y/\sqrt{x\nu}$. Let the displacement thickness δ

be defined by

$$\delta = \int_0^\infty (1-u/U_\infty)dy;$$

the corresponding Reynolds number is $R_\delta = U_\infty \delta/\nu$. In Blasius flow, $\delta = 1.72\sqrt{\nu x}$, and $R_\delta = 1.72\sqrt{x/\nu}$, where it is assumed that $U_\infty = 1$. δ and R_δ are increasing functions of x. For $R_\delta \geqq R_{\delta c}$ the Blasius solution is unstable to infinitesimal perturbations which satisfy equations (1) (see Lin [29]); $R_{\delta c} \simeq 520$, (See Jordinson [21]). These unstable modes are the Tollmien–Schlichting waves. The vortex interpretation of the waves is as follows: The boundary layer is a region of distributed vorticity imbedded in a shear flow. Vorticity imbedded in a shear tends to become organized into coherent macroscopic structures ("negative temperature states", "local equilibria", see Onsager [32], Chorin [8]). This tendency is counteracted by the diffusive effects. The latter become weaker as x increases, since the vorticity gradients decrease as the layer spreads. Far enough downstream (i.e., for R_δ large enough), the tendency towards coherence can overcome the diffusive effects; the Tollmien–Schlichting waves can be viewed as a weak train of organized vortex structures.

The value of $R_{\delta c}$ given above has to be lowered if the unperturbed flow is treated as a nonparallel flow and if edge effects are taken into account (Townsend [37]). More importantly, the boundary layer is unstable to perturbations of a finite amplitude for values of R_δ smaller than $R_{\delta c}$ (for analysis of similar situations, see Eckhaus [14], Meksyn and Stuart [31]). A survey of finite amplitude stability theory for the flat plate problem is given by Roshotko [33]). The boundary layer becomes more unstable if the outside flow is turbulent or contains vortical structures (see Schlichting [35], Rogler and Reshotko [33]). Since our calculation will by its very nature contain finite amplitude perturbations, vortices, a substantial amount of noise, and edge effects, the appropriate value of R_δ which separates stable from unstable regimes is unclear. Presumably, there exists a value $R'_{\delta c}$ such that for $R_\delta \leqq R'_{\delta c}$ all perturbations decay; the best guess of $R'_{\delta c}$ we can obtain by looking at the references above is $R'_{\delta c} \simeq 300$, with a substantial margin of error. Cebeci and Smith [4] suggest a value $R'_{\delta c} \simeq 320$.

For $R_\delta \geqq R_{\delta c}$, the perturbations can grow, but I found little information as to what they do in two dimensions; presumably they grow and reach some finite amplitude equilibrium; this is the typical situation in other two-dimensional stability problems, for example in the thermal convection problem (see e.g. Chorin [6]). All experimental studies I know deal with the more important and more realistic three dimensional problem which will be discussed further below.

The numerical methods in two dimensions. Consider first the Navier–Stokes equations (1) in the whole plane. Assume that $\xi = \sum_j \xi_j$, where the ξ_i are functions of small support (ξ_i is a "blob"). Let $\psi = \sum \psi_j$, where $\Delta\psi_j = -\xi_j$. (If we had $\xi_j = \kappa_j \delta(\mathbf{r}-\mathbf{r}_j)$, $\kappa_j =$ constant, we would have concluded that $\psi_j = -(\kappa_j/2\pi)\log|\mathbf{r}-\mathbf{r}_j|$.) For ξ_j smooth but of small support, let $\kappa_j \equiv \int \xi_j \, dxdy$, and we must have

$$\lim_{|\mathbf{r}|\to\infty} \frac{\psi_j}{(1/2\pi)\log|\mathbf{r}-\mathbf{r}_j|} = -\kappa_j.$$

For $|\mathbf{r}-\mathbf{r}_j|$ small, ψ_j differs from $(\kappa_j/2\pi)\log|\mathbf{r}-\mathbf{r}_j|$ (or else it would introduce undesirable unbounded velocities, see Chorin [7], Hald [18]). We set

$$\psi_j(\mathbf{r}) = \begin{cases} \dfrac{\kappa_j}{2\pi}\log|\mathbf{r}-\mathbf{r}_j|, & |\mathbf{r}| \geqq \sigma, \quad \text{(3a)} \\ \dfrac{\kappa_j}{2\pi}\dfrac{|\mathbf{r}|}{\sigma} + \text{const.}, & |\mathbf{r}| < \sigma. \quad \text{(3b)} \end{cases}$$

This is the form introduced in Chorin [7]; it differs from the forms described by Hald in [18] for reasons which will become apparent below. Clearly $\xi_j = -\Delta\psi_j$ is of small support. σ is a cut-off which remains to be determined.

Equations (1) state that the vorticity moves with the velocity field which it induces, i.e., let $\mathbf{u}_j = (u_j, v_j)$ be the velocity field induced by the jth blob, and let $\mathbf{r}_i = (x_i, y_i)$ be the center of the ith blob, then

$$\frac{d\mathbf{r}_i}{dt} = \sum_{j \neq i} \mathbf{u}_j, \qquad (\mathbf{u}_j \text{ evaluated at } r_i).$$

This equation can be approximated by

$$\mathbf{r}_j^{n+1} = \mathbf{r}_i^n + k \sum_{j \neq i} \mathbf{u}_j \tag{4}$$

where k is a time step and $\mathbf{r}_i^n \equiv \mathbf{r}_i(nk)$. Hald [19] has shown that a higher order method is indeed more accurate but we shall use (4) for the sake of simplicity.

The heat equation is well known to be solvable by a random walk algorithm (see Chorin [7]). As a result, equations (1) can be solved by moving the blobs according to the law

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + k \sum_{j \neq i} \mathbf{u}_j + \boldsymbol{\eta} \tag{5}$$

where $\boldsymbol{\eta} = (\eta_1, \eta_2)$, η_1, η_2 independent Gaussian random variables with mean 0 and variance $2k/R$.

Suppose we wish to solve equations (1) in a domain D with boundary ∂D. The normal boundary condition $\mathbf{u} \cdot \mathbf{n} = 0$ on ∂D, $\mathbf{n}$ normal to ∂D, can be readily taken into account by solving $\Delta\psi = -\xi$ subject to the appropriate boundary condition, with the help of potential theory. In the case of flow over a flat plate, the method of images will do the job. The no-slip boundary condition $\mathbf{u} \cdot \mathbf{s} = 0$, $\mathbf{s}$ tangent to ∂D, can be imposed through the creation of the appropriate amount of vorticity: Let u_0 be the velocity component tangent to the wall created by the algorithm as described so far, and suppose $u_0 \neq 0$. The no-slip condition and the viscosity will create a boundary layer in which the total vorticity per unit length is

$$\int_{\text{wall}}^{\text{interior}} \xi \, dn = \int \frac{\partial u}{\partial n} \, dn = u_0.$$

In the algorithm presented in [7], we reproduced this effect numerically by creating a vortex sheet of strength u_0 at the wall, dividing its vorticity among blobs, and allowing these blobs to participate in the subsequent motion of the blobs according to the laws (5). If a blob is created at every piece of boundary of length h, its intensity is

$$\kappa = u_0 h. \tag{6}$$

If a blob inside the fluid happens to cross the boundary, it is removed. It should be apparent that the amount of vorticity created at each time step depends on the cut-off σ. If σ is small, the backwash of the vortex may be large, and a vortex whose center is near the boundary will create a vortex whose intensity will have an opposite sign, etc. If σ is large, the backwash of a newly created vortex may not be sufficient to annihilate u_0, and more vortices will be created, all of the same sign. Presumably, on the average the total amount of vorticity is independent of σ. The algorithm in this form is not accurate (see Chorin et al. [12]). This lack of accuracy as well as the desire to reduce the amount of

computational labor have led to the formulation of the vortex sheet algorithm with which one can solve the boundary layer equations (2) (Chorin [10]). The computational elements are segments of a vortex sheet. Let u_0 be the velocity component parallel to the wall. A segment S of a vortex sheet is a segment of a straight line, of length h, parallel to the wall, such that u above S differs from u below S by an amount ξ; ("above" means "further from the wall"), $u_{\text{above}} - u_{\text{below}} = \xi$. ξ is the intensity of the sheet.

Consider a collection of N segments S_i, with intensities ξ_i, $i = 1, \cdots, N$. Let the center of S_i be $\mathbf{r}_i = (x_i, y_i)$. To describe their motion, one begins with equations (2b) and (2c). Equation (2b) can be integrated in the form

$$u(x, y) = U_\infty - \int_y^\infty \xi(x, \alpha)\, d\alpha, \tag{7a}$$

where U_∞ is the velocity at infinity seen by the layer. Equation (2c) yields

$$v(x, y) = -\partial_x \int_0^y u(x, \alpha)\, d\alpha \tag{7b}$$

Equations (7a) and (7b) allow one to determine u, v if $\xi = \xi(x, y)$ is given. One can visualize each sheet as casting a shadow between itself and the wall. The darker the shadow, the smaller u becomes. Whatever fluid enters a shadow region from the left and cannot leave on the right must leave upwards. From equations (7) one can derive the following expression for $\mathbf{u}_i = (u_i, v_i)$ at the center $\mathbf{r}_i$ of the ith sheet

$$u_i = U_\infty - \tfrac{1}{2}\xi_i - \sum_j \xi_j d_j, \tag{8a}$$

where $d_j = 1 - |x_i - x_j|/h$ is a smoothing function, and the summation is over all S_j for which $0 \leqq d_j \leqq 1$ and $y_j \geqq y_i$. This is of course a small subset of all the sheets; only the sheets which lie in a narrow vertical strip around u_i affect u_i. Similarly,

$$v_i = -(I_+ - I_-)/h, \tag{8b}$$

where

$$I_\pm = U_\infty - \textstyle\sum_\pm \xi_j d_j^\pm y_j^*, \tag{8c}$$

$$d_j^\pm = 1 - |x_i \pm h/2 - x_j|/h, \tag{8d}$$

$$y_j^* = \min(y_i, y_j). \tag{8e}$$

The sum $\sum_+$ (resp. $\sum_-$) is over all S_i such that $d_j^+ \leqq 1$ (resp. $d_j^- \leqq 1$). The motion of the sheets is then given by

$$x_i^{n+1} = x_i^n + k u_i, \tag{9a}$$

$$y_i^{n+1} = y_i^n + k v_i + \eta. \tag{9b}$$

These formulas are analogous to (4); η is a Gaussian random variable with mean 0 and variance $2\nu k$; it appears only in the y component because equations (2) take into account diffusion in the y direction only.

This vortex sheet algorithm generates a velocity field $\mathbf{u} = (u, v)$ which satisfies the boundary condition $u = U_\infty$ at $y = \infty$, $v = 0$ at $y = 0$. The no-slip boundary condition $u = 0$ at $y = 0$ can be satisfied by the following vorticity generation procedure (see [10]): Continue the flow from $y > 0$ to $y < 0$ by antisymmetry, i.e., $\mathbf{u}(x, -y) = -\mathbf{u}(x, y)$. Since $\xi = -\partial u/\partial y$, and both u and y change signs, we have $\xi(x, -y) = \xi(x, y)$; if $u(x, 0) = u_0 \neq 0$, we also have a vortex sheet of strength $2u_0$ at the wall. This sheet can be divided into

segments and allowed to participate in the subsequent motion. The antisymmetry can be imposed by reflecting any sheet which crosses the wall back into the fluid. One can require that all the sheets created satisfy the requirement $|\xi_i| \leqq \xi_{\max}$, where $\xi_{\max}$ is some reasonably small quantity. To do this, one may have to create more than one sheet at any one point at any given time step. The sheet method can be modified to make it more efficient and to reduce the variance of the results (see [10]). The interaction of the sheets is not singular and no cut-off is needed. The amount of vorticity created at the wall is unambiguous, and the cost of the calculation is small. This is of course balanced by the fact that the Prandtl equations are not uniformly valid approximations to the Navier–Stokes equations, and the transition from sheets to blobs involves in general a decision process which in turn is not unambiguous.

Note that the antisymmetry just described cannot be used directly with the vortex blob method. Indeed, if $\mathbf{u}(x, -y) = -\mathbf{u}(x, y)$, it does not follow in general that

$$\xi(x, -y) \equiv \left(-\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \quad \text{at } (x, -y) = \xi(x, y),$$

since x does not change sign. Thus, to impose the boundary conditions accurately on the blob method we shall have to use the sheet method as a transition near the wall, see below.

The version of the sheet method that we shall use is almost identical to the one described in [10] and documented in detail in Cheer [5]; this includes tagging and variance reduction techniques. The only difference is the following: In the earlier program, sheets were created at the wall, and on the average, half of them disappeared at each step. In the present program, we make exactly half of them disappear at each step and this reduces the total number of sheets retained. This is accomplished as follows: At each point at which sheets are created, their intensity is adjusted so that their number is even. A rejection technique (Handscomb and Hammersley [20]) is then used to ensure that any two successive η's used at the wall will have differing signs. This rejection technique can be used only at the wall, or else it would destroy the independence of the successive η's in the interior and thus fail to describe the diffusion process correctly.

The sheets and the blobs are objects of a very similar nature; they are determined by the same parameters, position and intensity. A computational element (x_i, y_i, ξ_i) can be treated as either a sheet or a blob, depending on the circumstances. A sheet of negative intensity casts a shadow which slows the fluid under it; by the equation of continuity, this creates an upward flow to the left and a downward flow to the right, just as if the sheet were a vortex. The circulation around a sheet of intensity ξ is ξh, and if the sheet becomes a blob, the latter's intensity must be $\kappa = \xi h$, in agreement with equation (5).

These facts can be used to create a transition between the blobs and the wall. Pick a length l such that a blob has a small probability of jumping more than $2l$ in one random jump, i.e. l a multiple of the standard deviation $\sqrt{2k/R}$ of η. Any vortex which finds itself less than l from the boundary (inside or outside) becomes a sheet and is reflected accordingly, and also taken into account accordingly when u_0 is computed. If a blob is further outside the domain than l it is removed (presumably this happens rarely). If a sheet is inside the domain and its distance from the boundary is more than l it becomes a blob again.

The cut-off σ remains to be determined. A natural condition to impose is the following: consider a collection of blobs. As they approach each other and the boundary, their interaction should converge to the interaction of the corresponding

sheets. Consider a sheet of intensity ξ at (X, Y), as well as vortex of intensity ξh at (X, Y), together with its image vortex at $(X, -Y)$ required to satisfy the boundary conditions (the sheets need no images). If $\sigma = h/\pi$, the velocity fields induced along the vertical line $x = X$ are identical (Fig. 1). The lateral effects will tend to each other as $y \to 0$. Thus, if $\sigma = h/\pi$, the interaction of the blobs will approximate the interaction of the sheets when the blobs approach the boundary. Hence $\sigma = h/\pi$ is a natural choice for σ. Note that the form (3) of ψ ensures that for $|\mathbf{r}| \leqq \sigma$ the magnitude of $\mathbf{u}$ is constant. This is the reason (3) is used. Remarks: (i) the value of σ is twice the value used in [7]. (ii) The choice of σ has the greatest effect near the wall, and thus it is natural to determine the value of σ by considering what happens near the wall. (iii) Our value of σ is large compared to the mean distance between blobs which is of order $R^{-1/2}$; this is in agreement with the requirements in Hald's proof. In summary the computational elements should be viewed as sheets near the wall, and as blobs far from the wall.

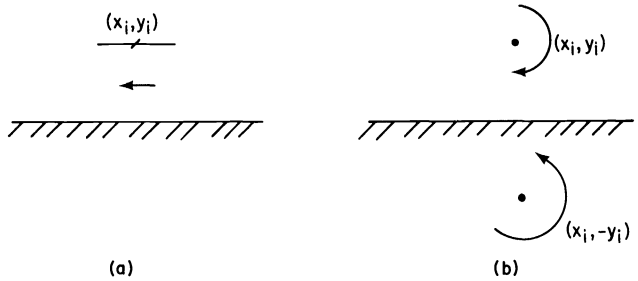


FIG. 1. *Sheets and vortices near a wall.*

A heuristic error analysis in [7] provides error estimates for the expected value of the velocity field produced by our methods in the form: error $= O(k) + O(R^{-1/2})$, $R =$ Reynolds number based on a velocity and length scales typical of the flow away from the wall. Hald's analysis of the inviscid case suggests that this could be reduced to $O(k^2) + O(R^{-1/2})$ if the time integration were carried out more accurately. The standard deviation of a smooth functional of the velocity should be $O(R^{-1/2})$.

Application of the numerical method in two dimensions. In this section we describe the application of the vortex methods to the specific problem at hand. Note that if the sheet method is used by itself on the flat plate problem and if it converges in the mean to a stationary solution of the Prandtl equations (2); that solution is a function of the similarity variable μ only; more specifically, if two computer runs are made, with the same numerical parameters k, h, $\xi_{\max}$, etc., the same sequence of random numbers, and the same impulsive initial conditions, but with two distinct values of ν, the resulting computed solutions will be identical for equal values of $y/\sqrt{x\nu}$ and x. These facts are straightforward consequences of equations (8) (see Chorin and Marsden [11]). As a consequence, the instability of the boundary layer cannot be seen with the sheet method, and our main tool will be the blob method. We shall use the sheet method for the following limited purposes: (i) to provide a rational argument in favor of the value $\sigma = h/\pi$; (ii) as a vorticity creation algorithm, (iii) as a way of imposing an approximate Blasius flow before allowing unstable modes to grow; and (iv) as a diagnostic tool.

The number of vorticity elements required to describe the flow is large, since enough of them must be included to resolve the Tollmien–Schlichting waves, and those

have a short wave length. From linear stability theory (see e.g. [29], [21]) one finds that the wave number of unstable Tollmien–Schlichting waves is between roughly $0.1/\delta$ and $0.4/\delta$ for moderate values of R_δ, say very roughly $0.3/\delta = 0.2/\sqrt{x\nu}$. The corresponding wave length is $\sim 10\pi\sqrt{x\nu}$; the number of waves between 0 and x is roughly x divided by $10\pi\sqrt{x\nu}$, i.e. $\sim R_\delta/50$. The first unstable modes occur when $R_\delta \sim 500$, i.e., one has to be able to resolve at least 10 waves between the leading edge and the first occurrence of growing modes. One can also see that the time period is correspondingly small. For this reason stability calculations based on the Navier–Stokes equations are very expensive indeed (see e.g. Fasel [15]).

There is an additional constraint in the present work. It is interesting to compare the behavior of the growing modes in two dimension with the corresponding behavior in three dimensions; the two cases are quite different, and the contrast is very instructive when one is interested in the transition to turbulence. We wish to use comparable numerical parameters in two and in three dimensions, so that the comparison of the results be believable; the cost of three dimensional calculations is of course much larger even than the cost of two dimensional calculations; we must therefore look for ways of representing the boundary layer which are as economical as possible and yet exhibit a correct behavior.

There is no obvious way in which the steady Blasius profile can be imposed exactly on our array of vortex elements at the initial time. On the other hand, a calculation which starts from impulsive initial data contains a large and rather long-lived transient component whose behavior is not easily distinguishable from that of a growing mode. Part of this problem can be removed as follows: Start the calculation by using the sheet representation only (which is cheap and allows no instability), and run for a time $0 < t < T_0$, T_0 large enough so that the Blasius profile will have been reached with some not unreasonable accuracy. At time $t = T_0$ allow some or all of the sheets to become blobs. In all the two dimensional runs described below we set $T_0 = 1$.

It is quite obvious that we shall not be able to duplicate the results of linearized stability theory. The initial data will not coincide exactly with the Blasius solution. The perturbations will not be small. In Fasel [15] the perturbation amplitude was about 0.05 of the free-stream velocity—an impossibly low level for our method. Our results should be compared with the behavior of finite amplitude perturbations in noisy flow. The advantages of our numerical method can be seen from the fact that the method requires no scaling. The very same program can be used to solve an interior flow problem. The algorithm provides its own scaling and concentrates the computing effort where it is needed. This should be particularly important in other problems where thin shear layers occur at locations which are not known in advance.

In the calculations described below, the vorticity is created at walls in the form of sheets, with all $|\xi_i| \leqq \xi_{\max}$. If the amount of vorticity needed to satisfy the boundary conditions is less than ξ_0, no sheets are created; here, $\xi_0 = \xi_{\max}/2$. When sheets find themselves at $y > l$ at time $t > T_0$, they become blobs; they become sheets again if $y < l$. l must be such that the probability that $\eta > 2l$ is small. We checked that as long as $l \sim 1.5\times$ the standard deviation of η, the results were insensitive to the value of l. Detailed calculations were performed for $0 \leqq x \leqq 1$; i.e., the typical streamwise length L is 1, and thus $R = U_\infty L/\nu = \nu^{-1}$. Both sheets and blobs were followed for $x > 1$ but allowed to move only with the random component in their laws of motion. When they reached $x = X$ they were deleted. This was done to ensure that the right boundary at $x = 1$, which is introduced only for computational convenience, behaves as an absorbing boundary and does not affect adversely the calculations in the region of interest $0 \leqq x \leqq 1$. We usually picked $X = 2$.

The interaction of two elements at least one of which was a sheet, was computed as if both were sheets. Two blobs interacted as blobs. In the computation of the tangential velocity at the wall, all elements were treated as sheets.

After much experimentation we picked $\xi_{\max} = 0.6$. This is a large value of $\xi_{\max}$ and produces a crude and noisy boundary layer; however, it is sufficient for exhibiting the main effects. A relatively large value of $\xi_{\max}$ reduces the number of elements in the calculation, and, as explained above, this is of particular importance since we intend to present a three dimensional calculation. The choices of h and k are described in the next section.

In the steady state, the drag $D(x)$ on the piece of boundary between 0 and x can be computed by the momentum defect formula (see e.g. [35]).

$$D(x) = \int_0^\infty u(U_\infty - u)\, dy, \qquad u = u(x, y). \tag{10a}$$

The normalized drag is defined as

$$d(x) = D(x)/D_0(x), \tag{10b}$$

where $D_0(x)$ is the Blasius drag $D_0(x) = 0.6641\sqrt{x\nu}$, which can be obtained from the Blasius solution. The velocities for use in formulas such as (10) are computed as if all the elements were sheets. We shall use $d(x)$ defined by (10) as a measure of the amplitude of the growing modes even when the flow is not steady and $D(x)$ is not really the drag on $[0, x]$.

Finally, we observed that if k was too large the solution exhibited large oscillation of no possible physical significance. This is readily understood. We are solving a moderately large system of ordinary differential equations by Euler's method. The remedy is to reduce k. $k \leqq h$ is adequate.

Numerical results in two dimensions. In Table 1 and Fig. 2 we display the normalized "drag" $d(x)$ at $X = 1/2$ as a function of R and t. ($d(X)$ is the ratio $D(X)/D_0$, see formula (10) above). These calculations were made with $k = h = 0.05$;

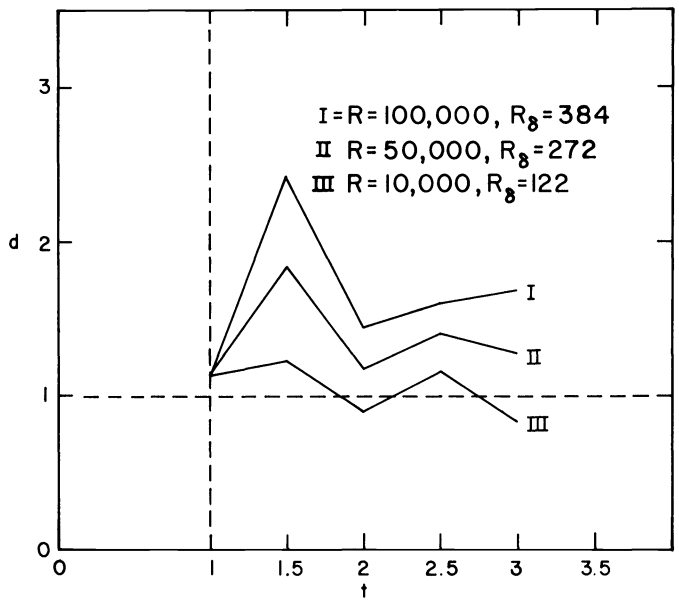


FIG. 2. *Growth of an unstable layer in two dimension.*

TABLE 1
Drag as a function of Reynolds number and time.

t	$R = 10000$ ($R\delta = 122$)	$R = 50000$ ($R\delta = 272$)	$R = 100000$ ($R\delta = 384$)
1	1.11	1.11	1.11
1.5	1.23	1.87	1.97
2	0.89	1.18	1.39
2.5	1.15	1.44	1.57
3	.77	1.25	1.65

the other parameters are as described in the preceding section: $\xi_{\max} = 0.6$, $X = 2$, $\sigma = h/\pi$. The point $X = \frac{1}{2}$ is in the middle of the region of interest. In our units, $\nu = R^{-1}$, and $R_\delta = \delta/\nu = 1.72\sqrt{R/2}$. From Table 1 and Fig. 2 one can see that $d(X)$ is growing for $R_\delta = 394$, $R = 10^5$; $d(X)$ is not growing for $R_\delta = 122$, $R = 10^4$, and $d(X)$ is initially excited but ultimately slowly decaying for $R_\delta = 272$, $R = 5 \times 10^4$. This last fact is debatable; the value $R_\delta = 272$ seems to be the approximate value of $R'_{\delta c}$. These results are reasonable in view of what is known from the theory and from experiments.

In Fig. 4 we exhibit the edge of the boundary layer as a function of x for $t = 3$, $R = 10^4$. The edge is defined as the smallest value of y for which $u = U_\infty$. The edge is not at infinity because we have finite number of vortex elements and thus the tail of the probability distribution of the locations of the elements is not accurately approximated. The layer is stable at this value of R, yet the edge is ragged and the layer appears to be "intermittent" (for a definition of intermittency see e.g. Cebeci and Smith [4]). The "intermittency" is due to the presence of discrete vortices; this connection will be

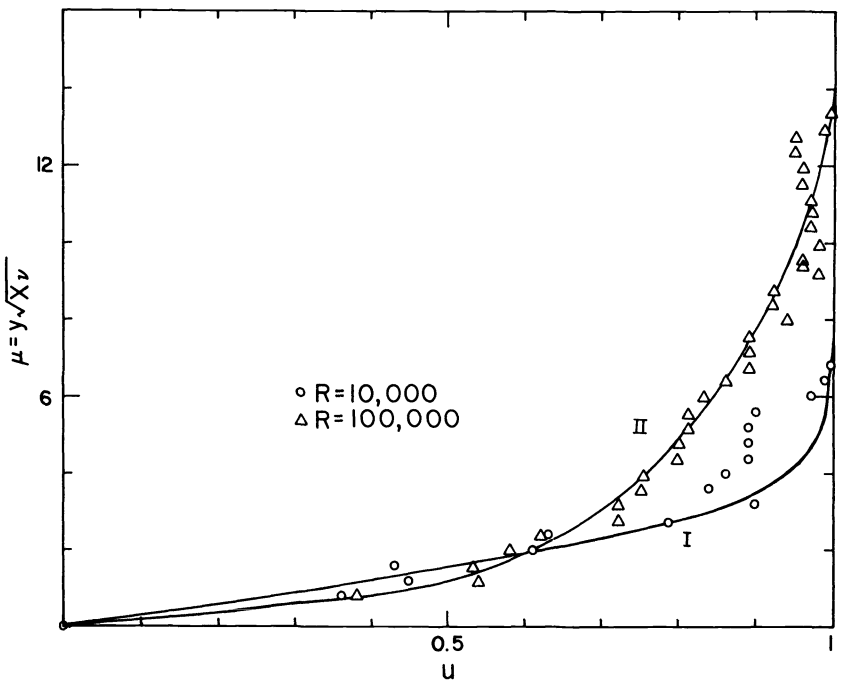


FIG. 3. *Velocity profiles in two dimensions.*

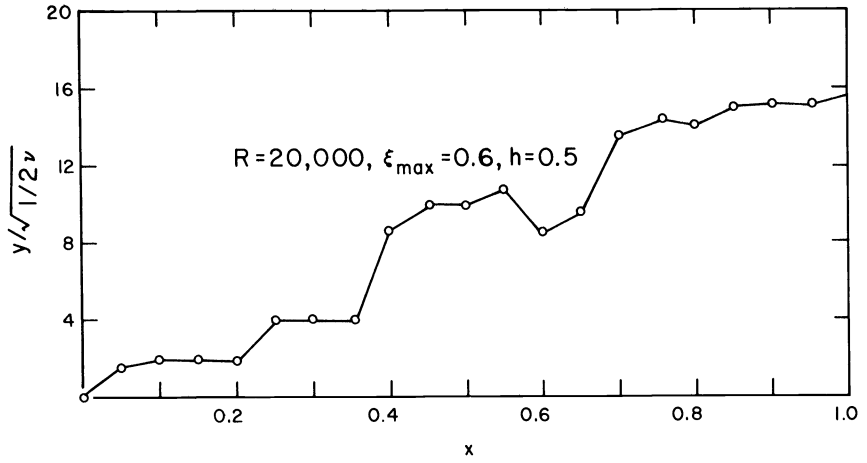


FIG. 4. *Boundary layer shape.*

exploited elsewhere for producing models of intermittency. It is obvious from Fig. 4 that the wave length of the growing modes cannot be determined directly from the instantaneous velocity distribution. However, it can be estimated indirectly. Consider the following question: how small must h be to allow us to distinguish between stable and unstable layers? Suppose that for $h > h_0$ this distinction can be made, but for $h \leqq h_0$ the layer appears to be stable even when it should not be. Then h_0 is an estimate of the wave length of the growing modes, since when $h \leqq h_0$ these modes are suppressed. In Table 2 we present the values of $d(X)$ at $X = \frac{1}{2}$ as a function of h for $R = 10^5$. We see that $10 < h_0 < 15$, in a reasonable if rough agreement with the Tollmien–Schlichting theory.

TABLE 2
Drag as a function of h, $R = 100000$, $R_\delta = 384$.

$h = 1/20$,	$k = 1/20$	$t = 1$	1.5	2	2.5	3
		$d = 1.11$	1.97	1.39	1.57	1.65
$h = 1/15$,	$k = 1/15$	$t = 1.27$	2	2.67	3.33	
		$d = 0.98$	1.48	1.66	1.70	
$h = 1/10$,	$k = 1/10$	$t = 1$	2	3		
		$d = 0.98$	1.10	1.08		

In Fig. 3 we display the velocity as a function of $\mu = y/\sqrt{\nu X}$ at $X = \frac{1}{2}$ for $R = 10^4$ and $R = 10^5$, averaged over 10 steps between $t = 2.5$ at $t = 3$. Curve I is the laminar steady Blasius profile, and curve II was drawn in what appears to the eye as a reasonable neighborhood of the points obtained at $R = 10^5$. The fluctuations are large (as one may well expect since $\xi_{max} = 0.6$), but the points at $R = 10^4$ are in a reasonable agreement with the Blasius curve; curve II (an unstable case) has a different shape. The gradients are first sharper, then smaller than in the stable case. This is consistent with experience in the unstable regime of thermal convection (see e.g. [6]). It is also consistent with data for a turbulent boundary layer in the following sense: The Tollmien–Schlichting waves are large scale structures in comparison with boundary layer thickness, while in the

stable regime there are no organized structures. In the turbulent regime one can associate a velocity with an eddy size; the changes in the profile due to the transition from the stable to the unstable regime should be of the same nature as the changes in the velocity profile which occur when the eddy size increases. This is indeed the case (see Favre et al., [16]; their data are reproduced in Lighthill, [28]).

A typical run from $t=0$ to $t=3$ with the numerical parameters used here took about 10 minutes on the UC Berkeley CDC 6400 computer. At the end of the calculation, there were about 200 sheets and 300 blobs.

The physical problem in three space dimensions. We now consider the three dimensional version of the preceding problem.

Consider a semi-infinite flat plate placed on the half plane $z=0$, $x>0$. A fluid of density 1 occupies the half space $z>0$. At time $t<0$ the fluid is at rest, at $t=0$ the fluid is impulsively set into motion with velocity $U_\infty=1$. The Navier–Stokes equations in three dimensional space can be written in the form:

$$\partial_t \boldsymbol{\xi} + (\mathbf{u}\cdot\nabla)\boldsymbol{\xi} - (\boldsymbol{\xi}\cdot\nabla)\mathbf{u} = R^{-1}\Delta\boldsymbol{\xi}, \tag{11a}$$

$$\boldsymbol{\xi} = \operatorname{curl}\mathbf{u}, \tag{11b}$$

$$\operatorname{div}\mathbf{u} = 0. \tag{11c}$$

$\mathbf{u}=(u, v, w)$ is the velocity vector, and $\mathbf{r}=(x, y, z)$ is the position vector. The boundary conditions are

$$\mathbf{u} = (U_\infty, 0, 0) \quad \text{for } z=\infty,\ t>0, \tag{12a}$$

$$\mathbf{u} = 0 \quad \text{for } z=0,\ x>0, \tag{12b}$$

$$\frac{\partial w}{\partial y} = 0 \quad \text{for } z=0,\ x<0. \tag{12c}$$

Appropriate Prandtl equations can also be written. We shall need below only a simplified version of the equations, as well as the following fact about three dimensional boundary laminar layer approximations: The vertical component of the vorticity vanishes, i.e., for a solution of the Prandtl equations, $\boldsymbol{\xi}=(\xi_1, \xi_2, 0)$.

The Prandtl equations in three dimensions admit a two dimensional solution, the Blasius solution. That solution is unstable at high enough R. Squire's theorem (Lin, [29, p. 27]) states that the problem of instability to three dimensional infinitesimal perturbation is equivalent to a two dimensional problem at lower R.

Once the two dimensional perturbations begin to grow, several striking phenomena occur. In particular, before turbulence sets in, streamwise vortices (i.e. vortices whose axis is parallel to the mean flow) make their appearance. Intense secondary instabilities follow, and spots of intense motion emerge at random locations. An experimental investigation of boundary layer instability can be found in Klebanoff et al. [23]. Experimental investigations of turbulent boundary layers, in which phenomena resembling those which first arise immediately after the onset of instability persist and may be responsible for some of the observed features, are described e.g. in Favre et al. [16], Kline et al. [24], Willmarth [38]; theoretical aspects of several aspects of instability are found in Greenspan and Benney [17], Benney [3], Lighthill [28]. One of the major conclusions from the experimental data in Klebanoff et al., [23] is that the perturbed flow is periodic in the transverse direction (i.e., y direction). It is therefore natural to consider in three dimensions equations (11) with the added periodicity conditions

$$\mathbf{u}(x, y+q, z) = \mathbf{u}(x, y, z), \qquad \boldsymbol{\xi}(x, y+q, z) = \boldsymbol{\xi}(x, y, z), \tag{13}$$

etc. Furthermore, from Klebanoff et al. (1962) we conclude that q is roughly equal to the streamwise wave length of the first unstable Tollmien–Schlichting waves; roughly, $q = 0.1$ in our units. We shall therefore be solving equations (11) with the boundary conditions (12) and (13), and $q = 0.1$.

The numerical methods in three dimensions. We consider first the three dimensional analogue of the blob method. The three dimensional problem is more difficult because the vorticity $\boldsymbol{\xi}$ is now a stretchable vector quantity which must satisfy div $\boldsymbol{\xi} = 0$.

In earlier three dimensional calculations (Leonard, [26], [27], Del Prete [13], Chorin, (unpublished)), the vorticity field was represented as a sum of vortex filaments. The difficulties with this approach are: (i) a huge amount of bookkeeping is required to keep track of the changing vortex configurations; (ii) there is no obvious way to generate the filaments at the boundary in a consistent manner. We bypass these difficulties by representing the vorticity as a sum of vortex segments (Fig. 5). Each vortex segment moves in the flow field induced by all the others. The condition div $\boldsymbol{\xi} = 0$ will be satisfied only approximately. The segments have no independent physical significance. The two dimensional blobs do not have one either; physical vortices or vortex tubes are expected to emerge from the superposition of the computational blobs or segments. A segment Λ is defined by seven quantities: The coordinates $\mathbf{r}^{(1)} = (x^{1)}, y^{(1)}, z^{(1)})$ of the center of its base, the coordinates $\mathbf{r}^{(2)} = (x^{(2)}, y^{(2)}, z^{(2)})$ of the center of its top, and its intensity κ. We shall write $\Lambda_i = (x_i^{(1)}, y_i^{(1)}, z_i^{(1)}, x_i^{(2)}, y_i^{(2)}, z_i^{(2)}, \kappa_i)$, $i = 1 \cdots, N$, N = number of segments. The base and the top are circles of radius σ, (the cut-off), which will be determined below.

Given a vorticity yield $\boldsymbol{\xi}(\mathbf{r})$, the velocity field in a fluid which fills out the whole space is given by the Biot–Savart formula (see e.g. [2]):

$$\mathbf{u}(\mathbf{r}) = -\frac{1}{4\pi}\int \frac{\mathbf{a}\times\boldsymbol{\xi}(\mathbf{r}')}{a^3}\,d\mathbf{r}' \qquad \mathbf{a} = \mathbf{r}-\mathbf{r}', \quad a = |\mathbf{a}|. \tag{14}$$

If the vorticity field is a sum of N closed vortex lines with the ith line having intensity κ_i, (14) becomes

$$\mathbf{u}(\mathbf{r}) = -\frac{1}{4\pi}\sum_{i\neq 1}^{N}\kappa_i\int_{i\text{th line}}\frac{\mathbf{a}\times\mathbf{s}}{a^3}\,ds. \tag{15}$$

$\mathbf{s} = \mathbf{s}(\mathbf{r}')$ is the unit tangent vector to the ith line at r', $ds = ds(r')$ is the arc length along the ith line, and as before $\mathbf{a} = \mathbf{r} - \mathbf{r}'$. We now seek an interaction law between vortex segments which will approximate the motion induced by (14) or (15).

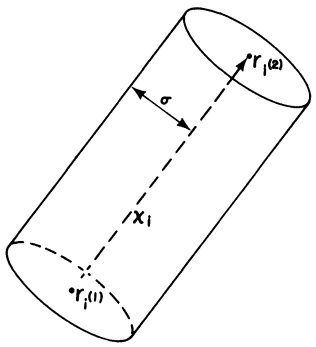


FIG. 5. *A vortex segment.*

Inside the segment the velocity field must be kept bounded, just as is the case in two dimension. Furthermore, the field must be modified inside the segments in such a way that the segments will be compatible with the boundary calculations (see below). The problem of the finding the correct formulation of the vortex method in three dimensions is difficult, (see e.g. Leonard [27]). The formulation offered here is plausible but not rigorously justified.

We require that the motion of a vortex ring or line made up of vortex segments should preserve the shape of the ring or line. This can be accomplished by ensuring that the configuration of the vectors **a** and of the velocity vectors which enter the formula for the motion of the tips of the segment is the appropriate translate of the corresponding configurations which determine the motion of the bases. Thus, let Λ_i, Λ_j be two vortex segments; define

$$\mathbf{r}_i^{(1)} = (x_i^{(1)}, y_i^{(1)}, z_i^{(1)}), \qquad \mathbf{r}_i^{(2)} = (x_i^{(2)}, y_i^{(2)}, z_i^{(2)}),$$

$$\mathbf{s}_j = \mathbf{r}_j^{(2)} - \mathbf{r}_j^{(1)},$$

$$\mathbf{a}_{ij}^{(1)} = \mathbf{r}_j^{(1)} - \mathbf{r}_i^{(1)}, \ \mathbf{a}_{ij}^{(2)} = \mathbf{r}_j^{(2)} - \mathbf{r}_i^{(2)}, \quad \text{with } a_{ij}^{(1)} = |\mathbf{a}_{ij}^{(1)}|, \text{ etc.}$$

The velocity fields $\mathbf{G}_{ij}^{(1)}$, $\mathbf{G}_{ij}^{(2)}$ induced by Λ_j at $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ will be approximated by:

If $a_{ij}^{(1)} \geqq \sigma$ and $a_{ij}^{(2)} \geqq \sigma$:

$$\mathbf{G}_{ij}^{(1)} = \frac{-\kappa_j}{4\pi} \frac{\mathbf{a}_{ij}^{(1)} \times \mathbf{s}_j}{(a_{ij}^{(1)})^3}, \tag{16a}$$

$$\mathbf{G}_{ij}^{(2)} = \frac{-\kappa_j}{4\pi} \frac{\mathbf{a}_{ij}^{(2)} \times \mathbf{s}_j}{(a_{ij}^{(2)})^3} \tag{16b}$$

If either $a_{ij}^{(1)} < \sigma$ or $a_{ij}^{(1)} < \sigma$:

$$\mathbf{G}_{ij}^{(1)} = \frac{-\kappa_j}{4\pi} \frac{\mathbf{a}_{ij}^{(1)} \times \mathbf{s}_j}{\sigma^2 a_{ij}^{(1)}}, \tag{17a}$$

$$\mathbf{G}_{ij}^{(2)} = \frac{-\kappa_j}{4\pi} \frac{\mathbf{a}_{ij}^{(2)} \times \mathbf{s}_j}{\sigma^2 a_{ij}^{(2)}}. \tag{17b}$$

The equations of motion for each segment can now be obtained by summing the contributions of all the other segments and then adding to that sum the appropriate random component. This yields

$$\mathbf{r}_i^{(1)n+1} = \mathbf{r}_i^{(1)n} + k \sum_{j \neq i} \mathbf{G}_{ij}^{(1)} + \boldsymbol{\eta}, \tag{18a}$$

$$\mathbf{r}_i^{(2)n+1} = \mathbf{r}_i^{(2)n} + k \sum_{j \neq 1} \mathbf{G}_{ij}^{(2)} + \boldsymbol{\eta}, \tag{18b}$$

where $\mathbf{r}_i^{(1)n} \equiv \mathbf{r}_i^{(1)}(nk)$, etc., and $\boldsymbol{\eta}$ is a vector $\boldsymbol{\eta} = (\eta_1, \eta_2, \eta_3)$, with η_1, η_2, η_3, Gaussian random variable with means 0 and variances $2k/R$, independent of each other. $\boldsymbol{\eta}$ in (18a) is identical to $\boldsymbol{\eta}$ in (18b), since diffusion does not introduce rotation or stretching. The boundary condition at the wall can be satisfied as before by the introduction of appropriate image segments.

One can write the boundary layer equation in three dimensions and solve them by a method in which the computational elements are pieces of a vortex sheet (= "tiles") with sides h_1 in the x direction and h_2 in the y direction. Each tile carries a two dimensional vortex with components ξ_1, ξ_2. As observed earlier, $\xi_3 = 0$ in the boundary layer equations. However, we shall use the tiles only near the boundary, where vortex

stretching is presumably negligible, or to create an initial Blasius profile, in which stretching is exactly zero. Therefore, the boundary layer equations we shall be solving reduce to

$$\partial_t \xi_1 + (\mathbf{u}\cdot\nabla)\xi_1 = \nu \frac{\partial^2 \xi_1}{\partial z^2}, \qquad \partial_t \xi_2 + (\mathbf{u}\cdot\nabla)\xi_2 = \nu \frac{\partial^2 z_2}{\partial z^2}$$

$$\xi_1 = \frac{\partial v}{\partial z}, \qquad \xi_2 = -\frac{\partial u}{\partial z}, \qquad \operatorname{div}\mathbf{u} = 0, \qquad \mathbf{u} = (u, v, w).$$

These equations can be solved by a straightforward extension of the sheet method described earlier. No vortex stretching will be taken into account, and we shall not take the trouble to write out the equations in full. The rejection and variance reduction techniques carry over from the two-dimensional case. Care is taken to ensure that $\sqrt{\xi_1^2+\xi_2^2} \leqq \xi_{\max}$.

A tile created near the wall can become a segment if $t > T$ or if $z_i > l$. A segment which falls below l becomes a tile again. The transformation of tiles into segments (and vice versa) must obey the following conditions:

(i) A tile must become a segment parallel to the wall; i.e., if a tile $(x_i, y_i, z_i, \xi_{1i}, \xi_{2i})$ becomes a segment $(x_i^{(1)}, y_i^{(1)}, z_i^{(1)}, x_i^{(2)}, y_i^{(2)}, z_i^{(2)}, \kappa_i)$ we must have

$$z_i^{(2)} - z_i^{(2)} = 0. \tag{19a}$$

(ii) A flow which is two dimensional when described by tiles must remain two dimensional when described by segments. The two dimensionality of a flow described by segments will be preserved only if the flow fields seen by the tips of the segments are translates of the flow fields seen by the bases, with a translation vector normal to the plane of the flow and pointing in the direction of a fixed normal $\mathbf{n}$ to that plane.

(iii) The stretching of the several segments represents the stretching of vorticity, which will be represented accurately only if the length of the segments is reasonably small. A reasonable normalization of that length in our problem is

$$y_i^{(2)} - y_i^{(1)} = h_2 \quad \text{when a segment is created.} \tag{19b}$$

(iv) The circulation around a vortex line made up of tiles must equal the circulation around a vortex line made up of segments. If $y_i^{(2)} - y_i^{(1)}$ is normalized by (19b) this requirement leads to

$$\kappa_i = h_1 \sqrt{\xi_{1i}^2 + \xi_{2i}^2}\, \operatorname{sgn}(\boldsymbol{\xi}' \cdot \mathbf{n}), \tag{19c}$$

where $\boldsymbol{\xi}' = (\xi_{1i}, \xi_{2i})$, $\mathbf{n}$ is the fixed normal to the plane of the flow and $\operatorname{sgn}(\alpha) = 1$ if $\alpha \geqq 0$, $\operatorname{sgn}(\alpha) = -1$ if $\alpha < 0$.

The remaining connecting formulas between segments and tiles are obviously

$$x_i^{(1)} = x_i, \tag{19d}$$

$$y_i^{(1)} = y_i, \tag{19e}$$

$$z_i^{(1)} = z_i. \tag{19f}$$

Formulas (19) are of course invertible, and the computational elements can be treated as either tiles or segments, as the occasion warrants.

When two segments interact, their interaction is given by formulas (19); when a segment and a tile interact, they are both viewed as tiles.

Finally, the cut-off must be determined. We must require that if we consider on one hand the interaction of two infinite vortex lines parallel to the y axis represented by

segments, and on the other hand the interaction of the same vortex lines represented by tiles, the former should approach the latter as the lines approach the wall. This requirement obviously reduces to the condition imposed on σ in two dimensions, and yields $\sigma = h_1/\pi$. This conclusion is of course legitimate only if most of the vorticity does indeed point in a direction parallel to the y axis.

Application of the numerical methods in three dimensions. In this section we discuss some of the features of the numerical method which are specific to the particular application at hand. Most of the numerical parameters are chosen just as they were chosen in the two-dimensional case; in particular, l and L. We picked $h_1 = k = \frac{1}{15}$, since the two-dimensional calculations showed that this was a minimal but adequate choice. We picked $h_2 = q/4$, after some experimentation showed that this value was sufficient to exhibit important effects.

The two major difficulties we encountered in three dimensions were: the large amount of computational labor, and the difficulty in imposing periodic boundary conditions on a grid-free method. The amount of labor is large not only because three-dimensional calculations are always more costly than two-dimensional calculations, but also (and especially) because the specific nature of the secondary instabilities which arise in three dimensions (see the next section) requires the creation of large amounts of vorticity at the walls. In consequence we used $\xi_{\max} = 1$. This value seems to yield results which are compatible with two-dimensional results obtained with smaller values of $\xi_{\max}$, but it is obviously so large that one may legitimately argue that what we have is a model rather than an approximation.

Periodic boundary conditions can be imposed on a vortex calculation, but the price in computing labor is high. There again we did the least we could reasonably do. For each vortex segment with base located at (x, y, z) (or its image created to satisfy the normal boundary condition, with a base at $(x, y, -z)$) we created two more segments, based at $(x, y \pm q, z)$, q = the period and took their velocity fields into account when we moved the segment. Similarly, new tiles must be created outside the strip $0 \leqq y \leqq q$ with locations and strengths determined by periodicity. Some rather complex programming is needed to keep track of the several image systems as the tiles become segments and vice versa.

Finally, we note that if $\xi_1 = 0$ at $t = 0$, i.e., if there is no streamwise vorticity at all at $t = 0$, none will ever be created by our algorithm. Thus, if we are to observe the effects of streamwise vorticity, we must introduce some by artificial means. We proceeded as follows: At $t = 0$, for one time step, we changed the velocity at infinity. Instead of $\mathbf{u}(x, y, \infty) = (U_\infty, 0, 0)$ we set

$$\mathbf{u}(x, y, \infty) = \begin{cases} (U_\infty, A, 0) & \text{for } \dfrac{q}{4} < y < \dfrac{3q}{4}, \\ (U_\infty, 0, 0) & \text{elsewhere.} \end{cases}$$

We usually picked $A = 10^{-3}$ (note that $U_\infty = 1$). For $t > k$, we reverted to $\mathbf{u}(x, y, \infty) = (U_\infty, 0, 0)$ everywhere. The effect of this initial perturbation is to create a small streamwise vortex at the boundary, whose subsequent history is determined by diffusion, transport, and stretching.

Numerical results in three dimensions. Calculations done in three dimensions with $A = 0$ (i.e. with no perturbation which could trigger three dimensional effects) produce results similar to the results of two dimensional calculations. They afford a check on both, but are not worth discussing separately.

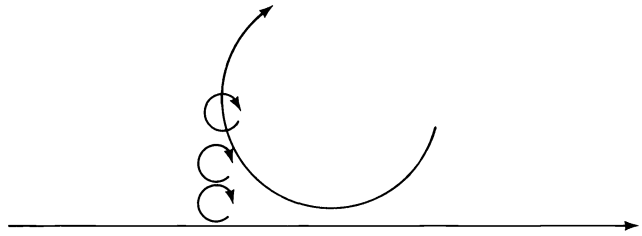


FIG. 6. *Amplification of streamwise rotation.*

Even a very small value of A (i.e. a very small three dimensional perturbation) has a substantial effect at all values of R we tried. The first phenomenon one observes when $A \neq 0$ is that the boundary layer becomes thicker than in the case $A = 0$. The mechanics of this effect are somewhat complex. A reasonable qualitative explanation is as follows: the rotation whose axis is parallel to the flow induces the creation of new streamwise vorticity at the wall. The new vorticity is then collected in streamwise strips in which the flow induced by the streamwise vortex leads away from the wall, while the regions where the induced flow points to the wall are depleted (see Fig. 6). The part of the boundary layer which thus expands can expand substantially, while the part which contracts cannot contract below zero. As a result the computed boundary layer thickness δ increases; δ at (X, Y) is defined by

$$\delta = \int_0^\infty (1 - u(x, y, \alpha)/U_\infty)\, d\alpha.$$

In Figs. 7 and 8 we plot the ratio δ/δ_b where δ = computed boundary layer thickness at $X = \frac{1}{2}$ averaged over a period in y, and δ_b = boundary layer thickness at $X = \frac{1}{2}$ computed from the steady Blasius solution. In Fig. 7, $R = 20000$. Note that at $t = 3$, R_δ computed with the steady δ is $R_\delta = 187$, and thus the layer should be steady. However, if R_δ is evaluated with the computed boundary layer thickness, R_δ at $X = \frac{1}{2}$ is approximately 300, and R_δ at $X = 1$ is approximately 440, well over the value at which the layer becomes unstable in the two-dimensional calculation. In Fig. 8, $R = 100000$,

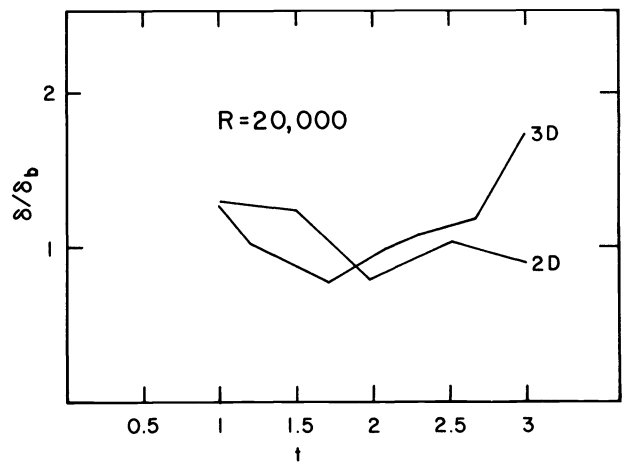


FIG. 7. *Growth of boundary layer thickness,* $R = 20000$.

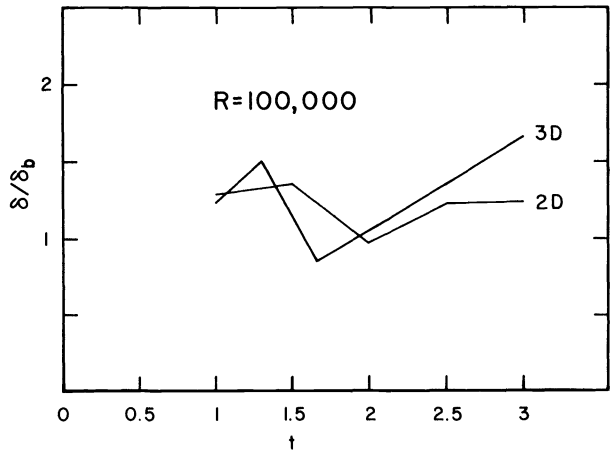


FIG. 8. *Growth of boundary layer thickness,* $R = 100000$.

and the same effect is reproduced. The computed values of the drag are not greatly affected by this thickening of the layer. (This is quite plausible, in view of the extra factor u in the integrand in the formula for the drag; the effect of this factor is to reduce the dependence of the drag on the velocity profile near the wall.)

When the layer becomes unstable to Tollmien–Schlichting waves, the streamwise vorticity begins to grow. The possible mechanisms for this growth are well known: The waves stretch lines; furthermore, they can create situations in which a horizontal streamwise line tilts away from the horizontal; its higher parts move faster than the lower parts, and stretching results. All segments are initially created with length h_2. If they stretch their length becomes $|\mathbf{r}_i^{(2)} - \mathbf{r}_i^{(1)}|$. The ratio $g = |\mathbf{r}_i^{(2)} - \mathbf{r}_i^{(1)}|/h_2$ is the stretching ratio. In Figs. 8 and 9 we plot $\bar{g}$, the average value of g, averaged over all segments. It is

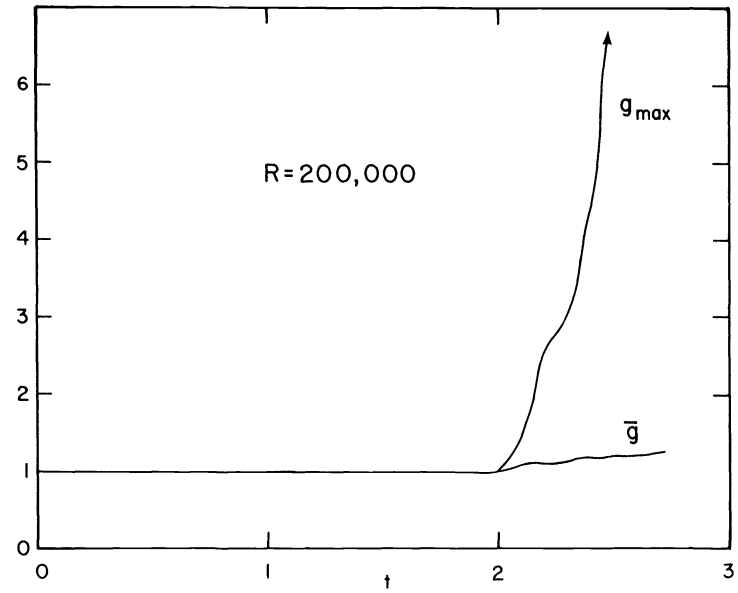


FIG. 9. *Amplification of boundary layer disturbances in three dimensions,* $R = 20000$.

seen to grow slowly with time. These figures are for $R = 20000$ and 100000. Note that at the time when $\bar{g}$ begins to grow with $R = 20000$ the layer had become thicker as a result of secondary motion and R_δ is larger than the critical value. In Fig. 9 we also plotted s, the total streamwise vorticity, and r, the ratio of newly created streamwise vorticity to newly created transverse vorticity. Roughly, r is an indication of the rate of growth of s. All these quantities are seen to grow slowly and steadily. The growth can be started earlier by increasing A. At value of R smaller than 10000, we never did succeed in inducing such growth within a time we could afford and without using very large values of A (i.e. A of order 1—not a plausible value for our problem).

The more interesting graph in Figs. 9 and 10 is the graph of the maximum value $g_{\max}$ of the stretching ratio. This value can become very large (~17), i.e., some vortices are stretched by a large amount. This suggests an extraordinary spottiness of the stretching process. This spottiness can be explained as follows: because our method is random, the local velocity profile can differ from point to point. At some points the local profile may be much more unstable than at others, and as a result secondary instabilities, whose growth rate is very large (Greenspan [17]) will occur at some points and not at others. One can also argue that as a result of the variation in local profiles, at some points the segments may depart from the horizontal more than at others, and therefore the stretching mechanism is more intense there. These two explanations may of course be identical. The "spots" make the major contribution to the growth of the mean quantities. Their presence indicates that the layer contains a mechanism for amplifying greatly small differences in local conditions. However, one should remember that our numerical layer is much noisier than a real layer is likely to be.

In Table 3 we display the values of the streamwise component of $\mathbf{u}$ at $x = ih_1$, $y = jh_2$, $z = 0$ and $R = 2 \times 10^5$, $t = 2.6$. The details of the fluctuations do not seem to have any particular physical significance. The values of R and t were picked somewhat arbitrarily; the table shows the spottiness of the field, and also shows that, as expected, the streamwise component of ξ increases as the layer thickens.

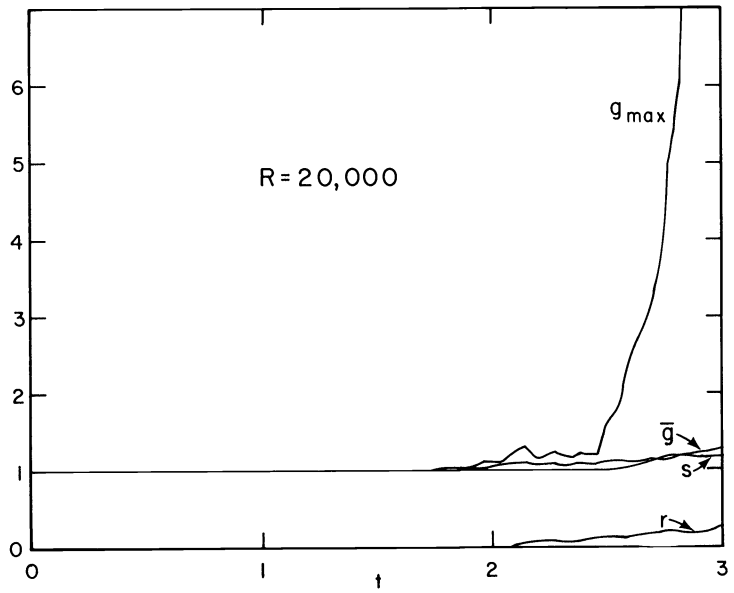


FIG. 10. *Amplification of boundary layer disturbances in three dimensions,* $R = 100000$.

When g increases, more and more segments and tiles have to be created; this is why we needed a large value of ξ_{max}. A further consequence is that computing for larger times than what we displayed is more expensive than we could afford. A typical run for $0 \leqq t \leqq 3$ took about one hour of CDC 6400 time at Berkeley.

TABLE 3
Streamwise vorticity at the boundary, $R = 2 \times 10^5$, $t = 2.6$

	$j=1$	2	3	4
$i=1$	.000	.000	.000	−.001
2	.000	.000	.000	.000
3	.000	.000	.000	.000
4	.003	.000	.000	.000
5	.028	.000	.001	.000
6	.000	.000	.000	.001
7	−.046	−.012	.001	.001
8	−.046	−.004	−.008	−.010
9	−.038	−.028	−.002	−.244
10	−.187	−.054	−.007	−.093
11	.091	−.065	.030	.076
12	−.062	−.310	.136	.205
13	2.260	.507	.480	1.070
14	−.077	.150	.826	−.621
15	−.552	.422	.001	−.273

Conclusions. Our vortex methods, including the new three dimensional version and the new vorticity creation procedure, seem to be able to reproduce important features of boundary layer behavior in two and three dimensions and at Reynolds numbers where instability is expected. The three-dimensional calculation does exhibit a growth of streamwise vorticity as well as spottiness; however, it was not performed for times long enough for anything resembling fully developed turbulence to be present. Unlike other methods, our methods are not limited at high R by the difficulty in distinguishing real from numerical diffusion; they are however limited, like other methods, by the fact that effects not resolved cannot be seen; i.e., if there are not enough computational elements to represent a phenomenon, that phenomenon will not be observed. Since fully turbulent flow is very complicated, our methods do not remove the need for careful modeling in some practical applications.

REFERENCES

[1] W. ASHURST, *Numerical simulation of turbulent mixing layer dynamics*, SANDIA (Livermore) Report (1977).
[2] G. K. BATCHELOR, *An Introduction to Fluid Mechanics*, Cambridge University Press, London, 1967.
[3] D. J. BENNEY, *A nonlinear theory for oscillations in a parallel flow*, J. Fluid Mech., 10 (1960), p. 209.
[4] T. CEBECI AND A. M. O. SMITH, *Analysis of Turbulent Boundary Layers*, Academic Press, New York, 1974.
[5] A. CHEER, *Program BOUNDL*, LBL-6443 Suppl. Report, Lawrence Berkeley Lab. (1978).
[6] A. J. CHORIN, *A numerical method for solving incompressible flow problems*, J. Comput. Physics, (1967), p. 12.
[7] ———, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), p. 785.
[8] ———, *Gaussian fields and random flow*, Ibid., 63 (1974), p. 21.
[9] ———, *Lectures on Turbulence Theory*, Publish/Perish, Boston (1976).
[10] ———, *Vortex sheet approximation of boundary layers*, J. Comput. Physics, 27 (1978), 428.

[11] A. J. CHORIN AND J. E. MARSDEN, *A Mathematical Introduction to Fluid Mechanics*, Springer Verlag, New York, 1979.
[12] A. J. CHORIN, T. J. R. HUGHES, M. F. MCCRACKEN AND J. E. MARSDEN, *Product formulas and numerical algorithms*, Comm. Pure Appl. Math., 31 (1978), p. 205.
[13] V. DEL PRETE, *Numerical simulation of vortex breakdown*, LBL Math. & Comp. Report 1978.
[14] W. ECKHAUS, *Studies in Nonlinear Stability Theory*, Springer Verlag, New York, 1965.
[15] H. FASEL, *Investigation of the stability of boundary layers in a finite difference model of the Navier–Stokes equations*, J. Fluid Mech., 78 (1970), p. 355.
[16] A. FAVRE, J. GARIGLIO AND J. DUMAS, Phys. Fluids, 12, Suppl. (1967), p. 138.
[17] H. P. GREENSPAN AND D. J. BENNEY, *On shear layer instability, breakdown and transition*, J. Fluid Mech., 15 (1963), p. 133.
[18] O. HALD, *The convergence of vortex methods II*, SIAM J. Numer. Anal., 16 (1979), p. 726.
[19] O. HALD AND V. DEL PRETE, *The convergence of vortex methods*, Math. Comp., 32 (1978), p. 791.
[20] J. M. HAMMERSLEY AND D. C. HANDSCOMB, *Monte Carlo Methods*, Methuen, 1964.
[21] R. JORDINSON, *The flat boundary layer, Part I. Numerical integration of the Orr–Sommerfeld equation*, J. Fluid Mech., 43 (1970), p. 801.
[22] H. T. KIM, S. J. KLINE AND W. C. REYNOLDS, *The production of turbulence near a smooth wall in a turbulent boundary layer*, Ibid., 50 (1971), p. 133.
[23] P. S. KLEBANOFF, K. D. TIDSTROM AND L. D. SARGENT, *The three dimensional nature of boundary layer instability*, Ibid., 12 (1962), p. 1.
[24] S. J. KLINE, W. C. REYNOLDS, F. A. SCHRAUB AND P. W. RUNDSTADLER, *The structure of turbulent boundary layers*, Ibid., 30 (1967), p. 741.
[25] J. LAMPERTI, *Probability Theory*, Benjamin, New York, 1966.
[26] A. LEONARD, *Numerical simulation of interacting, three dimensional vortex filaments*, Proc. 4th Int. Conf. Num. Methods Fluid Dynamics, Springer Verlag, New York, 1975.
[27] ———, *Simulation of unsteady three dimensional separated flows with interacting vortex filaments*, Proc. 5th Int. Conf. Num. Mech. Fluid Dynamics, Springer Verlag, New York, 1977.
[28] M. J. LIGHTHILL, *Turbulence*, Osborne Reynolds and Engineering Science Today, Manchester Univ. Press, 1976.
[29] C. C. LIN, *The Theory of Hydrodynamic Stability*, Cambridge University Press, London, 1966.
[30] M. F. MCCRACKEN AND C. S. PESKIN, *The vortex method applied to blood flow through heart valves*, Proc. 6th Int. Conf. Num. Methods Fluid Dynamics, Springer Verlag, New York, 1978.
[31] D. MEKSYN AND J. T. STUART, *Nonlinear instability*, Proc. Roy. Soc. London, A, 208 (1951), p. 517.
[32] L. ONSAGER, *Statistical hydrodynamics*, Nuovo Cimento, 6, Suppl. (1949), p. 229.
[33] E. RESHOTKO, *Boundary layer stability and transition*, Ann. Rev. Fluid Mech., 8 (1976), p. 311.
[34] H. L. ROGLER AND E. RESHOTKO, *Disturbances in a boundary layer introduced by a low intensity array of vortices*, SIAM J. Appl. Math., 28 (1975), p. 431.
[35] H. SCHLICHTING, *Boundary Layer Theory*, McGraw-Hill, New York, 1960.
[36] A. I. SHESTAKOV, *A hybrid vortex—ADI solution for flows of low viscosity*, J. Comput. Phys., 31 (1979), p. 313.
[37] A. A. TOWNSEND, *Boundary Layer Research*, Freiburg Symposium, H. Gortler, Ed., Springer Verlag, 1958.
[38] W. W. WILLMARTH, *Structure of turbulence in boundary layers*, Advances in Appl. Mech., 15 (1976), p. 159.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0002 $01.00/0

ROBUST METHODS FOR SOLVING SYSTEMS OF NONLINEAR EQUATIONS*

JAMES L. BLUE†

Abstract. In solving systems of nonlinear equations, $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, most investigations have concentrated on methods that are efficient if the initial guess is close to a zero of $\mathbf{f}$. For many problems a good initial guess is not available, and a more robust method is needed.

We introduce a new idea based on the singular-value decomposition. Partial-rank "pseudo-Newton" steps are defined which are a natural bridge between the steepest-descent method and Newton's method, and are particularly useful when the Jacobian matrix has partial rank.

Illustrative examples show the improvement possible over the usual hybrid methods which combine steepest descent and Newton's method.

Key words. *nonlinear equations, singular-value decomposition, partial-rank matrices, Newton's method, steepest descent*

1. Introduction. We consider the problem of finding a root $\mathbf{x}^*$ of a system of n nonlinear equations in n unknowns, $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, given a starting iterate $\mathbf{x}_0$. Most investigations have concentrated on efficient local convergence, when $\mathbf{x}_0$ is close to $\mathbf{x}^*$. Various methods have been analyzed and proven quadratically or superlinearly convergent, if $\mathbf{x}_0$ is close enough to $\mathbf{x}^*$, and some useful computer programs exist. For many problems, however, a good initial guess is not available, and a more robust algorithm is needed.

We introduce a new idea based on the singular-value decomposition (SVD). Partial-rank "pseudo-Newton" steps are defined using the SVD. These pseudo-Newton steps are a natural bridge between the steepest-descent method and Newton's method. The steepest descent method often works well if $\mathbf{x}$ is far from a root, since there the Jacobian matrix is likely to be effectively of rank one. The quasi-Newton method works well when $\mathbf{x}$ is near a zero, if the Jacobian is of full rank. The new idea overcomes some of the difficulties which previous algorithms have when the Jacobian effectively has intermediate rank, between 1 and n.

2. Methods with rapid local convergence. All the rapidly-converging methods are based on a linear approximation to $\mathbf{f}$, valid near $\mathbf{x}_k$, the kth iterate,

$$\mathbf{f}(\mathbf{x}_k + \mathbf{p}_k) \approx \mathbf{f}(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)\mathbf{p}_k = \mathbf{f}_k + \mathbf{J}_k\mathbf{p}_k.$$

Here $\mathbf{J}(\mathbf{x}_k) = \partial\mathbf{f}/\partial\mathbf{x}$ is the Jacobian matrix of $\mathbf{f}$ at $\mathbf{x}_k$. Newton's method sets the left side to zero, solves the resulting set of linear equations

$$\mathbf{J}_k\mathbf{p}_k = -\mathbf{f}_k$$

for the correction $\mathbf{p}_k$, and then sets $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$.

If $\mathbf{x}_k$ is not close to $\mathbf{x}^*$, there is no guarantee that $\mathbf{x}_{k+1}$ is any better than $\mathbf{x}_k$; the iteration process need not converge. Since the relative distances of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ to $\mathbf{x}^*$ are not known, frequently the norms $\|\mathbf{f}_k\|$ and $\|\mathbf{f}_{k+1}\|$ are used instead. (We use the Euclidean norm throughout the paper.) The new iterate, $\mathbf{x}_{k+1}$, is rejected if $\|\mathbf{f}_{k+1}\| > \|\mathbf{f}_k\|$. A common modification to Newton's method is to let $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda\mathbf{p}_k$, and choose λ so that $\|\mathbf{f}_{k+1}\| < \|\mathbf{f}_k\|$. If $\mathbf{J}_k$ is exact, it is easy to show that such a λ exists.

If the Jacobian matrix is singular, the linear equations have either no solution or a whole family of solutions, and Newton's method as usually implemented fails. One might expect singular Jacobian matrices to be rare, and this seems to be the case in

* Received by the editors August 31, 1979.

† Bell Laboratories, Murray Hill, New Jersey 07974. Currently at Center for Applied Mathematics, National Bureau of Standards, Washington, D.C. 20234.

practice. However, it is quite common for $\mathbf{J}$ to be ill-conditioned, especially when $\mathbf{x}$ is far from $\mathbf{x}^*$; in finite precision, $\mathbf{J}$ may be effectively singular.

Even if the Jacobian matrix is not singular, a linear approximation to $\mathbf{f}$ may be entirely inadequate. Then the correction $\mathbf{p}$ may be more or less random; experimentally it is usually much too large. If the Jacobian is sufficiently ill-conditioned, the correction $\mathbf{p}$ may not be computed accurately enough. This is especially likely if the Jacobian is calculated by a numerical approximation rather than analytically, or if the Jacobian is updated after each iteration rather than calculated anew.

For many problems, obtaining the Jacobian matrix analytically is impractical or too expensive. A numerical approximation requires at least n evaluations of $\mathbf{f}$. Instead of calculating a new Jacobian at each step, a rank-one update may be used, such as [4],

$$\mathbf{J}_{k+1} \approx \mathbf{J}_k + \frac{[\mathbf{f}_{k+1} - \mathbf{f}_k - \mathbf{J}_k(\mathbf{x}_{k+1} - \mathbf{x}_k)](\mathbf{x}_{k+1} - \mathbf{x}_k)^T}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2},$$

giving a quasi-Newton method. Final convergence is superlinear, rather than quadratic [5].

3. Hybrid methods. Powell's hybrid method [14], [16] is a significant advance. It attempts global rather than local convergence. We give only the main idea; many details of the algorithm are omitted. Powell's idea is to combine a quasi-Newton method with a steepest-descent method. The steepest-descent direction is $-\frac{1}{2}\nabla\|\mathbf{f}\|^2 = -\mathbf{J}_k^T\mathbf{f}_k$. A steepest-descent step is defined by

$$\mathbf{g}_k = -\mu_k \mathbf{J}_k^T \mathbf{f}_k,$$

with μ_k chosen so that, if $\mathbf{f}$ is linear, $\|\mathbf{f}(\mathbf{x}_k - \mu \mathbf{J}_k^T\mathbf{f}_k)\|$ is minimized as a function of μ.

$$\mu_k = \frac{\|\mathbf{J}_k^T\mathbf{f}_k\|^2}{\|\mathbf{J}_k\mathbf{J}_k^T\mathbf{f}_k\|^2}.$$

A Newton step, $\mathbf{p}_k$, is defined as the solution of $\mathbf{J}_k\mathbf{p}_k = -\mathbf{f}_k$. It may be shown that $\|\mathbf{g}_k\| \leqq \|\mathbf{p}_k\|$.

Powell's hybrid method also uses a parameter, Δ_k, the maximum allowed step size $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$. The algorithm also includes a prescription for updating Δ_k. Powell chooses $\mathbf{x}_{k+1}$ as follows.

If $\|\mathbf{p}_k\| \leqq \Delta_k$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$.
Else if $\|\mathbf{g}_k\| \leqq \Delta_k \leqq \|\mathbf{p}_k\|$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{g}_k + (1-\alpha_k)\mathbf{p}_k$, and choose α_k, $0 < \alpha_k \leqq 1$, so that $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \Delta_k$.
Else $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{g}_k\Delta_k/\|\mathbf{g}_k\|$.

The convergence is monitored by $\|\mathbf{f}_k\|$. When convergence is not going well, Δ_k can be decreased, giving a bias towards steepest descent, which often works well when $\mathbf{x}$ is far from $\mathbf{x}^*$. When convergence is going well, Δ_k can be increased, giving a bias towards the quasi-Newton method, which often works well when $\mathbf{x}$ is near $\mathbf{x}^*$. Eventually, final convergence is superlinear because a quasi-Newton method is used.

Powell's algorithm starts by obtaining an approximation $\mathbf{J}_0$ to the Jacobian matrix at the starting point, $\mathbf{x}_0$. Then $\mathbf{J}_0$ is inverted to obtain $\mathbf{J}_0^{-1}$. Thus his method fails whenever the initial Jacobian is singular to working precision, even though the steepest-descent method could be used. (Powell does a rank-one update to $\mathbf{J}^{-1}$, rather than to $\mathbf{J}$.)

More modern methods, such as [1], avoid this problem by using a QR factorization of the Jacobian. The initial factorization takes $O(n^3)$ operations; rank-one updates take $O(n^2)$ operations, as does solving the linear equations to obtain $\mathbf{p}_k$. When the Jacobian

is singular, and even when its estimated condition is too large, the Newton step can be avoided.

In many problems, good progress is made initially with steepest descent; good progress and final convergence is made later with the hybrid method; but slow progress is made in between. (Figure 4, to be discussed later, is a graphic demonstration of one example.) The slow intermediate progress usually occurs when the Jacobian matrix is effectively singular, so that Newton's method cannot be used, but steepest descent is slow. The slow intermediate progress of steepest descent is not surprising [9].

4. A new idea. In order to shed light on the problem, we consider a singular-value decomposition (SVD) of **J**. Dropping the subscripts referring to the iteration number, we have

$$\mathbf{J} = \mathbf{U}\mathbf{S}\mathbf{V}^T,$$

where **U** and **V** are $n \times n$ orthogonal matrices and $\mathbf{S} = \text{diag}\,(\sigma_1, \sigma_2, \cdots, \sigma_n)$ is an $n \times n$ diagonal matrix, with $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq 0$. Let $\mathbf{b} = \mathbf{U}^T\mathbf{f}$. If $\sigma_n > 0$, the Newton step is

$$\mathbf{p} = -\mathbf{V}\mathbf{S}^{-1}\mathbf{b} = -\mathbf{V}\,\text{diag}\,(1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_n)\mathbf{b}.$$

The steepest-descent step is

$$\mathbf{g} = -\mathbf{V}\left\{\frac{\|\mathbf{S}\mathbf{b}\|^2}{\|\mathbf{S}\mathbf{S}\mathbf{b}\|^2}\mathbf{S}\right\}\mathbf{b}.$$

Suppose **J** is exactly a rank one matrix. Then $\sigma_2 = \sigma_3 = \cdots = \sigma_n = 0$, and

$$\mathbf{g} = -\mathbf{V}\,\text{diag}\,(1/\sigma_1, 0, \cdots, 0)\mathbf{b}.$$

In this case, $\mathbf{S}^{-1}$ does not exist, but its pseudo-inverse is $\tilde{\mathbf{S}}^{-1} = \text{diag}\,(1/\sigma_1, 0, 0, \cdots, 0)$. Thus, for a rank-one Jacobian, the steepest-descent step is exactly equal to the Newton step using the pseudo-inverse of **S**. We call the latter step a rank-one pseudo-Newton step. For a general **J**, the rank-one pseudo-Newton step is not necessarily close to the steepest-descent step.

Experimentally, the situation is frequently as follows. When a steepest-descent step gives good progress ($\|\mathbf{f}_{k+1}\| < \|\mathbf{f}_k\|/2$, say), the Jacobian is usually of rank one or effectively so, with $\sigma_1 \gg \sigma_2$. A Newton step can be used only if the Jacobian is of full rank. When the Jacobian is effectively of intermediate rank, a Newton step cannot be used, a steepest-descent step usually gives little improvement, and convergence is slow.

Intermediate-rank Jacobian matrices contain more information than the steepest-descent method can extract. We can utilize this information using intermediate-rank pseudo-inverses of **J**. If $\sigma_j > 0$, we define the rank-j pseudo-Newton step, $\mathbf{p}^{(j)}$, to be

$$\mathbf{p}^{(j)} = -\mathbf{V}\,\text{diag}\,(1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_j, 0, 0, \cdots)\mathbf{b}.$$

Since **V** is orthogonal, $\|\mathbf{p}^{(j)}\|^2 = \sum_{i=1}^{j} (b_i/\sigma_i)^2$, and therefore $\|\mathbf{p}^{(j+1)}\| \geqq \|\mathbf{p}^{(j)}\|$.

Thus we have a number of possible pseudo-Newton steps which are candidates for using to obtain the next **x**. As in the hybrid method, we prefer to maintain a maximum allowable step size, Δ_k. Let σ_r, $r \leqq n$, be the smallest positive singular value; r is the rank of the matrix.

If $\|\mathbf{p}^{(r)}\| \leqq \Delta_k$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}^{(r)}$. If $r = n$, this is a quasi-Newton step.

Else if $\|\mathbf{p}^{(j)}\| \leqq \Delta_k < \|\mathbf{p}^{(j+1)}\|$ for some $j \geqq 1$, $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}^{(j)} + \alpha_k[\mathbf{p}^{(j+1)} - \mathbf{p}^{(j)}]$, where α_k is chosen so that $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \Delta_k$.

Else $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}^{(1)}\Delta_k/\|\mathbf{p}^{(1)}\|$. If $r = 1$, this is along the steepest-descent direction.

If $\sigma_1 \gg \sigma_2$, so that the rank is effectively one, it is almost a steepest-descent step.

Thus a desired step length Δ_k can be used to determine the rank of the step. As a solution progresses, Δ_k is increased and the lengths $\|\mathbf{p}^{(j)}\|$ decrease; larger-rank pseudo-Newton steps can be used. As with the hybrid methods, final convergence is superlinear because Newton's method (or a quasi-Newton method) is used.

If the problem is linear, it may be shown that a step of any Δ reduces $\|\mathbf{f}\|$. If $\mathbf{J}$ is exact, it may be shown that there is some Δ which reduces $\|\mathbf{f}\|$.

We mention in passing that this idea is also useful when the number of equations is less than, rather than equal to, the number of unknowns.

5. Other methods. The essence of the idea presented in the previous section is to use a replacement for $\mathbf{S}^{-1}$ which, depending on Δ, can give Newton's method if $\mathbf{J}$ is of rank n, can give the steepest-descent method if $\mathbf{J}$ is of rank 1, and do something reasonable if $\mathbf{J}$ is of intermediate rank.

Another reasonable replacement for $\mathbf{S}^{-1}$ is that of Levenberg [10] and Marquardt [11],

$$\mathbf{S}^{-1} \approx \operatorname{diag}\left(\frac{\sigma_1}{\sigma_1^2+\beta^2}, \frac{\sigma_2}{\sigma_2^2+\beta^2}, \cdots\right),$$

with β chosen to give a step size of Δ. If $\beta = 0$, this gives Newton's method. If $\sigma_1 \gg \beta \gg \sigma_2$, this gives (approximately) the steepest descent method.

A priori, there is no reason to expect one replacement to be markedly better than another. The Levenberg–Marquardt method has the computational advantage that the SVD can be avoided, as Moré has shown in the context of unconstrained optimization [12].

For systems of nonlinear equations, $\mathbf{J}$ is frequently calculated by a numerical approximation, and rank-one updates to $\mathbf{J}$ are typical. After a number of partial-rank steps, before final convergence, we expect sizable errors to accumulate in $\mathbf{J}$, especially in the higher-rank portions. Philosophically, we prefer not to use any of the highest-rank components, since they may be entirely garbage part of the time.

Practically, we find that the Levenberg–Marquardt method is quite satisfactory. For several test problems, we also calculated each step $\mathbf{x}_{k+1} - \mathbf{x}_k$ using the Levenberg–Marquardt replacement for $\mathbf{S}^{-1}$. The results were comparable with the results using the pseudo-Newton replacement.

6. Illustration. In this section, we choose one $\mathbf{f}$ and three different starting points $\mathbf{x}_0$. For each, we illustrate the situation for the first step in an iterative solution process. We consider the Chebyquad family of problems [6]. Let $T_m^*(u)$ be the mth Chebyshev polynomial, shifted to the interval $[0, 1]$. Let $\mathbf{f} = (f_1, f_2, \cdots, f_n)^T$ and $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$.

$$f_j = \frac{1}{n}\sum_{m=1}^{n} T_j^*(x_m) - \int_0^1 T_j^*(u)\, du.$$

Since $\mathbf{f}$ is unchanged by a permutation of the x_m, the Jacobian is singular when any two or more of the x_m are equal; such points are saddle points.

The examples were done in single precision on a Honeywell 6070, relative precision $\varepsilon \approx 1.5 \times 10^{-8}$. The Jacobian matrices were approximated by forward differences, using a difference step of $\varepsilon^{1/2}\|\mathbf{x}_0\|$.

We choose $n = 9$, which is a difficult problem when the initial guess is not a good one. First we use the standard guess: $\mathbf{x}_0 = (1/10, 2/10, \cdots, 9/10)^T$. The initial norm is

TABLE 1

Chebyquad 9, standard starting point; ten possible first steps.

j	σ_1/σ_j	Δ	$\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$
SD	—	.021	.738
1	1	.019	.785
2	1.06	.019	.785
3	1.16	.019	.785
4	1.48	.022	.735
5	2.00	.022	.735
6	3.06	.039	.617
7	6.50	.039	.617
8	3.5E1	.712	4.1E1
9	1.6E2	.713	4.2E1

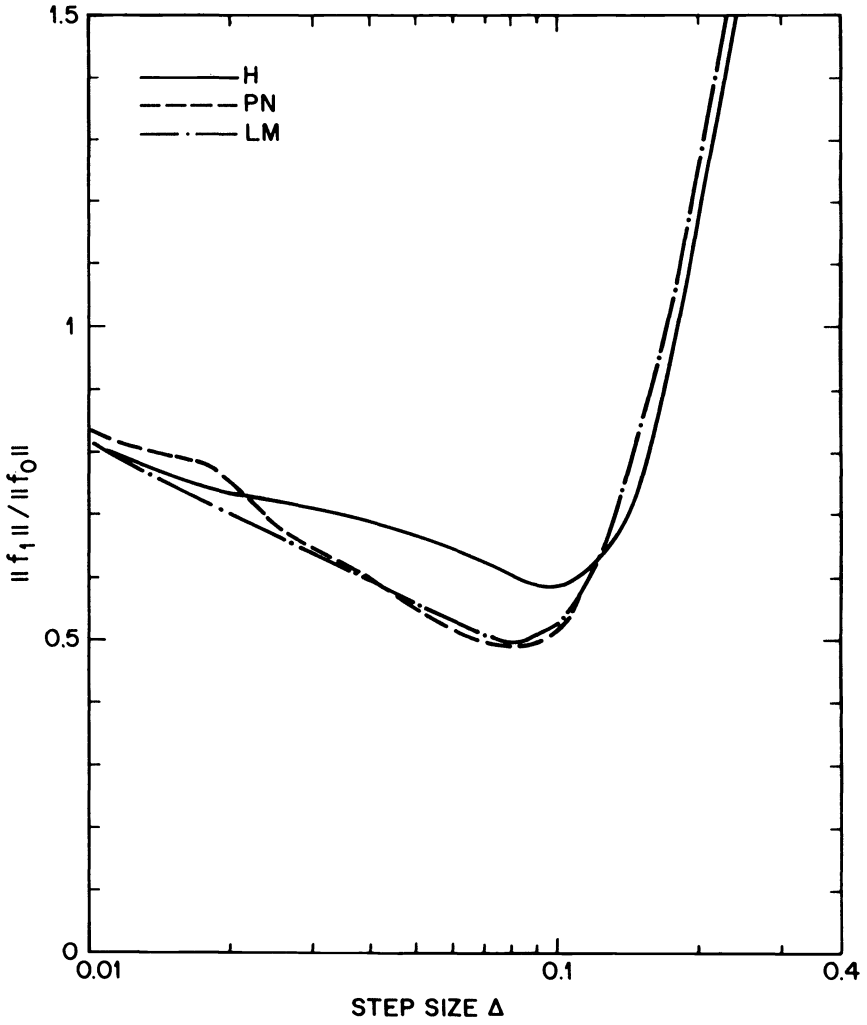


FIG. 1. *For Chebyquad 9,* $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ *vs.* Δ, *first* $\mathbf{x}_0$, *for hybrid steps* (H), *pseudo-Newton steps* (PN), *and for Levenberg–Marquardt steps* (LM).

$\|\mathbf{f}_0\| = 0.17$. The steepest-descent step (SD) has length 0.021, and produces $\|\mathbf{f}_1\| = \|\mathbf{f}(\mathbf{x}_0 + \mathbf{g}_0\| \approx 0.738\|\mathbf{f}_0\|$, as entered in line SD of Table 1. Table 1 also has similar data for nine pseudo-Newton steps. To obtain the jth step, Δ is chosen equal to $\|\mathbf{p}^{(j)}\|$. For example, $\|\mathbf{p}^{(6)}\| \approx 0.039$, and $\|\mathbf{f}_1\| = \|\mathbf{f}(\mathbf{x}_0 + \mathbf{p}^{(6)}) \approx 0.617\|\mathbf{f}_0\|$. The ratios of singular values are also shown. (The repeated values occur because of the symmetry of the problem and the starting point.) For this problem and starting point, Newton's method diverges rapidly. In Fig. 1, we plot $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ against Δ, for hybrid steps, pseudo-Newton steps, and Levenberg–Marquardt steps.

With a worse starting point, $\mathbf{x}_0 = (\frac{1}{5}, \frac{2}{5}, \cdots, \frac{9}{5})^T$, the initial norm is $\|\mathbf{f}_0\| = 1.35 \times 10^5$. Results are given in Table 2. The matrix is effectively of rank one, since $\sigma_1/\sigma_2 > 100$. An initial steepest-descent reduces $\|\mathbf{f}\|$ by a factor of approximately $1/e$. Since the condition number is about 3.5×10^7 and the machine precision is about 1.5×10^{-8}, the last few singular values and pseudo-Newton steps may not be computed very accurately, although the rank-8 step seems to be satisfactory.

TABLE 2

Chebyquad 9, worse starting point; ten possible first steps.

j	σ_1/σ_j	Δ	$\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$
SD	—	.158	.370
1	1	.158	.370
2	1.1E2	.554	.633
3	4.6E3	.814	.494
4	8.4E4	1.15	.537
5	4.2E5	1.17	.538
6	4.2E5	1.18	.536
7	6.9E5	1.24	.535
8	1.5E6	1.60	.534
9	3.5E7	4.53	1.3E2

TABLE 3

Chebyquad 9, third starting point; ten possible first steps.

j	σ_1/σ_j	Δ	$\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$
SD	—	.0047	.991
1	1	.0024	.996
2	3.95	.0232	.976
3	1.5E1	.239	.884
4	7.0E1	.578	2.2E2
5	2.0E3	2.4E1	—
6	8.8E4	1.0E3	—
7	4.4E6	1.8E4	—
8	4.7E7	1.5E5	—
9	2.4E8	3.4E6	—

The steepest-descent step and the rank-one pseudo-Newton step, which are nearly identical, reduce $\|\mathbf{f}\|$ most. In Fig. 2, we plot $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ against Δ, for hybrid steps and for pseudo-Newton steps. The plot for Levenberg–Marquardt steps is essentially the same as for pseudo-Newton steps, because the relevant singular values are widely spaced.

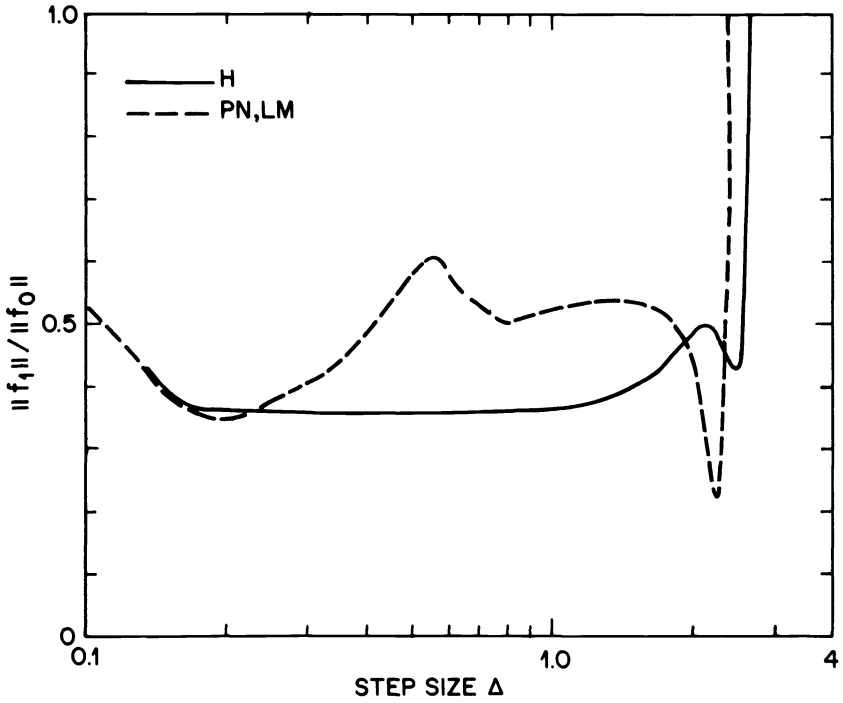


FIG. 2. *For Chebyquad 9,* $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ *vs.* Δ, *second* $\mathbf{x}_0$, *for hybrid steps* (H), *pseudo-Newton steps* (PN), *and for Levenberg–Marquardt steps* (LM).

A final starting point is a point reached by ZONE [1] in solving Chebyquad 9 starting point $(1, 2, \cdots, 9)^T$. The point is $\mathbf{x}_0 = (.8340, .8416, .9510, .9534, .9645, .9659, .9877, .9982, 1.0053)^T$. Many steepest-descent steps have resulted in reaching a place where steepest descent gives very little improvement. The Jacobian is ill-conditioned, since $\mathbf{x}_0$ is near a saddle point. As in the previous example, the last few singular values and pseudo-Newton steps may not be computed very accurately. The initial norm is $\|\mathbf{f}_0\| = 1.338$. Results are given in Table 3.

For this $\mathbf{x}_0$, the usual hybrid methods are unable to make good progress. In Fig. 3, we plot $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ against Δ for three types of steps. The minimum values are approximately .99 for hybrid, .845 for pseudo-Newton, and .88 for Levenberg–Marquardt.

7. Examples. We have implemented a version of the partial-rank pseudo-Newton method in an experimental program, ZSVD [2]. The singular-value decomposition is done by a standard method [8], using program MINFIT from Eispack [7]. Given a matrix, $\mathbf{J} = \mathbf{USV}^T$, and a vector, $\mathbf{f}$, MINFIT is used to produce the diagonal matrix $\mathbf{S}$, the orthogonal matrix $\mathbf{V}$, and the product $\mathbf{U}^T\mathbf{f}$. The matrix $\mathbf{U}$ need never be formed directly. Because the SVD takes $O(n^3)$ operations, and because there is no known economical method of updating the SVD after a rank-one change to $\mathbf{J}$, ZSVD will have considerably higher overhead than the usual hybrid method. Therefore, ZSVD is usually operated in a mode in which the SVD is done only when it seems to be necessary, and a standard hybrid method is employed otherwise. For the examples reported here, except the calculations for Fig. 4, ZSVD was changed so that it always did the SVD. The Jacobian matrices were approximated by forward differences, using a difference step of $\varepsilon^{1/2}\|\mathbf{x}\|$.

Figure 4 illustrates the progress of ZONE and ZSVD for Chebyquad 9 with a terrible guess, $\mathbf{x}_0 = (1, 2, \cdots, 9)^T$. The log of $\|\mathbf{f}\|$ is plotted against the number of calls to

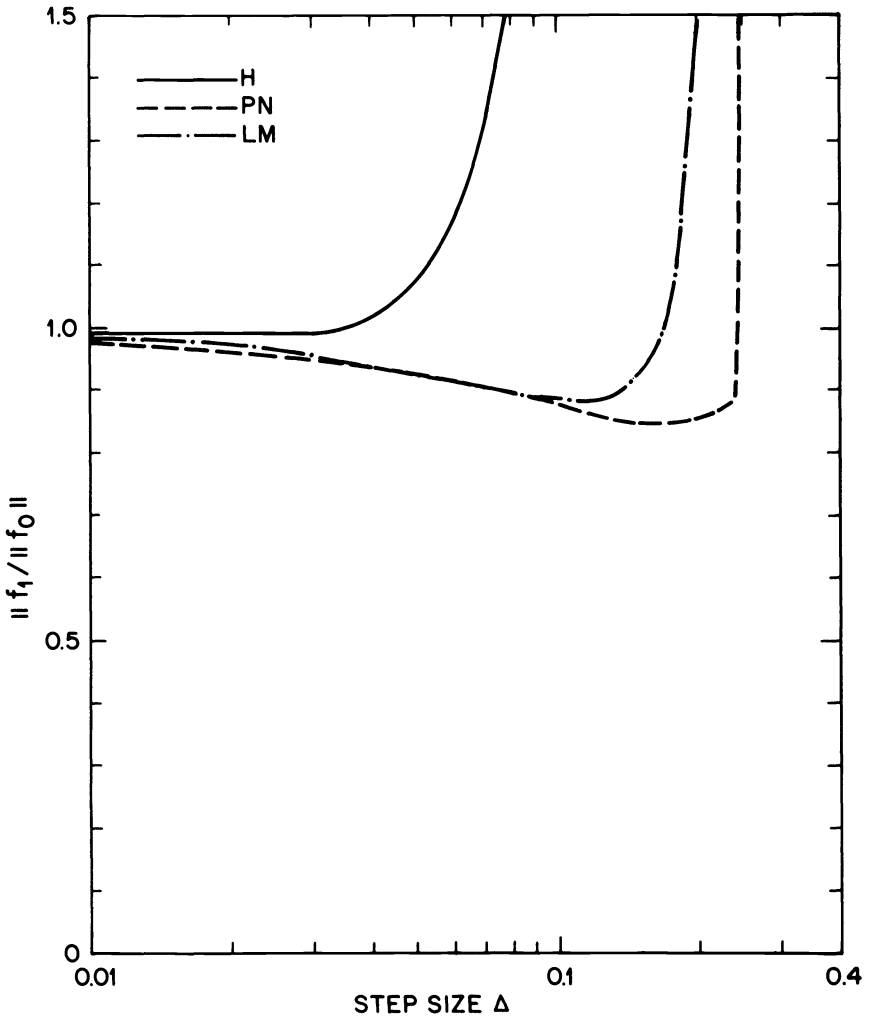


FIG. 3. *For Chebyquad* 9, $\|\mathbf{f}_1\|/\|\mathbf{f}_0\|$ *vs.* Δ, *third* $\mathbf{x}_0$, *for hybrid steps* (H), *pseudo-Newton steps* (PN), *and for Levenberg–Marquardt steps* (LM).

the function subprogram. The horizontal lines about 9 calls wide indicate function calls used to evaluate new Jacobians. Each curve shows initial rapid convergence, when steepest-descent works well. Then comes an intermediate region, where convergence is slower. Since the Jacobian is effectively of intermediate rank here, the convergence of the hybrid method, ZONE, is much slower than that of ZSVD. Finally, each program converges rapidly, with quasi-Newton steps. Table 3, given earlier, illustrates the situation near the knee of the ZONE curve, at function call 144.

Few high-quality nonlinear equation programs are widely available for comparison with ZSVD. Results are presented for two programs in MINPACK1 [13], available from Argonne National Laboratory. BRENT1 is an implementation of an algorithm by Brent [3]. It requires that each component of $\mathbf{f}$ be available separately, which is not always possible; BRENT1 is not robust, but is efficient when it does converge. HYBRD1 is a modern version of Powell's hybrid method, and is more robust than BRENT1 or Powell's implementation [16].

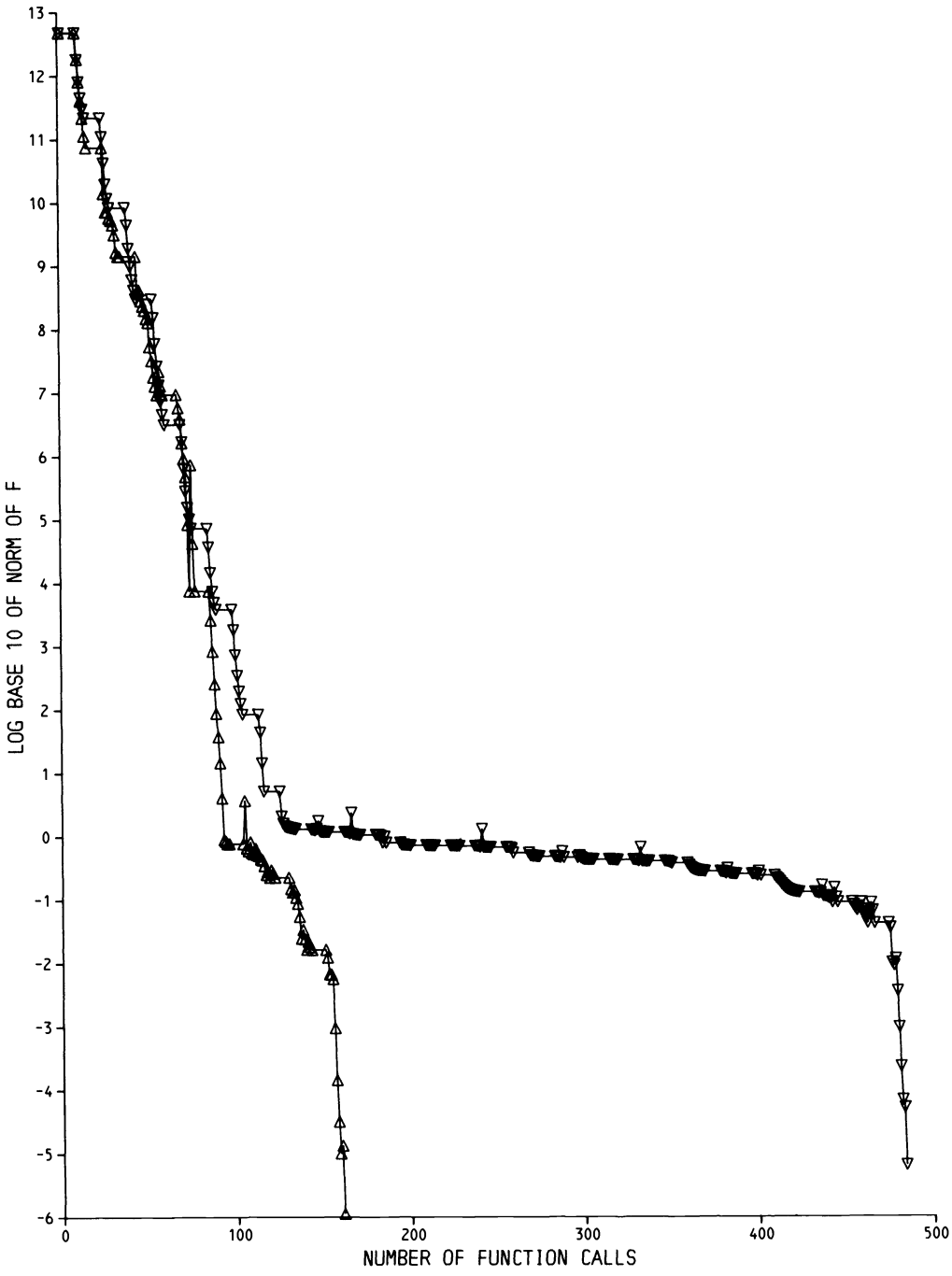


FIG. 4. *For Chebyquad* 9, $\log_{10}\|\mathbf{f}\|$ *vs. number of function calls, for starting point* $\mathbf{x}_0 = (1, 2, \cdots, 9)^T$. ZSVD *marked with* Δ, ZONE *marked with* ∇.

All examples were done in single precision on a Honeywell 6070, relative precision $\varepsilon \approx 1.5 \times 10^{-8}$. The convergence tolerances were $\|\mathbf{f}\| = 10^{-5}$ for ZSVD and HYBRD1, and $\|\mathbf{f}\|_\infty = 10^{-5}$ for BRENT1. The first two problems were run with three starting vectors. Besides a vector $\mathbf{x}_0$, chosen to be close to the solution, we used $10\mathbf{x}_0$ and $100\mathbf{x}_0$,

to test the effect of very bad initial guesses. The number of function calls for each program is given. For BRENT1, the equivalent number of vector function calls is given, i.e., the number of component evaluations divided by n, rounded to the nearest integer. To test the effect of moderately bad initial guesses, we used 20 random initial guesses in (0, 1), using the nearly-portable random number generator RAND in [17]. The initial seed for the generator was 12345; the same numbers were used for all three programs. The average number of function calls for the successful runs is given. For the random starts, the number of successful solutions is given in parentheses.

Two-point boundary value problem. The first example is from [15]. The standard central finite difference approximation to

$$u''(t) = \tfrac{1}{2}[u(t) + t + 1]^3, \qquad 0 < t < 1, \quad u(0) = u(1) = 0,$$

with $h = 1/(n+1)$, $x_k = u(kh)$, and $x_0 = x_{n+1} = 0$, gives the system

$$f_k(\mathbf{x}) = 2x_k - x_{k-1} - x_{k+1} + \tfrac{1}{2}h^2(x_k + kh + 1)^3, \qquad 1 \leqq k \leqq n.$$

The close initial guess $\mathbf{x}_0$ has components $x_k = kh(kh-1)$; all components of the solution are in $[-0.5, 0]$. Table 4 gives the results for $n = 10$.

For this relatively easy problem, there is no advantage in using ZSVD.

TABLE 4

	$\mathbf{x}_0$	$10\mathbf{x}_0$	$100\mathbf{x}_0$	random
BRENT1	14	27	60	21 (20)
HYBRD1	13	15	42	14 (20)
ZSVD	17	17	40	17 (20)

TABLE 5

	$\mathbf{x}_0$	$10\mathbf{x}_0$	$100\mathbf{x}_0$	random
BRENT1	10	33	78	22 (20)
HYBRD1	14	230	div	30 (20)
ZSVD	14	73	122	27 (20)

TABLE 6

	$\mathbf{x}_0$	$10\mathbf{x}_0$	$100\mathbf{x}_0$	random
BRENT1	6	82	div	38 (20)
HYBRD1	22	717	div	41 (16)
ZSVD	19	148	188	44 (20)

TABLE 7

	$\mathbf{x}_0$	$10\mathbf{x}_0$	$100\mathbf{x}_0$	random
BRENT1	23	div	div	72 (13)
HYBRD1	48	div	div	27 (7)
ZSVD	55	178	321	81 (20)

TABLE 8

	x_0	random
BRENTI	div	div (0)
HYBRID1	52	73 (16)
ZSVD	65	71 (20)

Chebyquad. The standard starting point is $\mathbf{x}_0 = (1/(n+1), 2/(n+1), \cdots, n/(n+1))^T$. Except for the smallest values of n, even a small change in the starting point, if it is far from a solution, can result in a completely different path being taken. The problem is nonetheless illustrative; it is extremely difficult if the initial guess is not good.

Tables 5, 6 and 7 contain the results for $n = 5$, 7 and 9. ZSVD is more robust than the other two routines, which diverged for some of the random guesses and some of the very bad guesses.

Sum of exponentials. A final example has a Jacobian matrix which is of rank 1 at the solution, and of low rank far from the solution.

$$f_i = \exp\left[-\sum_{j=1}^{i} (x_j - 1)^2\right] - 1, \qquad 1 \leqq i \leqq n-1,$$

$$f_n = x_n - 1.$$

The close guess is $\mathbf{x}_0 = (\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})$. For $n = 10$, the results follow. BRENT1 diverged with all guesses. HYBRD1 converged for the standard guess and for 16 out of 20 random guesses. ZSVD converged for all guesses.

REFERENCES

[1] J. L. BLUE, *Solving systems of nonlinear equations*, Bell Laboratories Computing Science Technical Report #50, September, 1976.

[2] ———, *Robust methods for solving systems of nonlinear equations*, Bell Laboratories Computing Science Technical Report # 65, September, 1977.

[3] R. P. BRENT, *Some efficient algorithms for solving systems of nonlinear equations*, SIAM J. Numer. Anal., 10 (1973), pp. 327–344.

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

[5] C. G. BROYDEN, J. E. DENNIS JR. AND J. J. MORÉ, *On the local and superlinear convergence of quasi-Newton methods*, J. Inst. Math. Applic., 12 (1973), pp. 223–245.

[6] R. FLETCHER, *Function minimization without evaluating derivatives—a review*, Comp. J., 8 (1965), pp. 33–41.

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

[8] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Handbook for Automatic Computation, vol. II: Linear Algebra, J. H. Wilkinson and C. Reinsch, eds., Springer-Verlag, New York–Heidelberg–Berlin, 1971.

[9] J. KOWALIK AND M. R. OSBORNE, *Methods for Unconstrained Optimization Problems*, American Elsevier, New York, 1968.

[10] K. LEVENBERG, *A method for the solution of certain nonlinear problems in least squares*, Quart. Appl. Math., 2 (1944), pp. 164–168.

[11] D. W. MARQUARDT, *An algorithm for least squares estimation of nonlinear parameters*, SIAM J. Appl. Math., 11 (1963), pp. 431–441.

[12] J. J. MORÉ, *The Levenberg–Marquardt algorithm: implementation and theory*, Proceedings of the 1977 Dundee Conference on Numerical Analysis.

[13] ———, Argonne National Laboratory, private communication.
[14] M. J. D. POWELL, *A hybrid method for nonlinear equations*, Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London, 1970.
[15] J. J. MORÉ AND M. Y. COSNARD, *Numerical solution of nonlinear equations*, ACM Trans. Math. Software, 5 (1979), pp. 64–85.
[16] M. J. D. POWELL, *A Fortran subroutine for solving systems of nonlinear algebraic equations*, Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London, 1970.
[17] L. SCHRAGE, *A more portable Fortran random number generator*, ACM Trans. Math. Software, 5 (1979), pp. 132–138.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0003 $01.00/0

LARGE DISPLACEMENT CALCULATIONS OF FLEXIBLE PIPELINES BY FINITE ELEMENT AND NONLINEAR PROGRAMMING METHODS*

J. F. BOURGAT†, J. M. DUMAY‡ AND R. GLOWINSKI§

Abstract. We describe in this paper the application of some nonlinear programming and finite element methods to the calculation of the large displacements of inextensible and flexible pipelines. The difficulties related to the inextensibility of the pipelines are treated by introducing an appropriate augmented Lagrangian and then using a special algorithm closely related to alternate direction methods. Static and dynamic calculations are discussed and several numerical applications are presented.

Key words. finite elasticity, nonlinear equality constraints, augmented Lagrangians, Hermite cubic approximations, alternate direction methods, difference methods for initial value problems

CONTENTS

Introduction. In this paper we would like to discuss the application of some *nonlinear programming* methods to the numerical study of a class of *nonlinear mechanical* problems in *finite elasticity*. These problems concern the structural behavior of *flexible* and *inextensible* pipelines; for simplicity we shall suppose that we have a *large displacement*, but *small strain*, situation, i.e., the nonlinearity is *geometric*. Similar and related problems have been considered by many authors, from different points of view (mathematical, computational, etc. . . .) and a list of related references is given in § 10.

In our opinion our specific *augmented Lagrangian* method (directly inspired from Fortin–Glowinski [1, Chap. 3]) is more important than the particular class of problems on which it has been tested and we have the feeling that it has a broad field of applications in finite elasticity, particularly for the numerical simulation of *incompressible* media (in fact this belief has been very recently justified by the promising numerical results obtained by P. Le Tallec and the third author for a class of Mooney–Rivlin incompressible materials).

1. A class of pipeline problems. The increasing development of *off-shore oil exploitation* has strongly motivated the numerical simulation of the related structures. Among these structures, pipelines of various types play an important role, and

* Received by the editors July 3, 1979.
† IRIA-Laboria, Domaine de Voluceau, Rocquencourt, 78150 Le Chesnay, France.
‡ COFLEXIP, 75116 Paris, France.
§ Université Pierre et Marie Curie, L.A. 189, Paris, France. and IRIA-Laboria.

particularly the COFLEXIP pipelines designed and manufactured by the French company COFLEXIP which is the pioneer in this advanced technology. Engineers have been interested in the *static* and *dynamic* behavior of these pipes, by the effects of streams and waves, by the contact problems on the sea bed and other obstacles (on the pipe itself for example), etc.... Figure 1.1 explains some further notation associated with the problem to follow.

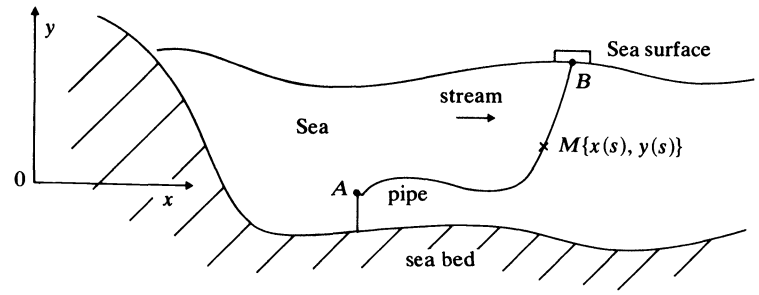


FIG. 1.1. *A,B: extremities of the pipe; s: curvilinear abscissa; s(A) = 0, s(B) = L (L: length of the pipe). M(s): generic point of the pipe with coordinates x(s), y(s).*

1.1. Simplifying hypotheses. For simplicity, but also because it provides interesting preliminary results on the behavior of the pipe we suppose that:

(i) *torsional effects are neglected*;
(ii) *the pipe is inextensible*;
(iii) *the pipe diameter is small with respect to the length L,*
(iv) *we only consider two-dimensional displacements of the pipe*;
(v) *the pipe is flexible and therefore can handle large displacements while still obeying a linear strain-stress relation.*

Remark 1.1. With the numerical methods described below we can simulate the effects of streams (see § 7), waves, contact with obstacles (possibly nonhorizontal) and also solve time dependent (dynamic) problems (see § 8).

2. Mathematical modeling of the static problem. We only consider in this section *static* problems; we suppose also that there is *no stream* acting on the pipe (the static problem *with stream* is considered in § 7). Considering the pipe as a nonlinear *beam* we have from the *inextensibility condition*

$$x'^2 + y'^2 = 1 \quad \text{on } [0, L] \tag{2.1}$$

(where $x' = dx/ds$, $y' = dy/ds$, $x'' = d^2x/ds^2$, $y'' = d^2y/ds^2$), that the displacement fields corresponding to stable equilibrium positions are solutions of the *local minimization problem*

$$\operatorname*{Min\,loc}_{\{x,y\}\in\mathscr{E}} \left\{ \frac{EI}{2} \int_0^L (x''^2 + y''^2)\, ds + \rho g \int_0^L y\, ds \right\}, \tag{2.2}$$

where in (2.2),

(i) $EI(>0)$ is the *flexural stiffness* of the pipe;
(ii) g is the *gravity acceleration*; and

(iii) ρ is the *linear density* of the pipe. If the pipe lies *under water*, to take into account the *hydrostatic pressure* we have to use ρ defined by

$$\rho = \rho_0 - \sigma\rho_w,$$

where ρ_0, σ are the *intrinsic linear density* and the *cross-section* of the pipe, respectively, and where ρ_w is the *volumic density* of the water.

Finally the *local minimizers* we are seeking have to be found in the (*nonconvex*) set $\mathscr{E}$ defined by

$$\mathscr{E} = \{\{x, y\} | x, y \in C^1[0, L],\ x'', y'' \in L^2(0, L), \text{ plus convenient boundary conditions, and } x'^2 + y'^2 = 1 \text{ on } [0, L]\}. \tag{2.3}$$

Remark 2.1. The *nonlinear, nonquadratic, nonconvex* character of the local minimization problem (2.2) is clear from (2.3).

Remark 2.2. With the numerical methods to be described in the following, we can handle cases in which EI and/or ρ are functions of s and where more complicated external forces and energy terms are present in the functional to be minimized.

3. Mathematical analysis of the static problem. The *mathematical analysis* of problems like (2.2) is by itself a fascinating subject which goes back to Euler (the *elastica problem*); however since this work is mostly computationally oriented we shall limit our analysis to a very simple *existence theorem* and to some comments on *nonuniqueness* properties; for a very complete mathematical analysis of related problems we refer to Antman–Rosenfeld [2] (see also Benjamin [3]). We suppose that the boundary conditions are given either by

$$\begin{aligned} x(0) &= x_A, \qquad y(0) = y_A, \\ x(L) &= x_B, \qquad y(L) = y_B \end{aligned} \tag{3.1}$$

with x_A, y_A, x_B, y_B given, or by

$$\begin{aligned} x(0) &= x_A, \quad y(0) = y_A, \quad x'(0) = \alpha_0, \quad y'(0) = \beta_0, \\ x(L) &= x_B, \quad y(L) = y_B, \quad x'(L) = \alpha_L, \quad y'(L) = \beta_L, \end{aligned} \tag{3.2}$$

where, in (3.2), x_A, y_A, x_B, y_B, α_0, β_0, α_L, β_L are given with $\alpha_0^2 + \beta_0^2 = 1$, $\alpha_L^2 + \beta_L^2 = 1$. Let us prove the following about existence properties.

THEOREM 3.1. *Suppose that* $|\overrightarrow{AB}| < L$ *and that* (3.1) *or* (3.2) *holds, then the minimization problem* (2.2) *has at least one solution.*

Proof. Since $|\overrightarrow{AB}| < L$, the set $\mathscr{E}$ defined by (2.3), with (3.1) or (3.2), is not empty. Let $\{x_n, y_n\}_n$ be a minimizing sequence; then we clearly have some *constant* C such that

$$\|x_n\|_{C^1[0,L]} \leqq C, \qquad \|y_n\|_{C^1[0,L]} \leqq C \quad \forall n \geqq 0, \tag{3.3}$$

$$\|x_n''\|_{L^2(0,L)} \leqq C, \qquad \|y_n''\|_{L^2(0,L)} \leqq C \quad \forall n. \tag{3.4}$$

It follows from (3.3), (3.4) that $\{x_n\}_n$, $\{y_n\}_n$ are bounded in $H^2(0, L)$; then the *compactness* of the embedding of $H^2(0, L)$ in $C^1[0, L]$ implies the existence of a subsequence—still denoted by $\{x_n, y_n\}_n$—and of $\{\bar{x}, \bar{y}\} \in H^2(0, L) \times H^2(0, L)$ such that

$$\{x_n, y_n\} \to \{\bar{x}, \bar{y}\} \quad \textit{strongly in } C^1[0, L] \times C^1[0, L], \tag{3.5}$$

$$\{x_n'', y_n''\} \to \{\bar{x}'', \bar{y}''\} \quad \textit{weakly in } L^2(0, L) \times L^2(0, L). \tag{3.6}$$

Since $\{x_n, y_n\} \in \mathscr{E}$, $\forall n$, it follows from (2.3), (3.1), (3.2), (3.5), (3.6) that $\{\bar{x}, \bar{y}\} \in \mathscr{E}$. It follows also from (3.5), (3.6) and from the *weak semi-continuity*, on $H^2(0, L) \times$

$H^2(0, L)$, of the functional $J : H^2(0, L) \times H^2(0, L) \to \mathbb{R}$ defined by

$$J(x, y) = \frac{EI}{2} \int_0^L (x''^2 + y''^2)\, ds + \rho g \int_0^L y\, ds, \tag{3.7}$$

that

$$J(\bar{x}, \bar{y}) \leqq \underline{\lim}\, J(x_n, y_n) = \inf_{\{x,y\} \in \mathscr{E}} J(x, y). \tag{3.8}$$

Clearly, (3.8) and $\{\bar{x}, \bar{y}\} \in \mathscr{E}$ imply that $J(\bar{x}, \bar{y}) = \inf_{\{x, y\} \in \mathscr{E}} J(x, y)$; this completes the proof of the theorem. □

Remark 3.1. If (3.1) holds and if $|\overrightarrow{AB}| = L$, then (2.2) clearly has a *unique solution* which is given by

$$\bar{x}(s) = x_A + \frac{s}{L}(x_B - x_A), \qquad \bar{y}(s) = y_A + \frac{s}{L}(y_B - y_A).$$

If $|\overrightarrow{AB}| = L$ and if $\mathscr{E}$ is defined from (3.2), then $\mathscr{E}$ is *empty*, in general.

Remark 3.2. *Existence results* similar to those obtained above, can be proved by similar methods for *boundary conditions* different from (3.1) or (3.2).

3.1. Comments on the nonuniqueness of the solutions. If $EI = 0$ in $J(\cdot, \cdot)$ (see (3.7)) and if the boundary conditions are given by (3.1), then (2.2) has a unique solution corresponding to a *catenoid* curve (we suppose of course that $|\overrightarrow{AB}| \leqq L$). If $\rho g = 0$ in (3.7), then (2.2) reduces to the *elastica problem* considered by Euler; if the boundary conditions are given by (3.1), then to each solution of (2.2) corresponds the solution obtained by symmetry with respect to the line AB. If $EI > 0$ and if $\rho g > 0$, then we also have *nonuniqueness* in general, and it will be interesting to study the *branches of solutions*, their *limit points* and *bifurcation points*, using for example, the methods of [2].

To illustrate this nonuniqueness of the solutions of (2.2) let us consider, for example, the particular problem (2.2) corresponding to

$$\begin{aligned} &x(0) = x_A = 0, \qquad y(0) = y_A = 0, \\ &x(L) = x_B = (1 - \lambda)L \quad \textit{with} \quad -1 \leqq \lambda \leqq 1, \qquad y(L) = y_B = 0, \\ &x'(0) = 1, \qquad y'(0) = 0, \, x'(L) = 1, \qquad y'(L) = 0. \end{aligned} \tag{3.9}$$

If $\lambda = 0$, problem (2.2) has clearly a unique solution which is given by

$$\bar{x}(s) = s, \qquad \bar{y}(s) = 0; \tag{3.10}$$

if $\rho g = 0$ we have two branches of *stable* solutions, starting from the solution (3.10) as indicated by Fig. 3.1.

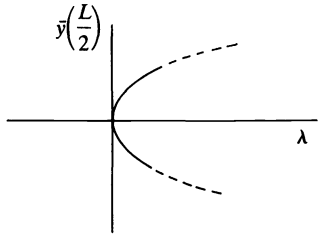


FIG. 3.1

In order to study the behavior of the solutions in the neighborhood of $\lambda = 0$, if $\rho g \neq 0$, we use the standard transformation

$$x' = \cos\phi, \qquad y' = \sin\phi \tag{3.11}$$

which reduces the above particular problem (2.2) to

$$\operatorname*{Min\,loc}_{\phi \in \mathscr{F}} \left\{ \frac{EI}{2} \int_0^L |\phi'|^2 \, ds + \rho g \int_0^L \left(\frac{L}{2} - s \right) \sin\phi \, ds \right\}, \tag{3.12}$$

with

$$\mathscr{F} = \left\{ \phi \in H_0^1(0, L), \int_0^L \cos\phi \, ds = (1-\lambda)L, \int_0^L \sin\phi \, ds = 0 \right\}. \tag{3.13}$$

This problem (3.12), (3.13) can be studied by the methods of [2]; since our goal is more modest and since we are mostly interested in the behavior of the solutions in the neighborhood of the solution (3.10), we shall restrict our analysis to a *second-order* approximation of the problem (3.12), (3.13). The formulation of this approximate problem is

$$\operatorname*{Min\,loc}_{\phi \in \mathscr{F}^*} \left\{ \frac{EI}{2} \int_0^L |\phi'|^2 \, ds + \rho g \int_0^L \left(\frac{L}{2} - s \right) \phi \, ds \right\} \tag{3.14}$$

with

$$\mathscr{F}^* = \left\{ \phi \in H_0^1(0, L), \int_0^L \phi^2 \, ds = 2\lambda L, \int_0^L \phi \, ds = 0 \right\}. \tag{3.15}$$

Since all the functionals occuring in (3.14), (3.15) are C^1 over $H_0^1(0, L)$, it follows easily from, e.g., Berger [4, § 3.1 F] that to each solution ψ of the minimization problem (3.14), (3.15) we can associate two *Lagrange multipliers* γ and δ ($\in \mathbb{R}$) such that the triple $\{\psi, \gamma, \delta\}$ obeys

$$-EI\psi'' + \gamma\psi = -\rho g \left(\frac{L}{2} - s \right) - \delta \quad \text{on } (0, L), \tag{3.16}$$

$$\psi(0) = \psi(L) = 0, \tag{3.17}$$

$$\int_0^L \psi^2 \, ds = 2\lambda L, \tag{3.18}$$

$$\int_0^L \psi \, ds = 0. \tag{3.19}$$

In fact (3.16)–(3.19) characterizes ψ as a *stationary point* on $\mathscr{F}^*$ of the functional $j : H^1(0, L) \to \mathbb{R}$ defined by

$$j(\phi) = \frac{EI}{2} \int_0^L |\phi'|^2 \, ds + \rho g \int_0^L \left(\frac{L}{2} - s \right) \phi \, ds; \tag{3.20}$$

it does not guarantee that ψ is a local minimizer.

Let us consider the solution of the system (3.16)–(3.19); assuming that γ and δ are known it is fairly easy to compute ψ from (3.16), (3.17) and then adjust γ and δ to satisfy (3.18), (3.19). In all the discussion which follows we suppose that $\lambda > 0$.

3.2. Hypothesis $\rho g = 0$. The analysis of this situation in which *gravity has been neglected*, is reduced to the discussion of a *linear eigenvalue-eigenfunction problem.*

3.2.1. Case 1: $\gamma > 0$. It is convenient to set $\gamma = \omega^2$ ($\omega > 0$); if $\delta \neq 0$, it follows from the *maximum principle* that the unique solution of (3.16), (3.17) has the sign of $-\delta$ on $(0, L)$; thus (3.19) cannot be satisfied. If $\delta = 0$, then $\psi = 0$ is the unique solution of (3.16), (3.17) and (3.18) cannot be satisfied if $\lambda > 0$.

3.2.2. Case 2: $\gamma < 0$. We set this time $\gamma = -\omega^2$ ($\omega > 0$); then the solution of (3.16), (3.17) is given by

$$\psi_{\gamma,0}(s) = A \sin n\pi \frac{s}{L}, \qquad n \text{ integer} \geqq 1 \quad \text{if } \delta = 0, \tag{3.21}$$

and

$$\psi_{\gamma,\delta}(s) = \frac{\delta}{\omega^2}\left(1 - \cos\frac{\omega}{\sqrt{EI}}s - tg\frac{\omega}{\sqrt{EI}}\frac{L}{2}\sin\frac{\omega}{\sqrt{EI}}s\right) \quad \text{if } \delta \neq 0. \tag{3.22}$$

Using (3.18), (3.19) we obtain that in (3.21), $n = 2m$, m integer $\geqq 1$, $A = \pm 2\sqrt{\lambda}$. Using (3.19) we have that, in (3.22), ω has to be a solution (positive for example) of

$$\frac{\omega L}{\sqrt{EI}} = 2tg\frac{\omega}{\sqrt{EI}}\frac{L}{2}. \tag{3.23}$$

The positive solutions of (3.23) make a sequence $\{\omega_{2n}\}_{n \geqq 1}$ with

$$\omega_{2n} = \frac{\sqrt{EI}}{L}\xi_n, \tag{3.24}$$

where $\{\xi_n\}_{n \geqq 1}$ is the sequence of the positive solutions of

$$\xi = 2tg\frac{\xi}{2}. \tag{3.25}$$

We have (see Fig. 3.2) that

$$\begin{aligned} &2n\pi < \xi_n < (2n+1)\pi, \\ &\lim_{n \to +\infty} ((2n+1)\pi - \xi_n) = 0; \end{aligned} \tag{3.26}$$

once ω_{2n} is known, adjusting δ to satisfy (3.18) is an easy task which produces δ_{2n} and $-\delta_{2n}(\delta_{2n} > 0)$.

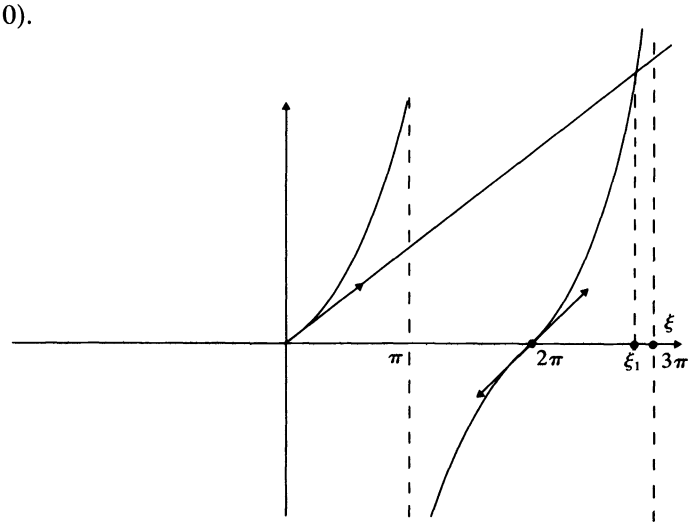


FIG. 3.2. *Solutions of* $\xi = 2tg(\xi/2)$.

3.2.3. Recapitulation. If $\rho g = 0$, the local minimization problem (3.14) has *only two solutions*

$$\psi_1^+(s) = 2\sqrt{\lambda}\sin 2\pi\frac{s}{L}, \qquad \psi_1^-(s) = -2\sqrt{\lambda}\sin 2\pi\frac{s}{L} \tag{3.27}$$

which in fact are *global minimizers*. In addition to $\psi_1^{\pm}$, the functional j possesses an infinity of *stationary solutions on* $\mathscr{F}^*$ (of *saddle-point* type), given by

$$\psi_{2n-1}^{\pm}(s) = \pm 2\sqrt{\lambda}\sin 2n\pi\frac{s}{L}, \qquad n \geqq 2, \tag{3.28}$$

$$\psi_{2n}^{\pm}(s) = \pm\frac{\delta_{2n}}{\omega_{2n}^2}\left(1 - \cos\xi_n\frac{s}{L} - \frac{\xi_n}{2}\sin\xi_n\frac{s}{L}\right), \qquad n \geqq 1. \tag{3.29}$$

3.3. Hypothesis $\rho g > 0$. As will be seen below, *gravity* effects will complicate the structure of the solutions of (3.16)–(3.19), compared to the *nongravity* case discussed above.

3.3.1. Case 1: $\gamma > 0$. Setting again $\gamma = \omega^2$ ($\omega > 0$) we can easily show, since $\lambda > 0$, that (3.16)–(3.19) has no solution corresponding to $\delta \neq 0$. Taking $\delta = 0$ in (3.16) we obtain as a solution of (3.16), (3.17), (3.19),

$$\psi_{\gamma,0}(s) = \frac{\rho g}{\omega^2}\left\{\frac{L}{2}\frac{sh(\omega/\sqrt{EI})((L/2)-s)}{sh(\omega/\sqrt{EI})(L/2)} - \left(\frac{L}{2}-s\right)\right\}. \tag{3.30}$$

3.3.2. Case 2: $\gamma < 0$. We set $\gamma = -\omega^2$ ($\omega > 0$); with regards to the solutions of (3.16), (3.17), (3.19) we can prove the following results.

(i) If $\omega = \omega_{2n-1} = 2n\pi(\sqrt{EI}/L)$, $n \geqq 1$, then (3.16), (3.17), (3.19) *has no solution.*

(ii) If $\omega \neq \omega_n$, $n \geqq 1$, then (3.16), (3.17), (3.19) has a unique solution, for which $\delta = 0$, and which is given by

$$\psi_{\gamma,0}(s) = \frac{\rho g}{\omega^2}\left\{\left(\frac{L}{2}-s\right) - \frac{L}{2}\frac{\sin(\omega/\sqrt{EI})((L/2)-s)}{\sin(\omega/\sqrt{EI})(L/2)}\right\}. \tag{3.31}$$

(iii) If $\omega = \omega_{2n}$, $n \geqq 1$, then (3.16), (3.17), (3.19) has a unique solution (with $\gamma_{2n} = -\omega_{2n}^2$) given by

$$\begin{aligned}\psi_{\gamma_{2n},\delta}(s) = {} & \frac{\rho g}{\omega_{2n}^2}\left\{\left(\frac{L}{2}-s\right) - \frac{L}{2}\frac{\sin(\omega_{2n}/\sqrt{EI})((L/2)-s)}{\sin(\omega_{2n}/\sqrt{EI})(L/2)}\right\} \\ & + \frac{\delta}{\omega_{2n}^2}\left(1 - \cos\frac{\omega_{2n}}{\sqrt{EI}}s - \frac{\omega_{2n}}{\sqrt{EI}}\frac{L}{2}\sin\frac{\omega_{2n}}{\sqrt{EI}}s\right).\end{aligned} \tag{3.32}$$

3.3.3. Determination of ω and δ. Recapitulation. We suppose first that $\delta = 0$; the multiplier γ is obtained from (3.18), (3.30), (3.31) and (3.32) (with $\delta = 0$); in that direction it is convenient to plot, as done in Fig. 3.3, the graph of the function

$$\omega \text{ sign } (\gamma) \to \int_0^L |\psi_{\gamma,0}(s)|^2\, ds.$$

The feasible γ are obtained by taking the intersection of this graph with the horizontal line of ordinate $2\lambda L$ (two positions of this line have been shown on Fig. 3.3)

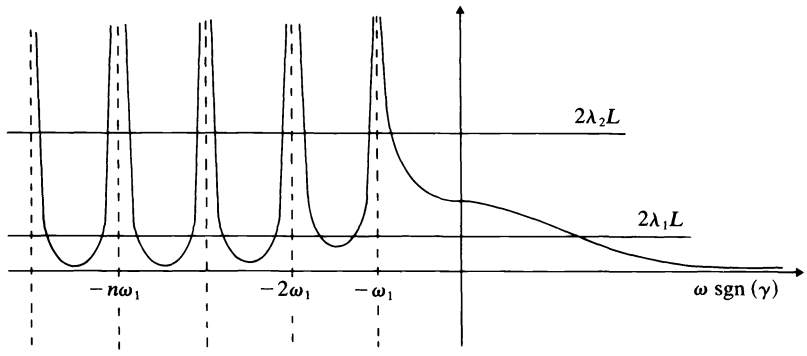


FIG. 3.3. *Determination of* γ.

The poles of the above function are the *negative* multiples of $\omega_1 = 2\pi(\sqrt{EI}/L)$ (see Fig. 3.3). If $\omega = \omega_{2n}$ (situation corresponding to a *bifurcation phenomenon*) the feasible δ are obtained again from (3.18). In (3.32) we see easily that the first (resp. second) component of $\psi_{\gamma_{2n},\delta}$ is an *odd* (resp. *even*) function of $((L/2)-s)$; from these properties these two components are $L^2(0, L)$-*orthogonal.* Hence the graph of the function

$$\delta \to \int_0^L |\psi_{\gamma,\delta}(s)|^2 \, ds$$

is a *parabola*, symmetric with respect to the ordinate axis and strictly above the δ-axis (see Fig. 3.4).

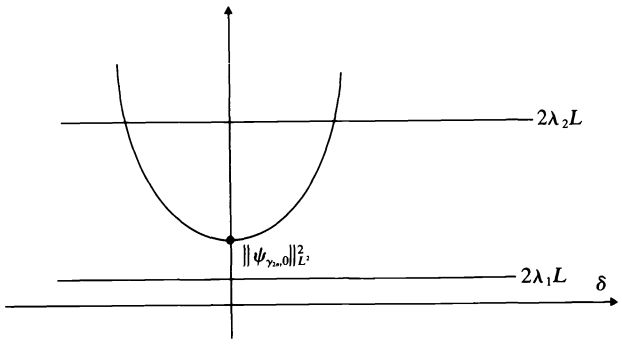


FIG. 3.4. *Determination of* δ.

For λ *sufficiently large* and n *given* we have two solutions δ_1, δ_2 with $\delta_1 = -\delta_2$; this implies that for λ sufficiently large $(\lambda > (1/2L)\|\psi_{\gamma_2,0}\|^2_{L^2(0,L)})$, we have *two stationary solutions* of type (3.32) *for each value of* n, $n \geqq 1$.

It follows from the above analysis that we have a (countable) *infinity* of stationary points of the functional j, $\forall \lambda > 0$; it follows also that we have an infinity of bifurcation points. Finally it can be proved that the branch of solutions associated to ω sign $(\gamma) \in (-\omega_1, +\infty)$ corresponds to *global minimizers* of j on $\mathscr{F}^*$.

4. Numerical solution of the static problem. I: Generalities. The numerical solution of problems closely related to (2.2) has been considered by several authors; let us mention among others Hibbit–Becker–Taylor [5], Maier–Andreuzzi–Gianessi–Jurina–Taddei [6].

This problem (2.2) is actually a nontrivial one from a computational point of view, as can be observed by introducing a Lagrangian functional associated to the functional J of (3.7) and to the nonlinear (inextensibility) constraint (2.1)

$$\mathscr{L}(x, y, \mu) = J(x, y) + \frac{1}{2}\int_0^L \mu(x'^2 + y'^2 - 1)\, ds. \tag{4.1}$$

Suppose that a *Lagrange multiplier* function λ exists, corresponding to a local minimizer $\{\bar{x}, \bar{y}\} \in \mathscr{E}$; from the *stationarity* of $\mathscr{L}$ we obtain that the triple $\{\bar{x}, \bar{y}, \lambda\}$ has to satisfy

$$EI\frac{d^4\bar{x}}{ds^4} - \frac{d}{ds}\left(\lambda\frac{d\bar{x}}{ds}\right) = 0 \quad \text{on } (0, L) + \textit{boundary conditions}, \tag{4.2}$$

$$EI\frac{d^4\bar{y}}{ds^4} - \frac{d}{ds}\left(\lambda\frac{d\bar{y}}{ds}\right) = -\rho g \quad \text{on } (0, L) + \textit{boundary conditions}, \tag{4.3}$$

$$\bar{x}'^2 + \bar{y}'^2 - 1 = 0 \quad \text{on } [0, L]. \tag{4.4}$$

It appears from (4.2)–(4.4) that λ can be seen as a *generalized eigenvalue*, with $\{\bar{x}, \bar{y}\}$ the corresponding *generalized eigenvector.*

Since the main difficulty in problem (2.2) lies in the nonlinear inextensibility constraint (2.1), it seems quite natural to overcome it using the transformation $x' = \cos\phi$, $y' = \sin\phi$, where ϕ is clearly the angle formed by the oriented tangent at the pipe, at $M = \{x, y\}$, with Ox, and where $d\phi/ds$ is the *curvature* of the pipe at M (see Fig. 4.1).

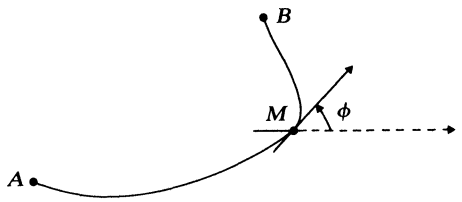


FIG. 4.1.

This transformation, which has been used computationally in related problems (see e.g. [6]) and also in problem (2.2) by M. O. Bristeau and the third author, leads to a second-order differential problem (compared to the fourth-order original problem) which has a more complicated nonlinear structure since it involves transcendental functions, like cos and sin, whose repetitive use may be costly; furthermore, as can be seen on the specific problem (2.2), (3.9) which leads to (3.12), (3.13), boundary conditions of type (3.1), (3.2) on x, y imply *nonlinear integral relations* on ϕ (cf. (3.13)).

Another argument to work directly with x, y and (2.1), instead of ϕ, is that it is a good starting point toward the solution of much more complicated *finite elasticity* problems, related to *incompressible media* in which the domain under consideration is two- or three-dimensional (and in fact, as mentioned in the introduction, the methodology described in § 6 has led to a class of very efficient algorithms for calculating two-dimensional hyperelastic incompressible media of Mooney–Rivlin type).

5. Numerical solution of the static problem. II: Approximation.

5.1. Approximation of $H^2(0, L)$ and J. Since $\mathscr{E}$ is a subset of $H^2(0, L) \times H^2(0, L)$, a most important step toward the numerical solution of (2.2) is to define a convenient

approximation of $H^2(0, L)$; let us introduce $\{s_i\}_{i=0}^N$ such that $s_i \in [0, L]$, $\forall i$, $s_0 = 0$, $s_N = L$ and $s_i < s_{i+1}$, $\forall i = 0, \cdots, N-1$. We then approximate $H^2(0, L)$ by

$$V_h = \{v_h \in C^1[0, L],\ v_h|_{[s_i, s_{i+1}]} \in P_3, \quad \forall i = 0, \cdots, N-1\}, \tag{5.1}$$

where, in general, P_k = space of the polynomials in one variable of degree $\leqq k$; we have $V_h \subset H^2(0, L)$ and dim $V_h = 2(N+1)$. As usual $h = \max_i(s_{i+1} - s_i)$. If $v_h \in V_h$ it is convenient to define it from

$$\{v_h(s_i)\}_{i=0}^N, \qquad \left\{\frac{dv_h}{ds}(s_i)\right\}_{i=0}^N.$$

Using these degrees of freedom it is clear that V_h corresponds to a *finite element approximation of Hermite cubic* type (see Fig. 5.1).

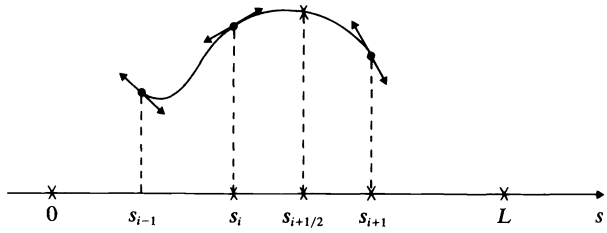


FIG. 5.1.

Since $V_h \times V_h \subset H^2(0, L) \times H^2(0, L)$ we have that the functional J (cf. (3.7) for its definition) is defined on $V_h \times V_h$; moreover, since $x_h, y_h \in V_h$ implies that $(x_h''^2 + y_h''^2)|_{[s_i, s_{i+1}]} \in P_2$, the two integrals occurring in J can be computed *exactly* using *Simpson's formula* on each sub-interval $[s_i, s_{i+1}]$, $i = 0, 1, \cdots, N-1$. By restricting J to $V_h \times V_h$ one obtains a functional of $4(N+1)$ variables.

5.2. Approximation of $\mathscr{E}$. Since we use a *piecewise Hermite cubic* approximation there is no difficulty to approximate *boundary conditions* like those given by (3.1), (3.2). Concerning the *inextensibility condition* (2.1), the obvious choice is to use

$$x_h'^2(s_i) + y_h'^2(x_i) = 1 \quad \forall i = 0, 1, \cdots, N. \tag{5.2}$$

Since $\{x_h'(s_i)\}_{i=0}^N$, $\{y_h'(s_i)\}_{i=0}^N$ are precisely part of the degrees of freedom defining x_h and y_h, the computer implementation of (5.2) is fairly easy. We have observed however that for some *stiff problems* (involving *strong variations of curvature*) we can obtain a poor accuracy using (5.2), unless we use a local refinement of the mesh in the sensitive parts of the pipe. Such a procedure may increase N considerably and with regard to accuracy we have found more convenient to approximate (2.1) by

$$\begin{aligned} &x_h'^2(s_i) + y_h'^2(s_i) = 1 \quad \forall i = 0, 1, \cdots, N, \\ &x_h'^2(s_{i+1/2}) + y_h'^2(s_{i+1/2}) = 1 \quad \forall i = 0, \cdots, N-1, \end{aligned} \tag{5.3}$$

where $s_{i+1/2} = \frac{1}{2}(s_i + s_{i+1})$ (see Fig. 5.1).

In fact the numerical results presented in § 9 have been obtained using (5.3) to approximate (2.1).

Using (5.2) (resp. (5.3)) to approximate (2.1), introduces about N (resp. $2N$) *quadratic equality constraints* (the exact number depends upon the boundary conditions).

In the following, the approximation of $\mathscr{E}$ obtained by approximating (2.1) by either (5.2) or (5.3) will be denoted by $\mathscr{E}_h$; it is obvious that $\mathscr{E}_h$ is a *closed* subset of $V_h \times V_h$.

5.3. Approximation of the minimization problem (2.2). From the considerations developed in §§ 5.1, 5.2 we shall approximate (2.2) by

$$\text{Min loc}_{\{x_h, y_h\} \in \mathscr{E}_h} J(x_h, y_h), \tag{5.4}$$

where

$$J(x_h, y_h) = \frac{EI}{2} \int_0^L (x_h''^2 + y_h''^2)\, dx + \rho g \int_0^L y_h\, ds. \tag{5.5}$$

Concerning the existence of solutions for the approximate problem (5.4) we have the following discrete variant of Theorem 3.1.

PROPOSITION 5.1. *Assume that $\mathscr{E}_h$ is not empty (a sufficient condition is $|\overrightarrow{AB}| < L$ if the boundary conditions are given by* (3.1) *or* (3.2)); *then problem* (5.4) *has at least one solution.*

Proof. Since $\mathscr{E}_h$ is not empty we can introduce a minimizing sequence $(\{x_h^n, y_h^n\})_n$ for J on $\mathscr{E}_h$, i.e.,

$$\{x_h^n, y_h^n\} \in \mathscr{E}_h \quad \forall n$$

and

$$\lim_{n \to +\infty} J(x_h^n, y_h^n) = \operatorname*{Inf}_{\{x_h, y_h\} \in \mathscr{E}_h} J(x_h, y_h).$$

Since $\mathscr{E}_h$ is a *closed* subset in the *finite dimensional* space $V_h \times V_h$, and since J is *continuous*, if we can prove that $(\{x_h^n, y_h^n\})_n$ is *bounded*, the result to be proved can be obtained by elementary *compactness* arguments. Let us prove, therefore, the *boundedness* of $(\{x_h^n, y_h^n\})_n$

We have $\forall s \in [s_i, s_{i+1}]$, $\forall i = 0, \cdots, N-1$,

$$x_h'(s) = x_h'(s_i) + \int_{s_i}^s x_h''(\sigma)\, d\sigma$$

which implies, combined with (5.2) or (5.3) and using Schwarz's inequality,

$$|x_h'(s)| \leqq 1 + h^{1/2} \|x_h''\|_{L^2(0,L)} \quad \forall s \in [0, L], \quad \forall \{x_h, y_h\} \in \mathscr{E}_h.$$

We similarly have

$$|y_h'(s)| \leqq 1 + h^{1/2} \|y_h''\|_{L^2(0,L)} \quad \forall s \in [0, L], \quad \forall \{x_h, y_h\} \in \mathscr{E}_h.$$

We have, on the other hand,

$$x_h(s) = x_h(0) + \int_0^s x_h'(\sigma)\, d\sigma,$$

$$y_h(s) = y_h(0) + \int_0^s y_h'(\sigma)\, d\sigma.$$

If the boundary conditions are given by (3.1) or (3.2) we have $x_h(0) = x_A$, $y_h(0) = y_A$, $\forall h$, and from the above estimates on $|x_h'(s)|$, $|y_h'(s)|$ we obtain

$$|x_h(s)| \leqq |x_A| + L(1 + h^{1/2} \|x_h''\|_{L^2(0,L)}) \quad \forall s \in [0, L], \quad \forall \{x_h, y_h\} \in \mathscr{E}_h,$$

$$|y_h(s)| \leqq |y_A| + L(1 + h^{1/2} \|y_h''\|_{L^2(0,L)}) \quad \forall s \in [0, L], \quad \forall \{x_h, y_h\} \in \mathscr{E}_h.$$

It is clear from the above relations that $(\{x_h^n, y_h^n\})_n$ will be bounded if we can prove that $(x_h^n)''$, $(y_h^n)''$ are bounded in $L^2(0, L)$.

We clearly have from the above estimates that

$$\left|\int_0^L y_h(s)\,ds\right| \leqq L|y_A| + L^2(1+h^{1/2}\|y_h''\|_{L^2(0,L)}) \quad \forall \{x_h, y_h\} \mathscr{E}_h$$

which implies that

$$J(x_h, y_h) \geqq \frac{EI}{2}\int_0^L (x_h''^2 + y_h''^2)\,ds - \rho g L|y_A| - \rho g L^2(1+h^{1/2}\|y_h''\|_{L^2(0,L)}) \qquad \forall \{x_h, y_h\} \in \mathscr{E}_h.$$

Using the above inequality it is then very easy to prove that for the minimizing sequence $(\{x_h^n, y_h^n\})_n$ we have $(x_h^n)''$, $(y_h^n)''$ bounded in $L^2(0, L)$.

5.4. Convergence of the approximate solutions. We shall prove in this subsection several results concerning the convergence, as $h \to 0$, of the solutions of the approximate problem (5.4). The following lemma will play an important role in the proofs of these results.

LEMMA 5.1. *Suppose that* $\forall h$, $\{x_h, y_h\} \in V_h \times V_h$ *and satisfies either* (5.2) *or* (5.3); *suppose also that*

$$\lim_{h\to 0} \{x_h, y_h\} = \{x, y\} \quad \textit{weakly in } H^2(0, L) \times H^2(0, L); \tag{5.6}$$

then

$$x'^2 + y'^2 = 1 \quad \textit{on } [0, L]. \tag{5.7}$$

Proof. It follows from (5.6) that there exists a constant C such that

$$\|x_h\|_{H^2(0,L)} \leqq C, \qquad \|y_h\|_{H^2(0,L)} \leqq C \quad \forall h. \tag{5.8}$$

It follows also from (5.6) that $\forall s, s' \in [0, L]$ we have

$$|x_h'(s') - x_h'(s)| = \left|\int_s^{s'} x_h''(\sigma)\,d\sigma\right| \leqq |s'-s|^{1/2}\|x_h''\|_{L^2(0,L)} \leqq C|s'-s|^{1/2} \quad \forall h. \tag{5.9}$$

We similarly have

$$|y_h'(s') - y_h'(s)| \leqq C|s'-s|^{1/2} \quad \forall h. \tag{5.10}$$

If $\{x_h, y_h\}$ obeys (5.2) and if $s \in [s_i, s_{i+1}]$, $i = 0, \cdots, N-1$, we clearly have from (5.9), (5.10) that

$$||x_h'(s)| - |x_h'(\sigma_i)|| \leqq C\left(\frac{h}{2}\right)^{1/2} \quad \forall h, \tag{5.11}$$

$$||y_h'(s)| - |y_h'(\sigma_i)|| \leqq C\left(\frac{h}{2}\right)^{1/2} \quad \forall h, \tag{5.12}$$

where $\sigma_i = s_i$ if $s \in [s_i, s_{i+1/2})$ and $\sigma_i = s_{i+1}$ if $s \in [s_{i+1/2}, s_{i+1}]$. If $\{x_h, y_h\}$ obeys (5.3) and if $s \in [s_i, s_{i+1}]$, $i = 0, \cdots, N-1$, we have

$$||x_h'(s)| - |x_h'(\sigma_i)|| \leqq \frac{C}{2}h^{1/2} \quad \forall h, \tag{5.13}$$

$$||y_h'(s)| - |y_h'(\sigma_i)|| \leqq \frac{C}{2}h^{1/2} \quad \forall h, \tag{5.14}$$

where (setting $h_i = s_{i+1} - s_i$) $\sigma_i = s_i$ if $s \in [s_i, s_i + h_i/4)$, $\sigma_i = s_{i+1/2}$ if $s \in$

$[s_i + h_i/4, s_{i+1} - h_i/4]$, $\sigma_i = s_{i+1}$ if $s \in (s_{i+1} - h_i/4, s_{i+1}]$. It follows from (5.6) that

$$\lim_{h \to 0} \|x_h' - x'\|_{C^0[0,L]} = 0, \tag{5.15}$$

$$\lim_{h \to 0} \|y_h' - y'\|_{C^0[0,L]} = 0. \tag{5.16}$$

Since (5.8) implies that $\forall h$ and $\forall s \in [0, L]$ we have

$$|x_h'(s)| \leqq C\left(L + \frac{1}{L}\right)^{1/2}, \qquad |y_h'(s)| \leqq C\left(L + \frac{1}{L}\right)^{1/2} \tag{5.17}$$

We finally have from (5.2), (5.3), (5.11)–(5.17) that

$$x'^2 + y'^2 = \lim_{h \to 0} \{x_h'^2(s) + y_h'^2(s)\} = 1 \quad \forall s \in [0, L]. \quad \square$$

Remark 5.1. Since (5.3) implies (5.2), we can prove (5.7) if (5.3) holds, using only (5.11), (5.12). We have found it interesting to show that using (5.3) produces a better approximation of the inextensibility constraint (2.1), since the constants occcuring in (5.13), (5.14) are smaller than those in (5.11), (5.12). Numerical experiments confirm that (5.3) is a better approximation of (2.1) than (5.2). $\square$

We now introduce the concept of *isolated solution* of the local minimization problem (2.2).

DEFINITION 5.1. Let $\{\bar{x}, \bar{y}\}$ be a solution of (2.2); we say that $\{\bar{x}, \bar{y}\}$ is an *isolated solution* of (2.2) if there exists a neighborhood η of $\{\bar{x}, \bar{y}\}$ such that

$$\begin{aligned} &J(\bar{x}, \bar{y}) < J(x, y) \quad \forall \{x, y\} \in \eta \cap \mathscr{E}, \\ &\{x, y\} \neq \{\bar{x}, \bar{y}\}. \end{aligned} \tag{5.18}$$

We then have the convergence results given by

THEOREM 5.1. *Suppose that the boundary conditions occuring in the definition of* $\mathscr{E}$ *are given by* (3.1) *or* (3.2). *Suppose also that* $\mathscr{E}_h$ *is defined from* (5.2), *then*:

If $\{\bar{x}, \bar{y}\}$ *is an isolated solution of* (2.2), *then for h sufficiently small the approximate problem* (5.4) *has a solution* $\{\bar{x}_h, \bar{y}_h\}$ *in the neighborhood of* $\{\bar{x}, \bar{y}\}$; *we have moreover*

$$\lim_{h \to 0} \{\bar{x}_h, \bar{y}_h\} = \{\bar{x}, \bar{y}\} \quad \textit{strongly in } H^2(0, L) \times H^2(0, L). \tag{5.19}$$

Proof. Let $\{\bar{x}, \bar{y}\}$ be an isolated solution of (2.2); there exists then $\delta > 0$ such that

$$J(\bar{x}, \bar{y}) < J(x, y) \quad \forall \{x, y\} \in \bar{B}_\delta \cap \mathscr{E}, \quad \{x, y\} \neq \{\bar{x}, \bar{y}\}, \tag{5.20}$$

where in (5.20), $\bar{B}_\delta$ denotes the closed ball of $H^2(0, L) \times H^2(0, L)$ of center $\{\bar{x}, \bar{y}\}$ and radius δ.

We now consider the *finite dimensional* problem

$$\text{Min } J(x_h, y_h), \qquad \{x_h, y_h\} \in \bar{B}_\delta \cap \mathscr{E}_h. \tag{5.21}$$

Using *compactness arguments* it can be easily proved that (5.21) has at least one solution $\{\bar{x}_h, \bar{y}_h\}$.

Let π_h be the *interpolation operator* defined on $C^1[0, L]$ by

$$\pi_h v \in V_h \quad \forall v \in C^1[0, L],$$
$$\pi_h v(s_i) = v(s_i) \quad \forall i = 0, \cdots, N,$$
$$(\pi_h v)'(s_i) = v'(s_i) \quad \forall i = 0, \cdots, N.$$

If $\mathscr{E}_h$ is defined from (5.2) we clearly have

$$\{\pi_h x, \pi_h y\} \in \mathscr{E}_h \quad \forall \{x, y\} \in \mathscr{E}. \tag{5.22}$$

On the other hand we have from standard results on *finite element approximation* (see, e.g., Strang–Fix [7], Oden–Reddy [8], Ciarlet [9]) that

$$\lim_{h \to 0} \{\pi_h x, \pi_h y\} = \{x, y\} \quad \text{strongly in } H^2(0, L) \times H^2(0, L) \quad \forall \{x, y\} \in H^2(0, L) \times H^2(0, L). \tag{5.23}$$

It follows, in particular, from (5.22), (5.23) that

$$\{\pi_h \bar{x}, \pi_h \bar{y}\} \in \bar{B}_\delta \cap \mathscr{E}_h \quad \text{for } h \text{ sufficiently small} \tag{5.24}$$

and also that

$$\lim_{h \to 0} J(\pi_h \bar{x}, \pi_h \bar{y}) = J(\bar{x}, \bar{y}). \tag{5.25}$$

Let us now consider the behavior of $(\{\bar{x}_h, \bar{y}_h\})_h$ as $h \to 0$; since this family is *bounded* in $H^2(0, L) \times H^2(0, L)$ there exists a subsequence—still denoted by $(\{x_h, y_h\})_h$—and $\{x^*, y^*\} \in H^2(0, L) \times H^2(0, L)$ such that

$$\{\bar{x}_h, \bar{y}_h\} \to \{x^*, y^*\} \quad \text{weakly in } H^2(0, L) \times H^2(0, L). \tag{5.26}$$

Since $\bar{B}_\delta$ is a closed ball of $H^2(0, L) \times H^2(0, L)$ and from Lemma 5.1 we have, from (5.26), that

$$\{x^*, y^*\} \in \bar{B}_\delta \cap \mathscr{E}. \tag{5.27}$$

We have on the other hand, from (5.24), that

$$J(\bar{x}_h, \bar{y}_h) \leqq J(\pi_h \bar{x}, \pi_h \bar{y}) \quad \text{for } h \text{ sufficiently small.} \tag{5.28}$$

From (5.23), (5.25), (5.26) and also from the *continuity* and *convexity* of J over $H^2(0, L) \times H^2(0, L)$, we have at the limit in (5.28),

$$J(x^*, y^*) \leqq \liminf J(\bar{x}_h, \bar{y}_h) \leqq \limsup J(\bar{x}_h, \bar{y}_h) \leqq J(\bar{x}, \bar{y}). \tag{5.29}$$

Combined with (5.20), (5.27), the above relation (5.29) implies that $\{x^*, y^*\} = \{\bar{x}, \bar{y}\}$; therefore the *whole* $(\{\bar{x}_h, \bar{y}_h\})_h$ converges to $\{\bar{x}, \bar{y}\}$ with

$$\lim_{h \to 0} J(\bar{x}_h, \bar{y}_h) = J(\bar{x}, \bar{y}). \tag{5.30}$$

Let us now prove that the *strong convergence* of $(\{\bar{x}_h, \bar{y}_h\})_h$ to $\{\bar{x}, \bar{y}\}$; we have

$$J(\bar{x}_h, \bar{y}_h) = \frac{EI}{2} \int_0^L (\bar{x}_h''^2 + \bar{y}_h''^2)\, ds + \rho g \int_0^L \bar{y}_h\, ds. \tag{5.31}$$

Since $\lim_{h\to 0}\int_0^L \bar{y}_h\,ds = \int_0^L \bar{y}\,ds$ we have from (5.30), (5.31) that

$$\lim_{h\to 0}\int_0^L (\bar{x}_h''^2+\bar{y}_h''^2)\,ds = \int_0^L (\bar{x}''^2+\bar{y}''^2)\,ds$$

which combined with (5.26) implies the *strong convergence* in $H^2(0,L)\times H^2(0,L)$. From this strong convergence property we have that *for h sufficiently small* $\{\bar{x}_h, \bar{y}_h\}$ *belongs to the interior of* $\bar{B}_\delta$, and therefore is a *local minimizer* for J on $\mathscr{E}_h$. This completes the proof of the theorem. □

Using similar techniques we can also prove the following

THEOREM 5.2. *Suppose that* $\mathscr{E}$ *and* $\mathscr{E}_h$ *are the same as in the statement of Theorem* 5.1. *Then if* $(\{\bar{x}_h, \bar{y}_h\})_h$ *is a family of global minimizers of* J *upon* $\mathscr{E}_h$ *we have (at least for a subsequence)*

$$\lim_{h\to 0}\{\bar{x}_h, \bar{y}_h\} = \{\bar{x}, \bar{y}\} \quad \textit{strongly in } H^2(0,L)\times H^2(0,L),$$

where $\{\bar{x}, \bar{y}\}$ *is a global minimizer of* J *upon* $\mathscr{E}$.

Remark 5.2. The proof of the convergence if $\mathscr{E}_h$ is defined from (5.3) is much more difficult and has not been considered here.

We have also omitted the analysis of the behavior of the approximate solutions in the neighborhood of *turning points* and *genuine bifurcation points*; in that direction let us mention the work of F. Kikuchi [10], Yamaguti–Fujii [11], Kesavan [12], Brezzi–Rappaz–Raviart [13].

6. Numerical solution of the static problem. III: Iterative solution.

6.1. Generalities and synopsis. The *equality constraint* (2.1) (or its discrete variants (5.2), (5.3)) is the major difficulty to overcome if one wishes to solve numerically the local minimization problem (2.2) working directly with the displacements x, y. Roughly speaking, to solve (2.2) and its discrete variants we have the two following classes of methods.

6.1.1. Multiplier and penalty methods. As seen in § 4 we can associate a *Lagrange multiplier function* λ to the equality constraint (2.1); doing so we have to solve, with respect to $\{\bar{x}, \bar{y}, \lambda\}$, the *nonlinear differential system* (4.2)–(4.4) (in fact its discrete variants). Actually the discrete variants of (4.2)–(4.4) may be solved by the *variable metric methods* developed by Powell [14], which generalize the celebrated Davidon–Fletcher–Powell method (see also Strang [15] for related methods oriented to nonlinear mechanics). We have the feeling that these methods are more delicate to handle than the methods to be described in § 6.2, and also more storage demanding for large scale problems.

It is of course natural to associate to the *Lagrangian* functional $\mathscr{L}$, defined by (4.1), the *augmented Lagrangian* $\mathscr{L}_r$ defined (with $r>0$) by

$$\mathscr{L}_r(x,y,\mu) = \mathscr{L}(x,y,\mu)+\frac{r}{4}\int_0^L (x'^2+y'^2-1)^2\,ds. \tag{6.1}$$

If $\mathscr{L}$ is replaced by $\mathscr{L}_r$, the *stationarity* of $\mathscr{L}_r$ leads to the following variant of (4.2)–(4.4),

$$EI\bar{x}^{(4)} - \frac{d}{ds}\left(\lambda\frac{d\bar{x}}{ds}\right) - r\frac{d}{ds}\left((\bar{x}'^2+\bar{y}'^2-1)\frac{d\bar{x}}{ds}\right) = 0 \tag{6.2}$$

on $(0,L)$ + *boundary conditions*,

$$(6.3)\quad EI\bar{y}^{(4)}-\frac{d}{ds}\left(\lambda\frac{d\bar{y}}{ds}\right)-r\frac{d}{ds}\left((\bar{x}'^2+\bar{y}'^2-1)\frac{d\bar{y}}{ds}\right)=-\rho g$$

on $(0, L)$ + *boundary conditions*

$$(6.4)\qquad \bar{x}'^2+\bar{y}'^2-1=0 \quad \textit{on } [0, L]$$

which is obviously equivalent to (4.2)–(4.4). Clearly the above augmented Lagrangian approach seems to complicate an already complicated problem, since (6.2)–(6.4) is "*more nonlinear*" than (4.2)–(4.4); moreover (6.2), (6.3) are "*more coupled*" than (4.2), (4.3). If one takes $\lambda=0$ in (6.2), (6.3) and does not consider (6.4) we obtain the necessary optimality conditions for a problem obtained from (2.2) by *penalty*. □

6.1.2. Direct minimization on manifolds. Instead of relaxing (2.1) by *Lagrange multiplier* and/or *penalty* we can try to minimize directly on the manifold defined by (2.1), as it is done in Lichnewsky [16] (by *steepest descent* and *conjugate gradient* methods); however the methods of [16], very elegant in their principle and very effective on many problems, since they basically carry out the minimization along the geodesic curves of the manifold, are rather difficult to use if the number of constraints is very large, which is the case for the discrete variants of (2.2) (see § 5).

The methods that we have used are quite different from the two types of methods mentioned above; they have however common features in that:

(i) They are based also on an *augmented Lagrangian* procedure; but in our case the constraints which are treated by Lagrange multiplier and penalty are *linear*, which is a great simplification.

(ii) We kept the idea of *direct minimization* on a manifold associated (in some sense) with the nonlinear equality constraint (2.1).

6.2. Iterative solution of (2.2) using a decomposition-coordination principle via an augmented Lagrangian. We use in fact an idea, which from a numerical point of view seems to go back to Glowinski–Marrocco [17]; for a general study of this class of methods, in a convex context, we refer to [1] (see also Gabay–Mercier [18]). The basic idea behind these methods is very simple and is to decouple as much as possible *nonlinearities* and *derivatives* by introducing new variables, coupled to the old ones by *linear equality constraints.*

For the present problem the starting point is the obvious

PROPOSITION 6.1. *Problem* (2.2) *is equivalent to*

$$(6.5)\qquad \operatorname*{Min\,loc}_{\{x,y,p,q\}\in\tilde{\mathscr{E}}}\left\{\frac{EI}{2}\int_0^L (x''^2+y''^2)\,ds+\rho g\int_0^L y\,ds\right\}$$

with

$$(6.6)\qquad \tilde{\mathscr{E}}=\{\{x, y, p, q\}\in W\times(L^2(0, L))^2 \mid x'=p,\ y'=q,\ p^2+q^2=1\},$$

where W *is the subspace of* $H^2(0, L)\times H^2(0, L)$ *defined from the boundary conditions specified for* $\{x, y\}$ *in the definition of* $\mathscr{E}$.

The next step is to relax the functional relation between $\{x, y\}$ and $\{p, q\}$ by introducing the following *augmented Lagrangian* (with $r>0$),

$$(6.7)\qquad \begin{aligned}\mathscr{L}_r(x, y, p, q, \lambda, \mu)=&\frac{EI}{2}\int_0^L (x''^2+y''^2)\,ds+\rho g\int_0^L y\,ds+\int_0^L \lambda(p-x')\,ds\\&+\int_0^L \mu(q-y')\,ds+\frac{r}{2}\int_0^L |p-x'|^2\,ds+\frac{r}{2}\int_0^L |q-y'|^2\,ds.\end{aligned}$$

By analogy with the *convex case* we suppose that $\{\bar{x}, \bar{y}, \bar{p}, \bar{q}, \bar{\lambda}, \bar{\mu}\}$ is a (*local*) *saddle-point* for $\mathscr{L}_r$ over $W \times S \times (L^2(0, L))^2$ where

$$S = \{\{p, q\} \in (L^2(0, L))^2,\ p^2 + q^2 = 1 \text{ a.e.}\}. \tag{6.8}$$

We have then that $\{\bar{x}, \bar{y}\} \in \mathscr{E}$ with $\bar{p} = \bar{x}'$, $\bar{q} = \bar{y}'$ and that $\bar{\lambda}, \bar{\mu}$ are Lagrange multipliers for the linear equality constraints $p - x' = 0$, $q - y' = 0$. From that saddle-point property it is then natural to extend to $\mathscr{L}_r$ the following iterative method whose convergence has been proved in the convex case under rather general assumptions (see [1] for more details).

6.2.1. Description of the basic iterative method: Variants. We use an algorithm of Uzawa (see [19, Chap. 2], [20]) to obtain the saddle-points of $\mathscr{L}_r$ over $W \times S \times (L^2(0, L))^2$. *If*

$$\lambda^0, \mu^0 \text{ are given} \tag{6.9}$$

then for $n \geqq 0$, *assuming that* λ^n, μ^n *are known, we compute* $x^n, y^n, p^n, q^n, \lambda^{n+1}, \mu^{n+1}$ *by*:

$$\begin{aligned} &\text{find } \{x^n, y^n, p^n, q^n\} \in W \times S \quad \text{such that} \quad \forall \{x, y, p, q\} \in W \times S, \\ &\mathscr{L}_r(x^n, y^n, p^n, q^n, \lambda^n, \mu^n) \leqq \mathscr{L}_r(x, y, p, q, \lambda^n, \mu^n) \quad (\textit{locally at least}) \end{aligned} \tag{6.10}$$

$$\begin{aligned} \lambda^{n+1} &= \lambda^n + \tilde{\rho}\left(p^n - \frac{dx^n}{ds}\right), \\ \mu^{n+1} &= \mu^n + \tilde{\rho}\left(q^n - \frac{dy^n}{ds}\right). \end{aligned} \tag{6.11}$$

The only nontrivial part in (6.9)–(6.11) is the solution of the minimization problem (6.10). From the very particular structure of $\mathscr{L}_r$ (cf. (6.7)), it is quite natural to solve (6.10) using a *block-relaxation* method in which one minimizes alternatively with respect to $\{x, y\}$ and $\{p, q\}$; if we restrict in particular the number of *inner iterations* to *only one*, we obtain the following variant of (6.9)–(6.11). *If*

$$\lambda^1, \mu^1, x^0, y^0 \text{ are given}; \tag{6.12}$$

then for $n \geqq 1$, *assuming that* $x^{n-1}, y^{n-1}, \lambda^n, \mu^n$ *are known, we compute* $\{p^n, q^n\}, \{x^n, y^n\}$ *and* $\{\lambda^{n+1}, \mu^{n+1}\}$ *by*:

$$\begin{aligned} &\text{find } \{p^n, q^n\} \in S \quad \text{such that} \quad \forall \{p, q\} \in S, \\ &\mathscr{L}_r(x^{n-1}, y^{n-1}, p^n, q^n, \lambda^n, \mu^n) \leqq \mathscr{L}_r(x^{n-1}, y^{n-1}, p, q, \lambda^n, \mu^n), \end{aligned} \tag{6.13}$$

$$\begin{aligned} &\text{find } \{x^{n-1/2}, y^{n-1/2}\} \in W \quad \text{such that} \quad \forall \{x, y\} \in W, \\ &\mathscr{L}_r(x^{n-1/2}, y^{n-1/2}, p^n, q^n, \lambda^n, \mu^n) \leqq \mathscr{L}_r(x, y, p^n, \lambda^n, \mu^n), \end{aligned} \tag{6.14}$$

$$\begin{aligned} x^n &= x^{n-1} + \omega(x^{n-1/2} - x^{n-1}), \\ y^n &= y^{n-1} + \omega(y^{n-1/2} - y^{n-1}) \end{aligned} \tag{6.15}$$

(*where* $\omega > 0$ *is a relaxation factor*),

$$\begin{aligned} \lambda^{n+1} &= \lambda^n + \tilde{\rho}\left(p^n - \frac{dx^n}{ds}\right), \\ \mu^{n+1} &= \mu^n + \tilde{\rho}\left(q^n - \frac{dy^n}{ds}\right). \end{aligned} \tag{6.16}$$

The iterative methods that we have computationally used have been described for the *continuous problem* whose formalism is much simpler; but in fact (6.9)–(6.11) and (6.12)–(6.16) have been used on the discrete variants of (2.2), discussed in § 5. From a practical point of view it is of fundamental importance to have more explicit formulations of (6.13), (6.14); in this direction we observe that in fact (6.14) is equivalent to the following system of fourth-order two-point boundary value problems:

$$EI\frac{d^4x^{n-1/2}}{ds^4}-r\frac{d^2x^{n-1/2}}{ds^2}=-\frac{d}{ds}(\lambda^n+rp^n)+\textit{boundary conditions}, \tag{6.17}$$

$$EI\frac{d^4y^{n-1/2}}{ds^4}-r\frac{d^2y^{n-1/2}}{ds^2}=-\frac{d}{ds}(\mu^n+rq^n)-\rho g+\textit{boundary conditions}. \tag{6.18}$$

If the boundary conditions are given by (3.1) or (3.2), then (6.17) and (6.18) can be solved *independently* and their discrete versions are *linear systems* with *the same matrix*, which is *sparse, symmetric, positive definite* and *independent of n* if r is *fixed*; in this case we do a *Cholesky factorization, once and for all* and thus, at each step of (6.12)–(6.16) we only have to solve 4 *sparse, well-posed, triangular* systems to obtain $\{x^{n-1/2}, y^{n-1/2}\}$ and therefore $\{x^n, y^n\}$ (via (6.15)).

Let us now discuss the solution of (6.13); to obtain $\{p^n, q^n\}$ we have to solve a.e. on $[0, L]$ the *two-dimensional* minimization problem

$$\operatorname*{Min}_{\{p(s),q(s)\}}\left\{\frac{r}{2}(p^2(s)+q^2(s))+\left(\lambda^n(s)-r\frac{dx^{n-1}}{ds}(s)\right)p(s)+\left(\mu^n(s)-r\frac{dy^{n-1}}{ds}(s)\right)q(s)\right\} \tag{6.19}$$

$$\text{with } \{p(s), q(s)\}\in\mathbb{R}^2, \quad p^2(s)+q^2(s)=1.$$

But since $p^2(s)+q^2(s)=1$, (6.19) reduces

$$\operatorname*{Min}_{\{p(s),q(s)\}}\left\{\left(\lambda^n(s)-r\frac{dx^{n-1}}{ds}(s)\right)p(s)+\left(\mu^n(s)-r\frac{dy^{n-1}}{ds}(s)\right)q(s)\right\}, \tag{6.20}$$

$$\text{with } \{p(s), q(s)\}\in\mathbb{R}^2, \quad p^2(s)+q^2(s)=1.$$

Let us define

$$\begin{aligned} X^n(s)&=\lambda^n(s)-r\frac{dx^{n-1}}{ds}(s),\\ Y^n(s)&=\mu^n(s)-r\frac{dy^{n-1}}{ds}(s); \end{aligned} \tag{6.21}$$

then, if $\{X^n(s), Y^n(s)\}\neq 0$, we have

$$\begin{aligned} p^n(s)&=-\frac{X^n(s)}{\sqrt{|X^n(s)|^2+|Y^n(s)|^2}},\\ q^n(s)&=-\frac{Y^n(s)}{\sqrt{|X^n(s)|^2+|Y^n(s)|^2}}. \end{aligned} \tag{6.22}$$

Remark 6.1. We have seen that (6.13) is a *well-posed* problem if the boundary conditions are given by either (3.1) or (3.2). The minimization problem (6.19), (6.20) is also well-posed as long as $\{X^n(s), Y^n(s)\}\neq\{0, 0\}$; if $X^n(s)=Y^n(s)=0$ the whole circle

$p^2+q^2=1$ is a solution. Actually in all the experiments we have done, we observed that for r *sufficiently large* this cumbersome situation never occurred; an a posteriori semiexplanation of such a behavior is possible, but it will not be given here.

6.2.2. On the choice of r, $\tilde{\rho}$, ω. For related *convex* problems, the larger r is, the faster the convergence of algorithm (6.9)–(6.11) if we use $\tilde{\rho}=r$ (and if *round-off errors* are not considered). However, for large r, problem (6.10) is not well-conditioned and its solution by inner iterations can be a costly task. On the other hand analyzing the convergence of (6.12)–(6.16) is a complicated problem, even in convex situations for which we refer (if $\omega=1$) to [1], [18].

We shall see in § 6.3 the relations existing between algorithm (6.12)–(6.16) and some *alternate direction methods* (ADI); from these relations it will then be possible to consider (6.12)–(6.16) as resulting from the approximate integration in time of some dynamic problem by a *fractional-step* method, with r as the inverse of a *time-step*. From this interpretation it follows that a larger r yields a safer algorithm (cf. Remark 6.1).

Once r has been chosen, we have used $\tilde{\rho}=r$ by analogy with the convex cases analyzed in [1], [18], since it seems to correspond to a quasi-optimal choice. Concerning the choice of ω, all the experiments discussed in § 9 have been done with $\omega=1$ (the influence of ω will be discussed elsewhere, once enough experiments in this direction have been done).

Remark 6.2. In algorithm (6.12)–(6.16) we first solved the problem in $\{p, q\}$ and then the problem in $\{x, y\}$; we can use, of course, the opposite order. However the convergence analysis done (with $\omega=1$) in [1], [18] for convex problems, seems to show that it is essential for the second step to correspond to the minimization of a functional whose gradient has good *monotonicity* properties. We have therefore taken the $\{x, y\}$ problem as second step since it is a linear problem associated to a *strongly elliptic operator* (unlike the $\{p, q\}$ problem which is in fact associated to a *multivalued, nonmonotonic* operator).

6.3. Relations between algorithm (6.12)–(6.16) and alternate direction methods.

6.3.1. An elementary model problem in Hilbert spaces. We consider for simplicity a model problem (possibly nonlinear) less complicated than (2.2). Let V be a *Hilbert* space on $\mathbb{R}$. Let $f\in V$; we consider the following problem

$$A(u)=f, \tag{6.23}$$

where the operator A is defined from $V\to V$. We suppose that A is the derivative of a functional J_0 (i.e., $A=J_0'$) *differentiable* (Fréchet or Gâteaux), *strictly convex* such that

$$\lim_{\|v\|\to+\infty}\frac{J_0(v)}{\|v\|}=+\infty. \tag{6.24}$$

From these properties of J_0, (6.23), which can also be written equivalently as:

$$\begin{aligned}&\textit{find } u\in V \textit{ such that}\\ &J_0(u)-(f,u)\leqq J_0(v)-(f,v)\quad \forall v\in V\end{aligned} \tag{6.25}$$

(or

$$J_0'(u)-f=0) \tag{6.26}$$

has a *unique* solution (cf., e.g., [20], [21]). Suppose now that

$$J_0=J_1+J_2, \tag{6.27}$$

where each J_i is also a convex differentiable functional, with

$$J_i' = A_i. \tag{6.28}$$

Hence

$$A = J_0' = J_1' + J_2' = A_1 + A_2. \tag{6.29}$$

We give also a decomposition of f, i.e.,

$$f = f_1 + f_2, \qquad f_i \in V. \tag{6.30}$$

Since (6.23), (6.25) are clearly equivalent to the following minimization problem

$$\begin{aligned} &\text{find } \{u, p\} \in \mathcal{V} \quad \text{such that} \quad \forall \{v, q\} \in \mathcal{V}, \\ &j(u, p) \leqq j(v, q), \end{aligned} \tag{6.31}$$

where

$$j(v, q) = J_1(v) - (f_1, v) + J_2(q) - (f_2, q) \tag{6.32}$$

$$\mathcal{V} = \{\{v, q\} \in V \times V,\ v - q = 0\}, \tag{6.33}$$

we naturally associate with (6.23), (6.25) the following *augmented Lagrangian* (with $r > 0$),

$$\mathcal{L}_r(v, q, \mu) = j(v, q) + \frac{r}{2}\|v - q\|^2 + (\mu, v - q). \tag{6.34}$$

We can easily prove (see [1]) the following

PROPOSITION 6.2. *Suppose that the above hypotheses upon A, J_0, J_1, J_2 hold. Then $\mathcal{L}_r$ has a unique saddle-point $\{u, p, \lambda\}$ on $V \times V \times V$, such that $p = u$, $\lambda = A_2(u) - f_2 = f_1 - A_1(u)$, where u is precisely the solution of problem* (6.23), (6.25).

From the above results we introduce the following algorithm to solve (6.23), (6.25). This algorithm is closely related to algorithm (6.12)–(6.16).

$$p^0, \lambda^1 \text{ arbitrarily given in } V \times V, \tag{6.35}$$

then for $n \geqq 1$, assuming that p^{n-1}, λ^n are known, we compute u^n, p^n and λ^{n+1} by:

$$\begin{aligned} &\text{find } u^n \in V \quad \text{such that} \quad \forall v \in V, \\ &\mathcal{L}_r(u^n, p^{n-1}, \lambda^n) \leqq \mathcal{L}_r(v, p^{n-1}, \lambda^n), \end{aligned} \tag{6.36}$$

$$\begin{aligned} &\text{find } p^n \in V \quad \text{such that} \quad \forall q \in V, \\ &\mathcal{L}_r(u^n, p^n, \lambda^n) \leqq \mathcal{L}_r(u^n, q, \lambda^n), \end{aligned} \tag{6.37}$$

$$\lambda^{n+1} = \lambda^n + \tilde{\rho}(u^n - p^n); \tag{6.38}$$

in fact (6.36), (6.37) reduce to (the *well-posed*) problems

$$ru^n + A_1(u^n) = rp^{n-1} + f_1 - \lambda^n, \tag{6.39}$$

$$rp^n + A_2(p^n) = ru^n + f_2 + \lambda^n, \tag{6.40}$$

respectively.

6.3.2. Convergence of algorithm (6.35)–(6.40). Supposing that A_1, A_2 obey very reasonable *monotonicity* and *continuity* properties it is proved in [1] (see also [22]) that

$\forall \{p^0, \lambda^1\} \in V \times V$ and

$$\forall \tilde{\rho} \in \left(0, \frac{1+\sqrt{5}}{2} r\right) \tag{6.41}$$

we have

$$\lim_{n \to +\infty} \|u^n - u\| = 0,$$

$$\lim_{n \to +\infty} \|p^n - u\| = 0, \tag{6.42}$$

$$\lim_{n \to +\infty} \lambda^n = \lambda \quad \textit{weakly in } V.$$

If $J_1(\cdot)$ is *linear* (or *affine*) it is proved in [18] that (6.42) holds if $\tilde{\rho} \in (0, 2r)$.

6.3.3. An A.D.I. interpretation of algorithm (6.35)–(6.40). Suppose that $\tilde{\rho} = r$; it follows then from (6.38), (6.40) that

$$\lambda^{n+1} = A_2(p^n) - f_2. \tag{6.43}$$

It follows in turn from (6.39), (6.43) that we have

$$ru^n + A_1(u^n) = rp^{n-1} + f - A_2(p^{n-1}). \tag{6.44}$$

From (6.39) we can eliminate $\lambda^n + ru^n$ in (6.40); doing so we obtain

$$rp^n + A_2(p^n) = rp^{n-1} + f - A_1(u^n). \tag{6.45}$$

Putting $p^{n-1/2} = u^n$ we finally obtain from (6.44), (6.45) that

$$rp^{n+1/2} + A_1(p^{n+1/2}) = rp^n + f - A_2(p^n), \tag{6.46}$$

$$rp^{n+1} + A_2(p^{n+1}) = rp^n + f - A_1(p^{n+1/2}) \tag{6.47}$$

which is an *alternate direction* algorithm of Douglas–Rachford type (see [23]).

In Lions–Mercier [24] the convergence of (6.46)–(6.47) is proved for situations in which A, A_1, A_2 are possibly *not the gradients* of functionals J_0, J_1, J_2, which are possibly *multivalued* but still *maximal monotone operators* (see e.g. [25] for this last concept).

6.3.4. An initial value problem interpretation of algorithm (6.35)–(6.40). We can associate to the steady state problem (6.23) the *initial value problem*

$$\frac{du}{dt} + A(u) = f, \qquad u(0) = u_0. \tag{6.48}$$

From the particular structure of A and f, it is quite natural to use for the *approximate time integration* of (6.48), *splitting* techniques like *fractional step* or *alternate direction* methods based on (6.29), (6.30). One of those methods will lead to (6.46), (6.47); with that interpretation, r appears as the reciprocal of *time step* ($r = 1/\Delta t$) and this fact explains why a larger r gives a safer algorithm (6.12)–(6.16) (see § 6.2).

Applications of several alternate direction methods to the time integration of initial value problems like (6.48) (with A maximal monotone, possibly not the gradient of a functional and possibly multivalued) are considered in [24], where proofs of several interesting convergence results are given.

6.3.5. Applications to more general problems in Hilbert space. In order to generalize § 6.3.1, we consider the following family of minimization problems whose

connection with (2.2) is quite clear (see § 6.2):

$$\operatorname*{Min}_{v\in V} J(v), \tag{6.49}$$

where J is a differentiable, convex function such that

$$J(v)=J_1(\Lambda v)+J_2(v) \tag{6.50}$$

where $\Lambda\in\mathcal{L}(V,H)$, where H is an Hilbert space whose norm is $|\cdot|$ and whose inner product is denoted by $(\cdot,\cdot)$; we suppose also that J_1, J_2 are convex and differentiable on H and V, respectively. We have that (6.49) is equivalent to

$$\operatorname*{Min}_{\{v,q\}\in\mathcal{V}} j(v,q), \tag{6.51}$$

where

$$\mathcal{V}=\{\{v,q\}\in V\times H,\ q-\Lambda v=0\} \tag{6.52}$$

and

$$j(v,q)=J_1(q)+J_2(v). \tag{6.53}$$

We then associate (6.51) with the augmented Lagrangian

$$\mathcal{L}_r(v,q,\mu)=j(v,q)+\frac{r}{2}|q-\Lambda v|^2+(\mu,\Lambda v-q). \tag{6.54}$$

Using the above $\mathcal{L}_r$ we obtain the following generalization of algorithm (6.35)–(6.40) (with slightly different notation)

$$u^0,\lambda^1 \textit{ arbitrarily given in } V\times H, \tag{6.55}$$

then for $n\geqq 1$, *assuming that* u^{n-1}, λ^n *are known, we compute* p^n, u^n *and* λ^{n+1} *by*:

$$\begin{aligned}&\textit{find } p^n\in H \quad \textit{such that} \quad \forall q\in H,\\ &\mathcal{L}_r(u^{n-1},p^n,\lambda^n)\leqq\mathcal{L}_r(u^{n-1},q,\lambda^n),\end{aligned} \tag{6.56}$$

$$\begin{aligned}&\textit{find } u^n\in V \quad \textit{such that} \quad \forall v\in V,\\ &\mathcal{L}_r(u^n,p^n,\lambda^n)\leqq\mathcal{L}_r(v,p^n,\lambda^n),\end{aligned} \tag{6.57}$$

$$\lambda^{n+1}=\lambda^n+\tilde{\rho}(\Lambda u^n-p^n). \tag{6.58}$$

The analogies between algorithms (6.12)–(6.16) and (6.55)–(6.58) are obvious; actually (6.56), (6.57) reduce to

$$rp^n+J_1'(p^n)=r\Lambda u^{n-1}+\lambda^n, \tag{6.59}$$

$$r\Lambda^*\Lambda u^n+J_2'(u^n)=r\Lambda^*p^n-\Lambda^*\lambda^n, \tag{6.60}$$

respectively.

Assuming that $\tilde{\rho}=r$, there is not—in general—a simple A.D.I. interpretation of (6.59), (6.60), (6.58) as was the case in § 6.3.1 (some partial results in that direction are given in [24]); by elimination of λ^n we obtain some relations between $\{p^n\}_n$ and $\{u^n\}_n$ which indicate however an alternate direction behavior; we have more precisely

$$\Lambda^*(rp^n+J_1'(p^n))=r\Lambda^*\Lambda u^{n-1}-J_2'(u^{n-1}), \tag{6.61}$$

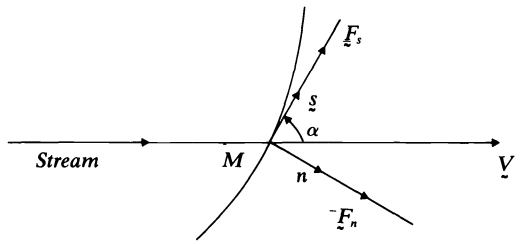
$$r\Lambda^*\Lambda u^n+J_2'(u^n)=r\Lambda^*\Lambda u^{n-1}-\Lambda^*J_1'(p^n). \tag{6.62}$$

We observe that if Λ^* is an *isomorphism* from H onto V (which is not the case in many important applications) then (6.61), (6.62) is actually an alternate direction algorithm.

7. Computations with stream.

7.1. Generalities. We would like to extend in this section the computational methods of the above sections to situations where the pipe is subjected to the *hydrodynamical forces* generated by a *horizontal stream.* We suppose that the stream velocity is *time-independent,* possibly a function of the depth; we suppose also in this report that the stream is parallel to the plane of the pipe.

We suppose finally that the forces acting on the pipe and originated from the stream are given by the so-called *Morison formulas* to be given below. (See the notation of Fig. 7.1.)



(In this case $\alpha = \phi$)

FIG. 7.1. *$\underset{\sim}{V}$ is the stream velocity (in the sequel $V = |\underset{\sim}{V}|$); $\underset{\sim}{s}$, the positively oriented unit vector, tangential at the pipe at M; $\underset{\sim}{n}$, the unit vector, normal at the pipe at M such that angle $(\underset{\sim}{n}, \underset{\sim}{s}) = \pi/2$; α, the angle $(\underset{\sim}{V}, \underset{\sim}{s})$; F_s (resp. F_n), the component along $\underset{\sim}{s}$ (resp. $\underset{\sim}{n}$) of the hydrodynamical forces (per unit length).*

Using these notations the Morison formulas are, for a pipe with a circular cross-section,

$$F_n = \tfrac{1}{2}\rho_w C_d D V^2 \sin\alpha, \tag{7.1}$$

$$F_s = \tfrac{1}{2}\rho_w C_f \pi D V^2 \cos\alpha, \tag{7.2}$$

where ρ_w is the volumic density of the water, C_d and C_f some friction coefficients, and D the pipe diameter. If one uses the M.K.S.A. system, we have $C_d = 1.2$, $C_f = 0.03$; for sea water we take $\rho_w = 1{,}026\ \text{kg/m}^3$.

7.2. Formulation of F_n, F_s with respect to x', y'. We introduce

$$C_n = \tfrac{1}{2}\rho_w C_d D, \qquad C_s = \tfrac{1}{2}\rho_w C_f \pi D; \tag{7.3}$$

then

$$F_n = C_n V^2 \sin\alpha, \qquad F_s = C_s V^2 \cos\alpha.$$

Using the fact that angle $(\underset{\sim}{V}, 0x) = 0$ or π and also that $x' = \cos\phi$, $y^1 = \sin\phi$ we obtain

$$\underset{\sim}{F_n} = \{F_{nx}, F_{ny}\} = \{\tilde{C}_n V^2 y'^2, -\tilde{C}_n V^2 x' y'\}, \tag{7.4}$$

$$\underset{\sim}{F_s} = \{F_{sx}, F_{sy}\} = \{\tilde{C}_s V^2 x'^2, \tilde{C}_s V^2 x' y'\}, \tag{7.5}$$

where

$$\begin{aligned} \tilde{C}_n &= C_n \cos(\underset{\sim}{V}, 0x), \\ \tilde{C}_s &= C_s \cos(\underset{\sim}{V}, 0x). \end{aligned} \tag{7.6}$$

7.3. Formulation of the static problem with a horizontal stream. Using the *virtual feasible displacement principle* we obtain that any solution of the static problem with stream has to satisfy

$$EI\int_0^L (x''\xi'' + y''\eta'')\,ds + \rho g\int_0^L \eta\,ds - \int_0^L (F_{sx}+F_{nx})\xi\,ds - \int_0^L (F_{sy}+F_{ny})\eta\,ds = 0 \quad \forall\{\xi,\eta\}\in D\mathscr{E}(x,y),\{x,y\}\in\mathscr{E}, \tag{7.7}$$

where the set of the *feasible displacements* $D\mathscr{E}(x, y)$ is defined $\forall\{x, y\}\in\mathscr{E}$, by

$$D\mathscr{E}(x,y) = \{\{\xi,\eta\}\in H^2(0,L)\times H^2(0,L);\ \textit{the boundary conditions on } \{\xi,\eta\} \textit{ are compatible with the boundary conditions in } \mathscr{E};\ x'(\xi')+y'\eta' = 0\}. \tag{7.8}$$

Remark 7.1. If the boundary conditions in $\mathscr{E}$ are defined by (3.1) (resp. (3.2)) the above compatibility condition implies that

$$\xi(0)=\xi(L)=0, \qquad \eta(0)=\eta(L)=0, \tag{7.9}$$

respectively

$$\xi(0)=\xi(L)=0, \qquad \eta(0)=\eta(L)=0, \qquad \xi'(0)=\xi'(L)=0, \qquad \eta'(0)=\eta'(L)=0. \tag{7.10}$$

Remark 7.2. Using the *finite element approximations* of $H^2(0, L)$ and $\mathscr{E}$, introduced in § 5, we can easily approximate (7.7) by a finite dimensional problem of the same structure (see § 9.3 for additional details).

7.4. Iterative solutions of (7.7). The discussion will be done on the continuous problem since its formalism is simpler than the one of the discrete problem. Obviously the algorithms to be computationally used, are in fact discrete variants of the algorithms described below.

From (7.3)–(7.8), problem (7.7) is "*more*" *nonlinear* than (2.2): we have in particular to realize that $\underline{Y}$ is possibly a function of y ($\underline{Y}(y)$ in the sequel), since we have supposed that the horizontal stream may vary with the depth. Another difficulty lies in the fact that the hydrodynamical forces *are not the derivative of any functional.* Such a negative property seems to preclude the use of augmented Lagrangian techniques like those described in § 6.2; in fact using very natural extensions (to be described below) of the algorithms in § 6 we have been able to obtain very good numerical solutions for (7.7), *even with rather strong streams.*

7.4.1. A generalization of algorithm (6.9)–(6.11). We use the notation of § 6.2. Returning to algorithm (6.9)–(6.11), (6.10) implies that

$$\{p^n,q^n\}\in S \quad\text{and}\quad \mathscr{L}_r(x^n,y^n,p^n,q^n,\lambda^n,\mu^n)\leqq\mathscr{L}_r(x^n,y^n,p,q,\lambda^n,\mu^n) \quad \forall\{p,q\}\in S, \tag{7.11}$$

$$\{x^n,y^n\}\in W \quad\text{and}\quad \mathscr{L}_r(x^n,y^n,p^n,q^n,\lambda^n,\mu^n)\leqq\mathscr{L}_r(x,y,p^n,q^n,\lambda^n,\mu^n) \quad \forall\{x,y\}\in W, \tag{7.12}$$

and (7.12) is equivalent to the system

$$EI\frac{d^4x^n}{ds^4}-r\frac{d^2x^n}{ds^2}=-\frac{d}{ds}(\lambda^n+rp^n) \quad \text{on } (0,L)+\textit{boundary conditions}, \tag{7.13}$$

$$EI\frac{d^4y^n}{ds^4}-r\frac{d^2y^n}{ds^2}=-\frac{d}{ds}(\mu^n+rq^n)-\rho g \quad \text{on } (0,L)+\textit{boundary conditions}. \tag{7.14}$$

Thus to solve (7.7) we can use the straightforward generalization of algorithm (6.9)–(6.11). *If*

$$\lambda^0, \mu^0 \textit{ are given}, \tag{7.15}$$

then for $n \geqq 0$, *assuming that* λ^n, μ^n *are known, we obtain* p^n, q^n, x^n, y^n *by solving the (nonlinear) system*

$$\mathscr{L}_r(x^n, y^n, p^n, q^n, \lambda^n, \mu^n) \leqq \mathscr{L}_r(x^n, y^n, p, q, \lambda^n, \mu^n) \quad \forall \{p,q\} \in S, \quad \{p^n, q^n\} \in S, \tag{7.16}$$

$$EI\frac{d^4x^n}{ds^4}-r\frac{d^2x^n}{ds^2}=-\frac{d}{ds}(\lambda^n+rp^n)+F_{sx}(x^n,y^n)+F_{nx}(x^n,y^n) + \textit{boundary conditions}, \tag{7.17}$$

$$EI\frac{d^4y^n}{ds^4}-r\frac{d^2y^n}{ds^2}=-\frac{d}{ds}(\mu^n+rq^n)-\rho g+F_{sy}(x^n,y^n)+F_{ny}(x^n,y^n) + \textit{boundary conditions}, \tag{7.18}$$

and then λ^{n+1}, μ^{n+1} *by*

$$\lambda^{n+1}=\lambda^n+\tilde{\rho}\left(p^n-\frac{dx^n}{dx}\right), \qquad \mu^{n+1}=\mu^n+\tilde{\rho}\left(q^n-\frac{dy^n}{ds}\right). \tag{7.19}$$

In (7.16) $\mathscr{L}_r$ is still defined by (6.7); in (7.17), (7.18) F_{sx}, F_{nx}, F_{ny}, F_{sy} are given by (7.3)–(7.6) (with $V = V(y)$). To solve the coupled system (7.16)–(7.18) we suggest (and have used) a *block relaxation* method. Let us describe such a method for the computation of $\{x^n, y^n, p^n, q^n\}$ from $\{\lambda^n, \mu^n\}$ (below, m denotes an *inner iteration indice*).

$$x^{n,0}=x^{n-1}, \qquad y^{n,0}=y^{n-1}; \tag{7.20}$$

then for $m \geqq 1$, *assuming that* $x^{n,m-1}$, $y^{n,m-1}$ *are known, we compute successively* $\{p^{n,m}, q^{n,m}\}$, $\{x^{n,m}, y^{n,m}\}$ *by solving*:

find $\{p^{n,m}, q^{n,m}\} \in S$ *such that* $\forall \{p,q\} \in S$,

$$\mathscr{L}_r(x^{n,m-1}, y^{n,m-1}, p^{n,m}, q^{n,m}, \lambda^n, \mu^n) \leqq \mathscr{L}_r(x^{n,m-1}, y^{n,m-1}, p, q, \lambda^n, \mu^n), \tag{7.21}$$

$$EI\frac{d^4x^{n,m-1/2}}{ds^4}-r\frac{d^2x^{n,m-1/2}}{ds^2}=-\frac{d}{ds}(\lambda^n+rp^{n,m})+F_{sx}(x^{n,m-1},y^{n,m-1}) + F_{nx}(x^{n,m-1},y^{n,m-1}) + \textit{boundary conditions}, \tag{7.22}$$

$$EI\frac{d^4y^{n,m-1/2}}{ds^4}-r\frac{d^2y^{n,m-1/2}}{ds^2}=-\frac{d}{ds}(\mu^n+rq^{n,m})-\rho g+F_{sy}(x^{n,m-1},y^{n,m-1}) + F_{ny}(x^{n,m-1},y^{n,m-1}) + \textit{boundary conditions} \tag{7.23}$$

$$
\begin{aligned}
x^{n,m} &= x^{n,m-1} + \omega(x^{n,m-1/2} - x^{n,m-1}),\\
y^{n,m} &= y^{n,m-1} + \omega(y^{n,m-1/2} - y^{n,m-1}),
\end{aligned}
\tag{7.24}
$$

ω being a *relaxation factor.*

Since problem (7.21) is a variant of problem (6.13), we refer to § 6.2 where the solution of this latter problem has been discussed in detail. Concerning (7.22), (7.23) we observe that these two problems can be solved independently if the boundary conditions in $\mathscr{E}$ are given by (3.1), (3.2) (for more details, see in § 6.2 the discussion about the solution of (6.17), (6.18)).

Remark 7.3. If one uses the inner iterative process (7.20)–(7.24) to solve system (7.16)–(7.18) in algorithm (7.15)–(7.19), it is necessary to give not only $\{\lambda^0, \mu^0\}$ but also $\{x^{-1}, y^{-1}\}$ for the initialization of algorithm (7.20)–(7.24) (case $n = m = 0$).

Remark 7.4. The numerical solutions of (7.7), presented in § 9, have been obtained using $\omega = 1$.

7.4.2. A generalization of algorithm (6.12)–(6.16). A variant of algorithm (7.15)–(7.19) can be obtained if the number of inner iterations in (7.20)–(7.24) is limited to *only one.* The resulting algorithm which generalizes algorithm (6.12)–(6.16), can be easily written from (7.15)–(7.19) and (7.20)–(7.24).

Remark 7.5. All the remarks and comments in § 6.2 concerning the choice of $\tilde{\rho}$, r, ω also apply for the algorithms discussed in § 7.3.

8. Dynamic calculations.

8.1. Synopsis. In order to simulate numerically the *dynamical* behavior of the pipelines considered in § 1, we extend in this section the methods described and discussed in §§ 4, 5, 6. For simplicity we suppose that the boundary conditions are given by either (3.1) or (3.2) (with x_A, y_A, x_B, y_B, α_0, β_0, α_L, β_L possibly dependent on time t) and we neglect hydrodynamical forces and internal dissipation (the dynamical problem with hydrodynamical forces will be considered in a subsequent paper). After giving in § 8.2 the formulation to be used for the dynamical problem, we shall describe in § 8.3, an *implicit time discretization scheme* of *Houbolt type* which reduces the time integration to a *sequence of static problems* very close to problem (2.2). Since the Houbolt scheme is *multistep* a *starting procedure* is needed; such a procedure will be discussed in § 8.3 also. The corresponding numerical results are presented in § 9.4, where some additional details about the space discretization of the acceleration terms are also given. The possibility of using a *Crank–Nicolson time discretization scheme* will be also discussed in § 8.3.

8.2. Formulation of the time dependent problem. Using *Hamilton's principle* (see e.g. Clough–Penzien [26]) we have from the above physical hypotheses that the time dependent behavior of the pipe is given by the vector-function

$$\{s, t\} \to \{x(s, t), y(s, t)\}$$

solution of the following *initial value wave problem.*

$$
\begin{aligned}
&\textit{Find } \{x(t), y(t)\} \in \mathscr{E}(t) \quad \textit{such that} \quad \forall \{\xi, \eta\} \in D\mathscr{E}(x(t), y(t)) \textit{ we have a.e. in } t,\\
&\rho \int_0^L (\ddot{x}\xi + \ddot{y}\eta)\, ds + EI \int_0^L (x''\xi'' + y''\eta'')\, ds + \rho g \int_0^L \eta\, ds = 0,
\end{aligned}
\tag{8.1}
$$

$$
\{x(0), y(0)\} = \{x_0, y_0\}, \qquad \{\dot{x}(0), \dot{y}(0)\} = \{x_1, y_1\}. \tag{8.2}
$$

In (8.1), (8.2) we have used the following notation:

(i) $x(t)$ (resp. $y(t)$) denotes the function $s \to x(s, t)$ (resp. $s \to y(s, t)$);

(ii) $\mathscr{E}(t)$ is the subset of $H^2(0, L)\times H^2(0, L)$ defined by the boundary conditions at time t, and the inextensibility condition (2.1);

(iii) $D\mathscr{E}(x(t), y(t))$ is the subset of $H^2(0, L)\times H^2(0, L)$ associated to $\{x(t), y(t)\}$ by (7.8) (Remark 7.1 still holds for $D\mathscr{E}(x(t), y(t))$);

(iv) $\dot{x} = \partial x/\partial t$, $\dot{y} = \partial y/\partial t$, $\ddot{x} = \partial^2 x/\partial t^2$, $\ddot{y} = \partial^2 y/\partial t^2$;

(v) $x' = \partial x/\partial s$, $y' = \partial y/\partial s$, $x'' = \partial^2 x/\partial s^2$, $y'' = \partial^2 y/\partial s^2$.

Remark 8.1. To our knowledge, the wave problem (8.1), (8.2) is mathematically open. From the fact that $\{x(t), y(t)\}$ obeys the inextensibility condition (2.1) a.e. in t we can reasonably suppose that the initial values (8.2) have to satisfy some *compatibility conditions*; it seems reasonable to require that $\{x(0), y(0)\}$ obeys (2.1). Moreover by derivation with respect to t of

$$\left|\frac{\partial x}{\partial s}(s, t)\right|^2 + \left|\frac{\partial y}{\partial s}(s, t)\right|^2 = 1,$$

we obtain that

$$\frac{\partial x}{\partial s}\frac{\partial \dot{x}}{\partial s} + \frac{\partial y}{\partial s}\frac{\partial \dot{y}}{\partial s} = 0;$$

therefore, at $t = 0$, we have (using the notation of (8.2))

$$x'(0)\dot{x}'(0) + y'(0)\dot{y}'(0) = 0 \quad \text{that is} \quad x_0' x_1' + y_0' y_1' = 0, \tag{8.3}$$

a compatibility condition between the initial data.

8.3. Numerical solution of (8.1), (8.2).

8.3.1. Generalities. The *numerical integration* of dynamical linear and nonlinear structural problems has motivated a very large number of papers, books and conferences. Among recent engineering contributions in this field let us mention [26], Argyris–Dunne [27], Oden [28], Bathe–Wilson [29], Zienkiewicz [30], Geradin [31], Clough–Wilson [32], Hughes–Pister–Taylor [33], Belitschko–Yen–Mullen [34], Felippa–Park [35], Sander–Geradin–Nyssen–Hogge [36], Argyris–Doltsinis–Knudson–Vaz–Willam [37], Warburton [38]; see also the references therein. For the numerical integration of the standard wave equation $(\partial^2 u/\partial t^2) - \Delta u = 0$ by *alternate direction techniques* we refer to McKee [39], Ciment–Leventhal [40], Lees [41], Jain–Ahuja–Bhattacharya [42], Iyengar–Mittal [43], Konovalov [44].

Let us also mention [45] where time dependent pipeline calculations have been performed by methods different from those which follow.

With regards to the wave problem (8.1), (8.2), the situation is considerably complicated by the presence of the *inextensibility condition* (2.1). As mentioned before, we have not included in our model the hydrodynamical forces resulting from the *friction* of the water; in fact, we have the feeling that these friction forces, in spite of their complicated analytical expression, will make the numerical integration easier, since they will damp the mechanical phenomenon under consideration. With regards precisely to *dissipation*, we have chosen to solve (8.1), (8.2), a *Houbolt time integration scheme* (described in § 8.3.2), in spite of the *numerical dissipation* associated to it (see, e.g., [29] for more details), because *underwater calculations* (i.e. in a *dissipative medium*) are precisely our final goal in this class of pipeline problems. Another reason to choose a Houbolt scheme is that it is well-suited to match the difficulty associated to the *inextensibility* condition (2.1); actually the major inconvenience of the Houbolt scheme is that it requires a *starting procedure* (described in § 8.3.3), complicated by the inextensibility condition (2.1) (and the associated compatibility condition (8.3)).

8.3.2. Time discretization of (8.1), (8.2) via Houbolt's scheme. Again we only consider the continuous problem; there is no practical difficulty to extend the considerations which follow to the variants of (8.1), (8.2) obtained via the space approximations discussed in § 5. We reduce problem (8.1), (8.2) to a sequence of static problems (variants of problem (2.2)) using the following multistep time discretization scheme:

$$\{x^j, y^j\} \in \mathscr{E}^j \quad \textit{is given for} \quad j = 0, 1, 2; \tag{8.4}$$

then, for $n \geqq 2$, *assuming that* $\{x^j, y^j\} \in \mathscr{E}^j$ *are known for* $j = n-2, n-1, n$ *we obtain* $\{x^{n+1}, y^{n+1}\} \in \mathscr{E}^{n+1}$ *as the solution of*:

$$\begin{aligned} &\textit{find } \{x^{n+1}, y^{n+1}\} \in \mathscr{E}^{n+1} \quad \textit{such that} \quad \forall \{\xi, \eta\} \in D\mathscr{E}^{n+1} \textit{ we have} \\ &\rho \int_0^L \left\{ \left(\frac{2x^{n+1} - 5x^n + 4x^{n-1} - x^{n-2}}{|\Delta t|^2} \right) \xi + \left(\frac{2y^{n+1} - 5y^n + 4y^{n-1} - y^{n-2}}{|\Delta t|^2} \right) \eta \right\} ds \\ &\qquad + EI \int_0^L \{ (x^{n+1})'' \xi'' + (y^{n+1})'' \eta'' \} \, ds + \rho g \int_0^L \eta \, ds = 0. \end{aligned} \tag{8.5}$$

We have used in (8.4), (8.5) the following notation:

(i) Δt is a *time step* and $\{x^j, y^j\}$ is an *approximation* (at least we hope so) of $\{x(j\Delta t), y(j\Delta t)\}$, where $\{x(t), y(t)\}$ is the solution of (8.1), (8.2).

(ii) $\mathscr{E}^j$ is the subset of $H^2(0, L) \times H^2(0, L)$ defined by the boundary conditions at time $t = j\Delta t$ and the inextensibility condition (2.1).

(iii) $D\mathscr{E}^j$ is the subset of $H^2(0, L) \times H^2(0, L)$ associated to $\{x^j, y^j\}$ by (7.8) (Remark 7.1 still holds for $D\mathscr{E}^j$).

The above time discretization scheme is obviously a Houbolt scheme from the choice which has been made to discretize $\partial^2 x/\partial t^2$ and $\partial^2 y/\partial t^2$ in (8.2) (further properties of the Houbolt scheme are analyzed in, e.g. [29]).

It is clear that the above scheme *cannot be used* to compute $\{x^j, y^j\}$, $j = 1, 2$, from the initial data (8.2); thus a *starting procedure* is needed to obtain these two vectors. This procedure is described in § 8.3.3.

8.3.3. A Crank–Nicolson starting procedure for the Houbolt scheme (8.4) (8.5). In a "more standard" situation (i.e. without the inextensibility condition (2.1)) we could have used (following, e.g. [29]) the *initial data* and the *wave equation itself* to obtain $\{\ddot{x}(0), \ddot{y}(0)\}$. Using then $\{x(0), y(0)\}$, $\{\dot{x}(0), \dot{y}(0)\}$, $\{\ddot{x}(0), \ddot{y}(0)\}$ we can construct a *quadratic approximation*—say $\{x^*(t), y^*(t)\}$—of $\{x(t), y(t)\}$ in the neighborhood of $t = 0$. We define then, for $j = 1, 2$, $\{x^j, y^j\}$ by $\{x^j, y^j\} = \{x^*(j\Delta t), y^*(j\Delta t)\}$; the corresponding approximation errors are $O(|\Delta t|^2)$.

Problem (8.1), (8.2) is more delicate to handle and will require a more sophisticated procedure.

The initial vector $\{x^0, y^0\}$ being known from (8.2)—we take $\{x^0, y^0\} = \{x_0, y_0\}$—suppose that $\{x^1, y^1\}$ is also known; we then approximate $\{\ddot{x}(\Delta t), \ddot{y}(\Delta t)\}$ by $\{(x^2 + x^0 - 2x^1)/|\Delta t|^2, (y^2 + y^0 - 2y^1)/|\Delta t|^2\}$ and we discretize (8.2) at $t = \Delta t$, using the following *Crank–Nicholson scheme*

$$\begin{aligned} &\rho \int_0^L \left\{ \left(\frac{x^2 + x^0 - 2x^1}{|\Delta t|^2} \right) \xi + \left(\frac{y^2 + y^0 - 2y^1}{|\Delta t|^2} \right) \eta \right\} ds + EI \int_0^L \{ (\tilde{x}^1)'' \xi'' + (\tilde{y}^1)'' \eta'' \} \, ds \\ &\qquad + \rho g \int_0^L \eta \, ds = 0 \quad \forall \{\xi, \eta\} \in D\mathscr{E}(\tilde{x}^1, \tilde{y}^1), \quad \{\tilde{x}^1, \tilde{y}^1\} \in \mathscr{E}^1, \end{aligned} \tag{8.6}$$

where

$$\{\tilde{x}^1, \tilde{y}^1\} = \left\{\frac{x^0+2x^1+x^2}{4}, \frac{y^0+2y^1+y^2}{4}\right\}. \tag{8.7}$$

In fact $\{x^1, y^1\}$ *is not known*; we may however overcome this difficulty as follows:

We have (by Taylor's expansion)

$$x(t) \simeq x^*(t) = x(0)+\dot{x}(0)t+\tfrac{1}{2}\ddot{x}(0)t^2, \tag{8.8}$$

$$y(t) \simeq y^*(t) = y(0)+\dot{y}(0)t+\tfrac{1}{2}\ddot{y}(0)t^2. \tag{8.9}$$

We then define $\{x^1, y^1\}$, $\{x^2, y^2\}$ by

$$x^1 = x^*(\Delta t) = x(0)+\dot{x}(0)\Delta t+\ddot{x}(0)((\Delta t)^2/2), \tag{8.10}$$

$$y^1 = y^*(\Delta t) = y(0)+\dot{y}(0)\Delta t+\ddot{y}(0)((\Delta t)^2/2), \tag{8.11}$$

$$x^2 = x^*(2\Delta t) = x(0)+2\dot{x}(0)\Delta t+2\ddot{x}(0)(\Delta t)^2, \tag{8.12}$$

$$y^2 = y^*(2\Delta t) = y(0)+2\dot{y}(0)\Delta t+2\ddot{y}(0)(\Delta t)^2 \tag{8.13}$$

or, using the notation in (8.2), we have from (8.10)–(8.13),

$$x^1 = x_0+x_1\Delta t+\ddot{x}(0)((\Delta t)^2/2), \tag{8.14}$$

$$y^1 = y_0+y_1\Delta t+\ddot{y}(0)((\Delta t)^2/2), \tag{8.15}$$

$$x^2 = x_0+2x_1\Delta t+2\ddot{x}(0)(\Delta t)^2, \tag{8.16}$$

$$y^2 = y_0+2y_1\Delta t+2\ddot{y}(0)(\Delta t)^2. \tag{8.17}$$

By elimination between (8.14), (8.15) and (8.16), (8.17), respectively, we obtain

$$x^1 = \tfrac{1}{4}(3x_0+2\Delta t x_1+x^2), \tag{8.18}$$

$$y^1 = \tfrac{1}{4}(3y_0+2\Delta t y_1+y^2) \tag{8.19}$$

which by substitution in (8.6) imply in turn, with the notation of (8.2),

$$\begin{aligned}\rho\int_0^L \left\{\left(\frac{x^2-x_0-2\Delta t x_1}{2|\Delta t|^2}\right)\xi+\left(\frac{y^2-y_0-2\Delta t y_1}{2|\Delta t|^2}\right)\eta\right\} ds + EI\int_0^L \{(\tilde{x}^1)''\xi''+(\tilde{y}^1)''\eta''\}\, ds \\ +\rho g\int_0^L \eta\, ds = 0 \quad \forall\{\xi,\eta\}\in D\mathscr{E}(\tilde{x}^1,\tilde{y}^1), \quad \{\tilde{x}^1,\tilde{y}^1\}\in\mathscr{E}^1\end{aligned} \tag{8.20}$$

with this time

$$\{\tilde{x}^1, \tilde{y}^1\} = \left\{\frac{5x_0+2\Delta t x_1+3x^2}{8}, \frac{5y_0+2\Delta t y_1+3y^2}{8}\right\}. \tag{8.21}$$

Using the above method we don't need to know $\{x^1, y^1\}$ and the unknown in (8.20) is $\{x^2, y^2\}$; in fact a more practical unknown is $\{\tilde{x}^1, \tilde{y}^1\}$; eliminating $\{x^2, y^2\}$ between (8.20) and (8.21) we finally obtain that $\{\tilde{x}^1, \tilde{y}^1\}$ is the solution of the following variant of the static problem (2.2),

$$\begin{aligned}\frac{4}{3}\rho\int_0^L \left\{\left(\frac{\tilde{x}^1-x_0-\Delta t x_1}{|\Delta t|^2}\right)\xi+\left(\frac{\tilde{y}^1-y_0-\Delta t y_1}{|\Delta t|^2}\right)\eta\right\} ds + EI\int_0^L \{(\tilde{x}^1)''\xi''+(\tilde{y}^1)''\eta''\}\, ds \\ +\rho g\int_0^L \eta\, ds = 0 \quad \forall\{\xi,\eta\}\in D\mathscr{E}(\tilde{x}^1,\tilde{y}^1), \{\tilde{x}^1,\tilde{y}^1\}\in\mathscr{E}^1.\end{aligned} \tag{8.22}$$

Once $\{\tilde{x}^1, \tilde{y}^1\}$ is known, we can easily obtain $\{x^2, y^2\}$, and then $\{x^1, y^1\}$ from (8.18), (8.19) and (8.21). In fact it is very unlikely that $\{x^1, y^1\}$, $\{x^2, y^2\}$ obtained by the above process may satisfy the inextensibility condition (2.1); we observe however that $\{\tilde{x}^1, \tilde{y}^1\}$, which follows from (8.22), is also an approximation of $\{x(\Delta t), y(\Delta t)\}$, and finally the vector $\{x^1, y^1\}(\in \mathscr{E}^1)$ in (8.4), (8.5) is taken equal to the solution of (8.22).

Using a variant of the above process, and also the initial data (8.2) and the solution of (8.22), we can similarly define a vector $\{x^2, y^2\} \in \mathscr{E}^2$ to be used in (8.1), (8.2).

Remark 8.2. We think that it is important to have an accurate starting procedure since, as we are dealing with *a wave probem without damping*, an inaccurate starting procedure will produce large errors which will propagate and spoil the approximate solution.

Remark 8.3. The set $\mathscr{E}^1$ occurring in (8.6), (8.20), (8.22) corresponds to the boundary conditions satisfied by

$$\left\{\frac{x(0)+2x(\Delta t)+x(2\Delta t)}{4}, \frac{y(0)+2y(\Delta t)+y(2\Delta t)}{4}\right\};$$

these boundary conditions are in general different from those satisfied by $\{x,(\Delta t), y(\Delta t)\}$ (they coincide if the boundary conditions are time dependent).

Remark 8.4. We can imagine starting procedures rather different from the one above, using for example, a *projection* operation on the manifold defined by the inextensibility condition (various norms can be used for this projection). At any rate these methods will require the solution of problems very close to the static problem (2.2).

8.3.4. Time discretization of (8.1), (8.2) via a Crank–Nicolson scheme. An alternative to the Houbolt scheme (8.4), (8.5) is the following scheme of *Crank–Nicolson* type (which has not been tested numerically yet):

$$\{x^j, y^j\} \in W^j \quad \textit{is given for } j = 0, 1, 2 \quad \textit{with } \{\tilde{x}^1, \tilde{y}^1\} \in \tilde{\mathscr{E}}^1; \tag{8.23}$$

then for $n \geqq 2$, *assuming that* $\{x^j, y^j\} \in W^j$ *are known for* $j = n-1$, n, *we obtain* $\{x^{n+1}, y^{n+1}\}$ *as the solution of*:

$$\begin{aligned} &\textit{find } \{x^{n+1}, y^{n+1}\} \in W^{n+1} \quad \textit{such that} \quad \forall \{\xi, \eta\} \in D\tilde{\mathscr{E}}^n, \\ &\rho \int_0^L \left\{ \left(\frac{x^{n+1} - 2x^n + x^{n-1}}{|\Delta t|^2} \right) \xi + \left(\frac{y^{n+1} - 2y^n + y^{n-1}}{|\Delta t|^2} \right) \eta \right\} ds \\ &\qquad + EI \int_0^L \{(\tilde{x}^n)''\xi'' + (\tilde{y}^n)''\eta''\}\, ds + \rho g \int_0^L \eta\, ds = 0, \qquad \{\tilde{x}, \tilde{y}^n\} \in \tilde{\mathscr{E}}^n, \end{aligned} \tag{8.24}$$

where

$$\{\tilde{x}^n, \tilde{y}^n\} = \left(\frac{x^{n-1} + 2x^n + x^{n+1}}{4}, \frac{y^{n-1} + 2y^n + y^{n+1}}{4} \right). \tag{8.25}$$

Most of the notations in (8.23), (8.24) are in common with (8.4), (8.5); the new notations are

(i) W^j is the subspace of $H^2(0, L) \times H^2(0, L)$, associated with the boundary conditions satisfied by $\{x(j\Delta t), y(j\Delta t)\}$.

(ii) $\mathscr{E}^j$ is the subset of $H^2(0, L)\times H^2(0, L)$ defined by the inextensibility condition (2.1) and the boundary condition satisfied by

$$\left\{\frac{x((j-1)\Delta t)+2x(j\Delta t)+x((j+1)\Delta t)}{4}, \frac{y((j-1)\Delta t)+2y(j\Delta t)+y((j+1)\Delta t)}{4}\right\}.$$

(iii) $D\tilde{\mathscr{E}}^j$ is the subset of $H^2(0, L)\times H^2(0, L)$ associated to $\{\tilde{x}^j, \tilde{y}^j\}$ by (7.8).

The above scheme also needs a starting procedure; in fact using the method described in § 8.3.3., we may obtain $\{\tilde{x}^1, \tilde{y}^1\}$ (by (8.22)) and then $\{x^1, y^1\}, \{x^2, y^2\}$. Since

$$x^{n+1}=4\tilde{x}^n-2x^n-x^{n-1}, \qquad y^{n+1}=4\tilde{y}^n-2y^n-y^{n-1}, \tag{8.26}$$

we obtain by substitution in (8.24) that $\{\tilde{x}^n, \tilde{y}^n\}$ is the solution of the following variant of the static problem (2.2):

Find $\{\tilde{x}^n, \tilde{y}^n\}\in\tilde{\mathscr{E}}^n$ *such that* $\forall\{\xi, \eta\}\in D\tilde{\mathscr{E}}^n$ *we have*

$$4\rho\int_0^L\left\{\left(\frac{\tilde{x}^n-x^n}{|\Delta t|^2}\right)\xi+\left(\frac{\tilde{y}^n-y^n}{|\Delta t|^2}\right)\eta\right\}ds+EI\int_0^L\{(\tilde{x}^n)''\xi''+(\tilde{y}^n)''\eta\}\,ds +\rho g\int_0^L\eta\,ds=0. \tag{8.27}$$

Once $\{\tilde{x}^n, \tilde{y}^n\}$ is known, $\{x^{n+1}, y^{n+1}\}$ follows from (8.26).

Remark 8.5. We have to observe that $\{x^n, y^n\}$ obtained by the above Crank–Nicolson method *does not satisfy* (2.1); an alternative is to take $\{\tilde{x}^n, \tilde{y}^n\}$ (which obeys (2.1)) as approximate solutions at $t=n\Delta t$; but with that latter choice, and if the *boundary conditions are time dependent*, then $\{\tilde{x}^n, \tilde{y}^n\}$ does not obey, in general, the same boundary conditions as $\{x(n\Delta t), y(n\Delta t)\}$.

Remark 8.6. The above method is less storage demanding that the Houbolt scheme of § 8.3.2., but it has the difficulties mentioned in Remark 8.5. For *linear problems* it is well-known that the Crank–Nicolson schemes are *less dissipative* than Houbolt's, and like this latter scheme they are second-order accurate.

Remark 8.7. In fact (8.23), (8.24) is the particular case, corresponding to $\theta=0.25$, of a general class of Crank–Nicholson methods for which

$$\{\tilde{x}^n, \tilde{y}^n\}=\{\theta x^{n+1}+(1-2\theta)x^n+\theta x^{n-1}, \theta y^{n+1}+(1-2\theta)y^n+\theta y^{n-1}\}, \qquad 0<\theta<\tfrac{1}{2}. \tag{8.28}$$

We took $\theta=\frac{1}{4}$ since in "good" linear cases, this choice leads to *unconditionally stable* schemes, with regards to Δt, which possess a very small numerical dissipation compared to Houbolt's method. Another reason to use $\theta=\frac{1}{4}$ is that Dahlquist [46] has recently proved (at least for linear, scalar, second-order differential equations) that Crank–Nicolson's scheme, with $\theta=\frac{1}{4}$, *minimizes the truncation error* among the unconditionally stable, multistep schemes for second-order differential *equations.*

8.3.5. Numerical solution of (8.5) and (8.27). A most important step toward the numerical integration of the wave problem (8.1), (8.2) is the solution of (8.5) (resp. (8.27)), if scheme (8.4), (8.5) (resp. (8.23)–(8.25)) is used. Both problems are variants of the static problem (2.2) and can be reformulated as problems of the *calculus of variations*; if $\{x^{n+1}, y^{n+1}\}$ (resp. $\{\tilde{x}^n, \tilde{y}^n\}$) is a solution of (8.5) (resp. (8.27)) it is also a *stationary point* on $\mathscr{E}^{n+1}$ (resp. $\tilde{\mathscr{E}}^n$) of the functional

$$\{\xi, \eta\}\to\frac{\rho}{|\Delta t|^2}\int_0^L(\xi^2+\eta^2)\,ds+\frac{EI}{2}\int_0^L(\xi''^2+\eta''^2)\,ds+\rho g\int_0^L\eta\,ds +\frac{\rho}{|\Delta t|^2}\int_0^L\{(-5x^n+4x^{n-1}-x^{n-2})\xi+(-5y^n+4y^{n-1}-y^{n-2})\eta\}\,ds, \tag{8.29}$$

respectively

$$\{\xi, \eta\} \to \frac{2\rho}{|\Delta t|^2}\int_0^L (\xi^2+\eta^2)\,ds + \frac{EI}{2}\int_0^L (\xi''^2+\eta''^2)\,ds + \rho g\int_0^L \eta\,ds - \frac{4\rho}{|\Delta t|^2}\int_0^L (x^n\xi + y^n\eta)\,ds. \tag{8.30}$$

Both functionals, in (8.29), (8.30), are particular cases of the following family of functionals, parametrized by $\alpha(>0)$ and $f=\{f_1, f_2\}$:

$$J(\alpha, \underset{\sim}{f}; \xi, \eta) = \frac{\alpha}{2}\rho\int_0^L (\xi^2+\eta^2)\,ds + \frac{EI}{2}\int_0^L (\xi''^2+\eta''^2)\,ds - \int_0^L (f_1\xi + f_2\eta)\,ds. \tag{8.31}$$

Concerning the possible multiplicity of the solutions of (8.5) (resp. (8.27)), we concentrate on those solutions, *local minimizers* of the functional (8.29) (resp. (8.30)) on $\mathscr{E}^{n+1}$ (resp. $\tilde{\mathscr{E}}^n$), which are obtained via an iterative process with initialization by $\{x^n, y^n\}$ (resp. $\{\tilde{x}^{n-1}, \tilde{y}^{n-1}\}$ or $\{x^n, y^n\}$). We are in fact looking for the local minimizer which is the closest to the solution of the previous time-step; we are fully conscious that the above selection of solutions may be somewhat controversial, but the numerical results obtained using this approach (see § 9.4) are in full agreement with what could have been predicted by mechanical intuition (at least for the example that we have considered).

From the above considerations, we can solve (8.5) and (8.27) (in fact their variants obtained by space discretization) by iterative methods *identical* (only the Lagrangian functionals are slightly different) to those described in § 6; it is therefore not necessary to describe them again.

9. Numerical experiments.

9.1. Synopsis. We describe and discuss in this section the numerical results obtained for the solution of some test problems, using the methods described in the previous sections.

Section 9.2 is concerned with the *static* problem (2.2), discussed in §§ 2, 3, 4, 5, 6; § 9.3, with the *static* problem *with stream* (7.7). In § 9.4 we present the results obtained for a specific *time dependent problem* (8.1), (8.2) (see § 8.2), using the Houbolt scheme of § 8.3. Finally some additional comments are given in § 9.5.

9.2. Numerical solution of some static problems (2.2).

9.2.1. Description of the problems.

Mechanical parameters:

$$EI = 7000\ \text{Nm}^2, \qquad \rho = 7.67\ \text{Kg/m}, \qquad L = 32.6\ \text{m}.$$

Boundary conditions:

$$x(0)=y(0)=0, \qquad x'(0)=1, \qquad y'(0)=0,$$

$$x(L)=1, 2, 3, 4, 5, 6, 7, 8, \qquad y(L)=0, \qquad x'(L)=1, \qquad y'(L)=0.$$

9.2.2. Additional information about the numerical process. We have used a *uniform mesh*, with $h = L/50$, and approximated the inextensibility condition by (5.3). The approximation problems (5.4) have been solved by a discrete variant of algorithm (6.12)–(6.16), with $\tilde{\rho} = r = 50{,}000$ and $\omega = 1$. As *termination criterion* we have used (with obvious notation)

$$\frac{\sum_i \{|x_i^n - x_i^{n-1}| + |y_i^n - y_i^{n-1}| + |x_i'^n - x_i'^{n-1}| + |y_i'^n - y_i'^{n-1}|\}}{\sum_i \{|x_i^n| + |y_i^n| + |x_i'^n| + |y_i'^n|\}} \leqq 10^{-5}. \tag{9.1}$$

9.2.3. Presentation of the numerical results. (i) We show in Fig. 9.1, the numerical results obtained as follows, for $x(L) = 2, 3, 4, 5, 6$:

We first computed the solution corresponding to $x(L) = 6$, using the initialization (6.12), $\lambda^1 = \mu^1 = 0$ and $\{x^0, y^0\}$ given by

$$x^0(s) = 3\left(1 - \cos \pi \frac{s}{L}\right), \qquad y^0(s) = -3 \sin \pi \frac{s}{L} \tag{9.2}$$

which corresponds to a half-circle of diameter AB; since the length of this half-circle is $3\pi = 9.424 \cdots$, we can see that our initial solution is fairly far from the actual solution; in fact convergence was achieved in 187 iterations of algorithm (6.12)–(6.16). For $x(L) = 5, 4, 3, 2$ (we proceeded in that order) we used a kind of *incremental* technique, since the initialization of (6.12)–(6.16) was done using the results obtained for the previous value of $x(L)$.

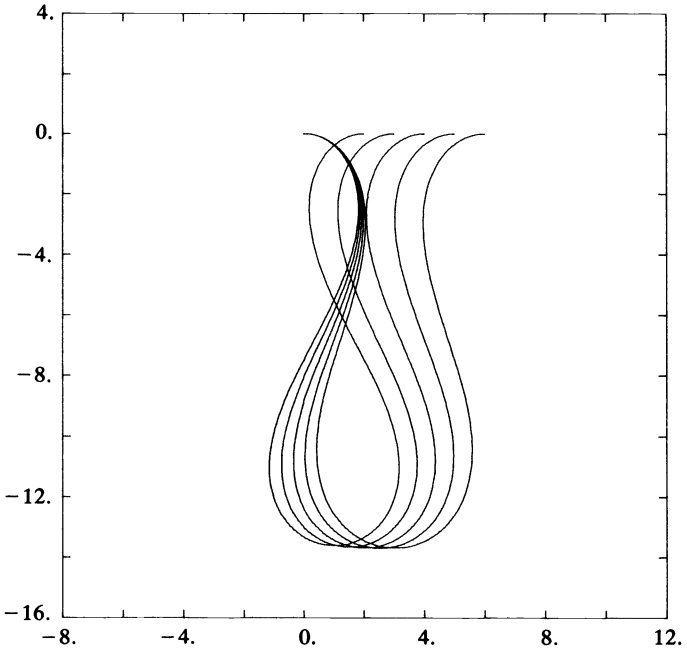


FIG. 9.1. ($x(L) = 2, 3, 4, 5, 6$).

For clarity we have indicated in Figs. 9.2, 9.3, 9.4 the solutions corresponding to $x(L) = 6, 4, 2$ respectively.

We have indicated in Table 9.1 the number of iterations necessary to obtain convergence according to the termination criterion (9.1).

TABLE 9.1

$x(L)$	Number of iterations
6	166
5	105
4	105
3	107
2	109

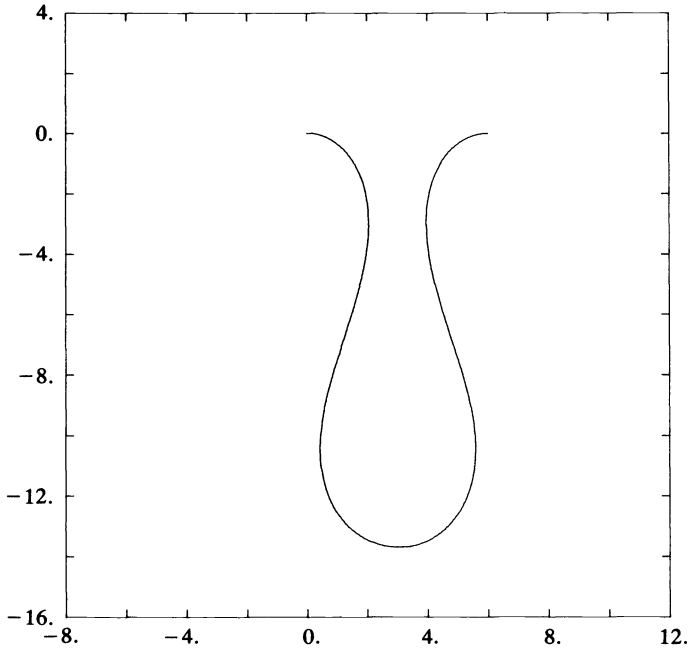


FIG. 9.2. ($x(L) = 6$)

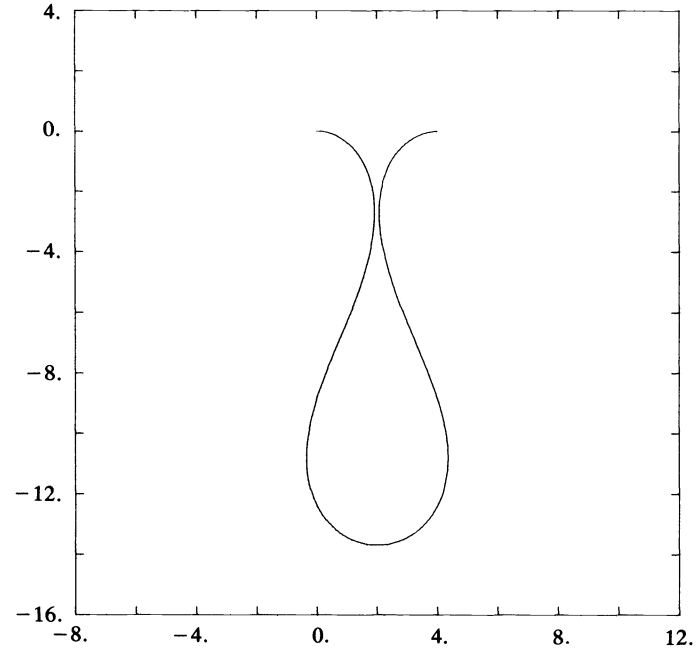


FIG. 9.3. ($x(L) = 4$)

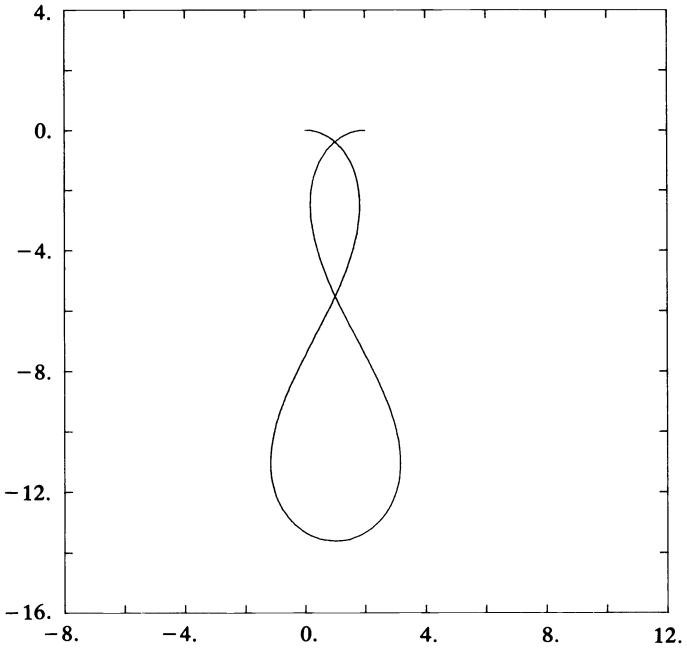


FIG. 9.4. ($x(L) = 2$)

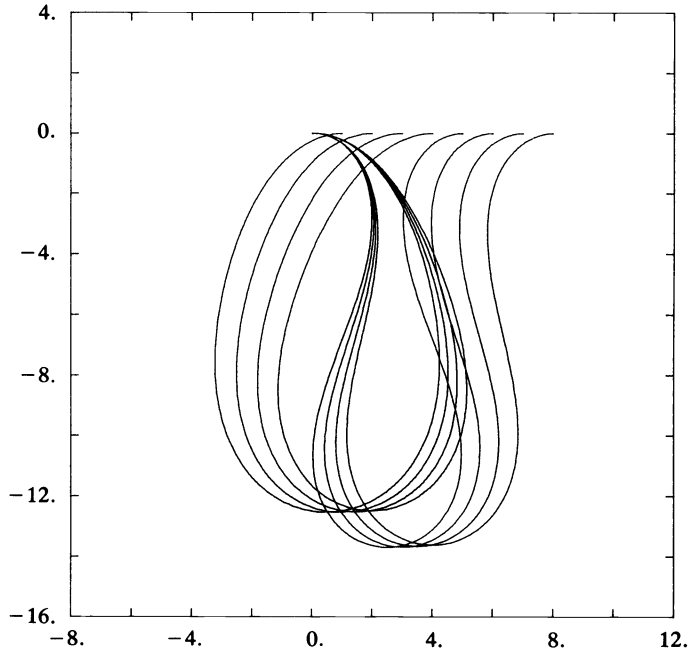


FIG. 9.5. ($x(L) = 1, 2, 3, 4, 5, 6, 7, 8$)

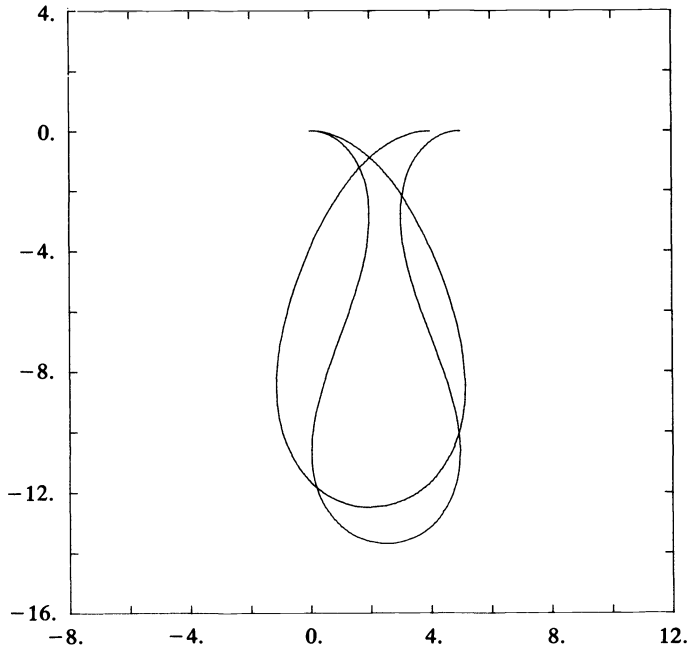


FIG. 9.6. ($x(L) = 4, 5$).

The five calculations in Table 9.1. were done in *one* computer run, corresponding to 3 minutes of CII/IRIS 80.

(ii) We show in Fig. 9.5, the numerical results obtained as follows for $x(L) = 1, 2, 3, 4, 5, 6, 7, 8$:

Each calculation is done using $\lambda^1 = \mu^1 = 0$ and $\{x^0, y^0\}$ corresponding to the lower half-circle of diameter AB to initialize (6.12)–(6.16); we start, therefore, very far from the actual solution and we proceed without incremental strategy. We observe in Fig. 9.5 two types of shapes for the calculated solutions (corresponding to different branches of solutions); we observe also that if $x(L)$ is sufficiently small, then the present solutions differ from the solutions obtained in (i). Since the critical value of $x(L)$ with regard to the above phenomenon seems to be between 4 and 5 we have shown in Fig. 9.6 the solutions

TABLE 9.2

$x(L)$	Number of iterations
1	220
2	220
3	220
4	220
5	133
6	166
7	179
8	187

calculated for $x(L) = 4$ and 5. Table 9.2 gives the number of iterations necessary to obtain convergence.

The eight calculations in Table 9.2 were done in *one* computer run, the corresponding computation time being 7 minutes on a CII/IRIS 80.

9.2.4. Further remarks. We have indicated in Table 9.3 the values taken by the functional J (of (3.7)) at those solutions of (2.2) discussed in § 9.2.3.

Table 9.3 shows the (not very surprising) following fact: using an incremental strategy we have been able to describe a branch of solutions, despite the fact that for the same values of $x(L)$, more stable solutions exist. In § 9.5, we try to explain this behavior using the initial value interpretation of algorithm (6.12)–(6.16) given in § 6.3.

TABLE 9.3

$x(L)$	8	7	6	5	4	3	2	1
Incremental strategy (case (i))	−8,561	−8,142	−7,688	−7,199	−6,674	−6,112	−5,510	−4,868
Non incremental strategy (case (ii))	−8,561	−8,142	−7,688	−7,199	−9,434	−9,702	−9,932	−10,124

9.3. Numerical solution of static problems with stream.

9.3.1. Generalities. The static problem with a horizontal stream has been formulated in § 7.3; in § 7.4 we described several iterative methods to solve the continuous problem (7.7), (7.8). Compared to the static problem (2.2), the discretization of (7.7), (7.8) requires an appropriate treatment of the hydrodynamical forces, since these forces are formulated via complicated formulas (see (7.4), (7.5)). We have used *numerical integration* to discretize the corresponding *virtual work terms* in formulation (7.7).

9.3.2. A first example.

9.3.2.1. Description of the problems.

$$EI = 4{,}500\ \text{Nm}^2, \qquad \rho = 82\ \text{Kg/m}, \qquad L = 32\ \text{m}, \qquad D = 0.277\ \text{m}.$$

$$\underline{V} = \{1\ \text{m/s}, 0\}, \qquad \{0, 0\}, \qquad \{-1\ \text{m/s}, 0\}.$$

Boundary conditions:

$$x(0) = 0, \qquad y(0) = 10, \qquad x'(0) = 0, \qquad y'(0) = -1,$$

$$x(L) = 0, 3.6, 10, \qquad y(L) = 25, \qquad x'(L) = 0, \qquad y'(L) = 1.$$

9.3.2.2. Additional information about the numerical process. We have used a *uniform mesh*, with $h = L/50$, and approximated the inextensibility condition by (5.3). The approximation problems have been solved by a discrete variant of (7.15)–(7.24) with only *one inner iteration* in (7.20)–(7.24); we used $\tilde{\rho} = r = 5{,}000$ and $\omega = 1$. The

termination criterion is (9.1).

9.3.2.3. Numerical results. We have shown in Fig. 9.7 (resp. Figs. 9.8, 9.9) the numerical results corresponding to $\underline{V} = \{0, 0\}$ (resp. $\{1, 0\}$, $\{-1, 0\}$), the values of $x(L)$ being 0, 3.6 and 10 meters.

The convergence is obtained, according to (9.1), in about 100 iterations (resp. 150 iterations) if the hydrodynamical forces due to the stream hold (resp. don't hold); hence *the presence of a stream accelerates the convergence of the iterative methods.* An explanation of this phenomenon will be given in § 9.3.4.

9.3.3. A second example. This second example is more complicated than the first one.

9.3.3.1. Description of the problems.
Mechanical parameters:

$$EI = 9{,}000\ \text{Nm}^2, \qquad \rho = 11.4\ \text{Kg/m}, \qquad D = 0.112\ \text{m}, \qquad L = 207.5\ \text{m}.$$

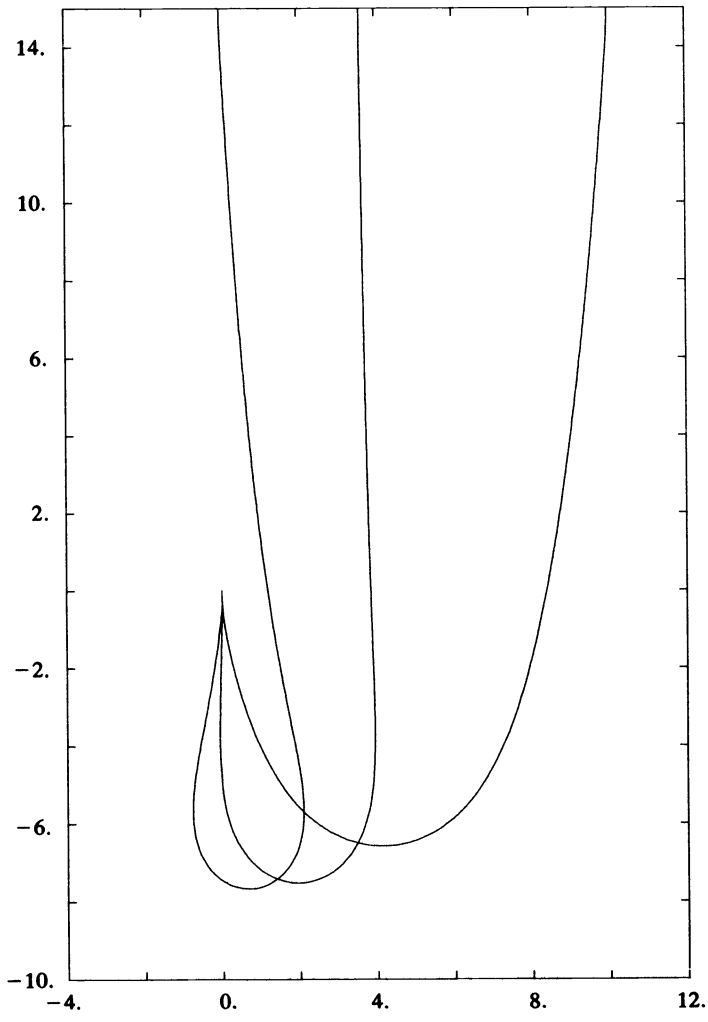


FIG. 9.7. ($\underline{V} = \{0, 0\}$).

The stream is still horizontal but its velocity depends upon the depth as follows: $\underset{\sim}{V}(180) = \pm\{1.8, 0\}$, $\underset{\sim}{V}(160) = \pm\{1.8, 0\}$, $\underset{\sim}{V}(90) = \pm\{1.2, 0\}$, $\underset{\sim}{V}(0) = \pm\{.75, 0\}$; $y = 180$ corresponds to the surface of the sea and we assume a *linear variation* of $V(y)$ between the values indicated above. We suppose also that the stream is only acting if $0 \leqq y \leqq 180$, i.e., $\underset{\sim}{V}(y) = \{0, 0\}$ if $y > 180$.

Boundary conditions:

$$x(0) = 0, \quad y(0) = 0, \quad x'(0) = 0, \quad y'(0) = 1,$$

$$x(L) = 0, -24, \quad y(L) = 196{,}200, \quad x'(L) = 0, \quad y'(L) = 1.$$

9.3.3.2. Additional information about the numerical process. We used a *uniform mesh*, with $h = L/100$; the inextensibility condition is still approximated by (5.3). The approximation problems have been solved, as in § 9.3.2.2, by a discrete variant of (7.15)–(7.24), using *one* (resp. *at most three*) *inner iteration* in (7.20)–(7.24) if a *stream is acting* (resp. *not acting*) on the pipe; we have used $\tilde{\rho} = r = 10^5$, $\omega = 1$. The termination criterion is still (9.1).

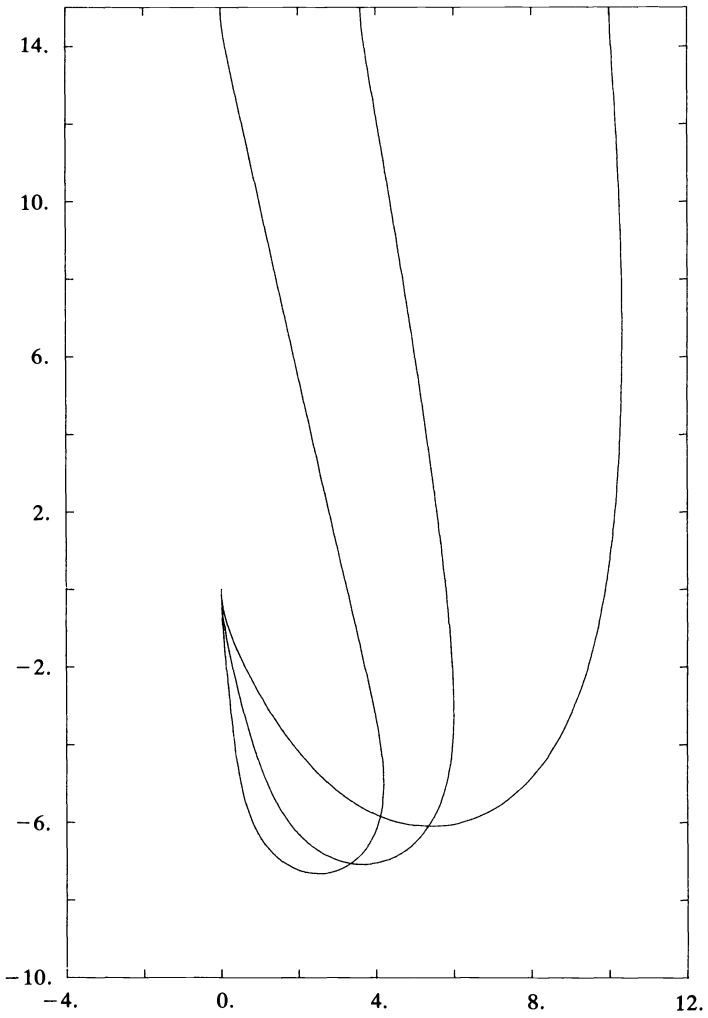


FIG. 9.8. ($\underset{\sim}{V} = \{1, 0\}$)

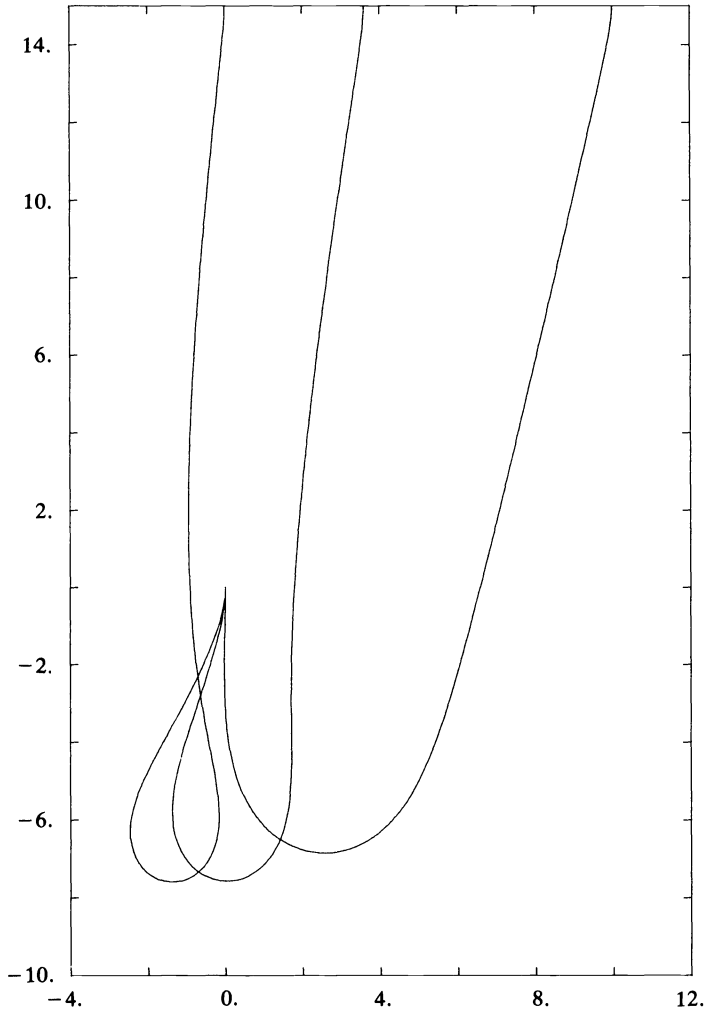


FIG. 9.9. ($\underline{Y} = \{-1, 0\}$)

9.3.3.3. Numerical results. We have shown in Fig. 9.10 (resp. Fig. 9.11) the numerical results corresponding to $x(L) = 0$, $y(L) = 200$ (resp. $x(L) = -24$, $y(L) = 196$); for each of these two cases, we have calculated the solutions corresponding to *no stream*, the *horizontal component* of $\underline{Y}$ is *positive*, the *horizontal component* of $\underline{Y}$ is *negative*, respectively.

We observe in Fig. 9.10 the symmetry of the two solutions corresponding to a positively and negatively oriented stream.

As in § 9.3.2., *the presence of a stream reduces the number of interations necessary to obtain the convergence*, since about 300 iterations are needed for the calculations without stream, compared to 100 iterations for the calculations with stream; with regards to *computation time*, "stream" calculations are also three times faster than "no stream" calculations.

9.3.4. Further remarks on the convergence of algorithm (7.15)–(7.24). We have seen in § 6.3 that using algorithm (6.12)–(6.16), with $\tilde{\rho} = r$ and $\omega = 1$, to solve the static problem (without stream) (2.2), can be interpreted, in fact, as an *approximate time*

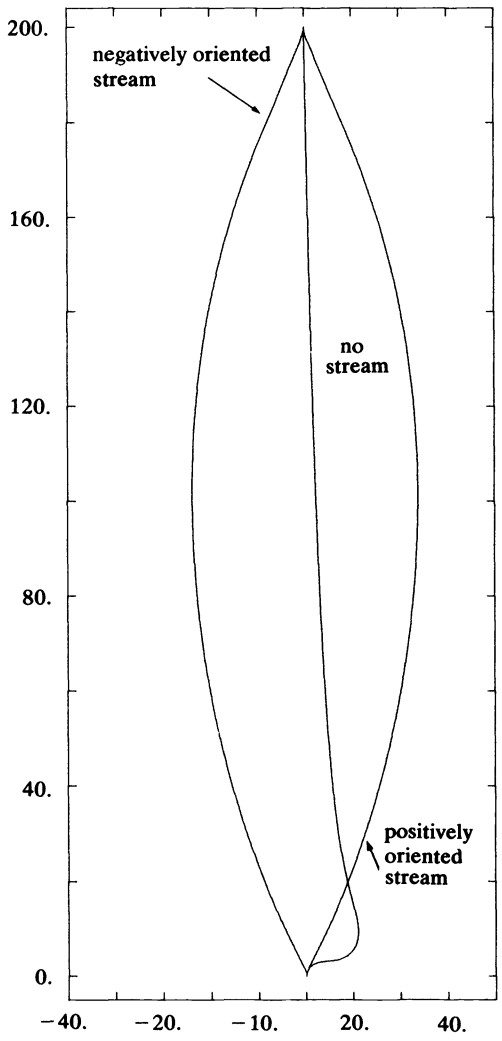


FIG. 9.10. ($x(L)=0$, $y(L)=200$)

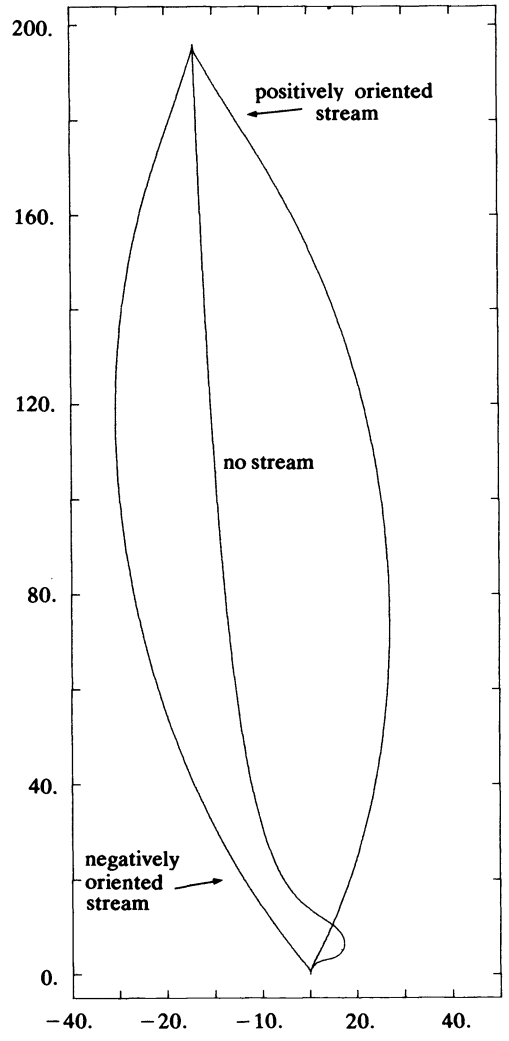


FIG. 9.11. ($x(L)=-24$, $y(L)=196$)

integration of the initial value problem (with $\underset{\sim}{z}=\{x, y\}$)

$$\frac{d}{dt}\underset{\sim}{z}+A(\underset{\sim}{z})=\underset{\sim}{f}, \qquad \underset{\sim}{z}(0)=\underset{\sim}{z}_0 \tag{9.3}$$

by an *implicit scheme of alternating direction* type; in (9.3), A is a (nonlinear) multi-valued operator which takes into account the *elastic bending* and the *inextensibility* of the pipeline; $\underset{\sim}{f}$ is a density of external forces acting on the pipeline. As mentioned in § 6.3, r appears as the *reciprocal of a time step.*

If we now consider the static problem with stream (7.7), (7.8), algorithm (7.15)–(7.24) can also be seen as an approximate time integration process for the following initial value problem

$$\frac{d}{dt}\underset{\sim}{z}+A(\underset{\sim}{z})+B(\underset{\sim}{z})=\underset{\sim}{f}, \qquad \underset{\sim}{z}(0)=\underset{\sim}{z}_0, \tag{9.4}$$

where A and $\underset{\sim}{f}$ are as in (9.3), and where B is a *nonlinear operator* associated to the

hydrodynamical forces; since, for the same value of $\underline{z}(0)$, steady states are reached in a shorter time with B than without, B seems to act as a *dissipative operator* (at least for the examples we have treated). This property, likely related to the fact that B derives from the friction of the water on the pipeline, would justify a mathematical analysis by itself.

To conclude this section we would like to emphasize the outstanding robustness of the algorithms described in § 7.4; to give an idea of this robustness we mention the following anecdote: in the early computations with stream we used (by mistake) a $\underline{Y}$ *ten times larger then the actual one*; algorithm (7.15)–(7.24) had no problem handling that unrealistic situation.

9.4. Numerical solution of a particular dynamical problem (8.1), (8.2).

9.4.1. Generalities. We discuss in this section the numerical results obtained when applying the Houbolt scheme of § 8.3.2. to discrete variants of the wave problem (8.1), (8.2) obtained via the finite element approximation of § 5; we have used in particular (5.3) to approximate the inextensibility condition. To approximate the discrete variants of the *acceleration* term $\int_0^L (\ddot{x}\xi + \ddot{y}\eta)\, ds$ we have used on each sub-interval $[s_i, s_{i+1}]$, $i = 0, 1, \cdots, N-1$, *a numerical integration procedure*, exact for the product of polynomials of degree $\leqq 3$.

9.4.2. Description of the problem.

Mechanical parameters:

$$EI = 700\ \text{Nm}^2, \qquad \rho = 7.67\ \text{Kg/m}, \qquad L = 32.6\ \text{m}.$$

Boundary conditions:

$$x(0) = y(0) = 0, \qquad x'(0) = 1, \qquad y'(0) = 0,$$

$$x(L) = 20, \qquad y(L) = 0, \qquad x'(L) = 1, \qquad y'(L) = 0.$$

9.4.3. Initial data. Using the notation of (8.2), we took

$$\{x(0), y(0)\} = \{x_0, y_0\}, \qquad \{\dot{x}(0), \dot{y}(0)\} = \{x_1, y_1\} = \{0, 0\}$$

with $\{x_0, y_0\}$ defined as follows: $\{x_0, y_0\}$ is the solution of a static problem of type (2.2), in which in addition to gravity, the pipeline is subjected to a concentrated vertical force of 50,000 Newtons, positively oriented, applied at the point $\{x(8), y(8)\}$ of the pipe; $\{x_0, y_0\}$ can be seen in Figs. 9.12, 9.14. It is clear that these initial data satisfy the compatibility condition (8.3).

9.4.4. Additional information about the numerical process. The *space discretization* is similar to § 9.2.2 (i.e., $h = L/50$); the *time discretization* is based on the Houbolt scheme of § 8.3.2., coupled to the Crank–Nicolson *starting procedure* of § 8.3.3.; we have used $\Delta t = 0.2$ second. The displacement $\{x^n, y^n\}$ at $t = n\Delta t$ has been computed by an approximate variant of algorithm (6.9)–(6.11), taking $\tilde{\rho} = r = 20{,}000$ and using as initializer the solution of the previous time step.

9.4.5. Numerical results. We have shown in Figs. 9.12, 9.13, 9.14, the positions occupied by the pipe at the different time steps. We observe that the pipeline is first falling and then oscillating from right to left and conversely.

For the above problem the computational cost of a single time step is at the moment 20 s. on CII/IRIS 80; since the number of iterations is between 30 and 100 for each time step, we have the feeling that if the value of r can be optimally adjusted it will imply a substantial computer time saving; we are presently working at that optimal choice of r as a function of Δt.

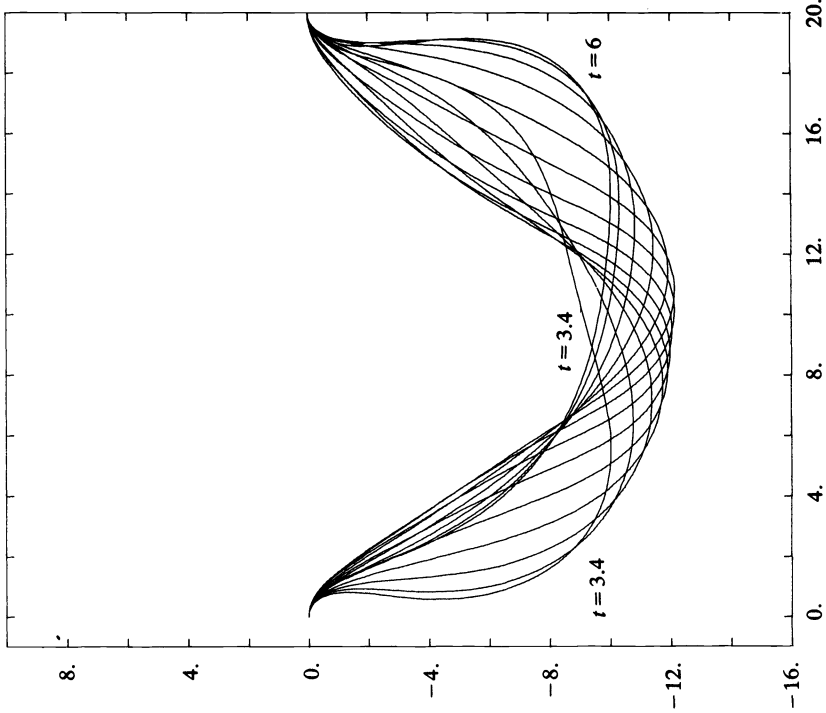


FIG. 9.13. ($3.4 \leqq t \leqq 6$)

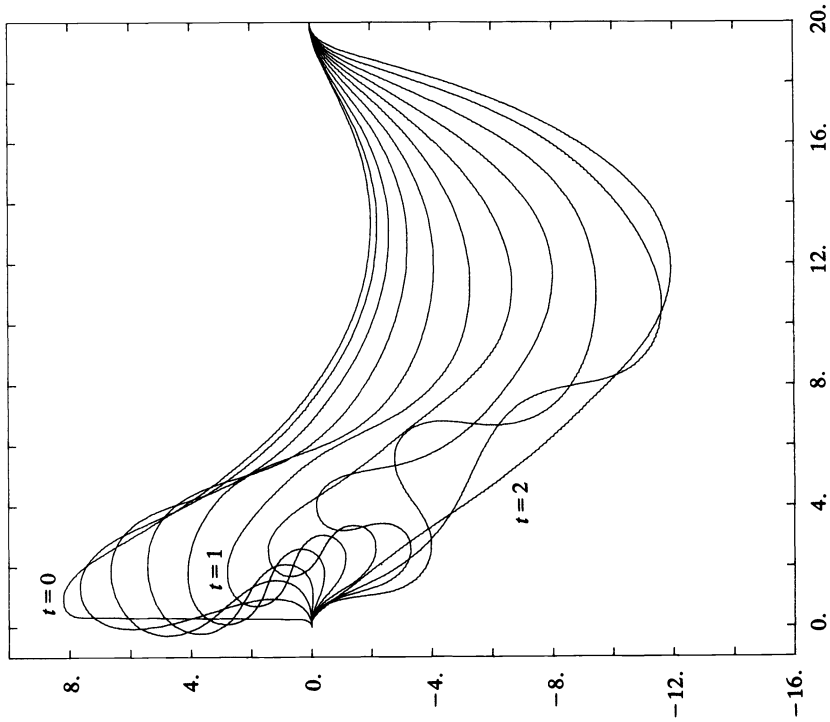


FIG. 9.12. ($0 \leqq t \leqq 2$)

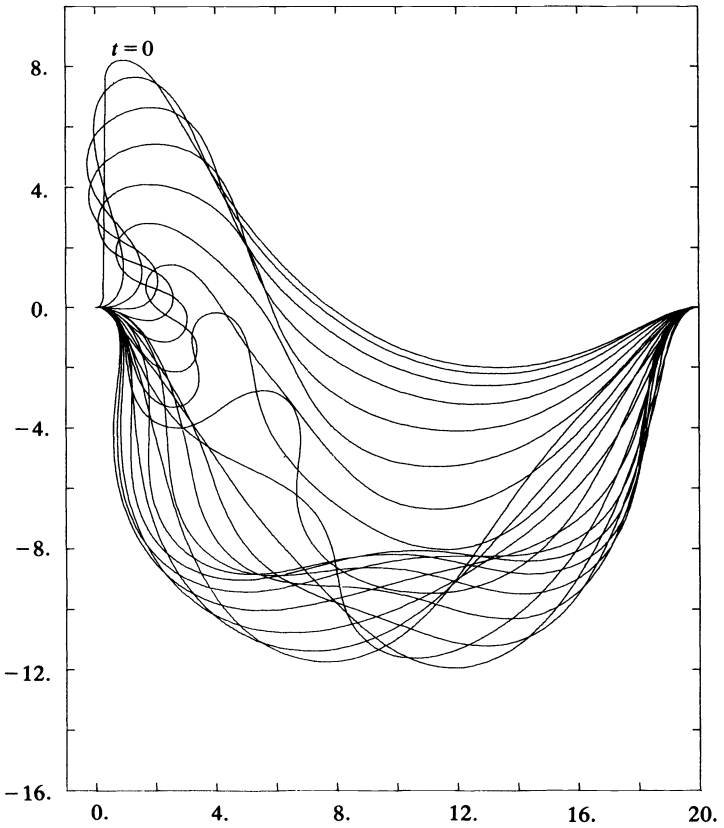


FIG. 9.14. ($0 \leqq t \leqq 4$)

9.4.6. Remarks on the Crank–Nicolson scheme. Some very recent results obtained on the test problem of § 9.4.2, with the Crank–Nicolson scheme of § 8.3.4., seem to indicate a greater stability for the Houbolt scheme; this fact is not surprising since the Crank–Nicolson scheme that we are using corresponds to $\theta = .25$ in (8.28), which is very close to the limit between conditionally and unconditionally stable Crank–Nicolson schemes.

9.5. Additional comments on the behavior of algorithms (6.9)–(6.11) and (6.12)–(6.16). We use the notation of § 6.3.4. We have seen, in § 9.2, that for a given value of r, different initial solutions may produce different results, via algorithms (6.9)–(6.11) or (6.12)–(6.16). In fact if $\bar{z} = \{\bar{x}, \bar{y}\}$ is an *isolated local minimizer* of J on $\mathscr{E}$, it is reasonable to suppose that $\bar{z}$ is an *attractor* in that for $z(0)$ sufficiently close to $\bar{z}$ we have for the solution $z(t)$ of (9.3),

$$\lim_{t \to +\infty} z(t) = \bar{z}. \tag{9.5}$$

This asymptotic property is preserved if one uses a convenient approximate time integration method with a sufficiently small time step Δt; from the interpretation of algorithm (6.12)–(6.16) given in § 6.3, it is clear that for $r = 1/\Delta t$ sufficiently large, a similar behavior can be expected if one uses this latter algorithm with initial data sufficiently close to the actual solution. This interpretation can also be used to explain the following fact.

If $r = 1/\Delta t$ is sufficiently small we may converge, using (6.9)–(6.11) or (6.12)–(6.16), to a solution $\underset{\sim}{z}^* = \{x^*, y^*\}$ of (2.2), $\underset{\sim}{z}^* \neq \underset{\sim}{\bar{z}}$, even if (9.5) holds for the continuous problem with the initial data used to obtain $\underset{\sim}{z}^*$ via (6.9)–(6.11) or (6.12)–(6.16) (we usually have $J(x^*, y^*) < J(\bar{x}, \bar{y})$).

10. Further comments. Conclusion. *Pipeline*, *rod* and *cable* problems (*static* and/or *dynamic*) have motivated the following recent engineering works (this list is far from complete): [5], [6], [36], [45] previously mentioned, and also Argyris–Dunne–Angelopoulos [47], Lehner–Batterman [48], Roussel [49], Hitchings–Ward [50], Klosowiak–Machura [51], Sangster–Batchelor [52], Felippa [53], Ali–Shore [54], Merichal–Clement [55]; see also the references in these publications.

For theoretical aspects of *nonlinear elasticity* we refer, among many others, to Prager [56], Dickey [57], Knops [58], Gurtin [59], Truesdell [60], and the references therein; for *rods* and *beams* see [2], and also Antman [61], [62].

Concerning the *numerical integration of time dependent problems*, we complete references [26]–[44] by [7], [8], Fujii [63], Nickell [64], Wellford–Hamdan [65] and also, for a *global view* on numerical methods for time dependent problems, by Richtmyer–Morton [66], Lascaux [67], Kreiss [68].

Augmented Lagrangian methods have been introduced by Hestenes [69] and Powell [70]; the corresponding literature is enormous (see Fortin–Glowinski [71] for a substantial bibliography on the subject); applications to *non convex programming* are considered in Rockafellar [72]. Applications to the numerical solution of nonlinear boundary value problems, via the introduction of artificial linear equality constraints, go back—to our knowledge—to [17]; related results may be found in [18], [22] and also Marrocco [73], Gabay [74]; the relation between this class of methods and ADI is to our knowledge due to Chan–Glowinski [75], [76]. We finally refer to the monograph in preparation of Fortin–Glowinski [71] for more theoretical results, and also applications to the numerical solution of *linear* and *nonlinear boundary value problems.*

10.1. Conclusion. We have shown in this paper that methods inspired by nonlinear programming and more specifically by augmented Lagrangians (in which one combines *duality* and *penalty* techniques), give a most effective way for solving *large displacement problems in elasticity*, where the main difficulty is a differential nonlinear equality constraint. In this paper these methods have also been applied to problems which are not extremization problems (see § 7, 8), producing very good numerical results and proving that this class of methods is not limited to extremization problems. We add that the first author has also devised an adaptive method for adjusting automatically the parameters occurring in the algorithms of § 6, 7; in that direction it would be interesting to adapt the *continuation techniques*, considered in H. B. Keller [78], Rheinboldt [79], Kikuchi [10], [77], Bergan–Holland–Söreide [80], to the problems treated in this paper.

We believe that the methods and ideas in this paper can also be very useful for solving numerically more complicated problems in the nonlinear mechanics of continuous media (as given in the Introduction).

Acknowledgments. We would like to thank Professor L. Tartar for his help at various stages of this work and Professor F. Brezzi for very helpful comments.

REFERENCES

[1] M. FORTIN AND R. GLOWINSKI, *Sur des méthodes de décomposition coordination par lagrangiens augmentés*, Résolution Numérique de Problèmes aux Limites par des Méthodes de Lagrangiens Augmentés, M. Fortin, R. Glowinski, eds., to appear.

[2] S. S. ANTMAN AND G. ROSENFELD, *Global behavior of buckled states fo nonlinearly elastic rods*, SIAM Rev., 20 (1978), pp. 513–566.
[3] T. B. BENJAMIN, *Generic bifurcation theory in fluid mechanics (supplement)*, Contemporary Developments in Continuum Mechanics and Partial Differential Equations, G. M. de la Penha, L. A. J. Medeiros, eds., North-Holland, Amsterdam, 1978, pp. 45–73.
[4] M. S. BERGER, *Nonlinearity and Functional Analysis*, Academic Press, New York, 1977.
[5] H. D. HIBBITT, E. B. BECKER AND L. M. TAYLOR, *Nonlinear analysis of some slender pipelines*, Computer Methods Appl. Mech. Engrg. 17/18 (1979), pp. 203–225.
[6] G. MAIER, F. ANDREUZZI, F. GIANNESSI, L. JURINA AND F. TADDEI, *Unilateral contact, elastoplasticity and complementarity with reference to offshore pipeline design*, Ibid., 17/18, (1979), pp. 469–496.
[7] G. STRANG AND G. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
[8] J. T. ODEN AND J. N. REDDY, *An Introduction of the Mathematical Theory of Finite Elements*, Wiley-Interscience, New York, 1976.
[9] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.
[10] F. KIKUCHI, *Numerical analysis of the finite element method applied to bifurcation problems of turning point type.* Part I, Theoretical Analysis, ISAS Rep. 564, University of Tokyo, 1978, pp. 217–246.
[11] M. YAMAGUTI AND H. FUJII, *On numerical deformation of singularities in nonlinear elasticity*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds., 1977; Lecture Notes in Mathematics, vol. 704, Springer-Verlag, Berlin, 1979, pp. 267–281.
[12] S. KESAVAN, *La méthode de Kikuchi appliquée aux équations de Von Karman*, Numer. Math., 32 (1979), pp. 209–232.
[13] F. BREZZI, J. RAPPAZ AND P. A. RAVIART, *Finite-dimensional approximation of non linear problems*, to appear.
[14] M. J. D. POWELL, *Variable metric methods for constrained optimization*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds., 1977, Lecture Notes in Mathematics, vol. 704, Springer-Verlag, Berlin, 1979, pp. 62–72.
[15] G. STRANG, *Applications of Optimization Techniques to Nonlinear Mechanics*, Proceedings of the 1979 TICOM Conference, Univ. of Texas, Austin, TX, to appear.
[16] A. LICHNEWSKY, *Minimisation de fonctionnelles définies sur une variété par la méthode du Gradient Conjugué*, Thèse de Doctorat d'Etat, Université Paris-Sud, 1979.
[17] R. GLOWINSKI AND A. MARROCCO, *Sur l'approximation par éléments finis d'ordre un et la résolution par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, Rev. Française Automat. Informat. Recherche Opérationnelle, R-2 (1975), pp. 41–76.
[18] D. GABAY AND B. MERCIER, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl. (1976), pp. 17–40.
[19] R. GLOWINSKI, J. L. LIONS AND R. TREMOLIERES, *Analyse Numérique des Inéquations Variationnelles*, Vol. 1, Dunod-Bordas, Paris, 1976.
[20] I. EKELAND AND R. TEMAM, *Analyse Convexe et Problèmes Variationnels*, Dunod-Gauthier-Villars, Paris, 1974.
[21] J. L. LIONS, *Contrôle Optimal de Systèmes Gouvernés par des Équations aux Dérivées Partielles*, Dunod-Gauthier-Villars, Paris, 1968.
[22] R. GLOWINSKI, *Numerical methods for nonlinear variational problems*, Lecture Notes, Tata Institute, Bombay (to appear in 1979/1980).
[23] J. DOUGLAS AND H. H. RACHFORD, *On the numerical solution of heat conduction problems in two or three space variables*, Trans. Amer. Math. Soc., 82 (1956), pp. 421–439.
[24] P. L. LIONS AND B. MERCIER, *Splitting algorithms for the sum of two nonlinear operators*, Rep. 29, Centre de Mathématiques Appliquées, Ecole Polytechnique, Palaiseau, France, 1978.
[25] H. BREZIS, *Opérateurs Maximaux Monotones et Semi-Groupes de Contraction dans les Espaces de Hilbert*, North-Holland, Amsterdam, 1973.
[26] R. W. CLOUGH AND J. PENZIEN, *Dynamics of Structures*, McGraw-Hill, Kogakusha, Tokyo, 1975.
[27] J. H. ARGYRIS AND P. C. DUNNE, *Some contributions to nonlinear solid mechanics*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds.; Lecture Notes in Computer Sciences, vol. 10, Springer-Verlag, Berlin, 1974, pp. 42–139.
[28] J. T. ODEN, *Formulation and application of certain primal and mixed finite element models of finite deformations of elastic bodies*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds., 1977; Lecture Notes in Computer Sciences, vol. 10, Springer-Verlag, Berlin, 1974, pp. 334–363.

[29] K. J. BATHE AND E. L. WILSON, *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
[30] O. C. ZIENKIEWICZ, *The Finite Element Method in Engineering Sciences*, McGraw-Hill, London, 1978.
[31] M. GERADIN, *Special applications of Hamilton's principle to structural dynamics*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds., 1977, Lecture Notes in Mathematics, vol. 704, Springer-Verlag, Berlin, 1979, pp. 222–238.
[32] R. W. CLOUGH AND E. L. WILSON, *Dynamic analysis of large structural systems with local non-linearities*, Comput. Methods, Appl. Mech. Engrg. 17/18 (1979), pp. 107–130.
[33] T. J. R. HUGHES, K. S. PISTER AND T. L. TAYLOR, *Implicit-explicit finite elements in nonlinear transient analysis*, Ibid., 17/18 (1979), pp. 159–182.
[34] T. BELYTSCHKO, H. J. YEN AND R. MULLEN, *Mixed methods for time integration*, Ibid., 17/18, (1979), pp. 259–276.
[35] C. A. FELIPPA AND K. C. PARK, *Direct time integration methods in nonlinear structural dynamics*, Ibid., 17/18 (1979), pp. 277–314.
[36] G. SANDER, M. GERADIN, C. NYSSEN AND M. HOGGE, *Accuracy versus computational efficiency in nonlinear dynamics*, Ibid., 17/18 (1979), pp. 315–340.
[37] J. H. ARGYRIS, J. S. DOLTSINIS, W. C. KNUDSON, L. E. VAZ AND K. J. WILLAM, *Numerical solution of transient nonlinear problems* Ibid., 17/18, (1979), pp. 341–410.
[38] G. B. WARBURTON, *The influence of the finite element method on developments in structural dynamics*, Energy methods in finite element analysis, R. Glowinski, E. Y. Rodin, O. C. Zienkiewicz, eds., John Wiley, Chichester, 1979, pp. 59–80.
[39] S. MCKEE, *High accuracy ADI methods for hyperbolic equations with variable coefficients*, J. Inst. Math. Appl., 11 (1973), pp. 105–109.
[40] M. CIMENT AND S. H. LEVENTHAL, *Higher order compact implicit schemes for the wave equation*, Math. Comp., 29 (1978), pp. 986–994.
[41] M. LEES, *Alternating direction methods for hyperbolic differential equations*, J. Indian Math. Soc. 10 (1962), pp. 610.
[42] M. K. JAIN, R. AHUJA AND S. BHATTACHARYA, *Difference schemes for second order hyperbolic equations*, Internat. J. Numer. Methods Engrg. 10 (1976), pp. 960–964.
[43] S. R. K. IYENGAR AND R. C. MITTAL, *High order difference schemes for the wave equation*, Ibid., 12 (1978), pp. 1623–1628.
[44] A. N. KONOVALOV, Dokl. Akad. Nauk., SSSR, 147 (1962), p. 25.
[45] B. NATH AND C. H. SOH, *Seismic response analysis of offshore pipelines in contact with the sea-bed*, Internat. J. Numer. Methods Engrg. 13 (1978), pp. 181–196.
[46] G. DAHLQUIST, *On accuracy and unconditional stability of linear multistep methods for second order differential equations*, BIT, 18 (1978), pp. 133–136.
[47] J. H. ARGYRIS, P. C. DUNNE AND T. ANGELOPOULOS, *Nonlinear oscillations using the finite element technique*, Computer Methods, Appl. Mech. Engrg. 2 (1973), pp. 203–250.
[48] J. R. LEHNER AND S. C. BATTERMAN, *Static and dynamic finite deformations of cables using rate equations*, Ibid., 2 (1973), pp. 349–366.
[49] P. ROUSSEL, *Numerical solution of static and dynamic equations of cables*, Ibid., 9 (1976), pp. 65–74.
[50] D. HITCHINGS AND P. WARD, *The nonlinear steady-state response of cable networks*, Ibid., 9 (1976), pp. 191–202.
[51] T. KLOSOWIAK AND M. MACHURA, *The use of minimization methods in the two-dimensional cross-spring hinge problem*, Ibid., 12 (1977), pp. 337–352.
[52] K. G. SANGSTER AND B. V. BATCHELOR, *Nonlinear dynamic analysis of 3-D cable networks*, Computational Methods in Nonlinear Mechanics, TICOM, Univ. of Texas, Austin, TX, 1974, pp. 301–310.
[53] C. A. FELIPPA, *Finite element analysis of three-dimensional cable structures*, Computational Methods in Nonlinear Mechanics, TICOM, Univ. of Texas, Austin, TX, 1974, pp. 311–324.
[54] A. ALI AND S. SHORE, *Forced nonlinear vibrations of sagged cables*, Computational Methods in Nonlinear Mechanics, TICOM, Univ. of Texas, Austin, TX, 1974, pp. 325–337.
[55] D. MARICHAL AND A. CLEMENT, *Etude tridimensionnelle de la forme et des tensions de câbles en équilibre dans un courant non uniforme*, J. Méc. Appl., 1 (1977), pp. 209–223.
[56] W. PRAGER, *Introduction to Mechanics of Continua*, Ginn and Co., Boston, 1961.
[57] R. W. DICKEY, *Bifurcation Problems in Nonlinear Elasticity*, Pitman, London, 1976.
[58] R. J. KNOPS, ed., *Nonlinear Analysis and Mechanics: Heriot–Watt Symposium*, Volumes I, II, II, Pitman, London, 1978–1979.

[59] M. E. GURTIN, *On the nonlinear theory of elasticity*, Contemporary Developments in Continuum Mechanics and Partial Differential Equations, G. M. de La Penha, L. A. J. Medeiros, eds., North-Holland, Amsterdam, 1978, pp. 237–253.

[60] C. TRUESDELL, *Some challenges offered to analysis by rational thermomechanics*, Contemporary Developments in Continuum Mechanics and Partial Differential Equations, G. M. de La Penha, L. A. J. Medeiros, eds. North-Holland, Amsterdam, 1978, pp. 495–603.

[61] S. ANTMAN, *The theory of rods*, Handbuch Der Physic, Vol. VI a/2, Springer-Verlag, Berlin, 1972, pp. 641–703.

[62] S. ANTMAN, *Kirchoff's problem for nonlinearly elastic rods*, Quart. Appl. Math., 32 (1974), pp. 221–240.

[63] H. FUJII, *Finite elements schemes: stability and convergence*, Advances in Computational Methods in Structural Mechanics and Design, University of Alabama Press, Huntsville, AL, 1972, pp. 201–218.

[64] R. E. NICKELL, *Nonlinear dynamics by mode superposition*, Computer Methods Appl. Mech. Engrg, 7 (1976), pp. 107–129.

[65] L. C. WELLFORD AND S. M. HAMDAN, *An analysis of an implicit finite element algorithm for geometrically nonlinear problems of structural dynamics*, Ibid., 14 (1978), pp. 377–399.

[66] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial Value Problems*, Wiley, New York, 1967.

[67] P. LASCAUX, *Numerical methods for time dependent equations: Applications to fluid flow problems*, Lecture Notes, vol. 52, Tata Institute, Bombay, 1976.

[68] H. O. KREISS, *Numerical methods for solving time-dependent problems for partial differential equations*, Séminaire de Mathématiques Supérieures, vol. 65, Presses de l'Université de Montréal, Montréal, 1978.

[69] M. HESTENES, *Multiplier and gradient methods*, J. Optimization Theory Appl., 4 (1969), pp. 303–320.

[70] M. J. D. POWELL, *A method for nonlinear constraints in minimization problems*, Optimization, R. Fletcher, ed., Academic Press, London, 1969. Chapter 19.

[71] M. FORTIN AND R. GLOWINSKI, Eds., *Résolution numérique des problèmes aux limites par des méthodes de lagrangiens augmentés*, to appear.

[72] T. R. ROCKAFELLAR, *Augmented Lagrange multiplier functions and duality in non convex programming*, SIAM J. Control Optimization, 12 (1974), pp. 268–285.

[73] A. MARROCCO, *Expériences numériques sur des problèmes non-linéaires résolus par éléments finis et lagrangien augmenté*, Rapport Laboria No. 309, Le Chesnay, France, Mai 1978.

[74] D. GABAY, *Résolution et décomposition des inéquations variationnelles par des méthodes de multiplicateur*, to appear.

[75] T. CHAN AND R. GLOWINSKI, *Numerical methods for a class of mildly nonlinear elliptic equations*, Atos Do Decimo Primeiro Coloquio Brasiliero de Matematica, Vol. I, Impa, Rio de Janeiro, 1978, pp. 279–318.

[76] T. CHAN AND R. GLOWINSKI, *Finite Element approximation and iterative solution of a class of mildly nonlinear elliptic equations*, STAN-CS-78-674, Computer Science Department, Stanford University, Stanford, CA, 1978.

[77] F. KIKUCHI, *Finite element approximation of bifurcation problems*, Functional Analysis and Numerical Analysis, Japan–France Seminar, Tokyo, and Kyoto, 1976, H. Fujita ed., Japan Society for the Promotion of Science, 1978, pp. 203–222.

[78] H. B. KELLER, *Constructive methods for bifurcation and nonlinear eigenvalue problems*, Computing Methods in Applied Sciences and Engineering, Part I, R. Glowinski, J. L. Lions, eds., 1977; Lecture Notes in Mathematics, Springer-Verlag, Berlin, 1979, pp. 241–251.

[79] W. C. RHEINBOLDT, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal. 15 (1978), pp. 1–11.

[80] P. G. BERGAN, I. HOLLAND AND T. H. SÖREIDE, *Use of the current stiffness parameter in solution of nonlinear problems*, Energy Methods in Finite Element Analysis, R. Glowinski, E. Y. Rodin, O. C. Zienkiewicz, eds., Wiley, Chichester, 1979, pp. 265–282.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0004 $01.00/0

NUMERICAL COMPUTATION OF THE SCHWARZ–CHRISTOFFEL TRANSFORMATION*

LLOYD N. TREFETHEN†

Abstract. A program is described which computes Schwarz–Christoffel transformations that map the unit disk conformally onto the interior of a bounded or unbounded polygon in the complex plane. The inverse map is also computed. The computational problem is approached by setting up a nonlinear system of equations whose unknowns are essentially the "accessory parameters" z_k. This system is then solved with a packaged subroutine.

New features of this work include the evaluation of integrals within the disk rather than along the boundary, making possible the treatment of unbounded polygons; the use of a compound form of Gauss–Jacobi quadrature to evaluate the Schwarz–Christoffel integral, making possible high accuracy at reasonable cost; and the elimination of constraints in the nonlinear system by a simple change of variables.

Schwarz–Christoffel transformations may be applied to solve the Laplace and Poisson equations and related problems in two-dimensional domains with irregular or unbounded (but not curved or multiply connected) geometries. Computational examples are presented. The time required to solve the mapping problem is roughly proportional to N^3, where N is the number of vertices of the polygon. A typical set of computations to 8-place accuracy with $N \leqq 10$ takes 1 to 10 seconds on an IBM 370/168.

Key words. conformal mapping, Schwarz–Christoffel transformation, Laplace equation, Gauss-Jacobi quadrature

1. Introduction. One of the classical applications of complex analysis is conformal mapping: the mapping of one open region in the complex plane $\mathbb{C}$ onto another by a function which is analytic and one-to-one and has a nonzero derivative everywhere. Such a map preserves angles between intersecting arcs in the domain and image regions; hence the name conformal. The Riemann mapping theorem asserts that any simply connected region in the plane which is not all of $\mathbb{C}$ can be mapped in this way onto any other such region. The problem of constructing such a mapping, however, is in general difficult. We consider here the special case in which the range is the interior of a polygon, where the problem can be considerably simplified.

Suppose that we seek a conformal map from the unit disk in the z-plane to the interior of a polygon P in the w-plane whose vertices are $w_1, \cdots, w_N$, numbered in counterclockwise order. For each k, denote by $\beta_k \pi$ the exterior angle of P at w_k, as indicated in Fig. 1. For any polygon we have a simple relationship among the numbers β_k:

$$\sum_{k=1}^{N} \beta_k = 2. \tag{1.1}$$

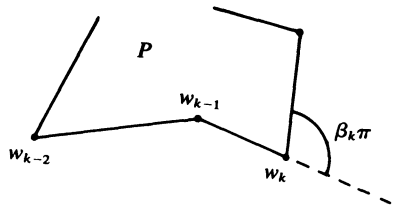


FIG. 1

* Received by the editors May 8, 1979, and in revised form October 9, 1979.

† Computer Science Department, Stanford University, Stanford, California 94305. This work was supported in part by the Office of Naval Research under Contract N00014-75-C-1132, and in part by a National Science Foundation Graduate Fellowship.

If w_k is a finite vertex, we have $-1 \leqq \beta_k < 1$. We shall not require, however, that P be bounded. It may have a number of vertices at complex infinity, and the exterior angles corresponding to these may fall anywhere in the range $1 \leqq \beta_k \leqq 3$. Such angles are defined to be equal to 2π minus the external angle formed in the plane by the intersection of the two sides involved, if they are extended back away from infinity. The example in Fig. 2 should illustrate what is meant by various values of β_k: it is a polygon with five vertices w_k (in this case $w_1 = w_4$), with corresponding values $(\beta_1, \cdots, \beta_5) = (\frac{1}{2}, \frac{4}{3}, \frac{2}{3}, \frac{1}{2}, -1)$. As always, (1.1) holds for this example.

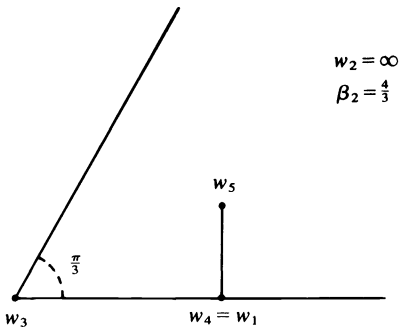


FIG. 2

Let us now pick at random N points z_k ("prevertices") in counterclockwise order around the unit circle and two complex constants C and w_c, and consider the *Schwarz–Christoffel formula*:

$$w = f(z) = w_c + C \int_0^z \prod_{k=1}^{N} \left(1 - \frac{z'}{z_k}\right)^{-\beta_k} dz'. \tag{1.2}$$

The quantities $(1 - z'/z_k)$ always lie in the disk $|w - 1| < 1$ for $|z| < 1$. Therefore, if we choose a branch of $\log(z)$ with a branch cut on the negative real axis by means of which to define the powers in (1.2), $w(z)$ defines an analytic function of z in the disk $|z| < 1$, continuous on $|z| \leqq 1$ except possibly at the vertices z_k.

The Schwarz–Christoffel formula is chosen so as to force the image of the unit disk to have corners in it with the desired exterior angles $\beta_k \pi$. It is not hard to see from (1.2) that at each point z_k, the image $w(z)$ must turn a corner of precisely this angle. This is in keeping with out purpose of mapping the disk onto the interior of P. What the map will in general fail to do is to reproduce the lengths of sides of P correctly, and to be a one-to-one correspondence. Only the angles are guaranteed to come out right.

The variables $z_1, \cdots, z_n$, C, and w_c are the *accessory parameters* of the Schwarz-Christoffel mapping problem. Our first problem—the *parameter problem*—is to determine values of the accessory parameters so that the lengths of sides of the image polygon do come out right. The central theorem of Schwarz–Christoffel transformations asserts that there always exists such a set of accessory parameters.

THEOREM 1 (Schwarz–Christoffel transformation). *Let D be a simply connected region in the complex plane bounded by a polygon P with vertices $z_1, \cdots, z_N$ and exterior angles $\pi\beta_k$, where $-1 \leqq \beta_k < 1$ if z_k is finite and $1 \leqq \beta_k \leqq 3$ if $z_k = \infty$. Then there exists an analytic function mapping the unit disk in the complex plane conformally onto D, and every such function may be written in the form* (1.2).

Proof. The proof is given in [8, Thm. 5.12e].

In fact, for any given polygon there are not just one but infinitely many such conformal mappings. To determine the map uniquely, we may fix exactly three points z_k at will, or fix one point z_k and also fix the complex value w_c, or (as in a standard proof of the Riemann mapping theorem) fix w_c and the argument of the derivative $f'(0)$.

The simplicity of the explicit formula (1.2) is attractive. But because the problem of determining the accessory parameters is intractable analytically, applications of it have almost always been restricted to problems simplified by having very few vertices or one or more axes of symmetry. General Schwarz–Christoffel maps do not appear to have been used as a computational tool, although experiments have been made in computing them.

Problems of numerical conformal mapping have attracted a modest amount of attention for at least thirty years. Gaier [4] produced a comprehensive work describing methods for various problems in this field. For the Schwarz–Christoffel problem, he proposed determining the accessory parameters z_k by setting up a constrained nonlinear system of $N-3$ equations relating (1.2) to the known distances $|w_k - w_j|$, and solving it iteratively by Newton's method [4, p. 171]. Such a procedure has been tried by at least three sets of people; see [7], [10], and [13]. The present work also takes this approach. We believe that this is the first fully practical program for computing Schwarz–Christoffel transformations, however, and the first which is capable of high accuracy without exorbitant cost.

2. Determination of the accessory parameters.

2.1. Formulation as a constrained nonlinear system. The first matter to be settled in formulating the parameter problem numerically is: what parameters in the map (1.2) shall we fix at the outset to determine the Schwarz–Christoffel transformation uniquely? One choice would be to fix three of the boundary points z_k: say, $z_1 = 1$, $z_2 = i$, $z_N = -i$. This normalization has the advantage that the resulting nonlinear system has size only $(N-3)\times(N-3)$, which for a typical problem with $N = 8$ may lead to a solution in less than half the time that a method involving an $(N-1)\times(N-1)$ system requires. Nevertheless, we have chosen here to normalize by the conditions

$$z_N = 1, \qquad w_c = \text{arbitrary point within } P \tag{2.1}$$

which lead to an $(N-1)\times(N-1)$ system. This choice is motivated in part by considerations of numerical scaling: it allows the vertices to distribute themselves more evenly around the unit circle than they might otherwise.(An earlier version of the program mapped from the upper half-plane instead of the unit disk, but was rejected: once points z_k began appearing far from the origin at $x = 10^4$, scaling became a problem.) After a map has been computed according to any normalization, it is of course an easy matter to transform it analytically to a different domain or a different normalization by a Möbius transformation.

Now the nonlinear system must be formulated. The final map must satisfy N complex conditions,

$$w_k - w_c = C\int_0^{z_k} \prod_{j=1}^{N} \left(1-\frac{z'}{z_j}\right)^{-\beta_j} dz', \qquad 1 \leqq k \leqq N. \tag{2.2}$$

These amount to $2N$ real conditions to be satisfied, but they are heavily overdetermined, for the form of the Schwarz–Christoffel formula (1.2) guarantees that the angles will be correct no matter what accessory parameters are chosen. We must reduce the number of operative equations to $N-1$. This is a tricky matter when unbounded polygons are allowed, for one must be careful that enough information about the polygon P is retained that no degrees of freedom remain in the computed solution.

We proceed as follows. First, we require that every connected component of P contain at least one vertex w_k. Thus even an infinite straight boundary must be considered to contain a (degenerate) vertex. This restriction eliminates any translational degrees of freedom. Second, at least one component of P must in fact contain two finite vertices, and w_N and w_1 will be taken to be two such. This restriction eliminates rotational degrees of freedom.

Now define

$$C = (w_N - w_c) \Big/ \int_0^{z_N} \prod_{j=1}^{N} \left(1 - \frac{z'}{z_j}\right)^{-\beta_j} dz', \tag{2.3}$$

where $z_N = 1$ is fixed permanently by (2.1). Next, impose the complex condition (real equations 1, 2)

$$w_1 - w_c = C \int_0^{z_1} \prod_{j=1}^{N} \left(1 - \frac{z'}{z_j}\right)^{-\beta_j} dz'. \tag{2.4a}$$

This amounts to two real equations to be satisfied.

Denote by $\Gamma_1, \cdots, \Gamma_m$ the distinct connected components of P, numbered in counterclockwise order. For each $l \geqq 2$, impose one more complex condition: if z_{k_l} is the last vertex of Γ_l in the counterclockwise direction, then (real equations $3, 4, \cdots, 2m$)

$$w_{k_l} - w_c = C \int_0^{z_{k_l}} \prod_{j=1}^{N} \left(1 - \frac{z'}{z_j}\right)^{-\beta_j} dz'. \tag{2.4b}$$

Finally, $N - 2m - 1$ conditions of side length are imposed. For each pair (z_k, z_{k+1}) beginning at $k = 1$ and moving counterclockwise, where both vertices are finite, we require (real equations $2m+1, \cdots, N-1$)

$$|w_{k+1} - w_k| = \left| C \int_{z_k}^{z_{k+1}} \prod_{j=1}^{N} \left(1 - \frac{z'}{z_j}\right)^{-\beta_j} dz' \right| \tag{2.4c}$$

until a total of $N-1$ conditions have been imposed. If P contains at least one vertex at infinity, then every bounded side will have been represented in a condition of the form (2.4c) except for the side (w_N, w_1), which is already taken care of by (2.1) and (2.4a). If P is bounded, then the last two sides in counterclockwise order—(w_{N-2}, w_{N-1}) and (w_{N-1}, w_N)—will not be so represented.

We have not stated over what contours the integrals of (2.4) are defined. This does not matter mathematically, as the integrand is analytic, but it may matter numerically. In this work we have evaluated them always over the straight line segment between the two endpoints, a procedure which poses no domain problems since the unit disk is strictly convex. Figure 3 illustrates what contours are involved in computing the integrals in (2.3) and (2.4) for a sample case with $N = 10$, $m = 3$.

The nonlinear system is now determined, and its unique solution will give the unknown parameters C and $z_1, \cdots, z_{N-1}$ for the Schwarz–Christoffel mapping. We must, however, take note of two special cases in which the solution is not completely determined by (2.4). It was remarked that if P is bounded, then nowhere in (2.4) does the point w_{N-1} appear. If $\beta_{N-1} \neq -1$ or 0, then this omission is of no consequence for the geometry of the problem forces w_{N-1} to be correct. If $\beta_{N-1} = 0$ or -1, however, then w_{N-1} is not determined *a priori*. The former case is of little consequence, for since $\beta_{N-1} = 0$, the value taken for z_{N-1} has no effect on the computed mapping, as may be seen in (1.2), nor is there any purpose in including w_{N-1} among the vertices of P in the

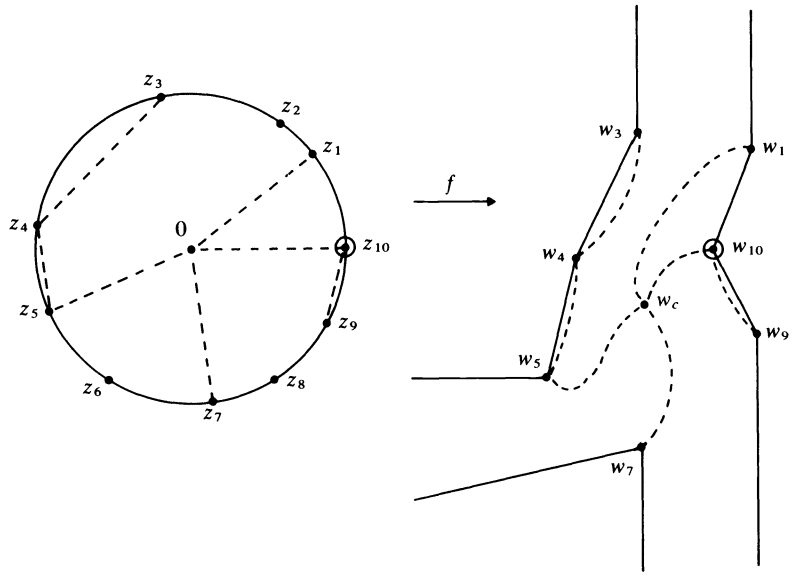


FIG. 3. *Contours of integration within the disk. A sample Schwarz–Christoffel problem is shown with* $N = 10$ *vertices of which* $m = 3$ *vertices are at infinity, illustrating what integrals are computed to evaluate the system* (2.4): (i) 1 *radial integral along* $(0 - z_{10})$ *defines* C (2.3); (ii) 1 *radial integral along* $(0 - z_1)$ *determines two real equations to fix* w_1 (2.4a); (iii) 2 *radial integrals along* $(0 - z_5)$ *and* $(0 - z_7)$ *determine four real equations to fix* w_5 *and* w_7 (2.4b); (iv) 3 *chordal integrals along* $(z_3 - z_4)$, $(z_4 - z_5)$, *and* $(z_9 - z_{10})$ *determine three real equations to fix* $|w_4 - w_3|$, $|w_5 - w_4|$, *and* $|w_{10} - w_9|$ (2.4c). *TOTAL:* $N - 1 = 9$ *real equations.*

first place. (Still, there may be problems in solving the system (2.4) numerically, for it is now undetermined.) The latter case, $\beta_{N-1} = -1$, is more serious, and must be avoided in the numbering of the vertices w_k.

2.2. Transformation to an unconstrained system. The nonlinear system (2.4) ostensibly involves $N-1$ complex unknown points $z_1, \cdots, z_{N-1}$ on the unit circle. In dealing with such a system, we naturally begin by considering not the points z_k themselves, but their arguments θ_k given by

$$z_k = e^{i\theta_k}, \qquad 0 < \theta_k \leqq 2\pi. \tag{2.5}$$

Now the system depends on $N-1$ real unknowns, and the solution in terms of the θ_k is fully determined.

However, the system (2.4) as it stands must be subject to a set of strict inequality constraints,

$$0 < \theta_k < \theta_{k+1}, \qquad 1 \leqq k \leqq N-1, \tag{2.6}$$

which embody the fact that the vertices z_k must lie in ascending order counterclockwise around the unit circle. To solve the system numerically, it is desirable to eliminate these constraints somehow. We do this by transforming (2.4) to a system in $N-1$ variables $y_1, \cdots, y_{N-1}$, defined by the formula

$$y_k = \log \frac{\theta_k - \theta_{k-1}}{\theta_{k+1} - \theta_k}, \qquad 1 \leqq k \leqq N-1, \tag{2.7}$$

where θ_0 and θ_N, two different names for the argument of $z_N = 1$, are taken for convenience as 0 and 2π, respectively.

We make no attempt to tailor the numerical solution procedure to the particular Schwarz–Christoffel problem under consideration. In particular, all iterations begin with the trivial initial estimate $y_k = 0$ $(1 \leqq k \leqq N-1)$. This corresponds to trial vertices spaced evenly around the unit circle. The following input parameters to NS01A have generally remained fixed: DSTEP $= 10^{-8}$ (step size used to estimate derivatives by finite differences), DMAX $= 10$ (maximum step size), MAXFUN $= 15(N-1)$ (maximum number of iterations).

A fourth parameter, EPS, defines the convergence criterion—how large a function vector (square root of sum of squares of functions values) will be considered to be satisfactorily close to zero. We have most often taken 10^{-8} or 10^{-14} here. The choice of EPS is not very critical, however, as convergence in NS01A is generally quite fast in the later stages.

In the course of this work about two hundred Schwarz–Christoffel transformations have been computed, ranging in complexity from $N = 3$ to $N = 20$. NS01A has converged successfully to an accurate solution in nearly all of these trials. Section 5.1 gives a series of plots showing this convergence graphically for a simple example.

3. Computation of the S–C map and its inverse. Determining the accessory parameters is the most formidable task in computing numerical Schwarz–Christoffel transformations. Once this is done, evaluation of the map and of its inverse follow relatively easily. The foundation of these computations continues to be compound Gauss–Jacobi quadrature.

3.1. From disk to polygon: $w = w(z)$. To evaluate the forward map $w(z)$ for a given point z in the disk or on the circle, we must compute the integral

$$w = w_0 + C\int_{z_0}^{z} \prod_{j=1}^{N} \left(1 - \frac{z'}{z_j}\right)^{-\beta_j} dz' \tag{3.1}$$

with $w_0 = w(z_0)$, where the endpoint z_0 may be any point in the closed disk at which the image $w(z_0)$ is known and not infinite. Three possible choices for z_0 suggest themselves:

(1) $z_0 = 0$; hence $w_0 = w_c$;

(2) $z_0 = z_k$ for some k; hence $w_0 = w_k$, a vertex of P;

(3) $z_0 =$ some other point in the disk at which w has previously been computed.

In cases (1) and (3), neither endpoint has a singularity, and an evaluation of (3.1) by compound Gauss–Jacobi quadrature reduces to the use of compound Gauss quadrature. In case (2) a singularity of the form $(1 - z/z_k)^{-\beta_k}$ is present at one of the endpoints and the other endpoint has no singularity.

The best rule for computing $w(z)$ is: if z is close to a singular point z_k (but not one with $w_k = \infty$), use choice (2); otherwise, use choice (1). In either case we employ compound Gauss–Jacobi quadrature, taking normally the same number of nodes as was used in solving the parameter problem. By this procedure we evaluate $w(z)$ readily to "full" accuracy—that is, the accuracy to which the accessory parameters have been computed, which is directly related to the number of points chosen for Gauss–Jacobi quadrature (see § 4.1). Quadrature nodes and weights need only be computed once, of course.

We should emphasize that even in the vicinity of a singularity z_k, the evaluation of the map $w = w(z)$ is inherently very accurate. This very satisfactory treatment of singular vertices is a considerable attraction of the Schwarz–Christoffel approach for solving problems of Laplace type. In particular, in a potential problem the Schwarz–Christoffel transformation "automatically" handles the singularities correctly at any number of reentrant corners.

3.2. From polygon to disk: $z = z(w)$. For computing the inverse mapping $z = z(w)$, at least two possibilities exist, both of them quite powerful. The most straightforward approach is to view the formula $w(z) = w$ as a nonlinear equation to be solved for z, given some fixed value w. The solution may then be found iteratively by Newton's method or a related device. $w(z)$ should be evaluated at each step of such a process by compound Gauss–Jacobi quadrature along a straight line segment whose initial point remains fixed throughout the iteration.

An alternative approach is to invert the Schwarz–Christoffel formula

$$\frac{dw}{dz} = C \prod_{k=1}^{N} \left(1 - \frac{z}{z_k}\right)^{-\beta_k}$$

to yield the formula

$$\frac{dz}{dw} = \frac{1}{C} \prod_{k=1}^{N} \left(1 - \frac{z}{z_k}\right)^{+\beta_k}. \tag{3.2}$$

This inversion is possible because $w = w(z)$ is a conformal mapping, which means $|dw/dz| > 0$ everywhere. Equation (3.2) may now be thought of as an ordinary differential equation (o.d.e.),

$$\frac{dz}{dw} = g(w, z), \tag{3.3}$$

in one complex variable w. If a pair of values (z_0, w_0) is known and the new value $z = z(w)$ is sought, then z may be computed by applying a numerical o.d.e. solver to the problem (3.3), taking as a path of integration any curve from w_0 to w which lies within the polygon P.

In our program we have chosen to combine these two methods, using the second method to generate an initial estimate for use in the first. We begin with the o.d.e. formulation, using the code ODE by Shampine and Gordon, and for convenience we integrate whenever possible along the straight line segment from w_c to w. (ODE, like most o.d.e. codes, is written for problems in real arithmetic, so that we must first express (3.2) as a system of first-order o.d.e.'s in two real variables.) Since P may not be convex, more than one line segment step may be required to get from w_0 to w in this way. It will not do to take $w_0 = w_k$ for some vertex w_k without special care, because (3.2) is singular at w_k.

From ODE we get a rough estimate $\tilde{z}$ of $z(w)$, accurate to roughly 10^{-2}. This estimate is now used as an initial guess in a Newton iteration to solve the equation $w(z) = w$. This method is faster than the o.d.e. formulation for getting a high-accuracy answer. More important, it is based on the central Gauss–Jacobi quadrature routine unlike the o.d.e. computation.

In summary, we compute the inverse map $z = z(w)$ rapidly to full accuracy by the following steps:

(1) Solve (3.2) to low accuracy with a packaged o.d.e. solver, integrating whenever possible along the line segment from w_c to w; call the result $\tilde{z}$;
(2) Solve the equation $w(z) = w$ for z by Newton's method, using $\tilde{z}$ as an initial guess.

4. Accuracy and speed.

4.1. Accuracy. The central computational step is the evaluation of the Schwarz–Christoffel integral, and the accuracy of this evaluation normally determines the accuracy of the overall computation. As a consequence of the quadrature principle

At each iterative step in the solution of the nonlinear system (2.4), we begin by computing a set of angles $\{\theta_k\}$ and then vertices $\{z_k\}$ from the current trial set $\{y_k\}$. This is easy to do, though not immediate since (2.7) are coupled. In this way the problem is reduced to one of solving an unconstrained nonlinear system of equations in $N-1$ real variables.

2.3. Integration by compound Gauss–Jacobi quadrature. The central computation in solving the parameter problem, and indeed in all Schwarz–Christoffel computations, is the numerical evaluation of the Schwarz–Christoffel integral (1.2) along some path of integration. Typically one or both endpoints of this path are prevertices z_k on the unit circle, and in this case a singularity of the form $(1-z/z_k)^{-\beta_k}$ is present in the integrand at one or both endpoints.

A natural way to compute such integrals quickly is by means of Gauss–Jacobi quadrature (see [2, p. 75]). A Gauss–Jacobi quadrature formula is a sum $\sum_{i=1}^{N_{\text{quad}}} w_i f(x_i)$, where the weights w_i and nodes x_i have been chosen in such a way that the formula computes the integral $\int_{-1}^{+1} f(x)(1-x)^{\alpha}(1+x)^{\beta}\,dx$ exactly for $f(x)$ a polynomial of as high a degree as possible. Thus Gauss–Jacobi quadrature is a generalization of pure Gaussian quadrature to the case where singularities of the general form $(1-x)^{\alpha}(1+x)^{\beta}$ $(a, \beta > -1)$ are present. The required nodes and weights can be computed numerically; we have used the program GAUSSQ by Golub and Welsch [5] for this purpose.

Gauss–Jacobi quadrature appears made-to-order for the Schwarz–Christoffel problem, and at least three previous experimenters have used it or a closely related technique [7], [10], [13]. We began by doing the same, and got good results for many polygons with a small number of vertices. In general, however, we found this method of integration very inaccurate. For a typical sample problem with $N = 12$ and $N_{\text{quad}} = 8$, it produced integrals accurate to only about 10^{-2}, and it does much worse if one chooses polygons designed to be troublesome.

What goes wrong is a matter of resolution. Consider a problem like the one shown in Fig. 4. We wish to compute the integral (1.2) along the segment from z_k to some point p. (In the parameter problem p might be 0 or z_{k-1}; in later computations it might be any point in the disk.) Now direct application of a Gauss–Jacobi formula will involve sampling the integrand at only N_{quad} nodes between z_k and p. If the singularity z_{k+1} is so close to the path of integration that the distance $\varepsilon = |z_{k+1} - z_k|$ is comparable to the distance between nodes, then obviously the Gauss–Jacobi formula will yield a very poor result. It turns out that in Schwarz–Christoffel problems the correct spacing of prevertices z_k around the unit circle is typically very irregular, so the appearance of this problem of resolution is the rule, not the exception. (See examples in § 5.)

To maintain high accuracy without giving up much speed, we have switched to a kind of *compound Gauss–Jacobi quadrature* (see [2, p. 56]). We adopt, somewhat arbitrarily, the following quadrature principle:

> No singularity z_k shall lie closer to an interval of integration than half the length of that interval.

To achieve this goal, our quadrature subroutine must be able to divide an interval of integration into shorter subintervals as necessary, working from the endpoints in. On the short subinterval adjacent to the endpoint, Gauss–Jacobi quadrature will be applied; on the longer interval (or intervals) away from the endpoint, pure Gaussian quadrature will be applied. The effect of this procedure is that number of integrand evaluations required to achieve a given accuracy is reduced from $O(1/\varepsilon)$ to $O(\log_2 1/\varepsilon)$.

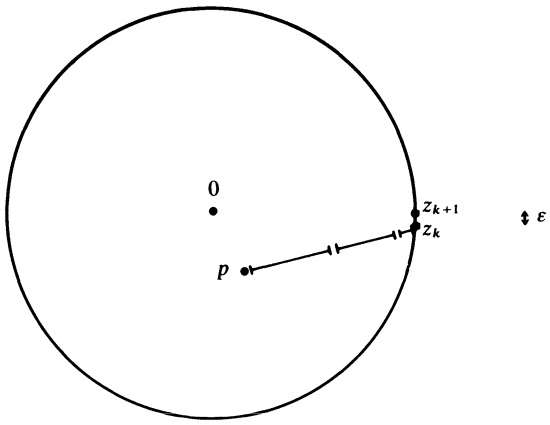


FIG. 4. *Compound Gauss–Jacobi quadrature. Division of an interval of integration into subintervals to maintain desired resolution.*

Figure 4 shows the intervals of integration that come into play in compound Gauss–Jacobi quadrature. For a plot comparing the accuracy of simple and compound Gauss–Jacobi quadrature in another typical problem see § 4.1.

With the use of compound Gauss–Jacobi quadrature, we now achieve high accuracy in little more than the time that direct Gauss–Jacobi quadrature takes. This is possible because only a minority of integrals have a singularity close enough that subdivision of the interval of integration is required. In the 12-vertex example mentioned above, the switch to compound Gauss–Jacobi integration decreased the error from 10^{-2} to $2 \cdot 10^{-7}$.

There remains one circumstance in which integration by compound Gauss–Jacobi quadrature as described here is unsuccessful. This is the case of an integration interval with one endpoint quite near to some prevertex z_k corresponding to a vertex $w_k = \infty$. We cannot evaluate such an integral by considering an interval which begins at z_k, for the integral would then be infinite. The proper approach to this problem is probably the use of integration by parts, which can reduce the singular integrand to one that is not infinite. Depending on the angle β_k, one to three applications of integration by parts will be needed to achieve this. We have not implemented this procedure.

The subtlety of the integration problem in Schwarz–Christoffel computations is worth emphasizing. It is customary to dispatch the integration problem as quickly as possible, in order to concentrate on the "difficult" questions: computation of accessory parameters and inversion of the Schwarz–Christoffel map. We believe, however, that the more primary problem of computing Schwarz–Christoffel integrals—the "forward" problem—should always remain a central concern. Any numerical approach to the parameter problem or the inversion problem is likely to employ an iterative scheme which depends at each step on an evaluation of the integral (1.2), and so the results can only be as accurate as that evaluation.

2.4. Solution of system by packaged solver. The unconstrained nonlinear system is now in place and ready to be solved. For this purpose we employ a library subroutine: NS01A, by M. J. D. Powell [11], which uses a steepest descent search in early iterations if necessary, followed by a variant of Newton's method later on. (The routine does not use analytic derivatives.) It is assumed that a variety of other routines would have served comparably well.

adopted in § 2.3—that no quadrature interval shall be longer than twice the distance to the nearest singularity z_k—the compound Gauss–Jacobi formulation achieves essentially the full accuracy typical of Gaussian quadrature rules operating upon smooth integrands. That is, the number of digits of accuracy is closely proportional to N_{quad}, the number of quadrature nodes per half-interval, with a very satisfactory porportionality constant in practice of approximately 1.

It is important not only to be capable of high accuracy, but to be able to measure how much accuracy one has in fact achieved in a given computation. To do this we employ an accuracy testing subroutine, which is regularly called immediately after the parameter problem is solved. Given a computed set of accessory parameters C and $\{z_k\}$,

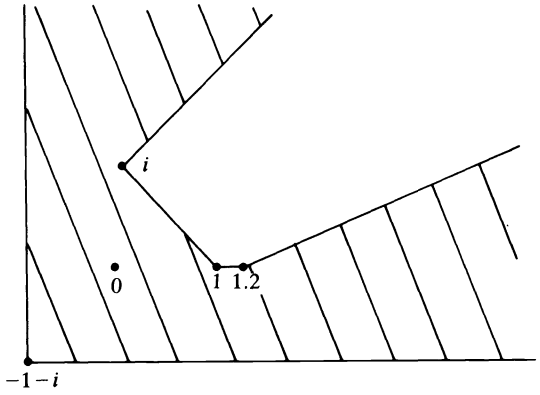


FIG. 5(a)

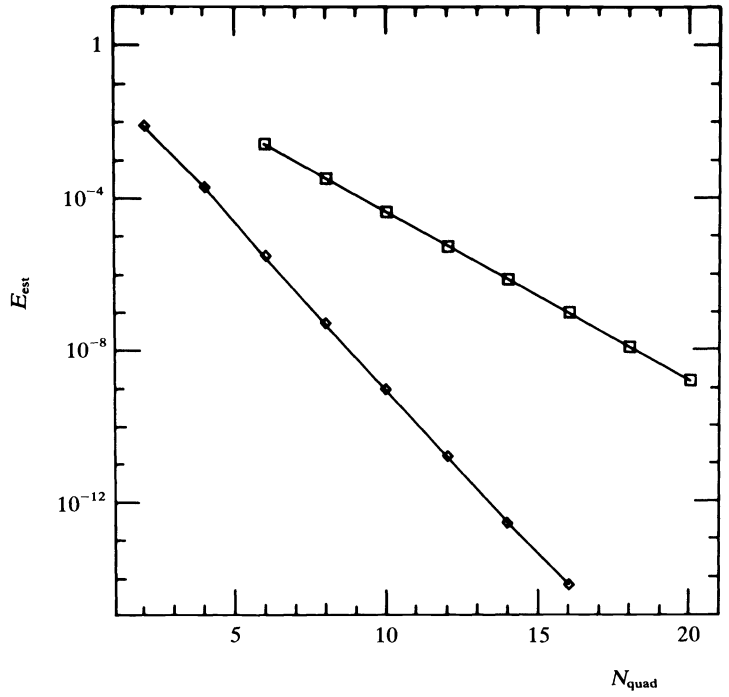


FIG. 5(b). *Quadrature accuracy as a function of number of nodes. The error estimate E_{est} is plotted as a function of N_{quad} for the polygon of Fig.* 5(a). *The upper and lower curves correspond to simple Gauss–Jacobi and compound Gauss–Jacobi quadrature, respectively.*

this subroutine computes the distances $|w_k - w_c|$ for each $w_k \neq \infty$ and the distances $|w_{k-1} - w_{k+1}|$ for each $w_k = \infty$, making use of the standard routine for compound Gauss–Jacobi quadrature. The numbers obtained are compared with the exact distances specified by the geometry of the polygon, and the maximum error, E_{est}, is printed as an indication of the magnitude of errors in the converged solution. It is now probable that subsequent computations of $w(z)$ or $z(w)$ will have errors no greater than roughly E_{est}.

Most often we have chosen to use an 8-point quadrature formula. Since each interval of integration is initially divided in half by the quadrature subroutine, this means in reality at least 16 nodes per integration. With this choice E_{est} consistently has magnitude $\sim 10^{-8}$ for polygons on the scale of unity.

Figure 5b gives an indication of the relationship between the number of quadrature nodes and the error E_{est}; it shows E_{est} as a function of N_{quad} for a 6-gon which is shown in Fig. 5a. Two curves are shown: one for simple Gauss–Jacobi quadrature, and one for compound Gauss–Jacobi quadrature. The exact quantities here should not be taken too seriously; examples could easily have been devised to make the difference in performance of the two quadrature methods much smaller or much greater.

4.2. Speed. Any application of Schwarz–Christoffel transformations consists of a sequence of steps; for convenience we use the names of the corresponding subroutines in our program:

INIT—set up problem
QINIT—compute quadrature nodes and weights
SCSOLV—solve parameter problem
TEST—estimate accuracy of solution
ZSC, WSC, etc.—compute forward and inverse transformations in various applications

Among these tasks INIT, QINIT and TEST all take negligible amounts of time relative to the other computations: typically less than 0.1 secs. on the IBM 370/168 for INIT and QINIT, and for TEST a variable time that is usually less than 5% of the time required by SCSOLV. What remains are three main time consumers: SCSOLV, ZSC, and WSC.

We begin with WSC, which performs the central evaluation of (1.2) by compound Gauss–Jacobi quadrature. This evaluation takes time proportional to N_{quad} (the number of quadrature nodes) and to N (the number of vertices). The first proportionality is obvious, and the second results from the fact that the integrand of (1.2) is an N-fold product. Very roughly, we may estimate

$$\text{time to solve } w = w(z)\text{: } 0.25 \cdot N_{\text{quad}} \cdot N \text{ msec.} \tag{4.1a}$$

for double precision computations on the IBM 370/168. Taking a typical value of $N_{\text{quad}} = 8$, which normally leads to 8-digit accuracy, (4.1a) may be rewritten

$$\text{time to solve } w = w(z)\text{: } 2N \text{ msec.} \tag{4.1b}$$

For the minority of cases in which the interval must be subdivided to maintain the required resolution, these figures will be larger.

To estimate the time required to solve the parameter problem, we combine (4.1) with an estimate of how many integrals must be computed in the course of solving this problem. To begin with, at each iteration about N integrals are required by NS01A (the exact number depends on the number of vertices at infinity). On top of this, it is a fair

estimate to say that $4N$ iterations will be required by NS01A to achieve a high-accuracy solution. We are therefore led to the estimate

(4.2a) time to solve parameter problem: $N_{\text{quad}} \cdot N^3$ msec.

or, taking again $N_{\text{quad}} = 8$,

(4.2b) time to solve parameter problem: $8N^3$ msec.

These estimates correspond fairly well with observed computation times for the parameter problem: two problems with $N = 5$ and $N = 18$ may be expected to take about 1 and 50 seconds, respectively. It is clear that computing a Schwarz–Christoffel transformation becomes quite a sizeable problem for polygons with more than ten vertices. In particular, such computations are too time-consuming for it to be very practical to approximate a curved domain by a polygon with a large number of vertices.

Finally, we must consider the time taken by subroutine ZSC to invert the Schwarz–Christoffel map. This too is proportional to N_{quad}, and quite problem dependent. We estimate very roughly:

(4.3a) time to solve $z = z(w)$: $N_{\text{quad}} \cdot N$ msec.

or, with $N_{\text{quad}} = 8$,

(4.3b) time to solve $z = z(w)$: $8N$ msec.

Note that inverting the Schwarz–Christoffel map is only about four times as time-consuming as computing it in the forward direction.

In practice, computational applications will vary considerably in the use they make of a Schwarz–Christoffel transformation once the parameter problem is solved. If only a few dozen applications of ZSC or WSC are required, then the computational time for solving the parameter problem will dominate. If thousands of such computations are needed, on the other hand, then the parameter problem may become relatively insignificant. The latter situation is most likely to hold when plotting is being done, or when a high-accuracy solution in the model domain is to be computed by means of finite differences.

In summary, high accuracy is cheap in Schwarz–Christoffel transformations; what consumes time is solving problems involving a large number of vertices.

5. Computed examples and applications.

5.1. Iterative process for a single example. Figure 6 shows graphically the process of convergence from the initial estimate in an example involving a 4-gon. Routine NS01A begins by evaluating the function vector (2.4) at the initial guess, then at each of $N-1$ input vectors determined by perturbing the initial guess by the small quantity DSTEP in each component. As a result, the first N pictures always look almost alike, which is why the series shown begins at IT = 4 rather than IT = 1. Each plot shows the current image polygon together with the images of concentric circles in the unit disk (which appear as "contours") and the images of radii leading from the center of the disk to the current prevertices z_k.

These pictures have an elegant bonus feature about them: they may be interpreted as showing not only the image polygon but simultaneously the domain disk, including the prevertices z_k along the unit circle. To see this, look at one of the inner "contour" curves, one which is apparently circular, and the radii within it. Since $w = w(z)$ is a conformal map within the interior of the disk, the radii visible in this circle must intersect at the same angles as their preimages in the domain disk. Thus the inner part of any one of these image plots is a faithful representation on a small scale of the circular

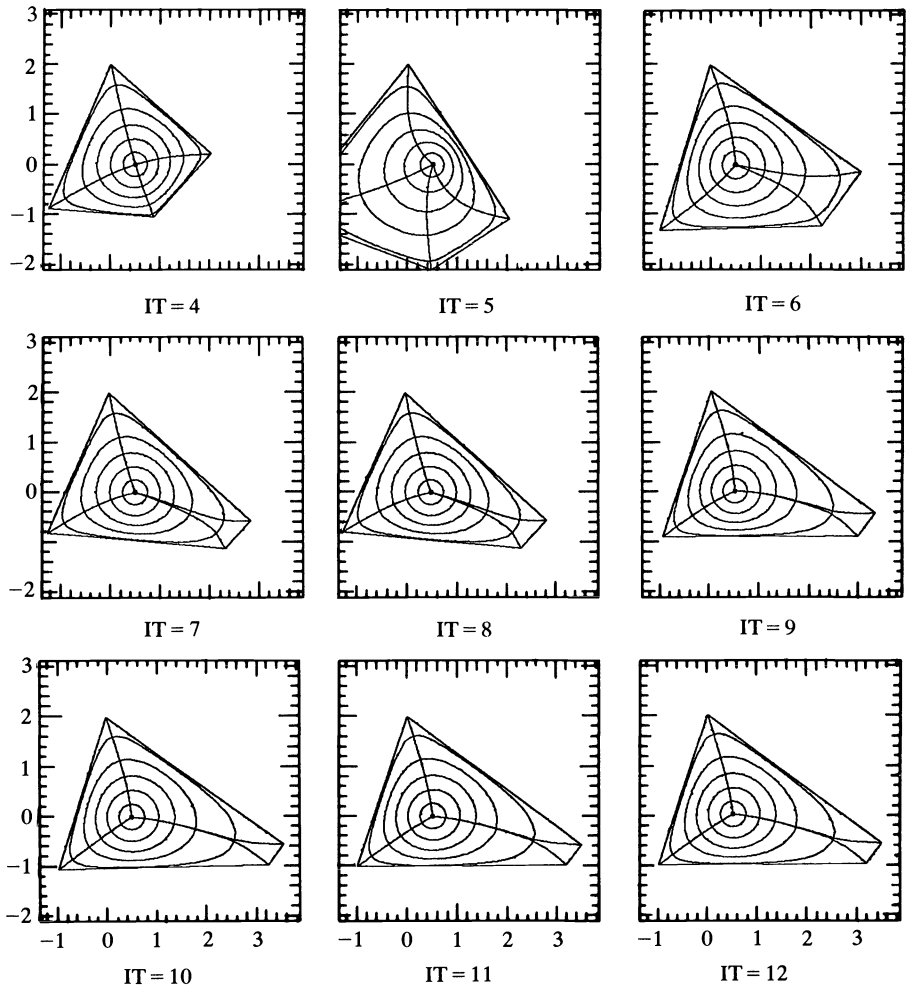


FIG. 6. *Convergence to a solution of the parameter problem. Plots show the current image polygon at each step as the accessory parameters* $\{z_k\}$ *and* C *are determined iteratively for a problem with* $N = 4$.

domain. We see in Fig. 6 that the prevertices are equally spaced around the unit circle initially (IT = 4), but move rapidly to a very uneven distribution. This behavior, which is typical, indicates why the use of a compound form of Gauss–Jacobi quadrature is so important (see § 2.3).

The sum-of-squares error in solving the nonlinear system is plotted as a function of iteration number in Fig. 7 for the same 4-vertex example. Convergence is more or less quadratic, as one would expect for Newton's method. The irregularity at iteration 19 is caused by the finite difference step size of 10^{-8} used to estimate derivatives, and would have been repeated at each alternate step thereafter if the iteration had not terminated.

5.2. Sample Schwarz–Christoffel maps. Figures 8 and 9 show plots of computed Schwarz–Christoffel maps for representative problems. The polygons of Fig. 8 are bounded and those of Fig. 9 are unbounded. Observe that contour lines bend tightly around reentrant corners, revealing the large gradients there, while avoiding the backwater regions near outward-directed corners and vertices at infinity. Like the plots

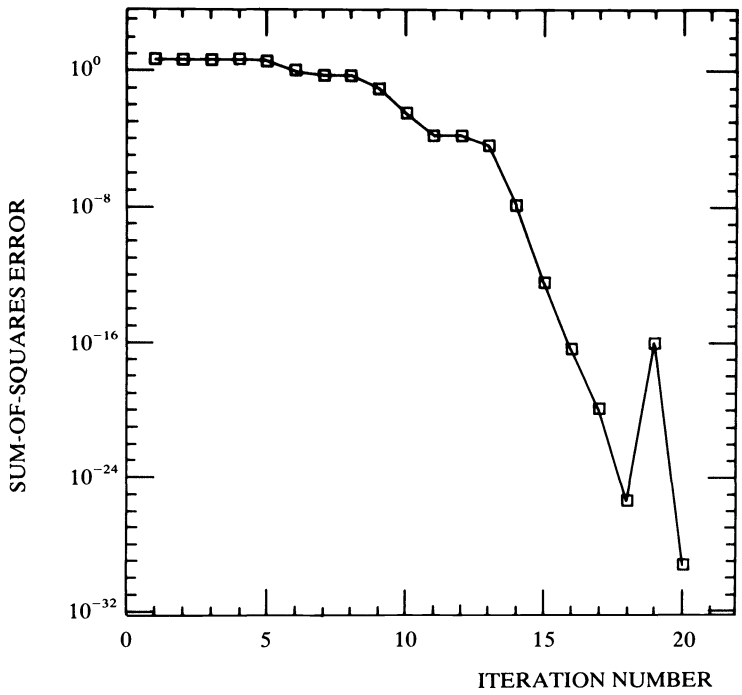


FIG. 7. *Rate of convergence. Sum-of-squares error in the nonlinear system* (2.4) *as a function of iteration number for the same problem as in Fig.* 6.

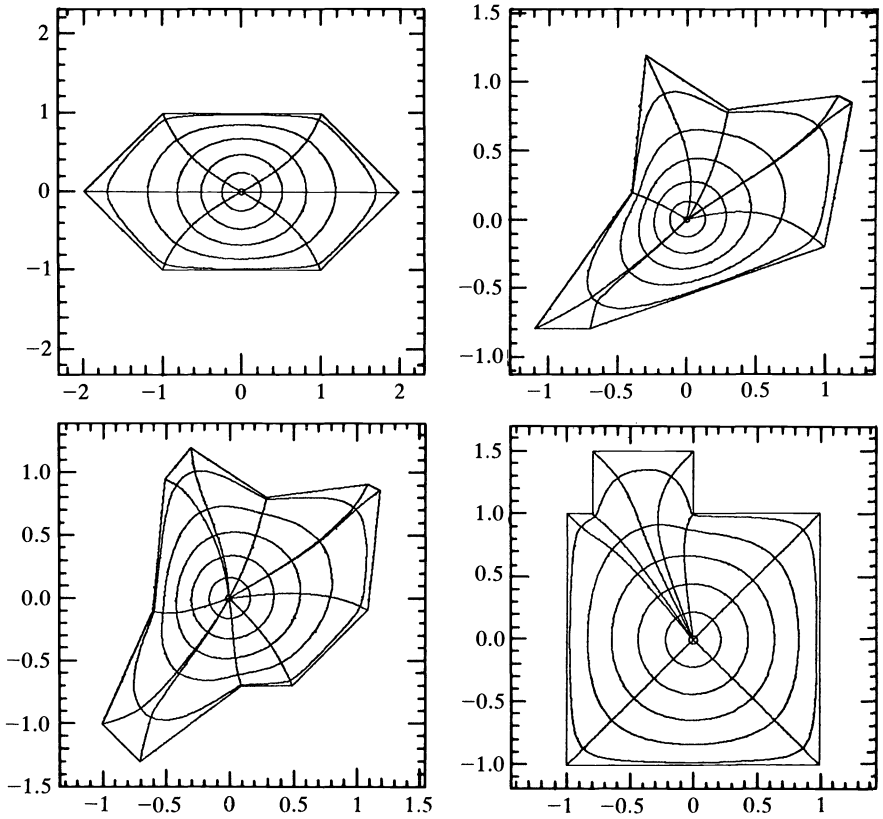


FIG. 8. *Sample Schwarz–Christoffel transformations (bounded polygons). Contours within the polygons are images of concentric circles at radii* .03, .2, .4, .6, .8, .97 *in the unit disk, and of radii from the center of the disk to the prevertices* z_k.

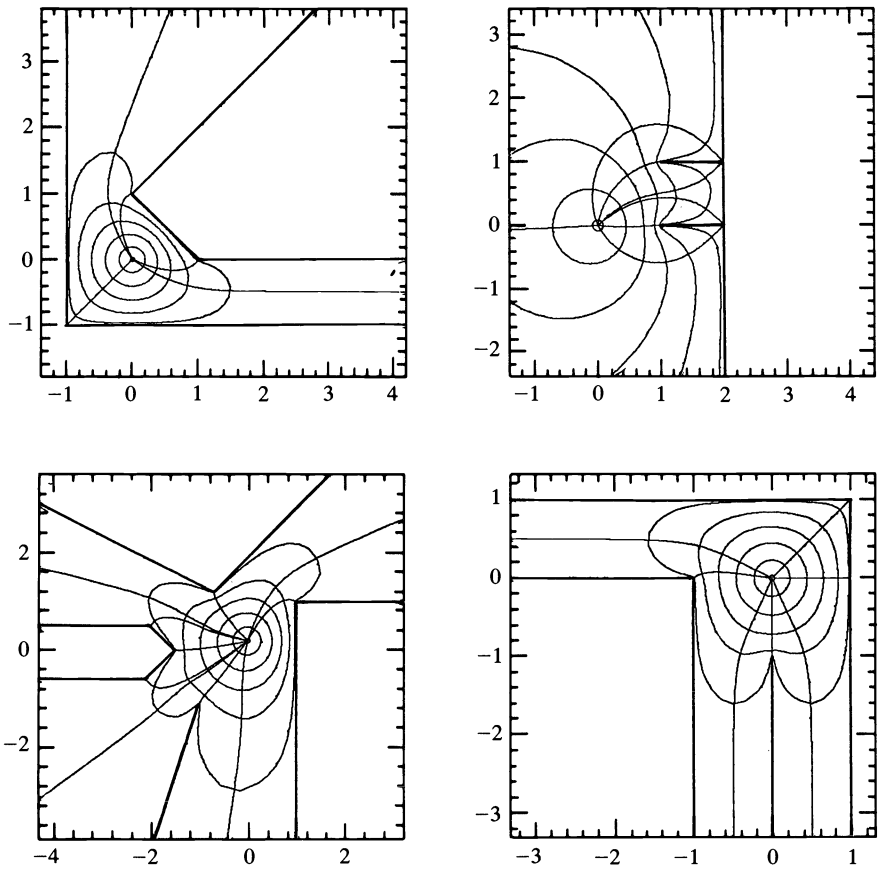


FIG. 9. *Sample Schwarz–Christoffel transformations (unbounded polygons). Contours are as in Fig. 8.*

of Fig. 6, these may be viewed as showing simultaneously the image polygon and the domain disk.

Figure 10 shows similar plots in which streamlines rather than contour lines have been plotted, so that the configuration may be thought of as portraying ideal irrotational fluid flow through a two-dimensional channel. To plot these streamlines an analytic transformation of the disk to an infinite channel with straight parallel sides was used in conjunction with the Schwarz–Christoffel transformation from the disk to the problem domain.

5.3. Discussion of applications. The usefulness of conformal mapping for applied problems stems from the fact that the Laplacian operator transforms in a simple way under a conformal map. Let $f: \mathbb{C} \to \mathbb{C}$ map a region Ω_z in the z-plane conformally onto a region Ω_w in the w-plane, and let Δ_z and Δ_w denote the Laplacian operators $\partial^2/\partial x^2 + \partial^2/\partial y^2$ and $\partial^2/\partial u^2 + \partial^2/\partial v^2$, respectively, where $z = x + iy$ and $w = u + iv$. Then we may easily show

$$\Delta_z \phi(z) = |f'(z)|^2 \Delta_w \phi(f^{-1}(w)) \tag{5.1}$$

for $\phi: \Omega_z \to \mathbb{R}$ suitably differentiable. A conformal map has $|f'(z)| > 0$ everywhere; thus from (5.1) it follows that if $\phi(z)$ is the solution to the Laplace equation $\Delta_z \phi = 0$ in Ω_z, subject to Dirichlet boundary conditions $\phi(z) = g(z)$ on the boundary Γ_z, then $\psi(w) = \phi(f^{-1}(w))$ is a solution to the Laplace equation $\Delta_w \psi = 0$ in the image region $\Omega_w = f(\Omega_z)$,

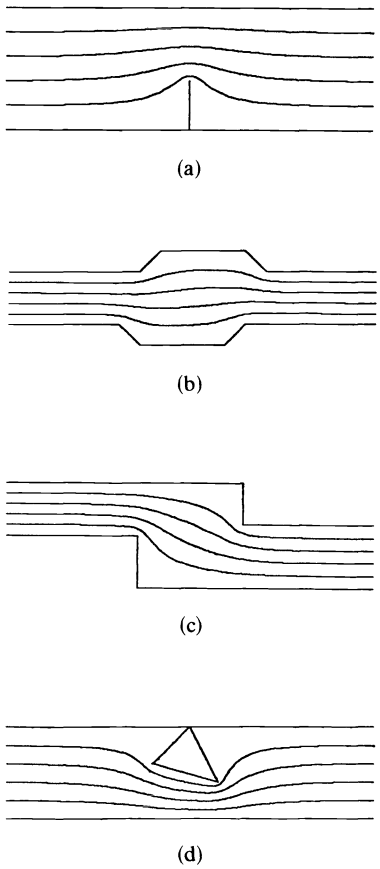


FIG. 10. *Sample Schwarz–Christoffel transformations. Contours show streamlines for ideal irrotational, incompressible fluid flow within each channel.*

subject to the image boundary conditions $\psi(w) = g(f^{-1}(w))$ on the boundary $\Gamma_w = f(\Gamma_z)$. (We have assumed that f maps Γ_z bijectively onto the boundary of Ω_w. This is not always true, but it is true if both regions are bounded by Jordan curves. See [8, Thm. 5.10e].)

More generally, from (5.1) we can see that Poisson's equation, $\Delta_z \phi(z) = \rho(z)$, transforms under a conformal transformation into a Poisson equation in the w-plane with altered right-hand side:

$$\Delta_w \psi(w) = |f'(f^{-1}(w))|^{-2} \rho(f^{-1}(w)). \tag{5.2}$$

Furthermore, more general boundary conditions than Dirichlet also transform in a simple way. For example, the Neumann condition $(\partial/\partial n_z)\phi(z) = h(z)$, where $\partial/\partial n_z$ is a normal derivative in the z-plane, transforms to $(\partial/\partial n_w)\psi(w) = |f'(f^{-1}(w))|^{-1} h(f^{-1}(w))$. We do not pursue such possibilities further here; for a systematic treatment see Chapter VI of [9].

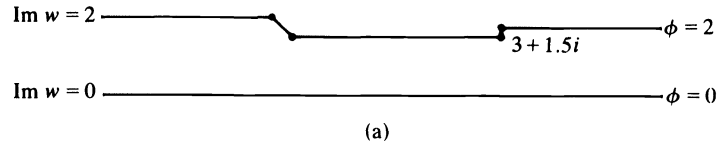
Traditionally, conformal mapping has been applied most often in two areas. One is plane electrostatics, where the electrostatic potential φ satisfies Laplace's equation. The other is irrotational, incompressible fluid flow in the plane, which may be described in terms of a velocity potential φ that also satisfies Laplace's equation. We will outline some ways in which a known conformal map might be used in such application.

Conformal maps do not solve problems, but they may reduce hard problems to easier ones. How much work must be done to solve the easier problem will vary considerably with the application.

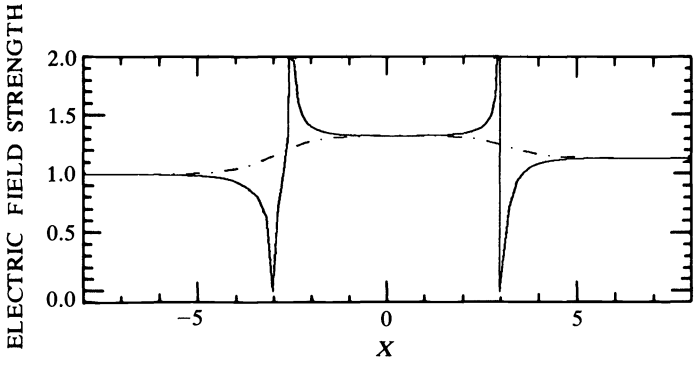
(1) In the best of circumstances, the original problem may be reduced to a model problem whose solution is known exactly. This is the case in the fluid flow problems of Fig. 10, in which a crooked channel may be mapped to an infinite straight channel of constant width.

(2) If a problem of Laplace's equation with pure Dirichlet or Neumann boundary conditions can be mapped conformally to a disk, then Poisson's formula or Dini's formula (see [9]) provide integral representations of the solution at each interior point. Such integrals may be evaluated readily on the computer to yield high accuracy solutions. The primary disadvantage of this approach is that a new integral must be evaluated for each point at which the solution is desired.

(3) If the solution will be required at many points in the domain, then it is probably more efficient to solve Laplace's equation by a trigonometric expansion of the form $b_0+\sum_{k=1}^{m} r^k(a_k \sin k\theta + b_k \cos k\theta)$; coefficients a_k and b_k are selected so as to fit the boundary conditions closely. A disadvantage of this method is that convergence of the expansion may be slow if the boundary conditions are not smooth.



(a)



(b)

w	ϕ	$\lvert E\rvert$	$\arg E/\pi$
$3.1 + 1.4i$	1.7564	1.3082	−.3823
$3.01 + 1.49i$	1.9486	2.4403	−.2833
$3.001 + 1.499i$	1.9889	5.2137	−.2572
$3.000 + 1.500i$	2.0000	∞	−.2500

(c)

FIG. 11. *Laplace equation example: electric potential and field between two infinite sheets.*

(a) *Problem domain: region between two conducting sheets.*

(b) *Field strength along the top boundary (solid line) and bottom boundary (broken line).*

(c) *Computed potential and field strength at three points near* $3+1.5i$.

(4) Finally, if simpler methods fail, a solution in the model domain may be found by a finite-difference or finite-element technique. For problems of Poisson's equation or more complicated equations this will probably normally be necessary.

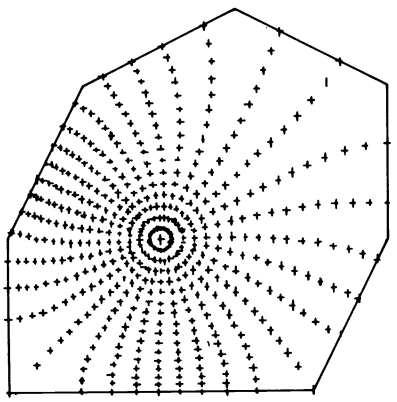
5.4. Laplace's equation. Figure 11 presents an example of type (1) as described in the last section. We are given an infinite region bounded by one straight boundary fixed at potential $\varphi = 0$ and one jagged boundary fixed at $\varphi = 2$. We may think of this as an electrostatics problem. The central question to be answered computationally will be: what are the voltage φ and the electric field $E = -\nabla\varphi$ at a given point, either within the field or on the boundary?

We proceed by mapping the given region onto the disk by a Schwarz–Christoffel transformation, then analytically onto an infinite straight channel (as in the examples of Fig. 10). In the straight channel φ and E are known trivially, and this information may be transferred to the problem domain through a knowledge of the conformal map that connects them and of its (complex) derivative. We omit the details, which are straightforward.

Figure 11(b) shows $|E|$ as a function of x on the upper and lower boundaries of the region. To see more of the behavior of the solution field near a reentrant corner, we also compute the field at three points near $3+1.5i$. These results are given in Fig. 11(c).

5.5. Poisson's equation. Consider the 7-sided region shown in Fig. 12(a). We wish to solve Poisson's equation

$$\Delta\phi(x, y) = \tfrac{1}{5}\sin 2x(1-2(y+1)^2)$$



(a)

Grid ($r\times\theta$)	Transformation and setup time	Fast Poisson solver time	Max. error	RMS error
4×8	1.3 secs.	<.01 secs.	0.132	0.0309
8×16	2 secs.	.01 secs.	0.055	0.0085
16×32	5 secs.	.03 secs.	0.031	0.0037
32×64	16 secs.	.15 secs.	0.026	0.0012

(b)

FIG. 12. *Poisson equation example. Problem is transplanted conformally to the unit disk and solved by finite differences.*

(a) *7-sided problem domain, including image of* 16×32 *finite-difference grid in the unit disk.*

(b) *Computed results for four different grids. Time estimates are for an IBM* 370/168.

on this region, subject to Dirichlet conditions

$$\phi(x, y) = \rho(x, y) = \tfrac{1}{10} \sin 2x(y+1)^2$$

on the boundary. We proceed by mapping the domain to the disk and solving a transformed problem in the disk in polar coordinates by means of a second-order fast finite difference solver (PWSPLR, by P. Swarztrauber and R. Sweet). $\rho(x, y)$ is the correct solution in the interior as well as on the boundary, so we can determine the accuracy of the numerical solution.

This is not as satisfactory a procedure as was available for Laplace equation problems. According to (5.2), the model problem here is Poisson's equation in the disk with an altered right-hand side containing the factor $|f'(z)|^2$, where f is the composite map from the disk to the 7-gon. Two difficulties arise. The first is that to set up the transformed equation in the disk, $\rho(w_{ij})$ must be computed for every $w_{ij} = w(z_{ij})$ which is an image of a grid point in the disk. This is time consuming, one hundred times more so in this experiment than the fast solution of Poisson's equation once it is set up. Second, $|f'(z)|^2$ is singular (unbounded, in this example) at each prevertex z_k, and this appears to interfere with the second-order accuracy which we would like to observe. The table in Fig. 12(b) attests to both of these problems.

5.6. Eigenfrequencies of the Laplace operator. Petter Bjørstad (Computer Science Dept., Stanford University) has recently combined the present Schwarz–Christoffel computation with a fast finite-difference scheme to successfully compute eigenvalues and eigenvectors of the Laplacian operator on polygonal regions. These results may be interpreted as giving the normal modes and frequencies of a thin membrane in two dimensions, or of a three-dimensional waveguide with constant cross-section. This work will be reported elsewhere.

6. Conclusions. A program has been described which computes accurate Schwarz–Christoffel transformations from the unit disk to the interior of a simply connected polygon in the complex plane, which may be unbounded. Key features of the computation have been:

(1) choice of the unit disk rather than the upper half-plane as the model domain, for better numerical scaling (§ 2.1);
(2) use of complex contour integrals interior to the model domain rather than along the boundary, making possible the treatment of unbounded polygons (§ 2.1);
(3) use of compound Gauss–Jacobi quadrature in complex arithmetic to evaluate the Schwarz–Christoffel integral accurately (§§ 2.3, 3.1);
(4) formulation of the parameter problem as a constrained nonlinear system in $N-1$ variables (§ 2.1);
(5) elimination of constraints in the nonlinear system by a simple change of variables (§ 2.2);
(6) solution of the system by a packaged nonlinear systems solver; no initial estimate required in practice (§ 2.4);
(7) computation of a reliable estimate of the accuracy of further computations, once the parameter problem has been solved (§ 4.1);
(8) accurate evaluation of the inverse mapping in two steps by means of a packaged o.d.e. solver and Newton's method (§ 3.2).

Previous efforts at computing Schwarz–Christoffel transformations numerically include [1], [6], [7], [10], and [13]. The present work differs from these in that it deals directly with complex arithmetic throughout, taking the unit disk rather than the upper

half-plane as the model domain and evaluating complex contour integrals. This makes possible the computation of transformations involving general unbounded polygons. (Cherednichenko and Zhelankina [1] also treat unbounded polygons, by a different method.) Two other important differences are the use of compound Gauss–Jacobi quadrature, and the application of a change of variables to eliminate constraints in the nonlinear system ((5), above). We believe that our program computes Schwarz–Christoffel transformations faster, more accurately, and for a wider range of problems tha previous attempts.

A variety of directions for further work suggest themselves. Here are some of them.

(1) More attention should be paid to the problems of evaluating the forward and inverse S–C maps once the parameter problem has been solved. The two-step method for the inverse map described in § 3.2 is reliable, but it uses too much machinery. Recently Petter Bjørstad and Eric Grosse of Stanford University have replaced (3.1) with a power series expansion for problems in which all the nodes of a finite-difference grid must be mapped from one domain to the other, thereby speeding up the evaluations of $w(z)$ and $z(w)$ by an order of magnitude. This kind of addition is very important for applications.

(2) The program could easily be extended to construct maps onto the exterior of a polygon—that is, the interior of a polygon whose interior includes the point at infinity. This extension would be necessary, say, for applications to airfoil problems.

(3) It should not be too great a step to raise the present program to the level of "software" by packaging it flexibly, portably, and robustly enough that naive users could apply it easily to physical problems. Conformal mapping is currently far behind many other areas of numerical mathematics in the development and distribution of software.

(4) The program might be extended to handle the rounding of corners in Schwarz–Christoffel transformations (see [8]). What about mapping doubly or multiply connected polygonal regions, perhaps by means of an iterative technique which computes an S–C transformation at each step?

(5) More generally, the Schwarz–Christoffel formula should be viewed in context as a particularly simple method in conformal mapping which is applicable only to a limited set of geometries. Direct comparisons with programs that can treat curved boundaries, especially those based on integral equations, would be informative. In some applications the S–C transformation might profitably be used as part of a larger program. In fact, the S–C formula (1.2) itself has a natural generalization to the case of curved boundaries, which may be obtained formally by allowing an infinite number of vertices with infinitesimal external angles. R. T. Davis [3] has implemented this formula numerically with very promising results.

Most important, further work is needed in the direction of applications to Laplace's equation, Poisson's equation, and related problems. Irregular or unbounded domains are generally troublesome to deal with by standard techniques, particularly when singularities in the form of reentrant corners are present. Schwarz–Christoffel transformations offer a means of getting around such difficulties in a natural way. More experience is needed here.

Note. This work is described in more detail in [12], and a program listing is given there. An experimental copy of the package with documentation and sample driver programs may be obtained from the author.

Acknowledgments. This work was suggested and guided by Peter Henrici, without whom it would not have been possible. Computations were performed at the Stanford Linear Accelerator Center of the U.S..Department of Energy. The author has benefited from discussions with Petter Bjørstad, William M. Coughran, Jr., Gene Golub, Eric Grosse, Randy LeVeque, and Cleve Moler.

REFERENCES

[1] L. A. CHEREDNICHENKO AND I. K. ZHELANKINA, *Determination of the constants that occur in the Christoffel–Schwarz integral.* Izv. Vyssh. Ucebn. Zaved. Elektromehanika, 10 (1975), pp. 1037–1040. (In Russian.)

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

[3] R. T. DAVIS, *Numerical methods for coordinate generation based on Schwarz–Christoffel transformations*, 4th AIAA Computational Fluid Dynamics Conference Proceedings, Williamsburg, VA, 1979.

[4] D. GAIER, *Konstruktive Methoden der konformen Abbildung*, Springer-Verlag, Berlin, 1964.

[5] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gaussian Quadrature Rules*, Math. Comp., 23 (1969), pp. 221–230.

[6] T. R. HOPKINS AND D. E. ROBERTS, *Kufarev's Method for the Numerical Determination of the Schwartz–Christoffel Parameters*, University of Kent, Kent, England, 1978.

[7] D. HOWE, *The application of numerical methods to the conformal transformation of polygonal boundaries*, J. Inst. Math. Appl., 12 (1973), pp. 125–136.

[8] P. HENRICI, *Applied and Computational Complex Analysis I*, Wiley, New York, 1974.

[9] L. V. KANTOROVICH AND V. I. KRYLOV, *Approximate Methods of Higher Analysis*, P. Noordhoff, Groningen, the Netherlands, 1958.

[10] E.-S. MEYER, *Praktische Verfahren zur konformen Abbildung von Geradenpolygonen.* Dissertation Universität Hannover, 1979.

[11] M. J. D. POWELL, *A Fortran subroutine for solving systems of non-linear algebraic equations*, Tech. Rep. AERE-R. 5947, Harwell, England, 1968.

[12] L. N. TREFETHEN, *Numerical computation of the Schwarz–Christoffel transformation*, Computer Science Dept. Tech. Rep. STAN-CS-79-710, Stanford Univ., Stanford, CA, 1979.

[13] V. V. VECHESLAVOV AND V. I. KOKOULIN, *Determination of the parameters of the conformal mapping of simply connected polygonal regions*, U.S.S.R. Computational Math. and Math. Phys., 13 (1974), pp. 57–65.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0005 $01.00/0

IMPLEMENTATION OF IMPLICIT FORMULAS FOR THE SOLUTION OF ODEs*

L. F. SHAMPINE†

Abstract. Implicit formulas are quite popular for the solution of ODEs. They seem to be necessary for the solution of stiff problems. Every code based on an implicit formula must deal with certain tasks studied in this paper: (i) A choice of basic variable has to be made. The literature is confusing as to what the possibilities are and the consequences of the choice. These matters are clarified. (ii) A test for convergence of the method for solving the implicit equations must be made. Ways of improving the reliability of this test are studied. (iii) Deciding when to form a new approximate Jacobian and/or iteration matrix is a crucial issue for the efficiency of a code. New insight which suggests rather specific actions will be developed. The paper closes with an interesting numerical example.

Key words. implicit formulas, stiff, convergence of iteration, ordinary differential equations

1. Introduction. Implicit formulas for the solution of the initial value problem for a system of ordinary differential equations (ODEs) are quite popular. They seem to be necessary for the solution of stiff problems. To be efficient enough to be practical, a formula for this purpose "must" have an infinite stability region R. Lambert puts it well in [1, p. 484] where he says, "Although no precise result concerning all possible classes of methods exists (naturally!) it is certainly true that for all commonly used methods, explicitness is incompatible with infinite R."

In this paper we examine various practical aspects of the implementation of implicit formulas. We have examined many codes based on such formulas. The documentation available to us has often given scant attention to the details of the algorithms we examine. When possible we have studied the codes because they are not always what is described. It is extremely difficult to assimilate the many ad hoc devices put in complex codes to achieve a given purpose, especially when documentation and/or codes are in foreign languages (natural and computer). We hope that we are not mistaken in our statements about specific codes, and express our regrets in advance for any slips. Because statements about codes are made only to illustrate points and to show that we are taking up real issues, mistakes of this kind do not invalidate our work.

In § 2 we attempt to clarify the confusing situation with respect to the choice of basic variable in a code. We also discuss the consequences of the possible choices. In the next section we examine the solution of the algebraic equations of an implicit method. Practically every code we have examined implements at least one good idea, but each seems to us to offer room for considerable improvement of its reliability. In the following section we gain new insight as to the effects of changing step size and approximate Jacobian. The ad hoc devices being used to decide when to form a new Jacobian or iteration matrix can be made more specific and put on a sounder basis with this insight. We close by presenting an example in § 5. It shows, in very reasonable circumstances, the difficulty of the task we address and, in particular, the dangers of a popular approach to solution of this task.

2. Formulation of the problem. There is quite a bit of variation possible in the way the implicit algebraic equations for advancing a step are formulated and solved. As a consequence the literature is extremely confusing. Our purpose in this section is to

* Received by the editors May 23, 1979, and in revised form October 15, 1979.
† Sandia Laboratories, Albuquerque, New Mexico 87185.

clarify the decisions involved. Some references to the literature will be made to show that we are discussing possibilities represented in production codes, but we shall make no attempt to be complete.

For notational convenience we shall suppress the independent variable in all our expressions and we shall not explicitly indicate vectors. It is convenient to introduce as a variable the derivative scaled by the step size h, $z = hy'(x)$. With these conventions the (vector) differential equation itself is

$$z = hf(y). \tag{1}$$

The implicit equations for advancing a step have the generic form

$$y = \gamma z + \Psi, \qquad \gamma > 0. \tag{2}$$

The backward differentiation formulas (BDF) are our prototype, but this form, or modest extensions, applies to a great many possibilities.

Ostensibly we seek a solution y^* of (2) at each step. However, for purposes of predicting an approximation y^0 to y^*, error estimation, interpolation, and the like, most codes also use the scaled derivative $z^* = hf(y^*)$. It is possible to take either y or z as the basic variable. Both choices are common. If y is the basic variable, we solve

$$y = h\gamma f(y) + \Psi \tag{3}$$

and define z by (1). If z is the basic variable, we solve

$$z = hf(\gamma z + \Psi) \tag{4}$$

and define y by (2).

We find no important reason for preferring one choice of basic variable to the other. In particular, the solution of (3) and (4) is virtually identical. All the common schemes for solving (3) can be described as linearizing f with an approximate Jacobian J and approximating $h\gamma$ by $h'\gamma'$ to yield

$$y^{m+1} = \Psi + h\gamma f(y^m) + h'\gamma' J(y^{m+1} - y^m). \tag{5}$$

It looks a bit odd to approximate $h\gamma$ when no approximation is necessary, but there are practical reasons for doing so, that we take up in § 4. The scheme (5) requires the solution of a system of linear equations with matrix $I - h'\gamma' J$ for each iteration. The choice $J = 0$ corresponds to the simple, or functional, iteration typical of codes for nonstiff problems. For stiff problems one needs a nontrivial approximation to the Jacobian matrix,

$$J \doteq \left(\frac{\partial f_i}{\partial y_j}(y^*)\right) = \mathscr{J}(y^*).$$

(Here we write the ith component of f as $f_i(y_1, \cdots, y_n)$). We shall have a lot more to say about (5). For now we just need to observe that the same linearization of f in (4) and approximation of $h'\gamma'$ leads to

$$z^{m+1} = hf(\gamma z^m + \Psi) + h'J\gamma'(z^{m+1} - z^m). \tag{6}$$

The iteration matrix is again $I - h'\gamma' J$, although it arises in a slightly different way. Comparison of (5) and (6) shows that they are essentially the same computation. In fact, if the predicted values satisfy $y^0 = \gamma z^0 + \Psi$, then all iterates satisfy $y^{m+1} = \gamma z^{m+1} + \Psi$ so that the computations are equivalent.

The difficulty with getting *both* y^* and z^* does not reveal itself until we realize that we do not compute these quantities exactly. If y is the basic variable, we shall accept

some approximation y^m. What do we take for the corresponding approximation z^m to z^*? Two possibilities spring to mind. One is to define z^m by (1), which is equivalent to saying that the pair (y^m, z^m) satisfies the differential equation exactly. The other is to define z^m by (2), which is equivalent to saying that the pair (y^m, z^m) satisfies the formula exactly. If z is the basic variable, it appears that the only reasonable way to define a corresponding approximation to y^* is by satisfying the formula (2) exactly. This is true, but for a number of formulas it is natural subsequently to define a new approximation for z^* by satisfying the equation (1). Thus essentially the same two possibilities arise with either choice of basic variable. Whether y or z is the basic variable does not matter. How the corresponding variable is obtained can be crucial. Before taking up this issue we shall connect up these questions with some codes and point out an alternative which is not well known.

The issue can be avoided if one chooses y as the basic variable and makes no use *at all* of z in his code. Krogh [2] does this with the BDF, and Klopfenstein [3] follows him in this with the generalized formulas he presents. Apparently unaware of these somewhat obscure references, Robertson and Williams [4] also suggest doing this. This way of proceeding has the appeal of simplicity.

It has been customary when solving nonstiff problems to take y as the basic variable and to solve (3) with simple iteration ((5) with $J = 0$). In this context such processes are described with a kind of shorthand. P is used to indicate the prediction y^0, E the evaluation of $f(y^m)$, and C the correction of y^m to get y^{m+1} from (5). The processes used have the form $P(EC)^kE$ or $P(EC)^kEC$. Ending with an evaluation defines z^m by (1) so as to satisfy the differential equation exactly. Ending with a correction y^{m+1} takes z^{m+1} as the last evaluation made, $z^{m+1} = hf(y^m)$, and so corresponds to satisfying the formula exactly. Both possibilities are seen in a number of codes.

Robertson and Williams [4, p. 31] and Robertson [5] take y as the basic variable for solving stiff problems and define z by satisfying the formula exactly. As best we can tell from [6], Brayton et al. do this with the BDF. The other choice is also made when solving stiff problems. Williams and de Hoog [7], Alt [8] and Klopfenstein and Davis [9] take y as the basic variable and define z by satisfying the equation exactly. All these codes are based on different methods.

Choosing z as the basic variable is quite natural when working with implicit Runge–Kutta formulas. They have not often been applied to nonstiff problems, but there are some examples [10], [11]. In this context it is often natural to define y^m from the formula and then define a new z^m so as to satisfy the differential equation. Norsett [12] does this with his code which is intended for stiff problems. More specifically, a derivative value $hf(y^m)$ is formed for the prediction of a solution at the next step and this, of course, corresponds to satisfying the equation exactly for at least this one value. The value is *not* used for error estimation. We provide these details because they show that his code is weakly affected by the inaccuracy of $hf(y^m)$.

The seminal code DIFSUB [13] of C. W. Gear uses z for both the Adams and BDF. The former are intended for nonstiff problems and the latter for stiff. There have been many codes built upon the structure of DIFSUB by Bickart, Byrne, Hindmarsh, Skeel, . . . which retain the choice of z as basic variable. All define the corresponding y by satisfying the formula.

All the possibilities are represented, but no widely accepted code for stiff problems defines the corresponding variable by the differential equation. Let us now see why. We continue to take y as the basic variable. Given y^m, suppose we define $z^m = hf(y^m)$. Then

$$z^m - z^* = hf(y^m) - hf(y^*) = h\mathcal{J}_m(y^m - y^*) \tag{7}$$

on using a mean value theorem. Here $\mathcal{J}_m$ is the Jacobian of f with entries evaluated at different points along the line between y^m and y^*. If on the other hand we define z^m so as to satisfy the formula, we have

$$z^m = (y^m - \Psi)/\gamma,$$

hence

$$z^m - z^* = \frac{1}{\gamma}(y^m - y^*). \tag{8}$$

A stiff problem has $\|h\mathcal{J}_m\| \gg 1$. From (7) and (8) we see that forming a scaled derivative by evaluation of the equation can be disastrously inaccurate for a stiff problem. Besides this crucial difference, it is more expensive because it requires an additional evaluation of f. Robertson and Williams [4, p. 32] report the increased efficiency observed in two codes when they were changed to use the accurate alternative.

For us the choices are clear when solving stiff differential equations. One should do one of the following:

(i) Compute y from (3) and make no use of z;
(ii) Compute y from (3) and then z from (2);
(iii) Compute z from (4) and then y from (2).

The choices are not so clear when solving nonstiff equations. Then h is typically of a size such that neither (7) nor (8) shows a clear advantage with respect to accuracy. Satisfying the differential equation exactly costs an extra function evaluation. This cost may be more than compensated by improved absolute stability.

In the remainder of this paper we take y as the basic variable and presume that z, if needed, is obtained in an appropriate way.

3. Convergence of the iteration. We are concerned with the solution of

$$y = h\gamma f(y) + \Psi \tag{9}$$

by the iterative scheme

$$y^{m+1} = \Psi + h\gamma f(y^m) + h'\gamma' J(y^{m+1} - y^m). \tag{10}$$

In the next section we shall examine the roles of h and J in some detail. For our purposes here it will be convenient to rewrite (10) as

$$y^{m+1} = G(y^m) = (I - h'\gamma' J)^{-1}[\Psi + h\gamma f(y^m) - h'\gamma' J y^m]. \tag{11}$$

A standard convergence result like [14, p. 300] says roughly that if y^0 is close enough to satisfying $y = G(y)$ and $h\gamma$ and $h'\gamma'$ are small enough, G is a contraction operator and the iteration converges to a solution y^* unique in a ball about y^0. If J is close enough to $G'(y^*)$, less stringent demands are placed on $h\gamma$, $h'\gamma'$. In any case, the convergence is linear.

It seems obvious that one wants a good approximation to y^*, but the documentation for many, if not most, codes refer only to making the difference

$$d_m = \|y^{m+1} - y^m\|$$

small. The theory of convergence of the numerical solution of differential equations is based on accepting a y^{m+1} which is either close to y^* or which has a small residual, $y^{m+1} - G(y^{m+1})$. Later we shall sort out the situation.

There are excellent reasons for insisting that G contract on a ball containing the predicted solution y^0. In general (9) has multiple solutions. The predicted solution y^0 is

made from past behavior of the solution of the differential equation, so it must be our guide as to which is the appropriate choice. This may not be correct as we shall show by example in § 5, but in the absence of other information we must seek the solution nearest y^0. Many methods compare y^0 to y^*—the predicted to the accepted solution—to estimate the error incurred in the step. If they are not close, the step will be rejected. As a practical matter the iteration must converge very quickly—a maximum of 3 iterations is most popular—so a good predicted solution is needed.

Despite the preceding arguments, it is *not* ordinarily required that G contract from y^0. If a rate of convergence is estimated at all, the rates exhibited in the latest iterations are given much more weight. Indeed, the line of codes following the structure of DIFSUB do not quit iterating even when $\|y^2 - y^1\| \gg \|y^1 - y^0\|$! We think that reliability demands that G be contracting from y^0, and we assume, henceforth, that it is doing so with constant r. The computable quantities

$$r_m = \frac{d_m}{d_{m-1}} = \frac{\|y^{m+1} - y^m\|}{\|y^m - y^{m-1}\|}$$

are lower bounds for r. If some $r_m > 1$, we are *certain* G is not contracting from y^0 and the iteration should be terminated. Lindberg [15] insists that convergence be rapid and terminates if any $r_m > 0.2$.

A reason many authors place most weight on the latest estimate r_m is that they focus their attention on the asymptotic behavior of the iteration. They say that the convergence is linear and that

$$r_m \to \lambda.$$

Convergence *is* linear. As [14, p. 301] proves, the root convergence rate is

$$R_1 = \rho(G'(y^*)), \tag{12}$$

where $\rho(\ )$ is the familiar notation for the spectral radius. The postulated limit (12) is not in general true, a result well known in another context: A perfectly reasonable possibility is that

$$G(y) = My + \phi.$$

With this G we find

$$y^{m+1} - y^m = M(y^m - y^{m-1}) = \cdots = M^m(y^1 - y^0),$$

which we recognize as the power method for computing the dominant eigenvalue of M. If this eigenvalue is real, (12) does hold with λ the magnitude of the eigenvalue. If the largest eigenvalue is a pair of complex conjugates, the ratio r_m will oscillate and assume values possibly much larger or smaller than the root convergence factor $\rho(M)$ [16]. In general we must anticipate the possibility that r_m is a misleading estimate for r and in particular, could be too small. Of course production codes discount this estimate to provide some robustness. However, we think that there is no point in trying to use the asymptotic behavior at all. It is far safer to presume a contraction with rate r from the initial point y^0 and to use the *largest* observed r_m as the best estimate for r available. We know this estimate is on the low side, so we must use it cautiously.

A well known result is

$$\|y^* - y^{m+1}\| \leqq \frac{r}{1-r}\|y^{m+1} - y^m\|. \tag{13}$$

A similar result for the residual of y^{m+1} is

$$\|y^{m+1}-G(y^{m+1})\| = \|G(y^m)-G(y^{m+1})\| \leqq r\|y^{m+1}-y^m\|. \tag{14}$$

The simplest examples show that no useful conclusion about the acceptability of y^{m+1} can be drawn from the size of $d_m = \|y^{m+1}-y^m\|$ without an estimate of the rate of convergence. Nevertheless, *all* codes known to this author will accept y^{m+1} (at least y^1), if d_m is small enough. A small difference says *nothing* about how close y^0 is to y^*, nor even that the process is converging. If the process *is* converging, $r<1$, a small difference does imply an acceptable result in terms of the residual. If it is converging at even a moderate rate, $r \leqq \frac{1}{2}$, a small difference implies an acceptable approximation to y^*. The codes try to obtain rapid convergence and it is to be presumed that they ordinarily succeed well enough that (13) and (14) justify the test on d_m. We insist that an estimate of the rate be made to establish some reliability. We much prefer the more stringent demand that y^{m+1} be close to y^*. Because of rapid convergence we see from (13) and (14) that this is only a little harder to get than a small residual. Thus if we want to test

$$\|y^*-y^{m+1}\| \leqq \tau,$$

we shall test

$$\frac{r}{1-r}\|y^{m+1}-y^m\| \leqq \tau, \tag{15}$$

using an approximation to r, and justify this by (13).

There is a subtle inconsistency in the algorithm of Hindmarsh [17, p. 3]. Because it has been followed in many codes, we shall clarify the matter. To do this we resort to the very clear and honest description of Hulme as to what he does in COLODE [18]. In our notation Hulme says that he seeks to make $d_m = \|y^{m+1}-y^m\|$ small. Here both y^{m+1} and y^m are computed. The test is described as a measure of the accuracy of y^m because y^{m+1} is (hoped to be) closer to y^* and so d_m approximates $\|y^*-y^m\|$. An estimate r_m is made of the asymptotic convergence rate and the approximation

$$d_{m+1} = \|y^{m+2}-y^{m+1}\| \doteq r_m\|y^{m+1}-y^m\| \tag{16}$$

is made. In this way one tests the acceptability of y^{m+1} without actually forming y^{m+2}. If the estimated difference is small enough, y^{m+1} is accepted. The inconsistency occurs when d_m is small enough that y^m is to be accepted. Hulme clearly states that as a heuristic measure, he will instead accept y^{m+1} on the grounds that if the process is converging, it will be more accurate. Hindmarsh does the same thing without comment. We have already noted that the approximation (16) is dangerous. It should be realized that neither author assumes that the approximation is at all good. For reasons already explained, the tests used by these authors ordinarily result in acceptable solutions.

We have not said anything about the relationship between the tolerance τ in (15) and the accuracy ε desired of the solution of the differential equation. This is a research question which needs attention. It is clear that τ must be smaller than ε. A small τ is one way to compensate for an inaccurate estimate of r used in the convergence test. However, the smaller τ is made, the more it costs to compute y^*. Experiment says that τ a great deal smaller than ε does not improve the solution of the differential equation. In current codes the relationship appears to be ad hoc. The choice varies wildly, but $\tau = 0.1\,\varepsilon$ is representative.

There can be difficulties with the precision. Most codes do not solve (10) as written. It is recast as

$$(I-h'\gamma' J)(y^{m+1}-y^m) = \Psi + h\gamma f(y^m) - y^m. \tag{17}$$

The right hand side is the residual of y^m in (9), so this is an attractive form. The iteration matrix is often rather ill-conditioned with the consequence that solving a linear system involving it may not be very accurate. If we work with (10), this means y^{m+1} itself is obtained inaccurately. However, because y^0 is close to y^*, the difference $y^{m+1}-y^m$ is rather small. Computing it from (17), we can usually get an accurate y^{m+1} even when $y^{m+1}-y^m$ has only a few digits correct. After a y^{m+1} is accepted, one must update the solution history. This is typically done as a correction to the predicted solution y^0. To control round-off, it is important that the correction $y^{m+1}-y^0$ be accumulated as $(y^1-y^0)+\cdots+(y^{m+1}-y^m)$ rather than formed directly from y^{m+1} and y^0. For these reasons we recommend using (17) instead of (10), but for the exposition of this paper either form will do. The form (17) leads to more accurate values of d_m, but as Watt notes in [19, p. 74] the difference could still be the result of round-off alone. When d_m is smaller relative to y^m than some multiple of the machine precision, we cannot reliably distinguish it from 0 nor can we use it to estimate the rate of convergence. In such a case it is reasonable to accept the last iterate on the grounds that it is as accurate as possible for the machine being used. This situation can come about because the prediction is extremely good or because the tolerance is very small. The algorithm for estimating the rate must be protected against this real possibility. Most current codes have convergence tests which cope with this automatically.

How many iterations should we allow? We have noted that current codes will terminate if $\|y^1-y^0\|$ is sufficiently small, usually rather smaller than is permitted in subsequent iterations. We have argued that an estimate of the rate of convergence is needed if one is to have a reliable convergence test. A number of codes in some circumstances use a rate of convergence estimated in the previous step to judge if y^1 is acceptable. It is easily seen that the rate is applicable to the new problem if the solution, $h\gamma$, and $h'\gamma'$ remain much the same. This is typical when solving stiff problems, but we could have no confidence in the rate estimate without verifying that these quantities *do* remain nearly constant. We know of no code which checks this. Even with this precaution, we do not think termination with y^1 a good idea. The defects of the estimation of the rate at the current step are serious enough without adding in more unreliability by allowing convergence at y^1, and the work saved is not very great.

With two iterations we get a current estimate of the rate of convergence. We think it best on the grounds of reliability to do at least two even though we might accept a y^2 unnecessarily close to y^*. Should one keep on iterating if necessary? Extra iterations are relatively cheap. Most codes stop at 3 or 4 iterations, although others such as Lindberg's [15] continue as long as the rate is satisfactory. We insist that y^0 be close to y^* so if many iterations are necessary, we must worry about the acceptability of y^*.

It is noteworthy that as soon as we have an estimate of the rate of convergence, we can begin predicting how many more iterations will be needed to finish up. Thus if we have computed y^m, an estimate for r, and a bound for $\|y^*-y^m\|$, we have the bound

$$\|y^*-y^{m+M}\| \leqq r^M \|y^*-y^m\|.$$

This can be useful in deciding whether it is profitable to continue the iteration.

A paper of Klopfenstein [3] sheds some light on the issue. He considers a generalization of the BDF when using a *fixed* number of iterations. In his analysis he considers the difference between the approximate Jacobian J and the (constant) true Jacobian $\mathcal{J}$ of his model problem. The integration is stable if the algebraic equations are solved exactly. If J does not differ much from $\mathcal{J}$, the solution arising from a fixed number of iterations is also stable. Klopfenstein considers as a function of the number of iterations, how much the two matrices can differ and still get a stable numerical solution.

As might be expected, the effect of more iterations is to permit J to differ more from $\mathcal{J}$. The interesting observation is that going from one to two iterations has a very beneficial effect, but that additional iterations are of rapidly decreasing value. This paper provides some support for using at least two iterations and not using many more than two.

There is evidence in the literature of unreliability of convergence tests. The GEAR and EPISODE codes permit the user to select three approximations to the Jacobian, viz., simple iteration, a diagonal approximation, and a full approximation. Regardless of the Jacobian approximation, if the convergence test is reliable, the codes should deliver a good solution to the problem. Of course the *efficiency* is affected, but the *accuracy* of the results should not be. In the tests of Byrne et al. [20], it is shown in Example 4 that both GEAR and EPISODE produce very large global errors when using the diagonal approximation to the Jacobian whereas the error is controlled satisfactorily with the other two approximations. Because the rest of the code is apparently unaltered by the choice of approximate Jacobian, it appears that the convergence test is unreliable, and that the potential unreliability can sometimes be exhibited as the result of a very poor approximate Jacobian.

4. Changing step size and Jacobian. In the previous section we were concerned with the solution of

$$y = h\gamma f(y) + \Psi \tag{18}$$

as an isolated computation. It must be done at every step. Viewing the integration as a whole reveals extremely important economies. We shall consider various aspects of this, particularly the effects of changing the step size or the approximate Jacobian.

Quite a lot of useful information can be gleaned from a simple expression for the error in an iteration. We are solving (18) for a root y^* by

$$y^{m+1} = \Psi + h\gamma f(y^m) + h'\gamma' J(y^{m+1} - y^m). \tag{19}$$

This implies

$$y^* - y^{m+1} = (I - h'\gamma' J)^{-1}[h\gamma(f(y^*) - f(y^m)) - h'\gamma' J(y^* - y^m)]$$

and

$$y^* - y^{m+1} = (I - h'\gamma' J)^{-1}[h\gamma(\mathcal{J}_m - J) + (h\gamma - h'\gamma')J](y^* - y^m). \tag{20}$$

Here $\mathcal{J}_m$ is the Jacobian matrix with its entries evaluated on a line between y^* and y^m. The most important situations we investigate in this section have $h\gamma = h'\gamma'$. In such a case (19) and (20) simplify to

$$y^{m+1} = \Psi + h\gamma f(y^m) + h\gamma J(y^{m+1} - y^m), \tag{19a}$$

$$y^* - y^{m+1} = (I - h\gamma J)^{-1} h\gamma(\mathcal{J}_m - J)(y^* - y^m). \tag{20a}$$

The error expression (20a) is essentially that given by Robertson and Williams [4], but they did not attempt to exploit it as we do.

In the first solutions of stiff problems, a new approximate Jacobian J was formed at every step, and corresponding iteration matrices formed and factored. As experience accumulated, it became clear that these computations are a large, and often dominant, part of the cost of solving a typical problem. The sole purpose of these computations is to secure adequate convergence to y^*. It was quickly realized that great savings are possible by using the same iteration matrix as long as $h\gamma$ is an acceptable step size and formula, and convergence is adequate. This has become standard. Further savings of this nature are attempted in the most recent codes. We shall discuss them below.

Another source of improved efficiency has been to take account of the structure of the Jacobian. This can reduce significantly the cost of forming J by differences. It can also reduce significantly the cost of factoring the iteration matrix and of solving the linear systems. The use of structure is important for problems of medium size and vital for large problems.

The fragments of theory available for stiff problems give a great deal of attention to model problems. For the typical models to have any validity, it is necessary that the solution of the differential equation be slowly varying and that the Jacobian be approximately constant in a neighborhood of the solution. We did not make it clear enough in [21] that there are important *practical* reasons for supposing this situation. If the problem is stiff, we hope to be able to use "large" step sizes h. According to (20a), this is not likely to be possible unless J is close to $\mathcal{J}_m$ for each m. This requires that the Jacobian be nearly constant near y^*. Using a factorization of $I - h\gamma J$ for several steps is not likely to be possible unless the solution changes slowly and the Jacobian changes slowly along the solution.

If the convergence test is reliable, the only role J plays is to affect the rate of convergence. Usually one thinks of J as an approximation to $\mathcal{J}(y^*)$, except when we take $J = 0$ to have simple iteration, but there are reasons for not proceeding so simply. The critical task is to make $\|\mathcal{J} - J\|$ of acceptable size; "small" elements of $\mathcal{J}$ need not be approximated at all well. Advantage can be gained by setting elements of J to zero if they approximate small elements of $\mathcal{J}$. The idea is to work with a J whose structure permits one to form it and to factor the iteration matrix more cheaply than with the actual structure of $\mathcal{J}$ [22].

Formation of a J by numerical differencing is extremely common despite serious scaling difficulties. The typical way of generating a column of J is by

$$\frac{\partial f(y)}{\partial e_j} \doteq \frac{f(y + \delta e_j) - f(y)}{\delta},$$

where e_j is the jth column of the identity matrix. The scalar δ is selected to get a reasonable approximation. The trouble is that to be small enough to give a reasonable approximation to large $\mathcal{J}_{ij}$, δ is frequently so small that the difference consists only of roundoff error for those components corresponding to small $\mathcal{J}_{ij}$. We see now why this standard procedure may give acceptable J even when the approximation of small elements is dreadful.

On general grounds we can argue that most codes form a new J far too frequently. For the common case of Jacobians which are fairly expensive, we would like to avoid this wasted effort. Sometimes Jacobians are inexpensive, e.g., they are "free" for linear problems. Avoiding the evaluation of a Jacobian in such a case is unnecessary, but should do no harm. In the absence of information about the relative cost of evaluating a Jacobian, we shall presume that it is substantial and seek to minimize it. If the problem is nonstiff locally, the code must use a small step size, so small that J is essentially ignored and, in effect, simple iteration is done. If the problem is stiff locally, the solution is slowly varying and the Jacobian is roughly constant. If J is a Jacobian approximation based on a solution of about the current size, the need for a new iteration matrix is likely to be due almost solely to changing $h\gamma$, in part because we have seen that J does not need to be very accurate. In either case the arguments suggest that a good tactic is to keep a copy of J, and when forming a new iteration matrix, to try first the old J. The substantial increase of the storage required by the code is probably why few codes do this. Experiments done some years ago by the author and M. K. Gordon showed that the cost could be reduced significantly in DIFSUB [13]. Recently A. R. Curtis [24, p. 271] and T. Chambers

[25, p. 243] make the same point based on their considerable experience. Below we make more specific recommendations, and so comment here only that provided sufficient storage is available, reuse of approximate Jacobians can save a lot of work.

There are two reasons for adjusting the step size. One is to match the accuracy of the solution at each step to the user's requirements. It is insisted that the accuracy he demands be achieved. For efficiency we try to use the largest step size which will satisfy these demands. This task has been well studied. The other reason for adjusting the step size is to secure an adequate rate of convergence in the use of an implicit formula. We have noticed no prior investigations of this task and all the codes appear to use ad hoc devices. Here we shall develop some insight as to the effects of changes of step size and formula which suggest specific ways to proceed.

The part of the error expression (20a) which depends on the step size and formula is the error matrix $(I-h\gamma J)^{-1}h\gamma I$. Suppose λ is an eigenvalue of J. There is a corresponding eigenvalue of the error matrix which is

$$e(\lambda)=\frac{h\lambda}{1-h\gamma\lambda}. \tag{21}$$

Speaking loosely, the "nonstiff" eigenvalues are those with $|h\lambda|\ll 1$. In this case

$$e(\lambda)\sim h\lambda \quad \text{if } |h\lambda|\ll 1, \tag{22}$$

so that errors corresponding to "nonstiff" eigenvalues are strongly damped. "Stiff" eigenvalues are those with $\operatorname{Re}(\lambda)\leqq 0$ and $|h\lambda|\gg 1$. In this case

$$e(\lambda)\sim -\frac{1}{\lambda} \quad \text{if } \operatorname{Re}(\lambda)\leqq 0, |h\lambda|\gg 1. \tag{23}$$

Perhaps surprisingly, the errors corresponding to "stiff" eigenvalues are very heavily damped. More generally we have the *bound*

$$|e(\lambda)|\leqq h\gamma \quad \text{if } \operatorname{Re}(\lambda)\leqq 0. \tag{24}$$

We have just explored the behavior of the iteration error when we advance from a point x with step size h and formula yielding γ in (18). Now we ask what would happen if instead we advanced with step size h'' and formula yielding γ''. Supposing that the same approximate Jacobian J is used in either case, the error matrix corresponding to the second computation has an eigenvalue

$$e''(\lambda)=\frac{h''\gamma''}{1-h''\gamma''\lambda}.$$

Comparing this to (21) gives us a lot of insight as to the effect of a change of $h\gamma$ to $h''\gamma''$ on the iteration error. It is not precise because the $\mathscr{J}_m$ in (20a) is altered somewhat for the new problem. Perhaps more significant is that in a suitable norm the eigenvalues of the error matrix tell us what happens in an iteration, but a code must use a specific, computable norm. With these qualifications in mind we see that, at least roughly, the effect of a change from $h\gamma$ to $h''\gamma''$ is to change the bound (24) on the rate proportionately. For special problems such as constant Jacobians and either a suitable norm or a special structure such as a diagonal Jacobian, the qualifications to the analysis do not apply and the conclusions are quantitative. In general, the approximation of (22) says that we are finding the correct behavior for "nonstiff" eigenvalues. In particular, our qualitative guide agrees with the true behavior when simple iteration is used. Now let us take up some applications of this argument that the rate of convergence is roughly proportional to $h\gamma$.

There are a variety of situations we must discuss. As we take them up in turn we shall apply our insight to arrive at a recommendation and state what current practice is. When we try a step with $h\gamma$ we are first faced with the possibility that we are unable to solve the implicit equation (18) with acceptable efficiency. If a solution is obtained, the local error is estimated and it is decided whether to accept the step. If the step is rejected, a new step size, and possibly a new formula, is chosen which is expected to yield the desired accuracy when the code tries again to take a step. If the step is accepted, a new step size, and possibly a new formula, is chosen which is expected to yield the desired accuracy on the *next* step.

We have argued that the typical code forms a new approximate Jacobian too often. We advocate keeping a copy of J and judiciously reusing it. This does require extra storage, but we think it worthwhile. As we take up the possibilities we shall see how this can reduce the number of Jacobian evaluations significantly.

Suppose we fail to get convergence quickly enough when solving (18), and that the J used was not computed at the current data. Certainly we can secure convergence with this J if we reduce the step size enough. However, the whole point of using approximate Jacobians in the iteration is to be able to use a step size appropriate to the truncation error of the formula. We have just tried such a step size and failed to get adequate convergence. It is plausible that we need a new J. We suggest that a new J be formed at every convergence failure with J out of date. There is no reason to think the step size unsatisfactory, so we suggest trying it again. These recommendations agree with current practice, in part because only a couple of codes keep a copy of J so that they can even consider its reuse.

Suppose we fail to get adequate convergence and that the J used was computed at the current data. We shall reduce the step size and form an iteration matrix using a copy of J. Current practice forms a new J. This serves no purpose at all except to avoid the storage for a copy. It is quite possible that we fail to secure convergence repeatedly, especially if we reduce the step size by a factor independent of the observed behavior of the iteration. Repeatedly forming the *same* approximate Jacobian J is obviously wasteful. Having monitored the behavior of the iteration, we find three situations in which we must select a step size. First it may happen that the iteration is diverging. Our arguments about the effect of reducing the step size do not then apply. In the absence of insight, we make a traditional reduction of the step size by an arbitrary fixed factor such as $\frac{1}{4}$. We remark that the action we suggest after a successful step is designed to prevent the occurrence of this situation. The second possibility is that the iteration is converging at rate r which we regard as too slow. We predict that on changing $h\gamma$ to $h''\gamma''$ the rate will be changed to roughly $r(h''\gamma''/h\gamma)$. This tells us how to reduce h to get an acceptable rate. Naturally we should insist on a substantial reduction because of the crudity of the approximations. With our new insight, we take into account the observed behavior and, in particular, we respond more quickly than with the traditional fixed factor. The third possibility is more exotic. It could happen that the rate of convergence is acceptably fast, but the predicted value is so much in error that convergence is not obtained in the permitted number of iterations. This shows a breakdown in the assumptions about the predictor, e.g., the derivative of the solution might be discontinuous in the course of the step. How one wants to handle such a software issue is a matter of taste. One might prefer to quit immediately. Our preference is to reduce the step size and try again. We prefer to terminate always with the statement that the code apparently needs an unreasonably small step size to continue. If our preference is followed, we have no information on which to base the reduction of the step size, so an arbitrary fixed factor is as reasonable as any way of making the reduction. We emphasize that of the three

possibilities raised, the second should be the most common by far and with our new insight, we are able to respond rationally.

Having dealt with the various manifestations of a failure to get adequate convergence, let us now take up the cases resulting from the successful solution of (18) with $h\gamma$. First suppose we got convergence, but we reject the step. There is no reason for forming a new J: Convergence was at a satisfactory rate, and we anticipate that with the reduction of $h\gamma$ to $h''\gamma''$, it will be accelerated. (This is what everyone has always expected but, in fact, it is not necessarily true that reducing the step size results in an iteration that converges even *as* fast. Our insight explains why it should usually result in a faster iteration.) The predicted value y^0 will be more accurate for the new try with a smaller step size h''. For these reasons the formation of a new approximate Jacobian, as is typical, seems to have no point except to avoid storing a copy of J. As with repeated failures to secure convergence, it is possible to fail repeatedly to accept a step due to a sudden change in the solution.

Finally suppose we got convergence and we accept the step. Unless the step size can be increased "substantially" for the next step, changing it will not be worth the cost of factoring the iteration matrix. If the increased step size appears worthwhile, we have to consider whether to form a new J. The older codes do; Curtis [24] does not. We suggest an intermediate tactic. Our argument says that if we use the old J, the observed rate of convergence r should slow down by a factor of roughly $h''\gamma''/h\gamma$. If this is still an acceptable rate, we suggest using the J stored. If it is not, it appears prudent to form a new J.

So far we have considered only "substantial" changes of step size because of the cost of factoring a new iteration matrix and the cost of forming a new approximate Jacobian. There are a number of reasons why "small" changes would be desirable. For one, "small" changes produce a smoother behavior of the true error with respect to the tolerance ε. The algorithm for choosing a step is conservative so as to prevent expensive rejections. After a successful step the algorithm may well say that the next step ought to be a little shorter for this reason. (It is only a little shorter because of the effect of the order of the formula.) This warning is usually ignored because of the expense of a change. In the extremely important case of the BDF, it is known that the integration can become unstable in the presence of step size and order changes. Gear, Tu, and Watanabe [26] prove stability if "small" step size changes are made.

It is very tempting to make small changes in the step size without changing the iteration matrix. This puts us in the general case of (20), where we have an iteration matrix $I-h'\gamma'J$ and take the step with $h\gamma$ which may differ from $h'\gamma'$. It is cheap to change the step size in these circumstances. Comparing the iteration matrix we have to the one we would like to have, $I-h\gamma J$, we are a little discouraged. The perturbation is one of full rank and in the solution of stiff problems, the term $h'\gamma'J$ often dominates I. Nevertheless, Hindmarsh has implemented this idea in the GEAR package, where he uses the old iteration matrix if the change in $h'\gamma'$ is not more than 30%. It is still required that an *increase* of the step size of at least 10% be possible to even consider a change. We think the idea particularly useful when a small *decrease* is called for. It seems to us that the observed rate of convergence ought to be extremely important in deciding whether to try to use an old iteration matrix. Our ideas for analyzing (20a) provide some insight.

There are two terms in the error matrix of (20), and it is not clear how to deal with their coupling because of the matrix $\mathscr{J}_m - J$. We shall treat them separately and presume the worst by supposing that they add. The situation is that we have an iteration matrix $I-h'\gamma'J$ and have just made a successful step with $h\gamma$ which might, or might not, be the

same as $h'\gamma'$. We consider taking a step with $h''\gamma''$ which is different from $h'\gamma'$. The effect on the first term is easily analyzed as we did in the more special case. We predict that the rate of convergence will be altered by the factor $(h''\gamma''/h\gamma)$. Proceeding similarly with the second term we see that if λ is an eigenvalue of J, the matrix $(I-h'\gamma'J)^{-1}(h''\gamma''-h'\gamma')J$ has an eigenvalue of magnitude

$$\left|\frac{h''\gamma''-h'\gamma'}{h'\gamma'}\,\frac{h'\gamma'\lambda}{1-h'\gamma'\lambda}\right| \leqq \left|\frac{h''\gamma''-h'\gamma'}{h'\gamma'}\right| \quad \text{if } \operatorname{Re}(\lambda)\leqq 0. \tag{25}$$

As with the first term in (20) we find that "nonstiff" eigenvalues are strongly damped. Unfortunately the qualitative behavior with respect to "stiff" eigenvalues is altered. Instead of being damped strongly, the damping depends on how close $h''\gamma''$ is to $h'\gamma'$ in a relative sense. This is quite consistent with our earlier remarks about the nature of the perturbation. These arguments *suggest* that if r is the observed rate of convergence, we inspect

$$r\left(\frac{h''\gamma''}{h\gamma}\right)+\left|\frac{h''\gamma''-h'\gamma'}{h'\gamma'}\right|$$

to see if it is less than the rate of convergence we are willing to tolerate. If it is, we might change the step size and retain the iteration matrix. Otherwise, this action seems imprudent. Qualitatively this appears to be the right kind of criterion because small perturbations to a rapidly convergent process appear the ones most likely to succeed. This criterion would not often be as bold as the one used by Hindmarsh but, in contrast, would likely be used to reduce the step size to forestall a failure.

We are quite conscious of the limitations of the analysis of this section, but the arguments do provide some insight. There is quite a range of possible action. For example, after a successful step we might retain the step size and iteration matrix, change the step size and retain the iteration matrix, change the step size and iteration matrix while retaining the approximate Jacobian, or change the step size and the approximate Jacobian and the iteration matrix. The decisions must be made, and they are of great practical importance. Any new insight is of obvious value. Our observations provide some guidance as to appropriate action.

5. An example. We have constructed a simple example to make a number of points. In very reasonable circumstances we show that

(i) an implicit method (3) can have more than one solution y^*;
(ii) the predicted solution y^0 can be closer to the "wrong" y^*;
(iii) a convergence test based on the difference of successive iterates can accept an approximate solution arising in a divergent iteration.

The example was originally constructed to explain in simple terms a numerical difficulty observed in the solution of some problems of chemical kinetics [27], [28]; accepting the "wrong" solution of (3) leads to disaster.

We shall solve

$$y' = -100|y| = f(y), \qquad y(0)=1. \tag{26}$$

On the grounds that the solution is physically uninteresting when it is sufficiently small, the problem is to be solved in the sense of absolute error. Notice that if we should somehow generate a negative approximation y_n, the differential equation is unstable and we shall get absurd results from then on. This is essentially what happens in the computations described in [27], [28]. The difficulty arises in our failure to recognize that

we might get into an unstable region and our consequent failure to take action to prevent negative values.

The numerical methods we use are quite reasonable. Steps are taken with the backward Euler method

$$y_{n+1} = y_n + hf(y_{n+1}) \quad \text{or} \quad y^* = y_n + hf(y^*). \tag{27}$$

The prediction is done with the forward Euler method

$$y^0 = y_n + hf(y_n).$$

As we use it, we solve (27) exactly so there is no error in $f(y_n)$ here. This is a common predictor for the backward Euler formula. It will be obvious that the other common predictor, which does linear extrapolation of y_n and y_{n-1}, behaves qualitatively the same in our example. To solve (27), we use

$$y^{m+1} = y_n + hf(y^m) + hf'(y_n)(y^{m+1} - y^m). \tag{28}$$

Here we are using the current h and as good an approximation to the Jacobian as we ever expect to have.

First suppose that $h \leqq 10^{-2}$. The predicted

$$y^0 = (1 - 100h)y_n > 0.$$

The scheme (28) is exact in one iteration and gives

$$y_{n+1} = y^* = y^1 = \frac{y_n}{1 + 100h}. \tag{29}$$

The estimated error of the step is a multiple of

$$|y^* - y^0| = \left| \frac{1}{1 + 100h} - (1 - 100h) \right| y_n. \tag{30}$$

We need be no more precise about the error estimate. Obviously if h and/or y_n is small enough, the error estimate satisfies any absolute error test. As (29) shows, the numerical solution monotonely decreases. Because of the form of (30), we see that the error test is going to be passed with an increasingly large margin. Thus automatic adjustment of the step size will begin to increase the step size. Eventually we are led to try an $h > 10^{-2}$.

Supposing now that $y_n > 0$ but $h > 10^{-2}$, we find that (27) has *two* solutions, namely,

$$y_1^* = \frac{1}{1 + 100h} y_n > 0, \qquad y_2^* = \frac{1}{1 - 100h} y_n < 0.$$

Furthermore

$$y^0 = (1 - 100h)y_n < 0.$$

Depending on h, y^0 can be closer to y_2^* than y_1^*. In fact, the reasonable value $h = 2 \times 10^{-2}$ has $y^0 = y_2^* = -y_n$! The iteration (28) has

$$y^1 = y_n - 100h|y^0| - 100h(y^1 - y^0),$$

hence

$$y^1 = \frac{y_n - 200h|y^0|}{1 + 100h}.$$

The iterate y^1 is certainly negative for $h \geqq 2 \times 10^{-2}$. If $y_n > 0$ is small enough, any convergence test based on $|y^1 - y^0|$ will be passed and y^1 accepted as the approximation of y^*.

To investigate the iteration (28), we write it in the form $y^{m+1} = G(y^m)$. We must take into account that $y_n > 0$ in obtaining the proper $f'(y_n)$, that $h > 10^{-2}$, and that $y^0 < 0$. We find

$$G(y) = (1+100h)^{-1}(y_n + 200hy) \quad \text{for } y < 0.$$

Because

$$G'(y) = \frac{200h}{1+100h} > 1,$$

G does not contract for negative y; in particular, y_2^* is a point of repulsion. The iteration begun with $y^0 < 0$ is diverging.

6. Acknowledgment. The author is grateful to A. R. Curtis who pointed out to him that he had left out an important term in (20). This led to substantial alteration of a draft of § 4. The published work of Curtis and of Robertson and Williams have materially influenced portions of this paper. The comments of an anonymous referee improved the presentation.

REFERENCES

[1] J. D. LAMBERT, *The initial value problem for ordinary differential equations*, The State of the Art in Numerical Analysis, D. Jacobs, ed., Academic Press, New York, 1977, pp. 451–500.

[2] F. T. KROGH, *The numerical integration of stiff differential equations*, Rept. 99900-6573-R000, TRW Systems Group, Redondo Beach, CA, March 1968.

[3] R. W. KLOPFENSTEIN, *Numerical differentiation formulas for stiff systems of ordinary differential equations*, RCA Rev., 32 (1971), pp. 447–462.

[4] H. H. ROBERTSON AND J. WILLIAMS, *Some properties of algorithms for stiff differential equations*, J. Inst. Math. Appl., 16 (1975), pp. 23–34.

[5] H. H. ROBERTSON, *Some factors affecting the efficiency of stiff integration routines*, Numerical Software—Needs and Availability, D. Jacobs, ed., Academic Press, New York, 1978, pp. 279–302.

[6] R. K. BRAYTON, F. C. GUSTAVSON AND G. D. HACHTEL, *A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas*, Proc. IEEE, 60 (1972), pp. 98–108.

[7] J. WILLIAMS AND F. DEHOOG, *A class of A-stable advanced multistep methods*, Math. Comput., 28 (1974), pp. 163–177.

[8] R. ALT, *A-stable one-step methods with step-size control for stiff systems of ordinary differential equations*, J. Comput. Appl. Math., 4 (1978), pp. 29–35.

[9] R. W. KLOPFENSTEIN AND C. B. DAVIS, *PECE algorithms for the solution of stiff systems of ordinary differential equations*, Math. Comput., 25 (1971), pp. 457–473.

[10] J. C. BUTCHER, *Implicit Runge–Kutta processes*, Math. Comput., 18 (1964), pp. 50–64.

[11] L. F. SHAMPINE, *Solving ODEs with discrete data in SPEAKEASY*, Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 177–192.

[12] S. P. NORSETT, *Semi-explicit Runge–Kutta methods*, Rept. No. 6/74 ISBN 82-7151-009-6, Dept. of Mathematics, University of Trondheim, Norway.

[13] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

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

[15] B. LINDBERG, *IMPEX 2 a procedure for solution of systems of stiff differential equations*, Rept. TRITA-NA-7303, Dept. of Information Processing Computer Science, Royal Inst. of Technology, Stockholm, Sweden.

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

[17] A. C. HINDMARSH, *The control of error in the GEAR package for ordinary differential equations*, Rept. UCID-30050 part 3, Lawrence Livermore Laboratory, Livermore, CA, August, 1972.

[18] B. L. HULME AND S. L. DANIEL, *COLODE: A colocation subroutine for ordinary differential equations*, Rept. SAND74-0380, Sandia Laboratories, Albuquerque, NM, December 1974.

[19] J. M. WATT, *General discussion of implementation problems*, Modern Numerical Methods for Ordinary Differential Equations, G. Hall and J. M. Watt, eds., Clarendon Press, Oxford, England, 1976.

[20] G. D. BYRNE, A. C. HINDMARSH, K. R. JACKSON AND H. G. BROWN, *A comparison of two ODE codes: GEAR and EPISODE*, Computers and Chem. Engrg., 1 (1977) pp. 133–147.

[21] L. F. SHAMPINE AND C. W. GEAR, *A user's view of solving stiff ordinary differential equations*, SIAM Rev., 21 (1979), pp. 1–17.

[22] A. R. CURTIS, M. J. D. POWELL AND J. K. REID, *The estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[23] A. C. HINDMARSH AND G. D. BYRNE, *EPISODE: An experimental package for the integration of systems of ordinary differential equations*, Rept. UCID-30112, Lawrence Livermore Laboratory, Livermore, CA, May 1975.

[24] A. R. CURTIS, *Solution of large, stiff initial value problems—the state of the art*, Numerical Software—Needs and Availability, D. Jacobs, ed., Academic Press, New York, 1978, pp. 257–278.

[25] T. CHAMBERS, *The use of numerical software in the digital simulation language PMSP*, Numerical Software—Needs and Availability, D. Jacobs, ed., Academic Press, New York, 1978, pp. 237–253.

[26] C. W. GEAR, K. W. TU AND D. S. WATANABE, *The stability of automatic programs for numerical problems*, Stiff Differential Systems, R. A. Willoughby, ed., Plenum Press, New York, 1974, pp. 111–121.

[27] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of ODEs*, Nordisk Tidskr. Informations behandling (BIT), 15 (1975), pp. 10–48.

[28] L. EDSBERG, *Numerical methods for mass action kinetics*, Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, 1976, pp. 181–195.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0006 $01.00/0

ITERATIVE METHODS FOR THE NUMERICAL SOLUTION OF SECOND ORDER ELLIPTIC EQUATIONS WITH LARGE FIRST ORDER TERMS*

JOHN STRIKWERDA†

Abstract. This paper presents iterative methods for the numerical solution of second order elliptic equations whose first order terms have coefficients that are orders of magnitude larger than those of the second order terms. Such equations arise in singular perturbation problems and also in the numerical grid generation technique of Mastin and Thompson. These equations exhibit boundary layer phenomena which usually require an unevenly spaced grid for their numerical solution. The methods are similar to successive overrelaxation, but have the advantage of not requiring the user to supply a parameter. The methods are shown to be stable even for variable coefficients by using the theory of pseudo-translation operators; however no proof of convergence is given. Numerical results are presented and discussed.

Key words. iterative method, elliptic equations, Reynold's number, boundary layer

1. Introduction.

Consider the elliptic equation

$$Au_{xx}+2Bu_{xy}+Cu_{yy}+Du_x+Eu_y+Fu=G(x,y) \tag{1.1}$$

defined in a domain Ω in $\mathbb{R}^2$. The coefficients are assumed to be smooth, slowly varying functions of the independent variables (x, y), and also

$$AC-B^2\geqq\delta>0, \qquad A\geqq 0,$$

on Ω. If L is a reference length for Ω, such as the diameter, then we define the Reynolds number for equation (1.1) as

$$R(x,y)=\frac{L\sqrt{D^2+E^2}}{\frac{1}{2}(A+C)}. \tag{1.2}$$

This definition of the Reynolds number is somewhat arbitrary and any similar expression that relates the magnitudes of the coefficients of the first order terms to those of the second order terms would suffice for our purposes.

In this paper we will consider elliptic equations for which the Reynolds number is large, on the order of a thousand at least. We will also assume that the coefficient $F(x, y)$ is of the same, or less, order of magnitude as the coefficients $D(x, y)$ and $E(x, y)$, and that the coefficients $A(x, y)$ and $C(x, y)$ are of the same order of magnitude with respect to the Reynolds number. Such equations as these arise frequently in applications, usually as singular perturbation problems.

The methods presented in this paper are designed for the numerical solution of elliptic equations with large Reynolds number. They formally resemble successive-over-relaxation (SOR) and they will be referred to as SRR—successive relaxation for large Reynolds number.

Unlike SOR which requires a priori knowledge of the iteration parameter, in SRR the iteration parameter is chosen at each grid point according to a formula derived from a stability criterion. This formula for the case of the frequently encountered five-point difference operator is derived in § 5. Computational results are described in the last

* Received by the editors April 30, 1979, and in final revised form September 27, 1979.

† Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665. This work was supported by the National Aeronautics and Space Administration under Contract NAS1-14101.

section and the notable features of SRR are presented. The author is currently working on a proof for the convergence of the SRR method.

Iterative numerical methods for elliptic equations with sizeable lower order terms have been studied by several other authors; we mention only Concus and Golub [2] and Widlund [11]. They, however, do not consider problems with boundary layers and nonuniform grids which are essential features of SRR.

2. Boundary layers. The solutions of Dirichlet problems for elliptic equations with a large Reynolds number frequently have boundary layers. Boundary layers are regions near the boundary where the solution has very large gradients, and they are located at those boundary points where

$$D(x, y)n_x + E(x, y)n_y > 0, \tag{2.1}$$

and (n_x, n_y) are the direction cosines for the interior normal at (x, y). The condition (2.1) can be established by the method of perturbation expansions (see Nayfeh [4], Van Dyke [10]).

In the numerical solution of such problems a nonuniform grid is frequently employed to place more grid points in the boundary layer region to resolve the solution more accurately. In order to study some effects of the grid on the solution, consider the one-dimensional equation

$$u_{xx} + Ru_x = 0 \quad \text{on } 0 \leqq x \leqq 1 \tag{2.2}$$

with the boundary data

$$u(0) = 0, \qquad u(1) = 1.$$

To resolve the boundary layer at $x = 0$ we introduce the coordinate transformation

$$q = q(x), \quad q(0) = 0, \quad q(1) = 1$$

where

$$q'(x) > 0 \quad \text{and} \quad q'(0) \gg 1.$$

Equation (2.2) then becomes

$$(q'u_q)_q + Ru_q = 0,$$

which can be approximated by the difference equations

$$\begin{gathered} h^{-2}(q'_{i+1/2}(u_{i+1} - u_i) - q'_{i-1/2}(u_i - u_{i-1})) + \tfrac{1}{2}Rh^{-1}(u_{i+1} - u_{i-1}) = 0, \\ i = 1, 2, \cdots, N-1, \end{gathered} \tag{2.3}$$

with

$$u_0 = 0, \qquad u_N = 1.$$

The solution to (2.2) is a monotone increasing function and we will now examine (2.3) to see when its solution is also monotone. We rewrite (2.3) as

$$u_i = \frac{1}{q'_{i+1/2} + q'_{i-1/2}}\left[\left(q'_{i+1/2} + \frac{Rh}{2}\right)u_{i+1} + \left(q'_{i-1/2} - \frac{Rh}{2}\right)u_{i-1}\right].$$

From this it is easy to see that the solution will be monotone if and only if both coefficients on the right-hand side of the above equation are positive. This shows that a

necessary and sufficient condition for the solution to be monotone is that

$$\frac{Rh}{q'_{i-1/2}} \leqq 2 \quad \text{for } i = 1, 2, \cdots, N. \tag{2.4}$$

The quantity on the left side of inequality (2.4) is called the cell Reynolds number. Note that the condition is essentially

$$R(x_i - x_{i-1}) \leqq 2, \tag{2.4'}$$

and it imposes a restriction on the grid spacing.

In computation, it is seen that if inequality (2.4) is violated at grid points away from the boundary layer then the resulting oscillations in the solution of equation (2.3) are not large and do not severely affect the accuracy of the solution. However, if the inequality is violated in the boundary layer region then the oscillations can be very large and will severely affect the accuracy of the solution.

It is important to note that the cell Reynolds number condition is a statement about the solution of the difference equations (2.3) and is independent of the solution procedure. For two-dimensional problems using a scheme analogous to the above one-dimensional problem, the cell Reynolds number condition remains approximately valid in the neighborhood of the boundary layer. One might also use a scheme based on one-sided or "up-wind" differencing for the first derivatives and such a scheme might not have a cell Reynolds number restriction for obtaining monotone solutions. For such a scheme the cell Reynolds number serves principally as a measure of the accuracy of the solution. For more discussion of the cell Reynolds number condition see Roache [6], and for a finite element approach to this subject see Christie et al. [1].

3. The SRR method: an example. Before introducing the method in general, we will consider as an illustration the equation

$$u_{xx} + u_{yy} + R_1 u_x + R_2 u_y = 0 \tag{3.1}$$

on the square

$$0 \leqq x \leqq 1, \qquad 0 \leqq y \leqq 1,$$

with u specified on the boundary. To resolve the boundary layers we introduce a change of coordinates given by

$$q = q(x), \qquad p = p(y)$$

where $q(x)$ and $p(y)$ are smooth, strictly increasing functions and

$$q(0) = p(0) = 0, \qquad q(1) = p(1) = 1.$$

Equation (3.1) then becomes, in the new coordinates,

$$q'(q'u_q)_q + p'(p'u_p)_p + R_1 q' u_q + R_2 p' u_p = 0, \tag{3.2}$$

where

$$q' = \frac{dq}{dx} \quad \text{and} \quad p' = \frac{dp}{dy}.$$

This equation can then be replaced by a difference approximation using a uniform grid

in the (q, p) unit square. We write this difference approximation only as

$$m_{ij}^{1,0}u_{i+1j}+m_{ij}^{-1,0}u_{i-1j}+m_{ij}^{0,1}u_{ij+1}+m_{ij}^{0,-1}u_{ij-1}$$
$$-(m_{ij}^{1,0}+m_{ij}^{-1,0}+m_{ij}^{0,1}+m_{ij}^{0,-1})u_{ij}=0,$$

where the coefficients $m_{i,j}^{a,b}$ depend on the precise form of differencing that is used.

By use of the natural ordering of points and immediate replacement, this system of difference equations can be solved by the following algorithm

$$u_{ij}^{n+1}=u_{ij}^{n}+\omega_{ij}\{m_{ij}^{1,0}u_{i+1j}^{n}+m_{ij}^{-1,0}u_{i-1j}^{n+1}+m_{ij}^{0,1}u_{ij+1}^{n}+m_{ij}^{0,-1}u_{ij-1}^{n+1}$$
$$-(m_{ij}^{1,0}+m_{ij}^{-1,0}+m_{ij}^{0,1}+m_{ij}^{0,-1})u_{ij}^{n}\}. \tag{3.3}$$

As will be shown later, the optimal choice of the iteration parameter ω_{ij} for large values of R_1 and R_2 is given by

$$\omega_{ij}=\frac{2}{\tilde{m}+\sqrt{\tilde{m}}\sqrt{(m^{1,0}-m^{-1,0})^2/(m^{1,0}+m^{-1,0})+(m^{0,1}-m^{0,-1})^2/(m^{0,1}+m^{0,-1})}} \tag{3.4}$$

where $\tilde{m}=m^{1,0}+m^{-1,0}+m^{0,1}+m^{0,-1}$ and the subscripts (i, j) have been omitted for convenience.

This example will be discussed at more length in § 6 and the derivation of the expression for ω_{ij} will be given in § 5. For now we point out only that for the particular case whose results are given in the first part of Table 1, the number of iterations required for convergence is nearly constant, independent of R for values of R between 6,000 and 80,000.

4. The SRR method. We now present the SRR method in detail. We begin with equation (1.1) and transform coordinates to the independent variables (q, p) in order to improve the resolution of the boundary layers. This transformed equation again has the form of equation (1.1). We will assume for simplicity of exposition that $\bar{\Omega}$, the image of Ω in the (q, p) coordinates, is the unit square. On $\bar{\Omega}$ we take a uniform grid with points

$$Q_\alpha=(\alpha_1 h_1, \alpha_2 h_2)$$

indexed by the multi-index $\alpha=(\alpha_1, \alpha_2)$. The α_i are integers with

$$0\leqq\alpha_i\leqq h_i^{-1}$$

where the quantities h_i^{-1} are also integers.

The difference approximation to the equation can then be written as

$$\sum_{|\beta|\leqq k} M_{\alpha\beta}u_{\alpha+\beta}=G_\alpha\simeq G(q_\alpha, p_\alpha) \tag{4.1}$$

for all multi-indices α with $Q_\alpha\in\bar{\Omega}$ and some positive integer k. The norm of the multi-index β is given by

$$|\beta|=|(\beta_1, \beta_2)|=|\beta_1|+|\beta_2|.$$

The class of iterative methods we discuss here are given in general by

$$u_\alpha^{n+1}=u_\alpha^{n}+\omega_\alpha\Big(\sum_{|\beta|\leqq k} M_{\alpha\beta}^{0}u_{\alpha+\beta}^{n}+\sum_{|\beta|\leqq k} M_{\alpha\beta}^{1}u_{\alpha+\beta}^{n+1}\Big) \tag{4.2}$$

where

$$M_{\alpha\beta}^{0}+M_{\alpha\beta}^{1}=M_{\alpha\beta}.$$

To determine the iteration parameter ω_α for this scheme, we will study the symbol of the scheme. The symbol for the scheme (4.2) is defined as

$$s(Q_\alpha, \xi) = \frac{\omega_\alpha^{-1} + \sum_{|\beta| \leqq k} M_{\alpha\beta}^0 e^{i\xi\cdot\beta}}{\omega_\alpha^{-1} - \sum_{|\beta| \leqq k} M_{\alpha\beta}^1 e^{i\xi\cdot\beta}}$$

where $\xi = (\xi_1, \xi_2)$ with $|\xi_i| \leqq \pi$. The iteration parameter ω_α is chosen so that

$$|s(Q_\alpha, \xi)| \leqq 1 \quad \text{for } |\xi_i| \leqq \pi. \tag{4.3}$$

The definition of the symbol of the iteration scheme is in accordance with the theory of pseudo-translation operators developed by Vaillancourt [8]. The condition (4.3) is motivated by the Lax–Nirenberg theorem (Lax–Nirenberg [3], Vaillancourt [9]), which guarantees stability for the iteration scheme when applied for $\bar{\Omega} = \mathbb{R}^2$ and considered as an initial value problem. We refer the reader to Richtmyer and Morton [5, p. 45] for the definition of stability for initial value problems. We point out that while such stability is necessary for the convergence of the scheme, it is not sufficient, in general, to guarantee the convergence. As shown by the computational examples of § 6, the set of values of ω_α determined by (4.3) can give very good convergence rates for large Reynolds numbers.

For the remainder of the paper we make the following ellipticity assumption.

Assumption 4.1. The difference approximation (4.1) satisfies

$$-\operatorname{Re} \sum_{|\beta| \leqq k} M_{\alpha\beta} e^{i\xi\cdot\beta} \geqq c(|\xi_1|^2 + |\xi_2|^2)$$

for $|\xi_i| \leqq \pi$ and some positive constant c.

We now obtain an expression for ω_α. Define the symbols

$$m^1 = m^1(Q, \xi) = \sum_{|\beta| \leqq k} M_{\alpha\beta}^1 e^{i\xi\cdot\beta},$$

$$m^0 = m^0(Q, \xi) = \sum_{|\beta| \leqq k} M_{\alpha\beta}^0 e^{i\xi\cdot\beta},$$

and

$$m = m(Q, \xi) = m^0(Q, \xi) + m^1(Q, \xi).$$

From $|s| \leqq 1$ we have

$$|\omega^{-1} + m^0|^2 \leqq |\omega^{-1} - m^1|^2,$$

or equivalently

$$2 \operatorname{Re} m\omega^{-1} \leqq |m^1|^2 - |m^0|^2.$$

From the ellipticity assumption we have

$$\omega^{-1} \geqq \frac{|m^0|^2 - |m^1|^2}{-2 \operatorname{Re} m}.$$

Define ω_α^* by

$$(\omega_\alpha^*)^{-1} = \sup_{|\xi_i| \leqq \pi} \frac{|m^0|^2 - |m^1|^2}{-2 \operatorname{Re} m}. \tag{4.4}$$

For ω_α in the interval $[0, \omega_\alpha^*]$ the iterative method (4.2) will be stable. Moreover, as shown in § 6 the convergence rate of this scheme is optimal for ω_α equal to ω_α^*, at least for the examples considered there.

Note that if $|m^1|$ is larger than $|m^0|$ for all values of ξ, then the scheme is unconditionally stable in the sense that any positive value of ω_α will satisfy inequality (4.3). We will not consider such schemes here.

5. Computation of ω^* for special cases. We first compute ω^* for the case in which the iteration operator has a five-point stencil given by

$$(5.1)\quad u_{ij}^{n+1} = u_{ij}^n + \omega_{ij}\{a_{ij}u_{i+1j}^n + b_{ij}u_{i-1j}^{n+1} + c_{ij}u_{ij+1}^n + d_{ij}u_{ij-1}^{n+1} - (a_{ij}+b_{ij}+c_{ij}+d_{ij})u_{ij}^n\}.$$

Equation (5.1) is of the same form as (3.3). As in the previous section we have

$$m^0 = a\,e^{i\theta} + c\,e^{i\phi} - (a+b+c+d),$$
$$m^1 = b\,e^{-i\theta} + d\,e^{-i\phi},$$

where we have dropped the subscripts (i, j) and $\xi = (\theta, \phi)$. Note that Assumption 4.1 is satisfied when $a+b$ and $c+d$ are positive,

$$-\mathrm{Re}\ m = -\mathrm{Re}\ (m^0+m^1) = (a+b)(1-\cos\theta)+(c+d)(1-\cos\phi)$$
$$= 2(a+b)\sin^2\tfrac{1}{2}\theta + 2(c+d)\sin^2\tfrac{1}{2}\phi.$$

We will use formula (4.4) to compute ω^*.

$$|m^0|^2 - |m^1|^2 = (a+b+c+d)(-\mathrm{Re}\ m) + 2(a+b)(a-b-c+d)\sin^2\tfrac{1}{2}\theta$$
$$+2(c+d)(c-d-a+b)\sin^2\tfrac{1}{2}\phi$$
$$+4((a+b)(c-d)+(c+d)(a-b))\sin\tfrac{1}{2}\theta\sin\tfrac{1}{2}\phi\cos\tfrac{1}{2}(\theta-\phi).$$

Let

$$A = 2(a+b)(a-b-c+d),$$
$$B = 2((a+b)(c-d)+(c+d)(a-b)),$$

and

$$C = 2(c+d)(c-d-a+b).$$

Then

$$|m^0|^2 - |m^1|^2 \leqq (a+b+c+d)(-\mathrm{Re}\ m)$$
$$+(A+|B|r)\sin^2\tfrac{1}{2}\theta + (C+|B|r^{-1})\sin^2\tfrac{1}{2}\phi,$$

where r is an arbitrary positive number. If r is chosen so that

$$\frac{A+|B|r}{a+b} = \frac{C+|B|r^{-1}}{c+d},$$

then

$$|m^0|^2 - |m^1|^2 \leqq (a+b+c+d)(-\mathrm{Re}\ m)$$
$$+\sqrt{(a+b+c+d)((a-b)^2/(a+b)+(c-d)^2/(c+d))}\,(-\mathrm{Re}\ m).$$

This implies that

$$(\omega^*)^{-1} \leqq \frac{a+b+c+d}{2} + \frac{1}{2}\sqrt{(a+b+c+d)((a-b)^2/(a+b)+(c-d)^2/(c+d))}.$$

Moreover the above inequalities are sharp as can be seen by taking

$$\theta = \phi/r \to 0.$$

Therefore

$$\omega^* = \frac{2}{a+b+c+d+\sqrt{(a+b+c+d)((a-b)^2/(a+b)+(c-d)^2/(c+d))}}. \tag{5.2}$$

The reader is reminded that the coefficients a, b, c, and d are all variable functions of the grid points and therefore ω^* is also a function of the grid point.

We now consider the iterative method obtained from the checkerboard ordering. We will show that in this case ω^* is also given by formula (5.2). Analogous to (5.1) we have

$$u_{ij}^{n+1} = u_{ij}^n + \omega_{ij}\{a_{ij}u_{i+1j}^{n+\varepsilon} + b_{ij}u_{i-1j}^{n+\varepsilon} + c_{ij}u_{ij+1}^{n+\varepsilon} + d_{ij}u_{ij-1}^{n+\varepsilon} - (a_{ij}+b_{ij}+c_{ij}+d_{ij})u_{ij}^n\} \tag{5.3}$$

where $\varepsilon = 0$ for $i+j$ even and $\varepsilon = 1$ for $i+j$ odd. To obtain the expression for ω_{ij}^* one sets

$$u_{kl}^n = z^{2n+\varepsilon} e^{ik\theta} e^{il\phi},$$

obtaining the equation

$$\frac{z}{\omega} - \frac{1}{z}\left(\frac{1}{\omega} - (a+b+c+d)\right) = (a+b)\cos\theta + (c+d)\cos\phi + i(a-b)\sin\theta + i(c-d)\sin\phi. \tag{5.4}$$

The value of ω^* is determined by the requirement that for ω less than or equal to ω^* the modulus of z is less than unity. Rather than to proceed with this calculation, we use an alternative approach which is to notice that if for the natural ordering one sets

$$u_{kl}^n = z^{2n+k+l} e^{ik\theta} e^{il\phi}$$

then the resulting formula for z is the same as (5.4). This shows that the expression for ω^* is the same for both orderings.

For elliptic equations containing mixed derivatives, i.e., $B \neq 0$ in (1.1), an estimate of ω^* can be given. Consider the iterative method whose formula is

$$\begin{aligned} u_{ij}^{n+1} = u_{ij}^n + \omega_{ij}\{&A_{ij}(u_{i+1j}^n - 2u_{ij}^n + u_{i-1j}^{n+1}) + B_{ij}(u_{i+1j+1}^n - u_{i+1j-1}^{n+1} - u_{i-1j+1}^n + u_{i-1j-1}^{n+1}) \\ &+ C_{ij}(u_{ij+1}^n - 2u_{ij}^n + u_{ij-1}^{n+1}) + D_{ij}(u_{i+1j}^n - u_{i-1j}^{n+1}) + E_{ij}(u_{ij+1}^n - u_{ij-1}^{n+1})\}, \end{aligned}$$

where A_{ij}, B_{ij}, etc., are essentially the same as the coefficients in (1.1) except for factors of Δx and Δy. For this iterative scheme an estimate for ω^* is given by

$$(\omega^*)^{-1} \leqq \tfrac{1}{2}\left\{A + C + D + E + \frac{2(1+\sqrt{2})(A|E| + C|D|) + 4|B|(2\sqrt{2}|D| + |D+E|)}{A + C - \sqrt{(A-C)^2 + 16B^2}}\right\}.$$

The estimate of ω^* given by the right-hand side of the above inequality has been used to numerically solve the elliptic equations that result from the grid generation technique of Thompson et al. [7]. In the case that the grid has a high degree of stretching, the equations have a large Reynolds number. Although no results on this are

presented in this paper, the algorithm using this estimate has performed very well in many computations.

6. Computational results. The SRR method presented in this paper has been tested on the differential equation

$$u_{xx}+u_{yy}+Ru_x=0, \qquad 0\leqq x, y\leqq 1,$$

with several boundary conditions. The coordinates were transformed by the mappings

$$\begin{aligned} x &= x_0' q+(1-x_0')q^4, \\ y &= \tfrac{1}{2}+(1-7c)(p-\tfrac{1}{2})+40c(p-\tfrac{1}{2})^3-48c(p-\tfrac{1}{2})^5, \end{aligned} \tag{6.1}$$

where x_0' and c are parameters. Note that

$$x_0'=\frac{dx}{dq}(0)$$

and

$$y_0'=\frac{dy}{dp}(0)=\frac{dy}{dp}(1)=1+8c,$$

and also

$$\frac{d^2x}{dq^2}(0)=\frac{d^3x}{dq^3}(0)=\frac{d^2y}{dp^2}(0)=\frac{d^2y}{dp^2}(1)=0.$$

Employing a uniform $(N+1)\times(N+1)$ grid in the (q, p) plane, we find the difference approximation is given by

$$\begin{aligned} &q_i'(q_{i+1/2}'(u_{i+1j}-u_{ij})-q_{i-1/2}'(u_{ij}-u_{i-1j}))h^{-2} \\ &\quad +p_j'(p_{j+1/2}'(u_{ij+1}-u_{ij})-p_{j-1/2}'(u_{ij}-u_{ij-1}))h^{-2}+\tfrac{1}{2}Rq_i'(u_{i+1j}-u_{i-1j})h^{-1}=0 \end{aligned}$$

for $1<i, j<N+1$, where $h=N^{-1}$.

Problem 1. The first problem has the boundary condition

$$u(x, y)=|x-y|$$

for (x, y) on the boundary. An exact solution is not known for this problem; however, the solution does satisfy

$$u(x, 1-y)=1-u(x, y)$$

and for R large and $\delta<y<1-\delta$

$$u(x, y)\simeq y\,e^{-xR}+(1-e^{-xR})(1-y).$$

In addition to the boundary layer at $x=0$, the solution to this problem has appreciable gradients along the boundaries $y=0$ and $y=1$.

Problem 2. The second problem has as its solution

$$u(x, y)=\left(y(1-y)-\frac{2x}{R}\right)e^{-Rx},$$

and the boundary conditions specify that u agrees with this solution on the boundary.

The concern in this paper is with the convergence properties of the method, and not so much with the accuracy of the results. For any particular problem the accuracy of the results depend primarily on having enough grid points to resolve the boundary layer and

having a coordinate transformation which places the grid points properly. The cell Reynolds numbers are a measure of the suitability of the placement of the grid points. A second consideration is the criteria to terminate the iterative method. In the following examples the iterative procedure was stopped when

$$\frac{\|u^n - u^{n-1}\|_2}{\|u^n\|_2} \leqq 10^{-4}. \tag{6.2}$$

The l^2 norm in the above is given by

$$\|f\|_2 = \left(h^2 \sum_{X_\alpha \in \bar{\Omega}} |f_\alpha|^2\right)^{1/2}.$$

The convergence criterion of 10^{-4} in inequality (6.2) was chosen because it was small enough to achieve satisfactory answers and yet large enough so that it was economically feasible to make the large number of runs required for testing the algorithm.

In Tables 1 and 2 are shown the results of solving Problems 1 and 2, respectively, by both the checkerboard and natural orderings for different values of the grid size N, the Reynolds number R, and the coordinate transformations. In these tables the value of ω_α was always taken to be ω_α^* as given by (5.2).

Notice from Tables 1 and 2 that for a given grid size, coordinate transformation, and ordering, the number of iterations required for convergence is essentially independent of R. However, when the cell Reynolds number at $x = 0$ is near or above 2, the convergence rate becomes poorer and the solution itself becomes highly oscillatory.

Included at the end of Table 1 are calculations for the equation

$$u_{xx} + u_{yy} + Ru_x + \delta Ru = 0$$

for $\delta = 0.5$ and $\delta = 1.0$. The value of ω_α was the same as derived in § 5 for the case $\delta = 0$. The computations indicate that stability is maintained for lower order terms of the same order as the first order terms.

Table 3 shows the effect of taking ω_α as a multiple of ω_α^* for the case when $N = 40$ and $R = 40{,}000$. The number of iterations required for convergence was least for $\omega_\alpha = \omega_\alpha^*$ and for ω_α larger than ω_α^* the method does not converge at all. Similar results were observed for other values of N and R but are not displayed.

In Table 4 is shown the relationship between the number of grid points along one side of the grid, N, and the number of iterations required for convergence. For both the checkerboard ordering and natural ordering, the number of iterations is proportional to N for larger values of N. This shows that ρ, the radius of convergence of the iterative scheme, satisfies

$$\rho = 1 - C/N + o(N^{-1})$$

and, from the earlier comments, C is independent of R. A similar formula holds for SOR when the iteration parameter is chosen properly (Young [12]), and this is an indication of the efficiency of the method.

More specifically, we have from the results of Table 4 that for the checkerboard ordering the radius of convergence satisfies

$$\rho \simeq 1 - \frac{5.8}{N}$$

and for natural ordering

$$\rho \simeq 1 - \frac{4.6}{N}.$$

By comparison, using SOR to solve Laplace's equation on the unit square with a uniform grid the radius of convergence satisfies

$$\rho \simeq 1 - \frac{2\pi}{N}.$$

This shows that using SRR for elliptic equations with large Reynolds numbers is about as efficient as using SOR for elliptic equations with very low Reynolds numbers.

TABLE 1
Results for Problem 1.

Grid parameters	Reynolds No. (thousands)	Iterations checkerboard ordering	Iterations natural ordering	Cell Reynolds No. at $x = 0$
$N = 40$	6	70	86	.15
$x_0' = 10^{-3}$	10	70	87	.25
$y_0' = 0$	30	71	89	.75
	60	72	90	1.5
	80	72	90	2.0
	90	74	122	2.25
	100	108	130	2.5
$N = 40$	80	107	120	.20
$x_0' = 10^{-4}$	100	109	125	.25
$y_0' = 0$	300	102	119	.75
	600	115	130	1.5
$N = 40$	10	94	109	.25
$x_0' = 10^{-3}$	30	101	118	.75
$y_0' = .5$	60	103	120	1.5
	80	104	121	2.0
Reversed natural ordering	10		81	.25
	30		85	.75
	60		87	1.5
	80		92	2.0
$N = 80$	100	130	165	1.3
$x_0' = 10^{-3}$	130	130	165	1.6
$y_0' = 0$	160	130	165	2.0
	180	131	165	2.3
$N = 40$ $\delta = .5$	10	101		.25
$x_0' = 10^{-3}$	30	112		.75
$y_0' = .5$	60	115		1.5
	80	116		2.0
$\delta = 1.$	10	112		.25
	30	128		.75
	60	132		1.5

TABLE 2
Results for Problem 2.

Grid parameters	Reynolds No. (thousands)	Iterations checkerboard ordering	Iterations natural ordering	Cell Reynolds No. at $x = 0$
$N = 40$	6	47	46	.15
$x'_0 = 10^{-3}$	10	76	73	.25
$y'_0 = 0$	30	75	73	.75
	40	67	58	1.0
	60	80	76	1.5
	80	93	92	2.0
	90	98	97	2.3
	100	103	99	2.5
$N = 40$	80	89	85	.20
$x'_0 = 10^{-4}$	100	95	94	.25
$y'_0 = 0$	300	92	91	.75
	600	153	149	1.5

TABLE 3
Iterations for ω as a multiple of ω^* in Problem 1.

$R = 40{,}000, \quad N = 40, \quad x'_0 = 10^{-3}, \quad y'_0 = 0, \quad \omega_\alpha = \eta\omega^*_\alpha$

η	Checkerboard ordering	Natural ordering
.80	89	105
.90	79	97
.95	75	93
1.00	72	89
1.01	>200	>200
1.05	diverged	diverged

TABLE 4
Iterations as a function of N in Problem 1.

$R = 40{,}000, \quad x'_0 = 10^{-3}, \quad y'_0 = 0, \quad \omega_\alpha = \omega^*_\alpha$

N	Checkerboard ordering		Natural ordering	
	Iterations	Iterations/N	Iterations	Iterations/N
20	64	3.2	72	3.6
40	72	1.8	89	2.2
60	101	1.7	127	2.1
80	130	1.6	163	2.0
100	155	1.6	202	2.0

7. Conclusion. The SRR method introduced in this paper is a stable, efficient algorithm for the numerical solution of elliptic equations with large Reynolds number. The formula for the iteration parameter given in § 5 for the five-point schemes gives convergence rates that are essentially independent of the Reynolds number over a wide range of values.

Acknowledgment. The author wishes to thank the referees for their helpful suggestions and comments.

REFERENCES

[1] I. CHRISTIE, D. F. GRIFFITH, A. R. MITCHELL AND O. C. ZIENKIEWICZ, *Finite element methods for second order differential equations with significant first derivatives*, Internat. J. Numer. Methods Engrg., 10 (1976), pp. 1389–1396.

[2] P. CONCUS AND G. H. GOLUB, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, Proceedings of the 2nd Internat. Symp. on Computing Methods in Applied Sciences and Engineering, (IRIA), Paris, December 1975; Lecture Notes in Economics and Mathematical Systems, Vol. 134, R. Glowinski and J. L. Lions, eds., Springer-Verlag, Berlin, 1976.

[3] P. D. LAX AND L. NIRENBERG, *On stability for difference schemes; a sharp form of Gårding's inequality*, Comm. Pure Appl. Math., 19 (1966), pp. 473–492.

[4] A. H. NAYFEH, *Perturbation Methods*, John Wiley, New York, 1973.

[5] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Interscience, New York, 1967.

[6] P. J. ROACHE, *Computational Fluid Dynamics*, Hermosa Press, Albuquerque, NM, 1972.

[7] J. F. THOMPSON, F. C. THAMES AND C. W. MASTIN, *Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies*, J. Computational Phys., 15 (1974), pp. 299–319.

[8] R. VAILLANCOURT, *Pseudo-translation operators*, thesis, New York University, New York, 1969.

[9] ———, *A simple proof of Lax–Nirenberg theorems*, Comm. Pure Appl. Math., 23 (1970), pp. 151–163.

[10] M. VAN DYKE, *Perturbation Methods in Fluid Mechanics*, Academic Press, New York, 1964.

[11] O. WIDLUND, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

[12] D. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0007 $01.00/0

ALTERNATING DIRECTION IMPLICIT METHODS FOR PARABOLIC EQUATIONS WITH A MIXED DERIVATIVE*

RICHARD M. BEAM† AND R. F. WARMING†

Abstract. Alternating direction implicit (ADI) schemes for two-dimensional parabolic equations with a mixed derivative are constructed by using the class of all A_0-stable linear two-step methods in conjunction with the method of approximate factorization. The mixed derivative is treated with an explicit two-step method which is compatible with an implicit A_0-stable method. The parameter space for which the resulting ADI schemes are second-order accurate and unconditionally stable is determined. Some numerical examples are given.

Key words. ADI methods, implicit methods, linear multistep methods, mixed derivatives, parabolic equations

CONTENTS

1. Introduction. When an alternating direction implicit (ADI) method is applied to a parabolic equation, for example,

$$\frac{\partial u(x, y, t)}{\partial t} = Lu(x, y, t), \tag{1.1a}$$

where

$$L = a(x, y, t)\frac{\partial^2}{\partial x^2} + b(x, y, t)\frac{\partial^2}{\partial x\,\partial y} + c(x, y, t)\frac{\partial^2}{\partial y^2}, \tag{1.1b}$$

$$a, c > 0, \qquad b^2 < 4ac, \tag{1.1c, d}$$

it reduces the computational problem to a sequence of one-dimensional (matrix) inversion problems. If the mixed derivative $\partial^2/\partial x\,\partial y$ of the operator L were absent ($b = 0$), an ADI method would reduce the operator $(1 - L)$ to the product of two one-dimensional spatial operators. In the method of Douglas and Gunn [9], the mixed derivative is kept implicit and their scheme requires four inversions; that is, the

* Received by the editors April 24, 1979.

† Computational Fluid Dynamics Branch, Ames Research Center, National Aeronautics and Space Administration, Moffett Field, California 94035.

operator $(1-L)$ is reduced to four one-dimensional operators. The two additional inversions are the direct result of keeping the mixed derivative implicit. Simpler and more efficient schemes can be obtained if the mixed derivative is evaluated explicitly. Perhaps not so obvious is the stability of these simpler schemes. In fact, it is rather surprising that one can develop unconditionally stable algorithms for (1.1) by computing the mixed derivative explicitly and the derivatives $\partial^2/\partial x^2$ and $\partial^2/\partial y^2$ implicitly. McKee and Mitchell [15] surveyed two-level, first-order accurate (in time) schemes and devised a new first-order accurate, unconditionally stable scheme for (1.1). Iyengar and Jain [12] generalized the method of McKee and Mitchell and presented a three-level, second-order accurate scheme for (1.1); however, the implementation of the scheme is complicated by the explicit computation of fourth differences although the original partial differential equation (1.1) contains only second derivatives.

The present authors constructed a second-order accurate ADI algorithm for the compressible Navier–Stokes equations [1]. When the algorithm is applied to the model equation (1.1), it leads to a simple three-level unconditionally stable scheme which is easy to implement. In a recent paper [20] we combined A-stable linear multistep methods (LMMs) and approximate factorization to construct a large class of multilevel unconditionally stable ADI schemes for a model partial differential equation with both convective (hyperbolic) and diffusive (parabolic) terms; that is,

$$L = a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2} - c_1\frac{\partial}{\partial x} - c_2\frac{\partial}{\partial y},$$

where a, b, c satisfy (1.1c, d) and c_1, c_2 are real constants. The general formulation of [20] is second-order accurate if $b = 0$ but first-order accurate in time if the mixed derivative is included. In both [1] and [20], the mixed derivative is treated explicitly. The purpose of the present paper is to modify and combine the algorithms of [1] and [20] to obtain a general, second-order accurate, unconditionally stable algorithm for the model parabolic equation (1.1). In a companion paper [21] we apply the method to derive a noniterative ADI algorithm for a hyperbolic-parabolic system of nonlinear equations with mixed derivatives.

The development and analysis of numerical methods for ordinary differential equations (ODEs) are more advanced than that for partial differential equations (PDEs). Therefore, it seems plausible to capitalize on this fact by making use of known results from the theory of difference methods for ODEs to construct methods for PDEs. For example, the time differencing schemes used to construct implicit methods for PDEs are invariably LMMs although this fact is seldom noted. Since a great deal is known about the properties of LMMs (see, e.g., [6], [10]), one can use this information to advantage when attempting to construct schemes for PDEs. With these observations in mind we use § 2 as a review of the theory and notation for linear multistep methods including A-stability, A_0-stability, and one-leg methods. In addition, we introduce the notion of an LMM combining two different LMMs—one implicit and the other explicit. In § 3 we investigate the stability of an implicit method for (1.1) obtained by using an A_0-stable LMM as the time differencing method. We then modify the scheme by treating the mixed derivative explicitly and reinvestigate the stability properties. The method of approximate factorization is applied in § 4 to obtain an unconditionally stable, second-order accurate, multistep ADI scheme in the $\rho(E)$ formulation. In § 5 we construct a general approximate factorization method for all second-order, two-step schemes and discuss the parameter space for unconditional stability. Details of the stability analyses are given in the Appendices A and B. In § 6 a simple modification is given for the case of

time-dependent coefficients a, b, c. Numerical examples are given in § 7 and some concluding remarks in § 8.

2. Preliminaries. In this section we briefly review the theory of linear multistep methods (LMMs) and the related one-leg (multistep) methods. In addition, we introduce combined LMMs.

2.1. Linear multistep methods. A linear k-step method for the first-order ordinary differential equation

$$\frac{du}{dt} = f(u, t), \qquad u(0) = u_0, \tag{2.1}$$

is defined by the difference equation

$$\rho(E)u^n = \Delta t\, \sigma(E) f^n, \tag{2.2}$$

where ρ and σ are the generating polynomials,

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \qquad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j, \tag{2.3a, b}$$

and E is the shift operator, that is,

$$Eu^n = u^{n+1}. \tag{2.4}$$

In (2.2), u^n is the numerical solution at the point $t = t^n = n\Delta t$, Δt is the time step, and $f^n = f(u^n, t^n)$. The method is explicit if $\beta_k = 0$ and implicit otherwise. Consistency and normalization are expressed by the relations:

$$\sum_{j=0}^{k} \alpha_j = 0, \qquad \sum_{j=0}^{k} j\alpha_j = \sum_{j=0}^{k} \beta_j = 1. \tag{2.5a, b, c}$$

As an example of an LMM, the most general consistent two-step method (i.e., $k = 2$ in (2.3)) can be written as

$$(1+\xi)u^{n+2} - (1+2\xi)u^{n+1} + \xi u^n = \Delta t[\theta f^{n+2} + (1-\theta+\phi)f^{n+1} - \phi f^n] \tag{2.6}$$

where (θ, ξ, ϕ) are arbitrary real numbers. The operators $\rho(E)$ and $\sigma(E)$ are

$$\rho(E) = (1+\xi)E^2 - (1+2\xi)E + \xi, \tag{2.7a}$$

$$\sigma(E) = \theta E^2 + (1-\theta+\phi)E - \phi. \tag{2.7b}$$

For the class of all two-step methods that are at least second-order accurate, the parameters (θ, ξ, ϕ) are related by

$$\phi = \xi - \theta + \tfrac{1}{2}, \tag{2.8}$$

and consequently, $\sigma(E)$ can be rewritten in terms of the two parameters (θ, ξ) as

$$\sigma(E) = \theta E^2 + (\xi - 2\theta + \tfrac{3}{2})E - (\xi - \theta + \tfrac{1}{2}). \tag{2.9}$$

Some well-known implicit second-order methods and their corresponding values (θ, ξ, ϕ) are listed in Table 1. Linear one-step methods are a subclass of (2.6) obtained by setting $\xi = \phi = 0$:

$$u^{n+1} - u^n = \Delta t[\theta f^{n+1} + (1-\theta) f^n], \tag{2.10}$$

where we have shifted the time index down by one. The trapezoidal formula ($\theta = \frac{1}{2}$ in (2.10))

$$u^{n+1} - u^n = \frac{\Delta t}{2}(f^{n+1} + f^n) \tag{2.11}$$

is the only second-order accurate one-step method. When the trapezoidal formula is applied to a parabolic equation, the resulting algorithm is usually called Crank–Nicolson.

TABLE 1
Partial list of second-order two-step methods.

θ	ξ	ϕ	Method	Symbol in Fig. 1
$\frac{1}{2}$	0	0	One-step trapezoidal formula	■
1	$\frac{1}{2}$	0	Backward differentiation	●
$\frac{1}{3}$	$-\frac{1}{2}$	$-\frac{1}{3}$	Lees type [14]	▼
$\frac{3}{4}$	0	$-\frac{1}{4}$	Adams type [17]	◆
$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	Two-step trapezoidal formula	▲

The linear stability of an LMM is analyzed by applying it to the linear test equation

$$\frac{du}{dt} = \lambda u, \tag{2.12}$$

where λ is a complex constant. The stability is determined by the location of the roots of the characteristic equation,

$$\rho(\zeta) - \lambda\, \Delta t\, \sigma(\zeta) = 0, \tag{2.13}$$

relative to the unit circle in the complex plane. The stability region of an LMM consists of the set of all values of $\lambda\ \Delta t$ for which the characteristic equation (2.13) satisfies the root condition; that is, its roots ζ_l all satisfy $|\zeta_l| \leqq 1$ and the roots of unit modulus are simple. An LMM is said to be A-stable if its stability region contains all of the left half of the complex $\lambda\ \Delta t$ plane including the imaginary axis [4]. A simple test for A-stability can be formulated in terms of positive real functions [7]. By applying the test to the linear two-step method (2.6), one finds that the method is A-stable if and only if

$$\theta \geqq \phi + \tfrac{1}{2}, \tag{2.14a}$$

$$\xi \geqq -\tfrac{1}{2}, \tag{2.14b}$$

$$\xi \leqq \theta + \phi - \tfrac{1}{2}. \tag{2.14c}$$

An LMM is said to be A_0-stable if the region of stability contains the negative real axis of the complex $\lambda\ \Delta t$ plane, that is, the interval $(-\infty, 0]$. Again, by applying the theory of positive real functions, one finds that LMM (2.6) is A_0-stable if and only if

$$\theta \geqq \phi + \tfrac{1}{2}, \tag{2.15a}$$

$$\xi \geqq -\tfrac{1}{2}, \tag{2.15b}$$

$$0 \leqq \theta + \phi. \tag{2.15c}$$

For first-order accurate schemes, the inequality (2.15c) is less stringent than (2.14c). This is not surprising since A_0-stability is a weaker requirement than A-stability. However, for the class of all second-order methods, the parameters (θ, ξ, ϕ) are related by (2.8) and both sets of inequalities (2.14) and (2.15) reduce to

$$\xi \leqq 2\theta - 1, \qquad \xi \geqq -\tfrac{1}{2}. \tag{2.16a, b}$$

The parameter space (θ, ξ) for which the class of two-step, second-order methods is A-stable, happens to coincide with the parameter space for which the class is A_0-stable and is shown by the shaded region of Fig. 1. The methods listed in Table 1 and indicated by the symbols in Fig. 1 are both A_0- and A-stable.

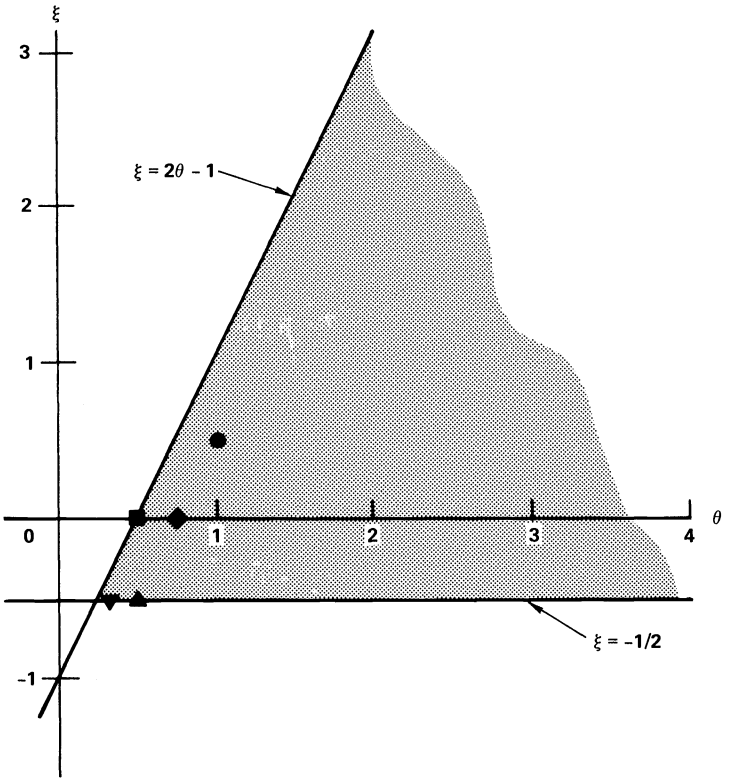


FIG. 1. *A_0- and A-stable domain of the parameters (θ, ξ) for the class of all second-order two-step methods. Symbols denote methods listed in Table* 1.

Dahlquist has proved that the order of accuracy of an A-stable LMM cannot exceed two [4]. On the other hand, Cryer [3] has proved there exist A_0-stable LMMs of arbitrarily high order. Since the eigenvalue associated with the parabolic equation (1.1) is real and negative, the application of an A_0-stable LMM will yield an unconditionally stable scheme, that is, no stability restriction on the size of Δt.

In this paper we restrict our attention to second-order accurate LMMs. This limitation is motivated by two practical considerations. First, conventional techniques for constructing alternating direction implicit (all A-stable and A_0-stable LMMs are implicit [4], [3]) schemes generally impose a second-order-temporal accuracy limitation independent of the accuracy of the LMM chosen as the time differencing approximation. The reason for this will become apparent in § 4. In principle, by altering

conventional procedures, ADI schemes of temporal order greater than 2 can be constructed for PDEs where A_0-stable LMMs are appropriate; however, the unconditional stability of the higher order ADI schemes is an open question. The second practical consideration is computer storage. Even if the question of unconditional stability is answered in the affirmative, the scheme must be at least a three-step method since the two-step method (2.6) contains no A_0-stable third-order subclass.

2.2 One-leg (multistep) methods. A class of methods closely related to LMMs is the one-leg (multistep) methods. The one-leg (k-step) method [5] corresponding to (2.2) is

$$\rho(E)\hat{u}^n = \Delta t f(\sigma(E)\hat{u}^n, \sigma(E)t^n) \tag{2.17}$$

where $\hat{u}^n$ denotes the one-leg method solution. Formally, the one-leg method (2.17) can be obtained by shifting the operator $\sigma(E)$ inside the argument parenthesis of $f^n = f(u^n, t^n)$ in the linear multistep formula (2.2). As an example, the trapezoidal formula (2.11) is an LMM and the implicit midpoint rule,

$$\hat{u}^{n+1} - \hat{u}^n = \Delta t f\left(\frac{\hat{u}^{n+1} + \hat{u}^n}{2}, t^n + \frac{\Delta t}{2}\right), \tag{2.18}$$

is the corresponding one-leg method. Dahlquist [5] has proved the following theorem relating solutions of a one-leg method and an LMM: Let $\{\hat{u}^n\}$ be a vector sequence that satisfies the one-leg difference equation (2.17) and set

$$u^n = \sigma(E)\hat{u}^n.$$

Then $\{u^n\}$ satisfies the LMM difference formula (2.2).

It should be noted that LMMs and one-leg methods are identical when applied to the linear test equation (2.12) and, consequently, the results of linear stability analysis are the same for both. However, one-leg methods simplify nonlinear stability analysis [5] and, in addition, they are more reliable than LMMs when used with rapidly varying integration step sizes [17]. As applied in this paper, the one-leg formulation makes it easy to construct an ADI scheme for the parabolic equation (1.1) with time-dependent coefficients.

2.3. Combined linear multistep methods. If the function $f(u, t)$ of the differential equation (2.1) is split into a sum, that is,

$$\frac{du}{dt} = f(u, t) = f_1(u, t) + f_2(u, t), \tag{2.19}$$

we can construct an integration formula by combining two different LMMs. For example,

$$\rho(E)u^n = \Delta t\, \sigma_1(E)f_1^n + \Delta t\, \sigma_2(E)f_2^n, \tag{2.20}$$

where the subscripts on σ_1 and σ_2 indicate that the coefficients β_j of the generating polynomial (2.3b) differ for the two functions f_1 and f_2. The analogous combined one-leg method (2.17) is

$$\rho(E)\hat{u}^n = \Delta t f_1(\sigma_1(E)\hat{u}^n, \sigma_1(E)t^n) + \Delta t f_2(\sigma_2(E)\hat{u}^n, \sigma_2(E)t^n). \tag{2.21}$$

3. Unconditionally stable schemes. In this section we examine the stability of an implicit method for the parabolic equation (1.1) where the time differencing approximation is an A_0-stable LMM. Next we modify the scheme by treating the mixed

derivative explicitly and determine the criteria for unconditional stability. Although not essential to the final goal of constructing unconditionally stable ADI schemes with the mixed derivative computed explicitly, this intermediate analysis does isolate the stability constraints due to the application of combined LMMs and those due to approximate factorization. Furthermore, the nonfactored scheme is formulated so that the ADI variant follows directly (§ 4).

Numerical methods for solving the parabolic equation (1.1) can be obtained by a direct application of the LMM (2.2). Since our interest is in constructing unconditionally stable schemes, we assume that the LMM is A_0-stable. By comparing (1.1) and (2.1), we identify

$$f(u) = Lu = \left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)u \tag{3.1}$$

where L is a linear differential operator. For simplicity we assume that the coefficients a, b, c are independent of time. The case of time-dependent coefficients is considered in § 6. Insertion of (3.1) into (2.2) yields

$$\rho(E)u^n = \Delta t\left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)\sigma(E)u^n. \tag{3.2}$$

To analyze the stability of (3.2) we assume a solution for the PDE (1.1) (with constant coefficients) of the form

$$u(x, y, t) = v(t)\, e^{i(\kappa_1 x + \kappa_2 y)} \tag{3.3}$$

where $v(t)$ is the Fourier coefficient and κ_1, κ_2 are the Fourier variables (wave numbers). Substitution of (3.3) into (1.1) yields an ODE for $v(t)$:

$$\frac{dv}{dt} = \lambda v, \tag{3.4a}$$

where

$$\lambda = -(a\kappa_1^2 + b\kappa_1\kappa_2 + ck_2^2). \tag{3.4b}$$

The quadratic form in the parenthesis of (3.4b) is positive definite if and only if the inequalities (1.1c, d) are satisfied. These constraints constitute the parabolicity condition of the PDE and ensure that the solution of (3.4) is damped with time. To complete the stability analysis of the PDE scheme (3.2), we need only consider the stability of the LMM (2.2) applied to (3.4) with $\lambda < 0$. Since we assumed that (2.2) is A_0-stable, the PDE scheme (3.2) is unconditionally stable. In practice, the spatial derivatives in (3.2) are replaced by appropriate difference quotients; however, as shown at the end of Appendix B, central spatial discretization does not alter the unconditional stability criteria obtained by assuming spatially continuous solutions.

For the one-step methods (2.10), the scheme (3.2) reduces to

$$u^{n+1} - u^n = \Delta t\left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)[\theta u^{n+1} + (1-\theta)u^n]. \tag{3.5}$$

With central spatial difference approximations, (3.5) is identical to a scheme suggested by Lax and Richtmyer [13]. Their paper appeared about the same time the original ADI methods were proposed by Peaceman and Rachford [18] and Douglas [8]. The first ADI methods [8], [18] did not include a mixed derivative term and its presence precludes the construction of an efficient ADI method. A simple way of circumventing

this difficulty is to treat the mixed derivative explicitly. One might expect, however, that this would have an adverse effect on the unconditional stability of the algorithm. This stability question is considered in the remainder of this section.

Consider the combined LMM (2.20) where σ_1 and σ_2 are defined as follows. Let (2.2) represent a second-order A_0-stable LMM and define

$$\sigma_1(E)=\sigma(E), \qquad \sigma_2(E)=\sigma_e(E), \tag{3.6a, b}$$

where $\sigma_e(E)$ is a second-order explicit LMM (i.e., $\beta_k=0$ in the generating polynomial (2.3b)) with the same generating polynomial $\rho(\zeta)$ as for the A_0-stable LMM. With these definitions, (2.20) becomes

$$\rho(E)u^n=\Delta t\,\sigma(E)f_1^n+\Delta t\,\sigma_e(E)f_2^n. \tag{3.7}$$

Henceforth, in reference to a combined implicit-explicit method such as (3.7), we refer to the LMM that defines $\rho(E)$ and $\sigma(E)$ as the generating LMM. The linear stability properties of (3.7) for second-order two-step methods are examined in Appendix A.

For didactic purposes in this section and practical reasons in the following section, we rewrite the LMM (3.7) as

$$\rho(E)u^n-\omega\,\Delta t\,\rho(E)f_1^n=\Delta t[\sigma(E)-\omega\rho(E)]f_1^n+\Delta t\,\sigma_e(E)f_2^n, \tag{3.8}$$

where

$$\omega=\beta_k/\alpha_k. \tag{3.9}$$

The parameter ω is defined so that the operator $\sigma(E)-\omega\rho(E)$ on the right-hand side of (3.8) is at least one degree lower than the operator $\rho(E)$ on the left-hand side. This can readily be seen by using the definitions (2.3) and writing out the highest degree term of the operator

$$\sigma(E)-\omega\rho(E)=\left(\beta_{k-1}-\frac{\beta_k}{\alpha_k}\alpha_{k-1}\right)E^{k-1}+\cdots. \tag{3.10}$$

Consequently, the right-hand side of (3.8) can be computed explicitly, that is, from known data when advancing the numerical solution from $n+k-1$ to $n+k$.

Finally, to apply the combined scheme (3.8) to the PDE (1.1), we split the linear differential operator of (3.1), i.e.,

$$f_1(u)=\left(a\frac{\partial^2}{\partial x^2}+c\frac{\partial^2}{\partial y^2}\right)u \quad \text{and} \quad f_2(u)=b\frac{\partial^2 u}{\partial x\,\partial y}. \tag{3.11}$$

By substituting (3.11) into (3.8), we obtain

$$\begin{aligned}&\left[1-\omega\,\Delta t\left(a\frac{\partial^2}{\partial x^2}+c\frac{\partial^2}{\partial y^2}\right)\right]\rho(E)u^n\\&\quad=\Delta t\left(a\frac{\partial^2}{\partial x^2}+c\frac{\partial^2}{\partial y^2}\right)[\sigma(E)-\omega\rho(E)]u^n+\Delta t\,b\frac{\partial^2}{\partial x\,\partial y}\sigma_e(E)u^n.\end{aligned} \tag{3.12}$$

This formula is implicit for u_{xx} and u_{yy} and explicit for u_{xy}.

Remark. The scheme (3.12) with the simplest evaluation of the right-hand side has $\sigma_e(E)$ given by

$$\sigma_e(E)=[\sigma(E)-\omega\rho(E)]u^n, \tag{3.13}$$

in which case the right-hand side of (3.12) is equal to

$$\Delta t\left(a\frac{\partial^2}{\partial x^2}+b\frac{\partial^2}{\partial x\,\partial y}+c\frac{\partial^2}{\partial y^2}\right)[\sigma(E)-\omega\rho(E)]u^n. \tag{3.14}$$

Unfortunately, the method with $\sigma_e(E)$ defined by (3.13) is only first-order accurate.

For the stability analysis of (3.12), we consider the second-order, two-step methods where $\rho(E)$ and $\sigma(E)$ are defined by (2.7a) and (2.9) and the explicit operator $\sigma_e(E)$ is obtained from (2.9) by setting $\theta = 0$,

$$\sigma_e(E) = (\xi+\tfrac{3}{2})E-(\xi+\tfrac{1}{2}). \tag{3.15}$$

The details of the stability analysis are given in Appendix A. Scheme (3.12) is found to be unconditionally stable for all values of a, b, c satisfying equalities (1.1c, d) if and only if

$$\xi \leqq \theta - 1, \qquad \xi \geqq -\tfrac{1}{2}. \tag{3.16a, b}$$

The values of the parameters (θ, ξ) satisfying these inequalities are shown by the shaded region of Fig. 2. Inequality (3.16a) is more restrictive than (2.16a) for the generating second-order, two-step method to be A_0-stable (see Fig. 1). Methods that fall in the region between the lines

$$\xi = 2\theta - 1 \quad \text{and} \quad \xi = \theta - 1 \tag{3.17}$$

and above $\xi = -\frac{1}{2}$ are not unconditionally stable for all values of the coefficient b satisfying inequality (1.1d). Note that, with the exception of the two-step trapezoidal formula, none of the methods listed in Table 1 falls in the shaded region of Fig. 2.

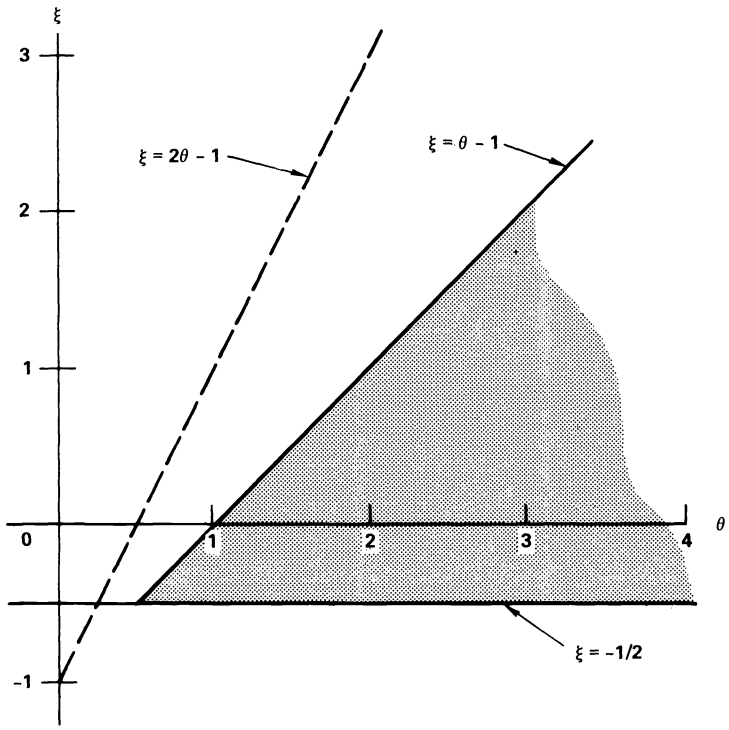


FIG. 2. *Unconditionally stable domain of the parameters* (θ, ξ) *for the unfactored scheme* (3.12) *with* $\rho(E)$, $\sigma(E)$, *and* $\sigma_e(E)$ *defined by* (2.7a), (2.9), *and* (3.15).

Remark. It is of interest to note that the explicit operator (3.15) can also be obtained from the implicit operator (2.9) applied to f^n,

$$\sigma(E)f^n = \theta f^{n+2} + (\xi - 2\theta + \tfrac{3}{2})f^{n+1} - (\xi - \theta + \tfrac{1}{2})f^n,$$

by using linear extrapolation, that is,

$$f^{n+2} = 2f^{n+1} - f^n + O(\Delta t^2),$$

to approximate f^{n+2}. Although the use of linear extrapolation might sound rather disreputable, the application of an explicit LMM does not.

In the following section we find the rather remarkable result that an approximate factorization of the left-hand side of (3.12) into a product of one-dimensional operators restores most of the parameter space (θ, ξ) for unconditional stability lost by the explicit treatment of the mixed derivative.

4. An ADI scheme: the $\rho(E)$ or Λ formulation. In the procedure suggested in [20] for designing ADI schemes, the implicit operator to be inverted is constructed so that the unknown variable to be determined at each time step is $\rho(E)u^n$. In the absence of a mixed derivative this choice ensures that approximate factorization into a product of one-dimensional operators does not upset either the temporal accuracy (second-order) or the unconditional stability of the scheme. In this section we construct an ADI scheme in the $\rho(E)$ formulation with the mixed derivative treated explicitly with second-order accuracy and examine the stability of the resulting algorithm.

If the spatial derivatives appearing in (3.12) are replaced by appropriate difference quotients, then one obtains in general an enormous linear system to solve for $\rho(E)u^n$. This difficulty can be overcome by an approximate factorization of the left-hand side of (3.12) which reduces the problem to a product of one-dimensional spatial operators; that is,

$$\begin{aligned}\left(1 - \omega\,\Delta t\, a\frac{\partial^2}{\partial x^2}\right)\left(1 - \omega\,\Delta t\, c\frac{\partial^2}{\partial y^2}\right)\rho(E)u^n \\ = \Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)[\sigma(E) - \omega\rho(E)]u^n + \Delta t\, b\frac{\partial^2}{\partial x\,\partial y}\sigma_e(E)u^n.\end{aligned} \tag{4.1}$$

Comparison of the left-hand sides of (4.1) and (3.12) shows that they differ by the cross-product term

$$\omega^2\,\Delta t^2\, a\frac{\partial^2}{\partial x^2}c\frac{\partial^2}{\partial y^2}\rho(E)u^n. \tag{4.2}$$

But by expanding $\rho(E)u^n$ in a Taylor series about u^n and using the consistency and normalization conditions (2.5a, b), one obtains

$$\rho(E)u^n = \Delta t\frac{\partial u^n}{\partial t} + O(\Delta t^{\alpha+1}), \qquad \alpha \geqq 1. \tag{4.3}$$

Consequently, the cross-product term (4.2)

$$\begin{aligned}\omega^2\,\Delta t^2\, a\frac{\partial^2}{\partial x^2}c\frac{\partial^2}{\partial y^2}\rho(E)u^n &= \omega^2\,\Delta t^3\, a\frac{\partial^2}{\partial x^2}c\frac{\partial^2}{\partial y^2}\frac{\partial}{\partial t}u^n + O(\Delta t^4) \\ &= O(\Delta t^3)\end{aligned} \tag{4.4}$$

is a third-order term and the formal accuracy of the scheme (3.12) is not upset by the approximate factorization (4.1).

In practice, the second-order, two-step methods defined by (2.7a) and (2.9) are of primary interest and in the remainder of this section we limit our attention to these methods. The explicit operator $\sigma_e(E)$ is given by (3.15). In numerical algorithms for partial differential equations it is conventional to use $n+1$ as the most advanced time level; hence, we multiply (4.1) by E^{-1}. The shifted difference operators for the second-order, two-step method can be written as

$$\Lambda u^n = E^{-1}\rho(E)u^n = [(1+\xi)E - (1+2\xi) + \xi E^{-1}]u^n, \tag{4.5a}$$

$$= [(1+\xi)\Delta - \xi\nabla]u^n, \tag{4.5b}$$

$$= (1+\xi)u^{n+1} - (1+2\xi)u^n + \xi u^{n-1}, \tag{4.5c}$$

$$E^{-1}\sigma(E)u^n = [\theta E + (\xi - 2\theta + \tfrac{3}{2}) - (\xi - \theta + \tfrac{1}{2})E^{-1}]u^n, \tag{4.6a}$$

$$= [1 + \theta\Delta + (\xi - \theta + \tfrac{1}{2})\nabla]u^n, \tag{4.6b}$$

$$E^{-1}\sigma_e(E)u^n = [(\xi + \tfrac{3}{2}) - (\xi + \tfrac{1}{2})E^{-1}]u^n, \tag{4.7a}$$

$$= [1 + (\xi + \tfrac{1}{2})\nabla]u^n, \tag{4.7b}$$

where the symbols Δ and ∇ are classical forward and backward difference operators defined by

$$\Delta u^n = u^{n+1} - u^n, \qquad \nabla u^n = u^n - u^{n-1}. \tag{4.8}$$

As a notational convenience, we have denoted the operator $E^{-1}\rho(E)$ by Λ. From (4.5) and (4.6) there follows

$$E^{-1}[\sigma(E) - \omega\rho(E)]u^n = [(\xi - \omega + \tfrac{3}{2}) - (\xi - \omega + \tfrac{1}{2})E^{-1}]u^n \tag{4.9a}$$

$$= [1 + (\xi - \omega + \tfrac{1}{2})\nabla]u^n, \tag{4.9b}$$

where

$$\omega = \frac{\theta}{1+\xi}. \tag{4.10}$$

The factored scheme (4.1) becomes

$$\begin{aligned}\left(1 - \omega\,\Delta t\, a\frac{\partial^2}{\partial x^2}\right)\left(1 - \omega\,\Delta t\, c\frac{\partial^2}{\partial y^2}\right)\Lambda u^n \\ = \Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)\left[1 + \left(\xi - \omega + \frac{1}{2}\right)\nabla\right]u^n + \Delta t\, b\frac{\partial^2}{\partial x\,\partial y}\left[1 + \left(\xi + \frac{1}{2}\right)\nabla\right]u^n.\end{aligned} \tag{4.11}$$

The computational sequence to implement the factored scheme (4.11) as an ADI method is not unique, but an obvious choice is

$$\left(1 - \omega\,\Delta t\, a\frac{\partial^2}{\partial x^2}\right)\Lambda u^* = \text{RHS (4.11)}, \tag{4.12a}$$

$$\left(1 - \omega\,\Delta t\, c\frac{\partial^2}{\partial y^2}\right)\Lambda u^n = \Lambda u^*, \tag{4.12b}$$

$$(1+\xi)u^{n+1} = \Lambda u^n + (1+2\xi)u^n - \xi u^{n-1}, \tag{4.12c}$$

where Λu^* is a dummy temporal difference (RHS stands for right-hand side).

Remark. If the first-order explicit method with $\sigma_e(E)$ defined by (3.13) were used rather than a second-order method, the right-hand side of (4.1) would be the same as (3.14). In this case, the right-hand side of (4.11) would be replaced by

$$\Delta t\left(a\frac{\partial^2}{\partial x^2}+b\frac{\partial^2}{\partial x\,\partial y}+c\frac{\partial^2}{\partial y^2}\right)\left[1+\left(\xi-\omega+\frac{1}{2}\right)\nabla\right]u^n \tag{4.13}$$

where we have used (4.9). Although the resulting scheme is first-order accurate in time for the mixed derivative, it is unconditionally stable for values of (θ, ξ) in the shaded region of Fig. 1. This result was established in [20, Appendix A] by constructing the scheme so that the characteristic equation (which determines the stability of the method) for the factored partial difference equation had the same form as the characteristic equation (2.13) for the ordinary differential equation.

The scheme (4.11) is second-order accurate in the mixed derivative but the characteristic equation no longer has the form (2.13). Consequently, the stability analysis must be redone in a less elegant manner than in [20] and is carried out in Appendix B. The stability analysis of Appendix B is for the general two-step ADI scheme developed in the following section. The Λ formulation (4.11) is a special case of the general two-step scheme and is found to be unconditionally stable for all values of a, b, c satisfying inequalities (1.1c, d) if and only if

$$\theta \geqq \frac{2(1+\xi)^2}{3+4\xi}, \qquad \xi \geqq -\frac{1}{2}. \tag{4.14a, b}$$

The parameter space (θ, ξ) satisfying these inequalities is shown by the shaded region of Fig. 3. The inequality (4.14a) is more stringent than (2.16a), and methods that fall in the

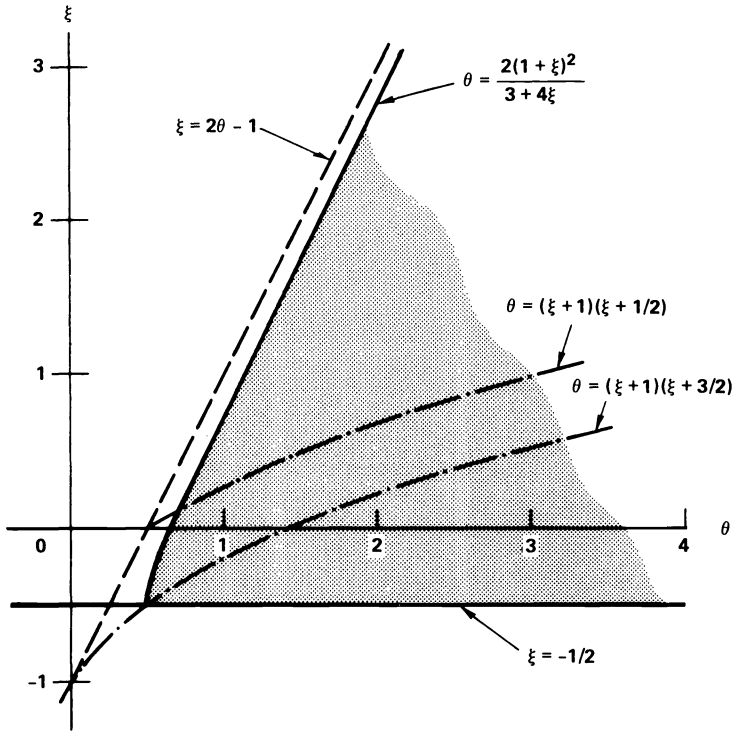


FIG. 3. *Unconditionally stable domain of the parameters* (θ, ξ) *for the factored* Λ *formulation* (4.11).

region between the curves

$$\xi = 2\theta - 1 \quad \text{and} \quad \theta = \frac{2(1+\xi)^2}{3+4\xi} \tag{4.15}$$

and above $\xi = -\frac{1}{2}$ are not unconditionally stable for all values of the coefficient b satisfying inequality (1.1d). This includes such popular schemes as the trapezoidal formula and Lees method (see, e.g., [2]). However, there remains a large class of unconditionally stable methods including, e.g., the backward differentiation formula (see Table 1).

5. A general two-step ADI scheme. In the ADI formulation described in the previous section it is essential that the unknown variable be at least a first difference to ensure that the approximate factorization does not degrade the second-order accuracy of the scheme; for example, the unknown variable $E^{-1}\rho(E)u^n = \Lambda u^n$ is an approximation to $\Delta t(\partial u/\partial t)$ (see (4.3)). The choice of unknown variable is not unique and in fact the most general form of a first difference using data from the three levels $n-1$, n, $n+1$ can be written as

$$\begin{aligned} u^{n+1} - (1+\alpha)u^n + \alpha u^{n-1} &= (\Delta - \alpha\nabla)u^n \\ &= (1-\alpha)\,\Delta t \frac{\partial u^n}{\partial t} + (1+\alpha)\frac{\Delta t^2}{2}\frac{\partial^2 u^n}{\partial t^2} + O(\Delta t^3), \end{aligned} \tag{5.1}$$

where Δ and ∇ are the forward and backward operators defined by (4.8) and α is an arbitrary real constant. The undivided difference (5.1) is a second difference if $\alpha = 1$ and a first difference otherwise. The most accurate first difference is obtained when $\alpha = -1$.

5.1. General formulation. A general formulation of two-step ADI schemes is obtained if we choose $(\Delta - \alpha\nabla)u^n$ as the unknown variable. We first rewrite the LMM (3.7) in a convenient form for the construction of an ADI scheme with $(\Delta - \alpha\nabla)u^n$ as the unknown variable. Multiplying (3.7) by E^{-1} and inserting the operators $E^{-1}\rho(E)$, $E^{-1}\sigma(E)$, and $E^{-1}\sigma_e(E)$ defined by (4.5b), (4.6b), and (4.7b), one can rewrite the resulting two-step method as

$$\Delta u^n - \omega\,\Delta t\,\Delta f_1^n = \frac{\Delta t}{1+\xi}\left[1 + \left(\xi - \theta + \frac{1}{2}\right)\nabla\right](f_1^n + f_2^n) + \omega\,\Delta t\,\nabla f_2^n + \frac{\xi}{1+\xi}\nabla u^n \tag{5.2}$$

After subtracting $\alpha\nabla u^n - \alpha\omega\,\Delta t\,\nabla f_1^n$ from both sides, we find

$$\begin{aligned} (\Delta - \alpha\nabla)u^n - \omega\,\Delta t(\Delta - \alpha\nabla)f_1^n &= \frac{\Delta t}{1+\xi}\left[1 + \left(\xi - \theta + \frac{1}{2}\right)\nabla\right](f_1^n + f_2^n) \\ &\quad + \omega\,\Delta t\,\nabla f_2^n + \alpha\omega\,\Delta t\,\nabla f_1^n + \left(\frac{\xi}{1+\xi} - \alpha\right)\nabla u^n. \end{aligned} \tag{5.3}$$

Using (3.11), one obtains

$$\begin{aligned} &\left[1 - \omega\,\Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)\right](\Delta - \alpha\nabla)u^n \\ &\quad = \frac{\Delta t}{1+\xi}\left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)\left[1 + \left(\xi - \theta + \frac{1}{2}\right)\nabla\right]u^n \\ &\quad\quad + \omega\,\Delta t\,b\frac{\partial^2}{\partial x\,\partial y}\nabla u^n + \alpha\omega\,\Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)\nabla u^n + \left(\frac{\xi}{1+\xi} - \alpha\right)\nabla u^n. \end{aligned} \tag{5.4}$$

The spatially factored form of (5.4) is

$$\left(1-\omega\,\Delta t\,a\frac{\partial^2}{\partial x^2}\right)\left(1-\omega\,\Delta t\,c\frac{\partial^2}{\partial y^2}\right)(\Delta-\alpha\nabla)u^n=\text{RHS(5.4)}, \tag{5.5}$$

which can be implemented as

$$\left(1-\omega\,\Delta t\,a\frac{\partial^2}{\partial x^2}\right)(\Delta-\alpha\nabla)u^*=\text{RHS(5.4)}, \tag{5.6a}$$

$$\left(1-\omega\,\Delta t\,c\frac{\partial^2}{\partial y^2}\right)(\Delta-\alpha\nabla)u^n=(\Delta-\alpha\nabla)u^*, \tag{5.6b}$$

$$u^{n+1}=(\Delta-\alpha\nabla)u^n+(1+\alpha)u^n-\alpha u^{n-1}. \tag{5.6c}$$

The stability analysis for the general factored scheme (5.5) is given in Appendix B. The scheme is found to be unconditionally stable for all values of a, b, c satisfying inequalities (1c, d) if and only if

$$\theta\geqq\frac{2(1+\xi)}{2+\sqrt{\dfrac{(1+\alpha)(1+2\xi)}{1+\xi}}},\qquad \xi\geqq-\frac{1}{2},\quad -1\leqq\alpha\leqq 1. \tag{5.7a, b, c}$$

The parameter space (θ, ξ) satisfying these inequalities is indicated in Fig. 4 for several values of α. For a given value of α, the stable range is to the right of the curve labeled with that value of α and above the curve $\xi=-\frac{1}{2}$. The extent of the (θ, ξ) parameter space for unconditional stability is a monotone increasing function of α in the range $[-1, 1]$ with the smallest region for $\alpha=-1$ and the largest for $\alpha=+1$. Inequality (5.7a) is more stringent than (2.16a) as is apparent in Fig. 4. Along the line $\xi=0$, inequality (5.7a) becomes

$$\theta\geqq\frac{2}{2+\sqrt{1+\alpha}},\qquad (\xi=0), \tag{5.8}$$

and the smallest allowed value for θ occurs when $\alpha=1$, in which case (5.8) becomes

$$\theta\geqq\frac{1}{1+\sqrt{1/2}}\simeq 0.586,\qquad (\xi=0). \tag{5.9}$$

Along the lower stability boundary $\xi=-\frac{1}{2}$, inequality (5.7a) becomes

$$\theta\geqq\tfrac{1}{2},\qquad (\xi=-\tfrac{1}{2}), \tag{5.10}$$

which is independent of the parameter α. As a consequence of inequalities (5.9) and (5.10), such popular generating LMMs as the trapezoidal formula $(\theta=\frac{1}{2}, \xi=0)$ and Lees method $(\theta=\frac{1}{3}, \xi=-\frac{1}{2})$ are not unconditionally stable for all values of the coefficient b satisfying inequality (1.1d).

If the parameter α is chosen to be $\xi/(1+\xi)$, scheme (5.5) reduces to the Λ formulation of § 4. This ADI method has the peculiar property that the unknown variable depends on the parameter ξ, that is, on the particular LMM chosen. The parameter space (θ, ξ) for which the Λ formulation is unconditionally stable is shown by the shaded region of Fig. 2 and the extent of the region is nearly as large as that for (5.5) with $\alpha=1$.

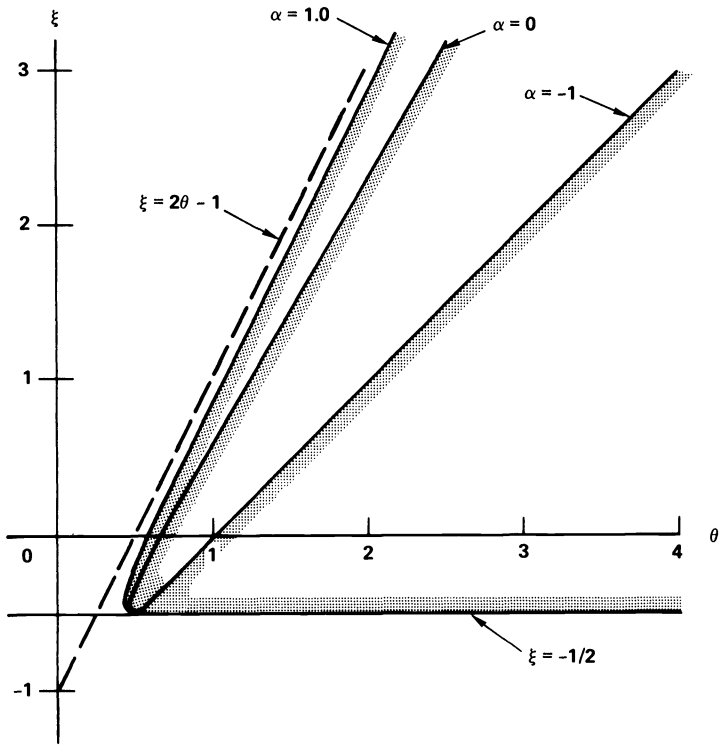


FIG. 4. *Unconditionally stable domain of the parameters* (θ, ξ) *for the factored general formulation* (5.5) *for several values of* α.

5.2. Special cases of general formulation. Various constant values of α in the range $[-1, 1]$ produce useful and interesting algorithms and we consider several in greater detail. The schemes are named according to the classical difference operator represented by (5.1) for the chosen values of α.

5.2a. The Δ formulation ($\alpha = 0$). If α is chosen to be zero, the general scheme (5.5) reduces to

$$\begin{aligned}\left(1-\omega\,\Delta t\,a\frac{\partial^2}{\partial x^2}\right)\left(1-\omega\,\Delta t\,c\frac{\partial^2}{\partial y^2}\right)\Delta u^n \\ =\frac{\Delta t}{1+\xi}\left(a\frac{\partial^2}{\partial x^2}+b\frac{\partial^2}{\partial x\,\partial y}+c\frac{\partial^2}{\partial y^2}\right)\left[1+\left(\xi-\theta+\frac{1}{2}\right)\nabla\right]u^n \\ +\omega\,\Delta t\,b\frac{\partial^2}{\partial x\,\partial y}\nabla u^n+\frac{\xi}{1+\xi}\nabla u^n,\end{aligned} \tag{5.11}$$

which we call the Δ formulation [1], [19] since Δu^n is the *unknown* variable. The parameter space for unconditional stability is given by the inequalities (5.7a, b) with $\alpha = 0$ and is shown graphically in Fig. 4.

5.2b. The δ^2 formulation ($\alpha = 1$). In the general ADI formulation (5.5) the unknown variable $(\Delta - \alpha\nabla)u^n$ is an approximation to $\Delta t(\partial u/\partial t)$ if $\alpha \neq 1$ (see (5.1)). A less natural choice for the unknown variable for a first-order (temporal) differential

equation is the second difference obtained when $\alpha = 1$. In this case, (5.1) becomes

$$(\Delta - \nabla)u^n = \delta^2 u^n = \Delta t^2 \frac{\partial^2 u^n}{\partial t^2} + O(\Delta t^4), \tag{5.12}$$

where the classical second difference operator δ^2 is defined by

$$\delta^2 u^n = u^{n+1} - 2u^n + u^{n-1}. \tag{5.13}$$

The possibility of using $\delta^2 u^n$ as the unknown variable does not arise for linear one-step methods (e.g., the trapezoidal formula (2.11)) since these methods only involve two time levels.

If α is set equal to one in the general two-step scheme (5.5), we obtain

$$\begin{aligned}\left(1 - \omega\,\Delta t\, a\frac{\partial^2}{\partial x^2}\right)\left(1 - \omega\,\Delta t\, c\frac{\partial^2}{\partial y^2}\right)\delta^2 u^n \\ = \frac{\Delta t}{1+\xi}\left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)\left[1 + \left(\xi + \frac{1}{2}\right)\nabla\right]u^n - \frac{1}{1+\xi}\nabla u^n.\end{aligned} \tag{5.14}$$

A possible advantage of the δ^2 formulation is that the cross-term error introduced by the approximate factorization is one order higher than in the general $(\Delta - \alpha\nabla)$ formulation with $\alpha \neq 1$. By comparing the left-hand sides of (5.4) and (5.5), we find they differ by

$$\omega^2\,\Delta t^2\, a\frac{\partial^2}{\partial x^2} c\frac{\partial^2}{\partial y^2}(\Delta - \alpha\nabla)u^n,$$

which, for $\alpha = 1$, becomes

$$\begin{aligned}\omega^2\,\Delta t^2\, a\frac{\partial^2}{\partial x^2} c\frac{\partial^2}{\partial y^2}\delta^2 u^n &= \omega^2\,\Delta t^4\, a\frac{\partial^2}{\partial x^2} c\frac{\partial^2}{\partial y^2}\frac{\partial^2 u^n}{\partial t^2} + O(\Delta t^5) \\ &= O(\Delta t^4),\end{aligned} \tag{5.15}$$

where we have used (5.12). The cross-term error for the $\Lambda = E^{-1}\rho(E)$ formulation is given by (4.4). The parameter space for which the δ^2 formulation is unconditionally stable is given by inequalities (5.7a, b) with $\alpha = 1$ and is shown graphically in Fig. 4.

A distinct disadvantage of the δ^2 formulation occurs when it is applied to convective (hyperbolic) model equations as briefly discussed in § 8.

5.2c. The $2\mu\delta$ formulation ($\alpha = -1$). As a final special case of the generalized formulation we choose $\alpha = -1$, that is,

$$\Delta - \alpha\nabla = \Delta + \nabla = 2\mu\delta, \tag{5.16}$$

where $2\mu\delta$ is the classical central difference operator

$$2\mu\delta u^n = u^{n+1} - u^{n-1}. \tag{5.17}$$

The scheme (5.5) becomes

$$\begin{aligned}\left(1 - \omega\,\Delta t\, a\frac{\partial^2}{\partial x^2}\right)\left(1 - \omega\,\Delta t\, c\frac{\partial^2}{\partial y^2}\right)2\mu\delta u^n \\ = \frac{\Delta t}{1+\xi}\left(a\frac{\partial^2}{\partial x^2} + b\frac{\partial^2}{\partial x\,\partial y} + c\frac{\partial^2}{\partial y^2}\right)\left[1 + \left(\xi - 2\theta + \frac{1}{2}\right)\nabla\right]u^n \\ + 2\omega\,\Delta t\, b\frac{\partial^2}{\partial x\,\partial y}\nabla u^n + \frac{1+2\xi}{1+\xi}\nabla u^n.\end{aligned} \tag{5.18}$$

The parameter space for unconditional stability of this formulation is given by inequalities (5.7a, b) with $\alpha = -1$,

$$\theta \geqq 1 + \xi, \qquad \xi \geqq -\tfrac{1}{2}, \tag{5.19}$$

which is identical to the parameter space for the unfactored scheme (3.12) (see inequality (3.16) and Fig. 2).

5.3. Algorithm selection. From the class of unconditionally stable methods one can choose a scheme with properties that are desirable with regard to computer storage, computational simplicity, and temporal behavior when applied to stiff problems and/or problems with nonsmooth data. The choice generally requires a compromise.

Consider, for example, the Λ formulation (4.11) of § 4. The computation of the right-hand side of (4.11) is obviously simplified if we set

$$\xi - \omega + \tfrac{1}{2} = 0. \tag{5.20}$$

Since $\omega = \theta/(1+\xi)$, this equation can be rewritten as

$$\theta = (\xi + 1)(\xi + \tfrac{1}{2}); \tag{5.21}$$

it is plotted in Fig. 3. Another variant is obtained by rewriting the right-hand side of (4.11) as

$$\begin{aligned} \text{RHS}(4.11) = {} & \Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)\left[\left(\xi - \omega + \frac{3}{2}\right) - \left(\xi - \omega + \frac{1}{2}\right)E^{-1}\right]u^n \\ & + \Delta t\, b\frac{\partial^2}{\partial x\,\partial y}\left[1 + \left(\xi + \frac{1}{2}\right)\nabla\right]u^n, \end{aligned} \tag{5.22}$$

where we have used (4.9a). The calculation of (5.22) is simplified if we let

$$\xi - \omega + \tfrac{3}{2} = 0, \tag{5.23}$$

which can be rewritten as

$$\theta = (\xi + 1)(\xi + \tfrac{3}{2}); \tag{5.24}$$

it is also plotted in Fig. 3. If in addition, we choose $\xi = -\frac{1}{2}$, then (5.22) becomes simply

$$\text{RHS}(4.11) = \Delta t\left(a\frac{\partial^2}{\partial x^2} + c\frac{\partial^2}{\partial y^2}\right)u^{n-1} + \Delta t\, b\frac{\partial^2 u^n}{\partial x\,\partial y}. \tag{5.25}$$

In this special case, each spatial derivative on the right-hand side of (4.11) requires evaluation at only a single time level. The time differencing ($\theta = \frac{1}{2}$, $\xi = -\frac{1}{2}$) corresponds to the two-step trapezoidal formula (see Table 1). A_0- and A-stable methods along the bottom boundary $\xi = -\frac{1}{2}$ of Fig. 1 are "symmetric" schemes. These methods have the unfortunate property that the modulus of at least one root of the characteristic equation (2.13) approaches 1 as $\lambda \Delta t \to \infty$. Consequently, these methods can produce slowly decaying numerical oscillations when applied to stiff problems. This observation illustrates that computational simplicity should not provide the sole basis for selecting a time-differencing scheme.

The computation of the right-hand side of the Δ formulation (5.11) is obviously simplified if we set

$$\xi - \theta + \tfrac{1}{2} = 0. \tag{5.26}$$

This special case of the Δ formulation was given by the authors in [1; see (42)]. The Δ

formulation is particularly attractive for its simplicity in programming logic and minimal computing storage requirements. Finally, the $\delta^2 u$ formulation (5.14) has the computational advantage that all spatial derivatives on the right-hand side operate on the same function, that is, $[1+(\xi+\frac{1}{2})\nabla]u^n$. For the special case $\xi=-\frac{1}{2}$, the function is conveniently u^n.

Although considerations of computer storage and computational simplicity may not be particularly important for the simple model equation (1.1), they are of primary concern when one deals with more complicated parabolic equations such as the compressible Navier–Stokes equations (see, e.g., [1], [21]).

5.4. General formulation with no mixed derivative. It is important to note that if $b\equiv 0$, that is, there is no mixed derivative, then inequalities (5.7) are replaced by

$$\xi\leqq 2\theta-1, \qquad \xi\geqq -\tfrac{1}{2}, \qquad -1\leqq\alpha\leqq 1, \tag{5.27}$$

and the general two-step ADI formula (5.5) is unconditionally stable for the same values of (θ, ξ) as for the original second-order, two-step method (see inequalities (2.16) and Fig. 1). In the absence of mixed derivatives, the natural extension of (5.5) to three spatial dimensions is also unconditionally stable for values of (α, θ, ξ) satisfying inequalities (5.27).

It is appropriate to mention the relation between the Douglas–Gunn method [9, § 3] for multilevel difference schemes and the general two-step ADI scheme (5.6) in the absence of a mixed derivative, that is, $b=0$. The difference (5.1) corresponds to the difference

$$u^{n+1}-u_*^{n+1} \tag{5.28a}$$

in the Douglas–Gunn development where

$$u_*^{n+1}=\phi_0 u^n+\phi_1 u^{n-1}, \qquad \phi_0+\phi_1=1. \tag{5.28b}$$

Hence, on comparing (5.28) and (5.1), one finds that

$$\phi_0=1+\alpha, \qquad \phi_1=-\alpha. \tag{5.29}$$

Douglas and Gunn give a formal procedure for devising an ADI scheme from a fully implicit scheme supplied by the user. For example, consider the second-order, two-step method ((3.2) with $b=0$), where $\rho(E)$ and $\sigma(E)$ are defined by (2.7a) and (2.9) and (θ, ξ) satisfy inequalities (2.16). If we apply the formulas (3.7) of [9], we obtain an ADI algorithm that can be shown to be equivalent to (5.6). The resulting scheme is unconditionally stable for $-1\leqq\alpha\leqq 1$ since the LMM is A_0-stable. Recall that the discussion of this paragraph is only for the case of no mixed derivative.

6. Time-dependent coefficients. If the coefficients a, b, c of the PDE (1.1) are functions of time, a difficulty arises when we insert (3.11) into (3.8) since

$$\rho(E)\left[a(t^n)\frac{\partial^2 u^n}{\partial x^2}+c(t^n)\frac{\partial^2 u^n}{\partial y^2}\right]\neq\left[a(t^n)\frac{\partial^2}{\partial x^2}+c(t^n)\frac{\partial^2}{\partial y^2}\right]\rho(E)u^n. \tag{6.1}$$

This is not an equality because the time dependence of the coefficients cannot be neglected when the temporal-difference operator $\rho(E)$ is applied. This problem can be avoided if we begin with the one-leg method (2.21) instead of the conventional LMM.

With σ_1 and σ_2 defined by (3.6), the one-leg method (2.21) is

$$\rho(E)\hat{u}^n=\Delta t f_1(\sigma(E)\hat{u}^n, \sigma(E)t^n)+\Delta t f_2(\sigma_e(E)\hat{u}^n, \sigma_e(E)t^n). \tag{6.2}$$

For the PDE (1.1) we identify f_1 and f_2 as (3.11) and obtain

$$\rho(E)\hat{u}^n = \Delta t\left[a(\bar{t})\frac{\partial^2}{\partial x^2}+c(\bar{t})\frac{\partial^2}{\partial y^2}\right]\sigma(E)\hat{u}^n+\Delta t\left[b(\bar{t}_e)\frac{\partial^2}{\partial x\,\partial y}\right]\sigma_e(E)\hat{u}^n, \tag{6.3}$$

where $\bar{t}$ and $\bar{t}_e$ are defined by

$$\bar{t}=\sigma(E)t^n, \qquad \bar{t}_e=\sigma_e(E)t^n. \tag{6.4a, b}$$

If (6.3) is modified to the prefactored form by subtracting

$$\omega\,\Delta t\left[a(\bar{t})\frac{\partial^2}{\partial x^2}+c(\bar{t})\frac{\partial^2}{\partial y^2}\right]\rho(E)\hat{u}^n$$

from each side, we obtain

$$\begin{aligned}&\left\{1-\omega\,\Delta t\left[a(\bar{t})\frac{\partial^2}{\partial x^2}+c(\bar{t})\frac{\partial^2}{\partial y^2}\right]\right\}\rho(E)\hat{u}^n\\ &\quad=\Delta t\left[a(\bar{t})\frac{\partial^2}{\partial x^2}+c(\bar{t})\frac{\partial^2}{\partial y^2}\right][\sigma(E)-\omega\rho(E)]\hat{u}^n+\Delta t\,b(\bar{t}_e)\frac{\partial^2}{\partial x\,\partial y}\sigma_e(E)\hat{u}^n.\end{aligned} \tag{6.5}$$

The prefactored form (6.5) is identical to (3.12) where a, c, and b are evaluated at $\bar{t}$ and $\bar{t}_e$ defined by (6.4). Consequently, the factored scheme (4.1) is valid for time-dependent coefficients provided a, c, b are evaluated at the appropriate times $\bar{t}$ and $\bar{t}_e$.

For second-order, two-step methods, the shifted-difference operators are defined by (4.6) and (4.7). For this case

$$E^{-1}\bar{t}=E^{-1}\sigma(E)t^n=t^n+(\xi+\tfrac{1}{2})\,\Delta t, \tag{6.6a}$$

$$E^{-1}\bar{t}_e=E^{-1}\sigma_e(E)t^n=t^n+(\xi+\tfrac{1}{2})\,\Delta t, \tag{6.6b}$$

and hence the time-dependent coefficients a, b, c are all evaluated at the same time which we denote by

$$t^*=t^n+(\xi+\tfrac{1}{2})\,\Delta t. \tag{6.7}$$

Therefore, the ADI scheme (4.12) is valid for time-dependent coefficients evaluated at t^*. Likewise, the same statement applies to the general two-step ADI scheme (5.6).

7. Numerical examples. In this section the ADI methods of the previous sections are used to solve the parabolic equation (1.1) for a test problem with variable coefficients. The purpose of these numerical experiments is not to find the optimum scheme but to demonstrate by numerical example that each of the formulations—Λ, Δ, and δ^2—achieves the purported second-order accuracy. In addition, we demonstrate the detrimental effect on the accuracy if the mixed derivative is treated with a first-order-accurate method or the variable coefficients are not evaluated at the proper time level.

For the example problem, the coefficients are

$$a(x, y, t)=(\tfrac{1}{2}x^2+y^2)(1+t^2), \tag{7.1a}$$

$$b(x, y, t)=-(x^2+y^2)(1+t^2), \tag{7.1b}$$

$$c(x, y, t)=(x^2+\tfrac{1}{2}y^2)(1+t^2). \tag{7.1c}$$

An exact solution is

$$u(x, y, t) = (x^2 y + x y^2) \exp\left[-\left(1 + \frac{t^2}{3}\right) t \right] \tag{7.2}$$

Numerical solutions were computed on the unit square ($0 \leqq x, y \leqq 1$) with the initial and boundary values computed from (7.2). For example, the initial condition is

$$u(x, y, 0) = x^2 y + x y^2, \qquad 0 \leqq x, y \leqq 1. \tag{7.3}$$

This model problem is a variant of an example given by McKee and Mitchell [15] modified so that the coefficients a, b, c are time-dependent.

In the numerical computations of this section, the spatial derivatives were approximated by the following central difference approximations

$$\left.\frac{\partial^2 Q}{\partial x^2}\right|_{j,k} = \frac{\delta_x^2 Q_{j,k}}{\Delta x^2} + O(\Delta x^2), \tag{7.4}$$

$$\left.\frac{\partial^2 Q}{\partial y^2}\right|_{j,k} = \frac{\delta_y^2 Q_{j,k}}{\Delta y^2} + O(\Delta y^2), \tag{7.5}$$

$$\begin{aligned} \left.\frac{\partial Q}{\partial x\, \partial y}\right|_{j,k} &= \frac{(\mu\delta)_x (\mu\delta)_y}{\Delta x\, \Delta y} Q_{j,k} + O(\Delta x^2, \Delta y^2) \\ &= \frac{1}{4\, \Delta x\, \Delta y} (Q_{j+1,k+1} - Q_{j+1,k-1} - Q_{j-1,k+1} + Q_{j-1,k-1}), \end{aligned} \tag{7.6}$$

where $x = j\,\Delta x$ and $y = k\,\Delta y$. Here δ and μ are classical finite-difference operators defined by

$$\delta_x Q_j = Q_{j+1/2} - Q_{j-1/2}, \qquad 2\mu_x Q_j = Q_{j+1/2} + Q_{j-1/2},$$

and hence

$$\delta_x^2 Q_j = Q_{j+1} - 2Q_j + Q_{j-1},$$

$$2(\mu\delta)_x Q_j = Q_{j+1} - Q_{j-1}, \quad \text{etc.}$$

Consider, for example, the Λ formulation (4.12). With the spatial derivatives replaced by the central-difference quotients (7.4)–(7.6), the x- and y-operators on the left-hand side of (4.12a, b) each requires the solution of a tridiagonal system. There is a well-known and highly efficient solution algorithm for tridiagonal systems (see, e.g., [11, p. 55]). The solution of the x-operator (4.12a) (along each y-constant line) requires the dummy temporal difference Λu^* along the left and right boundaries. In problems considered in this section, we assume that $u(t)$ is given on the boundaries, and consequently Λu^* can be determined by an explicit calculation using (4.12b) applied along both the left- and right-hand boundaries. This is the initial calculation made when advancing the solution from n to $n+1$. Application of the general two-step ADI scheme (5.6) requires an analogous computation of a dummy temporal difference along the left and right boundaries.

Since the algorithms considered in this paper are, in general, two-step (temporal) schemes, a solution at $t = \Delta t$ is needed together with the initial condition to start the computation. For the numerical examples computed herein, the exact solution (7.2) at $t = \Delta t$ was used to provide the additional level of data. In practice, one can use (4.12) as a one-step method on the first time step. This is accomplished by replacing the right-hand side of (4.11) by (4.13) and choosing $\theta = \frac{1}{2}$, $\xi = 0$.

The numerical differentiation formulas (7.4) and (7.5) are exact (i.e., the truncation error is zero) for a polynomial of degree not exceeding three, and (7.6) is exact for a polynomial of degree not exceeding two. Since the exact solution (7.2) is a quadratic polynomial of degree two in each spatial variable, the numerical solution of an unfactored algorithm would have the peculiar property that there would be no spatial discretization error. Consequently, the error in a numerical solution for the example problem (7.1) consists of the temporal discretization error and the cross-product error term from the approximate factorization (see, e.g., (4.4)), and, of course, roundoff error.

For each numerical experiment, we compute the L_2 norm of the error which is defined as follows. At a given time, $t = t^n = n\Delta t$, the error $e_{j,k}$ at each grid point is defined by

$$e_{j,k} = u^n_{j,k} - u(j\,\Delta x, k\,\Delta y, t^n), \tag{7.7}$$

where $u^n_{j,k}$ is the numerical solution and $u(j\,\Delta x, k\,\Delta y, t^n)$ is the analytical solution. The Euclidean or L_2 norm of the error is defined by

$$L_2 \text{ error} = \left[\left(\sum_{j=1}^{J}\sum_{k=1}^{K} e^2_{j,k}\right)\Big/ JK\right]^{1/2} \tag{7.8}$$

where J and K are the total number of grid points in the x- and y-directions.

The second-order backward differentiation method ($\theta = 1$, $\xi = \frac{1}{2}$) (see Table 1) was chosen as the generating LMM for the first computational experiment. The L_2 errors for the Λ, Δ, and δ^2 formulations (algorithms (4.12), (5.11) and (5.14)) are shown in Table 2. Each computation was carried out to a given time ($t = 1.0$) with a fixed ratio of $\Delta t/\Delta x = 1.0$. The computations were repeated with successively smaller values of Δt so that the L_2 rate could be computed. (The L_2 rate is the slope of a log-log graph of the L_2 error vs. Δt. For a second-order method without roundoff error, the L_2 rate should approach two as $\Delta t \to 0$.) The results show the second-order accuracy of the methods. Since the same time-differencing method was used for each computation, the differences in the L_2 error (for a given Δt) result from the cross-product error of the approximate factorization.

TABLE 2
L_2 error of the Λ, Δ, and δ^2 formulations at $t = 1.0$.

$\Delta t = \Delta x = \Delta y$	Number of time steps	$\Delta t/\Delta x^2$	Λ formulation		Δ formulation		δ^2 formulation	
			L_2 error	L_2 rate	L_2 error	L_2 rate	L_2 error	L_2 rate
0.2	5	5	0.785×10^{-2}		0.107×10^{-1}		0.134×10^{-2}	
				2.02		2.01		1.57
0.1	10	10	0.193×10^{-2}		0.266×10^{-2}		0.452×10^{-3}	
				2.01		2.03		1.72
0.05	20	20	0.479×10^{-3}		0.649×10^{-3}		0.137×10^{-3}	
				2.01		2.03		1.89
0.025	40	40	0.119×10^{-3}		0.160×10^{-3}		0.372×10^{-4}	

The next numerical experiment demonstrates the detrimental effect on the accuracy if the mixed derivative is treated with first-order accuracy. The errors listed in Table 3 were computed using the Λ formulation with the backward differentiation method as the generating LMM. For reference, the results listed under column (1) are repeated from Table 2. The L_2 errors and rates tabulated under column (2) were

obtained using (4.12) but with the right-hand side of (4.11) replaced by (4.13), that is, a first-order temporal treatment for the mixed derivative. The degradation in accuracy is obvious.

Column (3) of Table 3 shows the deterioration in accuracy when the time-dependent coefficients a, b, c are not evaluated at the proper time level. The coefficients should be evaluated at t^* defined by (6.7) and hence for the backward differentiation method ($\xi = \frac{1}{2}$) $t^* = t^n + \Delta t$. In obtaining the L_2 errors listed in column (3), the coefficients were evaluated at t^n rather than t^* and the loss of accuracy is apparent.

TABLE 3

Numerical experiments illustrating (1) *second-order* Λ *formulation,* (2) *degradation in accuracy when mixed derivative is computed with a first-order method,* (3) *deterioration in accuracy when time-dependent coefficients are not evaluated at proper time level.*

$\Delta t = \Delta x = \Delta y$	Number of time steps	$\Delta t/\Delta x^2$	(1)		(2)		(3)	
			L_2 error	L_2 rate	L_2 error	L_2 rate	L_2 error	L_2 rate
0.2	5	5	0.758×10^{-2}		0.585×10^{-2}		0.907×10^{-2}	
				2.02		0.59		1.72
0.1	10	10	0.193×10^{-2}		0.389×10^{-2}		0.275×10^{-2}	
				2.01		0.85		1.63
0.05	20	20	0.479×10^{-3}		0.216×10^{-2}		0.888×10^{-3}	
				2.01		0.94		1.47
0.025	40	40	0.119×10^{-3}		0.113×10^{-2}		0.320×10^{-3}	

It is important to note for $\xi = 0$ that the operator Λ defined by (4.5) in the Λ formulation becomes

$$\Lambda u^n = \Delta u^n.$$

Consequently, the Λ and Δ formulations are identical if $\xi = 0$. (Recall that the Δ algorithm is given by (5.6) with $\alpha = 0$.) An advantage of the Δ formulation is that u^{n-1} is not needed to compute u^{n+1} in the final step (5.6c); hence, the Δ form generally requires the least amount of storage. On the other hand, the Δ formulation for $\xi \neq 0$ has a significantly reduced parameter space (θ, ξ) for unconditional stability when applied to hyperbolic problems (see § 8 and [20, § 9]). Consequently, because the Λ and Δ formulations are identical for $\xi = 0$, this subclass of schemes has the simplicity of the Δ form and the robustness of the Λ form. Table 4 compares the L_2 error and rate for several schemes with $\xi = 0$. For a fixed value of ξ in the region of A_0-stability (see Fig. 1), the error constant is a monotone increasing function of θ. This is verified by comparing the L_2 errors for a given value of Δt in Table 4.

TABLE 4

L_2 error of the Λ formulation for $\xi = 0$ and several values of θ at $t = 1.0$.

$\Delta t = \Delta x = \Delta y$	Number of time steps	$\Delta t/\Delta x^2$	$\xi = 0, \theta = \frac{2}{3}$		$\xi = 0, \theta = \frac{3}{4}$		$\xi = 0, \theta = \frac{3}{2}$	
			L_2 error	L_2 rate	L_2 error	L_2 rate	L_2 error	L_2 rate
0.2	5	5	0.705×10^{-2}		0.871×10^{-2}		0.262×10^{-1}	
				2.06		2.07		1.85
0.1	10	10	0.169×10^{-2}		0.208×10^{-2}		0.726×10^{-2}	
				2.03		2.03		2.00
0.05	20	20	0.415×10^{-3}		0.511×10^{-3}		0.181×10^{-2}	
				2.02		2.02		2.01
0.025	40	40	0.102×10^{-3}		0.126×10^{-3}		0.448×10^{-3}	

According to the stability analysis of Appendix B, methods that fall in the region between the curves (4.15) are not unconditionally stable in the Λ formulation (4.11) for all values of the coefficient b satisfying inequality (1.1d). The last numerical experiment verifies this result for the LMM methods listed in Table 1. The parameters used in the computation are listed in the caption of Table 5. The loss of unconditional stability for the trapezoidal formula and Lees method is apparent from the large error for these two methods listed in Table 5.

TABLE 5

L_2 error of Λ formulation (4.11) at $t = 1.0$. Parameters are $\Delta x = \Delta y = 0.025$, $\Delta t = 0.005$, number of time steps $= 200$, $|a_{j,k}|_{\max} \Delta t/\Delta x^2 \simeq 23$.

Method	L_2 error
One-step trapezoidal	0.246×10^{2}
Backward differentiation	0.479×10^{-5}
Lees type	0.405×10^{19}
Adams type	0.505×10^{-5}
Two-step trapezoidal	0.772×10^{-5}

8. Concluding remarks. In this paper we combine the class of all A_0-stable second-order linear two-step methods and the method of approximate factorization to construct unconditionally stable ADI methods for parabolic equations (in two space dimensions) with a mixed derivative. For computational simplicity the mixed derivative is treated explicitly.

In the application of the approximate factorization method we consider several different solution variables. In § 4 we follow [20] and select $\rho(E)u^n$ as the unknown variable. This choice provides a natural framework for constructing unconditionally stable ADI methods for parabolic PDEs by combining A_0-stable LMMs with approximate factorization. The choice of the unknown variable is not unique and for completeness we derive a general two-step ADI scheme with $(\Delta - \alpha\nabla)u^n$ as the unknown variable in § 5. The general formulation contains a parameter α in addition to the parameters (θ, ξ) of the second-order, two-step method. The parameter space (α, θ, ξ) for unconditional stability is determined in Appendix B. Several general observations can be made regarding the stability of these schemes: (1) For a given value of α in the range $[-1, 1]$, the parameter space (θ, ξ) for unconditional stability is reduced from that of the unfactored implicit algorithm (3.2) (compare Fig. 1 with Figs. 3 and 4), but is increased from that of the unfactored implicit-explicit (i.e., explicit treatment of mixed derivative) algorithm (3.12) (compare Fig. 2 with Figs. 3 and 4). (2) For any allowed value of α, the reduced parameter space excludes the familiar (one-step) trapezoidal formula and the Lees type scheme (see Table 1). (3) The $\delta^2 u$ formulation retains the largest parameter space for unconditional stability. (4) The $\rho(E)$ or Λ formulation has the peculiar property that $\alpha = \xi/(1+\xi)$, that is, α depends on the particular LMM chosen. The extent of the parameter space for unconditional stability (Fig. 3) is nearly as large as for the δ^2 formulation (Fig. 4 with $\alpha = 1$). (5) Although not considered in this paper, it can be shown that if the general $(\Delta - \alpha\nabla)$ formulation is applied to the model convection equation

$$\frac{\partial u}{\partial t} = -c_1 \frac{\partial u}{\partial x} - c_2 \frac{\partial u}{\partial y}, \tag{8.1}$$

then only the $\rho(E)$ formulation (i.e., $\alpha = \xi/(1+\xi)$) retains the same parameter space (shaded region of Fig. 1) for unconditional stability as the generating A-stable LMM. In fact, the δ^2 formulation has no parameter values (θ, ξ) for which the scheme is unconditionally stable.

The emphasis of this paper is on the construction of unconditionally stable second-order accurate ADI methods for the model parabolic equation (1.1). By following the approach outlined herein (i.e., the use of A_0-stable LMMs in conjunction with the method of approximate factorization) one can easily construct algorithms for multidimensional nonlinear parabolic systems. For some auspicious reason, the parameter space (θ, ξ) for which the class of second-order two-step methods is A_0-stable happens to coincide with the parameter space for which this class of methods is A-stable. Consequently, one can use the class of time-differencing schemes of this paper to design second-order ADI algorithms for mixed hyperbolic-parabolic systems of nonlinear equations. A noniterative algorithm in the Δ-form for nonlinear systems was considered in [1] and a general development for the $\rho(E)$ formulation is in a companion paper [21].

Appendix A. Stability analysis of combined LMMs. In this appendix we examine the stability of the combined LMM (3.7) applied to the model split ODE:

$$\frac{du}{dt} = \lambda_1 u + \lambda_2 u; \qquad \lambda_1 < 0, \quad \lambda_1 + \lambda_2 \leqq 0, \tag{A.1a,b}$$

where λ_1 and λ_2 are real constants. In addition, we investigate the stability of the *unfactored* scheme (3.12) for the PDE (1.1). The analysis is for the class of all second-order, two-step methods.

Consider the combined LMM (3.7) where $\rho(E)$, $\sigma(E)$, and $\sigma_e(E)$ are defined by (2.7a), (2.9), and (3.15). If we apply this scheme to the model equation (A.1) with $f_1 = \lambda_1 u$ and $f_2 = \lambda_2 u$, we obtain a difference equation whose characteristic equation is

$$a_2\zeta^2 + a_1\zeta + a_0 = 0, \tag{A.2}$$

where

$$a_2 = (1+\xi) - \theta\lambda_1 \Delta t, \tag{A.3a}$$

$$a_1 = -(1+2\xi) - (\xi - 2\theta + \tfrac{3}{2})\lambda_1 \Delta t - (\xi + \tfrac{3}{2})\lambda_2 \Delta t, \tag{A.3b}$$

$$a_0 = \xi + (\xi - \theta + \tfrac{1}{2})\lambda_1 \Delta t + (\xi + \tfrac{1}{2})\lambda_2 \Delta t. \tag{A.3c}$$

Equation (A.2) is a von Neumann polynomial [16], that is, $|\zeta| \leqq 1$, if and only if

$$a_0 \leqq a_2, \tag{A.4a}$$

and

$$-(a_2 + a_0) \leqq a_1 \leqq a_2 + a_0, \tag{A.4b}$$

where without loss of generality a_2 is assumed to be positive. The inequalities (A.1b) and (A.4) lead to the following conditions for the stability of the combined two-step scheme:

$$\xi \geqq -\tfrac{1}{2}, \qquad 0 \leqq (1+2\xi) + (1-2\theta+\xi)\lambda_1 \Delta t + (1+\xi)\lambda_2 \Delta t. \tag{A.5a,b}$$

In particular, the conditions for A_0-stability are

$$\xi \geqq -\tfrac{1}{2}, \qquad \lambda_2 \geqq -\frac{(1-2\theta+\xi)}{1+\xi}\lambda_1. \tag{A.6a,b}$$

Note that for the special case $\lambda_2 = 0$, (A.6b) becomes $\xi \leqq 2\theta - 1$ and conditions (A.6) reduce to (2.16).

For the stability analysis of the unfactored scheme (3.12) for the PDE (1.1) we need only consider the stability of the linear two-step scheme (3.7) applied to the ODE (3.4) for the Fourier coefficient. In this appendix we consider only the spatially continuous solution; however, the results are applicable to the spatially discrete case (see last paragraph of Appendix B). The conditions on the parameters (θ, ξ) for the unconditional stability of (3.12) can be derived from the A_0-stability requirements (A.6) and the relations

$$\lambda_1 = -(a\kappa_1^2 + c\kappa_2^2), \qquad \lambda_2 = -b\kappa_1\kappa_2, \tag{A.7a,b}$$

obtained by comparing (A.1) and (3.4b). There follows

$$\xi \geqq -\tfrac{1}{2}, \qquad -b\kappa_1\kappa_2 \geqq \frac{(1-2\theta+\xi)}{(1+\xi)}(a\kappa_1^2 + c\kappa_2^2). \tag{A.8a,b}$$

The inequalities (A.8a,b) together with (1.1c,d) imply

$$\xi \geqq -\tfrac{1}{2}, \qquad \xi \leqq \theta - 1. \tag{A.9a,b}$$

Inequality (A.9b) is more restrictive than the inequality (2.16a) for the generating two-step method (2.6) to be A_0-stable. Consequently, we have the result that the second-order explicit treatment of the mixed derivative reduces the parameter space (θ, ξ) for which the *unfactored* scheme (3.12) is unconditionally stable (see Fig. 2).

Appendix B. Stability analysis for two-step ADI schemes. In this appendix we perform a linear stability analysis for the factored scheme (5.5). We assume that u^n is spatially continuous and seek a solution of the form

$$u^n = v^n e^{i(\kappa_1 x + \kappa_2 y)} \tag{B.1}$$

where v^n is the Fourier coefficient and $\kappa_1\kappa_2$, are the Fourier variables. Prior to an actual numerical computation, the spatial derivatives are replaced by appropriate difference quotients; however, as indicated at the end of this appendix, the stability proof for the spatially discrete case requires only a minor modification of the following stability proof.

By substituting (B.1) into (5.5), we find that the Fourier coefficient satisfies

$$\begin{aligned}(1+A)(1+C)(\Delta - \alpha\nabla)v^n = &-\frac{1}{\theta}(A+B+C)\left[1+\left(\xi-\theta+\frac{1}{2}\right)\nabla\right]v^n \\ &-B\nabla v^n - \alpha(A+C)\nabla v^n + \left(\frac{\xi}{1+\xi} - \alpha\right)\nabla v^n,\end{aligned} \tag{B.2}$$

where we have defined

$$A = \omega\,\Delta t\, a\kappa_1^2, \qquad B = \omega\,\Delta t\, b\kappa_1\kappa_2, \qquad C = \omega\,\Delta t\, c\kappa_2^2 \tag{B.3}$$

and $\omega = \theta/(1+\xi)$. The amplification factor is defined by

$$v^{n+1} = \zeta v^n, \tag{B.4}$$

and consequently it follows from (B.2) that ζ satisfies the quadratic equation

$$a_2\zeta^2 + a_1\zeta + a_0 = 0, \tag{B.5}$$

where

(B.6a) $$a_2 = (1+A)(1+C),$$

(B.6b) $$a_1 = -(1+\alpha)(1+A)(1+C)+\frac{1}{\theta}\left(\xi-\theta+\frac{3}{2}\right)(A+B+C) +B+\alpha(A+C)-\left(\frac{\xi}{1+\xi}-\alpha\right),$$

(B.6c) $$a_0 = \alpha(1+A)(1+C)-\frac{1}{\theta}\left(\xi-\theta+\frac{1}{2}\right)(A+B+C) -B-\alpha(A+C)+\left(\frac{\xi}{1+\xi}-\alpha\right).$$

In the one-dimensional case ($b = c = 0$ in (1.1) and $B = C = 0$ in (B.5)), the roots of the quadratic (B.5) have modulus bounded by unity for those values of (θ, ξ) shown in the shaded region of Fig. 1, that is,

(B.7a,b) $$\xi \leqq 2\theta - 1, \qquad \xi \geqq -\tfrac{1}{2}.$$

This one-dimensional result follows from the analysis [20]. Note that α only enters as a parameter in the two-dimensional *factored* algorithm (5.5). (Recall that (5.2) and (5.3) are actually identical.) One can easily verify that the coefficients (B.6) do not depend on α if $B = C = 0$. We must determine if there are additional restrictions on the parameters (θ, ξ) for the unconditional stability of the factored scheme (5.5) for arbitrary values of a, b, c subject only to the parabolicity conditions

(B.8a,b) $$a > 0, \qquad b^2 < 4ac$$

of the partial differential equation (1.1). Since the one-dimensional problem ($b = c = 0$) is a special case of the two-dimensional problem, we need not consider values of (θ, ξ) outside the domain (B.7). Hence $\omega > 0$, A and C as defined by (B.3) are positive, and

(B.9) $$A + B + C = \omega\, \Delta t\, (a\kappa_1^2 + b\kappa_1\kappa_2 + c\kappa_2^2) > 0,$$

since the positive definiteness of this quadratic form was the condition which led originally to (B.8).

The coefficients (B.6) of the quadratic (B.5) are real and consequently the roots ζ satisfy $|\zeta| \leqq 1$ if and only if the inequalities (A.4) of Appendix A are satisfied. If we insert a_0 and a_2 as given by (B.6) into (A.4a), there follows

(B.10) $$0 \leqq \frac{1}{1+\xi} + (1-\alpha)AC + \frac{1}{\theta}\left(\xi+\frac{1}{2}\right)(A+B+C),$$

which is satisfied for all allowable A, B, C if and only if

(B.11) $$\alpha \leqq 1.$$

(Recall that the parameters θ and ξ are required to satisfy inequalities (B.7).) Likewise the left inequality of (A.4b) is satisfied. If the coefficients (B.6) are inserted into the right inequality of (A.4b) one obtains

(B.12) $$0 \leqq (1+\alpha)AC - \frac{1}{\omega}B + \frac{1+2\xi}{1+\xi} + \left(2-\frac{1}{\omega}\right)(A+C).$$

The determination of necessary and sufficient conditions for this inequality is simplified

if it is rewritten as

$$0 \leqq (1+\alpha)ac(k_1k_2)^2 - \frac{1}{\omega}bk_1k_2 + \frac{1+2\xi}{1+\xi} + \left(2-\frac{1}{\omega}\right)(ak_1^2 + ck_2^2) \tag{B.13}$$

where we have used the definitions (B.3) and defined:

$$k_1 = \sqrt{\omega\,\Delta t}\,\kappa_1, \qquad k_2 = \sqrt{\omega\,\Delta t}\,\kappa_2. \tag{B.14a,b}$$

It can be shown that necessary and sufficient conditions for the polynomial P in k_1, k_2 defined by

$$P = e_1(k_1k_2)^2 + e_2k_1k_2 + e_3 + e_4k_1^2 + e_5k_2^2 \tag{B.15}$$

to be positive semidefinite (i.e., $P \geqq 0$ for all real k_1, k_2) are

$$e_1, e_3, e_4, e_5 \geqq 0, \tag{B.16a}$$

$$|e_2| \leqq 2\sqrt{e_1e_3} + 2\sqrt{e_4e_5}. \tag{B.16b}$$

Comparison of (B.15) and (B.13) leads to the conditions

$$(1+\alpha)ac \geqq 0, \qquad \frac{1+2\xi}{1+\xi} \geqq 0, \tag{B.17a,b}$$

$$\left(2-\frac{1}{\omega}\right)a \geqq 0, \qquad \left(2-\frac{1}{\omega}\right)c \geqq 0, \tag{B.17c,d}$$

$$\left|\frac{b}{\omega}\right| \leqq 2\sqrt{(1+\alpha)ac\frac{1+2\xi}{1+\xi}} + 2\sqrt{\left(2-\frac{1}{\omega}\right)^2 ac}. \tag{B.17e}$$

Inequalities (B.17a, b, c, d) are satisfied by virtue of inequalities (B.7), (B.8a), and (B.11) plus the constraint:

$$-1 \leqq \alpha. \tag{B.18}$$

Inequality (B.17e) can be rewritten as

$$\frac{1}{\omega}|b| \leqq 2\sqrt{ac}\left[\sqrt{\frac{(1+\alpha)(1+2\xi)}{1+\xi}} + \left(2-\frac{1}{\omega}\right)\right], \tag{B.19}$$

which is satisfied for all allowable a, b, c (see inequalities (B.8)) if and only if

$$\frac{1}{\omega} \leqq \sqrt{\frac{(1+\alpha)(1+2\xi)}{1+\xi}} + \left(2-\frac{1}{\omega}\right), \tag{B.20}$$

or

$$\theta \geqq \frac{2(1+\xi)}{2+\sqrt{\dfrac{(1+\alpha)(1+2\xi)}{1+\xi}}} \tag{B.21}$$

Hence the final inequalities which must be satisfied are (B.7b), (B.11), (B.18), and (B.21).

In the above stability analysis we assumed that the spatial derivatives were continuous. Since in practice the spatial derivatives are replaced by discrete difference quotients, it remains to consider the spatially discrete case. If, for example, the spatial derivatives in (5.5) are replaced by the second-order difference quotients (7.4)–(7.6), then the stability analysis proceeds as above with the exception that the exponential in

(B.1) is replaced by

$$u_{j,k}^{n} = v^{n}\, e^{i(\kappa_1 j\Delta x + \kappa_2 k\Delta y)}, \tag{B.22}$$

where $x = j\,\Delta x$, $y = k\,\Delta y$. If we make the following correspondence

$$\kappa_1 \leftarrow \frac{2\sin(\theta_1/2)}{\Delta x}, \qquad \kappa_2 \leftarrow \frac{2\sin(\theta_2/2)}{\Delta y}, \tag{B.23a,b}$$

$$B \leftarrow B\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}, \tag{B.24}$$

where

$$\theta_1 = \kappa_1\,\Delta x, \qquad \theta_2 = \kappa_2\,\Delta y,$$

between the parameters for the discrete and continuous case, then the amplification factor for the discrete case satisfies the same quadratic (B.5) with coefficients (B.6). Since the stability region defined by inequalities (B.7b), (B.11), (B.18), and (B.21) is valid for arbitrary values of κ_1 and κ_2 and

$$\left|B\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\right| \leqq |B|, \tag{B.25}$$

we obtain the same stability range for the discrete case. If one uses a noncentered approximation for the mixed derivative such as

$$\begin{aligned}\left.\frac{\partial^2 Q}{\partial x\,\partial y}\right|_{j,k} &= \frac{1}{2\,\Delta x\,\Delta y}[\nabla_x\,\Delta_y + \Delta_x\,\nabla_y]Q_{j,k} + O(\Delta x^2, \Delta y^2),\\ &= \frac{1}{2\Delta x\,\Delta y}(Q_{j+1,k} - 2Q_{j,k} + Q_{j-1,k} - Q_{j+1,k-1}\\ &\qquad + Q_{j,k+1} + Q_{j,k-1} - Q_{j-1,k+1}) + O(\Delta x^2, \Delta y^2),\end{aligned} \tag{B.26}$$

where

$$\Delta_x Q_j = Q_{j+1} - Q_j, \qquad \nabla_x Q_j = Q_j - Q_{j-1}, \quad \text{etc.},$$

the only modification necessary in the stability analysis is replacement of (B.24) by

$$B \leftarrow B\cos\left(\frac{\theta_1 - \theta_2}{2}\right). \tag{B.27}$$

REFERENCES

[1] R. M. BEAM AND R. F. WARMING, *An implicit factored scheme for the compressible Navier-Stokes equations*, AIAA J., 16 (1978), pp. 393–402.

[2] M. CIMENT, S. H. LEVENTHAL AND B. C. WEINBERG, *The operator compact implicit method for parabolic equations*, J. Computational Phys., 28 (1978), pp. 135–166.

[3] C. W. CRYER, *A new class of highly-stable methods: A_0-stable methods*, BIT, 13 (1973) pp. 153–159.

[4] G. DAHLQUIST, *A special stability problem for linear multistep methods*, Ibid., 3 (1963), pp. 27–43.

[5] ———, *Error analysis for a class of methods for stiff non-linear initial value problems*, Numerical Analysis Dundee 1975, Lecture Notes in Mathematics 506, A. Dold and B. Eckmann, eds., Springer-Verlag, Berlin, 1976, pp. 60–72.

[6] ———, *The theory of linear multistep methods and related mathematical topics*, Lecture Notes (microfilm), Dept. of Numerical Analysis, Royal Inst. of Tech., Stockholm, 1976.

[7] ———, *Positive functions and some applications to stability questions for numerical methods*, Recent Advances in Numerical Analysis, C. de Boor and G. Golub, eds. Academic Press, New York, 1978, pp. 1–29.

[8] J. DOUGLAS, *On the numerical integration of* $u_{xx} + u_{yy} = u_t$ *by implicit methods*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 42–65.

[9] J. DOUGLAS AND J. E. GUNN, *A general formulation of alternating direction methods*, Numer. Math., 6 (1964), pp. 428–453.

[10] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

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

[12] S. R. K. IYENGAR AND M. K. JAIN, *Comparative study of two and three level ADI methods for parabolic equations with a mixed derivative*, Internat. J. Numerical Methods in Engr., 10 (1976), pp. 1309–1315.

[13] P. D. LAX AND R. D. RICHTMYER, *Survey of the stability of linear finite difference equations*, Comm. Pure Appl. Math., 9 (1956), pp. 267–293.

[14] M. LEES, *A linear three-level difference scheme for quasilinear parabolic equations*, Math. Comp., 20 (1966), pp. 516–522.

[15] S. MCKEE AND A. R. MITCHELL, *Alternating direction methods for parabolic equations in two space dimensions with a mixed derivative*, Comput. J., 13 (1970), pp. 81–86.

[16] J. J. H. MILLER, *On the location of zeros of certain classes of polynomials with applications to numerical analysis*, J. Inst. Math. Appl., 8 (1971), pp. 397–406.

[17] O. NEVANLINNA AND W. LINIGER, *Contractive methods for stiff differential equations*, BIT, 18 (1978), pp. 457–474.

[18] D. W. PEACEMAN AND H. H. RACHFORD, *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.

[19] R. F. WARMING AND R. M. BEAM, *On the construction and application of implicit factored schemes for conservation laws*, Symposium on Computational Fluid Dynamics, New York April 16–17, 1977: SIAM–AMS Proceedings, 11 (1978), pp. 85–129.

[20] ———, *An extension of A-stability to alternating direction implicit methods*, BIT, 19 (1979), pp. 395–417.

[21] ———, *An implicit factored scheme for the compressible Navier–Stokes equations II: The numerical ODE connection*, paper no. 79–1446, Proceedings of the AIAA 4th computational fluid dynamics conference, Williamsburg, Virginia, July 23–24, 1979.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0008 $01.00/0

NUMERICAL SOLUTIONS FOR MAXIMUM SUSTAINABLE CONSUMPTION GROWTH WITH A MULTI-GRADE EXHAUSTIBLE RESOURCE*

U. ASCHER† AND F. Y. M. WAN‡

Abstract. An efficient method is developed for the accurate numerical solution of the nonlinear boundary value problem (BVP) governing the optimal economic growth with a finite multi-grade deposit of nonrenewable and nonaugmentable essential resource under R. M. Solow's maximum sustainable per head consumption level criterion. Unusual computational features of the BVP include: (1) the semi-infinite (time) domain of the problem is divided into a number of subintervals of unknown (and unequal) lengths with a different set of differential equations for each subinterval and with the subinterval lengths to be determined in the solution process; (2) some solution components are known to be unbounded at infinity while others decay very slowly, and (3) in a certain range of parameter values, a previously used solution method is known to be sensitive to boundary data. The new method is used to generate new accurate numerical solutions for single-grade resource problems with a high unit extraction cost and more accurate results for two-grade deposit problems previously investigated. As well, it enables us to investigate for the first time problems with more than two grades of deposits. The implications of these new results are analyzed.

Key words. exhaustible resource, boundary value problem, semi-infinite domain, switch joints, collocation

1. Introduction. One of the principal societal concerns of the nineteen seventies has been the proper use of the Earth's natural resources. This concern is reflected in the considerable amount of research activities in natural resource economics[1]. In the area of exhaustible resources, questions on the proper management and exploitation of finite nonrenewable deposits, such as fossil fuel and minerals, are usually formulated quantitatively as mathematical problems in optimal control. An appropriate formulation of the relevant optimal control problem is not always straightforward, and an exact solution of the problem in terms of elementary or special functions is not always possible. When a numerical solution of the optimal control problem is necessary, the computational procedure required is not always routine.

In this article, we consider a problem of current interest in exhaustible resource economics, and develop a new efficient method for a numerical solution of the associated optimal control problem. The economic problem is a slightly more general version of R. M. Solow's optimal growth under the maximum sustainable consumption rate criterion with a single-grade [1], [3] and a two-grade [2], [3] nonrenewable and nonaugmentable resource deposit. We consider here the same optimal growth problem, but now for a multi-grade resource deposit. The relevant boundary value problem (BVP) for the determination of the optimal growth program for this problem is substantially more complex than the corresponding BVP investigated in [1], [2], [3]. Correspondingly, the computational difficulties associated with a numerical solution for our BVP are more extensive than those experienced in [2] and [3] and are summarized as follows:

* Received by the editors June 1, 1979, and in revised form October 18, 1979.

† Department of Computer Science, University of British Columbia, Vancouver, British Columbia, V6T 1W5 Canada. The work of this author was supported in part by the National Research Council of Canada under Grant A4306.

‡ Department of Mathematics, University of British Columbia, Vancouver, British Columbia, V6T 1W5 Canada. The work of this author was supported in part by the National Research Council under Grant A9259.

[1] For example, an entire 1974 volume of The Review of Economics Studies is devoted to investigations in resource economics.

(1) The domain of the solution is semi-infinite when the economic planning is for the entire future; the solution for some of the unknown quantities is known to be unbounded at infinity while other solution components decay slowly.
(2) The solution domain $[0, \infty)$ is divided up into a number of consecutive subintervals, $[0, T_1), [T_1, T_2), \cdots, [T_J, \infty)$, with a different set of differential equations for each of the subintervals; the location of the switch points $T_1, T_2, \cdots, T_J$ is not known and is to be determined as a part of the solution process.
(3) In a certain range of parameter values, a numerical solution of the BVP in its natural (reduced) form is very sensitive to boundary data [3].

Many of these features are direct consequences of economic considerations and their meaning will be clear once we describe the economic model which gives rise to the optimal control problem.

As pointed out in [3], an accurate numerical method of solution that is practical beyond the two-grade deposit case is needed for the above nonconventional nonlinear BVP. In this article, we develop such a numerical method which turns out to be also more efficient than those used in [2] and [3] when applied to problems with a single-grade or a two-grade deposit. Our approach is essentially to eliminate some of the computational difficulties by various reductions, transformations and simplifications of the BVP for the optimal program. The reduced problem is in a form suitable for the application of a BVP solver COLSYS [4], [5]. A brief description of this general purpose code is given in the Appendix of this paper.

To report the method developed and its applications to specific problems, we begin with a brief summary of the relevant optimal economic growth model in § 2; full details and justifications can be found in [1], [2], [3]. Here, we proceed to formulate the BVP for the determination of the optimal growth program for the general multi-grade deposit case. The two principal results of this section are: (1) the reduction of the complicated BVP to a sequence of simple BVP's of the same type over consecutive time intervals, each involving only one first order ODE, and (2) the demonstration of the coincidence of the optimal growth program and the program for maximum capital stock accumulation for the particular consumption rate of the optimal growth program. The second result generalizes a corresponding result in [3] and is obtained by a different, more general means.

The special case of a single-grade resource problem previously treated in [1], [3] is considered in § 3 in the framework of the reduced BVP of § 2. We use it as a vehicle to describe the way we handle the semi-infinite domain aspect of the problem (including the unboundedness and slow decay of the solution components). The same technique is also used later for the general multi-grade problem. With the efficient method of solution developed in § 3, we solve several more difficult single-grade resource problems with high extraction costs to show how close the actual solution for the consumption level is to its upper bound in some cases. These results explain, for the first time, why a larger resource deposit may not notably alter the consumption level under some circumstances.

The reduced BVP's of § 2 for the optimal growth program are coupled through the unknown constant consumption rate and must be solved simultaneously, each for an unknown solution domain. In § 4, we transform this sequence of BVP's over consecutive (unknown) time intervals into a single BVP over the interval (0, 1) for a system of $2(J+1)$ first order ODE's. The resulting problem is in a form suitable for the application of COLSYS. More accurate solutions of the two-grade resource problems analyzed in [2], [3] are obtained by the new method of solution to demonstrate its

efficiency. The method also enables us to study, for the first time, problems involving more than two grades of resource deposits. Sample results for a three-grade deposit problem are reported and analyzed for the effects of various input parameters on the maximum sustainable consumption level. From a computational viewpoint, we note the important fact that the computing time required for a solution with the same degree of accuracy increases only moderately with the number of resource grades. Thus, our method can be used for multi-grade resource problems with the number of different grade deposits considerably larger than three.

2. Problem formulation and simplifications. Recall the simple economic model studied in [1] and [2], in which a single nonrenewable and nonaugmentable resource is an essential input to the production of some commodity. At any instant t, let $k(t)$ and $r(t)$ be the capital stock and resource flow per head, respectively, and take the per head output of the commodity to be $q = k^a r^b$ with $0 < b < a + b < 1$. Capital accumulation in this model is governed by

$$\dot{k} = k^a r^b - \theta r - c, \tag{2.1}$$

where c is the per head consumption rate and θ is the known unit extraction cost of the resource which characterizes the quality of the deposit.

The Rawls–Solow max-min principle [1] requires that c be a constant. Our problem is to maximize a permanently sustainable consumption level c subject to equation (2.1), a prescribed initial stock of capital

$$k(0) = \bar{k}_o, \tag{2.2}$$

the nonnegativity constraints

$$r(t) \geqq 0 \quad \text{and} \quad k(t) \geqq 0 \tag{2.3}$$

(which are incorporated automatically into (2.1)) and the conditions of a finite stock of exhaustible resource in several grades of deposits specified below.

Suppose the stock of resource consists of $J+1$ different grade deposits of amount $\bar{D}_1, \bar{D}_2, \cdots, \bar{D}_{J+1}$, respectively, and with constant unit extraction costs $\theta_1, \theta_2, \cdots, \theta_{J+1}$, respectively $(0 \leqq \theta_1 < \theta_2 < \cdots < \theta_{J+1})$. According to [2], the resource stock must be depleted in the order of increasing unit extraction cost under the optimal program. Let T_j be the time when the jth resource is exhausted, $0 \equiv T_0 < T_1 < \cdots < T_J < T_{J+1} \equiv \infty$, and let the remaining resource stock at time t, $D(t)$ be defined by

$$\dot{D} = -r, \tag{2.4}$$

$$D(\infty) = 0, \tag{2.5}$$

as the stock of resource will be eventually exhausted under the optimal program. Then, we have also

$$D(T_j) = \sum_{i=j+1}^{J+1} \bar{D}_i, \qquad (j = 0, 1, \cdots, J) \tag{2.6}$$

and $\theta(t) = \theta_j$ in the interval $T_{j-1} < t < T_j$, $j = 1, 2, \cdots, J+1$.

Thus our problem is to choose $r(t)$ to maximize the constant c subject to the equations of state (2.1) and (2.4) and the auxiliary conditions (2.2), (2.5) and (2.6). In addition, k and D must be continuous across T_j, $j = 1, \cdots, J$. The switch points T_j are unspecified and are to be determined in the solution process.

The necessary conditions for an interior maximum c consist of a system of differential equations and transversality conditions for the Lagrange multipliers λ_k and

λ_D [6]:

$$-\dot{\lambda}_k = ak^{a-1}r^b\lambda_k, \qquad -\dot{\lambda}_D = 0 \qquad (T_{j-1} < t < T_j), \tag{2.7,8}$$

$$0 = \lambda_D - \lambda_k(bk^a r^{b-1} - \theta_j) \tag{2.9}$$

for $j = 1, 2, \cdots, J+1$, with

$$\lambda_k(T_j-) = \lambda_k(T_j+), \qquad \lim_{t\to\infty} \lambda_k(t) = 0, \tag{2.10,11}$$

$$[\lambda_k(k^a r^b - \theta_j r - c) - \lambda_D r]_{t=T_j-}^{T_j+} = 0 \tag{2.12}$$

for $j = 1, 2, \cdots, J$. The conditions (2.7)–(2.12) can be reduced to the corresponding conditions in [3] for the two-grade deposit case but have been obtained without the plausible assumption of maximum capital stock accumulation during the cheaper-grade phase of the growth program adopted there. Note that when the planning is for the entire future, the maximum principle may not apply so that these necessary conditions are formal conditions. In this article (and in [1], [2] and [3]), we are concerned only with the process of obtaining a continuous, piecewise differentiable interior solution [6] of these formal necessary conditions. All indications are that the interior solution obtained is unique. For brevity, we will henceforth refer to it as *the* optimal program for the economic growth problem.

In theory, the BVP defined by (2.1)–(2.12) determines within each subinterval the resource extraction rate $r(t)$, the resource stock depletion $D(t)$ and the capital accumulation $k(t)$ (as well as other quantities) for the optimal program, with $D(t)$ and $k(t)$ continuous across the switch points. The task on hand is to find an efficient method for the numerical solution of this complex BVP. For this purpose, we perform some preliminary reductions of the BVP. We begin the reductions by observing that the final (semi-infinite) phase of the optimal program is merely the optimal program for a single-grade resource case with an initial capital stock $\bar{k}_J = k(T_J)$ inherited from an earlier phase of the program. Also, the relevant system of ODE's is autonomous; so the starting time T_J for the final phase is just a reference time and has no substantive effect on the solution. Thus we can use the results in [3] to conclude that for the

$(J+1)$-th phase $(T_J < t < \infty)$:

$$(1-b)k^a r^b = c. \tag{2.13}$$

For the growth program to sustain the same consumption rate for all $t > 0$ in spite of the finiteness of the resource deposit needed for the production of consumption goods, it is necessary to have $k \to \infty$ as $t \to \infty$, as is apparent from (2.15) below (e.g., $k(t)$ is linear in t for the special case $\theta_{J+1} = 0$). To avoid a numerical solution of $k(t)$ directly, we use (2.13) to eliminate $k(t)$ from the BVP. We can write (2.1) for $t > T_J$ as

$$\dot{k} = \frac{c}{1-b} - \theta_{J+1}r - c = \frac{bc}{1-b} + \theta_{J+1}\dot{D} \tag{2.14}$$

or

$$k(t) = \frac{bc}{1-b}(t - T_J) + \theta_{J+1}(D - \bar{D}_{J+1}) + \bar{k}_J, \tag{2.15}$$

where $\bar{k}_J \equiv k(T_J+) = k(T_J-)$. This explicit integration not only reduces the size of the resulting BVP (by one differential equation per resource grade), but also takes care of

possible numerical sensitivity to small changes in $\bar{k}_0$ which was experienced in [3]. Using (2.13) again, we can write (2.4), (2.6) and (2.5) as

$$\text{(2.16, 17, 18)} \quad \dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} k^{-(a/b)}, \quad D(T_J+) = \bar{D}_{J+1}, \quad D(\infty) = 0.$$

In the neighborhood of T_J, we may use (2.9), (2.10) and (2.13) to deduce from the continuity condition (2.12)

$$[(1-b)k^a r^b - c]_{t=T_J^-} = [(1-b)k^a r^b - c]_{t=T_J^+} = 0$$

or

$$\text{(2.19)} \qquad [(1-b)k^a r^b]_{t=T_J^-} = c.$$

Incidentally, the condition (2.19) implies the continuity of r across the switch point T_J (and therewith $\lambda_k(T_J) \neq 0$). We now use (2.19) in a reduction similar to that for a single grade resource case to get for (see [3]) the

J-th phase $(T_{J-1} < t < T_J)$:

$$\text{(2.20)} \qquad (1-b)k^a r^b = c,$$

$$\text{(2.21)} \qquad k = \frac{bc}{1-b}(t - T_{J-1}) + \theta_J(D - \bar{D}_J - \bar{D}_{J+1}) + \bar{k}_{J-1},$$

$$\text{(2.22, 23, 24)} \quad \dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} k^{-(a/b)}, \quad D(T_{J-1}+) = \bar{D}_J + \bar{D}_{J+1}, \quad D(T_J-) = \bar{D}_{J+1},$$

where $\bar{k}_{J-1} \equiv k(T_{J+1}+) = k(T_{J-1}-)$.

Upon repeating the same reduction for each subinterval, we get for the j-th phase $(T_{j-1} < t < T_j)$:

$$\text{(2.25)} \qquad (1-b)k^a r^b = c,$$

$$\text{(2.26)} \qquad k = \frac{bc}{1-b}(t - T_{j-1}) + \theta_j\left(D - \sum_{i=j}^{J+1} \bar{D}_i\right) + \bar{k}_{j-1},$$

$$\text{(2.27, 28, 29)} \quad \dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} k^{-(a/b)}, \quad D(T_{j-1}) = \sum_{i=j}^{J+1} \bar{D}_i, \quad D(T_j) = \sum_{i=j+1}^{J+1} \bar{D}_i$$

for $j = 2, 3, \cdots, J$ with $\bar{k}_{j-1} \equiv k(T_{j-1}+) = k(T_{j-1}-)$.

Finally, the same reduction also gives for the 1-st phase $(0 < t < T_1)$:

$$\text{(2.30)} \qquad (1-b)k^a r^b = c,$$

$$\text{(2.31)} \qquad k = \frac{bct}{1-b} + \theta_1(D - \bar{D}) + \bar{k}_o, \qquad \bar{D} \equiv \sum_{i=1}^{J+1} \bar{D}_i,$$

$$\text{(2.32, 33, 34)} \quad \dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} k^{-(a/b)}, \quad D(0) = \bar{D}, \quad D(T_1) = \sum_{i=2}^{J+1} \bar{D}_i.$$

We now simplify the expressions (2.15), (2.21) and (2.26) for $k(t)$ by deriving an explicit expression for $\bar{k}_j$. From (2.26), (2.28) and (2.29) we obtain the recursive relation

$$\text{(2.35)} \qquad \bar{k}_j = k(T_j) = \frac{bc}{1-b}(T_j - T_{j-1}) - \theta_j \bar{D}_j + \bar{k}_{j-1}.$$

Hence

$$\bar{k}_j = \frac{bc}{1-b} T_j - \sum_{i=1}^{j} \theta_i \bar{D}_i + \bar{k}_0 \qquad (j = 1, \cdots, J). \tag{2.36}$$

Substituting back in (2.26), we get for $T_{j-1} < t < T_j$,

$$k(t) = \frac{bct}{1-b} + \theta_j D(t) + \bar{K}_j, \tag{2.37}$$

where

$$\bar{K}_j \equiv \bar{k}_0 - \theta_j \sum_{i=j}^{J+1} \bar{D}_i - \sum_{i=1}^{j-1} \theta_i \bar{D}_i \qquad (j = 1, \cdots, J+1). \tag{2.38}$$

Throughout the paper we use the convention that when a sum is empty, its value is 0. Thus the original complex BVP summarizing the necessary conditions for the optimal program is reduced to the much simpler BVP

$$\dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} \left[\frac{bct}{1-b} + \theta_j D + \bar{K}_j\right]^{-(a/b)} \qquad (T_{j-1} < t < T_j) \tag{2.39}$$

subject to the boundary conditions (2.6) for $j = 1, \cdots, J+1$, with $T_0 = 0$, $T_{J+1} = \infty$, $D(\infty) = 0$, and with $k(t)$ and $r(t)$ given in terms of $D(t)$ by (2.37) and (2.25) (or (2.4)), respectively. Even without solving this simpler problem, our reduction has already yielded the remarkable result that, independent of the number of grades of deposit, *the maximum sustainable per head consumption rate over an infinite horizon is achieved by steering the economy along the "optimal trajectory", $(1-b)k^a r^b = c$, in the capital-resource flow space. This trajectory was found in* [3] *(by a conceptually different argument) to be the path of maximum capital accumulation for the particular consumption rate c, and is of course the optimal growth path for the two-grade resource problem obtained there.*

For the solution of the reduced BVP, we may heuristically regard the BVP of the initial phase as one for determining $D(t)$, $k(t)$, $r(t)$ and T_1 for $0 < t < T_1$ with c as an unknown parameter. This solution gives us $T_1(c)$ to be used in the BVP of phase 2 for determining $D(t)$, $k(t)$, $r(t)$ and T_2 for $T_1 < t < T_2$, still with c as an unknown parameter. $T_2(c)$ will then be used in the BVP of phase 3, etc. Finally, the $(J+1)$th (semi-infinite) phase uses $T_J(c)$ from the Jth phase to determine $D(t)$, $k(t)$, $r(t)$ and c for $T_J(c) < t < \infty$.

3. Numerical solutions for a single-grade deposit. Before we discuss the numerical procedure for the general problem of a multi-grade resource deposit, we consider, in this section, the single-grade deposit case (with $\theta =$ constant for all $t > 0$) to illustrate how we handle the semi-infinite domain in our numerical scheme and to investigate the effects of high extraction costs on the solution.

The problem has been reduced in § 2 to

$$\dot{D} = -\left[\frac{c}{1-b}\right]^{1/b} \left[\frac{bct}{1-b} + \theta D - \theta \bar{D} + \bar{k}_0\right]^{-(a/b)} \qquad (0 < t < \infty), \tag{3.1}$$

$$D(0) = \bar{D}, \qquad D(\infty) = 0. \tag{3.2, 3}$$

It was pointed out in [3] that (3.1) (or the equivalent equation for $r(t)$) admits an exact solution in the form of a quadrature. However, the quadrature has to be evaluated numerically, and we might as well obtain a numerical solution of (3.1)–(3.3) directly. While it is possible to integrate the initial value problem (3.1) and (3.2) for a fixed c, and

then to iterate on c until (3.3) is satisfied [3], we prefer a different approach which takes advantage of an available BVP solver, COLSYS. A brief description of COLSYS can be found in the Appendix.

To use COLSYS, the consumption rate will be treated as a function of time, and an auxiliary differential equation

$$\dot{c} = 0 \tag{3.4}$$

is introduced to stipulate the fact that c is really independent of time. The system (3.1)–(3.4) now defines a standard two point BVP, and the BVP solver, COLSYS, will be applicable to this problem once we decide how to handle the semi-infinite interval.

Next we transform the semi-infinite domain $[0, \infty)$ into a finite interval $(0, 1]$ by making a nonlinear change of independent variable

$$x = \left(\frac{1}{1+t}\right)^{\sigma} \tag{3.5}$$

for some constant $\sigma > 0$. In terms of x, the BVP takes the form

$$\frac{dD}{dx} = x^{-(\sigma+1)/\sigma}\left[\frac{c}{1-b}\right]^{1/b}\frac{1}{\sigma}\left[\frac{bc}{1-b}(x^{-(1/\sigma)}-1)+\theta(D-\bar{D})+\bar{k}_0\right]^{-(a/b)} \tag{3.6}$$

$$(0 < x < 1)$$

$$\frac{dc}{dx} = 0 \tag{3.7}$$

with the boundary conditions

$$D(0) = 0, \qquad D(1) = \bar{D}. \tag{3.8, 9}$$

The second-order transformed problem (3.6)–(3.9) is now in standard form for COLSYS.

It should be noted that the mapping of the semi-infinite interval $[0, \infty)$ onto $(0, 1]$ by (3.5) is at the expense of a singular point in the ODE (3.6) at $x = 0$. Fortunately, such singularity poses no problem for COLSYS in this context, provided dD/dx is bounded there. For (3.6), dD/dx is bounded if $-((\sigma+1)/\sigma)+a/(b\sigma) \geqq 0$, or σ should satisfy

$$\sigma \leqq \frac{a-b}{b}. \tag{3.10}$$

In practice, σ should not be too small in order to avoid loss of significant digits in the right side of (3.6).

The frequently used procedure for handling a semi-infinite interval is simply to cut it at some finite point L, with L "large enough", and to solve the problem on $[0, L]$ with the boundary conditions, originally given at ∞, imposed at $t = L$. An adequate value for L is found experimentally and depends on the rate at which the solution components approach their asymptotic values. Here, however, the slow decay of $D(t)$ would necessitate taking extremely large values of L, particularly when $b \geqq a/2$, or when θ is relatively large. In fact, in [3] the asymptotic expression of $D(t)$ had to be used to provide a better approximation of the boundary condition at L, thus making the size of L manageable (but still very large) for one-grade resource problems. The transformation (3.5) together with COLSYS resolves this numerical difficulty in an automatic, elegant way.

Accurate numerical solutions for the maximum sustainable per head consumption rates c of the single-grade resource problem for several sets of parameter values have

been obtained by the method described above and are shown in Table 1 to illustrate the efficiency of our approach to the handling of the semi-infinite interval. (The corresponding $r(0)$ is obtained from (2.25).) For all runs reported in this paper, we use $n = 4$ collocation points per element, a tolerance tol $= 10^{-5}$ on all solution components and a uniform initial mesh of $N = 5$ elements. For all cases, unless otherwise stated, the initial guess for the nonlinear iteration consists of $c = 1.2$ and a linear interpolant of the boundary conditions for D. Also we fix $\sigma = 0.3$ if $b = .05$; $\sigma = 1/3$ otherwise[2].

TABLE 1
Single-grade resource ($a = 0.2$, $\bar{k}_0 = 2.4$, $\bar{D} = 50$).

Case	b	θ	c	E	Eest	N	CPU
(1)	.05	0	1.21287	.16–10	.15–9	20	0.7
(2)	.05	.03	1.16540		.40–9	20	0.8
(3)	.05	.09	1.10742		.51–10	40	1.3
(4)	.10	0	1.18618	.30–9	.28–8	12	0.4
(5)	.10	.03	1.15741		.27–10	20	0.4
(6)	.10	.09	1.09629		.34–9	20	0.8
(7)	.15	0	1.05198	.16–9	.15–8	10	0.3
(8)	.15	.03	1.04384		.67–9	10	0.3
(9)	.15	.09	1.02466		.80–8	10	0.3

For all cases in Table 1, we fix $a = 0.2$, $\bar{k}_0 = 2.4$ and $\bar{D} = 50$. Values of the maximum per head consumption rate c for various values of b and θ are tabulated, together with the estimated error in c (under "Eest"), the final mesh size ("N") and the computer run time in seconds ("CPU"). For the special case $\theta = 0$, the exact solution is known [1], [3] and the exact error is listed under "E". The notation $.\alpha - \beta$ means $.\alpha \times 10^{-\beta}$. All results reported here were run on an Amdahl 470 V/6-II computer using double precision (14 hexadecimal digits).

From Table 1, we see that the values of c are determined very accurately and efficiently by COLSYS. In fact these values are much more accurate than the 5 digits sought. This is because the solution component c is more stable than D. Also for c the estimate (A.4) applies. Compared to other solution methods for handling the semi-infinite interval, our experience with the cases reported here and others indicates that the transformation (3.5) together with COLSYS is superior, particularly for $b \geqq a/2$. For example, while the results for the same cases obtained in [3] agree with those given in Table 1 to at least four significant figures (and six significant figures for $b = 0.05$ cases), the efficiency of the method used in [3] varies with b. To achieve the same accuracy for cases with relatively large b in [3] required more iterations and a larger terminal time L (where we prescribed the expected asymptotic behavior). While it is possible to reduce L by using more terms in the asymptotic boundary value, the method developed here is more attractive in that even without any asymptotic analysis, the CPU time required for all cases considered does not vary by an order of magnitude.

For a given set of parameter values for a, b, θ and $\bar{k}_0$, it follows from a result of [3] that the sustainable per head consumption rate c is bounded by

$$c_{\max} = (1-b)\bar{k}_0^{a/(1-b)}\left(\frac{b}{\theta}\right)^{b/(1-b)}. \tag{3.11}$$

(It would not be possible to maintain the consumption with any finite amount of resources otherwise.) The actual solution for c is of course determined by the size of the

[2] This choice of σ makes $-(\sigma+1)/\sigma + a/(b\sigma)$ a nonnegative integer.

resource deposit; the larger the deposit, the closer c is to $c_{\max}$. However, the value $c_{\max}$ can not be attained for it corresponds to a constant resource extraction rate which would exhaust the finite resource deposit in finite time [3]. Table 2 shows how close the solution c is to $c_{\max}$ for cases with a high resource extraction cost and a relatively large amount of resource. More importantly, the results indicate that a larger deposit has very little effect on c in this range of parameter values since $c_{\max}$ is independent of $\bar{D}$ (see last two cases in Table 2). For larger values of b (or larger b/θ ratios) however, c is considerably smaller than $c_{\max}$ (e.g., in the last two cases of Table 1 as well as the corresponding results in [3]) and is expected to increase with $\bar{D}$.

The limitation on the parameter values for the purpose of optimal growth with a sustainable per head consumption rate may be viewed from still another perspective. For a given set of values of a, b and $\bar{k}_0$ and a desired level of per head consumption rate c, to sustain c for the whole future imposes an upper bound on θ,

$$\theta < \theta_{\max} \equiv b\bar{k}_0^{a/b}\left(\frac{1-b}{c}\right)^{(1-b)/b}. \tag{3.12}$$

For the cases in Table 2, the actual values for θ are extremely close to this bound.

TABLE 2

*Effect of high extraction costs in single-grade resource growth** ($a = 0.2$, $b = 0.05$, $\bar{k}_0 = 2.4$).

Case	$\bar{D}$	θ	$c_{\max}$	c	$\theta_{\max}$	CPU	N
(1)	50	.09	1.1075	1.1074	.09008	0.9	14
(2)	50	.10	1.1013	1.1013	.10004	0.8	16
(3)	50	.11	1.0958	1.0958	.11002	1.0	16
(4)	50	.12	1.0908	1.0908	.12001	0.8	16
(5)	50	.15	1.0781	1.0781	.15000	1.3	28
(6)	50	.20	1.0619	1.0619	.20000	2.1	18
(7)	50	.25	1.0495	1.0495	.25000	2.3	18
(8)	100	.25	1.0495	1.0495	.25000	3.1	20

* The initial guess $D \equiv \bar{D}$ was used for the nonlinear iterations in these cases. Also for $\theta \geqq .12$ a larger initial value for c is used.

4. Numerical solutions for the multi-grade resource problem. To facilitate an efficient solution of the (reduced) BVP for optimal growth with a multi-grade resource, we further transform (2.39) and (2.6) into a form suitable for the application of COLSYS. To avoid working with unknown subintervals (T_{j-1}, T_j), $j = 1, 2, \cdots, J+1$ (with $T_0 = 0$ and $T_{J+1} = \infty$), we map each of these subintervals onto (0, 1). Evidently, the switch points T_j, $j = 1, \cdots, J$, are mapped into boundary points and the continuity conditions across them become boundary conditions. In order to have only separated boundary conditions, we let

$$x = \begin{cases} \dfrac{t - T_{j-1}}{T_j - T_{j-1}} & \text{(for odd } j \leqq J\text{)}, \\ \dfrac{T_j - t}{T_j - T_{j-1}} & \text{(for even } j \leqq J\text{)}, \\ 1 - \left(\dfrac{T_J}{t}\right)^{\sigma} & \text{(for odd } j = J+1\text{)}, \\ \left(\dfrac{T_J}{t}\right)^{\sigma} & \text{(for even } j = J+1\text{)} \end{cases} \tag{4.1}$$

for $T_{j-1} < t < T_j$ with $0 < \sigma \leqq (a-b)/b$. The change of variable in the last subinterval (T_J, ∞) is again a transformation of the type (3.5) for some constant σ. With $D_j(x) \equiv D(t)$ in the subinterval $T_{j-1} < t < T_j$ and $(\ )' \equiv d(\ )/dx$, the differential equations (2.39) become

$$D_j' = \begin{cases} -\left(\dfrac{c}{1-b}\right)^{1/b}(T_j - T_{j-1})\left[\dfrac{bc}{1-b}\{T_{j-1} + x(T_j - T_{j-1})\} + \theta_j D_j + \bar{K}_j\right]^{-(a/b)} & (j \text{ odd}), \\ \left(\dfrac{c}{1-b}\right)^{1/b}(T_j - T_{j-1})\left[\dfrac{bc}{1-b}\{T_j - x(T_j - T_{j-1})\} + \theta_j D_j + \bar{K}_j\right]^{-(a/b)} & (j \text{ even}) \end{cases} \tag{4.2}$$

for $1 \leqq j \leqq J$ and

$$D_{J+1}' = \begin{cases} -\left(\dfrac{c}{1-b}\right)^{1/b}\dfrac{T_J}{\sigma}(1-x)^{-(1+\sigma)/\sigma}\left[\dfrac{bc}{1-b}T_J(1-x)^{-(1/\sigma)} + \theta_{J+1}D_{J+1} + \bar{K}_{J+1}\right]^{-(a/b)} & (J+1 \text{ odd}), \\ \left(\dfrac{c}{1-b}\right)^{1/b}\dfrac{T_J}{\sigma}x^{-(1+\sigma)/\sigma}\left[\dfrac{bc}{1-b}T_J x^{-(1/\sigma)} + \theta_{J+1}D_{J+1} + \bar{K}_{J+1}\right]^{-(a/b)} & (J+1 \text{ even}), \end{cases} \tag{4.3}$$

where $\bar{K}_j$, $j = 1, \cdots, J+1$ are given by (2.38). The switch points T_j, $j = 1, \cdots, J$ now appear as unknown parameters in the system of ODE's (4.2) and (4.3). To apply COLSYS, we add one ODE for each T_j and one ODE for the unknown constant c as in § 3:

$$T_j' = 0 \qquad (1 \leqq j \leqq J), \tag{4.4}$$

$$c' = 0. \tag{4.5}$$

Equations (4.2)–(4.5) are $2(J+1)$ first-order ODE's for the $2(J+1)$ unknowns D_j $(1 \leqq j \leqq J+1)$, T_j $(1 \leqq j \leqq J)$ and c. For them, we have the following $2(J+1)$ boundary conditions from (2.28) and (2.29):

$$D_j(0) = \begin{cases} \sum_{i=j}^{J+1} \bar{D}_i & (j \text{ odd}), \\ \sum_{i=j+1}^{J+1} \bar{D}_i & (j \text{ even}), \end{cases} \qquad (1 \leqq j \leqq J+1), \tag{4.6}$$

$$D_j(1) = \begin{cases} \sum_{i=j+1}^{J+1} \bar{D}_i & (j \text{ odd}), \\ \sum_{i=j}^{J+1} \bar{D}_i & (j \text{ even}), \end{cases} \qquad (1 \leqq j \leqq J+1). \tag{4.7}$$

The BVP defined by (4.2)–(4.7) on the known interval (0, 1) is in a form suitable for the application of COLSYS.

The above method for the solution of the constant consumption rate growth problem with a multi-grade exhaustible resource has been used to generate accurate solutions (to five significant figures) for all the two-grade resource problems studied in [3] and two others not reported there. It is known that the results for c obtained in [3] are also good to five significant figures, but the same can not be said about the results for T_1 and $\bar{k}_1$, the switchover time and the capital stock accumulated at that point. (Using

only the leading term asymptotic behavior for the boundary condition at the finite terminal time L, only two significant figure accuracy for T_1 was assured in some cases, mostly those with $b = 0.15$, by the maximum value of L allowed before the onset of error accumulation.) The results in Table 3 of this report show that the values for $\bar{k}_1$

TABLE 3
*Maximum per head consumption rate and switchover time for a two-grade resource deposit**
($a = .2$, $\theta_2 = .09$)

Case	b	$\bar{k}_0$	$\bar{D}_1$	$\bar{D}_2$	θ_1	c	T_1	$\bar{k}_1 = k(T_1)$	CPU	N
(1)	.05	2.4	10	50	0	1.1587	9.4760	2.9779	1.8	32
(2)	.05	2.4	10	50	.03	1.1406	11.030	2.7621	1.3	18
(3)	.10	2.4	10	50	0	1.1560	6.3030	3.2096	0.5	10
(4)	.10	2.4	10	50	.03	1.1386	6.6762	2.9446	1.0	20
(5)	.15	2.4	10	50	0	1.0743	10.096	4.3140	0.5	10
(6)	.15	2.4	10	50	.03	1.0674	9.8405	3.9535	0.5	10
(7)	.05	2.4	10	25	0	1.1566	10.015	3.0097	0.9	20
(8)	.05	2.4	10	25	.03	1.1392	11.489	2.7889	1.0	20
(9)	.10	2.4	10	25	0	1.1209	9.6156	3.5976	0.5	10
(10)	.10	2.4	10	25	.03	1.1080	9.8154	3.3084	0.6	10
(11)	.15	2.4	10	25	0	.98496	24.841	6.7177	0.6	10
(12)	.15	2.4	10	25	.03	.98053	24.014	6.2552	0.6	10
(13)	.05	2.4	50	50	0	1.2291	24.029	3.9545	1.0	20
(14)	.05	2.4	50	50	.03	1.1693	35.914	3.1102	1.9	28
(15)	.05	4.8	10	50	0	1.3154	10.389	5.5193	1.8	20
(16)	.05	4.8	10	50	.03	1.3036	11.325	5.2771	1.8	20
(17)	.10	4.8	10	50	0	1.2764	8.8278	6.0519	0.6	10
(18)	.10	4.8	10	50	.03	1.2675	8.9836	5.7652	0.5	10
(19)	.15	4.8	10	50	0	1.1254	18.391	8.4525	0.5	10
(20)	.15	4.8	10	50	.03	1.1222	18.106	8.0854	0.5	10

* The initial guess $T_1 \equiv 10$ was used for the nonlinear iterations here, with $c \equiv 1.2$ and D_j linear interpolants of their boundary conditions, $j = 1, 2$.

obtained in [3] are accurate to at least four significant figures except for one case with a 0.4% error. In contrast, the values for T_1 obtained in [3] are not as accurate though the percentage error is still less than 0.5% in all cases. Evidently, the method of solution developed here has enabled us to achieve a degree of accuracy in the numerical solution for the two-grade resource not practical hitherto. More importantly, the high accuracy is attained with no asymptotic analysis (which is needed in [3]), minimal programming and a relatively small amount of computing time for all cases investigated as shown under the CPU time column of Table 3. The effects of the various input parameters as suggested by Table 3 have already been analyzed in [3] and will not be repeated here.

With the same method, we can, for the first time, generate accurate numerical solutions for maximum constant consumption rate problems with a resource deposit of more than two grades. Again, the solution process requires minimal programming and a relatively small amount of computing time to achieve the desired accuracy (and of course no asymptotic analyses). To illustrate, we report in Table 4 some sample calculations for three-grade resource problems. In all fifteen cases, we have kept $a = 0.2$, $\bar{k}_0 = 2.4$, $\bar{D}_1 = 10$ and $\bar{D}_2 = 25$. The CPU seconds required for a single case (to achieve a five significant figure accuracy) range from 1.0 to 10.7, and increase with more low grade deposit $\bar{D}_3$ or higher unit extraction costs θ_j, $j = 1, 2, 3$. The CPU time is lower when $b = 0.1$ or $b = 0.15$ and is higher for $b = 0.05$.

TABLE 4

*Maximum per head consumption rate and switchover times for a three-grade resource deposit** ($a = .2$, $\bar{k}_0 = 2.4$, $\bar{D}_1 = 10$, $\bar{D}_2 = 25$)

Case	b	$\bar{D}_3$	θ_1	θ_2	θ_3	c	T_1	T_2	$k(T_1)$	$k(T_2)$	CPU	N
(1)	.05	50	0	.03	.09	1.1864	4.9364	26.981	2.7082	3.3348	2.3	18
(2)	.10	50	0	.03	.09	1.2090	3.6195	19.070	2.8862	4.2118	1.1	10
(3)	.15	50	0	.03	.09	1.1451	5.7698	46.515	3.5660	11.050	1.3	10
(4)	.05	100	0	.03	.09	1.1865	4.9307	26.892	2.7079	3.3293	4.7	36
(5)	.10	100	0	.03	.09	1.2202	3.2491	15.804	2.8405	3.7926	2.0	20
(6)	.15	100	0	.03	.09	1.2226	3.4120	18.347	3.1362	5.6086	1.0	10
(7)	.05	50	.03	.06	.09	1.1498	8.5208	37.518	2.6157	2.8705	4.3	36
(8)	.10	50	.03	.06	.09	1.1668	4.8616	21.508	2.7303	3.3884	1.1	10
(9)	.15	50	.03	.06	.09	1.1289	5.9716	41.451	3.2896	8.8578	1.3	10
(10)	.05	50	.03	.09	.25	1.1400	11.237	61.743	2.7742	3.5545	6.3	20
(11)	.10	50	.03	.09	.25	1.1356	6.9199	37.145	2.9731	4.5369	2.3	20
(12)	.15	50	.03	.09	.25	1.1139	6.6984	45.423	3.4167	8.7789	1.3	10
(13)	.05	100	.03	.09	.25	1.1400	11.237	61.743	2.7732	3.5545	10.7	48
(14)	.10	100	.03	.09	.25	1.1358	6.9055	36.888	2.9715	4.5051	4.4	20
(15)	.15	100	.03	.09	.25	1.1440	5.3403	27.623	3.1781	5.4265	1.4	10

* The initial guess, $T_1 \equiv 8$, $T_2 \equiv 16$, $c \equiv 1.2$ and linear interpolants of boundary conditions for D_j, $j = 1, 2, 3$, was used for all cases but No. 10, 13, 14. For the latter 3 cases, the initial guess was changed to $T_2 \equiv 50$, $c \equiv 2.0$, $D_3 \equiv \bar{D}_3$.

By comparing cases (1)–(3) with cases (4)–(6), respectively, we see that doubling the low grade resource deposit $\bar{D}_3$ hardly affects the $b = 0.05$ case, changes the consumption rate level by only 1% in the $b = 0.1$ case but changes c by about 7% in the $b = 0.15$ case. These results conform with our observations in § 3 for single-grade resource problems; evidently, c is still considerably less than $c_{\max}$ for the $b = 0.15$ case so that it can be increased by a larger low grade resource deposit $\bar{D}_3$. To achieve the 7% change in c by doubling $\bar{D}_3$ in this case involves a substantial change in the growth program. It allows the switch-over times T_1 and T_2 to go from 5.7698 and 46.515 down to 3.4120 and 18.347, respectively. Correspondingly, it allows $\bar{k}_1$ and $\bar{k}_2$ to go from 3.5660 and 11.050 down to 3.1362 and 5.6086. Evidently, with more low-grade resource, the economy requires less initial capital for the last phase of the program covering the semi-infinite interval (T_2, ∞) so that the higher grade deposits can be used up sooner to achieve a higher per head consumption rate c in all three phases of the program. In contrast, c is very nearly $c_{\max}$ in cases (10), (11), (13) and (14) with very high extraction costs; we can not increase c much by doubling $\bar{D}_3$.

Cases (1)–(3) in conjunction with cases (7)–(9) demonstrate the effects of an increase of the unit extraction costs θ_1 and θ_2 for the two high grade deposits. Higher values of θ_1 and θ_2 use up more output at each instant. To keep down the extraction cost per unit time, the new optimal programs of cases (7)–(9) reduce c to slow down the resource extraction and to build up a capital stock at the switch points comparable to those for cases (1)–(3), respectively. A good size capital stock in turn moderates the reduction in consumption level.

Finally, cases (10)–(12) in conjunction with cases (7)–(9) show that an increase in θ_2 and θ_3 gives rise to analogous effects.

By comparing the CPU time for related single-grade, two-grade and three-grade problems in Tables 1–4, e.g., cases (6), (4) and (2) in Tables 1, 3 and 4, respectively, we see that CPU time increases only moderately with the number of resource grades. This suggests that our solution procedure is still practical even when J is larger.

Appendix. The general purpose BVP solver COLSYS [4], [5] is based on spline collocation at Gaussian points and is capable of handling mixed order systems of nonlinear multi-point BVP's. Here it is sufficient to consider a two-point BVP for a first order system,

$$\mathbf{z}' = \mathbf{f}(x, \mathbf{z}), \qquad x \in (a, b), \tag{A.1}$$

$$\mathbf{g}^1(\mathbf{z}(a)) = 0, \qquad \mathbf{g}^2(\mathbf{z}(b)) = 0, \tag{A.2}$$

where $\mathbf{z}, \mathbf{f}$ are vector functions of order m, $\mathbf{g}^1$ is of order m_1 and $\mathbf{g}^2$ is of order $m_2 = m - m_1$. The functions $\mathbf{f}$, $\mathbf{g}^1$ and $\mathbf{g}^2$ may be nonlinear.

In COLSYS, the problem (A.1)–(A.2) is solved on a sequence of meshes, until user-specified error tolerances are satisfied. For a specific mesh $a = x_0 < x_1 < \cdots < x_N = b$, with $h_i = x_i - x_{i-1}$, $h = \max_{1 \leqq i \leqq N} h_i$, and an integer $n > 1$, the collocation solution $\mathbf{v}(x) = (v_1, \cdots, v_m)$ is a piecewise polynomial vector function: for each j, $1 \leqq j \leqq m$, $v_j \in C[a, b]$ is a polynomial of degree $\leqq n$ on each element (x_{i-1}, x_i), $i = 1, \cdots, N$. The piecewise polynomial solutions are represented in terms of a B-spline basis. The approximate solution is determined by requiring that it satisfy (A.2) and the differential equation (A.1) at the images of the n zeros of the appropriate Legendre polynomial in each element. Under sufficient smoothness conditions, the error in $\mathbf{v}$ for $x \in [x_i, x_{i+1})$ is given by

$$z_j(x) - v_j(x) = K z_j^{(n+1)}(x_i) h_i^{n+1} + O(h^{n+2}), \tag{A.3}$$

where K is a known bounded function of x, and at the mesh point x_i,

$$z_j(x_i) - v_j(x_i) = O(h^{2n}), \qquad (j = 1, \cdots, m) \quad (i = 0, 1, \cdots, N). \tag{A.4}$$

Expression (A.3) is used both for estimating the error accurately via mesh halving and for automatic new mesh selection. For nonlinear problems, the damped Newton's method is used for the first mesh to find the collocation solution, and modified Newton iterations with a fixed Jacobian are performed for subsequent refined meshes. Full details of the code can be found in [4], [5] and references therein.

REFERENCES

[1] R. M. SOLOW, *Intergenerational equity and exhaustible resources*, Review of Economic Studies (Symposium Issue), 41, 1974, pp. 29–45.

[2] R. M. SOLOW AND F. Y. M. WAN, *Extraction costs in the theory of exhaustible resources*, Bell J. of Econom., 7 (1976), pp. 359–370.

[3] F. Y. M. WAN, *Constant sustainable consumption rate in optimal growth with exhaustible resources*, Studies in Appl. Math., 58 (1979).

[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] ———, *Collocation software for boundary value ODE's*, IEEE Trans. of Math. Software, to appear.

[6] A. BRYSON AND Y. C. HO, *Applied Optimal Control*, Ginn and Co., Waltham, MA, 1969.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 1, March 1980

0196-5204/80/0101-0009 $01.00/0

FFT AS NESTED MULTIPLICATION, WITH A TWIST*

CARL DE BOOR†

Abstract. A simple, yet complete and detailed description of the fast Fourier transform for general N is given with the aim of making the underlying idea quite apparent. To help with this didactic goal, a simple twist, i.e., a shifting of information from rows to columns during the calculations, is introduced which allows us to give a simple meaning to intermediate results and assures that the final results need no further reordering.

Key words. FFT = fast Fourier transform, nested multiplication

1. Introduction. The discrete Fourier transform (DFT) $\hat{\mathbf{z}} = F_N \mathbf{z}$ of an N-vector $\mathbf{z}$ is given by the rule

$$\hat{z}_\nu = \sum_{n=1}^{N} z_n \omega_N^{(\nu-1)(n-1)}, \qquad \nu = 1, \cdots, N, \tag{1}$$

with

$$\omega_N = \exp(-2\pi\sqrt{-1}/N) \tag{2}$$

a principal Nth root of unity. Thus, $\hat{z}_\nu$ is given as the value of a polynomial of degree $<N$ at the point $\omega_N^{\nu-1}$ and can therefore be calculated, by nested multiplication, in N operations. Here, I follow Cooley and Tukey [1] in counting a complex multiplication followed by a complex addition as one *operation.*

In the last twenty years, various forms of a fast Fourier transform (FFT) have become popular. The various algorithms have in common that they produce the DFT on N points in about $N \ln(N)$ rather than N^2 operations. See Winograd [10] for the latest developments. But, while these ideas, notably through Cooley and Tukey [1], have found wide application in computations, their didactic treatment has left something to be desired.

In a recent article [7], H. R. Schwarz attempts, as he says, to remove the mystical aspect which the FFT has for many people. He does this by describing the FFT in terms of a factorization of the transformation matrix, an idea which he ascribes to Theilheimer [9] but which occurs already in Good [5], where a FFT different from that of Cooley and Tukey is given. A factorization of the transformation matrix is also the basic idea on which Glassman [4] builds his FFT, and Drubin [2] has refined this further; see Ferguson [3] for a lucid description and a simple Fortran program.

By contrast, I want to give here what I believe to be a simple description of the FFT for a general N in terms of *nested multiplication.* Certainly, Cooley and Tukey [1] thought of the FFT in these terms.

2. The case of two factors. Suppose that $N = PQ$ for two integers P and Q greater than 1. Think of the N-vector $\mathbf{z}$ as stored FORTRAN fashion in a one-dimensional array. Then we can interpret that array also FORTRAN fashion as a two-dimensional array Z of dimension (P, Q). This means that

$$Z(p, q) = z_{p+P(q-1)}, \qquad p = 1, \cdots, P, \quad q = 1, \cdots, Q. \tag{3}$$

* Received by the editors June 15, 1979, and in revised form November 8, 1979.

† Mathematics Research Center, University of Wisconsin, Madison, Wisconsin 53706. This research was sponsored by the United States Army under Contract DAAG29-75-C-0024.

Correspondingly, factor the sum (1) for $\hat{z}_\nu$ into a double sum,

$$\hat{z}_\nu = \sum_{p=1}^{P} \sum_{q=1}^{Q} Z(p,q)\ \omega_N^{(\nu-1)[p-1+P(q-1)]}$$
$$= \sum_{p=1}^{P} \left[\sum_{q=1}^{Q} Z(p,q)\ \omega_Q^{(\nu-1)(q-1)} \right] \omega_N^{(\nu-1)(p-1)}.$$

Here, we have made use of the fact that

$$\omega_N^P = \omega_Q.$$

This makes apparent the crucial fact that *the inner sum* in the last right-hand side *is Q-periodic in* ν; i.e., replacing ν by $\nu+Q$ does not change its value, due to the fact that $\omega_Q^Q = 1$. This means that we need only calculate this sum for $\nu = 1, \cdots, Q$ (and for each p). Thus, for each $p = 1, \cdots, P$, we calculate, from the Q-vector $Z(p, \cdot)$, the Q-vector whose entries are the numbers

$$\sum_{q=1}^{Q} Z(p,q)\ \omega_Q^{(\nu-1)(q-1)}, \qquad \nu = 1, \cdots, Q; \tag{4}$$

i.e., *we calculate the DFT* $F_Q Z(p, \cdot)$, $p = 1, \cdots, P$, at a total cost of $P \cdot Q^2 = N \cdot Q$ operations.

Now, we could store the transform of $Z(p, \cdot)$ over $Z(p, \cdot)$. But in anticipation of further developments, we choose to store the transform $F_Q Z(p, \cdot)$ in $Z_1(\cdot, p)$, where Z_1 is a two dimensional array of size (Q, P), rather than (P, Q).

With this, the calculation of $\hat{z}_\nu$ is reduced to the evaluation of the sum

$$\hat{z}_\nu = \sum_{p=1}^{P} Z_1(\nu_Q, p)\ \omega_N^{(\nu-1)(p-1)}, \qquad \nu = 1, \cdots, N. \tag{5}$$

Here, we have used the notation ν_Q to indicate the integer between 1 and Q for which $\nu - \nu_Q$ is divisible by Q. At this point, it becomes convenient to think of the one-dimensional array which is to contain the N-vector $\hat{\mathbf{z}}$ equivalently as a two-dimensional array Z_0, of size (Q, P). This means that

$$\hat{z}_{\nu+Q(\mu-1)} = Z_0(\nu, \mu) \quad \text{for } \nu = 1, \cdots, Q, \qquad \mu = 1, \cdots, P. \tag{6}$$

With this, (5) can be written equivalently as

$$Z_0(\nu, \mu) = \sum_{p=1}^{P} Z_1(\nu, p)\ \omega_N^{[\nu-1+Q(\mu-1)](p-1)}, \qquad \nu = 1, \cdots, Q, \quad \mu = 1, \cdots, P.$$

Here, the right-hand side is a polynomial of degree $<P$ in the quantity $\omega_N^{\nu-1+Q(\mu-1)}$. This quantity can be generated as one goes along, as in the following convenient arrangement of the calculations:

(7)

$x := 1$

for $\mu = 1, \cdots, P$, **do**:

  for $\nu = 1, \cdots, Q$, **do**:

    $Z_0(\nu, \mu) := \sum_{p=1}^{P} Z_1(\nu, p) x^{p-1}$

  $x := x \cdot \omega_N$.

The sum in the innermost loop is, of course, to be evaluated by nested multiplication. The total cost of this step is then $Q \cdot P^2 = N \cdot P$ operations (if we neglect the N

multiplications needed to generate the various x's). In this way, we have obtained in Z_0 the discrete Fourier transform $\hat{\mathbf{z}}$ of $\mathbf{z}$ at a cost of only $N(P+Q)$ rather than N^2 operations.

3. The general case. "It is easy to see how successive applications of the above procedure, starting with its application to (4), give an m-step algorithm requiring

$$T = N(P_1 + P_2 + \cdots + P_m)$$

operations, where

$$N = P_1 \cdot P_2 \cdot \cdots \cdot P_m." \tag{8}$$

So say Cooley and Tukey [1] (except for a change in symbols and equation numbers). In effect, they point out that the first step of the calculations above consists in forming the DFT of various Q-vectors. Hence, if Q itself is the product of two integers greater than 1, this calculation can be carried out in fewer than Q^2 operations by applying the same procedure to it, etc. The actual implementation of this idea may not be immediately obvious, though. For this reason, I now discuss a slightly different (and novel) view, according to which the entire transform can be effected by m applications of a slightly enlarged version of (7).

The basic idea is to interpret the storage arrays for the various N-vectors involved in various ways as multidimensional arrays and to shift information appropriately from "rows to columns" as we did earlier when storing the DFT of the row $Z(p, \cdot)$ of Z in the column $Z_1(\cdot, p)$ of Z_1. For this, I need some notation to indicate that a given one-dimensional array is being considered equivalently as a two- or three-dimensional array.

If Z is a one-dimensional array of length N, then Z^A denotes the equivalent two-dimensional array of dimension $(A, N/A)$, and $Z^{A,B}$ denotes the equivalent three-dimensional array of dimension $(A, B, N/(AB))$. Thus

$$\begin{aligned} Z^{A,B}(a,b,c) &= Z^A(a, b+B(c-1)) = Z^{AB}(a+A(b-1), c) \\ &= Z(a+A(b-1+B(c-1))). \end{aligned} \tag{9}$$

Let now Z be a one-dimensional array containing $\mathbf{z}$, as before, and, for $k = 0, \cdots, m$, let Z_k be a one-dimensional array satisfying

$$Z_k^A(\cdot, c) = F_A Z^{BP}(c, \cdot), \qquad c = 1, \cdots, BP \tag{10}$$

with

$$B := B_k := P_1 \cdot \cdots \cdot P_{k-1}, \quad P := P_k, \quad A := A_k := P_m \cdot \cdots \cdot P_{k+1}. \tag{11}$$

Then $Z_m = Z$, and Z_0 contains $\hat{\mathbf{z}} = F_N \mathbf{z}$. Further, with A, P, B as given by (11), one obtains Z_{k-1} from Z_k by the following slightly extended version of (7):

$x := 1$
for $p = 1, \cdots, P$, **do**:
  for $a = 1, \cdots, A$, **do**:
    for $b = 1, \cdots, B$, **do**:
      $Z_{k-1}^{A,P}(a, p, b) := \sum_{\pi=1}^{P} Z_k^{A,B}(a, b, \pi) \cdot x^{\pi-1}$
    $x := x \cdot \omega_{AP}$.

(12)

Indeed, the algorithm produces

$$Z_{k-1}^{A,P}(a,p,b)=\sum_{\pi=1}^{P} Z_k^{A,B}(a,b,\pi)\omega_{AP}^{[a-1+A(p-1)](\pi-1)}.$$

On the other hand, (10) implies that

$$Z_k^{A,B}(\cdot\,,b,\pi)=F_A Z^{B,P}(b,\pi,\cdot)=\sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\omega_A^{(\cdot-1)(\alpha-1)}.$$

Therefore,

$$Z_{k-1}^{A,B}(a,p,b)=\sum_{\pi=1}^{P}\sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\omega_{AP}^{P(a-1)(\alpha-1)+[a-1+A(p-1)](\pi-1)}.$$

But now, since $\omega_{AP}^{AP}=1$, we may add to the exponent on the right-hand side any integer multiple of AP; this allows the conclusion that

$$Z_{k-1}^{A,P}(a,p,b)=\sum_{\pi=1}^{P}\sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\omega_{AP}^{[a-1+A(p-1)][\pi-1+P(\alpha-1)]}$$

and so proves that Z_{k-1}, as produced by (12), satisfies (10) (with k replaced by $k-1$).

This shows that the DFT $\hat{\mathbf{z}}$ is obtainable, in Z_0, by m applications of algorithm (12), starting from Z_m containing $\mathbf{z}$. Since the kth such application costs $P_kA_kB_kP_k=N\cdot P_k$ operations, the total number of required operations is indeed given by (8).

In a FORTRAN implementation of the algorithm, one would, of course, need only two arrays to play the role, in alternation, of the $m+1$ arrays $Z_m,\cdots,Z_0$. See Fig. 1.

```
      SUBROUTINE FFT ( Z1, Z2, N, INZEE )
CONSTRUCTS THE DISCRETE FOURIER TRANSFORM OF  Z1 (OR Z2) IN THE COOLEY-
C  TUKEY WAY, BUT WITH A TWIST.
      INTEGER INZEE,N,    AFTER,BEFORE,NEXT,NEXTMX,NOW,PRIME(12)
      COMPLEX Z1(N),Z2(N)
C******  I N P U T  ******
C  Z1, Z2  COMPLEX N-VECTORS
C  N  LENGTH OF  Z1  AND  Z2
C  INZEE  INTEGER INDICATING WHETHER  Z1  OR  Z2  IS TO BE TRANSFORMED
C      = 1 , TRANSFORM  Z1
C      = 2 , TRANSFORM  Z2
C******  W O R K  A R E A S  ******
C  Z1, Z2  ARE BOTH USED AS WORKARRAYS
C******  O U T P U T  ******
C  Z1  OR  Z2  CONTAINS THE DESIRED TRANSFORM (IN THE CORRECT ORDER)
C  INZEE  INTEGER INDICATING WHETHER  Z1 OR Z2  CONTAINS THE TRANSFORM,
C      = 1 , TRANSFORM IS IN  Z1
C      = 2 , TRANSFORM IS IN  Z2
C******  M E T H O D  ******
C      THE INTEGER  N  IS DIVIDED INTO ITS PRIME FACTORS (UP TO A POINT).
C  FOR EACH SUCH FACTOR  P , THE P-TRANSFORM OF APPROPRIATE P-SUBVECTORS
C  OF  Z1 (OR Z2) IS CALCULATED IN  F F T S T P  AND STORED IN A SUIT-
C  ABLE WAY  IN  Z2 (OR Z1).  SEE TEXT FOR DETAILS.
C
      DATA NEXTMX,PRIME / 12, 2,3,5,7,11,13,17,19,23,29,31,37 /
      AFTER = 1
      BEFORE = N
      NEXT = 1
```

```
C
   10 IF ((BEFORE/PRIME(NEXT))*PRIME(NEXT) .LT. BEFORE) THEN
         NEXT = NEXT + 1
         IF (NEXT .LE. NEXTMX) THEN
                                          GO TO 10
         ELSE
            NOW = BEFORE
            BEFORE = 1
         END IF
      ELSE
         NOW = PRIME(NEXT)
         BEFORE = BEFORE/PRIME(NEXT)
      END IF
C
      IF (INZEE .EQ. 1)  THEN
         CALL FFTSTP( Z1, AFTER, NOW, BEFORE, Z2 )
      ELSE
         CALL FFTSTP( Z2, AFTER, NOW, BEFORE, Z1 )
      END IF
      INZEE = 3 - INZEE
      IF (BEFORE .EQ. 1)                  RETURN
      AFTER = AFTER*NOW
                                          GO TO 10
      END
      SUBROUTINE FFTSTP ( ZIN, AFTER, NOW, BEFORE, ZOUT )
CALLED IN  F F T .
CARRIES OUT ONE STEP OF THE DISCRETE FAST FOURIER TRANSFORM.
      INTEGER AFTER,BEFORE,NOW,    IA,IB,IN,J
      REAL ANGLE,RATIO,TWOPI
      COMPLEX ZIN(AFTER,BEFORE,NOW),ZOUT(AFTER,NOW,BEFORE),    ARG,OMEGA,
     *                                                         VALUE
      DATA TWOPI / 6.2831 85307 17958 64769 /
      ANGLE = TWOPI/FLOAT(NOW*AFTER)
      OMEGA = CMPLX(COS(ANGLE),-SIN(ANGLE))
      ARG = CMPLX(1.,0.)
      DO 100 J=1,NOW
         DO 90 IA=1,AFTER
            DO 80 IB=1,BEFORE
               VALUE = ZIN(IA,IB,NOW)
               DO 70 IN=NOW-1,1,-1
   70             VALUE = VALUE*ARG + ZIN(IA,IB,IN)
   80          ZOUT(IA,J,IB) = VALUE
   90       ARG = ARG*OMEGA
  100 CONTINUE
                                          RETURN
      END
```

FIG. 1

There is no claim that the above program is competitive with the carefully constructed codes such as that of Singleton [8]. Its virtue lies chiefly in its simplicity and transparency. On the other hand, Eric Grosse [6] found that the above code, modified to give special treatment in FFTSTP for the case NOW = 2, and to avoid subroutine calls for the complex arithmetic operations, and compiled by an optimizing compiler, needed only 1.5 to 2 times as much computing time as did Singleton's program for a variety of choices of N.

Finally, the above discussion is based on the FORTRAN convention whereby multidimensional arrays are stored "column by column", i.e., with the first index running fastest. It is easy to base the discussion instead on the ALGOL convention whereby arrays are stored "row by row", i.e., with the last index running fastest.

Acknowledgment. I am grateful to Warren Ferguson for several discussions concerning fast Fourier transforms and for comments on an earlier draft. I am indebted to Eric Grosse for carrying out the comparisons mentioned above and for suggesting that the more leisurely discussion in an earlier draft be replaced by showing directly that the Z_k as generated by (12) satisfy (10).

Finally, I am very grateful to the referees for additional information which I have collected, for the reader's edification, in the following

Postscript. (i) A "twist" (though not exactly the one used here) to avoid explicit sorting was used as early as 1969 by Uhrich [15] for the case $N = 2^r$, and has meanwhile been used in the mixed radix case by Temperton [14]. To this should be added that Glassman's [4] version of the FFT is very similar to Temperton's and, in particular, contains such a "twist" as becomes evident from [3]. Finally, I found the "twist" of help in other Kronecker product calculations; see [11].

(ii) The FORTRAN program presented here could be made much more efficient if factors of 4 were used whenever possible and special coding for the cases NOW = 2, 3, 4 and 5 were used in FFTSTP. In addition, the work for the polynomial evaluation (the innermost loop in FFTSTP) can be cut in half for an odd NOW by a standard device.

(iii) Brigham's book [12] is a good reference for the algorithm and for its many applications, while Jenkins and Watts [13; pp. 313–317] contains a very intuitive description of the algorithm.

REFERENCES

[1] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[2] M. DRUBIN, *Kronecker product factorization of the FFT matrix*, IEEE Trans. Computers, C-20 (1971), pp. 590–593.

[3] W. FERGUSON, *A simple derivation of Glassman's general N fast Fourier transform*, Mathematics Research Center Tech. Summary Rep. 2029, Univ. of Wisconsin, Madison, WI, 1979.

[4] J. A. GLASSMAN, *A generalization of the fast Fourier transform*, IEEE Trans. Computers, C-19 (1970), pp. 105–116.

[5] I. J. GOOD, *The interaction algorithm and practical Fourier series*, J. Roy. Statist. Soc. Ser. B, 20 (1958), pp. 361–372; Addendum, 22 (1960), pp. 372–375.

[6] ERIC GROSSE, private communication, April 6, 1979.

[7] H. R. SCHWARZ, *Elementare Darstellung der schnellen Fouriertransformation*, Computing, 19 (1977), pp. 107–116.

[8] R. C. SINGLETON, *Algorithm 339. An Algol procedure for the fast Fourier transform with arbitrary factors*, Comm. ACM, 11 (1968), p. 776ff.

[9] F. THEILHEIMER, *A matrix version of the fast Fourier transform*, IEEE Trans. Audio Electroacoustics, 17 (1969), pp. 158–161.

[10] S. WINOGRAD, *On computing the discrete Fourier transform*, Math. Comp., 32 (1978), pp. 175–199.

[11] C. DE BOOR, *Efficient computer manipulation of tensor products*, ACM Trans. Math. Software, 5 (1979), pp. 173–182.

[12] ORAN E. BRIGHAM, *The Fast Fourier Transform*, Prentice Hall, Englewood Cliffs, NJ, 1974.

[13] G. M. JENKINS AND D. G. WATTS, *Spectral Analysis and its Applications*, Holden Day, San Francisco, CA.

[14] CLIVE TEMPERTON, *Mixed-radix fast Fourier transforms withour reordering*, ECMWF Tech. Rep. 3, European Center for Medium Range Weather Forecasting, Fitzwilliam House, Skimped Hill, Bracknell, Berkshire, United Kingdom, February 1977.

[15] MARK L. UHRICH, *Fast Fourier transforms without sorting*, IEEE Trans. Audio Electroacoustics, 17 (1969), pp. 170–172.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0001 $01.00/0

COMPUTATION OF THE INTEGRAL OF THE BIVARIATE NORMAL DISTRIBUTION OVER CONVEX POLYGONS*

A. R. DIDONATO,† M. P. JARNAGIN, JR.† AND R. K. HAGEMAN†

Abstract. A method for computing the integral of the bivariate normal density function over a convex polygon is given. A comparison with two recently published methods is made.

Key words. Probability: special processes, geometric probability applications; statistics: multivariate analysis

1. Introduction. The integration of the general bivariate normal distribution (BND) over a convex polygon is an important computation in statistics. It is used to find the probability that an event or observation occurs in a specified polygonal area where the random variations in the position of an observation can be described by a BND. One of the most important applications is in weapon evaluations. A BND often is used to describe the distribution of impact points in the firing of projectiles. The aim point of the weapon is usually taken as the mean (μ_w, μ_z) of the distribution. The standard deviations σ_w and σ_z, in the w and z directions, are due to random errors in azimuth and elevation directions for some types of weapons and range and deflection directions for other types, and these are estimated from firing data. A correlation coefficient ρ is also estimated from the data. The problem then is to determine the probability of the projectile hitting, or perhaps missing, a specified circular, elliptical, or polygonal area in the wz-plane.

Many applications also occur outside the weapons field. H. L. Crutcher [2] presents a set of 13 applications to geophysical data. Regions over which the probabilities are found are either elliptical or circular. We briefly mention three of these applications.

a. *Lunar tides.* After fitting the circular normal distribution to 40 data points on a harmonic dial of lunar tides, estimate the probability of observing a lunar tide of magnitude less than 0.05 mm Hg.
b. *Solar wind ions.* An extensive set of observations by a spacecraft provided data on ions in the solar wind. When represented as points (E', ϕ), where $E' = 1 + \log_{10} E$ with E given in KEV and ϕ is longitude in the spacecraft coordinate system, the major portion of the data is well described by an elliptical BND. Estimate the proportion of solar wind ions with combinations of energy and direction in a particular range.
c. *Tropical storms.* Using data on the paths of tropical storms that have previously passed through a particular region, predict the probability that a new storm will be within 2 degrees of latitude of city A 36 hours after it passes through the specified region.

Of course, in each of these examples, the circular or elliptical region originally used can be replaced with a polygonal one. In fact, there is often more versatility with the use of polygons. Moreover, with convex polygons of 8 vertices or fewer, the computer program for finding the probability would, in most cases, be faster than one used for the computation over an ellipse or a circle.

Many methods have been developed for carrying out the integration of the BND over polygons numerically, e.g., [1], [3], [4], [5], [6], [7], [8]. We present a method which has led to a very fast, automatic computer program.[1]

* Received by the editors May 25, 1979 and in revised form January 24, 1980.

† Naval Surface Weapons Center, Dahlgren, Virginia.

[1] Our programs are in Fortran IV for the CDC 6700. It does about 10^6 arithmetic operations per second on 14 decimal digit mantissas.

Our approach is to find the probability over an angular region and then combine an appropriate number of these to obtain the probability over a convex polygon. We define an angular region A as the semi-infinite part of the plane bounded by two intersecting directed straight lines. Of course, by this definition there are four such regions, hence it is always necessary to specify which one is needed. Computing normal probabilities over angular regions appears to have first been considered explicitly by Gideon and Gurland [6]. We briefly discuss their work in § 3. In that section, we also compare our results with those given recently by Drezner [5].

In a subsequent paper, we shall show how the present program for probabilities over convex polygons can be used to compute automatically normal probabilities over arbitrary polygons. Fortran IV listings of these programs are available upon request.

2. Normal probability over an angular region. We are interested in the numerical evaluation, for the angular region $\tilde{A}$, of

$$P(\tilde{A}) \equiv \iint_{\tilde{A}} Z(w, z, \rho)\, dw\, dz, \tag{1}$$

where

$$Z(w, z, \rho) \equiv [2\pi\sigma_w\sigma_z(1-\rho^2)^{1/2}]^{-1} \cdot \exp\left\{-\left[\left(\frac{w-\mu_w}{\sigma_w}\right)^2 - 2\rho\left(\frac{w-\mu_w}{\sigma_w}\right)\left(\frac{z-\mu_z}{\sigma_z}\right) + \left(\frac{z-\mu_z}{\sigma_z}\right)^2\right] \Big/ 2(1-\rho^2)\right\}, \tag{2}$$

with (μ_w, μ_z) the mean and

$$\begin{bmatrix} \sigma_w^2 & \rho\sigma_w\sigma_z \\ \rho\sigma_w\sigma_z & \sigma_z^2 \end{bmatrix}$$

the covariance matrix of the random variable (w, z) with correlation coefficient ρ. The well-known linear transformation

$$x = \frac{\left[\dfrac{w-\mu_w}{\sigma_w} - \rho\dfrac{z-\mu_z}{\sigma_z}\right]}{\sqrt{1-\rho^2}}, \qquad y = \frac{(z-\mu_z)}{\sigma_z}, \quad |\rho| < 1, \tag{3}$$

reduces the integrand (2) to one with circular symmetry, i.e.,

$$P(\tilde{A}) = P(A) = \frac{1}{2\pi}\iint_{A} \exp\left[-\frac{(x^2+y^2)}{2}\right] dx\, dy, \tag{4}$$

where A is also an angular region since (3) takes straight lines into straight lines. Hence, hereafter we deal only with (4) unless noted otherwise.

The angular region A can be specified by the distance R of its vertex V from the origin and the angles θ_1 and θ_2 which the two sides 1 and 2 of A make with the extension L of the straight line passing through the origin and vertex. This is shown in Fig. 1. The angles are measured positive in the counterclockwise direction about the vertex at V from the line L.

It is convenient to make several elementary coordinate transformations, the first of which takes advantage of the circular symmetry shown in (4). First we rotate the axes through the angle ψ (Fig. 1), so that the new x-axis is along OV and contains the vertex V. Then we translate the axes through a distance R, putting the origin at the vertex V or

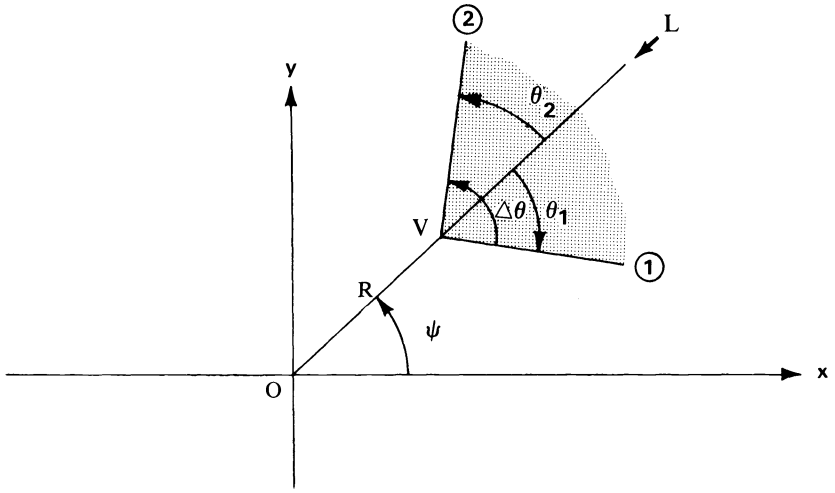


FIG. 1. *The angular region A specified by R, θ_1, θ_2.*

the point $(R, 0)$ relative to the rotated axes, and finally we introduce polar coordinates (r, θ) centered at the new origin V. If x, y are rectangular coordinates relative to the rotated axes with origin at O (see Fig. 2), and (r, θ) are these polar coordinates centered at V, the mutual relationships are given by

$$x = R + r\cos\theta, \qquad y = r\sin\theta \tag{5}$$

as shown in Fig. 2. In this case (4) becomes

$$\begin{aligned} P(A) &= \frac{1}{2\pi}\int_{\theta_1}^{\theta_2}\int_0^{\infty} \exp\left[-\frac{1}{2}(R^2 + 2rR\cos\theta + r^2)\right] r\,dr\,d\theta \\ &= \frac{1}{2\pi} e^{-R^2/2}\int_{\theta_1}^{\theta_2}\int_0^{\infty} e^{-r^2/2}\, e^{-pr} r\,dr\,d\theta, \end{aligned} \tag{6}$$

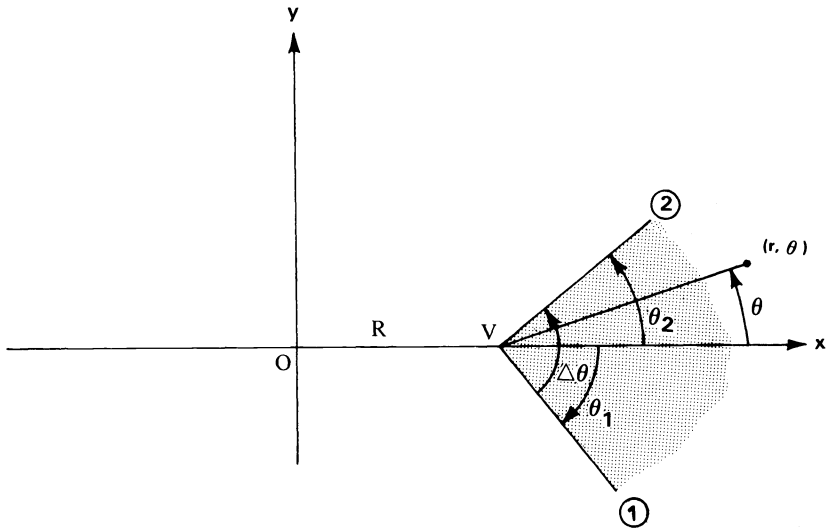


FIG. 2. *The angular region A relative to rotated axes.*

where

$$p = R\cos\theta. \tag{7}$$

An integration by parts for the integral in r yields

$$\int_0^\infty e^{-r^2/2}\, e^{-pr} r\, dr = 1 - 2u\frac{\operatorname{erfc}(u)}{z(u)}, \tag{8}$$

where

$$z(u) \equiv \frac{2}{\sqrt{\pi}}\exp(-u^2), \quad \operatorname{erfc}(u) \equiv \int_u^\infty z(t)\, dt, \quad u \equiv \frac{p}{\sqrt{2}} = \frac{R}{\sqrt{2}}\cos\theta. \tag{9}$$

Using (8) in (6) and carrying out the obvious part of the θ-integration reduces (6) to

$$P(A) = e^{-R^2/2}\left\{\frac{\theta_2 - \theta_1}{2\pi} - \frac{1}{\pi}\int_{\theta_1}^{\theta_2} u\left[\frac{\operatorname{erfc}(u)}{z(u)}\right] d\theta\right\}. \tag{10}$$

This relation has also been obtained in a much different way by Amos [1]. We note from (10) that if $R = 0$, then $P(A) = (\theta_2 - \theta_1)/2\pi$ as required.

The difficulty of evaluating the integral in (10) is resolved by using the minimax polynomial fit to $\operatorname{erfc}(u)/z(u)$ for $0 \leqq u \leqq C(\delta)$. For a given $\delta > 0$, a set of real numbers, $\{a_k\}$, and a least positive integer K are found such that

$$\left|\operatorname{erfc}(u) - z(u)\sum_0^K a_k u^k\right| \leqq \frac{2}{\sqrt{\pi}}\delta, \qquad 0 \leqq u \leqq C(\delta). \tag{11}$$

The constant $C(\delta)$ is chosen, once δ is specified, so that

$$\frac{1}{2\pi}\int_{A^*}\int \exp\left[-\frac{1}{2}(x^2+y^2)\right] dx\, dy = \frac{1}{2}\operatorname{erfc}(C) = \varepsilon \equiv \frac{\delta}{\sqrt{\pi}}, \tag{12}$$

where A^* denotes the angular region $R = \sqrt{2}C$, $\theta_2 = \pi/2 = -\theta_1$, i.e., the infinite region to the right of the line $x = \sqrt{2}C$. For $\delta \cong 5(-4)(= 5\times 10^{-4})$, $5(-7)$, $5(-10)$, $2(-13)$, the values of $C(\delta)$, K, and the minimax coefficients, a_k, are given in [4]. In case $\delta \cong 5(-4)$, $C = 2.46$, $K = 4$, and for $\delta \cong 5(-10)$, $C = 4.382$, $K = 14$.

If a maximum error of $(2/\sqrt{\pi})\delta$ can be incurred in approximating $\operatorname{erfc}(u)$, as indicated by (11), then the truncation error in computing $P(A)$ from (10) can be no larger than $\delta/\sqrt{\pi}$. Indeed, multiplying (11) by $ue^{-R^2/2}/[\pi z(u)]$ and integrating with respect to θ yields

$$\frac{e^{-R^2/2}}{\pi}\left|\int_{\theta_1}^{\theta_2}\left\{u\left[\frac{\operatorname{erfc}(u)}{z(u)}\right] - \sum_0^K a_k u^{k+1}\right\} d\theta\right| \leqq \frac{\delta}{\sqrt{\pi}} = \varepsilon, \qquad -\frac{\pi}{2} \leqq \theta_1 \leqq \theta_2 \leqq \frac{\pi}{2}, \tag{13}$$

where use has been made of the fact that

$$e^{-R^2/2}\int_{\theta_1}^{\theta_2} ue^{u^2}\, d\theta = \frac{\sqrt{\pi}}{2}\left[\operatorname{erf}\left(\frac{R}{\sqrt{2}}\sin\theta_2\right) - \operatorname{erf}\left(\frac{R}{\sqrt{2}}\sin\theta_1\right)\right] \leqq \sqrt{\pi}. \tag{14}$$

Recurrence relations allow us now to carry out the integration in (10) numerically. We have, using (11),

$$P(A) \cong \left(\frac{e^{-R^2/2}}{\pi}\right)\left(\frac{\theta_2 - \theta_1}{2} - \sum_0^K a_k J_{k+1}\right), \qquad -\frac{\pi}{2} \leqq \theta_1 \leqq \theta_2 \leqq \frac{\pi}{2}, \tag{15}$$

where

$$J_k \equiv \left(\frac{R}{\sqrt{2}}\right)^k \int_{\theta_1}^{\theta_2} \cos^k \theta \, d\theta, \tag{16}$$

and

$$J_{k+1} = \frac{1}{k+1}\left[\left(\frac{R}{\sqrt{2}}\cos\theta\right)^k \frac{R}{\sqrt{2}}\sin\theta\Big|_{\theta_1}^{\theta_2} + k\left(\frac{R}{\sqrt{2}}\right)^2 J_{k-1}\right], \tag{17}$$

with

$$J_0 = \theta_2 - \theta_1, \qquad J_1 = \frac{R}{\sqrt{2}}\sin\theta_2 - \frac{R}{\sqrt{2}}\sin\theta_1. \tag{18}$$

Since (11) only holds for $u \geqq 0$, we require $-\pi/2 \leqq \theta_1 \leqq \theta_2 \leqq \pi/2$. For cases outside this range we make use of the relation

$$P[A(R, 0, \theta)] = \frac{1}{2}\operatorname{erfc}\left(\frac{R}{\sqrt{2}}\sin\theta\right) - P[A(R, 0, \pi - \theta)], \qquad \frac{\pi}{2} \leqq \theta \leqq \pi, \tag{19}$$

where $A(R, 0, \theta)$ denotes the angular region with its vertex at $x = R$, $y = 0$, with $\theta_1 = 0$, $\theta_2 = \theta$ (see Fig. 2).

3. Normal probability over a convex polygon. The probabilities over an appropriate set of angular regions yield the probability $P(H)$ over a convex polygon $H(N)$ of N sides.[2] In [6] Gideon and Gurland propose using angular regions to determine probabilities over triangles and quadrilaterals and then using various combinations of these to find the probability $P(H)$. Our procedure in general will be more efficient, requiring only N angular regions for $H(N)$ as compared to $3(N-2)$ regions using triangles. If quadrilaterals are used for N even or quadrilaterals and one triangle for N odd, probabilities over $2N-4$ or $2N-3$ angular regions, respectively, are needed.

Let $H(t_1, \cdots, t_N)$ denote a convex polygon of N vertices at coordinate points $t_1, \cdots, t_N$, where $t_j \equiv (x_j, y_j)$. The vertices $\{t_j\}$ are ordered counterclockwise, i.e., so that the area of H is on the left as one traverses the boundary continuously. We then can write

$$P(H) = P(A_1) - \sum_{2}^{N-1} P(A_i) + P(A_N), \tag{20}$$

where A_1 is the angular region determined by an interior angle of H with its vertex assigned as t_1. The A_i, $i = 2, \cdots, N-1$ denote angular regions defined by the exterior angles and associated vertices $t_2, \cdots, t_{N-1}$ of H as shown in Fig. 3 for $N = 6$. The angular region A_N is obtained from the vertical angle of the interior angle of H at t_N. It is easy to visualize and argue the validity of (20). Note for the particular case of $N = 6$ in Fig. 3, that the probabilities over disjoint shaded regions E_2, E_3, E_4, E_5 diminish the value of $P(H)$ in (20) by an amount exactly compensated for by the addition of $P(A_N)$ since

$$\bigcup_{2}^{5} E_i = A_N.$$

The proof of (20) is not difficult; it is not given here.

[2] Note that (3) takes convex polygons into convex polygons.

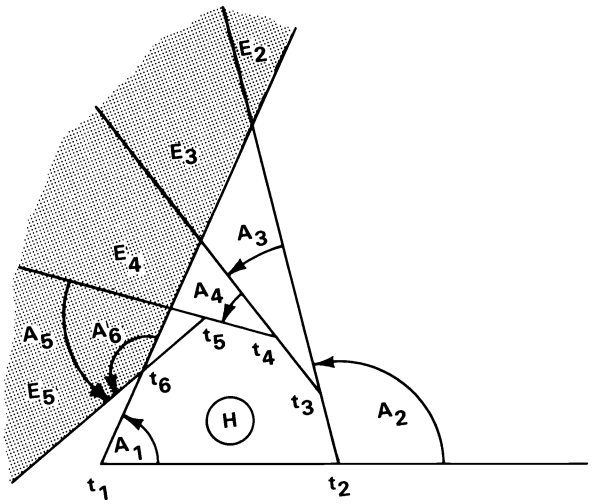


FIG. 3. *Shows angular regions* $A_1, \cdots, A_6$; E_2, E_3, E_4, E_5, *associated with* $H(6)$ *and equation* (20)

In keeping with our efforts for an efficient program, we make use of the following result. Let $\theta(i)$ denote the quantity $\theta_2 - \theta_1$ which appears in (10), for the ith angular region of H. Then, since the interior angles of $H(N)$ add up to $(N-2)\pi$ radians, we have

$$\theta(N) = -\theta(1) + \sum_{2}^{N-1} \theta(i). \tag{21}$$

Hence, no more than $N-1$ calls to the arctangent routine are needed. Other innovations to keep the program fast, such as when R is sufficiently small or sufficiently large, are discussed in [4].

When $\theta_2 - \theta_1$ is very close to either zero or π, there exists the possibility of a catastrophic error due to round-off. A procedure was found to test for this situation and to assure a correct result if it occurred. For example, suppose $\theta_2 - \theta_1 = \pi - \sigma$ where σ is a small positive number, but through round-off the computer shows $\pi + v$, v another small positive number. This slight deviation from the assumption $\Delta\theta \leqq \pi$ $(N \geqq 3)$ leads to incorrect signs for $\cos\theta_1$ and/or $\cos\theta_2$ which then will result in a wrong value for P. We keep this from occurring by sensing on the sign of $\sin(\theta_2 - \theta_1)$. If this quantity, which is computed from an algebraic relation, is negative, we set $\theta_2 - \theta_1 = \pi$ in this case, and appropriate measures are taken to obtain the correct result. Details are given in [4].

4. Comparison with other methods. In this section, we compare our method for computing $P(A)$ with [6], [7] and [5]. Programs were constructed for both [6], [7] and [5] procedures.

Our program is designed to give 3 levels of accuracy, i.e., the user may choose to obtain $P(A)$ accurate to 3, 6 or 9 decimal digits. We have also listed in [4] the necessary changes and appropriate values of some input parameters so that the program can be easily changed to also obtain 12 decimal digit accuracy.

The method of Gideon and Gurland [6], [7] is a novel and very efficient procedure for evaluating $P(A)$. It is limited, however, to 5-decimal-digit accuracy. They give an expression for $P[A(R, 0, \theta)]$;

$$P[A(R,0,\theta)] = \frac{1}{4}\operatorname{erfc}\left(\frac{R}{\sqrt{2}}\right)[(b_0 + b_1R + b_2R^2)\theta + (b_3 + b_4R)\theta^3 + (b_5 + b_6R)\theta^5], \tag{22}$$

where $|\theta| \leqq \pi/4$. The b_j coefficients were determined by least squares on 15 subintervals in R with $0 \leqq R \leqq 4$. In general, when $\theta_1 \neq 0$, they need to use (22) twice. Thus they need two calls to the arctangent routine (for $P(H(N))$, $2N-1$ calls are needed). Moreover, since (22) holds only for $|\theta| \leqq \pi/4$, which is not mentioned in [7], they require in addition to (19)

$$P[A(R, 0, \theta)] = \frac{1}{4}\operatorname{erfc}\left(\frac{R}{\sqrt{2}}\sin\theta\right)\operatorname{erfc}\left(\frac{R}{\sqrt{2}}\cos\theta\right) - P\left[A\left(R, 0, \frac{\pi}{2}-\theta\right)\right], \qquad \frac{\pi}{4} \leqq \theta \leqq \frac{\pi}{2}. \tag{23}$$

As noted earlier, we require one call to (19) if $\pi/2 < \theta < 3\pi/4$ or $3\pi/4 \leqq \theta < \pi$. In the second case, they also use (19) once; however, in the first case they need both (19) and (23). Hence counting the erfc function needed in (22) once for each case, it is easy to show by enumeration of cases that their method takes on the average 3.5 times as many erfc evaluations as ours. In addition, since they need both θ_1 and θ_2, whereas we use only $\theta_2 - \theta_1$, they require at least twice as many arctangents as we do. As a result, computer runs showed the average computing time per angular region was about 25% longer for their procedure for five-decimal-digit accuracy.

The bivariate normal integral is given [10] by

$$\Phi(m, k, \rho) \equiv (2\pi\sqrt{1-\rho^2})^{-1} \int_{-\infty}^{m}\int_{-\infty}^{k} \exp\left[-\frac{(w^2 - 2\rho wz + z^2)}{2(1-\rho^2)}\right] dw\, dz, \tag{24}$$

where the integration is taken over a right angular region, B, with $|\rho| < 1$. Drezner [5] gives a method for evaluating this integral. We found the program capable of giving very high accuracy, but at the same time it was much less efficient than ours. The relationship between $\Phi(m, k, \rho)$ and $P(A)$ for the corresponding A is given in [7]. The equations connecting m, k, ρ and the associated values of R, θ_1, θ_2 are given in [4].

Drezner's procedure is based on transforming Φ in (24) to

$$\Phi(m, k, \rho) = \left(\frac{\sqrt{1-\rho^2}}{\pi}\right)\int_0^{\infty}\int_0^{\infty} e^{-u^2} e^{-v^2} f(u, v)\, du\, dv, \tag{25}$$

where

$$u = \frac{(m-w)}{\sqrt{2(1-\rho^2)}}, \qquad v = \frac{(k-z)}{\sqrt{2(1-\rho^2)}}, \qquad M = \frac{m}{\sqrt{2(1-\rho^2)}}, \qquad K = \frac{k}{\sqrt{2(1-\rho^2)}},$$
$$f(u, v) = \exp\left[M(2u-M) + K(2v-K) + 2\rho(u-M)(v-K)\right]. \tag{26}$$

Gaussian integration is used to evaluate (25) when m, k, ρ are all nonpositive, i.e.,

$$\Phi(m, k, \rho) \cong \left(\frac{\sqrt{1-\rho^2}}{\pi}\right)\sum_{j=1}^{J}\sum_{i=1}^{J} C_i C_j f(u_i, v_j), \tag{27}$$

$$m \leqq 0, \qquad k \leqq 0, \qquad \rho \leqq 0, \tag{28}$$

where C_i are the positive weights and u_i, v_j are the Gaussian abscissae, as given in [9]. Drezner makes the significant observation that $f(u, v) \leqq 1$ when (28) holds, and consequently high accuracy can be obtained using (27). For $J = 5$ the maximum observed error is reported to be 5.5×10^{-7}. If (28) does not hold, the equivalent of (19) is used with (27), and, in fact, it can be shown that his method requires an erfc function calculation whenever ours does and vice versa.

A comparison of the procedures was carried out by running both programs over a span of arguments which covered a large part of the plane. The accuracy of the results agreed within claims made in both methods. However, for 6-decimal-digit accuracy the Drezner program was about 4 times as slow as ours and about 8 times as slow for 9-decimal-digits of accuracy. This was not surprising because of the large number of exponentials required by Drezner. For example, for $J = 5$ (6 digit accuracy), it requires 25 exponentials when $mk\rho \leqq 0$ and 50 exponentials when $mk\rho > 0$.

In programming the Drezner method, a need for the limiting values of Φ when $\rho \to 1$ or -1 arose. Rounding error can result in $\rho = 1$ or $\rho = -1$. The limiting values are given in [4], but do not appear in [5]. Two typographical errors were also found. In (10) of [5] $1/2\pi$ should be $1/\sqrt{2\pi}$, and in (12) the plus sign should be replaced with a minus sign in the expression for δ_{hk}.

Acknowledgment. The authors wish to express their thanks to David Hoaglin for furnishing reference [2] and for helpful suggestions which improved the presentation.

REFERENCES

[1] D. E. AMOS, *On computation of the bivariate normal distribution*, Math. Comp., 23 (1969), pp. 655–659.
[2] H. L. CRUTCHER, *Bivariate Normal Offset Circle Probability Tables*, Vol. III, Applications to Geophysical Data (National Weather Records Center, U.S. Dept. of Commerce. CAL No. XM-2464-G-1), Cornell Aeronautical Laboratory, Inc., Buffalo, NY, 1967.
[3] D. J. DALEY, *Computation of bi- and tri-variate normal integrals*, Appl. Statist., 23 (1974), pp. 435–438.
[4] A. R. DIDONATO, M. P. JARNAGIN, JR., AND R. K. HAGEMAN, *Computation of the Bivariate Normal Distribution over Convex Polygons*, Technical Report NSWC/DL TR-3886, Naval Surface Weapons Center, Dahlgren, VA 22448, September 1978.
[5] Z. DREZNER, *Computation of the bivariate normal integral*, Math. Comp., 32 (1978), pp. 277–279.
[6] R. A. GIDEON AND J. GURLAND, *A Method of Obtaining the Bivariate Normal Probability Over an Arbitrary Polygon*, Tech. Rept. No. 304, Dept. of Statistics, University of Wisconsin, Madison, WI, May 1972.
[7] R. A. GIDEON AND J. GURLAND, *A polynomial type approximation for bivariate normal variates*, SIAM J. Appl. Math. 34 (1978), pp. 681–684.
[8] R. R. SOWDEN AND D. SECREST, *Computation of the bivariate normal integral*, Appl. Statist., 18 (1969), pp. 169–180.
[9] N. M. STEEN, G. O. BYRNE AND E. M. GELBARD, *Gaussian quadrature formulas*, Math. Comp., 23 (1969), pp. 661–671.
[10] U.S. Department of Commerce, National Bureau of Standards, *Handbook of Mathematical Functions*, Appl. Math. Series, v. 55, U.S. Government Printing Office, Washington, DC, 1964, p. 936.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0002 $01.00/0

A MODIFICATION TO THE DISCRETE l_1 LINEAR APPROXIMATION ALGORITHM OF BARRODALE AND ROBERTS*

STEPHEN C. BANKS† AND HOWARD L. TAYLOR‡

Abstract. A modification to the discrete l_1 linear approximation algorithm of Barrodale and Roberts allows the determination of the vector x that minimizes $\|e\|_1 + \|x\|_1$, where e is the residual vector defined by the linear system $Ax + e = b$. The matrix A is $m \times n$, where m is not necessarily greater than or equal to n. Computational and storage properties of the unmodified algorithm are preserved.

Key words. l_1 approximation, l_1 norm, linear programming, simplex method, approximation theory.

1. Introduction. Given an overdetermined linear system $Ax = b$, where $A = (a_{ij})$ is $m \times n$ with $m \geqq n$, $x = (x_j)$ is $n \times 1$, and $b = (b_i)$ is $m \times 1$, the usual l_1 linear approximation problem is to minimize $\sum_{i=1}^{m} |e_i|$ where $e = b - Ax$. Claerbout and Muir [3] discussed the desirability, in geophysical applications, of applying objective functions whose arguments include the magnitudes of the components of both x and e. Taylor, Banks, and McCoy [4] applied the recommendations of Claerbout and Muir to geophysical problems involving seismic processing. In so doing, the following problem arose. Find vectors x and e to

$$\text{Minimize} \sum_{i=1}^{m} p_i|e_i| + \lambda \sum_{j=1}^{n} q_j|x_j| \quad \text{subject to } Ax + e = b, \tag{1}$$

where $\lambda \geqq 0$, and $p_i, q_j > 0$ for all i, j. The constants p_i, q_j are weighting factors, and the value of λ is chosen according to the particular application. This formulation is applicable to deconvolution and other pattern recognition and feature extraction processes [4]. The relative magnitudes of the positive integers m and n are not restricted to the case $m \geqq n$. For $\lambda > 0$, if we make the change of variables

$$e_i' = p_i e_i, \qquad x_j' = \lambda q_j x_j,$$

and then let

$$a_{ij}' = \frac{p_i a_{ij}}{(\lambda q_j)}, \qquad b_i' = p_i b_i,$$

the problem can be written:

$$\text{Minimize} \sum_{i=1}^{m} |e_i'| + \sum_{j=1}^{n} |x_j'|$$

$$\text{subject to } A'x' + e' = b',$$

which is in the form required by the following modified algorithm.

2. The algorithm modification. The following modification to the discrete l_1 linear approximation algorithm of Barrodale and Roberts [1], [2], ACM Algorithm 478, allows the determination of a vector x that minimizes $\sum_{i=1}^{m} |e_i| + \sum_{j=1}^{n} |x_j|$, where the residual vector e is defined by the linear system $Ax + e = b$. Assuming the $m \times n$ matrix A is suitably scaled, a solution x can be obtained in the case $m < n$ as well as $m \geqq n$. The

* Received by the editors July 20, 1979.
† Sun Production Co., Box 936, Richardson, Texas 75080.
‡ Consultant, Box 354, Richardson, Texas 75080.

modification does not change the capability of the original algorithm to minimize $\sum_{i=1}^{m} |e_i|$ when $m \geqq n$.

In order to minimize $\sum_{i=1}^{m} |e_i| + \sum_{j=1}^{n} |x_j|$ subject to $Ax + e = b$, let $x_j = r_j - s_j$, and $e_i = u_i - v_i$. The objective function is minimized by an optimal solution to the linear programming problem:

$$\text{Minimize } \sum_{j=1}^{n} (r_j + s_j) + \sum_{i=1}^{m} (u_i + v_i)$$

$$\text{subject to } \sum_{j=1}^{n} a_{ij}(r_j - s_j) + u_i - v_i = b_i, \quad i = 1, 2, \cdots, m, \tag{2}$$

$$\text{and } r_j, s_j, u_i, v_i \geqq 0.$$

This formulation corresponds to that of Barrodale and Roberts [1], [2] with the exception of the objective function.

If we assume each b_i is nonnegative, then in the initial condensed simplex tableau of Barrodale and Roberts [1] the marginal costs corresponding to the variables r_j are

$$-1.0 + \sum_{i=1}^{m} a_{ij}, \qquad j = 1, 2, \cdots, n.$$

The sum of the marginal costs of r_j and s_j is seen to be -2.0 for each j. So, regardless of the rank of A, we are not assured that any specific number of x_j's will be represented by a column vector $A_{\cdot j}$ or $-A_{\cdot j}$ in the final (optimal) basis. Therefore, we skip Stage 1 of the Barrodale and Roberts algorithm and go directly to Stage 2 (after setting the counting indices of Stage 1 to their initial values). The algorithm then proceeds exactly as described by Barrodale and Roberts [1], [2]. Note that the capability of passing through several simplex vertices at each iteration is preserved, and the declared storage space required is unchanged.

Additional changes in ACM Algorithm 478 [2] are required for the output. The solution is feasible after Stage 2 is completed, since Stage 1 was omitted. Each value in the feasible solution must then be checked to see if it is an x_j or e_i, since these are no longer partitioned in the final tableau. The sum of the solution values in the final tableau will be $\sum_{i=1}^{m} |e_i| + \sum_{j=1}^{n} |x_j|$. Finally, the variable that represented the rank of the coefficient matrix has lost its original meaning.

In order to preserve the algorithm's capability of minimizing $\sum_{i=1}^{m} |e_i|$, a flagging parameter can be passed into the subroutine and used to determine if:

(i) -1.0 should be subtracted from the initial marginal cost of each r_j,

(ii) Stage 1 should be omitted, and

(iii) the optimal solution checked for feasibility.

One last change makes the original algorithm and the modification more compatible. The original code can be changed so that, independent of the value of λ, each solution value is checked, regardless of its position in the final tableau, to see if it is an x_j or e_i.

The modification was coded in FORTRAN for use on a CDC Cyber 172, and has been tested extensively [4].[1]

The problem

$$\text{Minimize } \sum_{i=1}^{m} |e_i| + \sum_{j=1}^{n} |x_j|$$

$$\text{subject to } Ax + e = b \tag{3}$$

[1] A listing of the algorithm is available from the first author.

can be solved for x by the algorithm of Barrodale and Roberts when it is formulated as

$$\text{(4)} \qquad \begin{aligned} &\text{Minimize } \sum_{i=1}^{m+n} |d_i| \\ &\text{subject to } \left(\frac{A}{-I}\right)x + d = \left(\frac{b}{0}\right). \end{aligned}$$

However, the modified algorithm applied to (3) does not require the additional storage demanded by (4). The reduced data memory requirements are important in handling large scale seismic processing problems efficiently. We note the problems (3) and (4) are solvable when $m < n$.

In some applications [4], it is desirable to force many of the components of x to zero. This can be done by increasing λ in (1). As λ is increased, the execution time tends to decrease because fewer components of x will, in general, be brought into the basis for the optimal solution. The effect of λ on the solution x can be studied by using parametric linear programming [4].

Example. A seismic trace b can be modeled as the digital convolution of a spike train x of length n and a wave w of length k. If noise is added to the convolution we have

$$x * w + e = b, \qquad \text{(5)}$$

where the trace b and the noise term e are each of length $m = n + k - 1$ [3, 4].

The convolution (5) can be given the linear algebra formulation

$$Ax + e = b, \qquad \text{(6)}$$

where the $m \times n$ matrix A is formed from the wave w by setting

$$\begin{aligned} a_{ij} &= w_{i-j+1}, & 1 \leqq i - j + 1 \leqq k \\ &= 0 & \text{otherwise.} \end{aligned}$$

The columns of A are thus shifted copies of the wave w [4]. The vector x is $n \times 1$, and the vectors e and b are $m \times 1$.

Traces 1, 2, and 3 in Fig. 1 illustrate the model. The series x of trace 1 is convolved with the wavelet w of trace 2. Noise is then added and the result is trace 3. Time is shown in milliseconds (msec) and a sample occurs at time zero and then at multiples of 4 msec. The series x is of length $n = 110$, or 436 msec. The wave w is of length $k = 24$. The series b is of length $m = 133$.

We attempt to recover the original reflectivity series x of trace 1 assuming we are given the wave w of trace 2 and the noisy series b of trace 3. The approach is to find the vectors x and e that

$$\text{(7)} \qquad \begin{aligned} &\text{Minimize } \sum_{i=1}^{m} p_i|e_i| + \lambda \sum_{j=1}^{n} q_j|x_j| \\ &\text{subject to } Ax + e = b, \end{aligned}$$

where $\lambda \geqq 0$, $p_i, q_j > 0$ for all i, j, A is the wavelet matrix previously defined in (6), and b is trace 3.

For the results shown in Fig. 1, we let $p_i = 1$ for all i, and let

$$q_j = \frac{1}{100} \sum_{i=1}^{m} p_i|a_{ij}|.$$

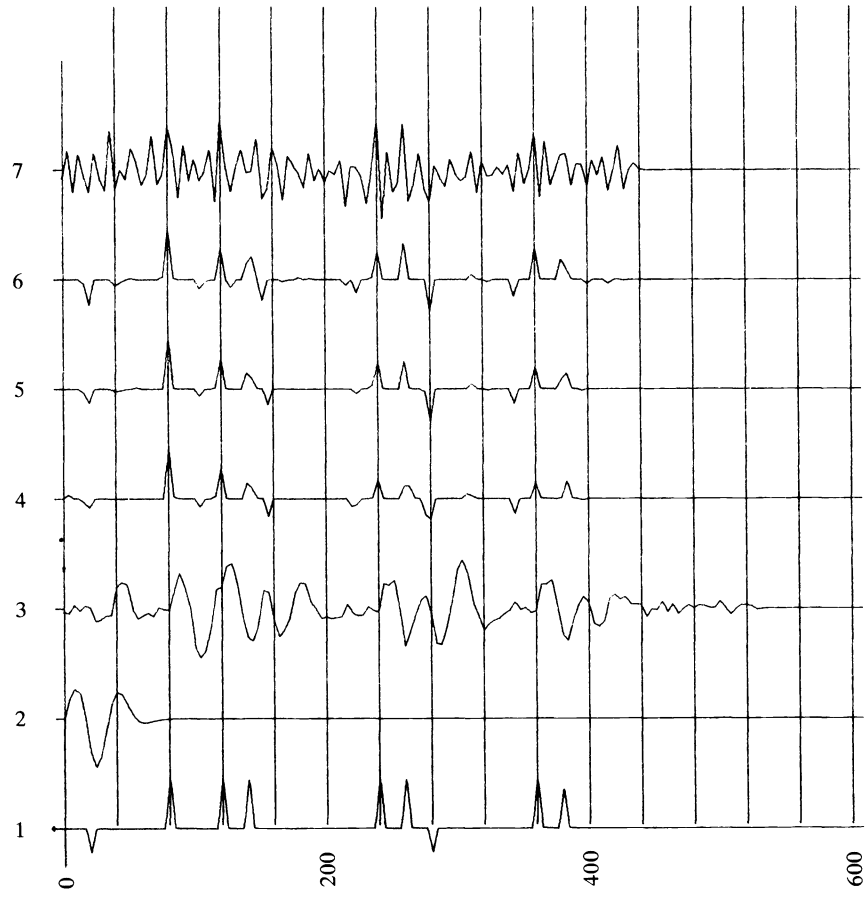


FIG. 1

With this choice of q_j we only need to consider $\lambda \in [0, 100]$ as the optimal solution of (7) for $\lambda > 100$ will then be $x_j = 0$ for all j.

The optimal solutions x corresponding to $\lambda = 30, 20, 10$, and 0 are shown in Fig. 1 by traces 4, 5, 6, and 7 respectively. These solutions can be improved by using a more complicated set of weights p_i [4]. The importance of the term including λ in the objective function can be seen by considering trace 7, where $\lambda = 0$, and comparing it to traces 4, 5, and 6 which more nearly reproduce trace 1. In our work of this type [4], we found that $\lambda \in [20, 30]$ produced satisfactory results.

REFERENCES

[1] 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.

[2] ———, *Solution of an overdetermined system of equations in the l_1 norm*, Comm. ACM, 17 (1974), pp. 319–320.

[3] J. F. CLAERBOUT AND F. MUIR, *Robust modeling with erratic data*, Geophysics, 38 (1973), pp. 826–844.

[4] H. L. TAYLOR, S. C. BANKS AND J. F. MCCOY, *Deconvolution with the l_1 norm*, Geophysics, 44 (1979), pp. 39–52.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0003 $1.00/0

A TEST OF MOVING MESH REFINEMENT FOR 2-D SCALAR HYPERBOLIC PROBLEMS*

WILLIAM D. GROPP†

Abstract. We present an algorithm for numerically solving hyperbolic equations with moving shock fronts. The central idea is a moving, finer mesh which follows the shock. Computational results are presented.

Key words. hyperbolic partial differential equations, finite differences, mesh refinement, shocks

Introduction. The solution of hyperbolic partial differential equations often contains shocks. In solving these problems by finite differences, it seems reasonable to use a finer grid in the region of the shocks. Since, in general, the positions of the shocks change with time, the position of the finer grid must also change to follow the shocks. Further, it may be desirable to use a different method near a shock. For example, a lower order method may be used on the fine grid to reduce unwanted oscillations in the solution (Harten and Zwas [4] do this, but they do not use a local refinement). A conservative scheme could be used near a shock and a simpler scheme used elsewhere.

Similar ideas have been successfully applied to problems where the fine grid remains fixed in time, for example, parabolic problems [2], boundary layers in hyperbolic problems [1], [7], and elliptic problems [5]. Grids which change in time have been used in some gas dynamics computations [9]. In this paper we describe a program which tests all of these ideas. Using $u_t+uu_x+uu_y=0$ on the unit square as a simple test problem, we compare its efficiency with a conventional finite difference method of equal accuracy.

This technique can be applied to problems which contain a shock or other sharp front, and which are relatively quiescent away from the shock front. Combustion along a moving front is one example. Shocks in gas dynamics furnish another example, though the algorithm presented here offers no improvement for the calculation of rarefaction waves.

Description of the grids. Two grids are used. One, called the coarse grid, is $n\times n$, with $\Delta x=\Delta y=1/(n-1)$ and $\Delta t/\Delta x=\lambda$. The coarse grid determines cells in space. Each cell is $\Delta x\times\Delta y$, with the corners of the cell at coarse grid points. Each cell which is refined contains a fine grid with mesh size $\delta x=\delta y=\Delta x/(m-1)$. The time step for the fine grid is $\delta t=\Delta t/(m-1)$, so the ratio $\delta t/\delta x=\Delta t/\Delta x$ is the same on both grids. It should be pointed out that Browning, Kreiss, and Oliger [1] have shown that the coarse grid must be sufficiently fine to represent accurately those parts of the solution which are not in a refined region. The example at the end of their paper demonstrates the need for this. Further, they show that if the solution is well represented on the coarse grid, there will be no problem with waves passing through the coarse grid-fine grid boundary.

Algorithm. The following steps are followed to integrate the solution from time T to $T+\Delta t$.

*Received by the editors September 5, 1979, and in revised form January 24, 1980.

†Lawrence Livermore Laboratory, University of California, Livermore, California 94550. Present address: Department of Computer Science, Stanford University, Stanford, California 94304. This research was conducted under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract W-7405-ENG-48.

(1) Integrate on the coarse grid one time step, saving the values of u on the grid from the previous time step. For convenience, call the values of u on the coarse grid at time $T+\Delta t$ the new coarse u-values, and the values of u on the coarse grid at time T the old coarse u-values.

(2) Using the new coarse u-values, determine where the fine grid will be for the time step from T to $T+\Delta t$. This is done by estimating the gradient of u across each cell. The new fine grid is then composed of two categories of cells. The first category is made up of those cells across which the estimated gradient is larger than a specified tolerance. The second category of cells are those cells adjacent to cells in the first category. The cells in the first category cover the area where the solution is changing rapidly. The cells in the second category provide a buffer zone which serves to both isolate the coarse grid from the shock (see step 5) and to leave room for the fine grid to dissipate oscillations. These oscillations come from the computation of the shock. We do not want these oscillations to enter the coarse grid. Thus we must leave enough space between the shock and the coarse grid-fine grid boundary for the viscosity in the method to damp out these oscillations.

(3) Next, we need the values of u on this fine grid for time T. The cells which are refined at this time step may or may not have been refined before. If a cell was refined for the previous time step, we already have the values of u on the fine grid at time T. Otherwise, we use the values of u at the corners of the cell to interpolate the values of u for that cell. This is why the old coarse u-values were saved. Because such a cell is in the buffer zone, the old coarse values of u are slowly varying there, and a simple first order interpolation scheme may be used.

(4) At this point, the fine grid is integrated from time T to $T+\Delta t$. A different type of integration scheme from the one on the coarse grid may be used here if desired. However, we need to specify boundary values on the fine grid where it abuts an unrefined cell. At these boundaries, we simply specify the value of u on the fine grid by using linear interpolation from the old and new coarse u-values. The physical boundaries at $x=0$, $x=1$, $y=0$, and $y=1$ are handled in the same way for both the fine and coarse grids. As this is problem dependent, it is discussed in the examples below.

(5) Finally, for a point which is an interior point on both the coarse grid and the fine grid, the value of u at $T+\Delta t$ on the coarse grid is taken to be the value of u just computed on the fine grid. This serves to isolate the values of u on the coarse grid from a shock. Since the stencil for the integrator is small, if the region of refinement is wide enough, then the values of u outside the region of refinement will be unaffected by the shock.

The solution at time $T+\Delta t$ is now complete; the algorithm is repeated until the desired final time is reached.

The following examples were programmed in FORTRAN and run on a CRAY-1 computer without vectorization.

Examples. We tried three test problems with the equation

$$u_t+uu_x+uu_y=0, \qquad 0\leqq x\leqq 1, \quad 0\leqq y\leqq 1, \quad 0\leqq t\leqq 1.125.$$

Note that under the rotation $x'=(x+y)/\sqrt{2}$, $y'=(-x+y)/\sqrt{2}$, the equation becomes $u_t+\sqrt{2}\,uu_{x'}=0$; the solutions of this equation are well known [3], [8].

In each example we compare calculations with and without mesh refinement. Those with mesh refinement are compared to a conventional calculation that pro-

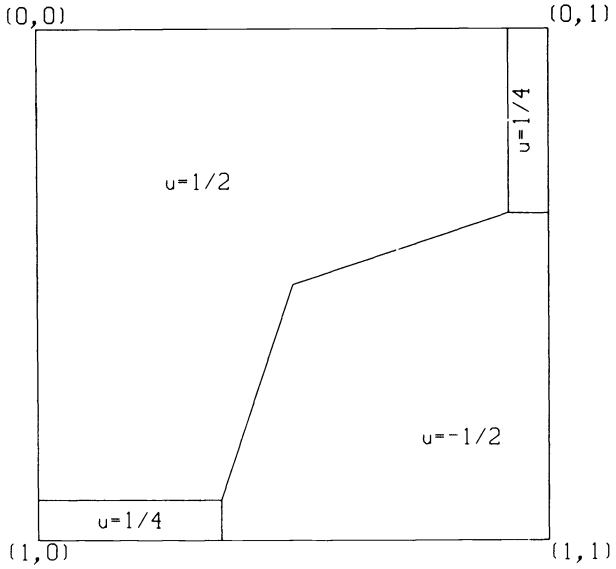


FIG 1. *Exact solution to Example* 1 *at* $T=1.125$.

duces the same answers. The boundary conditions for the differential equation are the values that u would have if the problem was naturally extended into the entire x-y plane. In all three examples, the coarse grid has $\Delta x=\Delta y=\frac{1}{20}$, and $\Delta t=\frac{1}{40}$. The fine mesh has $\delta x=\delta y=\frac{1}{40}$, $\frac{1}{80}$, and $\frac{1}{100}$. The tables compare the time and space used by mesh refinement with a calculation using a uniform grid with the same spacing as the fine grid in the mesh refinement calculation.

Example 1. The initial conditions are:

$$u(x,y,t=0)=\begin{cases}\frac{1}{2} & \text{if } 0\leqq x\leqq\frac{1}{2} \text{ and } 0\leqq y\leqq\frac{1}{2},\\ -\frac{1}{2} & \text{if } \frac{1}{2}<x\leqq 1 \text{ and } \frac{1}{2}<y\leqq 1,\\ \frac{1}{4} & \text{otherwise.}\end{cases}$$

The exact solution at $T=1.125$ is shown in Fig. 1.

The integrator used on both the coarse and fine grids is MacCormack's scheme [6], a second order explicit method. At the boundaries $x=0$, $x=1$, $y=0$, and $y=1$ the value of u is specified to be the exact solution. A cell is placed in the first category in step 2 if the change in u across a cell is greater than 0.10. The results of the timing studies are shown in Table 1; sample plots for $T=1.125$ are shown in Figs. 2 and 3.

TABLE 1

Spacing of finest mesh	Time, no refinement	Time, with refinement	Space saved with refinement
$\frac{1}{40}$	$3\frac{1}{2}$ secs	$2\frac{1}{2}$ secs	−239 words
$\frac{1}{80}$	27 secs	10 secs	2769 words
$\frac{1}{100}$	54 secs	21 secs	4389 words

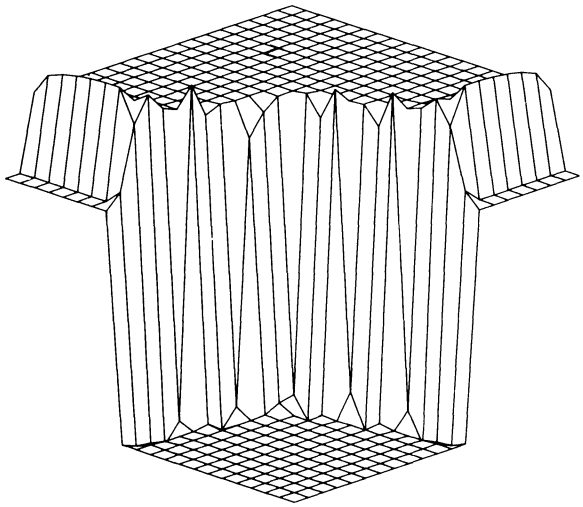


FIG 2. *Surface plot of calculated solution to Example* 1 *at* $T = 1.125$.

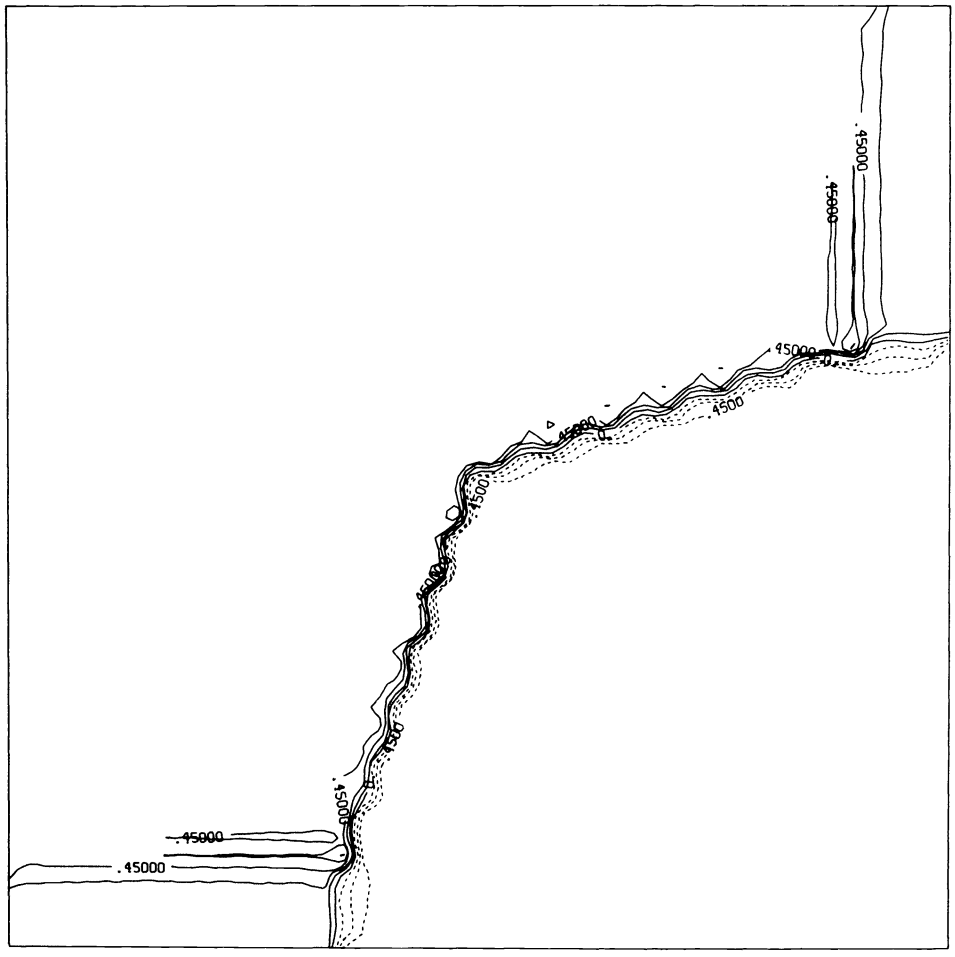


FIG 3. *Contour plot of Fig.* 2.

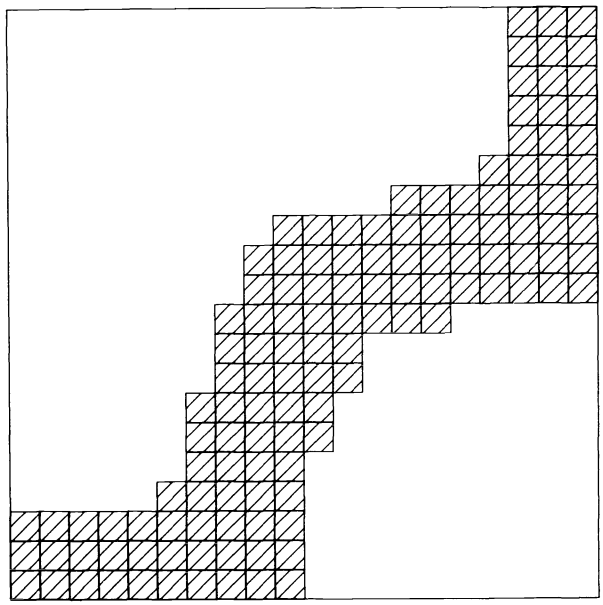


FIG 4. *Region of refinement at* $T=1.125$ *used for calculating Figs.* 2 *and* 3.

Fig. 4 shows the region of refinement at the same time. As can be seen from Table 1, mesh refinement produces the same results at about half the cost in both time and space.

Example 2. The initial conditions are:

$$u(x,y,t=0)=\begin{cases}\frac{1}{2} & \text{if } 0\leqq x\leqq\frac{1}{2} \text{ and } 0\leqq y\leqq\frac{1}{2},\\ \frac{1}{8} & \text{if } \frac{1}{2}<x\leqq 1 \text{ and } \frac{1}{2}<y\leqq 1,\\ \frac{1}{4} & \text{otherwise.}\end{cases}$$

The exact solution to this problem is similar to the solution of Example 1, except that all of the shocks move towards the lower right (cf. Fig. 1).

The integrator used on both the coarse and fine grids is again MacCormack's scheme. The boundaries at $x=0$ and $y=0$ are inflow boundaries, and the value of u on these is specified as the exact solution at all times. The boundaries at $x=1$ and $y=1$ are outflow boundaries; the values of u on these boundaries is computed by upstream differencing. A cell is placed in the first category in step 2 if the change in u across the cell is greater than 0.03.

The results of the timing studies are shown in Table 2. Again, using mesh refinement can save as much as 50% of the time and space needed to do a computation.

TABLE 2

Spacing of finest mesh	Time, no refinement	Time, with refinement	Space saved with refinement
$\frac{1}{40}$	$3\frac{1}{2}$ secs	3 secs	−654 words
$\frac{1}{80}$	28 secs	13 secs	664 words
$\frac{1}{100}$	54 secs	22 secs	3519 words

Example 3. We use the same problem as in Example 2, but with upstream differencing as the integrator on the fine grid. The boundaries are handled in the same way. The same criterion is used to decide which cells to refine. This computation produces a reasonably sharp shock front with no overshooting (Fig. 5) and is less expensive than a similar calculation without refinement.

Conclusion. It is clear from these examples that this technique can be a valuable method for hyperbolic problems whose major features are sharp shock fronts. The savings in time and space that result from using a fine grid only where necessary are considerable. For a system with more dependent variables the savings would be even greater because the addition of more variables increases the computation per grid point but has little effect on the overhead associated with the moving fine grid. By using a first order integrator on the fine mesh, a sharp clean shock front can be calculated. For problems with more complicated solutions, the method described in step two for deciding where to refine could be replaced by a method which chooses where to refine by estimating the local truncation error, for example.

Acknowledgments. I would like to thank M. Berger, G. Majda, and J. Oliger for many stimulating discussions. Most of all, the author would like to thank G. Hedstrom for suggesting the problem to me and for his many useful suggestions and advice.

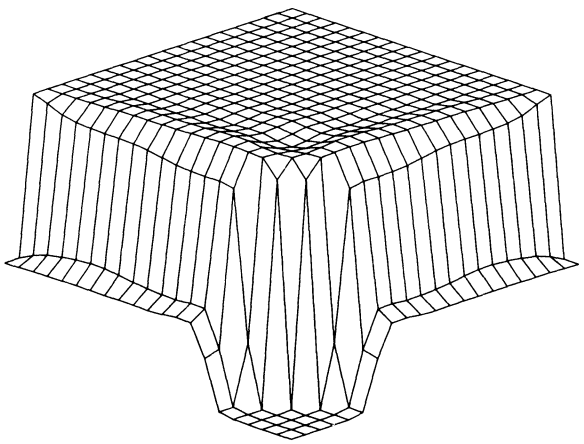


FIG 5. *Surface plot of calculated solution to Example* 3 *at* $T=1.125$.

REFERENCES

[1] G. BROWNING, H. -O. KREISS, AND J. OLIGER, *Mesh refinement*, Math. Comp., 27 (1973), pp. 29–39.

[2] M. CIMENT AND R. A. SWEET, *Mesh refinements for parabolic equations*, J. Comp. Phys., 12 (1973), pp. 513–525.

[3] J. GUCKENHEIMER, *Shocks and rarefactions in two space dimensions*, Arch. Rational Mech. Anal., 59 (1975), pp. 281–291.

[4] A. HARTEN AND G. ZWAS, *Self-adjusting hybrid schemes for shock computation*, J. Comp. Phys., 9 (1972), pp. 568–583.

[5] D. C. L. LAM AND R. B. SIMPSON, *Modeling coastal effluent transport using a variable finite difference grid*, Advances in Computer Methods for Partial Differential Equations II, R. Vichnevetsky, ed., IMACS(AICA), 1977, pp. 294–300.

[6] R. MACCORMACK, *Numerical Solutions of the Interaction of a Shock Wave with a Laminar Boundary Layer*, Lecture Notes on Physics, Vol. 8, M. Holt, ed., Springer-Verlag, New York, 1971.

[7] J. OLIGER, *Hybrid difference methods for the initial boundary-value problem for hyperbolic equations*, Math. Comp., 30 (1976), pp. 724–738.

[8] G. B. WHITHAM, *Linear and Nonlinear Waves*, Wiley, New York, 1974.

[9] N. M. YANENKO, V. D. LISSEIKIN, AND V. M. KOVENIA, *The Method of the Solution of Gas Dynamical Problems in Moving Meshes*, Computing Methods in Applied Sciences and Engineering (Proc. Third Internat. Sympos., Versailled, 1977), II, Lecture Notes in Phys., 91, Springer-Verlag, Berlin, 1979, pp. 48–61.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0004 $1.00/0

AN ALGORITHM FOR COMPUTING THE MATRIX COSINE*

STEVEN M. SERBIN† AND SYBIL A. BLALOCK†

Abstract. We present and analyze an algorithm for computing the cosine of a matrix which is based on the double-angle formula $\cos 2A = 2\cos^2 A - I$. We discuss the relevance of this computation to second-order matrix differential equations. We draw the analogy between this method for the cosine and the familiar scaling and squaring method for computing the matrix exponential. Numerical experiments employing polynomial and rational approximations to the cosine, in conjunction with the double angle technique, are presented.

Key words. matrix cosine

1. Introduction. In recent years, extensive work has been done to achieve algorithms for the accurate approximate calculation of the matrix exponential. An excellent survey of the many and varied approaches to this problem has been done recently by Moler and Van Loan [4]; of the many methods, one of the more successful, which proceeds without the determination of the eigensystem, is the scaling and squaring method proposed and analyzed by Ward [8].

One important interest in this computation is for the solution of the first-order differential system

$$y' = Ay, \qquad y(0) = y_0, \tag{1}$$

where A is a real $n \times n$ matrix, and y and y_0 are vectors in $\mathbb{R}^n$. This problem arises, for example, in the semidiscretization of the heat equation [7].

The purpose of this note, on the other hand, is to consider a scheme for the computation of the matrix cosine. This scheme seems to be the analogue of Ward's exponential method; in particular, it shares the advantage that no computation of eigenvalues or eigenvectors is required.

The interest in this computation stems from the fact that

$$\cos At \equiv \sum_{j=0}^{\infty} (-1)^j \frac{(At)^{2j}}{(2j)!}$$

satisfies the second-order system

$$Y''(t) + A^2 Y(t) = 0, \quad Y(0) = I, \quad Y'(0) = 0. \tag{2}$$

The corresponding vector-matrix system,

$$y''(t) + A^2 y(t) = 0, \tag{3}$$

can be obtained from semidiscretization of the wave equation (cf. [6]). Then, approximations for the cosine operator, along with analogous schemes for the sine operator, might be combined to produce schemes for the second-order system (3) (with general initial conditions) directly, rather than by converting it into an equivalent first-order system of twice the size.

*Received by the editors October 16, 1979. This research was supported by the U.S. Army Research Office under Grant DAAG 29-78-C0024.

†Department of Mathematics, University of Tennessee, Knoxville, Tennessee 37916.

The problem of computing the matrix cosine solution of (2) has been discussed by Apostol [1], but his method does require computation of eigenvalues, as well as solution of successively generated second-order scalar constant coefficient nonhomogeneous equations, and so does not appear to be well-suited for large scale problems and automatic computation.

The algorithm we propose can be applied in principle to any matrix; however, the case we analyze requires that A be normal and have real eigenvalues (e.g., Hermitian).

2. The double-angle method. The premise of scaling and squaring for the exponential is that local methods, such as Taylor series or the Padé rational schemes, provide extremely accurate approximations to $\exp A$ if $\|A\|$ is sufficiently small (but cannot be used directly for large $\|A\|$). Then, to compute $\exp A$ for general A, the matrix is scaled down by an appropriate factor 2^j, so that a good approximation $r(A/2^j)$ to $\exp(A/2^j)$ may be determined. Then, since the exponential enjoys the unique property

$$\{\exp(A/2^j)\}^{2^j} = \exp A, \tag{4}$$

the approximation to $\exp A$ is obtained by applying j successive squarings to $r(A/2^j)$.

The cosine operator enjoys a different, but analogous property. Let us define

$$Y_j = \cos A/2^{N-j}, \qquad j=0,\ldots,N. \tag{5}$$

Then, just as in the scalar case, it can be shown easily that

$$Y_{j+1} = \cos\frac{2A}{2^{N-j}} = 2Y_j^2 - I. \tag{6}$$

This is just the matrix version of the double-angle formula $\cos 2\theta = 2\cos^2\theta - 1$. The approximation scheme then becomes

(7) Let $U_0 = R(A/2^n)$, where $R(z)$ is some (in general, rational) approximation to $\cos z$; then let

$$U_{j+1} = 2U_j^2 - I, \qquad j=0,\ldots,n-1. \tag{8}$$

3. Error analysis. We now proceed to consider the error in the approximation. Since $Y_N = \cos A$, we wish to investigate $E_j \equiv Y_j - U_j$, $j=0,\ldots,N$. Let us first assume that A is normal. Then, if f is any function analytic on the spectrum $\sigma(A)$, and we denote by $\rho(A)$ the spectral radius of A and $\|A\|_2 \equiv [\rho(A^*A)]^{1/2}$, it follows that

(9) $f(A)$ is normal, and

$$\|f(A)\|_2 = \rho(f(A)) = \sup_{\lambda\in\sigma(A)} |f(\lambda)|. \tag{10}$$

If further we assume that A has real eigenvalues, then it follows easily from (10) that

$$\|Y_j\|_2 = \|\cos\frac{A}{2^{N-j}}\|_2 \leqq 1. \tag{11}$$

We make one final assumption: the (rational) approximation $R(z)$ to $\cos z$ is chosen to be analytic on $\sigma(A)$, such that

$$\|U_0\|_2 = \|R\left(\frac{A}{2^N}\right)\|_2 \leqq 1. \tag{12}$$

Recall that the Chebyshev polynominal of the first kind of degree two is $T_2(x)=2x^2-1$. It is well known that

$$\sup_{-1\leqq x\leqq 1}|T_2(x)|=1. \tag{13}$$

Then, (8) can be written

$$U_{j+1}=T_2(U_j). \tag{14}$$

It follows using (10), (12), (13), (14), and induction, that

$$\|U_j\|\leqq 1, \qquad j=0,\ldots,N. \tag{15}$$

Now,

$$\begin{aligned} E_{j+1}&=Y_{j+1}-U_{j+1}=(2Y_j^2-I)-(2U_j^2-I)\\ &=2(Y_j^2-U_j^2)\\ &=2(Y_j+U_j)(Y_j-U_j)=2(Y_j+U_j)E_j, \end{aligned}$$

where we have used the fact that Y_j and U_j are both analytic functions of A, and so $Y_jU_j=U_jY_j$. Taking norms and applying (11) and (15), we obtain

$$\|E_{j+1}\|_2\leqq 2\|Y_j+U_j\|_2\|E_j\|_2\leqq 4\|E_j\|_2. \tag{16}$$

From this, an induction gives

$$\|E_j\|_2\leqq 2^{2j}\|E_0\|_2, \qquad j=1,\ldots,N. \tag{17}$$

Even with the assumptions we have made, this bound appears pessimistic, but it still yields, with appropriate choice of $R(z)$ and N, in exact arithmetic, approximations to $\cos A$ of arbitrarily close accuracy. (This analysis does not include the effect of roundoff.)

Of the approximation $R(z)$, we demand

$$|R(z)|\leqq 1 \quad \forall |z|\leqq\eta, \qquad \text{(stability)}, \tag{18}$$

$$|\cos z-R(z)|\leqq C_\nu|z|^\nu \quad \forall |z|\leqq\zeta, \tag{19}$$

with η,ζ, C_ν constants. These conditions are met by the usual Taylor and Padé approximates as well as by a new family of rational approximates, introduced in [2], defined by

$$R_x^s(z)=\sum_{j=0}^{s}\phi_j^{(s)}(x)z^{2j}/(1+x^2z^2)^s, \tag{20}$$

with

$$\phi_j^{(s)}(x)=\sum_{k=0}^{j}\frac{(-1)^k}{(2k)!}\binom{s}{j-k}x^{2(j-k)}. \tag{21}$$

Here, x is a free parameter. For x large enough (i.e., $x\geqq x^{(s)}$) with η infinite, and if z is real and nonnegative and $\nu=2s+2$ in (19), then (18) holds. If $0\leqq x\leqq x^{(s)}$, then η is finite and must be determined for each s, but $\nu=2s+4$ is achievable (cf. [2]). The advantage to the approximation (20) lies in the computational efficiency it affords in the calculation of $R_x^s(A/2^N)$; replacing the scalar z by a matrix argument $B\equiv A/2^N$

requires that we solve

$$(I+x^2B^2)R_x^s(B)=\sum_{j=0}^{s}\phi_j^{(s)}(x)B^{2j}. \tag{22}$$

Then, performing a direct solution by elimination requires triangular factorization of only *one* real matrix. In contrast, the higher order Padé schemes require either factorization of a high order polynomial in B^2 into quadratic factors or complex linear factors, or explicit determination of the high order polynomial, followed by elimination. As a further remark, we observe that in (20), only even powers of B (hence A) are involved. Thus, if the differential equation (2) is actually given by $Y''+CY=0$, we never have to compute $A=C^{1/2}$.

Under assumptions (18) and (19), satisfied by choosing N large enough, it is easy to show from (17) that

$$\|\cos A-U_N\|\leqq C_\nu 2^{N(2-\nu)}[\rho(A)]^\nu. \tag{23}$$

Suppose that $2^{\beta-1}\leqq\rho(A)<2^\beta$. Then, from (23),

$$\|\cos A-U_N\|\leqq 2^{2N+(\beta-N)\nu}. \tag{24}$$

We first require that the order of accuracy in (19) satisfy $\nu>2$. Then in order that the bound in (24) decrease for ν fixed, we choose $N>\beta\nu/(\nu-2)$. For $\beta\leqq 0$, any positive N works. For $\beta>0$, it is easily seen that as ν increases, the function $g(\nu)=\beta\nu/(\nu-2)$ is decreasing; thus increasing the order of accuracy (ν) results (as we would expect) in a decrease in the number of steps N needed to achieve a given accuracy. In fact, it is easily seen from (24) that for any $\varepsilon>0$ we can achieve $\|\cos A-U_N\|\leqq\varepsilon$ if we choose $N>(\beta\nu-\log_2\varepsilon)/(\nu-2)$. Obviously, different choices of N and ν are possible to achieve the same tolerance ε. It would be possible to select optimal pairs on the basis of work estimates for a given ε.

The above direct analysis of the absolute error is limited to solutions for which the terms Y_j and U_j remain bounded for all j. A backward error analysis to obtain relative error bounds, similar to that performed by Moler and Van Loan [4] for the Padé approximations for the exponential, might extend the analysis appreciably, but the techniques of proof employed by Moler and Van Loan are particular to the exponential, and the required analysis for our case has not been performed.

4. Numerical results. We have performed many experiments using the double angle method to compute $\cos A$. We shall present here only a small representative sample of the results; other experiments are reported in [3]. The experiments we have run are for relatively small matrices ($n\leqq 10$), but the size of the matrix seems to have little effect on the performance of the method (except, of course, with regard to execution time). We have tested the method on both normal and non-normal matrices, but all sample matrices have only real eigenvalues and are diagonalizable. From the results obtained this far, although normality plays a crucial role in the error estimation we have done, it apparently is not needed for the successful application of the method to this class of problems.

We have used for $R(z)$, Taylor approximations of varying orders,

$$T^p(z)=\sum_{j=0}^{p}\frac{(-1)^j}{(2j)!}z^{2j}, \tag{25}$$

TABLE 1

N	$T^2(z)$ ($\nu=6$)	$T^3(z)$ ($\nu=8$)	$T^6(z)$ ($\nu=10$)	$R_{SN}(z)$ ($\nu=6$)	$R_x^2(z)$ ($\nu=8$)
1	.2952(−2)	.5301(−4)	.2469(−10)	.3972(−2)	.8237(−3)
2	.1639(−3)	.7330(−6)	.4552(−14)	.2393(−3)	.1432(−6)
4	.6181(−6)	.1725(−9)		.9256(−6)	.3634(−8)
6	.8497(−11)	.9853(−13)		.3613(−8)	.2057(−11)
8	.3883(−10)			.1301(−10)	.2903(−10)

for which $\nu=2p+2$; several Padé schemes, of which we single out the Störmer-Numerov approximation

$$R_{SN}(z)=\frac{1-\frac{5}{12}z^2}{1+\frac{1}{12}z^2}, \qquad \nu=6; \tag{26}$$

and, from the family of approximations (20), we select

$$R_x^2(z), \quad s=2, \quad x=\hat{x}\equiv\sqrt{\frac{5+\sqrt{15}}{60}}\ , \tag{27}$$

for which it can be shown that $\nu=8$. Both (26) and (27) are conditionally stable approximations, i.e., η finite in (18). It may be that for large problems with widely separated eigenvalues we might want to go to unconditionally stable approximations (cf. Table 3 below for some evidence).

The test problems have been constructed by applying a similarity transformation $A=P^{-1}DP$, to diagonal D, so that the actual cosine can easily be determined for error computation ($\cos A=P^{-1}\cos DP$). For the first class of problems, P is orthogonal (a Householder reflector) so that A is normal (in fact, symmetric). In Table 1 we show the error, measured for convenience in the matrix norm $\|E\|_\infty$, where E is the difference between the actual and approximated-by-double-angle cosine. In this case, $n=4$ and $D=\text{diag}\{-1, 1, -2, 2\}$. Recall that N represents the number of times the double angle formula is applied.

Apparently, for a small problem, the Taylor approximations of high order work well, as do the rational approximations used. One can see in the first and last columns the effect of roundoff error as N increases; eventually, for any method, increasing N will yield diminishing returns due to roundoff.

In Table 2, we show results of an experiment with $n=10$, $D=\text{diag}\{-1, -2,\ldots, -10\}$, A again symmetric.

TABLE 2

N	$T^2(z)$	$T^3(z)$	$T^4(z)$	$R_{SN}(z)$	$R_x^2(z)$
1	.1017(4)	.2562(3)	.3224	.2562(2)	.1829(2)
2	.2881(1)	.7218	.5844(−4)	.4142(1)	.1925(1)
4	.8957(−2)	.5267(−4)	.5441(−11)	.2583(−2)	.2341(−3)
6	.3341(−4)	.1224(−7)		.9906(−5)	.6063(−7)
8	.1302(−6)	.7213(−9)		.3866(−7)	.1780(−10)

TABLE 3

N	$T^2(z)$	$R_{SN}(z)$	$R_x^2(z)$	$R_{\tilde{x}}^2(z)$
1	.1343(12)	.4757(2)	.2763(2)	.1275
2	.5197(18)	.4372(4)	.1535(4)	.2089(1)
4	.9403(31)	.1811(14)	.6840(12)	.1748(1)
6	.1060(1)	.2545	.1366	.7562
8	.1680(−2)	.2461(−2)	.9206(−4)	.7724(−1)

This example clearly shows the need for scaling; other examples we have tested with A having larger norm are even more convincing, and show the effect of instability in the approximation. In Table 3, even for $n=2$, compare results with the Taylor and rational approximations to results using (20) with

$$x=\tilde{x}\equiv\sqrt{\tfrac{1}{4}+\sqrt{1/24}}\ ,$$

which can be shown to be unconditionally stable, with $v=6$. We use $D=\text{diag}\{-100, 100\}$. Obviously, if N is not large enough, the former approximations exhibit wild instability, while the latter one, although perhaps not too accurate, does not blow up. Of course, choosing N large enough brings $\|A/2^N\|$ back into the range for the conditionally stable methods, where they are more accurate.

Finally, as an example of a problem where A is not normal, we use similarity to construct $A=P^{-1}DP$; but this time P is not orthogonal. In fact,

$$P=\begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & -1 \\ 0 & & & & -1 & 2 \end{bmatrix}.$$

In Table 4 we present results for a problem with $n=10$, $D=\text{diag}\{\pm1, \pm2, \pm3, \pm4, \pm5\}$, which are qualitatively quite similar to those for the normal (symmetric) case.

We conclude that the double-angle method shows great promise for the cosine computation, and experiments and analysis should be extended to cases not treated here. We should mention that we have made some preliminary comparisons of the double angle method with one suggested by Parlett [5], in which A is first reduced to

TABLE 4

N	$T^2(z)$	$T^3(z)$	$T^6(z)$	$R_{SN}(z)$	$R_x^2(z)$
1	.4556(1)	.6235	.6797(−9)	.5435(1)	.3239(1)
2	.1292	.3337(−2)	.5405(−8)	.1657	.4522(−1)
4	.4068(−3)	.6470(−6)	.1313(−10)	.6043(−3)	.1329(−4)
6	.1569(−5)	.2900(−9)		.2353(−5)	.3427(−8)
8	.6422(−8)			.1053(−7)	.2916(−8)

real Schur form and then an analytic function of a quasi-triangular matrix is computed one superdiagonal at a time. Our results indicate that both methods arrive at the same cosine matrix, but no detailed comparison (e.g., work estimates) has been performed; this should be the subject of future investigation.

REFERENCES

[1] T. APOSTOL, *Explicit formulas for solutions of the second-order matrix differential equation* $Y'' = AY$, Amer. Math. Monthly, 82 (1975), pp. 159–162.

[2] G. A. BAKER, V. A. DOUGALIS AND S. M. SERBIN, *An approximation theorem for second-order evolution equations*, to appear in Numer. Math.

[3] S. BLALOCK, *Computation of the Cosine of a Matrix*, M.S. Thesis, University of Tennessee, 1979.

[4] C. B. MOLER AND C. F. VAN LOAN, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev., 20 (1978), pp. 801–836.

[5] B. N. PARLETT, *Computations of Functions of Triangular Matrices*, Memo No. ERL-M481, University of California, Berkeley, 1974.

[6] S. M. SERBIN, *Rational approximations of trigonometric matrices with application to second-order systems of differential equations*, Appl. Math. and Comput., Vol. 5, (1979), pp. 75–92.

[7] R. S. VARGA, *On higher order stable implicit methods for solving parabolic partial differential equations*, J. Mathematical Physics, 40 (1961), pp. 220–231.

[8] R. C. WARD, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–619.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0005 $1.00/0

ESTIMATING MATRIX CONDITION NUMBERS*

DIANNE PROST O'LEARY†

Abstract. In this note we study certain estimators for the condition number of a matrix which, given an LU factorization of a matrix, are easily calculated. The main observations are that the choice of estimator is very norm-dependent, and that although some simple estimators are consistently bad, none is consistently best. These theoretical conclusions are confirmed by experimental data, and recommendations are made for the one and infinity norms.

Key words. Matrix condition number

1. Introduction. Cline, Moler, Stewart, and Wilkinson [1] give an excellent exposition of various methods for estimating the condition number of a matrix

$$\kappa(A) = \|A\|\|A^{-1}\|,$$

where $\|\cdot\|$ is some matrix norm compatible with a vector norm. One of the applications considered is that of estimating κ given a LU factorization of a matrix formed using partial pivoting:

$$A = LU,$$

where L is unit lower triangular, U is upper triangular, and all elements of L are bounded in absolute value by 1. The strategy suggested is to solve two linear systems,

$$A^T x = e,$$

$$Ay = x,$$

and to use $\|y\|/\|x\|$ as the estimate for $\|A^{-1}\|$. Here the vector e is chosen during the first step of the solution procedure, finding z such that $U^T z = e$. Each element e_i is ± 1, with sign to promote growth in the subsequent components of z. $\|A\|_1$ or $\|A\|_\infty$ can easily be calculated exactly, and $\|A\|_2$ can be estimated using, for example, the power method.

The experiments in [1] show this to be a good algorithm for the one-norm, but the strategy is norm-dependent as suggested by the following example.

Example. Let A_k be the Hadamard matrix of dimension $n = 2^k$ defined by

$$A_1 \equiv \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \equiv L_1 U_1,$$

$$A_k \equiv \begin{bmatrix} A_{k-1} & A_{k-1} \\ A_{k-1} & -A_{k-1} \end{bmatrix} = \begin{bmatrix} L_{k-1} & 0 \\ L_{k-1} & L_{k-1} \end{bmatrix} \begin{bmatrix} U_{k-1} & U_{k-1} \\ 0 & -2U_{k-1} \end{bmatrix} \equiv L_k U_k, \quad k > 1.$$

It is easy to see that to estimate $\|A_k^{-1}\|$, the choice of e will be such that $A_k^T e_n = e$,

*Received by the editors September 26, 1979.

This work was supported by the National Bureau of Standards and the U.S. Office of Naval Research under Grant N00014-76-C-0391.

†Computer Science Department and Institute for Physical Science and Technology, University of Maryland, College Park, Maryland, and National Bureau of Standards, Gaithersburg, Maryland.

where e_n is the nth unit vector. Then $y = A_k^{-1} e_n = e/n$, and we obtain the estimates

$$\|A^{-1}\|_1 \simeq 1, \qquad \|A^{-1}\|_1 = 1,$$

$$\|A^{-1}\|_2 \simeq 1/\sqrt{n}, \qquad \|A^{-1}\|_2 = 1,$$

$$\|A^{-1}\|_\infty \simeq 1/n, \qquad \|A^{-1}\|_\infty = 1.$$

Although the one-norm estimate is exact, the infinity norm estimate is off by a factor of n.

2. Methods. To study the behavior of norm estimates, we develop some basic relations. Recall that the one-norm of a matrix is the maximum absolute column sum: if a matrix B has columns b_j with components b_{ij} then

$$\|B\|_1 = \max_{x \neq 0} \frac{\|Bx\|_1}{\|x\|_1} = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |b_{ij}| = \max_{j=1,\ldots,n} b_j^T \tilde{e}_j,$$

where $\tilde{e}_j$ is a vector with components ± 1, with signs chosen to match those of b_j. Also recall that $\|B\|_\infty = \|B^T\|_1$. The strategy defined above is designed to produce a vector x which is close to being a maximizer for $\|Bx\|/\|x\|$, where $B = A^{-1}$. In the course of computing the vector x, another estimate, based on the sizes of the vectors e and $x = B^T e$, is available for the condition number. Let

$$\mu_1 = \|y\|_1/\|x\|_1,$$

$$\nu_1 = \|x\|_\infty/\|e\|_\infty,$$

$$\mu_\infty = \|y\|_\infty/\|x\|_\infty,$$

$$\nu_\infty = \|x\|_1/\|e\|_1,$$

where the first two estimate $\|A^{-1}\|_1$ and the last two estimate $\|A^{-1}\|_\infty$. Then

$$\nu_1 = \max_{j=1,\ldots,n} |b_j^T e|,$$

$$\nu_\infty = \frac{1}{n} \sum_{j=1}^{n} |b_j^T e|,$$

$$\mu_1 = \frac{\sum_{i=1}^{n} \left| \sum_{j=1}^{n} (b_j^T e) b_{ij} \right|}{\sum_{j=1}^{n} |b_j^T e|},$$

$$\mu_\infty = \frac{\max_{i=1,\ldots,n} \left| \sum_{j=1}^{n} (b_j^T e) b_{ij} \right|}{\max_{i=1,\ldots,n} |b_i^T e|}.$$

From this we obtain the relationships

$$\frac{\nu_1}{\mu_1} = \frac{\sum_{j=1}^{n} |b_j^T e|}{\sum_{i=1}^{n} \left| \sum_{j=1}^{n} \frac{(b_j^T e)}{\nu_1} b_{ij} \right|},$$

$$\nu_1 \geqq \nu_\infty,$$

$$\frac{\nu_\infty}{\mu_\infty} = \frac{\frac{1}{n} \sum_{j=1}^{n} |b_j^T e|}{\max_{i=1,\ldots,n} \left| \sum_{j=1}^{n} \frac{(b_j^T e)}{\nu_1} b_{ij} \right|} \leqq \frac{\nu_1}{\mu_1},$$

The third relation says that the improvement gained by solving the second linear system, $Ay = x$, is greater for the infinity norm estimate than for the one-norm estimate. The second relation says that the first one-norm estimate is an upper bound for the first infinity norm estimate. For a symmetric matrix the two norms are equal and so the one-norm estimate is always more accurate. (No such relation exists between the estimates μ_1 and μ_∞.) The first equation gives the relation between the two one-norm estimators.

Unfortunately none of the estimators can be labeled as best. The estimate ν_1 (respectively, ν_∞) is sometimes greater than μ_1 (μ_∞) and sometimes less. Because of this, estimators which use all the information might be more useful:

$$\rho_1 = \max(\mu_1, \nu_1),$$

$$\rho_\infty = \max(\mu_\infty, \nu_\infty).$$

To gain a better understanding of the behavior of the condition number estimators, tests were performed on matrices with elements taken from a uniform distribution on $[-1, 1]$. The LINPACK [2] routine SGECO, which factors a matrix and returns an estimate of the condition number based on μ_1, was modified to compute all of the estimators. LINPACK's SGEDI was used to compute the inverse so that $\|A^{-1}\|$ could be calculated for comparison. Test results are summarized in Tables 1 and 2. Results for $5 \leqq n \leqq 50$ were obtained using 100 matrices of each dimension n. A distribution-free method gave confidence intervals for the medians [3]. From this data we make the following observations:

(1) For small $n (n < 20)$, ν_1 produced a better estimate than the LINPACK estimate in over 50% of the trials. For larger $n (n = 50)$ the estimate μ_1 was better approximately 80% of the time.

(2) The estimate ρ_1, the maximum of these estimates, was a noticeable improvement over both ν_1 and μ_1 for small n.

(3) For each set of trials, the estimate ν_1 had a higher maximum than μ_1 (except on symmetric matrices of dimension 50), and a lower minimum (except on general matrices of dimension 5). For small matrices ν_1 was often exact ($\nu_1 / \|A^{-1}\|_1 = 1$ in 42 trials out of 100 for $n = 5$) and for such matrices, μ_1 was often 30% smaller.

(4) The first estimate of the infinity norm, ν_∞, was unreliable and the second was almost always better.

TABLE 1

Results for matrices with elements from a uniform distribution[1]

GENERAL MATRICES

n	*Average* $\nu_1/\|\|A^{-1}\|\|_1$	*Average* $\mu_1/\|\|A^{-1}\|\|_1$	*Average* $\rho_1/\|\|A^{-1}\|\|_1$	% of Trials $\nu_1 > \mu_1$	*Average* $\nu_\infty/\|\|A^{-1}\|\|_\infty$	*Average* $\mu_\infty/\|\|A^{-1}\|\|_\infty$	*Average* $\rho_\infty/\|\|A^{-1}\|\|_\infty$	% of Trials $\nu_\infty > \mu_\infty$
5	.84	.69	.86	80	.45	.66	.67	3
10	.70	.60	.74	70	.31	.60	.60	0
20	.49	.52	.57	48	.19	.53	.53	0
30	.43	.48	.52	40	.15	.48	.48	0
40	.31	.43	.45	21	.11	.44	.44	0
50	.31	.45	.46	19	.10	.45	.45	0

SYMMETRIC MATRICES

n	*Average* $\nu_1/\|\|A^{-1}\|\|_1$	*Average* $\mu_1/\|\|A^{-1}\|\|_1$	*Average* $\rho_1/\|\|A^{-1}\|\|_1$	% of Trials $\nu_1 > \mu_1$	*Average* $\nu_\infty/\|\|A^{-1}\|\|_\infty$	*Average* $\mu_\infty/\|\|A^{-1}\|\|_\infty$	*Average* $\rho_\infty/\|\|A^{-1}\|\|_\infty$	% of Trials $\nu_\infty > \mu_\infty$
5	.83	.66	.83	88	.46	.66	.66	2
10	.66	.58	.71	69	.29	.55	.55	0
20	.53	.50	.59	58	.20	.49	.49	0
30	.38	.44	.48	33	.14	.44	.44	0
40	.35	.45	.47	31	.12	.44	.44	0
50	.30	.40	.42	27	.10	.41	.41	0

[1]*Note*. As part of the refereeing process for this paper, the results reported in Table 1 were substantiated through similar tests by Curtis Hunt at the University of New Mexico.

TABLE 2
99% confidence intervals for the medians for trials on general matrices

n	$\nu_1/\|\|A^{-1}\|\|_1$	$\mu_1/\|\|A^{-1}\|\|_1$	$\rho_1/\|\|A^{-1}\|\|_1$
5	.80–1.00	.67–.73	.83–1.00
10	.60– .80	.57–.65	.67– .80
20	.42– .55	.50–.57	.54– .61
30	.33– .50	.46–.52	.48– .56
40	.23– .34	.41–.49	.41– .50
50	.23– .33	.43–.49	.44– .50

(5) The infinity norm estimates had consistently larger relative error than the one-norm estimates.
(6) Results for matrices with elements randomly 1, 0, or -1 were similar to those tabulated.

3. Conclusion. An inexpensive algorithm to estimate the one-norm of A^{-1} using an LU factorization of A, is:

(a) Solve $A^T x = e$ where e is chosen as described above. Let $\nu_1 = \|x\|_\infty$.
(b) Solve $Ay = x$ and let $\mu_1 = \|y\|_1 / \|x\|_1$.
(c) Estimate $\|A^{-1}\|_1$ by $\rho_1 = \max(\nu_1, \mu_1)$.

This costs only n comparisons more than the LINPACK algorithm and gives more reliable results for small matrices.

To estimate the infinity norm of A^{-1}, the above algorithm should be applied to the matrix A^T. This gives better results than the procedure formed by interchanging the roles of the one and infinity norms in the three steps above.

Acknowledgment. This work benefited from helpful comments of G. W. Stewart.

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] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH, AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.
[3] WILLIAM J. MACKINNON, *Table for both the sign test and distribution-free confidence intervals of the median for sample sizes to* 1000, J. Amer. Statist. Assoc. (1964), pp. 935–956.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0006 $1.00/0

USE OF THE SINGULAR VALUE DECOMPOSITION WITH THE MANTEUFFEL ALGORITHM FOR NONSYMMETRIC LINEAR SYSTEMS*

PAUL E. SAYLOR†

Abstract. Optimum Chebyshev parameters may be computed dynamically by the Manteuffel algorithm for use with a generalization of Richardson's iterative method and the Jacobi semi-iterative method to solve nonsymmetric linear algebraic systems. The algorithm determines the convex hull of eigenvalues of a matrix associated with the system and from the convex hull determines the parameters. The singular value decomposition may be used to test the reliability of a simple technique to reduce execution and storage costs of the power method. The same technique also yields an inexpensive singular value decomposition, making it feasible to determine precisely when to call the Manteuffel algorithm.

Key words. singular value decomposition, Manteuffel algorithm, nonsymmetric, linear algebraic equations, Richardson's method, Jacobi method, eigenvalues, power method, matrix

1. Introduction. The main concern here is the efficient use of the Manteuffel algorithm [11] for computing Chebyshev parameters to accelerate the iterative solution of a real nonsymmetric system of linear algebraic equations, $Ax=b$. There are two objectives. One is to determine when to call the Manteuffel algorithm; the other is to reduce the costs of execution and storage. A result of knowing when to call the algorithm is that the iteration takes fewer steps.

Chebyshev acceleration parameters are computed from the convex hull of eigenvalues of A or $M^{-1}A$, where M results from a splitting [2] of A. Eigenvalues are obtained from the power method, which, for a real square matrix B with a single dominant eigenvalue, is based on the observation that under the proper assumptions the Krylov sequence $\{y_j: y_j=B^j y_0, 0\leqq j\leqq k\}$ tends to that eigenvector corresponding to the dominant eigenvalue. A dominant nonreal eigenvalue of a real matrix is not unique because eigenvalues and the corresponding eigenvectors appear in conjugate pairs. A dominant nonreal eigenvalue would cause y_k, for large enough k, to be a linear combination of two eigenvectors, one the conjugate of the other. If so, y_k, y_{k+1}, and y_{k+2} are linearly dependent. The coefficients expressing linear independence,

$$\gamma_0 y_k+\gamma_1 y_{k+1}+\gamma_2 y_{k+2}=0, \tag{1}$$

are seen to be the coefficients of a polynomial whose roots are the dominant eigenvalue and its conjugate. A least squares solution of the overdetermined system (1) yields the coefficients [17, p. 580].

The computation is less expensive and takes less storage if fewer, randomly selected components of y_k, y_{k+1}, y_{k+2} are used, which is equivalent to replacing y_k, y_{k+1}, y_{k+2} in the overdetermined system by their projections, $P(y_k)$, $P(y_{k+1})$, $P(y_{k+2})$ onto a subspace (cf. [9]). The projections must also be linearly dependent.

The linear dependence of a set of vectors, $\{v_1,\cdots,v_p\}$, is equivalent to the existence of a zero singular value of the matrix $(v_1\cdots v_p)$. The question of whether $P(y_k)$, $P(y_{k+1})$, and $P(y_{k+2})$ yield satisfactory eigenvalues reduces to a test of the smallest (in magnitude) singular value of the matrix, D, with these column vectors. Other tests are possible. As an alternative one could determine the rank of D^TD

*Received by the editors September 12, 1979, and in revised form March 27, 1980. This research was supported in part by the National Science Foundation under Grant NSF 76-81100 and NSF 79-06123.

†Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

(suggested by Manteuffel [12. p. 189] for a slightly different application), but an efficient, stable algorithm due to Businger and Golub [1] is available to compute singular values and is generally superior [13, p. 382] to this approach.

If the smallest singular value is too large, that is, if the column vectors of

$$D=(P(y_k), P(y_{k+1}), P(y_{k+2})) \tag{2}$$

are independent, there is no information in the singular values to suggest whether it is because k should have been larger or because the projection was onto a badly chosen subspace. The smallest singular value is like clinical temperature, which may show that a patient is sick, but gives no diagnosis. Linear dependence results in the limit as k increases if the matrix is not defective. However, linear dependence does not necessarily mean that the dominant eigenvalue or any eigenvalue can be approximately computed, for it is possible, although unlikely, that $P(y_k)=0$ for all k. Even if $P(y_k)$ is nonzero, the dominant eigenvalue cannot be computed unless $P(y_k)$ has a nonzero component along the projection of the dominant eigenvector. Despite what could go wrong, it is not unreasonable to dismiss these risks. First, the eigenvalues of interesting, application-derived matrices may be assumed to be distinct, which assures that the matrix is not defective. Second, it is improbable that there would be no trace of the dominant eigenvector in $P(y_k)$.

In practice, rather than test the smallest (in magnitude) singular value to determine linear dependence, the ratio of the smallest to the largest (in magnitude) should be tested. If there is linear dependence, the next step is to solve the least squares problem, which may be done at no additional cost by using the singular value decomposition.

No theoretical study is presented here of the problem of how small a number of components would be satisfactory, although numerical experiments are presented to support the technique. If the dominant eigenvalue were real and approximated by the least squares solution, γ_0, of

$$\gamma_0 P(y_k)=P(y_{k+1}),$$

then γ_0 would be a weighted mean of the ratio of the components of $P(y_{k+1})$ and $P(y_k)$. It is a familiar statistical sampling problem to compare the mean of a sample to the mean of the population. The problem of eigenvalue accuracy is thus a variant of the mean sampling problem.

Current implementations of the Manteuffel algorithm compute new eigenvalues after a fixed number of steps of the power method, a costly strategy if the eigenvector estimates are mature after fewer steps. Linear dependence should be monitored in order to compute new eigenvalues precisely when they are accurate. If the number of rows of D were small, it would be economical to compute singular values for this after each iterative step, and so new eigenvalues could be computed precisely when they were available, resulting in an efficient iteration. Computing singular values to monitor linear dependence thus yields two benefits: lower overhead in executing the Manteuffel algorithm, and fewer iterative steps.

The basic facts about Richardson's method are presented in §2, and the techniques to estimate iteration parameters dynamically (the Manteuffel algorithm) are explained briefly in §3. Richardson's method is used to explain the dynamic computation of parameters in the Manteuffel algorithm and the relation of the power

method to matrix iterative methods, but the Manteuffel algorithm can be applied to other iterative methods as well, for example, to the Jacobi semi-iterative method [16], [19]. In practice, Richardson's method is often used with a matrix splitting [2], [16]. In the remainder of the paper, details of the power method and the technique of using the singular value decomposition are presented, together with the results of numerical experiments.

2. Richardson's method. The basic iteration is a gradient procedure,

$$x^{(k+1)}=x^{(k)}-t_k(Ax^{(k)}-b). \tag{3}$$

The error, defined to be $e^{(k)}=x-x^{(k)}$, satisfies

$$e^{(k)}=(I-t_{k-1}A)e^{(k-1)}=R_k(A)e^{(0)}, \tag{4}$$

where

$$R_k(\lambda)=(1-t_{k-1}\lambda)\cdots(1-t_0\lambda),$$

called the *residual polynomial* [14, p. 6], is such that

$$R_k(0)=1. \tag{5}$$

For A nonsymmetric with eigenvalues in, say, the right halfplane, the gradient procedure converges if $R_k(\lambda)=P_k(\lambda)$, where

$$P_k(\lambda)=T_k[(d-\lambda)/c]T_k(d/c),$$

and where T_k is the Chebyshev polynomial of degree k. P_k will be called the *Chebyshev residual polynomial*. The resulting iteration is called Richardson's method [19]. Parameters d and c are such that d is the center and $d\pm c$ are the foci of a family of ellipses such that one member contains the eigenvalues of A and lies in the right (or left) halfplane. The minimax polynomial is that polynomial, subject to (5), whose maximum modulus over any ellipse of the family is less than or equal to the maximum of any such polynomial over that ellipse. If d and c^2 are real, the Chebyshev residual polynomial either is the minimax polynomial or else approximates it asymptotically [11, p. 315]. (A negative c^2 means a vertical major axis.)

The Chebyshev residual polynomial depends on the center and focal length of the enclosing ellipses. Optimal values of the center and focal length are determined by the convex hull of the eigenvalues, which may be computed dynamically by the power method, as explained next. (Richardson's method in the form above, (3), is awkward and unstable, but convenient for discussion. Another version of the iteration [11, p. 316] is recommended in practice.)

3. Dynamic estimation of the eigenvalues. The residual error at step k,

$$r^{(k)}=b-Ax^{(k)}=Ae^{(k)},$$

satisfies

$$r^{(k)}=P_k(A)r^{(0)},$$

analogous to (4) for the error. For large k, $P_k(\lambda)\doteq[M(\lambda)]^k$, where $M(\lambda)=(d-\lambda+[(d-\lambda)^2-c^2]^{1/2})/[d+(d^2-c^2)^{1/2}]$, an approximation which follows from $T_k(w)\doteq e^{k\cosh^{-1}w}/2$. Therefore,

$$r^{(k)}\doteq[M(A)]^k r^{(0)}.$$

Vectors $r^{(k)}$ thus are the iterates of the power method applied to the matrix $M(A)$. If μ_1 and $\bar{\mu}_1$ are the dominant eigenvalues of $M(A)$ corresponding to eigenvectors v_1 and

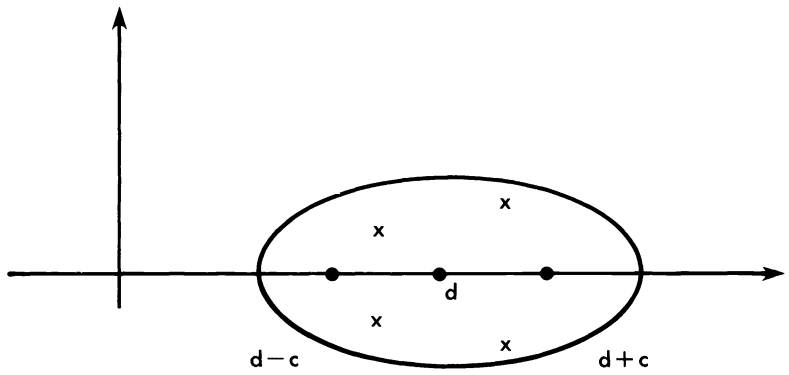


FIG 1. *The eigenvalues of A must lie in an ellipse with center d and foci d±c contained in the right (or left) halfplane.*

$\bar{v}_1$, which are also eigenvectors of A, then $r^{(k)}$ is approximately a linear combination of v_1 and $\bar{v}_1$ from which μ_1 may be computed. An eigenvalue, λ_1, of A is the solution of $\mu_1 = M(\lambda_1)$. This summarizes the method by which the Manteuffel algorithm computes eigenvalues dynamically [10], [11], [12] (cf. [9], [18]).

4. The power method for a nonsymmetric matrix. Let B be an $n \times n$ nonsymmetric real matrix with a dominant complex eigenvalue λ_1 corresponding to eigenvector v_1. Complex eigenvalues and eigenvectors occur in conjugate pairs. Assume for convenience that B is nondefective, that is, any vector y_0 may be expressed as a sum of eigenvectors,

$$y_0 = \beta_1 v_1 + \bar{\beta}_1 \bar{v}_1 + \cdots.$$

If $\beta_1 \neq 0$, multiplication by B^k yields

$$y_k = B^k y_0 = \beta_1 \lambda_1^k v_1 + \bar{\beta}_1 \bar{\lambda}_1^k \bar{v}_1 + \cdots, \tag{6}$$

in which the dominant term is a sum of two eigenvectors. An estimate of λ_1 may be computed from y_k and two additional terms, y_{k+1} and y_{k+2}, by a method [17, p. 580] which follows from the observation that if γ_0 and γ_1 are the coefficients of the polynomial $p_2(\lambda) = (\lambda - \lambda_1)(\lambda - \bar{\lambda}_1) = \gamma_0 + \gamma_1 \lambda + \lambda^2$, then

$$\gamma_0 y_k + \gamma_1 y_{k+1} \doteq -y_{k+2}. \tag{7}$$

Values of γ_0 and γ_1 that make the left side nearly equal the right are therefore approximate values of the coefficients of $p_2(\lambda)$. This is an overdetermined system in two unknowns, γ_0 and γ_1, a solution of which may be computed from the method of least squares. The solution of the least squares problem is only an approximation to the coefficients of p_2. Nevertheless, the same symbols are used to denote the solution. For convenience, only the computation of λ_1 and $\bar{\lambda}_1$ is discussed here. In practice the power method is used to compute at least four eigenvalues at a time [9, p. 18].

Let $E = (y_k, y_{k+1})$ be the $n \times 2$ matrix of system (7). Thus,

$$E\begin{pmatrix}\gamma_0\\ \gamma_1\end{pmatrix} = -y_{k+2}.$$

The least squares solution is the solution of the 2×2 system of normal equations

$$E^T E\begin{pmatrix}\gamma_0\\ \gamma_1\end{pmatrix} = -E^T y_{k+2}.$$

The work required to form the normal equations is $5(n-1)$ multiplications, which takes into account the symmetry of E^TE.

Equation (7) is not necessarily the proper way to state the least squares problem, but this form is the one followed elsewhere [10], [12]. Also, it is well known that a solution of the normal equations is not as accurate as a solution obtained from the QR decomposition.

The failing of (7) is an overemphasis on the coefficient of y_{k+2}, which is assumed to be unity. An unbiased formulation of the least squares problem is as follows. Compute the least squares solution of

$$Dc=0 \tag{8}$$

subject to

$$\|c\|_2^2=1, \tag{9}$$

where $D=(y_k, y_{k+1}, y_{k+2})$ and $c^T=(\gamma_0, \gamma_1, \gamma_2)$.

5. A reduction in the work to solve the least squares problem. To reduce the work, a subsystem of equations could be selected from the overdetermined system (8) at random, and the method of least squares applied to the subsystem. In this instance the number, n, of equations, or statistical observations, is so much greater than the number of parameters, γ_0, γ_1 and γ_2, that one can reasonably expect variations about the true value of the parameters to be uniform for a smaller number of observations. Reliability of the technique may be checked by computing the singular value decomposition of the subsystem.

Let $\eta_i, \eta_i', \eta_i''$ be the ith components of y_k, y_{k+1}, y_{k+2} respectively. The least squares solution of (8) will be approximated by the least squares solution of a smaller system

$$\begin{aligned}
\gamma_0\eta_{i_1}+\gamma_1\eta_{i_1}'+\gamma_2\eta_{i_1}''&=0,\\
&\vdots\\
\gamma_0\eta_{i_l}+\gamma_1\eta_{i_l}'+\gamma_2\eta_{i_l}''&=0,
\end{aligned}$$

subject to the same constraint (9). Whether the number of equations is adequate may be determined in a precise way. If there are no subdominant eigenvalues, the column vectors $P(y_k)=(\eta_{i_1}, \cdots, \eta_{i_l})^T, P(y_{k+1})=(\eta_{i_1}', \cdots, \eta_{i_l}')^T$, and $P(y_{k+2})=(\eta_{i_1}'', \cdots, \eta_{i_l}'')^T$ are linearly dependent. Therefore the matrix

$$D=(P(y_k), P(y_{k+1}), P(y_{k+2}))_{l\times 3}$$

has a singular value zero. If the system is small, it would be economical to compute the singular values as a check on accuracy.

There is an obvious alternative test of convergence of a sequence, $\lambda_1^{(k)}$, of approximate eigenvalues, namely, to compute $|\lambda_1^{(k)}-\lambda_1^{k-1}|/|\lambda_1^{(k)}|$ and halt when this ratio is less than some prescribed value. The singular value test is preferable because it determines convergence of the coefficients of the polynomial of which $\lambda_1^{(k)}$ is a root without any comparison to previous values. In a practical code, one may not want to compute eigenvalue approximations at each step but rather after, say, every five steps. It may well happen that $\lambda_1^{(k-5)}$ is a rough value while $\lambda_1^{(k)}$ is accurate, and so the alternative test could fail.

6. The singular value decomposition and the least squares problem. Any $l \times p$ matrix, D, may be decomposed in the form

$$D = U \Sigma V^T, \tag{10}$$

where U is an $l \times p$ orthogonal matrix, $\Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_p)$ is a $p \times p$ diagonal matrix, the diagonal elements of which are the singular values, and V is a $p \times p$ orthogonal matrix [5], [6], [13]. This is called the singular value decomposition (SVD). The ratio of the largest (in magnitude) singular value to the smallest measures the linear independence of the column vectors of D. The SVD reduces the question of linear independence to a single quantity, the ratio of the extreme singular values.

The singular value decomposition now yields the solution of the least squares problem,

$$\| Dc \|_2 = \text{minimum} \tag{11}$$

subject to

$$\| c \|_2 = 1, \tag{12}$$

where $c^T = (\gamma_0, \cdots, \gamma_{p-1})$. To express the solution, assume that the singular values of D are sorted in descending order of magnitude, with σ_p the smallest singular value [3, p. 11.4]. The solution of the least squares problem

$$\| \Sigma \tilde{c} \|_2 = \text{minimum}$$

subject to

$$\| \tilde{c} \|_2 = 1,$$

is $\tilde{c} = (0, \cdots, 0, 1)^T$. The solution of the least squares problem (11), (12) is $c = V\tilde{c}$. The approximate eigenvalues are the roots of $\gamma_0 + \cdots + \gamma_{p-1}\lambda^{p-1}$.

7. Numerical experiments. Two matrices were used in a set of numerical experiments, the tridiagonal heat flow matrix, $[-1, 2, -1]_{n \times n}$, and a test matrix of P. Eberlein [4], which is a 16×16 real matrix with complex eigenvalues. The purposes of the experiments are to compare the ratio of the singular values to the relative error of the eigenvalue approximation and to compare the eigenvalue accuracy obtained from all components of the iterates to the accuracy obtained from a small number of components. In each experiment the components of the initial iterate, y_0, were randomly generated.

The experiments are numbered to correspond to the figures. Experiments (2) through (5) were made with the heat flow matrix. At each step of the power method, an approximation to the dominant eigenvalue was computed from the matrix $(P(y_k), P(y_{k+1}))$. In each of Figs. 2 and 3 there are two curves, one the relative error in the approximate eigenvalue, and the other, the ratio of the smallest singular value to the largest singular value. The matrix is 64×64 ($n = 64$). In Fig. 2, the singular values and eigenvalues are computed from all the components, whereas in Fig. 3 only four randomly selected components are used: $P(y_k) = (y_{k,5}, y_{k,27}, y_{k,33}, y_{k,62})$. In both figures, relative error is correlated with the ratio of singular values.

In a practical code, one would not necessarily want to compute singular values at each step. Experience suggests that a computation every five steps is sufficient. Current implementations of the Manteuffel algorithm compute eigenvalues every twenty steps. Suppose we not only compute the SVD, and new eigenvalues if needed,

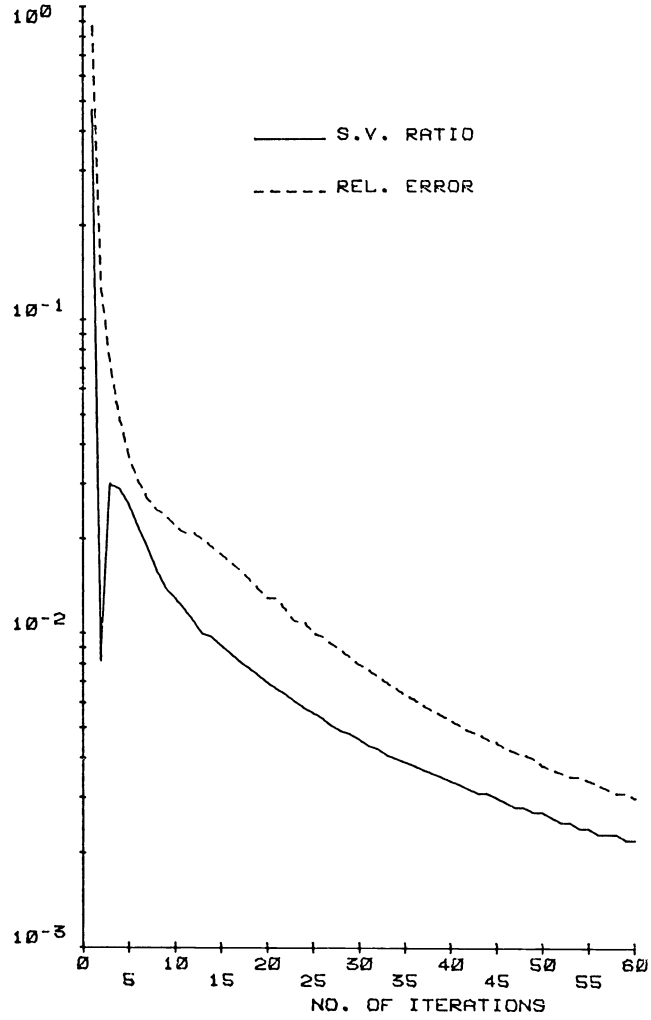


FIG 2. *Comparison of the ratio of the smallest to the largest singular value (solid line) with the relative error in the eigenvalue (dashed line). Number of components of* $y_k = 64$; *number of components of* $P(y_k) = 64$.

every five steps but also keep the work the same as now used by the Manteuffel algorithm, which is $5n$ multiplications to compute one eigenvalue and $14n$ to compute two, where n is the number of unknowns. If D is an $l \times p$ matrix, it takes $lp^2 - p^3/3$ multiplications [3, p. 9.21] to compute the QR decomposition of D, which is of the form $D = QR$, where Q is $n \times p$ and R is $p \times p$. The singular values and right singular vectors of R are the same as for D. Since R is a $p \times p$ matrix, the additional work to compute the SVD of R is negligible if $p \ll n$. The total work will be taken to be lp^2 multiplications. In experiment (3), the value $l = 4$ is 6% of $n = 64$.

The amount, 6%, is not a lower limit. In experiment (4), l is 1% of $n (n = 500)$. The approximate values obtained from five components oscillate around the values obtained from five hundred components. One would expect the five-component values to be less accurate, but for most steps they are more accurate. Even when the

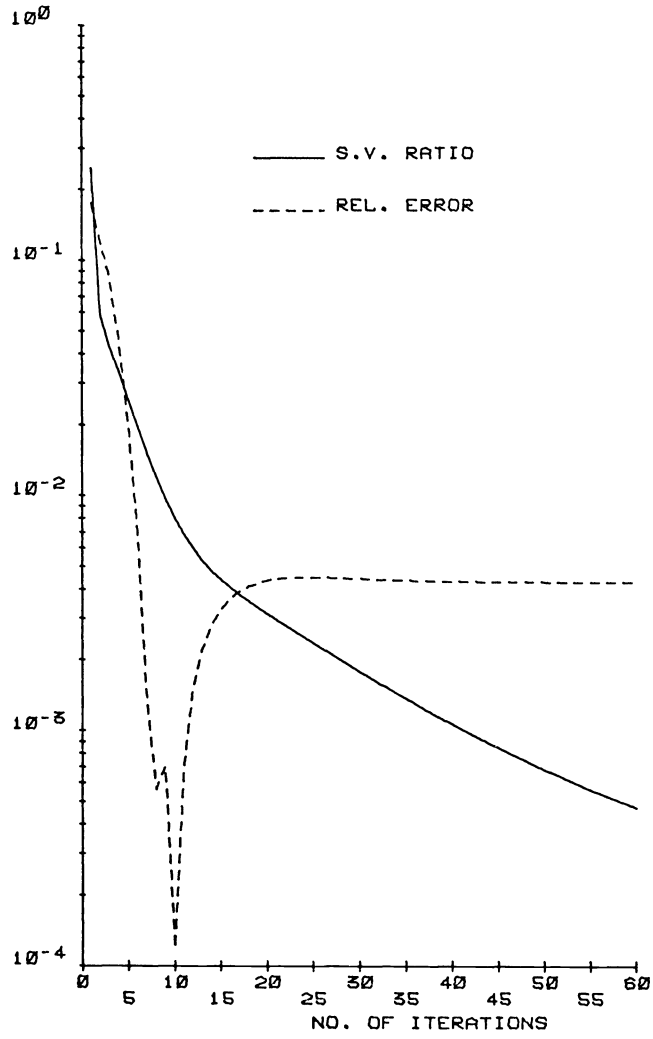


FIG 3. *Comparison of the ratio of the smallest to the largest singular value (solid line) with the relative error in the eigenvalue (dashed line). Number of components of* $y_k = 64$; *number of components of* $P(y_k) = 4$.

five-component values are less accurate they are still satisfactory. The oscillations smooth out if more components are used, as exemplified in Fig. 5.

The last two experiments were made with the Eberlein matrix. Fig. 6 shows that four randomly selected components may yield approximate eigenvalues almost indistinguishable from the sixteen-component approximations.

The dominant eigenvalues of the Eberlein matrix are $60 \pm 20i$. An eigenvalue of this magnitude causes a scaling problem in the matrix $D = (P(y_{k+2}), P(y_{k+1}), P(y_k))$. If $\| y_k \|_2 = 1$, then $\| y_{k+2} \|_2 \doteq 60^2 + 20^2 = 5200$. The largest singular value of D will be roughly distorted by five thousand.

In general, of course, the magnitude of the dominant eigenvalue is unknown. The column vectors of D should be normalized to have unit magnitude, but then the

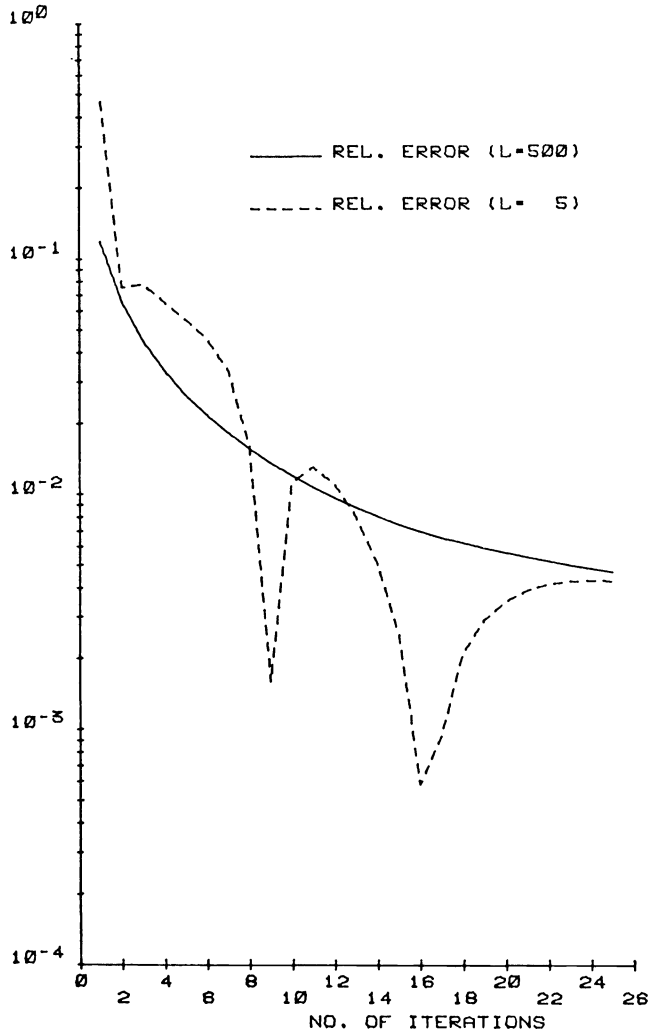


FIG 4. *The effect of a small number of components on the eigenvalue relative error for the heat flow matrix.* $n=500$, $l=5$.

eigenvalues could not be computed from the scaled matrix. To explain more precisely, let S be the diagonal matrix of scale factors such that the columns of DS are of unit magnitude. Linear independence should be determined from the singular values of DS whereas eigenvalues must be computed from the matrix of right singular vectors of D. Both computations may be performed at negligible cost, except that required to compute S, if it is assumed that $p \ll n$. For, let $D=QR$ be the QR decomposition of D, where R is $p \times p$. The singular values of DS are the same as those of RS, and the right singular vectors of D are the same as those of R. Two SVDs have to be computed, but, since $p \ll n$, the extra work is only that required to compute S, which for the 2-norm is pl multiplications. If the total work, which is now lp^2+lp multiplications, is to be the same as for current versions of the Manteuffel algorithm, then l must be 12% of n (when $p=5$).

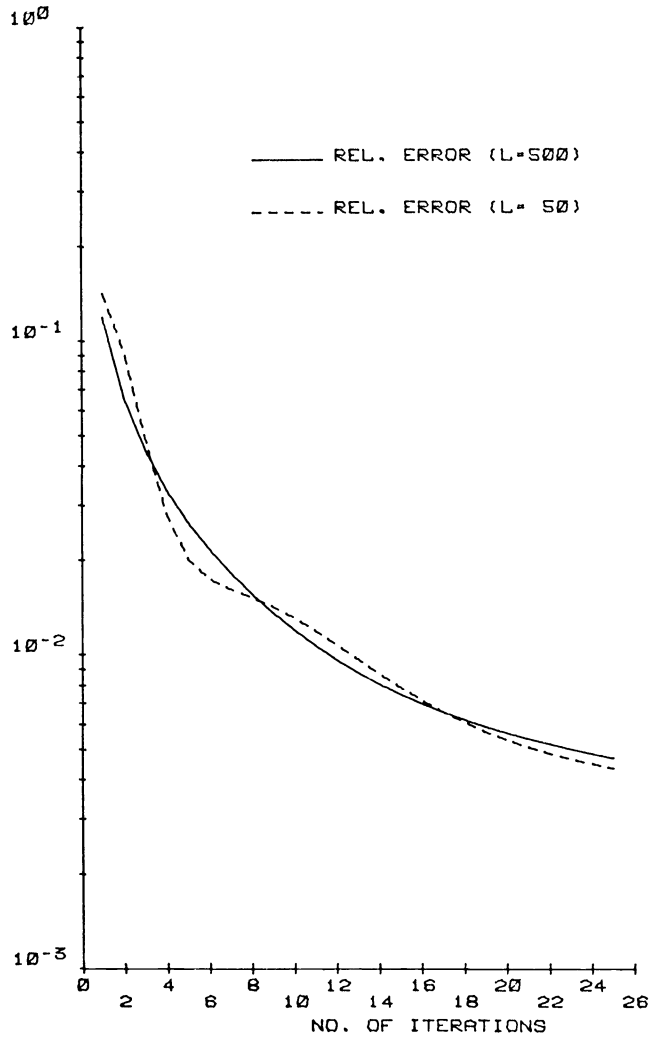


FIG 5. *Oscillations in the eigenvalue relative error are smoother for* $l=50$, $n=500$. *The heat flow matrix.*

8. Conclusion. Richardson's method employs Chebyshev acceleration parameters determined by the convex hull of eigenvalues of the residual polynomial matrix. The Manteuffel algorithm, which estimates parameters dynamically, performs two basic operations: It computes the convex hull by the power method, which requires a least squares procedure; and it computes the best parameters from the convex hull. Computation of the best parameters is inexpensive once the eigenvalues are known. The cost of the Manteuffel algorithm is in the solution of the least squares problem. Examination of a method to reduce execution time and storage requirements for the least squares problem by using fewer components of the power method iterates shows that the method works if the projected iterates, which form the columns of matrix D, are (approximately) linearly dependent. The singular values of D may be used to test linear dependence and at the same time provide the solution of the least squares problem. This summarizes the paper except for the results of numerical experiments.

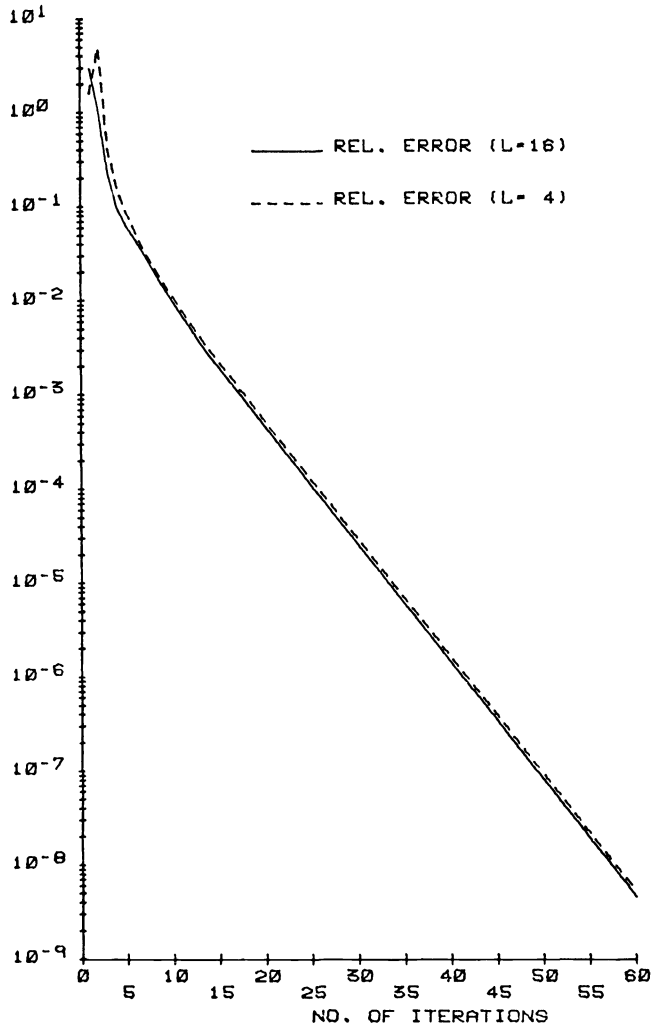


FIG 6. *The effect of fewer components on the eigenvalue relative error for the Eberlein matrix.* $n=16$, $l=4$.

If fewer compoments are used, there are fewer rows of D and the singular values are inexpensive to compute. Singular values have allowed the use of fewer compoments, but now fewer components allow the free use of singular values. Inexpensive singular values could be computed after each iteration to test whether new eigenvalues and therefore improved acceleration parameters are ready. Testing singular values displaces the old strategy of waiting a fixed number of iterations before improving acceleration parameters. The Manteuffel algorithm now falls in step with the iterative method further enhancing convergence.

Acknowledgments. I am indebted to T. Manteuffel, who made several helpful comments; to Gene Golub and the referee of another version of this paper, who suggested that the least squares problem should be unbiased; and to R. Roloff and D. Lee, who helped debug and run the computer programs.

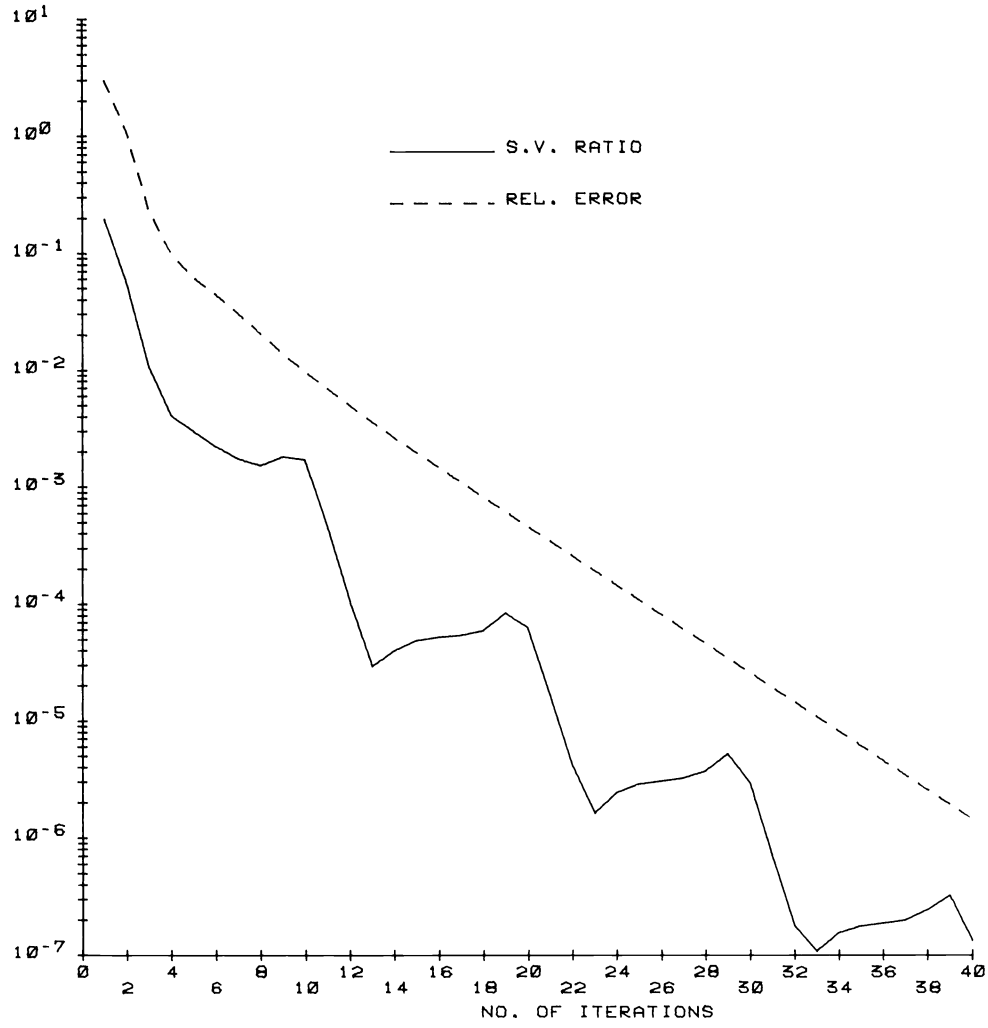


FIG 7. *Comparison of eigenvalue relative error and* σ_3/σ_1 *for the Eberlein matrix, with singular values computed from the matrix DS.* $n=16$, $l=16$.

REFERENCES

[1] P. A. BUSINGER AND G. H. GOLUB, *Algorithm* 358, *Singular value decomposition of a complex matrix*, Comm. ACM, 12 (1969), pp. 564–565.

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

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

[4] P. J. EBERLEIN, *A Jacobi-like method for the automatic computation of eigenvalues and eigenvectors of an arbitrary matrix*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 74–88.

[5] GEORGE FORSYTHE, MICHAEL A. MALCOLM AND CLEVE B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[6] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudoinverse of a matrix*, SIAM J. Numer. Anal., Ser. B, 2 (1965), pp. 205–224.

[7] G. H. GOLUB AND C. REINSCH, *Singular value decompositions and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

[8] L. W. HAGEMAN, *The Estimation of Acceleration Parameters for the Chebyshev Polynomial and the Successive Overrelaxation Iteration Methods*, Memorandum prepared for USAEC by Westinghouse Electric Corporation, June, 1972, available from National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield VA 22151.

[9] G. KJELLBERG, *On the convergence of successive over-relaxation applied to a class of linear systems of equations with complex eigenvalues*, Ericsson Technics, 2 (1958), pp. 245–258.

[10] T. A. MANTEUFFEL, *An Iterative Method for Solving Nonsymmetric Linear Systems with Dynamic Estimation of Parameters*, Rep. UIUCDCS-R-75-758, University of Illinois, Urbana, IL, Oct., 1975.

[11] T. A. MANTEUFFEL, *The Tchebyshev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.

[12] T. A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebyshev iteration*, Numer. Math., 31 (1978), pp. 183–208.

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

[14] EDUARD L. STIEFEL, *Kernel polynomials in linear algebra and their numerical application*, Nat. Bur. Stand. Appl. Math. Series, Vol. 49, pp. 1–22.

[15] H. L. STONE, *Iterative solution of implicit approximations of multi-dimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530–558.

[16] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[17] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[18] H. E. WRIGLEY, *Accelerating the Jacobi method for solving simultaneous equations by Chebyshev extrapolation when the eigenvalues of the iteration matrix are complex*, Comput. J., 5 (1963), p. 169.

[19] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0007 $1.00/0

TETRAHEDRAL FINITE ELEMENTS FOR INTERPOLATION*

OSCAR BUNEMAN†

Abstract. A uniform, space-filling array of tetrahedra is described whose vertices form a body-centered cubic grid. The symmetries of this system make it, in a sense, more isotropic than alternatives. Linear finite elements over this array result in a 9-point Laplace operator. Their use for interpolation of spectral data is studied, and the best mean-square fit to any harmonic is obtained; RMS errors are calculated. Aliassing limits are determined: this occurs outside a rhombic dodecahedron in wave-number space. An economical scheme is described for using FFT's to link discrete spectra with records over a tetrahedral mesh. Finally, the logic of tetrahedral indexing and the evaluation of the weights for interpolation are merged into one compact algorithm.

Key words. finite element, tetrahedral interpolation, body-centered cubic lattice, aliassing, spectral methods, tent function

1. Introduction. The motivation for this work arose from a need to minimize table look-ups for function evaluations in three dimensions. In a parallel processor (such as the CRAY), data retrieval from memory can only be vectorized if the data are adjacent or equi-spaced; in general, the look-ups have to be done one by one in scalar mode.

Rather than evaluating functions locally from a large data base, using high-order interpolation or high-order splines, one turns to low-order interpolation, from a minimal data base. Trilinear interpolation in three dimensions (linear in x times linear in y times linear in z) requires eight data look-ups. A genuine linear interpolation in three dimensions, involving no products of coordinate variables (approximant $ax+by+cz+d$), requires only four data. One therefore aims at making up the approximant from piecewise linear finite elements, defined over a tetrahedral mesh, with continuity between adjacent tetrahedra.

This latter feature is guaranteed by pegging the linear approximants to the vertices of each tetrahedron; this process is described in texts on finite elements such as Zienkiewicz [1]. On the triangular faces between adjacent tetrahedra the approximants are both the same. However, it is important to realize that making the tabulated values at the vertices identical with those of the function to be interpolated may not be the best choice ("best" in the sense of optimal mean-square fit). Fig. 1 illustrates the case of two-dimensional triangular interpolation; in the case of the convex function shown, it obviously pays to raise the vertex data a little above the function values. Section 6 below is devoted to the problem of determining the best fit when the spectral composition of the function is available.

Finding the tetrahedral finite elements which best interpolate a given function is, obviously, a typical Galerkin type problem. Having decided on a particular choice of finite elements, one can, however, explore and exploit their use for other variational problems, such as those associated with partial differential equations. In §4 we consider the Laplace problem as an example.

The choice of suitable tetrahedra is the first concern of this paper. It is not a trivial one. We do not assume any prior knowledge of the features of the function to

*Received by the editors August 22, 1979, and in final form March 18, 1980.
†Institute for Plasma Research, Stanford University, Stanford, California. This work was supported by the U.S. Department of Energy.

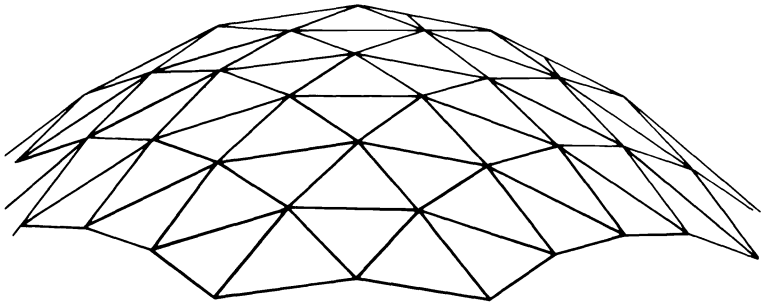


FIG 1. *A convex function of* 2 *variables approximated by triangular finite elements.*

be interpolated (i.e., where it varies slowly or rapidly), and so our set of tetrahedra should be uniform in space. Also, we aim at "isotropy" in the sense that any system of axes which serves to define the tetrahedra should possess a high degree of angular symmetry. To make clear what is meant, consider again the two-dimensional analogue. Cartesian axes, and any interpolation scheme based on them, will have 4-fold angular symmetry. A mesh of equilateral triangles, on the other hand (Fig. 11), has 6-fold symmetry, and it could be considered "more isotropic." A choice of triangles as shown in Fig. 2, however, has only 2-fold angular symmetry: all the diagonals run between NE and SW, and none between NW and SE. (Surprisingly, when these latter triangles are used as finite elements for the Laplace operator, one is led to the familiar 5-point operator which has 4-fold symmetry; see Pólya [2]).

The uninitiated will want to take a cue from the two-dimensional situation, and suggest a mesh of regular tetrahedra for interpolation. Unfortunately, there exists no close packing of regular tetrahedra, as crystallographers well know (see, for instance, [3]). The three-dimensional analogue of what is shown in Fig. 2 is undesirable because of its lack of symmetry, although here, again, the Laplace operator becomes the familiar 7-point operator which has the same symmetries as the Cartesian axes.

After experimentation with a variety of other configurations, the author eventually homed in on a tetrahedral mesh which is embedded into a body-centered cubic mesh as indicated in Fig. 3. The full array of tetrahedra is generated by drawing the lines for a regular Cartesian cubic mesh, then drawing the lines for the parallel mesh of the cube centers, and, finally, drawing all the space diagonals (these are common to the cubic meshes).

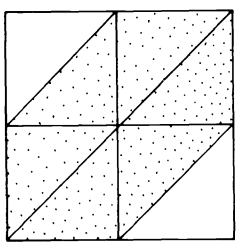


FIG 2. *Triangular linear finite elements leading to the* 5*-point Laplace operator. Dotted area: domain of influence of single entry.*

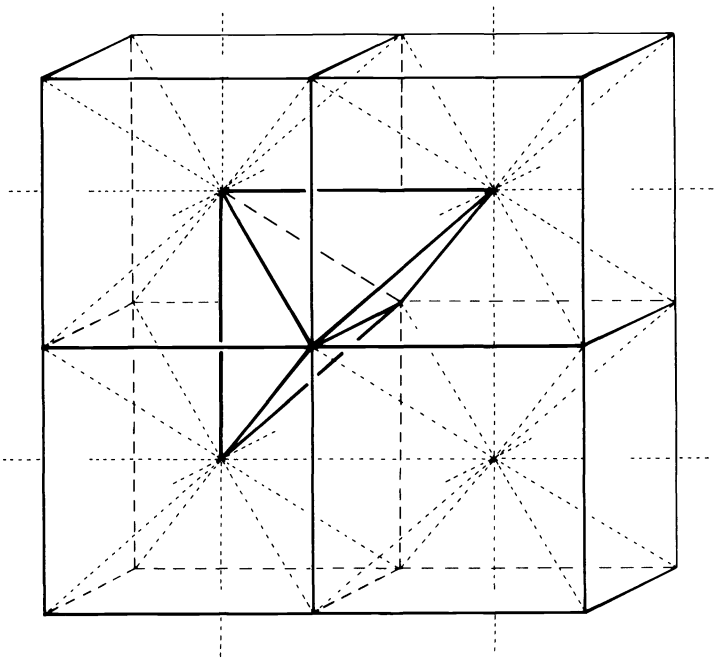


FIG 3. *Tetrahedral mesh connecting cube centers and cube corners, with two of the tetrahedra emphasized.*

2. Some features of the tetrahedral mesh. Taking the side of the original Cartesian cubic mesh as two units of length, the tetrahedra have two edges of length 2 and four of length $\sqrt{3}$. Their faces are isosceles triangles set at right angles to each other in pairs. The two long edges are parallel to two of the Cartesian axes, the third Cartesian axis is parallel to the line connecting the centers of these two long edges. The short edges run parallel to the four space diagonals of the Cartesian mesh.

There are 6 significantly different orientations of these tetrahedra. The prototype is shown in Fig. 4. The long edges are parallel to x and z, and one proceeds in the $+y$ direction from the midpoint of the side parallel to x, to the midpoint of the side parallel to z. This orientation, then, we shall define as the "x-y-z" orientation. The tetrahedra emphasized in Fig. 3 are, accordingly, in the "z-x-y" and "y-z-x" orientations, the latter meaning that proceeding in the positive z-direction takes one from the middle of the long edge parallel to y to the middle of the long edge parallel to x. The six permutations of x, y, z provide the six distinct orientations. Isotropy is assured by the fact that each orientation occurs with equal frequency.

At each meshpoint, 24 tetrahedra meet, 4 of them in each of the 6 orientations. There is no discrimination between the lattice points of what we introduced, for purposes of explanation, as the "original" cubic lattice, and the lattice of cube centers. They are like the Cs and Cl atoms in a cesium chloride crystal (see Fig. 5b), but it doesn't matter which is which. In two dimensions, Birdsall et al. [4] have found it beneficial to introduce information at the centers of a square mesh, and thus to create two "interlaced" square meshes. They then use bilinear square interpolation over each of these, and form the arithmetic mean of the results.

3. Domain of influence: tent functions; random walks. An important feature of any finite-element mesh is the "domain of influence" of a single entry. Suppose one

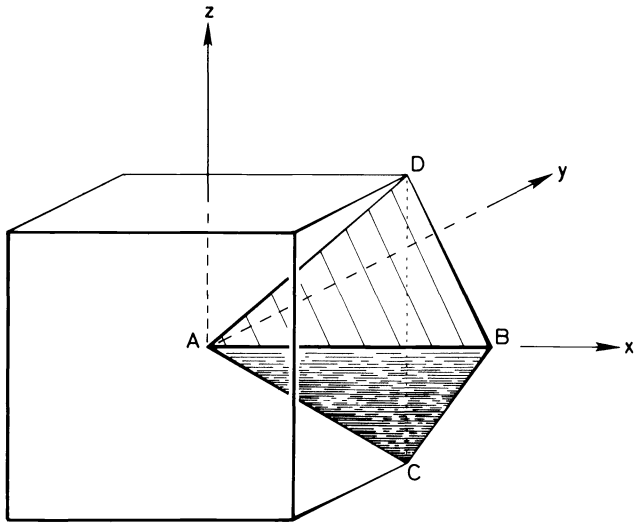


FIG 4. *Prototype tetrahedron in orientation "x-y-z". $AB=CD=2$ units, $AC=CB=BD=DA=\sqrt{3}$ units.*

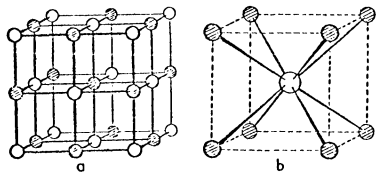


FIG 5. *The crystal structure of* (a) *sodium chloride*, (b) *cesuim chloride. From A. F. Wells*, [3].

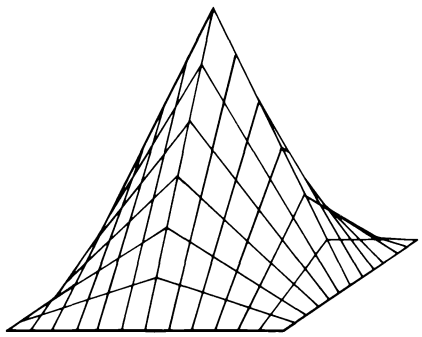


FIG 6. *Bilinear interpolation in* 2 *dimensions: interpolant due to a single nonvanishing entry.*

has a table of data in which all entries vanish except one. Interpolation will then result in zero values over most of space. Only the tetrahedra which share the point of a nonvanishing entry will contain nonvanishing interpolated values. Going back into two dimensions, one can illustrate the interpolating function over a plane; for instance, for standard bilinear interpolation over Cartesian square mesh (linear in x times linear in y) one gets what is shown in Fig. 6 (note the curvature in the surface generated from the products). For the triangular mesh shown in Fig. 2, the interpolant looks like a teepee with (nonregular) hexagonal base. For an equilateral triangular mesh, one gets a teepee with a base in the form of a regular hexagon. Fig. 11 illustrates such a "tent" function over a hexagonal base, together with contour lines which are nested hexagons. In three dimensions, we can only indicate the shape of the base to which nonvanishing interpolated values extend. This base surrounds the point of the nonvanishing entry. For the generalization of the mesh of Fig. 2 into three dimensions, one finds a domain which is strongly elongated along one of the space diagonals, shown in Fig. 7. It was this grossly anisotropic domain which made the author reject the associated tetrahedral mesh.

The new tetrahedra lead to a more isotropic domain of influence, namely the region covered by the 24 tetrahedra that meet at one point. The solid formed by these is a "rhombic dodecahedron." Fig. 8 shows this interesting solid, and how it envelops the cube which surrounds the point of nonzero value. Fig. 9 shows another view, and how the dodecahedron can be generated by translating a parallelepiped along its own diagonal. Obviously, we are much closer to a "sphere" of influence here than with the domain shown in Fig. 7. Also, there would seem to be an improvement in isotropy over the cube, which is the domain of influence for conventional "tri-linear" interpolation over a Cartesian mesh. In this very loose sense, then, we have progressed toward isotropy. (The rhombic dodecahedron will be encountered again in the section on aliassing).

Inside the domain of influence one has nonvanishing function values distributed as follows. The 24 linear finite elements generated by a single nonvanishing entry make up a continuous piecewise linear function of x, y, z, which can be visualized in terms of a nested set of rhombic dodecahedra, centered on the vertex where the

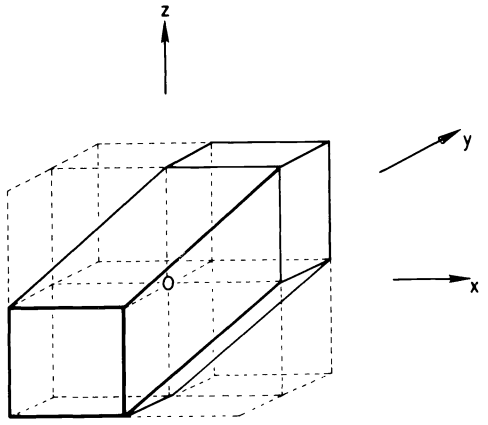


FIG 7. *Domain of influence for tetrahedral elements aligned along the + + + diagonal.*

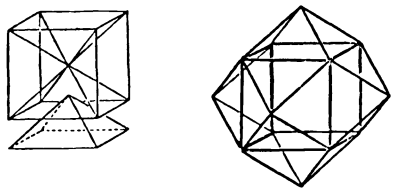


FIG 8. *Domain of influence for tetrahedral mesh shown in Fig.* 3: *rhombic dodecahedron, to be constructed from two cubes. From A. F. Wells* [3].

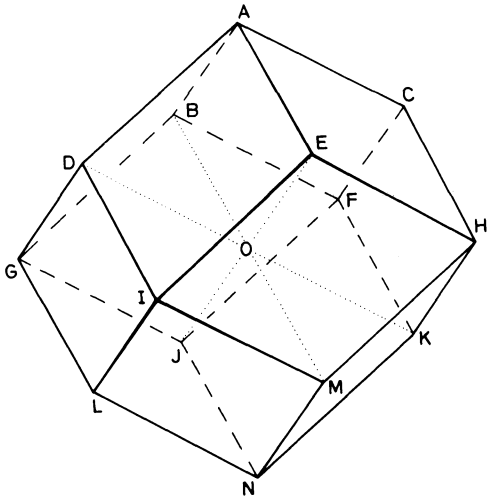


FIG 9. *Rhombic dodecahedron: construction from parallelepiped with sides along three body-diagonals OM, OJ, OD, translated along fourth body diagonal OC.*

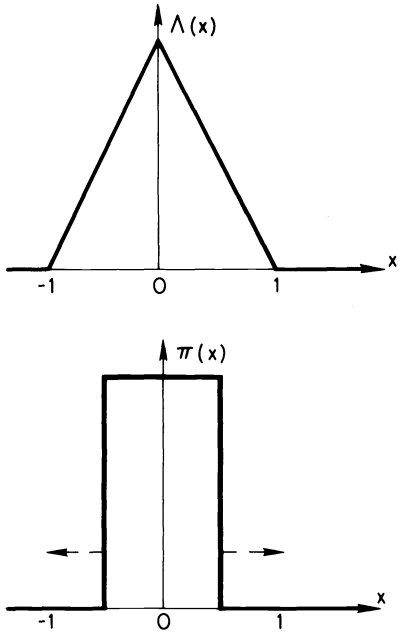


FIG 10. *Top-hat and triangle functions.*

nonvanishing entry occurs. On each of these nested surfaces the function adopts a constant value. From surface to surface the values rise linearly inwards toward the center, starting with zero on the outermost rhombic dodecahedron. We shall refer to this function as a "tent function," as a generalization of the case of a two-dimensional case where the corresponding function, plotted over its hexagonal base, gives the appearance of a tent; see Fig. 11.

The general trial function to be used in our variational (Galerkin) procedures is a superposition of such tent functions with different central vertices, scaled according to the function values at these vertices. Tent functions can be generated by convolution of π-functions (the "top-hat" function $\pi(x)$ is defined by $\pi(x)=0$ for $|x|>\frac{1}{2}$, $\pi(x)=1$ for $|x|<\frac{1}{2}$, $\pi(x)=\frac{1}{2}$ for $|x|=\frac{1}{2}$; see Fig. 10). Over a one-dimensional base the tent function becomes the triangle function $\Lambda(x)$, illustrated in Fig. 10, which is the convolution of $\pi(x)$ with itself. $\Lambda(x)$ can also be looked upon as the probability of arrival at x after a random walk of maximum length $\frac{1}{2}$ to either side (this produces the π function), followed by another such random walk.

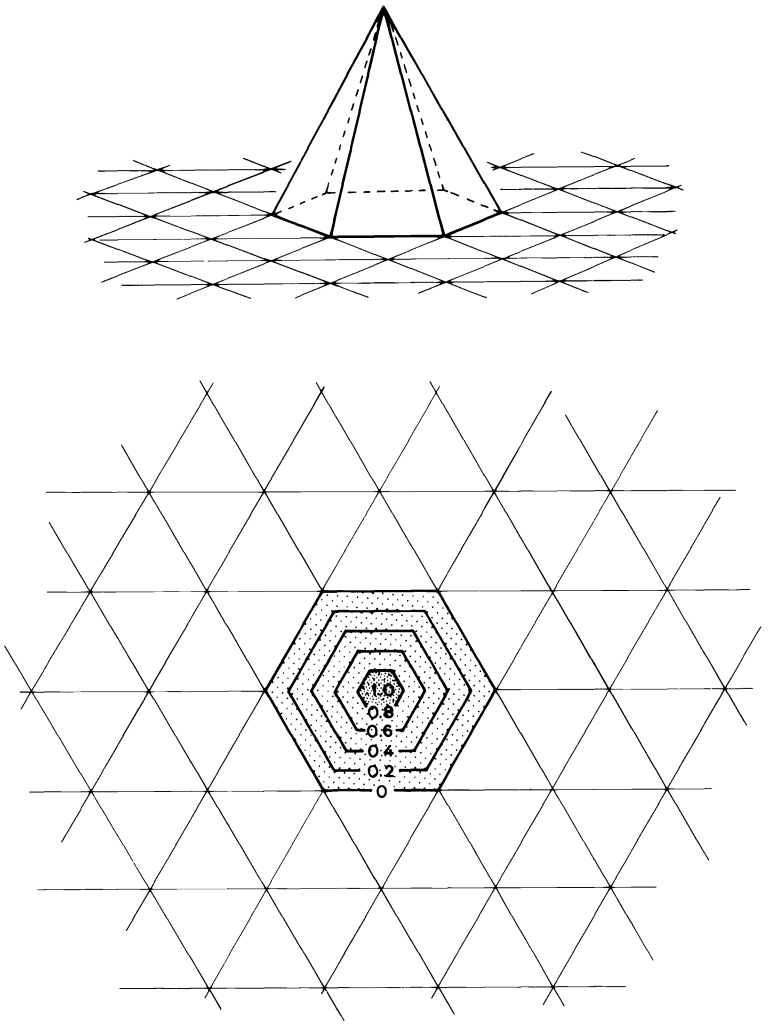


FIG 11. *Ground view and elevation of tent function over two dimensions.*

Over two dimensions one can obtain the symmetrical tent function with hexagonal base by performing three random walks, one in the 60° direction (along JOE in Fig. 12), another along −60° (DOK in Fig. 12), and the third along 0° (GOH in Fig. 12). After the first two walks, the probability of arrival is uniform inside the rhombus BGMH; after the third, when the rhombus has been shifted by arbitrary amounts (less than $\frac{1}{2}$) to the right and to the left, the probability of arrival is given by the tent function of Fig. 11. It is easy to check this, first, along the x-axis (where one generates a triangle function as in the one-dimensional case), and then along lines parallel to the x-axis, where one generates trapezoidal functions of x. A single random walk along

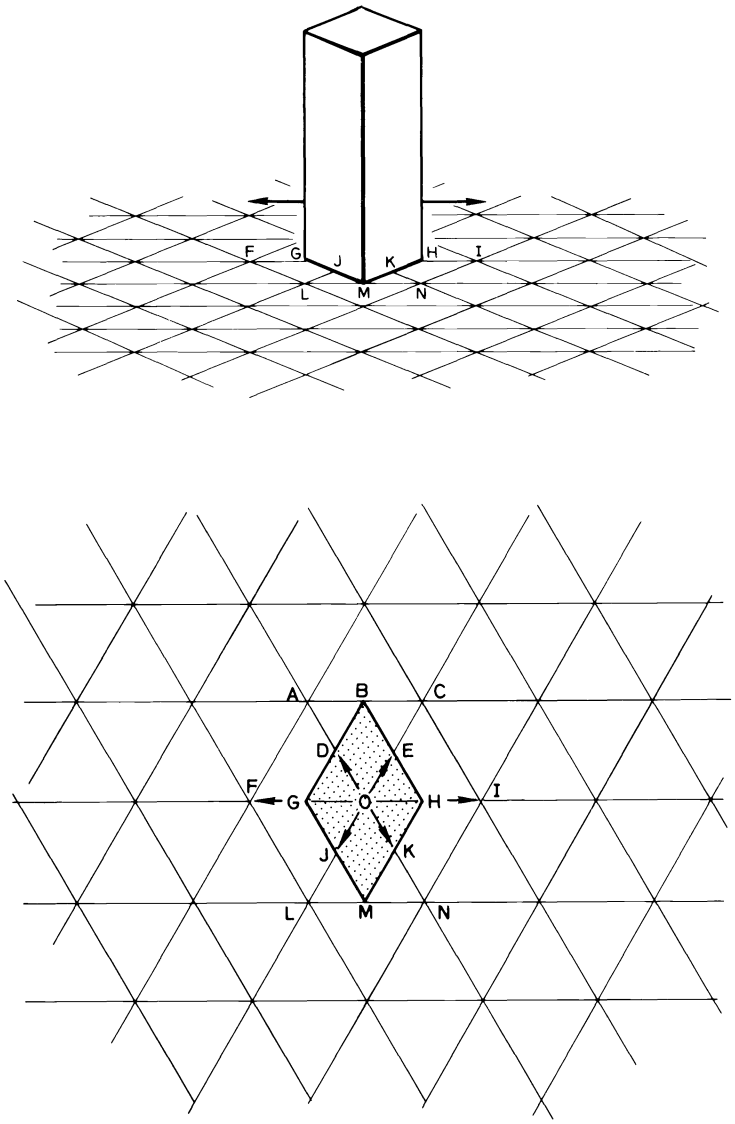


FIG 12. *Rhombic top-hat function due to convolution of two one-dimensional top-hat functions, prior to third convolution for generating tent function of Fig.* 11. *Ground view and elevation.*

the θ-direction or opposite is described by the expression $\pi(x\cos\theta+y\sin\theta)\delta(y\cos\theta-x\sin\theta)$; this is a function which, for $\theta=60°$, is unity along the stretch JOE in Fig. 12 and zero elsewhere. By convolving three such functions, with $\theta=60°$, $-60°$ and $180°$, one can obtain the tent function of Fig. 11 explicitly.

Proceeding now to three dimensions, one expects to be able to generate a symmetrical piecewise linear tent function by four random walks in four symmetrically placed directions. The four body-diagonals suggest themselves. After three such random walks, along say, JOE, DOK and BOM as shown by dotted lines in Fig. 9, one will have filled a parallelepiped uniformly with points of arrival. The parallelepiped is shaped like ODIMNLGJO in Fig. 9, or like CAEHKOBFC, but it lies midway between these two and envelops the origin O. The remaining random walk, along LOC, will then move this parallelepiped between the two extremes shown and fill up the entire rhombic dodecahedron. This introduces the linear variation of frequency of points of arrival; for instance, it generates a simple triangle function along LOC, with the crest at O. The linearity, plus the fact that the function comes to zero on the faces of the rhombic dodecahedron, guarantees that our original tent is thus generated.

A single one of the four random walks, for instance that along the $+++$ diagonal, results in a probability of arrival $(1/\sqrt{3})\pi(x'/\sqrt{3})\delta(y')\delta(z')$, where x', y', z' are orthonormal coordinates with x' along the $+++$ direction, for instance

$$x'=\frac{x+y+z}{\sqrt{3}}, \qquad y'=\frac{x-y}{\sqrt{2}}, \qquad z'=\frac{2z-x-y}{\sqrt{6}}.$$

We have chosen $(1/\sqrt{3})\pi(x'/\sqrt{3})$ rather than $\pi(x')$ so that after all four random walks the distance OC, for instance, comes out as $\sqrt{3}$ in conformity with the scales chosen for Fig. 4. The convolution of four such $\pi\delta\delta$-functions, each with x', y', z' as given, but with different signs in front of y and z, provides a formal expression for our tent function.

The resulting expression, after carrying out the convolutions, is complicated and not very helpful. However, we must deal with the normalization of the tent function so constructed. Its total, over the entire domain of influence, will be unity, since this tent function is identified as a probability of arrival anywhere. The volume of the domain is 16 units (the side of the cube in Fig. 8 is 2 units). In each of the tetrahedra that make up this domain, the mean value of the function is $\frac{1}{4}$ of the central vertex value; this is fairly obvious, but see §5 for confirmation. We conclude that the tent function obtained from the random walks must have $\frac{1}{4}$ as its central vertex value.

Convolutions are most easily performed in Fourier space. The Fourier transform of $\pi(x)$ is given by

$$\int_{-\infty}^{+\infty}\pi(x)e^{ikx}\,dx=\frac{2}{k}\sin\frac{k}{2}.$$

In three dimensions we use the exponential $e^{i\vec{k}\cdot\vec{r}}=e^{i(lx+my+nz)}$ for transforming. To transform the function $(1/\sqrt{3})\pi(x'/\sqrt{3})\delta(y')\delta(z')$, one introduces rotated coordinates λ, μ, ν in l, m, n-space: $\lambda=(l+m+n)/\sqrt{3}$, $\mu=(l-m)/\sqrt{2}$, $\nu=(2\nu-\lambda-\mu)/\sqrt{6}$, so that $lx+my+nz=\lambda x'+\mu y'+\nu z'$. The two delta functions transform to

unit factors, and $(1/\sqrt{3})\pi(x'/\sqrt{3})$ transforms to

$$\frac{2}{\lambda\sqrt{3}}\sin\frac{\lambda\sqrt{3}}{2}=\frac{2}{l+m+n}\sin\frac{l+m+n}{2}.$$

One concludes that the Fourier transform of the tent function, to be denoted N, is the product of four such functions with different sign combinations in front of l, m, and n:

$$N=\left(\frac{2}{m+n+l}\sin\frac{m+n+l}{2}\right)\left(\frac{2}{m+n-l}\sin\frac{m+n-l}{2}\right)$$

$$\times\left(\frac{2}{n+l-m}\sin\frac{n+l-m}{2}\right)\left(\frac{2}{l+m-n}\sin\frac{l+m-n}{2}\right).$$

4. The Laplace operator. The differential equation div grad $f=0$ is associated with a variational (Galerkin) problem: $L(f)\equiv\iiint\frac{1}{2}(\text{grad } f)^2\,dx\,dy\,dz=$ extremum. The gradient components of the linear function $f(x, y, z)$ which passes through the corner values f_A, f_B, f_C, f_D of the prototype tetrahedron shown in Fig. 4 are found by inspection:

$$\frac{\partial f}{\partial x}=\frac{f_B-f_A}{2},\qquad\frac{\partial f}{\partial y}=\frac{f_C+f_D}{2}-\frac{f_A+f_B}{2},\qquad\frac{\partial f}{\partial z}=\frac{f_D-f_C}{2}.$$

(The gradient is uniform inside the tetrahedron, and hence its x- and z- components can be read off the two long edges; the y-component is obtained by proceeding along the line connecting their midpoints). The integrand is the constant

$$\frac{f_A^2+f_B^2+f_C^2+f_D^2-(f_A+f_B)(f_C+f_D)}{4},$$

and the integration merely introduces the volume of the tetrahedron, $\frac{2}{3}$, as a factor, turning "4" into a "6."

Consider now the variation of f_A. The contribution of our prototype tetrahedron to $\partial L/\partial f_A$ is $\frac{1}{3}f_A-\frac{1}{6}(f_C+f_D)$. Note that C and D are corners of the cube which surrounds A. Adding the contributions from all the 24 tetrahedra which contain A, we see that C and D, like all the 8 corners of this cube, each occur just 6 times; hence the variational procedure yields:

$$8f_{\text{center}}-\Sigma f_{\text{corners}}=0,$$

a very plausible 9-point formula for connecting the vertex values. This applies at every meshpoint, irrespective of whether it is a cube center in the mesh used for this derivation, or whether it is a cube center in the complementary mesh (here characterized as the mesh of corners). We note a very slight improvement in isotropy over the 7-point formula: eight, instead of six symmetrically placed nearest neighbors are referred to.

5. Linear tetrahedral finite elements: their mean and their mean-square deviation. The use of linear finite elements as optimal interpolants ("optimal" in the sense of least mean-square error) will require more detailed information on the finite elements than needed for the Laplace application. Specifically, we shall need to know the mean value of the linear function through the four vertex data as well as its mean square.

One would guess that the mean of the linear function is the mean of the four vertex values. There are various ways of convincing oneself of this fact; it will result easily from the systematic calculation which follows.

It is not obvious that the mean-square deviation of the linear function within the tetrahedron is one-fifth of the mean-square deviation of the vertex values. To prove this, we perform a rigorous integration. In fact, we prove this for general tetrahedral elements, not only for the specific elements described above which possess considerable symmetry.

The two statements regarding the mean and the mean-square deviation need only be proved for the case where one of the vertex values is zero, since the addition of a uniform constant to all vertex values changes the levels of all the means by the same constant and the mean-square deviation not at all. Moreover, a common multiplier of the four vertex values will manifest itself as the same multiplier of the linear function, and hence we can normalize our linear function conveniently. Lastly, we can introduce coordinates x, y, z for the purpose of integration which may differ from those in routine use with these tetrahedral finite elements. We choose one of the axes, z, so that it points along the gradient of the finite element, and we place the origin at the vertex where a zero value is prescribed. With suitable normalization, the finite-element linear function then becomes just z itself.

We have to calculate three moments for the tetrahedron, the volume, $\iiint dx\,dy\,dz$, the first moment, $\iiint z\,dx\,dy\,dz$, and the second moment, $\iiint z^2\,dx\,dy\,dz$, in order to deduce the mean and the mean square. Fig. 13 shows the general tetrahedron ABCO for which these integrations are to be performed, and two

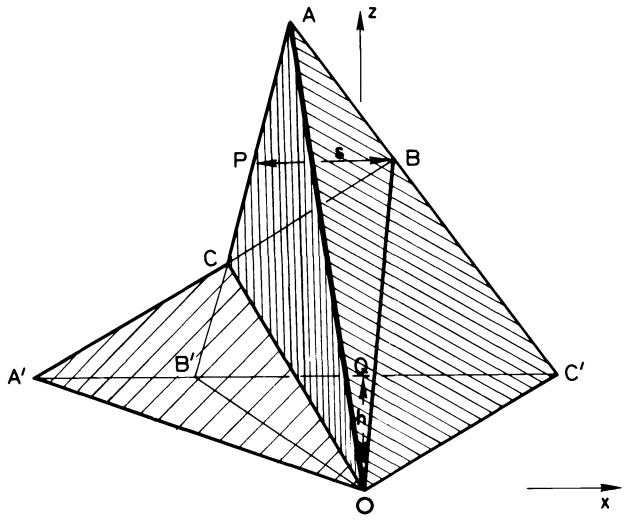


FIG 13. *General tetrahedron ABCO made up as AB'C'O plus A'B'CO minus A'BC'O. P at same y and z as B, Q on y-axis.*

projections of it into the plane $z=0$, one from A (base area $OB'C'$), the other through C (base area $OA'B'$). We orient the x-axis to run parallel to the intersection $A'B'C'$ of the ABC-face extension with the plane $z=0$.

The heights of A, B, and C, which are also the vertex values, will be denoted as

$$z(A)=a, \qquad z(B)=b, \qquad z(C)=c.$$

Further, we introduce

$$y(Q)=h, \qquad x(B)-x(P)=s,$$

where Q is the intercept of $B'C'$ on the y-axis, and P is the point on CA which is at the same z-level as B. By similar triangles CBP and $CA'B'$, one deduces

$$\overline{B'A'} : z = \overline{CB'} : \overline{PC} = c:(b-c),$$

and by similar triangles ABP and $AC'B'$:

$$\overline{C'B'} : s = \overline{AC'} : \overline{AB} = a:(a-b).$$

From these one finds the two base areas

$$\overline{A'B'O} = \frac{\frac{1}{2}hsc}{(b-c)} \qquad \text{and} \qquad \overline{B'C'O} = \frac{\frac{1}{2}hsa}{(a-b)}.$$

For the purpose of the integrations, we make up the tetrahedron $ABCO$ from three tetrahedra which each have one face in the plane $z=0$, namely as $AB'C'O$ plus $A'B'CO$ minus $A'BC'O$. In $AB'C'O$ the integration from 0 to a has to be done with a horizontal cross section which varies quadratically with the distance from A, and which is therefore $\overline{B'C'O}(1-z/a)^2$. The contribution from this tetrahedron to the nth moment is then

$$\overline{B'C'O}\int_0^a z^n(1-z/a)^2\,dz = \frac{2\,\overline{B'C'O}\,a^{n+1}}{(n+1)(n+2)(n+3)}.$$

Similarly, $A'B'CO$ contributes the amount

$$\frac{2\,\overline{A'B'O}\,c^{n+1}}{(n+1)(n+2)(n+3)},$$

while the contribution from $A'BC'O$,

$$\frac{2(\overline{A'B'O}+\overline{B'C'O})b^{n+1}}{(n+1)(n+2)(n+3)},$$

must be subtracted. Overall we get

$$\frac{2}{(n+1)(n+2)(n+3)}\left(\overline{B'C'O}\,(a^{n+1}-b^{n+1})+\overline{A'B'O}\,(c^{n+1}-b^{n+1})\right)$$

$$=\frac{hs}{(n+1)(n+2)(n+3)}\left(\frac{a^{n+1}-b^{n+1}}{a-b}a-\frac{c^{n+1}-b^{n+1}}{c-b}c\right),$$

on substitution of the previously calculated areas.

Taking now the three moments one by one we obtain, for $n=0$,

$$\text{volume} = hs(a-c)/6,$$

and for $n=1$,

$$\begin{aligned}\text{first moment} &= hs((a+b)a-(c+b)c)/24\\ &= hs(a+b+c)(a-c)/24\\ &= \text{volume}\cdot(a+b+c)/4,\end{aligned}$$

in confirmation of the statement,

$$\text{mean of function}, \langle z\rangle, = \text{mean of vertex values}, \langle z_{\text{vertex}}\rangle,$$

remembering that the fourth vertex value is zero. Finally, for $n=2$,

$$\begin{aligned}\text{second moment} &= hs\big((a^2+ab+b^2)a-(c^2+cb+c^2)c\big)/60\\ &= hs(a^2+b^2+c^2+bc+ca+ab)(a-c)/60\\ &= \text{volume}\cdot\big((a+b+c)^2+a^2+b^2+c^2\big)/20\\ &= \text{volume}\cdot\left(\frac{4}{5}\left(\frac{a+b+c}{4}\right)^2+\frac{a^2+b^2+c^2}{4\cdot 5}\right).\end{aligned}$$

Dividing by the volume, and subtracting the square of the mean, $((a+b+c)/4)^2$, on both sides, we deduce

$$\langle (z-\langle z\rangle)^2\rangle = \tfrac{1}{5}\langle (z_{\text{vertex}}-\langle z_{\text{vertex}}\rangle)^2\rangle.$$

Since z, as explained, is representative of any linear function in these calculations, we may replace it by f where appropriate. For instance, we can state

$$\langle f(x,y,z)\rangle = \langle f_{\text{vertex}}\rangle,$$

$$\langle f^2(x,y,z)\rangle = \tfrac{4}{5}\langle f_{\text{vertex}}\rangle^2 + \tfrac{1}{5}\langle f^2_{\text{vertex}}\rangle,$$

as a generalization of the corresponding equations above. Moreover, by applying these equations separately to the real and imaginary parts of a complex function f, and adding, one deduces:

$$\langle |f(x,y,z)|^2\rangle = \tfrac{4}{5}|\langle f_{\text{vertex}}\rangle|^2 + \tfrac{1}{5}\langle |f_{\text{vertex}}|^2\rangle.$$

6. Tetrahedral finite elements as interpolants. The next variational problem to be attacked has multiple uses. We determine, for an arbitrary harmonic function $F = e^{i(lx+my+nz)}$, what are the finite elements which interpolate this function with least mean-square error. Not only is this an instructive exercise per se in the use of finite elements, but from it we can get some quantitative results on the performance of our finite elements (by studying the residual error). Most important, perhaps, is the fact that field data are, in many applications, conveniently kept or supplied in spectral form (we are thinking here of flow fields, gravitational fields, electric and magnetic

fields, density distributions etc.); they have to be evaluated locally from their spectral record by fast Fourier transformation onto a mesh and subsequent interpolation.

There are, of course, situations where certain specific types of error are more damaging than others to the final result of a calculation which involves interpolations. For instance, in plasma simulations, the harmonic content of the error (the "aliasses" generated from the fundamental harmonic which is being interpolated) is rather important—some aliasses have worse effects than others (see Langdon [5] and Eastwood [6]). However, without knowing more about the nature of the application, let us give all error harmonics (aliasses) equal weight and minimize the sum of the squares of their amplitudes; by virtue of the orthonormality of Fourier harmonics, this is exactly equivalent to minimizing the mean-square error. Particularly obnoxious aliasses can always be de-emphasized further by introducing a filter into the physics which is handled in the spectral domain. Section 8 will be devoted to the topic of aliasses, and we shall at least answer the question where in wave-number space the aliasses are to be found when one uses tetrahedral finite elements.

The best-fit tetrahedral interpolant to our harmonic function must satisfy the variational problem

$$\iiint |e^{i(lx+my+uz)}-f(x, y, z)|^2\, dx\, dy\, dz = \text{minimum},$$

the integration now being over a very large spatial domain. In the most common applications, typically those involving Fast Fourier Transforms, l, m, and n are rational multiples of π and a large enough periodicity box can be found, with sides which are multiples of $2\pi/l$, $2\pi/m$, $2\pi/n$, such that the box contains an integral number of cells. The integration is then understood to be over this box. The finite elements which make up the piecewise linear function $f(x, y, z)$ are defined by the vertex values f_A, f_B etc., and so our task is to determine the "optimal" set of such vertex values which satisfy the above condition. In the introduction we indicated that the function values themselves, $\exp(ilx_A+imy_A+imz_A)$, $\exp(ilx_B+imy_B+imz_B)$ etc. are not necessarily the best choice.

However, without going through the details of the calculation, we can say that the phases of the vertex values must be the same as those of the function values: if the phases were not the same, function and approximant would get out of step sooner or later and differ by large amounts. All that is necessary, therefore, is to determine the real constant multiplier λ by which the function values have to be adjusted to give optimal vertex values.

Since harmonic functions always bend towards the coordinate axis, one expects the linear approximant to have to be raised somewhat from a piecewise linear function connecting function values. One wants the approximant to be somewhere between the "chord" and the "tangent". We therefore anticipate that $\lambda > 1$. Of course, the actual value depends on the harmonic numbers l, m, n. If $f_0(x, y, z)$ is the piecewise linear function through the vertex values, so that $f=\lambda f_0$, the variational condition above results in

$$\lambda = \frac{\iiint f_0^* e^{i(lx+my+nz)}\, dx\, dy\, dz}{\iiint |f_0|^2\, dx\, dy\, dz}.$$

The integrations should be over all the tetrahedra within the large box described above. Let us divide by the volume of this box both in the numerator and in the denominator. The volume of the box is 4 times the number of vertices in it (since the volume associated with each vertex is half that of the basic cube). In the numerator, f_0 can be treated as a superposition of tent functions with vertex values $\exp i(lx_v + my_v + nz_v)$, where x_v, y_v, z_v are the vertex coordinates. These are also the displacements of the vertices, so that each vertex contributes the same amount to the integral in the numerator, and after division by the volume, the contribution is just that of a single vertex at the origin with function-value $\frac{1}{4}$. Thus the numerator (after division by the large volume), is exactly the Fourier transform N calculated at the end of § 3.

In the denominator, after division by the volume, we have the mean square $\langle |f_0|^2 \rangle$ for all the tetrahedra. Any two tetrahedra in the same orientation contain the same distribution of $|f_0|$, since f_0 differs only by a complex phase factor between the two. Hence we only need to calculate $\langle |f_0|^2 \rangle$ once for each orientation, then average over the six orientations. For the prototype, oriented as shown in Fig. 4, one finds

$$\langle f_0 \rangle = \tfrac{1}{2} e^{i(l+\frac{1}{2}m)}(\cos l + \cos n)\cos \tfrac{1}{2}m + i(\cos n - \cos l)\sin \tfrac{1}{2}m,$$

by averaging the vertex values (each of which has unit absolute value). The formula at the end of §5 then yields

$$\langle |f_0|^2 \rangle = \frac{\cos^2 l + \cos^2 n + 2\cos l \cos m \cos n + 1}{5}$$

for this prototype in the "x-y-z" orientation. Averaging over the six orientations, i.e. over the permutations of l, m, n, then results in a denominator given by

$$D = \tfrac{1}{5} + \tfrac{2}{15}(\cos^2 l + \cos^2 m + \cos^2 n) + \tfrac{2}{5}\cos l \cos m \cos n.$$

By way of the trigonometric identity $\cos\beta - \cos\alpha = 2\sin((\alpha+\beta)/2)\sin((\alpha-\beta)/2)$, one can express D in terms of the angles $(\pm l \pm m \pm n)/2$, which are used in N:

$$D = 1 - \tfrac{1}{3}\left(\sin^2 \frac{l+m+n}{2} + \sin^2 \frac{m+n-l}{2} + \sin^2 \frac{n+l-m}{2} + \sin^2 \frac{l+m-n}{2}\right)$$

$$- \tfrac{8}{15} \sin \frac{l+m+n}{2} \sin \frac{m+n-l}{2} \sin \frac{n+l-m}{2} \sin \frac{l+m-n}{2}.$$

For small wave numbers, one now readily checks that $\lambda = N/D \approx 1 + (l^2+m^2+n^2)/6$, which exceeds unity, as expected.

These explicit formulas for λ provide the solution to the problem of finding the best-fit tetrahedral linear finite elements for the interpolation of harmonic functions. The main use of these λ will be in the interpolation of spectral data. If the λ-boosts are applied separately to each harmonic in a composite spectrum and the resulting spectrum is then FFT'ed onto our cubic-centered mesh, one generates exactly the table of corner values for our tetrahedral splines which guarantee optimal (least mean-square error) fit to the nonharmonic function represented by the composite spectrum.

7. The mean-square error. Approximating harmonic functions by tetrahedral linear finite elements allows one to measure the performance of these elements in a precise manner. It is intuitively clear that linear approximation will give "good" results for "smooth" functions and "poor" results for "rapidly varying" functions.

The epithets "smooth" and "rapidly varying" are made precise by specifying harmonic content; the harmonic number gives a numerical measure of smoothness. The epithets "good" and "poor," on the other hand, are made precise by the calculation of the mean-square error.

In this section, we calculate the mean-square error, in the first place, as a function of the three components l, m, n. Then we examine its dependence on the harmonic number $k=\sqrt{(l^2+m^2+n^2)}$ i.e., the magnitude of the wave vector, but we also study the effects of the direction of the wave vector on the error. This will give us a precise measure of the isotropy of interpolation.

The mean-square error is

$$\frac{1}{\text{volume}} \iiint |e^{i(lx+my+nz)}-\lambda f_0(x, y, z)|^2 \, dx\, dy\, dz.$$

This can be interpreted as a "percentage error" since the integral of $|e^{i(lx+my+nz)}|^2$ itself is the volume of integration. This volume is, of course, the same as in the optimizing procedure of §4. The integrations needed for determining the error have already been performed in the process of determining the optimal λ, and one finds directly,

$$\text{Mean-Square Error} = 1 - N^2/D,$$

with N and D as given at the ends of §3 and §6. A coarse record of the MSE is given in Table 1, over l, m, and n running independently in steps of $\pi/16$. The table is restricted to the range $n \leqq m \leqq l$, and to positive arguments. The MSE is sign- and permutation-insensitive, so Table 1 is sufficient as regards the directions of the wave-vector. The motivation for a cutoff at $m \leqq \pi - l$ will be given in the section on aliassing. The table shows in numbers how "poorly" a "very rippled" function is interpolated: the mean-square error reaches 0.5 on a radius of approximately $3\pi/4$. The mean-square error remains below 0.09 for wave numbers k less than $\pi/2$.

In the domain of "smoother," low-k, harmonics one might be interested in a finer tabulation of the error, but it is more instructive, in this range, to use small-argument expansion of the trigonometric functions in the expressions for N and D. After some effort, one encounters the happy cancelation of quadratic terms, and one is left with a fourth-order MSE given by

$$\text{MSE} = 1 - N^2/D = \frac{3(l^4+m^4+n^4)+2(m^2n^2+n^2l^2+l^2m^2)}{360}.$$

To see how this expression depends on the magnitude of the wave number, and how sensitive it is to direction in wave-vector space, we make up the numerator from the two symmetric fourth-order functions

$$k^4=(l^2+m^2+n^2)^2 \quad \text{and} \quad P_4=l^4+m^4+n^4-3m^2n^2-3n^2l^2-3l^2n^2.$$

TABLE 1.

Mean - square errors

	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	L=11	L=12	L=13	L=14	L=15	L=16
M=0	.000	.001	.004	.009	.020	.041	.075	.128	.205	.307	.429	.559	.681	.785	.865
M=1	.000	.001	.004	.010	.022	.043	.078	.134	.213	.317	.440	.569	.690	.792	
M=2	.001	.002	.005	.012	.026	.050	.089	.150	.236	.346	.472	.600	.717		
M=3		.004	.008	.017	.034	.064	.111	.182	.278	.396	.525	.650			
M=4			.014	.027	.050	.088	.147	.232	.341	.467	.595				
M=5				.046	.077	.128	.204	.305	.426	.555					
M=6					.122	.190	.284	.400	.528						
M=7			N=0			.277	.388	.512							
M=8							.507								
M=1	.000	.001	.004	.010	.023	.045	.082	.139	.220	.326	.450	.579	.699	.799	
M=2	.001	.002	.005	.013	.027	.052	.093	.155	.243	.354	.481	.609	.725		
M=3		.004	.008	.018	.035	.065	.114	.186	.283	.403	.532	.657			
M=4			.015	.027	.051	.089	.149	.235	.346	.472	.601				
M=5				.046	.078	.129	.205	.307	.429	.559					
M=6					.123	.191	.285	.402	.531						
M=7			N=1			.278	.389	.514							
M=8							.508								
M=2	.001	.003	.006	.015	.030	.058	.103	.170	.264	.380	.509	.635	.746		
M=3		.005	.009	.019	.038	.071	.123	.199	.302	.425	.555	.678			
M=4			.016	.029	.053	.094	.157	.246	.360	.490	.619				
M=5				.048	.080	.133	.211	.316	.441	.572					
M=6					.125	.194	.290	.409	.539						
M=7			N=2			.281	.393	.519							
M=8							.513								
M=3		.007	.012	.024	.045	.082	.141	.225	.336	.464	.594	.713			
M=4			.019	.033	.060	.104	.173	.269	.389	.521	.649				
M=5				.053	.086	.142	.224	.334	.463	.595					
M=6					.132	.202	.301	.423	.555						
M=7			N=3			.290	.403	.530							
M=8							.522								
M=4			.027	.044	.074	.124	.201	.306	.434	.568	.692				
M=5				.064	.100	.160	.249	.365	.499	.632					
M=6					.147	.220	.322	.448	.582						
M=7			N=4			.307	.421	.551							
M=8							.540								
M=5				.086	.128	.195	.292	.416	.553	.682					
M=6					.176	.253	.360	.490	.624						
M=7			N=5			.340	.455	.584							
M=8							.570								
M=6					.228	.310	.422	.552	.681						
M=7			N=6			.397	.510	.634							
M=8							.617								
M=7			N=7			.481	.588	.702							
M=8							.682								
M=8			N=8				.760								

Relation between L,M,N and ℓ,m,n : $\ell = L\pi/16$ $m = M\pi/16$ $n = N\pi/16$

The significance of P_4 is that its spherical average is zero (one checks that l^4 has average $\frac{1}{5}$ on the unit sphere and $m^2 n^2$ has average $\frac{1}{15}$). P_4 is, in fact, a fourth-order spherical harmonic. The resulting representation,

$$\text{MSE} = \frac{11k^4 + 4P_4}{1800},$$

now displays the isotropic part as $11k^4/1800$, along with the anisotropic modulation of it, $P_4/450$.

Another way of measuring the MSE and its anisotropy is to select the three most interesting directions in wave-number space, and to record the MSE as a function of k along them. One finds that

along an axis (e.g. $l=k$, $m=0$, $n=0$), $\quad \mathrm{MSE}=k^4/120$,

along a face diagonal (e.g. $l=m=k/\sqrt{2}$, $n=0$), $\quad \mathrm{MSE}=k^4/180$,

along a space diagonal ($l=m=n=k/\sqrt{3}$), $\quad \mathrm{MSE}=k^4/216$,

For the *root* mean-square error, one finds $0.091k^2$, $0.075k^2$, and $0.068k^2$ along the three directions.

It is interesting to compare these results concerning the error and its isotropy with those for conventional tri-linear interpolation over a cubic mesh. Such conventional interpolation is rarely done with best-fit piecewise linear functions; the interpolant is usually pegged to function values at the meshpoints. If we followed the latter practice, the resulting mean-square error would come out second order, rather than fourth order in k. For a fair comparison, we shall therefore optimize the approximant in tri-linear interpolation over a cubic mesh in the same way as we did for the tetrahedral mesh. In some plasma simulations, this type of optimization is actually practiced; see, for instance Eastwood [6].

This task is very easy compared with what had to be done in §4 above, since the best-fit calculation can be done in each dimension separately. Taking the mesh interval as two units, and placing the x-origin in the middle of any such interval, one has to minimize

$$\int_{-1}^{+1} |e^{ilx}-\lambda(\cos l+ix\sin l)|^2\,dx,$$

and one finds

$$\lambda=N_1/D_1 \quad \text{with} \quad N_1=\left(\frac{1}{l}\sin l\right)^2,\ D_1=1-\tfrac{2}{3}\sin^2 l;$$

(see also Eastwood, [6], for this numerator and denominator). This gives a mean-square error for one dimension

$$\mathrm{MSE}_1\equiv 1-N_1^2/D_1\approx l^4/45.$$

The mean-square errors for the three dimensions are additive in this case, so that

$$\mathrm{MSE}_3=\frac{l^4+m^4+n^4}{45}=\frac{3k^4+2P_4}{225},$$

by way of the principle of decomposition into an isotropic component and its modulation as above. We observe immediately that there is less isotropy than before; the ratio between the isotropic component and the modulation is 6:4 in place of 11:4 for the tetrahedral mesh.

Before comparing the magnitudes of the mean-square errors, however, we must make an adjustment to the scale of k. The tetrahedral mesh introduces information at the cube centers into the interpolation process; the data base is twice as large as for the "conventional" tri-linear cubic interpolation. Let us, therefore, shrink the mesh size by a factor $2^{1/3}=1.26$ in the calculation for the tri-linear cubic case so that the

number of data per periodicity box (or per large volume) is the same for the two cases. This means that the scale of k is changed by the factor $2^{1/3}$, and our formula for MSE_3 has to be divided by $2^{4/3}=2.52$.

While this does not affect the comparison of the degree of isotropy we get the following comparison for the magnitudes of the isotropic parts of the MSE's, i.e., the spherically averaged MSE's.

$$\langle \mathrm{MSE} \rangle_{\mathrm{tet}} = \frac{11k^4}{1800}, \langle \mathrm{MSE}_{\mathrm{cube}}(\text{with rescaling}) \rangle = \frac{9.5k^4}{1800},$$

or, for the rms errors after spherical averaging,

$$\mathrm{RMSE}_{\mathrm{tet}} = 0.078k^2, \mathrm{RMSE}_{\mathrm{cube}}(\text{scaled}) + 0.073k^2.$$

The error for the tetrahedral mesh, while more isotropic, is slightly (7%) worse on average.

We can also compare the errors for the three important directions: along an axis, along a face diagonal, and along a space diagonal. For tri-linear interpolation over the scaled cubic mesh one finds $(k/2^{1/3})^4/45$, $(k/2^{1/3})^4/90$ and $(k/2^{1/3})^4/135$ respectively for these MSE_3's. For the *root* mean-square errors this yields $0.094k^2$, $0.066k^2$, and $0.054k$ respectively. We see that in going from the conventional to the tetrahedral mesh we have reduced the worst error (along the axes) a little, but we have lost some of the good performance along the diagonals. However, these changes in magnitude are not dramatic and, apart from the issue of isotropy (*when* this becomes an issue), there is little to choose between tetrahedral and tri-linear cubic interpolation performance. What must be remembered, though, is that tetrahedral interpolation requires only four data references instead of eight, and also, as will be shown in the last section, that the weights are simpler.

8. Aliassing. In our original cubic mesh of side 2 (without cube centers) wave-vector components l, m, n larger than $\pi/2$ in magnitude would be aliassed into the range $|l| \leqq \pi/2$, $|m| \leqq \pi/2$, $|n| \leqq \pi/2$, since the sampling interval in each dimension is 2. Outside a cube of side π in wave-number space (volume π^3), the discrete Fourier transform, applied to meshpoint data, would give periodic repeats. If we were to shrink the mesh size of the grid in x, y, z-space by the factor $2^{1/3}$, as in the last section, the domain of unrepeated harmonic information would grow to a cube of side $2^{1/3}\pi$, and of volume $2\pi^3$.

The introduction of cube-center information in our original x-y-z grid should similarly double the volume of nontrivial harmonic information. However, the shape of the volume in l, m, n-space which holds this information is not cubical. Proceeding parallel to the x-axis, one crosses planes parallel to the x, y-plane which contain function information at *unit* intervals. One therefore expects no aliassing along the l-axis until one reaches $l = \pm\pi$. In other words, in each axial direction (l, m, or n) the "fundamental Brillouin zone" is pushed out by a factor 2.

To find the aliassing limit in other directions, consider our cubic-centered lattice in Cartesian axes $x, (z-y)/\sqrt{2}, (z+y)/\sqrt{2}$. Looking now along x, one sees function information over a square mesh of side $\sqrt{2}$. The corresponding transform space is that of $l, (n-m)/\sqrt{2}, (n+m)/\sqrt{2}$, and the alias-free domain in planes normal to the

l-axis is the square

$$-\frac{\pi}{\sqrt{2}}<\frac{(n-m)}{\sqrt{2}}<\frac{\pi}{\sqrt{2}},$$

$$-\frac{\pi}{\sqrt{2}}<\frac{(n+m)}{\sqrt{2}}<\frac{\pi}{\sqrt{2}}.$$

Outside this square, $|n\pm m|<\pi$, one gets repeats.

The same argument can be applied after cyclic permutations of l,m,n, and one concludes that only the domain

$$|n\pm m|<\pi, \qquad |m\pm l|<\pi, \qquad |l\pm n|<\pi$$

can be alias-free. This domain, bounded by 12 planes, is a rhombic dodecahedron, shaped just like the "domain of influence" of §3 and Figs. 8 and 9. This is the "fundamental Brillouin zone" whose volume is $2\pi^3$ (note the construction from pyramids taken out of the cube of side π, Fig. 8), as predicted on grounds of information content. Again we observe here some progress toward isotropy, from a cube in wave-vector space to a rhombic dodecahedron.

The aliassing limits established here account for why the table of errors was not extended further than shown in §7. Note that the errors do become rather large as one approaches the aliassing boundary.

By making up models of several rhombic dodecahedra all alike, one can convince oneself that these can be stacked into a close packing. All of l,m,n-space can thus be filled from repeats of the "central" rhombic dodecahedron (the one which is placed symmetrically around the origin).

9. Linking discrete spectra to tetrahedral mesh records. Since numerical spectra are necessarily discrete, the use of tetrahedral interpolation on functions represented by such spectra requires some prior discrete Fourier transforming of the spectral data on to our tetrahedral mesh, i.e., the combined mesh of cube corners and centers. One would want to do this transforming efficiently; for instance, it would be wasteful to generate records over a cubic mesh twice as fine as the original and then only use one quarter of the data so generated (discarding both the values at the edge centers and those at the face centers of the original cubes).

In order to achieve the desired economy, we first address the inverse problem of generating discrete spectra from the tetrahedral mesh record. The outer boundary of the domain to be covered by this mesh will be taken as a cube of side $2N$, and periodicity will be used as the boundary condition there. This causes the spectrum to become discrete: l, m, and n occur only in integral multiples of π/N. The total number of harmonics in the fundamental Brillouin zone of l,m,n-space is $2N^3$, the same as the number of data points on our combined (corner and center) mesh within the large periodicity cube.

Before transforming the mesh data into the discrete spectrum, we introduce "skew" coordinates

$$\xi=\frac{x-z}{2}, \qquad \eta=\frac{y+z}{2}, \qquad \zeta=\frac{z-y}{2},$$

which invert to

$$x=2\xi+\eta+\zeta, \qquad y=\eta-\zeta, \qquad z=\zeta+\eta.$$

These variables were chosen with the following in mind: the mean, or the semidifference of any two Cartesian coordinates (such as $(z+y)/2$ or $(x-z)/2$) will always be an integer on the combined (cubic corner plus cubic center) grid. Three choices of such means or differences are to be made, which are independent, so that they can replace the original Cartesian coordinates ($(x-z)/2$, $(z-y)/2$, $(y-x)/2$ will *not* do; they add up to zero). Unfortunately, there is no way of making a choice so that the new system is fully orthogonal. However, two of the new axes can be made mutually orthogonal; with our choice, the η- and the ζ-axes are orthogonal. The lack of symmetry thus introduced into the process of calculation does not produce any asymmetry in the final results. Variables ξ, η, ζ generated from those given above by permutations of x, y, z or by reflections ($x \to -x$ etc.) will yield the same results eventually.

The periodicity in x, y, z manifests itself in ξ, η, ζ as follows:

$$f(\xi+N,\eta,\zeta)=f(\xi,\eta,\zeta),$$
$$f(\xi,\eta+N,\zeta-N)=f(\xi,\eta,\zeta),$$
$$f(\xi-N,\eta+N,\zeta+N)=f(\xi,\eta,\zeta).$$

Combining the first and last gives

$$f(\xi,\eta+N,\zeta+N)=f(\xi,\eta,\zeta).$$

Combining this with the second gives

$$f(\xi,\eta+2N,\zeta)=f(\xi,\eta,\zeta).$$

The periodicities in ξ and η call for FFT-ing in these two variables over N points and $2N$ points respectively. This leads to the representation:

$$f(\xi,\eta,\zeta)=\sum_I \sum_M e^{2\pi i(I\xi/N+M\eta/2N)} f_{I,M}(\zeta),$$

where I runs through N contiguous integers, and M through $2N$ contiguous integers. The periodicity condition $f(\xi,\eta+N,\zeta-N)=f(\xi,\eta,\zeta)$ manifests itself in $f_{I,M}(\zeta)$ as follows:

$$f_{I,M}(\zeta+N)=f_{I,M}(\zeta), \qquad M \text{ even},$$
$$f_{I,M}(\zeta+N)=-f_{I,M}(\zeta), \qquad M \text{ odd}.$$

One therefore defines two functions of ζ,

$$f^{\text{even}}_{I,J}(\zeta)=f_{I,2J}(\zeta) \text{ (the case } M=\text{even}),$$
$$f^{\text{odd}}_{I,J}(\zeta)=e^{\pi i\zeta/N} f_{I,2J+1}(\zeta) \text{ (the case } M=\text{odd}),$$

both of which are now periodic in ζ with period N. After transforming them in ζ over N points, one gets the following representation of $f(\xi,\eta,\zeta)$:

$$f(\xi,\eta,\zeta)=\sum_I \sum_J \sum_K f^{\text{even}}_{I,J,K} e^{2\pi i(I\xi+J\eta+K\zeta)}$$
$$+f^{\text{odd}}_{I,J,K} e^{2\pi i(I\xi+(J+\frac{1}{2})\eta+(K-\frac{1}{2})\zeta)},$$

where I, J and K run through N contiguous integers each. We note that it takes two N-point FFT's plus one $2N$-point FFT, and the intermediate multiplication by $\exp(\pi i\zeta/N)$, to complete the transformation over the variables ξ,η,ζ.

By substituting for ξ,η,ζ in terms of x,y,z one can now compare the result with the direct triple Fourier transform

$$f(x,y,z)=\Sigma\Sigma\Sigma f_{l,m,n}e^{i(lx+my+nz)},$$

where l,m,n run through multiples of π/N. One establishes the following identification of the harmonics:

$f_{l,m,n}=f^{\text{even}}_{I,J,K}$, with

$$\left.\begin{aligned} l &= I\pi/N \\ \frac{l+m+n}{2} &= J\pi/N \\ \frac{l-m+n}{2} &= K\pi/N \end{aligned}\right\} \quad \text{or} \quad \begin{cases} m=(J-K)\pi/N \\ n=(K+J-I)\pi/N \end{cases}$$

when $l+m+n=$ even times π/N;

$f_{l,m,n}=f^{\text{odd}}_{I,J,K}$, with

$$\left.\begin{aligned} l &= I\pi/N \\ \frac{l+m+n}{2} &= \left(J+\tfrac{1}{2}\right)\pi/N \\ \frac{l-m+n}{2} &= \left(K-\tfrac{1}{2}\right)\pi/N \end{aligned}\right\} \quad \text{or} \quad \begin{cases} m=(J-K+1)\pi/N \\ n=(K+J-I)\pi/N \end{cases}$$

when $l+m+n=$ odd times π/N.

In l, m, n-space the cubic lattice characterizing the harmonics has two classes of points which must be dealt with differently, according to the parity of $(l+m+n)N/\pi$. These classes of points are positioned, respectively, like the sodium and chloride atoms in a common salt crystal (see Fig. 5a). I, J, K are useful indices for labeling and recording discrete harmonics. Note that the arguments of the sine-s in the boost-factor formula (λ, §4) are $J\pi/N$, $K\pi/N$, $(J-I)\pi/N$ and $(I-K)\pi/N$ for "even" harmonics, with obvious changes for "odd" harmonics.

The domain in l, m, n-space which is filled when I, J, K each run through a range of N contiguous integers is a parallelepiped. Making the range go from $-N/2$ to $+N/2$ will place this solid symmetrically around the origin. It is bounded in l by the two parallel planes $l=\pm\pi/2$. Any constant $-l$ cross section is a square, bounded by $m+n=\pm\pi-l$ and $n-m=\pm\pi-l$. This solid has the volume $2\pi^3$ and contains $2N^3$ points, but it is not identical with the rhombic dodecahedron described in the previous section. The parallelepiped just described extends into adjacent Brillouin zones in places, and has bits missing from the fundamental Brillouin zone elsewhere.

By constructing models of the solids in question, removing from each the parts projecting beyond their union, and moving these excesses around in accordance with the periodicity rules, one convinces oneself that both solids describe the same harmonics, and that the nonoverlapping parts are aliasses of each other. The situation is analogous to that encountered in one-dimensional Fourier transforming where one would often like to let the harmonic index run from $-N/2$ to $+N/2$ (the latter, perhaps, not inclusive); since programming languages require positive, or at least nonnegative indices, one lets the index run from 0 through $N-1$, say. This one-sidedness does not imply any lack of symmetry of the transformation process. Likewise, we emphasize again that the lack of symmetry in the definition of ξ, η and ζ, or the lack of symmetry of the parallelepiped in l, m, n-space, does not imply any lack of symmetry in the final result when each harmonic has been properly identified.

We have described the passage from the tetrahedral mesh data to harmonic data. The reversal of each step in this procedure is unambiguous and straightforward, so that mesh records can be generated from discrete spectra as necessary.

10. The algorithm for indexing and weight calculations. As a last task, we address the problem of generating both the indices of the corners of the appropriate tetrahedron, and the corresponding weights from the coordinates x, y, z of the position to be interpolated.

A set of skew coordinates ξ, η, ζ has already been introduced which identifies all the points of the combined mesh as the points with integer coordinates. Fourier transforms were performed over integers ξ, η, and ζ as indices, and we shall assume that "boosted" corner values of the function to be interpolated have been tabulated over these three indices. Obviously a first step is to generate ξ, η, ζ from x, y, z, and to obtain the integer triplet $[\xi], [\eta], [\zeta]$, from them by an IFIX or INT routine.

As we shall see, the four relevant index triplets do not necessarily include this particular triplet, but they will be among the eight triplets

$$[\xi],[\eta],[\zeta];\ [\xi]+1,[\eta],[\zeta];\ [\xi],[\eta]+1,[\zeta];\ [\xi]+1,[\eta]+1,[\zeta];$$
$$[\xi],[\eta],[\zeta]+1;\ [\xi]+1,[\eta],[\zeta]+1;\ [\xi],[\eta]+1,[\zeta]+1;$$
$$[\xi]+1,[\eta]+1,[\zeta]+1.$$

We also generate the fractional parts of ξ, η, ζ (FRAC, or ξ-IFIX(ξ), etc.). Under mass-production conditions, one should rescale x, y, z, so that the multiplications by 0.5 are eliminated in the formation of ξ, η, ζ.

When ξ, η, and ζ increase independently and continuously through the ranges $([\xi],[\xi]+1)$, $([\eta],[\eta]+1)$, $([\zeta],[\zeta]+1)$ respectively, the interpolating point in x, y, z-space describes the interior of a parallelepiped like that shown in Fig. 14. There we have illustrated the prototype parallelepiped, near the origin, which has $[\xi]$, $[\eta]$, and $[\zeta]$ all zero. In what follows, translation to similar parallelepipeds elsewhere is trivial and we shall explain the remainder of the index/weight algorithm in terms of the prototype. Henceforth, therefore, ξ, η, ζ are the fractional parts of ξ, η, ζ, and the eight corners of the parallelepiped in which the interpolating point lies are indexed 0,0,0; 1,0,0; 0,1,0; 1,1,0; 0,0,1; 1,0,1; 0,1,1; 1,1,1.

The parallelepiped contains six tetrahedra (designated by roman numerals I – VI), in all the six possible orientations, as seen in Fig. 14. In Table 2, the orientations (in

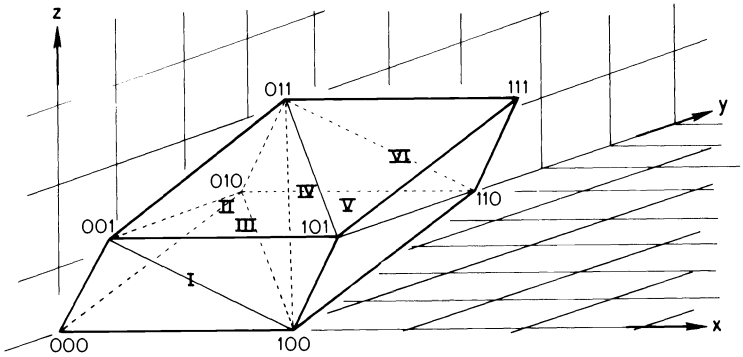


FIG 14. *Unit parallelepiped, divided into six tetrahedra.*

accordance with the definition given in §2) are listed; the four corners for each tetrahedron are also in the table.

We must decide, from the values of x, y, z or ξ, η, ζ, which is the relevant tetrahedron among the six. There are four planes separating the tetrahedra, namely

$$z+x=2; \qquad z+x=4; \qquad x+y=2; \qquad x-y=2.$$

We therefore introduce four "discriminators":

$$\alpha=\frac{z+x}{2}-1=\zeta+\eta+\xi-1; \qquad 1-\alpha=2-\frac{z+x}{2};$$

$$\gamma=\frac{x-y}{2}-1=\xi+\zeta-1; \qquad \delta=\frac{y+x}{2}-1=\eta+\xi-1,$$

whose signs indicate on which side of the planes the interpolating point is located. From Fig. 14 one reads off the appropriate sign combination for each of the six tetrahedra. These combinations are listed in Table 2. Where no sign is entered, the corresponding discriminator becomes irrelevant. For instance, after diagnosing a negative α, no further decision is needed; the relevant tetrahedron is number I. However, for tetrahedron II, we must make sure that α is positive *and* that $1-\alpha$ is positive *and* that γ is negative, *and* that δ is negative. The aggregate of the decisions in each column selects the corresponding tetrahedron uniquely.

Now the remarkable feature of tetrahedral interpolation is that no further work is needed for determining the weights associated with the corners of the selected tetrahedron. Consider, for example, tetrahedron III, which is oriented just like the prototype tetrahedron used in §§3–6 above. The corners are indexed as follows:

$$A\colon 001 \quad B\colon 101 \quad C\colon 100 \quad D\colon 011.$$

Let f now denote the linear approximant pegged at these corners. Then the gradients are as given in §4, and the corner A, where f must be f_{001}, has the Cartesian coordinates $x_0=1, y_0=-1, z_0=1$. Hence

$$f=f_{001}+\frac{f_{101}-f_{001}}{2}(x-1)+\left(\frac{f_{100}+f_{011}}{2}-\frac{f_{001}+f_{101}}{2}\right)(y+1)$$

$$+\frac{f_{011}-f_{100}}{2}(z-1)$$

$$=-f_{001}\delta+f_{100}(1-\zeta)+f_{101}\gamma+f_{011}\eta.$$

The four weights $-\delta, 1-\zeta, \gamma, \eta$ have been recorded in Table 2, under their respective corners in the column for tetrahedron III. The weights recorded in the other columns were obtained in similar manner.

We note that the weights are absolute values of discriminators used in the identification of the appropriate tetrahedron, unless they are coordinates ξ, η, ζ, or complements of coordinates, $1-\xi, 1-\eta, 1-\zeta$. With suitable scaling of x, y, z, the algorithm for determining corners and weights involves only additions or subtractions and logical operations. When using the table for the purpose of identifying tetrahedra, corners and weights, one must remember that it was written for the prototype

TABLE 2
Decisions, corners and weights.

Tetrahedron Orientations	I *xzy*	II *yxz*	III *xyz*	IV *zyx*	V *zxy*	VI *yzx*
Discriminators α	−	+	+	+	+	−
Discriminators $1-\alpha$		+	+	+	+	
Discriminators γ		−	+	−	+	
Discriminators δ		−	−	+	+	
1st Corner	000	001	001	010	011	011
Weight	$\lvert\alpha\rvert$	$\lvert\delta\rvert$	$\lvert\delta\rvert$	$\lvert\gamma\rvert$	$1-\xi$	$1-\xi$
2nd Corner	001	100	100	011	110	110
Weight	ζ	ξ	$1-\zeta$	ζ	$\lvert\delta\rvert$	$1-\zeta$
3rd Corner	100	010	101	110	100	111
Weight	ξ	$\lvert\gamma\rvert$	$\lvert\gamma\rvert$	$\lvert\delta\rvert$	$\lvert 1-\alpha\rvert$	$\lvert 1-\alpha\rvert$
4th Corner	010	011	011	100	101	101
Weight	η	$\lvert\alpha\rvert$	η	$1-\eta$	$\lvert\gamma\rvert$	$1-\eta$

parallelepiped. In general, "ξ" "η" and "ζ" must be interpreted as $\xi-[\xi], \eta-[\eta]$, $\zeta-[\zeta]$, both in the table and in the expressions for α, γ and δ. Also, a corner, such as "001", must be interpreted as the point with the index triplet $[\xi],[\eta],[\zeta]+1$, and similarly for the other seven corners.

The logic of tri-linear interpolation in a cubic mesh is trivial and familiar. So is the arithmetic of calculating the eight weights. One therefore naturally hesitates before embarking on the novel and, at first sight, more complicated procedures for a tetrahedral mesh. However, the effort has been rewarded; not only is the logic of locating the relevant tetrahedron and obtaining the indices of the four corners quite simple, but the arithmetic of calculating the four weights has virtually disappeared!

11. Summary. Driven by the need to minimize table look-ups in three-dimensional interpolation, we have found a space-filling array of tetrahedral finite elements with a number of favorable features. The tetrahedral mesh is generated by a cubic mesh plus the mesh connecting the cube centers plus the space diagonals. Used for Laplace's equation (as an example of a PDE), the linear tetrahedral finite elements yield the result that cube-center values must be averages of the eight nearest corner values and vice versa. Used as optimal interpolants, specifically for spectral data, these elements yield mean-square errors which are comparable in magnitude, but more isotropic than, those for tri-linear interpolation over a cubic mesh with the same information density. The "fundamental Brillouin zone" in wave-number space, outside of which aliassing occurs, is a rhombic dodecahedron rather than a cube. With suitable indexing of the grid points and the harmonics, one can employ FFT's economically for passing between discrete spectra and mesh data. The same grid-point indexing leads to a compact algorithm for indentifying the four corners of the tetrahedron which encloses the interpolating point (x, y, z), and for obtaining the four weights to be applied at these corners. Thus it seems that a 2:1 saving in interpolation and look-up effort (over tri-linear interpolation in a simple cubic mesh) is achieved at no loss in quality of interpolation.

REFERENCES

[1] O. C. ZIENKIEWICZ, *The Finite Element Method* (3rd ed.), McGraw- Hill, New York, 1977, Ch. 6.

[2] G. PÓLYA, *Sur une interprétation de la méthode des différences finies qui peut fournir des bornes supérieures ou inférieures*, Comptes Rendues, 235 (1952), pp. 995–997.

[3] A. F. WELLS, *The Third Dimension in Chemistry*, Clarendon Press, Oxford, 1956.

[4] C. K. BIRDSALL, LIU CHEN, AND A. B. LANGDON, *Reduction of the grid effects in simulation plasmas*, J. Computational Phys., 14 (1974), pp. 200–222.

[5] A. B. LANGDON, *Energy conserving plasma simulation algorithms*, J. Computational Phys., 12 (1973), pp. 247–268.

[6] J. W. EASTWOOD, *Optimal particle-mesh algorithms*, J. Computational Phys., 18 (1975), pp. 1–20.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0008 $1.00/0

AUTOMATIC COMPUTER CODE GENERATION FOR HYPERBOLIC AND PARABOLIC DIFFERENTIAL EQUATIONS*

BJÖRN ENGQUIST† AND TOM SMEDSAAS‡

Abstract. A program system which generates computer code for the numerical solution of systems of hyperbolic and parabolic differential equations is described. The input to the program is a mathematical formulation of a hyperbolic or parabolic initial boundary value problem in one space dimension. The differential equations and boundary conditions are analyzed by the program system, and a finite difference algorithm is designed for the given problem. The output is an executable FORTRAN program.

Key words. mathematical software, difference approximations, symbolic analysis, preprocessing

1. Introduction. In many areas of mathematical software such as linear algebra, optimization and ordinary differential equations there exist general and widely used programs and subroutine libraries. However, general software for partial differential equations (PDE) is in a less developed stage.

The most important reason for this is the great variety of mathematical properties of PDE. There is no single algorithm which can handle all problems. The numerical techniques must often be highly problem dependent, and the existing algorithms are developed for narrow classes of problems. Furthermore, the numerical computations are often very time-consuming. Special properties of a problem such as linearities and constant coefficients must be utilized in order to decrease the cost of the calculation. Consequently, general software for PDE must incorporate several algorithms and should be able to adapt such algorithms to a specific problem. This is very difficult to achieve in a general program package of conventional type which consists of a number of ready-made subroutines.

The aim of the work described in this paper is to develop methods for increasing the flexibility of program packages so that the requirements on PDE software can be met. Our main tools to achieve these goals are combinations of symbolic analysis and automatic code generation. We have developed software for general systems of hyperbolic and parabolic initial boundary value problems in one space variable. The program package is here called DCG (Differential equation and Code Generator). It is primarily intended as an academic experimental system, but has also been used in a number of practical applications.

Let us summarize some of the advantages in the analyzer and code generation technique for general PDE software.

In PDE software it is common that the user specifies his problem by means of parameters and subroutines. For example, the user may be asked to write a FORTRAN function $F(X, T, U, UX, UXX)$ representing the right-hand side in the general equation

$$u_t = f(x, t, u, u_x, u_{xx}). \tag{1.1}$$

*Received by the editors September 6, 1979, and in final revised form February 19, 1980. This work was supported by The Swedish Natural Science Research Council (NFR 2711-018) and The Swedish Board for Technical Development (STU 77-3690).

†Mathematics Department, University of California, 405 Hilgard Ave., Los Angeles, California 90024.
‡Department of Computer Sciences, Uppsala University, Sturegatan, 4B, 75223 Uppsala, Sweden.

This method has, however, several drawbacks. The most important one is that the structure of the problem is hidden for the solver. It is harder to check well-posedness and to choose algorithm.

Consider the example

$$u_t = f(x, t, u, u_x, u_{xx}) = a(x)u_x + b(t) \tag{1.2}$$

in the area $0 < x < 1, t > 0$ where the user has given a boundary condition at the left boundary. The problem is well-posed if $a(0) \leqq 0$. This is, of course, very simple to check. If the right-hand side of (1.2) is given as a general function $f(x, t, u, u_x, u_{xx})$, we have to approximate numerically the derivative of f with respect to u_x for each time interval, and check if it is nonpositive. This calculation is less exact and more time consuming. The problem is even more serious for systems of equations, because the corresponding check involves eigenvalue and eigenvector computations.

The efficiency of the numerical integration is also affected by the common approach with given FORTRAN functions. With m points in the space direction and n points in the time direction the function f in (1.2) needs to be evaluated at least mn times. Every evaluation of f will evaluate $a(x)$ and $b(t)$. However, since $a(x)$ is independent of t and $b(t)$ independent of x, m and n evaluations, respectively, are sufficient.

When the structure of a problem is available, the efficiency of implicit methods can be increased substantially, since linearities and constant coefficients can be discovered and the algorithms adjusted accordingly.

There are also nonnumerical advantages with the analyzer and code generator approach:

—Since the problem specification is done in a notation close to the mathematical formulation, the risk for errors is much less than when the user translates his problem into FORTRAN routines.

—The generated program is small and easy to handle, since it only contains code which is needed in a particular application.

—The code generation technique makes it possible to tailor a program not only to a certain problem but also to a particular computer configuration. It is thus easy to support versions for single and double precision, time sharing and batch environments, graphic equipment, etc.

—It is easy to extend the system with new algorithms.

We have decided to separate the system into two different programs, *the analyzer* and *the synthesizer*. The analyzer handles the user communication and the syntax and semantic analysis of the problem. An output file is generated with instructions to the synthesizer. This file contains patches of code to be used in the final FORTRAN program, and flags indicating the characteristics of a particular problem. The synthesizer, which is a general FORTRAN preprocessor, generates a program from a "preprogram" in a code library according to instructions from the analyzer (Fig. 1.1).

The analyzer is written in SIMULA 67 [3], and the synthesizer in FORTRAN.

In §2 we define the mathematical problem to be approximated and describe the numerical methods. Section 3 contains a brief presentation of the problem description language, and in §4 the structure and function of the software system is outlined. An

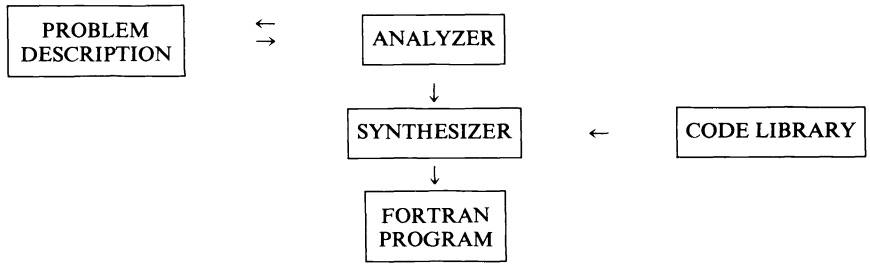


FIG. 1.1. *The overall structure of the system.*

example is presented in §5 where DCG is applied to a nonlinear system of hyperbolic differential equations describing the waterflow in a river. The necessary specifications and the analysis in DCG are outlined.

The DCG system has also been used for a number of other applications, including oscillations in elastic beams, radial motion in spherical shells, magnetohydrodynamic equations for isentropic plasmas, two phase flow, chemical convection problems, and heat conduction.

For more details of the system and the applications, see [5] and [8]. A preliminary version of the system was presented in [7]. The system can be used on computers with both SIMULA and FORTRAN compilers. A new version which is based only on FORTRAN is under development.

During the last few years successful special purpose PDE software has been developed. Finite element packages for problems in structural mechanics belong to this class. References to other general program packages for PDE problems are given in the surveys [13] and [16]. Preprocessor techniques have, for example, been used in [1], [9], and [14].

2. Mathematical problem and numerical methods. We want to approximate the solution $U(x,t)$ of a hyperbolic or parabolic system of differential equations

$$\begin{aligned} &U_t = F(x,t,U,U_x,U_{xx}), \qquad a<x<b, t>t_0, \\ &U=(u_1(x,t),\dots,u_d(x,t))^T, \\ &F=(f_1(x,t,U,U_x,U_{xx}),\dots,f_d(x,t,U,U_x,U_{xx}))^T. \end{aligned} \tag{2.1a}$$

("T" denotes the transpose of a vector or a matrix).

We are also given initial and boundary conditions

$$U(x,t_0)=U_0(x), \qquad a<x<b, \tag{2.1b}$$

$$U^{(1)}(a,t)=G_1(t,U(a,t)), \quad U^{(2)}(b,t)=G_2(t,U(b,t)), \tag{2.1c}$$

where G_1 and G_2 are vectors with dimensions d_1 and d_2 respectively. The vectors $U^{(1)}$ and $U^{(2)}$ denote subspaces of U containing d_1 and d_2 of the components $u_j(x,t)$ respectively. We may also have periodic boundary conditions $U(a,t)=U(b,t)$.

The linearized equations are assumed to be strongly well-posed. Hence, for hyperbolic problems where F in (2.1a) does not depend on U_{xx}, we assume the Jacobian matrix F_4 to have real eigenvalues. The matrix F_4 denotes the Jacobian of F with respect to its fourth argument U_x. It must also be possible to transform F_4 to diagonal form with smooth and bounded matrices. For parabolic problems, the Jacobian matrix with respect to the last argument U_{xx}, has eigenvalues with positive real part bounded away from zero.

We further assume that the boundary conditions are such that the linearized problem is well-posed in L_2 (see [11]). For hyperbolic problems, this implies that d_1 equals the number of negative eigenvalues of F_4 at $x=a$, and d_2 equals the number of positive eigenvalues of F_4 at $x=b$. For parabolic problems d boundary conditions are always given at both boundaries.

The problem must fulfill these conditions in order to have a solution which depends continuously on the initial and boundary data. The eigenvalue conditions are checked by the DCG-produced program. The QR algorithm is used in all eigenvalue computation when $d>2$. For smaller systems the eigenvalues are calculated explicitly. Another control of well-posedness is checking the singularity of the transformation matrix T_4 in (2.9). Appropriate error messages are given.

The DCG system is based on second order finite difference methods. They are very likely the most frequently used schemes for one-dimensional hyperbolic and parabolic problems.

For hyperbolic problems we use the explicit leap-frog scheme (A), a semi-implicit leap-frog scheme (B), and the implicit hyperbolic Crank-Nicolson scheme (C). For parabolic problems we have the explicit Du Fort and Frankel scheme coupled to leap-frog (D), the corresponding semi-implicit scheme (E), and the Crank-Nicolson scheme (F).

By this set of methods the user can choose a more or less implicit treatment of F. All these schemes are energy conserving when applied to symmetric hyperbolic equations but an option for adding dissipation is provided.

We let the user specify the difference method and the amount of dissipation. The user also determines which terms in F should be treated implicitly, and whether variable steps should be used. See [4] for the importance of treating stiff terms implicitly.

The system decides the extra boundary conditions that are necessary for the calculations. It also determines the time step in order to guarantee stability and accuracy.

The solution $U(x,t)$ is approximated by the meshfunction U_i^n at the points (x_i, t^n), where $i=0,1,\ldots,I$ and $n=0,1,\ldots$:

$$x_i=a+\sum_{\mu=0}^{i-1}\Delta x_\mu, \qquad t^n=t_0+\sum_{\nu=0}^{n-1}\Delta t_\nu. \tag{2.2}$$

When the solution is expected to have much larger derivatives in one region than in another, it is a wise policy to have a finer grid in the region with the rougher solution. We let the user specify the distribution of meshpoints by giving a transformation of the space interval,

$$x=f(z), \qquad z=f^{-1}(x). \tag{2.3}$$

The mesh will be uniform in the z variable. In the present version of DCG the function f in (2.3) must be time-independent.

Let us as an example of the difference methods give the formula for the scheme B at interior mesh points. We will give all formulas with constant step sizes for convenience. Assume for this scheme that the function F in (2.1a) is written

$$F=F^{(1)}(x,t,U,U_x)+F^{(2)}(x,t,U). \tag{2.4}$$

The function $F^{(2)}$ should contain the stiff terms for implicit differencing.

B.
$$U_i^{n+1}-\Delta tF^{(2)}(x_i,t^{n+1},U_i^{n+1})$$
$$=U_i^{n-1}+\Delta tF^{(2)}(x_i,t^{n-1},U_i^{n-1})+2\Delta tF^{(1)}\left(x_i,t^n,U_i^n,\frac{U_{i+1}^n-U_{i-1}^n}{2\Delta x}\right),$$
$$i=1,2,\ldots,I-1,\quad n=1,2,\ldots$$

For nonlinear hyperbolic problems many schemes, including the linearly energy conserving schemes A and C, can generate oscillations and nonlinear instability (see [12]). This can often be cured by adding dissipative terms.

In the dissipative case the algorithms A, B and C are changed by adding the following terms to the right-hand side.

$$-\frac{r}{16}(U_{i+2}^{n-1}-4U_{i+1}^{n-1}+6U_i^{n-1}-4U_{i-1}^{n-1}+U_{i-2}^{n-1}) \quad \text{for } i=2,\ldots,I-2, \tag{2.5a}$$

$$\frac{r}{4}(U_{i+1}^{n-1}-2U_i^{n-1}+U_{i-1}^{n-1}) \quad \text{for } i=1,\ldots,I-1,\quad 0\leqq r<1, \tag{2.5b}$$

where r is the dissipation coefficient. (The index is n instead of $n-1$ for scheme C.) The explicit formulas A and B are conditionally stable (see [6]) when

$$\Delta t<\frac{(1+\sqrt{1-r}\,)^{1/2}}{\sqrt{2}\,\rho(F_4)}\Delta x. \tag{2.6}$$

Here $\rho(F_4)$ denotes the spectral radius of the functional matrix F_4. The time step in the program is chosen in order to guarantee stability. The implicit scheme C and the parabolic schemes are unconditionally stable.

There is also an option to let the local error influence the choice of time steps. The length of the time step is determined such that a difference approximation of the local truncation error is below a given value. This is similar to the step size control in many codes for ordinary differential equations.

The extra initial conditions for the two step schemes A (or B), D (or E) are given by an Euler (or semi-implicit Euler) step.

For parabolic problems all components of U are given at the boundary by (2.1c). Generally, this is not the case for hyperbolic problems. Extra numerical boundary conditions are needed and it is not trivial to determine them in a stable way. Let us exemplify by the algorithm for the schemes A and B at $x=a$. The right boundary is treated analogously. Our choice of formulas is justified by the analysis in [10].

a) If $d_1 = d$, there is no need for extra boundary conditions, and (2.1c) is used to determine U.

b) If $d_1 = 0$, we need d extra boundary conditions and we use oblique extrapolation,

$$U_0^{n+1} = 2U_1^n - U_2^{n-1}. \tag{2.7}$$

c) When $d_1 \neq d$ and $d_1 \neq 0$, the choice of stable boundary conditions is more delicate. We use extrapolation in the characteristic coordinates to get the necessary number of d conditions. The formula (2.1c) gives d_1 conditions only. Assume

$$U_i^n = \begin{pmatrix} U^{(1)} \\ U^{(2)} \end{pmatrix}_i^n, \tag{2.8}$$

where $U^{(1)}$ is given through (2.1c) at $x = a$. If U does not have this form the unknowns can be reordered to achieve this separation of $U^{(1)}$ and $U^{(2)}$ in U.

A matrix $T(=T^n)$ is then determined so that it diagonalizes the Jacobian F_4 at $x = a$:

$$TF_4T^{-1} = \begin{pmatrix} D_{(-)} & 0 \\ 0 & D_{(+)} \end{pmatrix} \qquad T = \begin{pmatrix} T_1 & T_2 \\ T_3 & T_4 \end{pmatrix}, \tag{2.9}$$

where $D_{(-)}$ is a diagonal d_1 by d_1-matrix with negative diagonal elements. The $(d-d_1)$ by $(d-d_1)$-matrix $D_{(+)}$ is diagonal with positive diagonal elements. Let T (in (2.9)) be partitioned in the same way as D. There exists such a nonsingular transformation matrix T since the initial boundary value problem for the differential equations is assumed to be strongly hyperbolic [11].

We extrapolate in the characteristic quantities

$$T_3U^{(1)} + T_4U^{(2)}, \tag{2.10}$$

corresponding to the positive eigenvalues of F_4; the formulas for $U^{(2)}$ at the boundary can now be written as

$$\left(T_3U^{(1)} + T_4U^{(2)}\right)_0^{n+1} = 2\left(T_3U^{(1)} + T_4U^{(2)}\right)_1^n - \left(T_3U^{(1)} + T_4U^{(2)}\right)_2^{n-1}. \tag{2.11}$$

Equation (2.11) is coupled to the given boundary conditions (2.1c):

$$U_0^{(1)n+1} = G_1\left(t^{n+1}, U_0^{n+1}\right). \tag{2.12}$$

The boundary for the scheme B is treated in the same way. For Crank-Nicolson we determine the extra boundary conditions by the box scheme instead of oblique extrapolation [10].

The implicit or semi-implicit schemes give rise to systems of algebraic equations. They are solved by Gaussian elimination or (for nonlinear equations) by Newton's method. The algorithms are adjusted to the band structure of the problem. The boundary equations (2.11) and (2.12) are also solved by the standard Newton's method. If G_1 is linear in U_0^{n+1} or independent of U_0^{n+1}, Gaussian elimination is used for (2.11)-(2.12) or just (2.11), respectively.

3. Problem description language. The user specifies the problem in the problem description language (PDL). This specification is interactively processed by the

analyzer. The syntax of PDL is not restricted to one space dimension; however, several space dimensions cannot yet be semantically and numerically processed. (A primitive version for two space dimensions has now been implemented.) A formal grammar for PDL and other details are given in [8].

The problem description is divided into sections, of which there are two different groups. The first group describes the PDE problem itself and must always be present. This group consists of four sections:

GEOMETRY	Declares the independent variables and their domain.
EQUATIONS	Specify the partial differential equations. The equations may contain the ordinary FORTRAN operators and functions, and also differentiation operators and user-supplied functions.
INITIAL CONDITIONS	Specify the solution at the initial time.
BOUNDARY CONDITIONS	Specify boundary conditions.

The sections from the second group may be used to request different options in the problem processing. They determine time and space discretization, numerical methods, output, etc.

METHOD	Specifies the numerical method to be used.
DISSIPATION	Determines the amount of dissipation.
STIFF TERMS	Request implicit treatment of certain terms in the equations.
STEP SIZES, NUMBER OF STEPS	Specify time and space discretization.
OUTPUT AT VALUE, OUTPUT AT STEP	Specifies output at certain time and space values.
SHORTHANDS	Declare substitutions of expressions to be used in the problem specification.
TRANSFORMATIONS	Determine a variable space step size.

See §5 for an example.

4. System structure and function. Let us briefly describe the organization of the system, the datastructures and the nonnumerical algorithms. As mentioned in the introduction, the system consists of an analyzer, a synthesizer, and a code library (Fig. 1.1).

The system also includes routines for basic numerical and utility functions like linear and nonlinear equation solvers, eigenvalue routines, IO-routines, etc. These routines need not be preprocessed, and thus can be compiled and stored in a separate library. Moreover, an interactive graphical postprocessor has been developed to facilitate the study of the produced solution [5].

The *analyzer* processes the problem symbolically. The principal parts of this program are the internal structure, the editor, the parser, the semantic analyzer, and the output generator.

The internal structure plays a fundamental role in the analyzer. We use a general list structure almost identical to the list structure in LISP [2]. The elements in this

structure are atoms and lists. Atoms are used to represent identifiers, constants, operators, keywords, etc. Lists are used to put atoms together to expressions, statements, and sections. Expressions are stored in prefix notation. For example, the equation

$$v_t = cu_x + v, \tag{4.1}$$

which in PDL is written

$$V.T = C*U.X + V, \tag{4.2}$$

is represented by

$$(=(.VT)(+(*C(.UX))V)) \tag{4.3}$$

in the internal structure. (The dot denotes differentiation in PDL.) This internal structure is simple, general, and suitable for the operations we want to perform.

The editor enables the user to specify and edit interactively a problem description on disc storage. The parser performs the syntactical analysis of PDL, and builds an internal representation of the problem description. The parsing is done by recursive descent with semantic routines for building the internal structure.

The semantic analyzer classifies the problem, computes various functional matrices, investigates linearities, etc. Expressions to be used in the numerical algorithms are constructed. Typical operations involved are symbolic differentiation and evaluation.

The output generator writes a file with instructions to the synthesizer. This file contains code to be used in the final FORTRAN program, and different flags indicating the characteristics of the particular problem.

The purpose of the *synthesizer* and the *code library* is to compose a complete FORTRAN program for the numerical treatment of the problem. There are essentially two things that have to be done. The synthesizer has to select code from the library according to instructions from the analyzer, and expressions generated by the analyzer have to be inserted in the code. We have developed a general FORTRAN preprocessor for this task [15]. The input to the preprocessor is a file containing a combination of preprocessor and FORTRAN statements.

5. Example: streaming water. In this application, a program is constructed for calculating the water level in a river. The flow is controlled at the upper ($x=0$) and the lower ($x=A$) end points of a particular section. The shallow water equations (see, e.g., [12]) are used, and the unknowns are the water velocity $v(x,t)$ and the water level $h(x,t)$. When $v>0$, as we expect in this application, the differential equations can be written

$$\begin{aligned} v_t &= vv_x - 9.81h_x + s(x) - f(x)v^2, \\ h_t &= -hv_x - vh_x, \end{aligned} \qquad 0<x<A, \quad t>0 \quad (A=200\,000), \tag{5.1a}$$

with the initial and boundary conditions

$$v(x,0)=a(x), \qquad h(x,0)=b(x), \tag{5.1b}$$

$$v(0,t)h(0,t)=q_L(t), \qquad v(A,t)h(A,t)=q_R(t). \tag{5.1c}$$

Here $s(x)$, $f(x)$, $a(x)$, $b(x)$, $q_L(t)$ and $q_R(t)$ are given as FORTRAN functions by the user. In this application, they usually are interpolations from a table of measured data. The function $s(x)$ in (5.1a) contains the effect of the slope of the river. The term $f(x)v^2$ represents the friction, and q_L and q_R are the flow at the upper and lower end points respectively.

The input to DCG, of course, must contain the differential equations, the space and time intervals, and the initial and boundary conditions. We further give the number of space steps to be used and output control statements. In this case we want the unknowns to be printed at each half hour (1800 sec), and at the first ten time steps. We also specify that we want to treat the friction term implicitly, and hence declare $f(x)v^2$ as a stiff term. Since the functions q_L and q_R change rather abruptly in our application, we have requested some dissipation. Fig. 5.1 shows the complete problem definition in PDL. The specification is given via a dialogue between the user and the system. From the problem specification, DCG constructs FORTRAN code which together with the user supplied functions gives an executable program.

In the first step of this process the parser reads the input, checks the syntax, and builds an internal representation. The analyzer then extracts, among other things, the following information from the specification.

1. The problem is quasilinear. Hence the eigenvalues of the functional matrix, which are used when controlling the time step, must be checked throughout the time interval. Similarly, the eigenvectors for the boundary conditions must regularly be recalculated. The functional matrix can be determined analytically.

2. The stiff term $f(x)v^2$ has an especially simple structure. In general, this term introduces a system of nonlinear algebraic equations for each meshpoint. The sizes of

```
EQUATIONS
   V.T = -V*V.X - 9.81*H.X + S(X) - FR;
   H.T = -H*V.X - V*H.X;

METHOD
   LEAP FROG;

STIFF TERMS
   FR = F(X)*V*V;

GEOMETRY
   SPACE : ( X ! X>0, X<2E5 );
   TIME  : ( T ! T>0, T< 3*24*3600 );

BOUNDARY CONDITIONS
   X = 0    : V = QL(T)/H;
   X = 2E5  : V = QR(T)/H;

INITIAL CONDITIONS
   V = A(X);
   H = B(X);

NUMBER OF STEPS
   X : MAX 150;

DISSIPATION
   0.5;

OUTPUT AT VALUE
   T : 1800(1800);

OUTPUT AT STEP
   T : 1(1)10;
```

FIG 5.1. *Specification of the problem* (5.1). *User written* FORTRAN *routines for the functions* S, F, QL, QR, A, *and* B *are added to the* DCG*-produced program.*

```
C
C                DIFFERENCE APPROXIMATIONS FOR SPACE DERIVATIVES
C
      DO 10 I= 1, NEQU
         DUDX(I) = (UPRES(I,J+1) - UPRES(I,J-1)) / DX2
   10 CONTINUE
C
C                EVALUATE EQUATION EXPRESSIONS
C
      F(1) = (((-(UPRES(1,J)*DUDX(1)))-(9.81*DUDX(2)))+S(X(J)))
      F(2) = ((-(UPRES(2,J)*DUDX(1)))-(UPRES(1,J)*DUDX(2)))
C
C                LEAP FROG STEP
C
      DO 20 I = 1, NEQU
         UFUTU(I,J) = UPAST(I,J) + DT2*F(I)
   20 CONTINUE
C
C                ADD DISSIPATION
C
      IF ( J.EQ.LX .OR. J.EQ.RX )  GOTO 40
      DO 30 I= 1, NEQU
         IF( J.GT.LXP1 .AND. J.LT.RXM1 )
     $      UFUTU(I,J) = UFUTU(I,J) - 0.0625*DISSIP *
     $                   ( UPAST(I,J+2) - 4.*UPAST(I,J+1) +
     $                          6.*UPAST(I,J)  +
     $                     UPAST(I,J-2) - 4.*UPAST(I,J-1)    )
         IF ( J.EQ.LXP1 .OR. J.EQ.RXM1 )
     $      UFUTU(I,J) = UFUTU(I,J) + 0.25*DISSIP *
     $            (UPAST(I,J+1) - 2.*UPAST(I,J) + UPAST(I,J-1))
   30 CONTINUE
   40 CONTINUE
```

FIG 5.2. *Generated* FORTRAN *code for the leap-frog step. The arrays* UFUTU, UPRES *and* UPAST *contain the solution. The indices* LX *and* RX *for the endpoints of the x-interval are* 1 *and* 150 *respectively.*

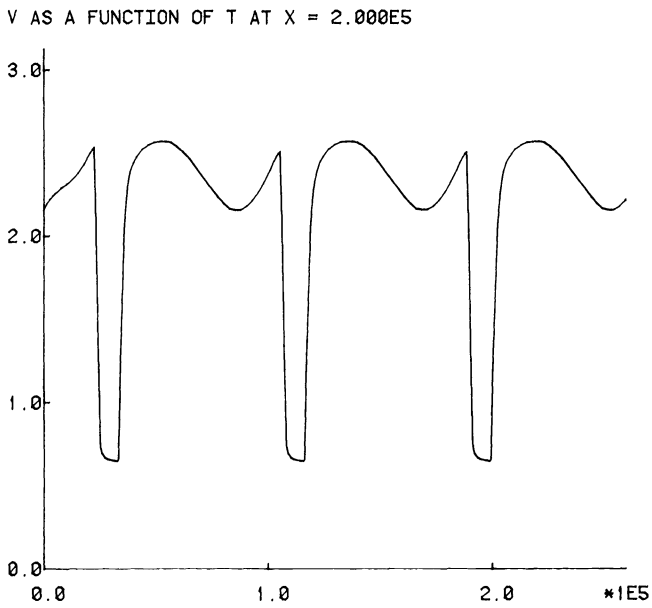


FIG 5.3. *The water velocity as a function of time. In this calculation* q_L *and* q_R *are stepfunctions and the number of space steps is* 150.

these systems are equal to the number of unknowns. Here each local system is simply one scalar quadratic equation.

The semi-implicit leap-frog type scheme is chosen, and preprocessor code is generated by the analyzer. This code then interacts in the preprocessor with code from the library, and results in a FORTRAN program. Fig. 5.2 shows the generated code for the leap-frog step.

The FORTRAN program of course also contains routines for output, and for the calculation of necessary eigenvalues and eigenvectors which are used to determine the time step and the numerical boundary conditions. In Fig. 5.3 the solution is presented using an interactive graphics system developed for this project.

REFERENCES

[1] A. F. CARDENAS AND W. J. KARPLUS, PDEL—*a language for partial differential equations*, Comm. ACM 13 (1970), pp. 184–191.

[2] J. MCCARTHY, LISP 1.5 *Programmers Manual*, MIT Press, Cambridge MA, 1972.

[3] O. J. DAHL, B. MYHRHAUG, AND K. NYGAARD, SIMULA 67 *Common Base Language*, Norwegian Computer Centre pub. S-22, Oslo, 1971.

[4] G. DAHLQUIST, *A special stability problem for linear multistep methods*, BIT, 3 (1963), pp. 27–43.

[5] DCG *Progress Report* 1977, Dept. of Comp. Sciences, Uppsala University, Report no. 70, 1977.

[6] B. ENGQUIST, B. GUSTAFSSON, AND J. VREEBURG, *Numerical solution of a* PDE*-system describing a catalytic converter*, J. Comp. Phys., 27 (1978), pp. 295–314.

[7] B. ENGQUIST AND T. SMEDSAAS, DCG—*a system for automatic code generation for hyperbolic problems*, SIGNUM Newsletter, 10 (1975), pp. 20–22.

[8] ——— *A System for Automatic Code Generation for Hyperbolic Problems*, Dept. of Comp. Sciences, Uppsala University, Report no. 69, 1977.

[9] J. GARY AND R. HELGASON, *An extension of* FORTRAN *containing finite difference operators*, Software Practice & Experience, 2 (1972), pp. 321–336.

[10] B. GUSTAFSSON, H. -O. KREISS, AND A. SUNDSTROM, Stability theory of difference approximations for mixed initial boundary value problems, II, Math. Comp., 26 (1972), pp. 649–686.

[11] H. -O. KREISS, *Initial Boundary Value Problems for Hyperbolic Systems*, Comm. Pure Appl. Math., 23 (1970), pp. 277–298.

[12] H. -O. KREISS AND J. OLIGER, *Methods for Approximate Solution of Time Dependent Problems*, GARP Publication Series, No. 10, 1973.

[13] R. NILSEN AND W. KARPLUS, *Continuous System Simulation Languages: A State of the Art Survey*, Annales de l'Association Internationale pour le Calcul analogic, No. 1, 1974.

[14] R. RICE, ELLPACK: *A Research Tool for Elliptic Partial Differential Equations Software*, in Mathematical Software III, Academic Press, New York, 1977, pp. 319–341.

[15] T. SMEDSAAS, PRELAN—*A Macro Preprocessor for Adaptable Software*, Dept. of Comp. Sciences, Uppsala University, Report No. 78, 1979.

[16] R. SWEET, *A Survey of the State of Software for Partial Differential Equations*, Dept. of Comp. Sciences Report, Stanford University, Stanford CA, 1979.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0009 $1.00/0

COLLOCATION WITH POLYNOMIAL AND TENSION SPLINES FOR SINGULARLY-PERTURBED BOUNDARY VALUE PROBLEMS*

JOSEPH E. FLAHERTY† AND WILLIAM MATHON‡

Abstract. Collocation methods using both cubic polynomials and splines in tension are developed for second-order linear singularly-perturbed two-point boundary value problems. Rules are developed for selecting tension parameters and collocation points. The methods converge outside of boundary layer regions without the necessity of using a fine discretization. Numerical examples comparing the methods are presented.

Key words. Collocation methods, splines under tension, two-point boundary value problems, singular perturbations, boundary layers

1. Introduction. We consider the numerical solution by collocation methods of the singularly-perturbed second-order linear boundary value problem

$$Ly \equiv \varepsilon y'' + p(x)y' + q(x)y = f(x), \qquad a \leqq x \leqq b, \tag{1.1}$$

$$\alpha_{11}y(a,\varepsilon) + \alpha_{12}y'(a,\varepsilon) = A,\ \alpha_{21}y(b,\varepsilon) + \alpha_{22}y'(b,\varepsilon) = B, \tag{1.2}$$

where ε is a small positive parameter. The functions p, q, and f are assumed to be smooth functions of x on $[a, b]$ which, along with α_{ij}, $i, j = 1, 2$, A, and B, may depend on ε provided they are bounded as $\varepsilon \to 0$. We further assume that (1.1, 2) has a unique solution on $[a, b]$ for all ε sufficiently small.

The problem (1.1, 2) has been intensively studied analytically (cf. Cole [6], Eckhaus [11], or O'Malley [18]) and it is known that its solution generally has a multiscale character; i.e., it features regions called "boundary layers" where the solution varies rapidly. Away from the boundary layers, the solution is approximately determined by neglecting the $\varepsilon y''$ term in (1.1) and perhaps one or both of the boundary conditions (1.2).

The problem has also been extensively studied numerically, and it is known that most classical methods fail when ε is small relative to the mesh width h that is used for the discretization of the operator L. There are, however, several finite difference methods (cf. Abrahamson, Keller, and Kreiss [1], Berger, et al. [4], Il'in [16], Kreiss [17], and Pearson [19], [20]), Galerkin finite element methods (cf. Hemker [15], de Groen and Hemker [10], and Heinrich, et al. [13], [14]), and methods based on singular perturbation theory (cf. Flaherty and O'Malley [12] and Steele [29]) that do not require h/ε to be small.

Our aim in this paper is to show that collocation methods with either piecewise polynomials or splines in tension can furnish accurate numerical approximations of (1.1, 2) when h/ε is either small or large. Splines in tension were first used by Schweikert [27] as a means of eliminating spurious oscillations in curve fitting with cubic splines. They have been subsequently studied by Cline [5], Pruess [22], Späth

*Received by the editors November 29, 1979, and in final revised form April 21, 1980. This work was supported by the U.S. Air Force Office of Scientific Research, under Grant AFOSR-75-2818.

†Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York 12181.
‡IBM Federal Systems Division, Gaithersburg, Maryland 20760.

[28], and de Boor [9, Chap. 16]. Between knot points a spline in tension is an L-spline satisfying the differential equation

$$(\tau'' - \rho^2\tau)'' = 0, \tag{1.3}$$

subject to appropriate continuity conditions at the knots. The quantity ρ is called the tension parameter. When $\rho = 0$, τ is a cubic polynomial spline; however, when $\rho > 0$, the solution of (1.3) is a linear combination of the four functions 1, x, $e^{\rho x}$, $e^{-\rho x}$. The exponential functions should be better suited than polynomials at following the rapid variations that are typically found in singular perturbation problems. Indeed, exponentials have been used in the Galerkin methods of Hemker [15] and de Groen and Hemker [10], Il'in's difference method [16], and the singular perturbation methods of Flaherty and O'Malley [12] and Steele [29].

Equation (1.3) is the same as that for the transverse deflection of a classical Euler-Bernoulli elastic beam that is subjected to a tensile force proportional to ρ^2; hence, the name spline in tension.

In §2 of this paper, we construct a basis for the tension splines and use it to obtain the collocation equations. In §3, we obtain asymptotic approximations to the solution of (1.1, 2) in the two special cases when $|p(x)| \geqq \bar{p} > 0$ and when $p(x) \equiv 0$ on $[a, b]$, and use these to select tension parameters. In §4, we discuss the selection of collocation points that are in some sense optimal and present some formal error estimates in regions not containing boundary layers. This section is somewhat technical and may be skipped by readers who are primarily interested in the algorithm. In §5, we apply our methods to some examples; and in §6, we discuss the results.

Not surprisingly, the results indicate that tension splines provide better approximations within boundary layers and polynomials provide better approximations elsewhere. This suggests the possibility of applying tension only within boundary layers. This would require either an a priori knowledge of the location of the boundary layers, or an automatic procedure for finding them. A tentative procedure for automatically locating boundary layers is presented in §5.

We anticipate that our methods would be useful on other problems, such as initial-boundary value problems for parabolic partial differential equations involving diffusion, convection and/or reaction.

2. Collocation equations and the tension spline basis. In the usual method of collocation (cf. Ascher, Christiansen, and Russell [2], de Boor and Swartz [8], Russell [24], or Russell and Shampine [26]), one introduces a partition

$$\Delta_N \equiv \{a = x_0 < x_1 < \cdots < x_N = b\} \tag{2.1}$$

of $[a, b]$ into N subintervals, and approximates the solution of (1.1, 2) by piecewise polynomials $y_h(x) \in M(\Delta_N, k, m)$, where

$$M(\Delta_N, k, m) \equiv \{w \in C^m[a, b] \mid w_{\text{restr. to } I_i} \in P_k(I_i);\ i = 1, 2, \cdots, N\}. \tag{2.2}$$

Here

$$I_i \equiv (x_{i-1}, x_i), \tag{2.3}$$

and $P_k(E)$ denotes the class of polynomials having at most degree k on E. The dimension of $M(\Delta_N, k, m)$ is $N(k-m)+m+1$, and one determines $y_h(x)$ by collocat-

ing at $N(k-m)+m-1$ points $z_i \in [a, b]$, i.e., by enforcing

$$Ly_h(z_i) = f(z_i), \qquad i = 1,2,\cdots, N(k-m)+m-1, \tag{2.4}$$

and by requiring y_h to satisfy the boundary conditions (1.2). Convergence and the order of accuracy of these methods depend on m, k, Δ_N, and the choice of z_i and are discussed in, e.g., [8]. However, Ly_h should exist, and necessary continuity conditions on $M(\Delta_N, k, m)$ are $k \geqq 2$ and $m \geqq 1$. In addition, de Boor and Swartz [8] show that the maximal order of convergence in the largest subinterval length is achieved by selecting the collocation points as an appropriate number of Gauss-Legendre points on each subinterval.

Unfortunately, collocation at the Gauss-Legendre points with piecewise polynomials is known to behave rather poorly on singularly-perturbed problems for any partition where ε is much smaller than the minimum subinterval length (cf. Hemker [15] and our Example 1 in §5). It would be overly restrictive and in most cases impractical to require a partition with subinterval lengths of order ε, and we seek to avoid this situation by changing the locations of the collocation points and/or adding exponential functions to $M(\Delta_N, k, m)$. Thus, we also consider approximations $y_h(x) \in E(\Delta_N, k, m, \boldsymbol{\rho})$, where

$$E(\Delta_N, k, m, \boldsymbol{\rho}) \equiv \{ w \in C^m[a, b] \mid w_{\text{restr. to } I_i} \in \operatorname{span}(P_k(I_i), e^{\rho_i x/h_i}, e^{-\rho_i x/h_i}), \quad i = 1,2,\cdots, N \} \tag{2.5}$$

and

$$h_i = x_i - x_{i-1}, \qquad i = 1,2,\cdots, N. \tag{2.6}$$

The quantities ρ_i, $i = 1,2,\cdots, N$, are called tension parameters and will subsequently be selected to approximate the rapidly varying part of the exact solution of (1.1, 2).

In this paper, we have confined our attention to collocation with piecewise cubic polynomials belonging to $M(\Delta_N, 3, 1)$ and splines in tension belonging to $E(\Delta_N, 1, 1, \boldsymbol{\rho})$, which are both spaces of dimension $2(N+1)$. For reasons of computational convenience and efficiency, it is usual to construct a basis for $M(\Delta_N, 3, 1)$ that satisfies the $C^1[a,b]$ continuity requirement, and where each element has support extending over only two subintervals. We proceed similarly for $E(\Delta_N, 1, 1, \boldsymbol{\rho})$; thus, we write $y_h(x)$ in the form

$$y_h(x) = \sum_{i=0}^{N} [c_i \tau_i(x, \boldsymbol{\rho}) + d_i \sigma_i(x, \boldsymbol{\rho})], \tag{2.7}$$

where the basis $\tau_i(x, \boldsymbol{\rho})$, $\sigma_i(x, \boldsymbol{\rho})$, $i = 0, 1, \cdots, N$, is defined in terms of the "canonical basis elements" η_0 and η_1 as follows:

$$\tau_i(x, \boldsymbol{\rho}) = \begin{cases} \eta_0\left(\dfrac{x_i - x}{h_i}, \rho_i\right), & x \in [x_{i-1}, x_i], \\ \eta_0\left(\dfrac{x - x_i}{h_{i+1}}, \rho_{i+1}\right), & x \in [x_i, x_{i+1}], \\ 0, & \text{otherwise}; \end{cases} \tag{2.8}$$

and

$$\sigma_i(x,\rho)=\begin{cases} -h_i\eta_1\left(\dfrac{x_i-x}{h_i},\rho_i\right), & x\in[x_{i-1},x_i], \\ h_{i+1}\eta_1\left(\dfrac{x-x_i}{h_{i+1}},\rho_{i+1}\right), & x\in[x_i,x_{i+1}], \\ 0, & \text{otherwise.}\end{cases} \tag{2.9}$$

The functions $\eta_0(s,\rho)$ and $\eta_1(s,\rho)$ are defined on $0\leqq s\leqq 1$ as

$$\eta_0(s,\rho)=1-s+K_0\left[2s-1-\frac{\sinh\rho/2(2s-1)}{\sinh\rho/2}\right], \tag{2.10}$$

$$\eta_1(s,\rho)=\frac{1}{2}K_0\left[2s-1-\frac{\sinh\rho/2(2s-1)}{\sinh\rho/2}\right] + \frac{1}{2}K_1\left[1-\frac{\cosh\rho/2(2s-1)}{\cosh\rho/2}\right], \tag{2.11}$$

where

$$K_0=1/\rho\omega(\rho/2),\quad K_1=(1/\rho)\coth\rho/2,\quad \omega(z)=\coth z-1/z. \tag{2.12}$$

Observe that η_0 and η_1 each satisfy the differential equation

$$(\eta_k''-\rho^2\eta_k)''=0,\qquad 0\leqq s\leqq 1,\quad k=0,1, \tag{2.13}$$

subject to the boundary conditions

$$\eta_k(0,\rho)=\delta_{0k},\quad \eta_k(1,\rho)=0,\quad \eta_k'(0,\rho)=\delta_{1k},\quad \eta_k'(1,\rho)=0,\quad k=0,1, \tag{2.14}$$

where δ_{ik} denotes the Kronecker delta and, in this context, $(\cdot)'$ denotes differentiation with respect to s. Using (2.14) and (2.7-9), we see that $c_i=y_h(x_i)$ and $d_i=dy_h(x_i)/dx$, $i=0,1,\cdots,N$.

As ρ tends to zero η_0 and η_1 approach the usual canonical basis elements for $M(\Delta_N,3,1)$, i.e., that of a cubic Hermite interpolating polynomial on $0\leqq s\leqq 1$. Thus,

$$\eta_0(s,\rho)=(1-s)^2(1+2s)+O(\rho^2),\qquad \eta_1(s,\rho)=s(1-s)^2+O(\rho^2). \tag{2.15}$$

For large values of ρ, η_0 and η_1 become

$$\begin{aligned}\eta_0(s,\rho)&=1-s-\frac{1}{\rho-2}\left[2s-1+e^{-\rho s}-e^{-\rho(1-s)}\right]+O(e^{-\rho}/\rho),\\ \eta_1(s,\rho)&=\frac{\rho-1}{\rho(\rho-2)}\left[1-s-e^{-\rho s}\right]-\frac{1}{\rho(\rho-2)}\left[s-e^{-\rho(1-s)}\right]+O(e^{-\rho}/\rho).\end{aligned} \tag{2.16}$$

Thus, in the interior of $(0,1)$, η_0 and η_1 are asymptotically given by the linear functions

$$\eta_0(s,\rho)\sim(1-s)-(2s-1)/(\rho-2),\qquad \eta_1(s,\rho)\sim(1-s)/\rho-(2s-1)/\rho(\rho-2).$$

Both η_0 and η_1 converge uniformly on $0\leqq s\leqq 1$ as $\rho\to\infty$ to $1-s$ and 0, respectively; however, their derivatives exhibit boundary layer behavior at $s=0$ and 1. Since τ_i and

σ_i (via. (2.8, 9)) behave similarly at the knots x_{i-1}, x_i, x_{i+1}, the numerical approximation y_h may have internal boundary layer jumps of height $O(1/\rho)$ even when the exact solution is smooth. We demonstrate this phenomena in Example 1 of §5. The functions η_0 and η_1 are plotted for a small and a large value of ρ in Fig. 1.

A discrete system for determining c_i, d_i, $i=0,1,\cdots,N$ is obtained for a given set of tension parameters ρ_i, $i=1,2,\cdots,N$ by collocating at $2N$ points z_i, $i=1,2,\cdots,N$ on $[a,b]$ and by satisfying the boundary conditions (1.2). For simplicity we place two collocation points symmetrically disposed on each subinterval, i.e.,

$$z_{2i-1}=x_{i-1}+t_ih_i, \qquad z_{2i}=x_{i-1}+(1-t_i)h_i, \qquad i=1,2,\cdots,N, \tag{2.17}$$

for an appropriate choice of $t_i\in[0,1/2)$. Then using (2.17), (2.7-9), and (1.1) in (2.4), we find the discrete system on I_i to be

$$\begin{bmatrix} l_{i1}(t_i) & l_{i2}(t_i) & l_{i3}(t_i) & l_{i4}(t_i) \\ l_{i1}(1-t_i) & l_{i2}(1-t_i) & l_{i3}(1-t_i) & l_{i4}(1-t_i) \end{bmatrix} \begin{bmatrix} c_{i-1} \\ d_{i-1} \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \hat{f}_i(t_i) \\ \hat{f}_i(1-t_i) \end{bmatrix}, \tag{2.18}$$

$$i=1,2,\cdots,N,$$

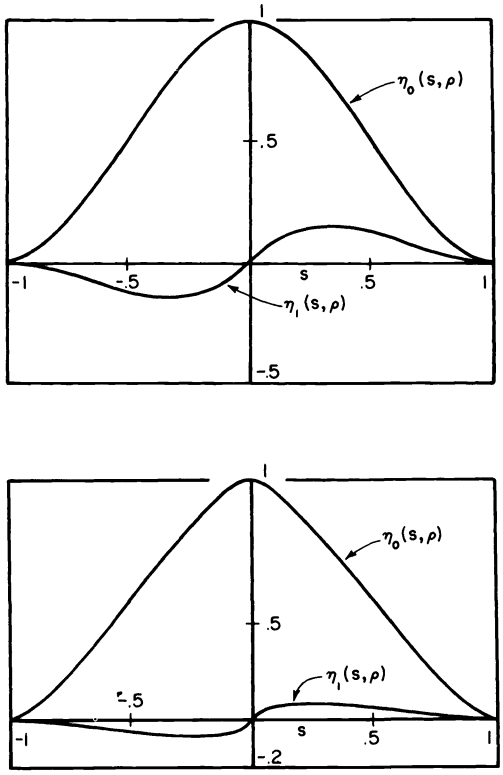


FIG 1. *Canonical basis functions* $\eta_0(s,\rho)$ *and* $\eta_1(s,\rho)$ *on* $-1\leqq s\leqq 1$ *for* $\rho=0.01$ (*top*) *and* $\rho=10$ (*bottom*).

where

$$l_{i1}(t)=(\varepsilon/h_i^2)\eta_0''(t,\rho_i)+(\hat{p}_i(t)/h_i)\eta_0'(t,\rho_i)+\hat{q}_i(t)\eta_0(t,\rho_i),$$

$$l_{i2}(t)=(\varepsilon/h_i)\eta_1''(t,\rho_i)+\hat{p}_i(t)\eta_1'(t,\rho_i)+\hat{q}_i(t)h_i\eta_1(t,\rho_i),$$

$$l_{i3}(t)=(\varepsilon/h_i^2)\eta_0''(1-t,\rho_i)-(\hat{p}_i(t)/h_i)\eta_0'(1-t,\rho_i)+\hat{q}_i(t)\eta_0(1-t,\rho_i), \tag{2.19}$$

$$l_{i4}(t)=-\left[(\varepsilon/h_i)\eta_1''(1-t,\rho_i)-\hat{p}_i(t)\eta_1'(1-t,\rho_i)+\hat{q}_i(t)\eta_1(1-t,\rho_i)\right],$$

and $\hat{f}_i(t)\equiv f(x_{i-1}+th_i)$, etc. Substituting (2.7-9) into (1.2) gives the boundary conditions

$$\alpha_{11}c_0+\alpha_{12}d_0=A, \qquad \alpha_{21}c_N+\alpha_{22}d_N=B. \tag{2.20}$$

Thus, the $2(N+1)$-dimensional discrete system (2.18-20) has the following structural form:

$$\begin{bmatrix} x & x & & & & & & & & & & \\ x & x & x & x & & & & & & & & \\ x & x & x & x & & & & & & & & \\ & & x & x & x & x & & & & & & \\ & & x & x & x & x & & & & & & \\ & & & \cdots & & & & & & & & \\ & & & & & & x & x & x & x & & \\ & & & & & & x & x & x & x & & \\ & & & & & & & & x & x & & \end{bmatrix}\begin{bmatrix} c_0 \\ d_0 \\ c_1 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ c_N \\ d_N \end{bmatrix}=\begin{bmatrix} A \\ \hat{f}_1(t_1) \\ \hat{f}_1(1-t_1) \\ \cdot \\ \cdot \\ \cdot \\ \hat{f}_N(t_N) \\ \hat{f}_N(1-t_N) \\ B \end{bmatrix}, \tag{2.21}$$

where each x denotes a nonzero entry. We solve (2.18-20) for prescribed values of ρ_i and t_i, $i=1,2,\cdots,N$, by an alternating row and column pivoting algorithm due to Varah [30]. This procedure is numerically stable and requires no storage additional to that needed for the nonzero entries in (2.21).

3. Asymptotic solutions, Green's functions, and the selection of tension parameters. In this section, we present asymptotic approximations of the solution of (1.1, 2) and of its Green's function. They will be used to select the tension parameters, and in §4 to select collocation points. We shall not attempt to do this in all generality, but rather by considering two special cases of the problem

$$Ly\equiv\varepsilon y''+p(x)y'+q(x)y=f(x), \qquad a\leqq x\leqq b, \tag{3.1}$$

$$y(a,\varepsilon)=A, \qquad y(b,\varepsilon)=B, \tag{3.2}$$

when (Problem 1) $|p(x)|\geqq\bar{p}>0$ and (Problem 2) when $p(x)\equiv 0$, $q(x)\leqq\bar{q}<0$ for $x\in[a,b]$. In either case, any boundary layers are at the ends of $[a,b]$; thus, there are no turning points and no interior nonuniformities. We consider such problems among the examples of §5.

The Green's function $G(x,\xi)$ associated with the operator L and homogeneous boundary conditions (3.2) on the interval $[a, b]$ satisfies

$$L^*G(x,\xi)\equiv\varepsilon G_{\xi\xi}-(p(\xi)G)_\xi+q(\xi)G=0,(a,x)\cup(x,b), \tag{3.3}$$

$$G(x,a)=G(x,b)=0, \tag{3.4a}$$

$$G(x,x^+)-G(x,x^-)=0, \tag{3.4b}$$

$$G_\xi(x,x^+)-G_\xi(x,x^-)=1/\varepsilon, \tag{3.4c}$$

where the subscript ξ denotes partial differentiation.

We use the WKB method to construct our asymptotic approximations of the solutions of (3.1, 2) and (3.3, 4). Since the details of this method are well known (cf. Wasow [31]), we only present the results and omit their development.

3.1. Problem 1: $|p(x)|\geqq\bar{p}>0$ for $x\in[a,b]$. We consider the case when $p(x)>0$ on $[a, b]$. The case when $p(x)<0$ is handled in an analogous manner. Using the WKB method, we find the following $O(\varepsilon)$ approximations to the two fundamental solutions of (3.1) (cf. Hemker [15]):

$$Y(x,\xi)=\exp\left\{\int_x^\xi q(z)/p(z)dz\right\}, \tag{3.5a}$$

$$\Pi(\xi,x)=\exp\left\{-\int_\xi^x(p(z)/\varepsilon-q(z)/p(z))dz\right\}. \tag{3.5b}$$

The solution of (3.1) satisfying the boundary conditions (3.2) is given by

$$y(x,\varepsilon)=Y_R(x)+[A-Y_R(a)]\Pi(a,x)+O(\varepsilon), \tag{3.6a}$$

where

$$Y_R(x)=BY(x,b)-\int_x^b(f(z)/p(z))Y(x,z)dz. \tag{3.6b}$$

The term $[A-Y_R(a)]\Pi(a,x)$ is exponentially small outside of a boundary layer of width $O(\varepsilon)$ near $x=a$. For $a<x\leqq b$ $y(x,\varepsilon)\sim Y_R(x)$, where $Y_R(x)$ is the solution of the reduced problem

$$p(x)Y_R'(x)+q(x)Y_R(x)=f(x),\qquad a\leqq x\leqq b, \tag{3.7a}$$

$$Y_R(b)=B, \tag{3.7b}$$

obtained by neglecting the $\varepsilon y''$ term in (3.1) and the boundary condition (3.2) at $x=a$. The problem with $p(x)<0$ has a solution with a boundary layer near $x=b$ and a reduced solution satisfying (3.7a) subject to the initial condition $Y_R(a)=A$.

In a similar manner, an $O(\varepsilon)$ approximation to $G(x,\xi)$ satisfying (3.3, 4) is found as

$$G(x,\xi)=\alpha(x)\{\Pi(\xi,b)Y^*(a,b)[\Pi(a,x)-Y^*(x,a)]$$

$$-\Pi(a,x)Y^*(a,\xi)+\left\{\begin{array}{l}\Pi(\xi,x),\ a\leqq\xi\leqq x\\ Y^*(x,\xi),\ x\leqq\xi\leqq b\end{array}\right\}+O(\varepsilon), \tag{3.8a}$$

where

$$\alpha(x) = -p(x)/\left[\, p^2(x) - 2\varepsilon q(x) + \varepsilon p'(x)\,\right], \tag{3.8b}$$

$$Y^*(x,\xi) = \exp\left\{ \int_x^{\xi} (q(z) - p'(z))/p(z)\,dz \right\}, \tag{3.8c}$$

and $\Pi(\xi, x)$ is as in (3.5b). As a function of ξ, $G(x,\xi)$ has boundary layers at $\xi = b^-$ and $\xi = x^-$.

3.2. Problem 2: $p(x) \equiv 0$, $q(x) \leqq \bar{q} < 0$ for $x \in [a,b]$. In this case, the WKB method gives the following $O(\sqrt{\varepsilon}\,)$ approximations to the two fundamental solutions of (3.1) (cf. Hemker [15]).

$$\Pi_1(x,\xi) = q(x)^{-1/4}\Pi(x,\xi), \tag{3.9a}$$

$$\Pi_2(x,\xi) = q(x)^{-1/4}\Pi(\xi,x), \tag{3.9b}$$

where for this problem

$$\Pi(\xi,x) = \exp\left\{ -\int_{\xi}^{x} \sqrt{-q(z)/\varepsilon}\; dz \right\}. \tag{3.9c}$$

The character of the solution depends critically on the sign of $q(x)$. When $q(x) < 0$, the solution is exponential; and when $q(x) > 0$, the solution oscillates rapidly with period $2\pi\sqrt{\varepsilon/q(x)}$. We would not expect tension spline approximations to be useful in the oscillatory case and, thus, we confine our attention to problems with $q(x) < 0$ on $[a, b]$. The use of "splines under compression" for oscillatory problems is currently under investigation by Coyle and Flaherty [7].

Using (3.9), the solution of (3.1, 2) is given as

$$\begin{aligned} y(x,\varepsilon) = {} & \left[\, A - f(a)/q(a)\,\right]\left[\, q(a)/q(x)\,\right]^{1/4}\Pi(a,x) \\ & + \left[\, B - f(b)/q(b)\,\right]\left[\, q(b)/q(x)\,\right]^{1/4}\Pi(x,b) + f(x)/q(x) + O(\sqrt{\varepsilon}\,). \end{aligned} \tag{3.10}$$

Outside of the boundary layers, which extend over $O(\sqrt{\varepsilon}\,)$ neighborhoods of $x = a$ and b, the solution $y(x,\varepsilon) \sim Y_R(x)$, where $Y_R(x)$ is the solution of the reduced problem

$$q(x)Y_R(x) = f(x),$$

obtained by neglecting the $\varepsilon y''$ term in (3.1) and both boundary conditions (3.2).

Problem 2 is self-adjoint, so the WKB approximations (3.9) can also be used to construct the following $O(1)$ approximation to $G(x,\xi)$ satisfying (3.3, 4):

$$\begin{aligned} G(x,\xi) = {} & \frac{1}{2}\left[\, \varepsilon^2 q(x)q(\xi)\,\right]^{-1/4}\{\Pi(a,x)\Pi(a,\xi) + \Pi(x,b)\Pi(\xi,b) \\ & - \begin{cases} \Pi(\xi,x), & a \leqq \xi \leqq x \\ \Pi(x,\xi), & x \leqq \xi \leqq b \end{cases} + O(\sqrt{\varepsilon}\,)\}. \end{aligned} \tag{3.11}$$

As a function of ξ, $G(x,\xi)$ has boundary layers on both sides of $\xi = x$, and is unbounded as $O(1/\sqrt{\varepsilon}\,)$ as $\varepsilon \to 0$.

3.3. Selection of tension parameters. We want the tension parameters to approximate the rapidly decaying solutions (3.5b) or (3.9) of Problems 1 or 2, respec-

tively, and so we choose ρ_i on the subinterval I_i as

(3.12a)

$$\rho_i = h_i \begin{cases} |p(x_k)/\varepsilon - q(x_k)/p(x_k)| & \text{if } |p(x_{i-1})+p(x_i)|/\varepsilon \geqq |q(x_{i-1})+q(x_i)|, \\ \sqrt{-[q(x_{i-1})+q(x_i)]/2\varepsilon} & \text{if } |p(x_{i-1})+p(x_i)|/\varepsilon < |q(x_{i-1})+q(x_i)|, \end{cases}$$

$$i = 1, 2, \cdots, N,$$

where

(3.12b)
$$k = \begin{cases} i-1 & \text{if } [p(x_{i-1})+p(x_i)]/\varepsilon > 0, \\ i & \text{if } [p(x_{i-1})+p(x_i)]/\varepsilon < 0. \end{cases}$$

Thus, for Problem 1 with $p(x)>0$ on $[a, b]$

(3.13a)
$$\rho_i = h_i |p(x_{i-1})/\varepsilon - q(x_{i-1})/p(x_{i-1})|,$$

and for Problem 2

(3.13b)
$$\rho_i = h_i \sqrt{-[q(x_{i-1})+q(x_i)]/2\varepsilon}\,.$$

However, we use (3.12) computationally even when the conditions of Problem 1 or 2 are not satisfied, e.g., when there are turning points.

The solution of the collocation equations (2.18-20) with the tension parameters specified by (3.12) will give the exact solution of (1.1, 2), for any choice of $t_i \in [0, 1/2)$, whenever $f(x)$ is a linear polynomial and either 1) $p(x)\equiv\bar{p}$ and $q(x)\equiv 0$, or 2) $p(x)\equiv 0$ and $q(x)\equiv\bar{q}<0$. This is because the solutions of these problems are elements of the approximating space $E(\Delta_N, 1, 1, \boldsymbol{\rho})$.

We close this section by applying the method of collocation with splines under tension to the example

(3.14)
$$Ly \equiv \varepsilon y'' + p(x)y' = f(x), \qquad a \leqq x \leqq b, \quad y(a)=A, \quad y(b)=B,$$

with $p(x)>0$ on $[a, b]$. Using (3.5, 6) the solution of this problem is

(3.15a)
$$y(x, \varepsilon) = Y_R(x) + [A - Y_R(a)] \exp\left\{ -\int_a^x (p(z)/\varepsilon)\,dz \right\} + O(\varepsilon),$$

where

(3.15b)
$$Y_R(x) = B - \int_x^b (f(z)/p(z))\,dz.$$

For the present, we choose $t_i = 0$, and then use (2.10-12) and (2.19) in (2.18) to obtain the discrete system

(3.16)
$$(\varepsilon\rho_i/h_i)\left[\left(\frac{\nabla c_i}{h_i} - \frac{1}{2}\mu d_i\right)\omega^{-1}(\rho_i/2) + \frac{1}{2}\nabla d_i \coth \rho_i/2\right] + p(x_{i-1})d_{i-1} = f(x_{i-1}),$$

$$(\varepsilon\rho_i/h_i)\left[\left(\frac{\nabla c_i}{h_i} - \frac{1}{2}\mu d_i\right)\omega^{-1}(\rho_i/2) - \frac{1}{2}\nabla d_i \coth \rho_i/2\right] + p(x_i)d_i = f(x_i),$$

$$i = 1, 2, \cdots, N, \quad c_0 = A, \quad c_N = B,$$

where

$$\nabla(\cdot)_i \equiv (\cdot)_i - (\cdot)_{i-1}, \qquad \mu(\cdot)_i \equiv (\cdot)_i + (\cdot)_{i-1}, \tag{3.17}$$

and $\omega(z)$ is defined by (2.12). Using (3.13a) we select $\rho_i = h_i p(x_{i-1})/\varepsilon$, and assume that the partition has been chosen so that $\rho_i \gg 1$, $i = 1,2,\cdots,N$. In fact, suppose that ρ_i is large enough to approximate $\omega(\rho_i/2)$ and $\coth \rho_i/2$ by $(1-2/\rho_i)$ and 1, respectively. Then (3.16) become

$$\begin{aligned} &(\varepsilon/h_i)(2\nabla c_i/h_i - \mu d_i) + p(x_{i-1})\nabla c_i/h_i = f(x_{i-1}), \\ &(\mu p(x_i))d_i = \mu f(x_i), \qquad i = 1,2,\cdots,N, \quad c_0 = A, \quad c_N = B. \end{aligned} \tag{3.18}$$

Thus, the solution is approximately determined as the solution of

$$c_N = B, \tag{3.19a}$$

$$p(x_{i-1})\nabla c_i/h_i = f(x_{i-1}) + O(\varepsilon/h_i), \qquad i = N, N-1,\cdots,2, \tag{3.19b}$$

$$(\mu p(x_i))d_i = \mu f(x_i), \qquad i = N, N-1,\cdots,1, \tag{3.19c}$$

$$c_0 = A, \tag{3.19d}$$

$$d_0 = -(p(a)/\varepsilon)[c_0 - c_1 + h_1 f(a)/p(a) + O(\varepsilon/h_1)]. \tag{3.19e}$$

Equations (3.19a, b, c) can be recognized as $O(h)$ (where h is the maximum subinterval length) "upwind" difference approximations to the reduced problem, while (3.19e) gives the initial slope of the solution in the boundary layer correct to $O(h/\varepsilon)$.

4. Selection of collocation points. Our aim in this section is to suggest some special choices of collocation points that may be used to reduce the errors in methods for Problems 1 and 2. We confine our attention to the two limiting cases of zero tension ($\rho_i \equiv 0$) and large tension ($\rho_i \gg 1$), for $i = 1,2,\cdots,N$. For simplicity, we consider uniform partitions with $h = h_i$, $i = 1,2,\cdots,N$, and assume that $hp(x)/\varepsilon \gg 1$ for Problem 1 and $h\sqrt{-q(x)/\varepsilon} \gg 1$ for Problem 2. No detailed rigorous error analysis will be given; however, some formal error estimates are obtained on subintervals not containing boundary layers.

4.1. Error formulas. It is well known (cf. [8], [24], or [25]) that the pointwise error in collocation methods for (1.1, 2) satisfies an equation of the form

$$e^{(k)}(x) \equiv y^{(k)}(x) - y_h^{(k)}(x) = \int_a^b \frac{\partial^k}{\partial x^k} G(x,\xi) r(\xi)\, d\xi, \qquad k = 0,1, \tag{4.1}$$

where the residual

$$r(\xi) = Ly(\xi) - Ly_h(\xi) = f(\xi) - Ly_h(\xi). \tag{4.2}$$

It is convenient to introduce the local transformation

$$\xi = x_{i-1} + hs, \qquad 0 \leqq s \leqq 1, \tag{4.3}$$

on the subinterval I_i and, as in §2, let $\hat{f}_i(s) \equiv f(x_{i-1} + hs)$, etc. Then (4.1) becomes

$$e^{(k)}(x) = h \sum_{i=1}^{N} \int_0^1 \frac{\partial^k}{\partial x^k} G(x, x_{i-1} + hs)\hat{r}_i(s)\, ds, \qquad k = 0,1, \tag{4.4a}$$

where

$$\hat{r}_i(s)=\hat{f}_i(s)-\hat{L}_i\hat{y}_{h_i}(s). \tag{4.4b}$$

Let $P\hat{f}_i(s)$ be the linear interpolant

$$P\hat{f}_i(s)=\left[(1-t_i-s)\hat{f}_i(t_i)+(s-t_i)\hat{f}_i(1-t_i)\right]/(1-2t_i) \tag{4.5}$$

to $\hat{f}_i(s)$ at the two collocation points $s=t_i$ and $s=1-t_i$ on I_i. Since the collocation equations (2.4, 17) imply $\hat{L}_i\hat{y}_{h_i}=\hat{f}_i$ at $s=t_i$, $1-t_i$, we have $P\hat{L}_i\hat{y}_{h_i}=P\hat{f}_i$, and (4.4a) may be written as

$$e^{(k)}(x)=h\sum_{i=1}^{N}\int_0^1\frac{\partial^k}{\partial x^k}G(x,x_{i-1}+hs)(1-P)\hat{r}_i(s)ds,\qquad k=0,1. \tag{4.6}$$

The interpolation error

$$(1-P)\hat{r}_i(s)=(s-t_i)(s-1-t_i)\hat{r}_i[t_i,1-t_i,s], \tag{4.7}$$

where $\hat{r}_i[s_0,s_1,\cdots,s_k]$ denotes the kth divided difference of $\hat{r}_i$ at the points $s_0,s_1,\cdots,s_k$. This form of $(1-P)\hat{r}_i(s)$ suffices when $\rho_i\equiv 0$; however, when $\rho_i\gg 1$ a more detailed form is needed. In this case, we assume that ρ_i is large enough to neglect terms of $O(e^{-\rho_i/2})$ relative to unity and use the large tension approximations (2.16) as well as (2.7-9), (4.4b), and (4.5) to get

(4.8)

$$\begin{aligned}(1-P)\hat{r}_i(s)=(s-t_i)(s-1+t_i)&\left\{\hat{f}_i[t_i,1-t_i,s]-\hat{L}_i\hat{w}_i(s)[t_i,1-t_i,s]\right\}\\
&-\frac{\rho_i}{\rho_i-2}\Bigg\{\beta_i\hat{u}_i(s)g(s,t_i)-\gamma_i\hat{v}_i(s)g(1-s,t_i)+(s-t_i)(s-1+t_i)\\
&\cdot\frac{e^{-\rho_i t_i}-e^{-\rho_i(1-t_i)}}{1-2t_i}\left(\beta_i\hat{u}_i[1-t_i,s]+\gamma_i\hat{v}_i[1-t_i,s]\right)-(s-t_i)\\
&\cdot(s-1+t_i)\left(e^{-\rho_i t_i}\beta_i\hat{u}_i[t_i,1-t_i,s]+e^{-\rho_i(1-t_i)}\gamma_i\hat{v}_i[t_i,1-t_i,s]\right)\Bigg\},\end{aligned}$$

where

$$\beta_i=\nabla c_i/h-d_{i-1}-(\nabla d_i)/\rho_i,\qquad \gamma_i=\nabla c_i/h-d_i+(\nabla d_i)/\rho_i, \tag{4.9}$$

$$\hat{u}_i(s)=\varepsilon\rho_i/h-\hat{p}_i(s)+\hat{q}_i(s)h/\rho_i,\qquad \hat{v}_i(s)=\varepsilon\rho_i/h+\hat{p}_i(s)+\hat{q}_i(s)h/\rho_i, \tag{4.10}$$

$$g(s,t_i)=\left[(1-2t_i)e^{-\rho_i s}+(s-1+t_i)e^{-\rho_i t_i}-(s-t_i)e^{-\rho_i(1-t_i)}\right]/(1-2t_i), \tag{4.11}$$

and $\hat{w}_i(s)$ is the linear polynomial part of $\hat{y}_{h_i}(s)$.

The choice of ρ_i given by (3.12) makes $\hat{u}_i(s)=O(h)$ when $\hat{p}_i(s)>0$ on I_i, $\hat{v}_i(s)=O(h)$ when $\hat{p}_i(s)<0$ on I_i, and $\hat{u}_i(s)=\hat{v}_i(s)=O(h^2/\rho_i)$ when $\hat{p}_i(s)\equiv 0$ on I_i. The assumption that terms of $O(e^{-\rho_i/2})$ are negligible will specifically allow us to drop all terms in (4.8, 11) involving the factor $e^{-\rho_i(1-t_i)}$, since $t_i<\frac{1}{2}$. This will be done in all further uses of (4.8, 11) except where noted.

It remains to use the formulas (4.7) or (4.8) for $(1-P)\hat{r}_i(s)$ together with the approximations (3.8) or (3.11) for the Green's functions in (4.6), and to find appropriate choices for collocation points. One problem is that the errors given by (4.6) depend on the unknown numerical solution y_h. This would not be a serious difficulty

if y_h and y_h' were bounded as $\varepsilon \to 0$ for fixed h. We have shown by example in §3 that c_i and d_i (hence, y_h and y_h') are bounded away from the boundary layer region for $\varepsilon/h \ll 1$ when $t_i = 0$ and ρ_i is selected according to (3.12). It is reasonable to assume that this remains so when t_i is sufficiently small; however, it is also relatively easy to show that d_i can be unbounded at every knot point as $\varepsilon/h \to 0$ when $\rho_i = 0$ and t_i are the Gauss-Legendre points. Little is known about the behavior of the numerical solution for other choices of t_i and ρ_i. In this paper, we shall not attempt to find conditions for y_h to be bounded as $\varepsilon \to 0$, but rather we shall make some suggestions for collocation points that should generally reduce the error in methods for Problems 1 and 2. We note in passing that if y_h and y_h' were bounded, arguments similar to those used by Pruess [21] or Russell and Christiansen [25] on related non singularly-perturbed problems could be used to remove the dependence on y_h from the leading order terms in $e(x)$.

4.2. Collocation points for Problem 1. We again consider the case when $p(x) \geqq \bar{p} > 0$ for $x \in [a, b]$. Let x be a knot point, say x_j, so that there are no discontinuities in derivatives of the Green's function on any subinterval, and apply the transformation (4.3) to (3.5b) and (3.8c) to get

$$\Pi(x_{i-1}+hs, x_j) = \Pi(x_i, x_j)\hat{\pi}_i(s) = \delta_{ij}\hat{\pi}_i(s), \qquad i \leqq j, \tag{4.12a}$$

$$Y^*(x_j, x_{i-1}+hs) = Y^*(x_j, x_{i-1})\hat{\zeta}_i(s), \qquad i > j, \tag{4.12b}$$

where

$$\hat{\pi}_i(s) = \exp\left\{ -h\int_s^1 \left[\hat{p}_i(z)/\varepsilon - \hat{q}_i(z)/\hat{p}_i(z) \right] dz \right\}, \tag{4.12c}$$

$$\hat{\zeta}_i(s) = \exp\left\{ h\int_0^s \left[(\hat{q}_i(z) - \hat{p}_i'(z))/\hat{p}_i(z) \right] dz \right\}. \tag{4.12d}$$

The last form of (4.12a) follows from our assumption that terms of $O(e^{-\rho_i/2})$ are negligible; hence, the boundary layer in $\Pi(x_i, x_j)$ at x_j is well within subinterval I_j.

Using (3.8a) and (4.12) in (4.6) we have

$$\begin{aligned} e(x_j) = {} & h\left[\tilde{Y}^*(a, b)\delta_{j0} - \tilde{Y}^*(x_j, b) \right] S_N - h\delta_{j0} \sum_{i=1}^{N} \tilde{Y}^*(a, x_{i-1}) R_i \\ & + h\alpha(x_j) S_j + h \sum_{i=j+1}^{N} \tilde{Y}^*(x_j, x_{i-1}) R_i, \end{aligned} \tag{4.13a}$$

$$\begin{aligned} e'(x_j) = {} & -h\left\{ \tilde{Y}^*(a, b)\left[\rho_j^*/h - \alpha'(x_j)/\alpha(x_j) \right]\delta_{j0} + \tilde{Y}_x^*(x_j, b) \right\} S_N \\ & + h\left[\rho_j^*/h - \alpha'(x_j)/\alpha(x_j) \right]\delta_{j0} \sum_{i=1}^{N} \tilde{Y}^*(a, x_{i-1}) R_i \\ & - h\alpha(x_j)\left[\rho_j^*/h - \alpha'(x_j)/\alpha(x_j) \right] S_j + h \sum_{i=j+1}^{N} \tilde{Y}_x^*(x_j, x_{i-1}) R_i, \end{aligned} \tag{4.13b}$$

where

$$\rho_j^* \equiv h\left[\, p(x_j)/\varepsilon - q(x_j)/p(x_j)\,\right], \tag{4.14}$$

$$\tilde{Y}^*(x_j, x_i) \equiv \alpha(x_j) Y^*(x_j, x_i), \tag{4.15}$$

and

$$S_i = \int_0^1 \hat{\pi}_i(s)(1-P)\hat{r}_i(s)\,ds, \tag{4.16a}$$

$$R_i = \int_0^1 \hat{\xi}_i(s)(1-P)\hat{r}_i(s)\,ds. \tag{4.16b}$$

We call ρ_j^* the adjoint tension parameter, and note that $\rho_j^* = \rho_{j+1}$ when the tension parameters are selected according to (3.13a). Observe that ρ_j^* is large whenever $hp(x)/\varepsilon$ is large, and this can induce large errors in $e'(x_j)$ (cf. (4.13b)). These large errors can be confined to the boundary layer near $x=a$ if S_j is sufficiently small. However, in order for $e'(x_j)$ to be small within the boundary layer, R_i, $i=1,2,\cdots,N$, must also be sufficiently small. In this paper, we have concentrated on producing good approximations outside of boundary layer subintervals.

We first consider the tension spline approximation where $(1-P)\hat{r}_i(s)$ is given by (4.8). Expanding $f(x)$, $p(x)$, and $g(x)$ in Taylor's series about a suitable point on I_i would reveal that $\hat{f}_i[t_i, 1-t_i, s]$ and $\hat{u}_i[t_i, 1-t_i, s]$ are $O(h^2)$ while $\hat{u}_i[1-t_i, s]$ and $\hat{v}_i[1-t_i, s]$ are $O(h)$. As previously noted, the choice of ρ_i given by (3.13a) makes $u_i(s)$ of $O(h)$ and

$$\hat{v}_i(s) = \hat{p}_i(0) + \hat{p}_i(s) + O(h/\rho_i). \tag{4.17}$$

If we further assume that c_i, d_i/ρ_i, and $\nabla c_i/h$ are bounded, then $\hat{L}_i\hat{w}_i[t_i, 1-t_i, s]$ is $O(h^2)$, and to leading order (4.8) becomes

$$(1-P)\hat{r}_i(s) \approx \gamma_i \hat{v}_i(s) g(1-s, t_i). \tag{4.18}$$

This is not surprising since this term is due to the presence of the $e^{-\rho_i(1-s)}$ functions in the tension spline approximation and these functions are not present in the exact solution of Problem 1.

Substituting (4.18) into (4.16) gives

$$S_i \approx \gamma_i \int_0^1 \hat{\pi}_i(s) g(1-s, t_i)\hat{v}_i(s)\,ds, \tag{4.19a}$$

$$R_i \approx \gamma_i \int_0^1 \hat{\xi}_i(s) g(1-s, t_i)\hat{v}_i(s)\,ds. \tag{4.19b}$$

S_i may be further approximated by using Laplace's method (cf. Bender and Orszag [3, Chapt. 6]). The essential idea is that $\hat{\pi}_i(s)$ is exponentially small outside of a small neighborhood of $s=1$ when $h\hat{p}_i(s)/\varepsilon$ is large; hence, the integrand in (4.12c) may be replaced by its value at $s=1$. Using (4.14) this gives

$$S_i \approx \gamma_i \int_0^1 e^{-\rho_i^*(1-s)} g(1-s, t_i)\hat{v}_i(s)\,ds. \tag{4.20}$$

Now, we see that S_i may be reduced in magnitude by selecting t_i such that $e^{-\rho_i^*(1-s)}$ is

orthogonal to $g(1-s,t_i)$; i.e., using (4.11) we require

$$(4.21)\qquad \int_0^1 e^{-\rho_i^*(1-s)}g(1-s,t_i)ds=\frac{1}{\rho_i^*}\left[\frac{1}{1+\rho_i/\rho_i^*}-\frac{1-t_i-1/\rho_i^*}{1-2t_i}e^{-\rho_i t_i}\right]=0.$$

If terms of $O(1/\rho_i^*)$ are neglected, this implies

$$(4.22)\qquad t_i=(1/\rho_i)\ln(1+\rho_i/\rho_i^*),\qquad i=1,2,\cdots,N.$$

We refer to this choice of t_i as Method 1. It can only be used when ρ_i and ρ_i^* are such that $t_i<\frac{1}{2}$. When ρ_i and ρ_i^* are large, $t_i=O(1/\rho_i)$ and collocation is performed near the ends of each subinterval. Using (4.11), (4.17), and (4.22) in (4.19b) implies

$$(4.23)\qquad R_i\approx\gamma_i\hat{\xi}_i(1)\hat{v}_i(1)e^{-\rho_i t_i}\approx\gamma_i\hat{p}_i(1)\hat{\xi}_i(1).$$

If both c_i and d_i were bounded outside of the boundary layer, then γ_i would be $O(h)$ (cf. (4.9)). Thus, from (4.13) the best that we could expect from Method 1 is for $e(x_j)$ and $e'(x_j)$ to be $O(h)$. The computation evidence in §5 indicates that this is the case.

A second possibility is to select t_i so that R_i given by (4.19b) is reduced in magnitude. This can be done by requiring

$$\int_0^1 g(1-s,t_i)ds=0.$$

Using (4.11) this leads to

$$(4.24)\qquad t_i=\frac{1}{2}\left[1-\frac{2}{\rho_i}\cosh^{-1}\left(\frac{2}{\rho_i}\sinh\frac{\rho_i}{2}\right)\right],$$

and we refer to this choice of t_i as Method 2. We retained the $O(e^{-\rho_i(1-t_i)})$ term in (4.11) when obtaining (4.24), in order that t_i approach the Gauss-Legendre point (cf. (4.30)) as $\rho_i\to 0$. When ρ_i is large, $t_i\approx(1/\rho_i)\ln(\rho_i/2)$ and in this case (3.12), (4.15), and (4.20) imply that

$$(4.25)\qquad S_i\approx\gamma_i\hat{p}_i(1)/\rho_i^*.$$

The computational evidence in §5 indicates that S_i is not small enough to insure an accurate approximation of $e'(x_j)$ at any knot point. In fact, the indications are that $e(x_j)\approx O(h^2)$ while $e'(x_j)\approx O(h^2/\varepsilon)$ when ρ_i is large. Method 2 may still be used in this case if one is not interested in predicting the slope of the solution as long as ρ_i is not so large that it causes the discrete system (2.18-20) to be ill-conditioned. When ρ_i is small, t_i approaches the Gauss-Legendre point and $e(x_j)$ and $e'(x_j)$ will approach $O(h^4)$ (cf. de Boor and Swartz [8]).

For polynomial approximations, we use (4.7) in (4.16) to get

$$(4.26a)\qquad S_i\approx\int_0^1 e^{-\rho_i^*(1-s)}(s-t_i)(s-1+t_i)\hat{r}_i[t_i,1-t_i,s]ds,$$

$$(4.26b)\qquad R_i=\int_0^1\hat{\xi}_i(s)(s-t_i)(s-1+t_i)\hat{r}_i[t_i,1-t_i,s]ds,$$

where, once again, Laplace's method was used to approximate the singular integral S_i.

Either S_i or R_i may be reduced in magnitude by selecting t_i such that either $e^{-\rho_i^*(1-s)}$ or 1 is orthogonal to $(s-t_i)(s-1+t_i)$, respectively, i.e., by requiring either

$$(4.27a)\qquad \int_0^1 e^{-\rho_i^*(1-s)}(s-t_i)(s-1+t_i)ds=0$$

or

$$\int_0^1 (s-t_i)(s-1+t_i)\,ds=0. \tag{4.27b}$$

The option (4.27a) gives

$$t_i=\frac{2}{\rho_i^*}\,\frac{\omega(\rho_i^*/2)}{1+\sqrt{1-4\omega(\rho_i^*/2)/\rho_i^*}}, \tag{4.28}$$

where $\omega(z)$ was defined in (2.12). This method is referred to as Method 3. Once again, t_i approaches the Gauss-Legendre point as $\rho_i\to 0$ and $t_i\approx 1/\rho_i^*$ when ρ_i is large. Assuming that y_h and y_h' are bounded outside of the boundary layer region and using (4.4b) and Taylor's series expansions of $f(x)$ and $Ly_h(x)$ about x_i in (4.26b) leads to

$$R_i\approx -\frac{h^2}{12}\hat{\zeta}_i(1)\left[f(x_i)-Ly_h(x_i)\right]''. \tag{4.29}$$

This in turn via (4.13) would imply that $e(x_j)$ and $e'(x_j)$ are $O(h^2)$ at knot points outside of the boundary layer. However, it is typically possible to replace $(Ly_h(x_j))''$ by $(Ly(x_j))''+O(h)$ (cf. Pruess [21] or Russell and Christiansen [25]). If this were so, this R_i, $e(x_j)$, and $e'(x_j)$ would all be $O(h^3)$ at knots away from the boundary layer. This is in accord with the computational results of §5.

The final possibility is to select t_i so that (4.27b) is zero, and this gives t_i as the Gauss-Legendre point on the interval [0, 1], i.e.,

$$t_i=\left(1-1/\sqrt{3}\,\right)/2. \tag{4.30}$$

This choice of t_i is known to give poor results when $hp(x)/\varepsilon\gg 1$; however, in §5, we show that it may be used outside of subintervals containing boundary layers provided that y_h and y_h' are computed accurately enough at the ends of subintervals containing boundary layers.

4.3. Collocation points for Problem 2. Again, let x be a knot point, say x_j, and use (3.9) and (4.3) to write

$$\Pi(x_{i-1}+hs, x_j)=\Pi(x_i, x_j)\hat{\pi}_i(s)=\delta_{ij}\hat{\pi}(s), \qquad i\leqq j, \tag{4.31a}$$

$$\Pi(x_j, x_i+hs)=\Pi(x_j, x_i)\hat{\lambda}_i(s)=\delta_{ij}\hat{\lambda}_i(s), \qquad i\geqq j, \tag{4.31b}$$

where now

$$\hat{\pi}_i(s)=\exp\left\{-h\int_s^1\sqrt{-\hat{q}_i(s)/\varepsilon}\;ds\right\}, \tag{4.31c}$$

$$\hat{\lambda}_{i+1}(s)=\exp\left\{-h\int_0^s\sqrt{-\hat{q}_{i+1}(s)/\varepsilon}\;ds\right\}. \tag{4.31d}$$

The last forms of (4.31) again follow from our assumption that terms of $O(e^{-\rho_i/2})$ are

negligible. Using (3.11) and (4.31) in (4.6), we have

$$e(x_j)=\frac{h}{2}\left(\delta_{0j}\tilde{S}_1+\delta_{Nj}S_N-S_j-\tilde{S}_{j+1}\right), \tag{4.32a}$$

$$\begin{aligned} e'(x_j)=\frac{h}{2}\Big[&-(\rho_0^*/h+q_0'/q_0)\delta_{0j}\tilde{S}_1+(\rho_N^*/h-q_N'/q_N)\delta_{Nj}S_N \\ &+(\rho_j^*/h+q_j'/q_j)S_j-(\rho_j^*/h-q_j'/q_j)\tilde{S}_{j+1}\Big], \end{aligned} \tag{4.32b}$$

where

$$q_j=q(x_j), \tag{4.33a}$$

$$\rho_j^*=h\sqrt{-q(x_j)/\varepsilon}\ , \tag{4.33b}$$

and

$$S_j=\int_0^1\hat{\pi}_j(s)\left[\varepsilon^2q(x_j)\hat{q}_j(s)\right]^{-1/4}(1-P)\hat{r}_j(s)ds, \tag{4.34a}$$

$$\tilde{S}_{j+1}=\int_0^1\hat{\lambda}_{j+1}(s)\left[\varepsilon^2q(x_j)\hat{q}_j(s)\right]^{-1/4}(1-P)\hat{r}_{j+1}(s)ds. \tag{4.34b}$$

Thus, in this problem, there are no regular integrals to consider.

It suffices to find collocation points for S_j, since analogous results for $\tilde{S}_{j+1}$ will follow by replacing s by $1-s$ and making the appropriate sign changes in (4.34b). Thus, using Laplace's method we approximate (4.34a) by

$$S_j\approx\int_0^1 e^{-\rho_j^*(1-s)}\left[\varepsilon^2q(x_j)\hat{q}_j(s)\right]^{-1/4}(1-P)\hat{r}_j(s)ds. \tag{4.35}$$

For tension spline approximations, the choice of ρ_j given by (3.13b) reduces both $\hat{u}_j(s)$ and $\hat{v}_j(s)$ (cf. (4.10)) to $O(h^2/\rho_j)$ and it is still reasonable to select t_j according to (4.22), i.e., so that (4.21) is satisfied with ρ_j and ρ_j^* give by (3.13b) and (4.33b), respectively. We continue to refer to this choice of t_j as Method 1. Likewise, for polynomial approximations, we select t_j according to (4.28), i.e., so that (4.27a) is satisfied, and still refer to this as Method 3. Both methods reduce S_j (hence, $\tilde{S}_{j+1}$) to at least $O(ht_j)$. Since t_j is $O(1/\rho_j)$ or $O(1/\rho_j^*)$ for Method 1 or 3, respectively, and h/ρ_j and h/ρ_j^* are both $O(\sqrt{\varepsilon}\,)$ (cf. (3.13b) and (4.33b)), we would expect (cf. (4.23a)) $e(x_j)$ to be at most $O(h\sqrt{\varepsilon}\,)$ at knots away from boundary layers. The computational results of §5 support this conclusion.

5. Numerical results. In this section, we apply Methods 1, 2, and 3 of §4 to three examples having known exact solutions. Our calculations are performed on a uniform partition with spacing h, and the adjoint tension parameters ρ_j^* are approximated as

$$\rho_j^*=h\begin{cases}|p(x_k)/\varepsilon-q(x_k)/p(x_k)|, & |\mu p(x_j)/\varepsilon|\geqq|\mu q(x_j)|,\\ \sqrt{-\mu q(x_j)/2\varepsilon}\ , & |\mu p(x_j)/\varepsilon|<|\mu q(x_j)|,\end{cases} \quad j=1,2,\cdots,N, \tag{5.1a}$$

where

$$k=\begin{cases}j-1 & \text{if } \mu p(x_j)/\varepsilon<0,\\ j & \text{if } \mu p(x_j)/\varepsilon>0.\end{cases} \tag{5.1b}$$

The tension parameters ρ_j and collocation points t_j on I_j are chosen as follows.

Method 1: Select ρ_j according to (3.12) and

$$t_j=\min\left\{(1/\rho_j)\ln(1+\rho_j/\rho_j^*),(1-1/\sqrt{3}\,)/2\right\}. \tag{5.2a}$$

Method 2: Select ρ_j according to (3.12) and

$$t_j=\frac{1}{2}\left[1-\frac{2}{\rho_j}\cosh^{-1}\left(\frac{2}{\rho_j}\sinh\frac{\rho_j}{2}\right)\right]. \tag{5.2b}$$

Method 3: Select $\rho_j=0$ and

$$t_j=\frac{2}{\rho_j^*}\,\frac{\omega(\rho_j^*/2)}{1+\sqrt{1-4\omega(\rho_j^*/2)/\rho_j^*}}. \tag{5.2c}$$

We also consider "partial tension" methods, where the above rules for selecting ρ_j and t_j are only applied on subintervals containing boundary layers and collocation is performed at the Gauss-Legendre point with $\rho_j=0$ on all other subintervals. These methods can potentially converge as $O(h^4)$ outside of boundary layer subintervals provided that the numerical solution and its derivative have been computed accurately enough at the ends of subintervals containing boundary layers. Thus, we would not expect partial tension to be useful with Method 2, since there can be large errors in the derivative of the computed solution at the knots. We denote the partial tension methods that use either Method 1 or 3 within boundary layer subintervals as either Method 1P or 3P, respectively. In order to automatically locate subintervals containing boundary layers we first compute a preliminary solution, c_j^0, d_j^0, $j=0,1,\cdots,N$ using either Method 1 or 3 on all subintervals. Using this solution we calculate

$$\bar{y}_j''=|\mu[\,f(x_j)-p(x_j)d_j^0-q(x_j)c_j^0\,]/2\varepsilon|, \qquad j=1,2,\cdots,N, \tag{5.3}$$

and set $\rho_j=0$ and $t_j=(1-1/\sqrt{3}\,)/2$ on any subinterval having

$$\bar{y}_j''<\delta\min\{1,\bar{y}_1'',\bar{y}_2'',\cdots,\bar{y}_N''\}, \tag{5.4}$$

where δ is a threshold constant which we normally take as 50. The problem is then re-solved using the new values of ρ_j and t_j. This procedure is somewhat ad hoc and has not been totally effective, especially when h is relatively large and c_j^0 and d_j^0 are inaccurate. This can cause errors in $\bar{y}_j''$ which can lead to the erroneous conclusion that a subinterval contains a boundary layer when in fact it does not, and vice versa.

Each example was solved for various values of ε and $h=(b-a)/N$ with $N=2^k$, $k=2,3,\cdots,7$. The error in the solution and its derivative on a partition with spacing h are measured by

$$\|e^{(i)}\|_{h,\Delta_0}\equiv\max_{x_j\in\Delta_0}|y^{(i)}(x_j)-y_h^{(i)}(x_j)|, \qquad i=0,1, \tag{5.5}$$

where Δ_0 is a fixed uniform partition that is specified with each example. The order of convergence r_i is computed as

$$r_i=\ln\left\{\|e^{(i)}\|_{h,\Delta_0}/\|e^{(i)}\|_{h/2,\Delta_0}\right\}/\ln 2, \qquad i=0,1. \tag{5.6}$$

All calculations were performed in double precision on an IBM 3033 computer. Errors that are less than 5×10^{-15} are recorded as zero in the tables.

Example 1.

$$\varepsilon y'' + \big((1+x)^2 y\big)' = e^{-x/2}\big[(1+x)(3-x)+\varepsilon/2\big]/2, \qquad 0<x<1,$$
$$y(0)=0, \qquad y(1)=e^{-1/2}-e^{-7/3\varepsilon}. \tag{5.7}$$

The exact solution of this problem is

$$y(x)=\exp(-x/2)-\exp\big[-x(x^2+3x+3)/3\varepsilon\big].$$

There is a boundary layer of width $O(\varepsilon)$ near $x=0$.

In order to demonstrate how poorly collocation at the Gauss-Legendre points with cubic polynomials can behave on singularly-perturbed problems, we solved (5.7) with $\varepsilon=10^{-4}$ and $h=\frac{1}{8}$ by this method and plotted the computed solution in Fig. 2. It bears little resemblance to the exact solution, which is essentially $e^{-x/2}$ for $x>10^{-3}$. Pointwise the numerical solution approximately lies on the straight line joining the two boundary values $y(0)$ and $y(1)$.

We solved (5.7) for $\varepsilon=10^{-i}$, $i=2,4,6,8$, using Methods 1, 2, 3, and 1P. The errors $\|e\|_{h,\Delta_0}$ and $\|e'\|_{h,\Delta_0}$ on the partition $\Delta_0=\{\frac{1}{8},\frac{1}{4},\frac{3}{8},\cdots,\frac{7}{8}\}$ are presented in Tables 1 and 2, respectively, for $\varepsilon=10^{-2}$, 10^{-4}, and 10^{-8}. (The results for $\varepsilon=10^{-6}$ were

TABLE 1

Error and order of convergence for Example 1 *measured on* $\Delta_0=\{\frac{1}{8},\frac{1}{4},\frac{3}{8},\cdots,\frac{7}{8}\}$.

Method	N	$\varepsilon=10^{-2}$		$\varepsilon=10^{-4}$		$\varepsilon=10^{-8}$	
		$\|e\|_{h,\Delta_0}$	r_0	$\|e\|_{h,\Delta_0}$	r_0	$\|e\|_{h,\Delta_0}$	r_0
1	4	3.87E-2		4.63E-2		4.37E-2	
	8	1.95E-2	1.0	2.44E-2	0.8	2.44E-2	0.8
	16	7.06E-3	1.5	1.09E-2	1.2	1.09E-2	1.2
	32	2.02E-3	1.8	5.15E-3	1.1	5.15E-3	1.1
	64	2.50E-4	3.0	2.49E-3	1.0	2.53E-3	1.0
	128	5.69E-6	5.5	1.21E-3	1.0	1.25E-3	1.0
2	4	2.34E-2		2.09E-3		1.28E-3	
	8	1.28E-2	0.9	1.13E-3	0.9	4.40E-4	1.5
	16	1.07E-4	6.9	9.59E-5	3.6	1.07E-4	2.0
	32	3.34E-5	1.7	2.14E-5	2.2	2.63E-5	2.0
	64	4.83E-6	2.8	4.41E-6	2.3	6.50E-6	2.0
	128	4.37E-7	3.5	7.39E-7	2.6	1.62E-6	2.0
3	4	4.37E-3		3.57E-5		1.34E-5	
	8	3.70E-3	0.2	5.56E-5	−0.6	1.77E-6	2.9
	16	5.79E-7	12.6	2.76E-7	7.7	1.99E-7	3.2
	32	5.92E-8	3.3	3.37E-8	3.0	3.38E-8	2.6
	64	1.93E-9	4.9	4.19E-9	3.0	4.23E-9	3.0
	128	6.43E-11	4.9	5.19E-10	3.0	5.37E-10	3.0
1P	4	2.76E-6		3.69E-7		1.10E-2	
	8	1.06E-6	1.3	3.38E-8	3.4	3.29E-8	18.4
	16	4.64E-6	−2.1	1.86E-9	4.2	2.05E-9	4.0
	32	2.12E-6	1.1	2.64E-11	6.1	1.33E-10	6.0
	64	9.11E-8	4.5	5.89E-11	−1.2	8.68E-12	3.9
	128	4.68E-9	4.3	1.78E-11	1.7	6.24E-13	3.8

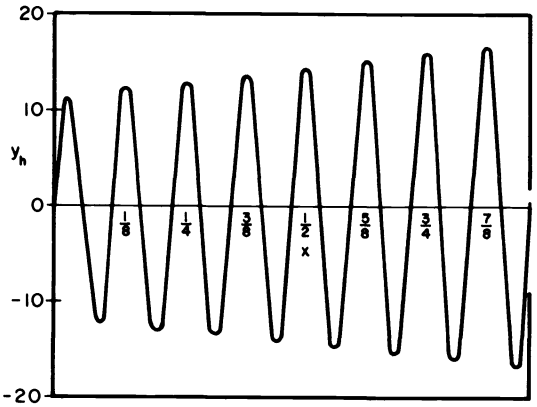


FIG 2. *Solution of Example* 1 *using cubic polynomials and collocation at the Gauss-Legendre points.*

TABLE 2

Error and order convergence in the derivative of the solution of Example 1 *measured on* $\Delta_0 = \{\frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \cdots, \frac{7}{8}\}$.

Method	N	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-8}$	
		$\|e'\|_{h,\Delta_0}$	r_1	$\|e'\|_{h,\Delta_0}$	r_1	$\|e'\|_{h,\Delta_0}$	r_1
1	4	6.38E-2		6.97E-2		6.99E-2	
	8	3.54E-2	0.8	4.34E-2	0.7	4.35E-2	0.7
	16	1.22E-2	0.3	1.93E-2	1.2	1.94E-2	1.2
	32	3.14E-3	2.0	9.15E-3	1.1	9.22E-3	1.1
	64	1.15E-4	4.8	4.43E-3	1.0	4.50E-3	1.0
	128	8.09E-5	0.5	2.15E-3	1.0	2.22E-3	1.0
2	4	4.14E 0		1.01E-2		1.01E 6	
	8	1.75E 0	1.2	2.43E-1	2.1	2.45E 5	2.0
	16	3.84E-2	2.2	5.58E 0	2.1	6.02E 4	2.0
	32	6.64E-3	2.5	1.37E 0	2.0	1.49E 4	2.0
	64	8.24E-4	3.0	3.38E-1	2.0	3.72E 3	2.0
	128	7.10E-5	3.5	8.24E-2	2.0	9.27E 2	2.0
3	4	3.05E-3		5.68E-5		2.14E-5	
	8	1.99E-2	−2.7	9.62E-5	−0.8	3.14E-6	2.8
	16	8.65E-5	7.8	4.90E-7	7.6	3.18E-7	3.3
	32	1.92E-5	5.2	6.00E-8	3.0	6.01E-8	2.4
	64	1.37E-6	3.8	7.44E-9	3.0	7.53E-9	3.0
	128	9.56E-8	3.8	9.23E-10	3.0	9.54E-10	3.0
1P	4	9.18E-4		6.08E-5		1.76E-2	
	8	1.41E-4	2.7	2.41E-5	1.3	2.19E-5	9.7
	16	5.86E-4	1.3	6.46E-6	1.9	5.48E-6	2.0
	32	2.68E-4	1.1	7.33E-7	3.1	1.52E-6	1.9
	64	1.15E-5	4.5	8.11E-7	−0.1	4.25E-7	1.8
	128	5.93E-7	4.3	2.18E-7	1.9	1.46E-7	1.5

essentially the same as those for 10^{-8}.) $\|e\|_{h,\Delta_0}$ is also plotted as a function of $1/h$ in Fig. 3 for Methods 1, 2, and 3, and $\varepsilon = 10^{-i}$, $i = 2,4,6,8$. Tables 1 and 2 indicate that when $\varepsilon/h \ll 1$, Methods 1, 2, and 3 are converging as $O(h)$, $O(h^2)$, and $O(h^3)$, respectively, and that $\|e'\|_{h,\Delta_0}$ is converging as $O(h^2/\varepsilon)$ for Method 2. For larger values of ε, e.g. $\varepsilon = 10^{-2}$, the order of convergence can be seen to increase as h decreases (ε/h increases) and the collocation points move closer to the Gauss-Legendre points. Partial tension with Method 1P yields a dramatic improvement in the results obtained by Method 1.

In order to provide some indication of how Methods 1, 2, and 3 behave on subintervals containing boundary layers and between knot points, we plotted their computed solutions $y_h(x)$ on $0 \leqq x \leqq 2h$ (Fig. 4) and their errors $e(x)$ and $e'(x)$ on $0 \leqq x \leqq 1$ (Figs. 5, 6, 7) for $\varepsilon = 10^{-4}$ and $h = 1/8$. The error in the Method 2 solution

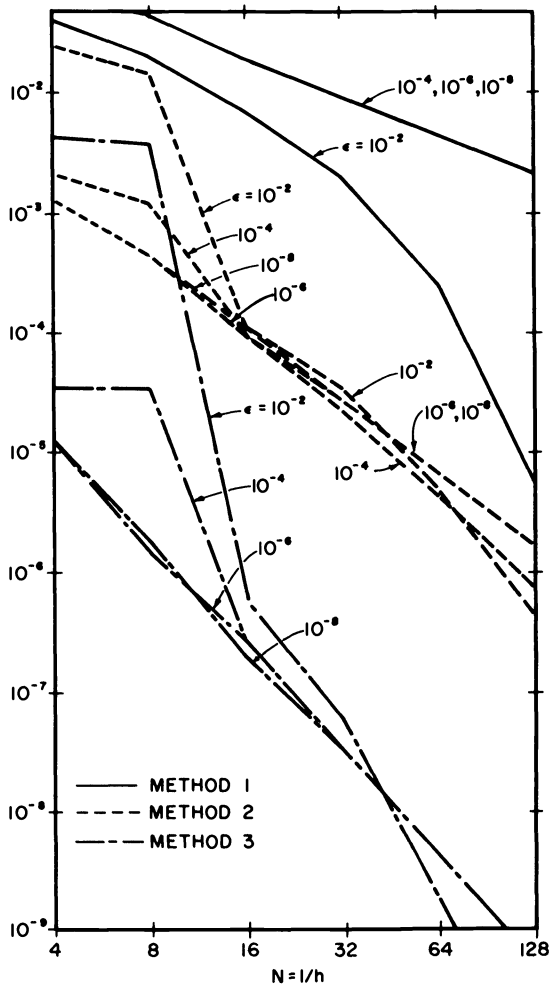


FIG 3. *Error* $\|e\|_{h,\Delta_0}$ *for Example* 1 *using Methods* 1, 2, *and* 3 *and measured on the partition* $\Delta_0 = \{\frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \cdots, \frac{7}{8}\}$.

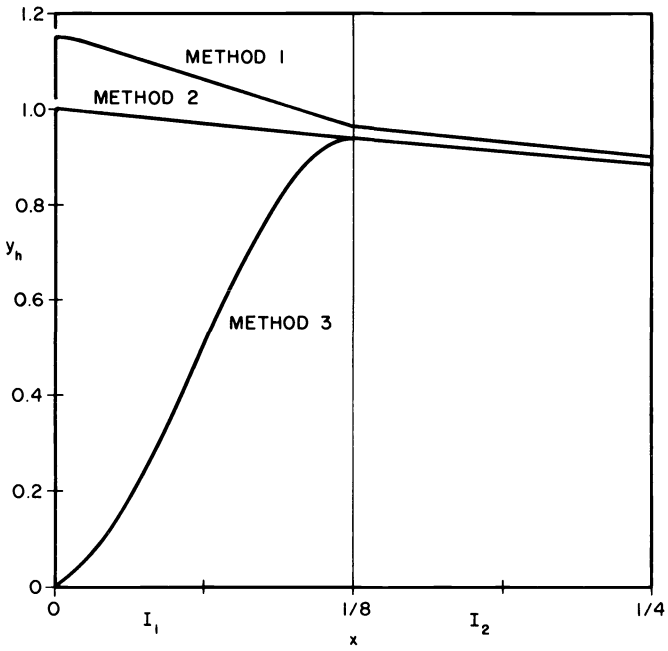


FIG 4. *Solutions of Example* 1 *using Methods* 1, 2, *and* 3 *on* $0 \leq x \leq \frac{1}{4}$ *for* $\varepsilon = 10^{-4}$ *and* $h = \frac{1}{8}$.

shown in Fig. 4 is less than 3.2×10^{-3} for all $x \in I_1 \cup I_2$. Method 1 yields poor accuracy for $x \in I_1$, but it does predict the correct width of the boundary layer. Method 3 dissipates the boundary layer; however, the dissipation is largely confined to the subinterval I_1 containing the boundary layer. Figs. 5–7 show that all three methods have spurious internal boundary layer jumps in $y_h'(x)$ at the ends of each subinterval and that Method 2, because of the singular behavior of y_h' in ε at the knots, has spurious jumps in $y_h(x)$ itself. The jumps in $y_h(x)$ are small, and one is not normally interested in the solution at points other than the knot points; however, the results indicate that some caution is needed when using (2.7-12) to interpolate the solution between knot points.

Figs. 5–7 further indicate that the pointwise error is largest at $x_1 = h$, and decreases at knots that are farther from the boundary layer. This is generally true for other values of h as well and, thus, we have tabulated $|e(h)|$ for Methods 1, 2, and 3 with $\varepsilon = 10^{-i}$, $i = 2,4,6,8$ and $N = 2^i$, $i = 2,3,\cdots,7$, in Table 3. The results for Methods 2 and 3 indicate that $|e(h)|$ cannot be reduced below $O(\varepsilon)$ until ε/h^2 and ε/h^3, respectively, are sufficiently small.

Example 2.

$$\varepsilon^2 y'' - (x^2+\varepsilon)y = -(x^2+\varepsilon)(1+\sin \pi x) - \varepsilon^2 \pi^2 \sin \pi x, \qquad 0 < x < 1,$$
$$y(0) = y(1) = 0. \tag{5.8}$$

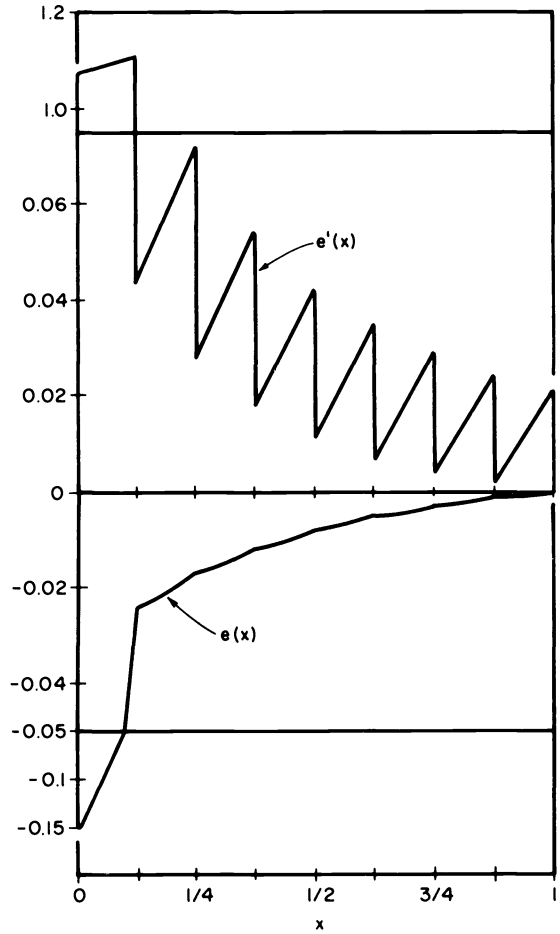


FIG 5. *Error* $e(x)$ *and its derivative* $e'(x)$ *in the solution of Example* 1 *by Method* 1 *for* $\varepsilon = 10^{-4}$ *and* $h = \frac{1}{8}$. (*Note*: $e'(0) = -5.76 \times 10^3$.)

The exact solution of this example is

$$y(x) = 1 + \sin \pi x - \frac{1}{\operatorname{erf}\left(\sqrt{1/\varepsilon}\right)} \left\{ \left(1 - e^{-1/2\varepsilon}\right) W\left(x/\sqrt{\varepsilon}\right) e^{-x^2/2\varepsilon} + \left[1 - e^{-1/2\varepsilon} W\left(\sqrt{1/\varepsilon}\right)\right] e^{-(1-x^2)/2\varepsilon} \right\},$$

where

$$W(z) = e^{z^2} \operatorname{erfc}(z).$$

(Note that y'' is multiplied by ε^2 instead of ε for notational simplicity.) The exact solution of (5.8) has a boundary layer of width $O(\sqrt{\varepsilon})$ near $x = 0$ and one of width $O(\varepsilon)$ near $x = 1$. This problem does not satisfy the assumptions of Problem 2 since

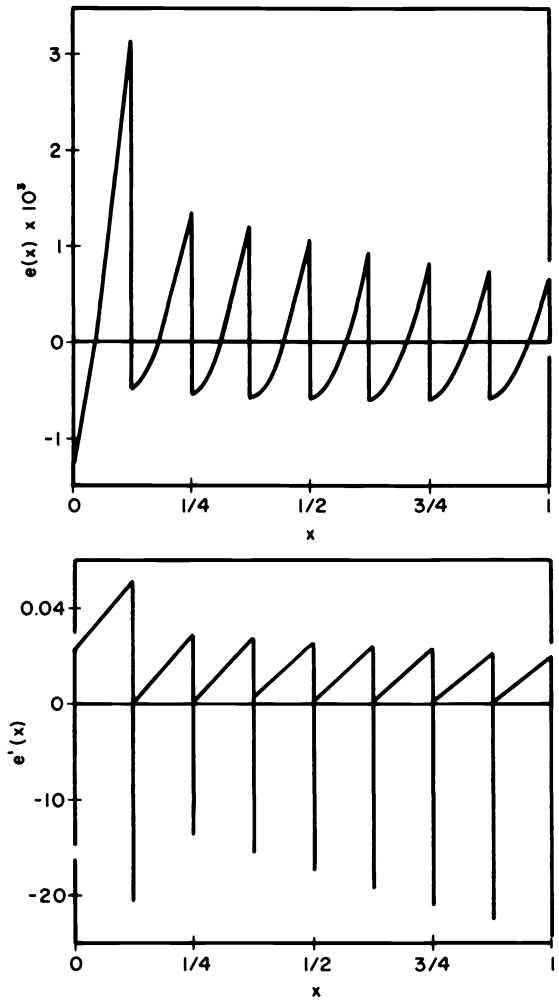


FIG 6. *Error $e(x)$ and its derivative $e'(x)$ in the solution of Example* 1 *by Method* 2 *for* $\varepsilon = 10^{-4}$ *and* $h = \frac{1}{8}$.

$q(x) = -(x^2+\varepsilon)$ cannot be bounded away from zero at $x=0$ as $\varepsilon \to 0$; thus, $x=0$ is a turning point.

We computed solutions by Methods 1 and 3 for $\varepsilon = 10^{-i}$, $i=2,4,6,8$. Solutions were also calculated using the collocation points of Method 3 (cf. (5.2c)), but with splines under tension instead of polynomial splines, and these are denoted as Method 3T. The errors $\|e\|_{h,\Delta_0}$ and $\|e'\|_{h,\Delta_0}$ are presented in Tables 4 and 5, respectively, for the partition $\Delta_0 = \{\frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \cdots, \frac{7}{8}\}$. Partial tension solutions using Method 1 and 3T were also calculated, and the results were marginally more accurate than those of Method 1.

Tables 4 and 5 indicate that when $\varepsilon/h \ll 1$, $\|e\|_{h,\Delta_0}$ and $\|e'\|_{h,\Delta_0}$ are $O(\varepsilon h)$ and $O(h^2)$, respectively, for Method 1 and $O(\varepsilon h^3)$ and $O(h^2)$, respectively, for Method 3T. The small errors in Method 3 make it difficult to estimate the order of convergence.

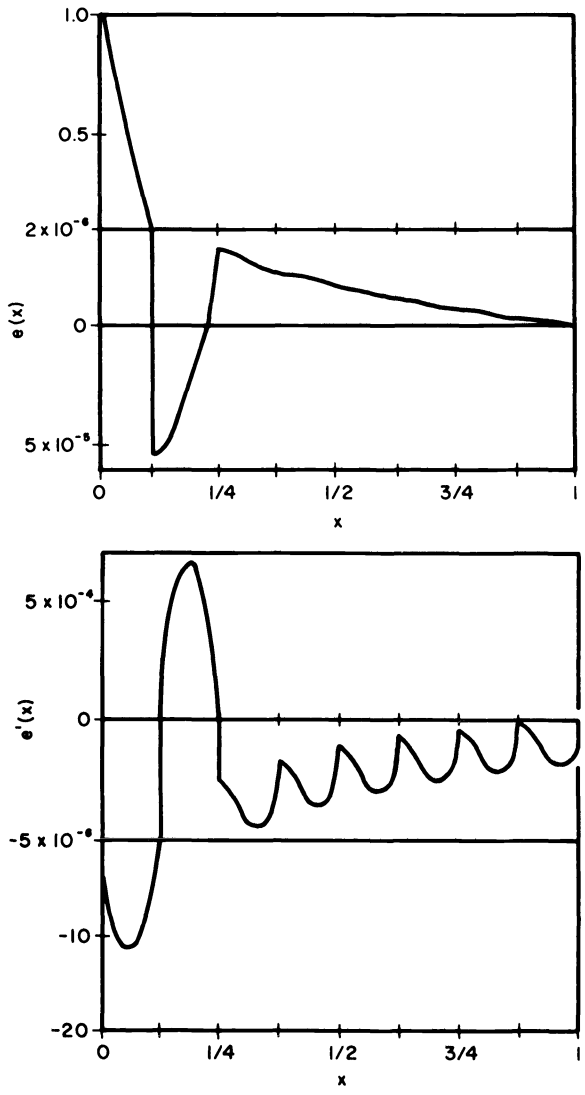


FIG 7. *Error* $e(x)$ *and its derivative* $e'(x)$ *in the solution of Example* 1 *by Method* 3 *for* $\varepsilon = 10^{-4}$ *and* $h = \frac{1}{8}$. (*Note*: $e'(0) = 1.0 \times 10^4$.)

The largest error on the partition $\Delta = \{0, h, 2h, \cdots, Nh = 1\}$ used for the computation was once again at the end of a subinterval containing a boundary layer, i.e., either at $x_1 = h$ or $x_{N-1} = (N-1)h$. To indicate how this error behaves, we present results for $\|e\|_{h,\Delta}$ in Table 6. For small values of ε/h we see that the polynomial solution (Method 3) is not converging in h, and that this situation is remedied by using tension splines (Method 3T). Although we have not done so, we suspect that it would have been sufficient to apply tension only within subintervals containing boundary layers. Furthermore, Methods 1 and 3 do not appear to be uniformly convergent in h for all ε. All methods are converging as $O(\varepsilon)$, which accounts for the remarkable accuracy when ε is small.

TABLE 3
Error at the knot point x_1 for Example 1.

Method	N	$\varepsilon=10^{-2}$		$\varepsilon=10^{-4}$		$\varepsilon=10^{-6}$		$\varepsilon=10^{-8}$	
		$\lvert e(h)\rvert$	r	$\lvert e(h)\rvert$	r	$\lvert e(h)\rvert$	r	$\lvert e(h)\rvert$	r
1	4	3.87E-2		4.36E-2		4.37E-2		4.37E-2	
	8	1.95E-2	1.0	2.44E-2	0.8	2.45E-2	0.8	2.44E-2	0.8
	16	7.89E-3	1.3	1.28E-2	0.9	1.29E-2	0.9	1.29E-2	0.9
	32	2.25E-3	1.8	6.57E-3	1.0	6.62E-3	1.0	6.62E-3	1.0
	64	3.08E-4	2.9	3.30E-3	1.0	3.35E-3	1.0	3.35E-3	1.0
	128	6.23E-6	5.6	1.63E-3	1.0	1.69E-3	1.0	1.69E-3	1.0
2	4	2.34E-2		2.09E-3		1.30E-3		1.28E-3	
	8	1.28E-2	0.9	1.13E-3	0.9	4.52E-4	1.5	4.40E-4	1.5
	16	4.19E-3	1.6	7.15E-4	0.7	1.36E-4	1.3	1.26E-4	1.8
	32	2.47E-4	4.1	5.39E-4	0.4	4.32E-5	1.7	3.35E-5	1.9
	64	4.45E-5	2.5	4.36E-4	0.3	1.76E-5	1.3	8.72E-6	1.9
	128	5.48E-6	3.0	3.51E-4	0.3	1.05E-5	0.7	2.31E-6	1.5
3	4	4.37E-3		3.57E-5		1.28E-5		1.34E-5	
	8	3.70E-3	0.2	5.56E-5	−0.6	1.62E-6	3.0	1.77E-6	2.9
	16	1.38E-3	1.4	6.16E-5	−0.2	3.35E-7	2.3	3.04E-7	2.5
	32	7.65E-4	0.9	6.34E-5	−0.0	8.50E-7	−1.3	2.99E-8	1.1
	64	8.48E-5	3.2	6.29E-5	0.0	7.13E-7	0.3	2.99E-8	0.0
	128	4.39E-6	4.3	6.05E-5	0.1	6.79E-7	0.1	3.61E-7	−3.6

TABLE 4
Error and order of convergence for Example 2 measured on $\Delta_0=\{\frac{1}{8},\frac{1}{4},\frac{3}{8},\cdots,\frac{7}{8}\}$.

Method	N	$\varepsilon=10^{-2}$		$\varepsilon=10^{-4}$		$\varepsilon=10^{-6}$		$\varepsilon=10^{-8}$	
		$\lVert e\rVert_{h,\Delta_0}$	r_0	$\lVert e\rVert_{h,\Delta_0}$	r_0	$\lVert e\rVert_{h,\Delta_0}$	r_0	$\lVert e\rVert_{h,\Delta_0}$	r_0
1	4	7.35E-3		1.54E-4		1.57E-6		1.57E-8	
	8	3.83E-3	0.9	8.75E-5	0.8	9.19E-7	0.8	9.21E-9	0.8
	16	3.99E-4	3.3	3.05E-5	1.5	3.24E-7	1.5	3.24E-9	1.5
	32	4.67E-5	3.1	1.30E-5	1.8	1.49E-7	1.1	1.49E-9	1.1
	64	3.20E-6	3.9	5.84E-6	1.2	7.28E-8	1.0	7.30E-10	1.0
	128	2.05E-7	4.0	2.50E-6	1.2	3.61E-8	1.0	3.63E-10	1.0
3	4	3.13E-2		4.66E-4		4.67E-6		4.67E-8	
	8	2.93E-2	0.1	5.92E-4	−0.3	5.95E-6	−0.3	5.96E-8	−0.4
	16	3.42E-4	6.4	9.92E-7	9.2	1.02E-10	15.8	1.04E-14	22.5
	32	4.58E-6	6.2	5.97E-10	10.7	1.00E-13	10.0	0.0	
	64	2.87E-7	4.0	1.57E-10	1.9	2.46E-14	2.0	0.0	
	128	1.80E-8	4.0	2.37E-11	2.7	6.00E-15	2.0	0.0	
3T	4	4.93E-3		8.55E-5		8.65E-7		8.65E-9	
	8	1.05E-3	2.2	4.85E-5	0.8	5.06E-7	0.8	5.06E-9	0.8
	16	8.70E-5	3.6	3.84E-6	3.7	3.94E-8	3.7	3.94E-10	3.7
	32	5.76E-6	3.9	4.73E-7	3.0	4.88E-9	3.0	4.88E-11	3.0
	64	3.66E-7	4.0	5.74E-8	3.0	6.08E-10	3.0	6.09E-11	3.0
	128	2.29E-8	4.0	6.66E-9	3.1	7.60E-11	3.0	7.61E-13	3.0

TABLE 5

Error and order of convergence in the derivative of the solution of Example 2 measured on $\Delta_0 = \{\frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \cdots, \frac{7}{8}\}$.

Method	N	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-8}$	
		$\|\|e'\|\|_{h,\Delta_0}$	r_1	$\|\|e'\|\|_{h,\Delta_0}$	r_1	$\|\|e'\|\|_{h,\Delta_0}$	r_1	$\|\|e'\|\|_{h,\Delta_0}$	r_1
1	4	1.36E-1		2.81E-1		2.79E-1		2.78E-1	
	8	2.68E-2	2.3	8.65E-2	1.7	8.76E-2	1.7	8.76E-2	1.7
	16	4.46E-3	2.6	2.37E-2	1.9	2.49E-2	1.8	2.49E-2	1.8
	32	1.43E-4	5.0	5.88E-3	2.0	6.45E-3	1.9	6.46E-3	1.9
	64	8.35E-6	4.1	1.36E-3	2.1	1.63E-3	2.0	1.64E-3	2.0
	128	5.10E-7	4.0	3.02E-4	2.2	4.07E-4	2.0	4.09E-4	2.0
3	4	2.04E 0		2.97E 0		2.97E 0		2.98E 0	
	8	2.37E 0	−0.2	4.82E 0	−0.7	4.85E 0	−0.7	4.85E 0	−0.7
	16	2.87E-2	6.4	8.37E-3	9.2	3.93E-8	26.9	8.40E-7	22.5
	32	4.12E-5	9.4	6.34E-7	10.4	4.09E-9	3.3	2.72E-8	4.9
	64	2.13E-5	1.0	3.78E-8	4.1	4.86E-10	3.1	2.74E-8	0.0
	128	1.34E-6	4.0	3.48E-9	3.4	9.52E-11	2.4	2.59E-8	0.1
3T	4	1.10E-1		2.01E-1		2.02E-1		2.02E-1	
	8	1.45E-2	2.9	7.31E-2	1.5	7.45E-2	1.4	7.45E-2	1.4
	16	5.18E-4	4.8	1.89E-2	2.0	2.04E-2	1.9	2.04E-2	1.9
	32	1.25E-4	2.1	4.63E-3	2.0	5.33E-3	1.9	5.33E-3	1.9
	64	1.15E-5	3.4	1.07E-3	2.1	1.35E-3	2.0	1.35E-3	2.0
	128	7.98E-7	3.8	2.28E-4	2.2	3.36E-4	2.0	3.39E-4	2.0

TABLE 6

Error and order of convergence for Example 2 measured on $\Delta = \{0, h, 2h, \cdots, Nh = 1\}$.

Method	N	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-6}$		$\varepsilon = 10^{-8}$	
		$\|\|e\|\|_{h,\Delta}$	r	$\|\|e\|\|_{h,\Delta}$	r	$\|\|e\|\|_{h,\Delta}$	r	$\|\|e\|\|_{h,\Delta}$	r
1	4	7.35E-3		1.54E-4		1.57E-6		1.57E-8	
	8	3.83E-3	0.9	8.75E-5	0.8	9.19E-7	0.8	9.21E-9	0.8
	16	3.99E-4	3.3	4.61E-5	0.9	4.76E-7	0.9	4.78E-9	0.9
	32	4.74E-5	3.1	1.35E-3	−4.9	2.38E-7	1.0	2.41E-9	1.0
	64	1.63E-5	1.5	2.81E-3	−1.1	8.28E-8	1.5	1.21E-9	1.0
	128	1.42E-6	3.5	6.32E-4	2.2	9.01E-7	−3.4	6.01E-10	1.0
3	4	3.13E-2		4.66E-4		4.67E-6		4.67E-8	
	8	2.93E-2	0.1	5.92E-4	−0.3	5.95E-6	−0.3	5.96E-8	−0.4
	16	2.21E-2	0.4	9.59E-4	−0.7	9.70E-6	−0.7	9.70E-8	−0.7
	32	7.52E-3	1.6	1.72E-3	−0.8	1.76E-5	−0.9	1.76E-7	−0.9
	64	9.96E-4	2.9	3.20E-3	−0.9	3.35E-5	−0.9	3.35E-7	−0.9
	128	6.75E-5	3.9	5.98E-3	−0.9	6.55E-5	−1.0	6.55E-7	−1.0
3T	4	4.93E-3		8.55E-5		8.65E-7		8.65E-9	
	8	1.05E-3	2.2	4.85E-5	0.8	5.06E-7	0.8	5.06E-9	0.8
	16	1.43E-4	2.9	2.86E-5	0.8	2.63E-7	0.9	2.63E-9	0.9
	32	6.88E-6	4.4	1.28E-3	−5.5	1.32E-7	1.0	1.32E-9	1.0
	64	2.15E-5	−1.6	8.97E-4	0.5	9.07E-8	0.5	6.65E-10	1.0
	128	2.00E-6	3.4	3.48E-4	1.4	8.80E-7	−3.3	3.35E-10	1.0

Example 3. (cf. Hemker [15])

$$\begin{aligned} &\varepsilon y'' + xy' = -\varepsilon\pi^2 \cos \pi x - \pi x \sin \pi x, \qquad -1 < x < 1, \\ &y(-1) = -2, \qquad y(1) = 0. \end{aligned} \tag{5.9}$$

The exact solution of this example is

$$y(x) = \cos \pi x + \operatorname{erf}\left(x/\sqrt{2\varepsilon}\right)/\operatorname{erf}\left(1/\sqrt{2\varepsilon}\right).$$

The problem has a turning point at $x=0$, and the exact solution features an interior or "shock" layer there.

Solutions wer calculated by Methods 1, 2, 3, and 1P for $\varepsilon = 10^{-i}$, $i=2,4,6,8$. The errors $\|e\|_{h,\Delta_0}$ and $\|e'\|_{h,\Delta_0}$ on the partition $\Delta_0 = \{-\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$ are presented in Tables 7 and 8, respectively for $\varepsilon = 10^{-2}$, 10^{-4}, and 10^{-8}. (The results for $\varepsilon = 10^{-6}$ were essentially the same as those for $\varepsilon = 10^{-8}$.) The results for $\|e\|_{h,\Delta_0}$ are indicating the same orders of convergence as found in Example 1; however, for Methods 1 and 3, $\|e'\|_{h,\Delta_0}$ is much more accurate than the corresponding values for Example 1.

In order to include the behavior of the solution in the turning point region, we have tabulated $\|e\|_{h,\Delta}$ on the partition $\Delta = \{-1, -1+h, \cdots, -1+Nh=1\}$ used for the calculation in Table 9. Note that $\|e\|_{h,\Delta_0} = \|e\|_{h,\Delta}$ for Method 1, so the maximum error is not in the turning point region. Methods 2 and 3 are both exhibiting regions of nonuniform convergence in h.

TABLE 7

Error and order of convergence for Example 3 *measured on* $\Delta_0 = \{-\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

Method	N	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-8}$	
		$\|e\|_{h,\Delta_0}$	r_0	$\|e\|_{h,\Delta_0}$	r_0	$\|e\|_{h,\Delta_0}$	r_0
1	4	5.26E-1		5.70E-1		5.71E-1	
	8	2.57E-1	1.0	3.40E-1	0.7	3.41E-1	0.7
	16	9.60E-2	1.4	1.83E-1	0.9	1.83E-1	0.9
	32	1.79E-2	2.4	9.40E-2	1.0	9.50E-2	0.9
	64	1.88E-4	6.6	4.73E-2	1.0	4.83E-2	1.0
	128	1.20E-5	4.0	2.33E-2	1.0	2.43E-2	1.0
2	4	3.16E-1		6.06E-1		6.30E-1	
	8	1.00E-1	1.7	4.24E-1	0.5	4.53E-1	0.5
	16	1.45E-2	2.8	3.58E-2	3.6	3.43E-2	3.7
	32	1.97E-3	2.9	9.78E-3	1.9	1.02E-2	1.7
	64	1.76E-4	3.5	2.51E-3	2.0	2.66E-3	1.9
	128	1.25E-5	3.8	6.09E-4	2.0	6.76E-4	2.0
3	4	5.34E-2		2.84E-2		2.77E-2	
	8	2.12E-2	1.3	6.52E-3	2.1	6.26E-3	2.1
	16	2.45E-3	3.1	8.09E-4	3.0	8.10E-4	3.0
	32	1.45E-4	4.1	1.03E-4	3.0	1.03E-4	3.0
	64	8.46E-6	4.1	1.29E-5	3.0	1.30E-5	3.0
	128	5.00E-7	4.1	1.61E-6	3.0	1.63E-6	3.0
1P	4	2.74E-2		5.70E-1		5.71E-1	
	8	1.49E-2	0.9	1.51E-1	1.9	1.52E-1	1.9
	16	7.80E-4	4.3	8.41E-6	17.5	9.44E-6	14.0
	32	6.76E-5	3.5	2.09E-7	5.3	5.88E-7	4.0
	64	4.34E-6	4.0	3.29E-7	−0.7	3.67E-8	4.0
	128	2.72E-7	4.0	2.56E-8	3.7	2.29E-9	4.0

TABLE 8

Error and the order of convergence in the derivative of the solution of Example 3 measured on $\Delta_0 = \{-\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

Method	N	$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-8}$	
		$\|\|e'\|\|_{h,\Delta_0}$	r_1	$\|\|e'\|\|_{h,\Delta_0}$	r_1	$\|\|e'\|\|_{h,\Delta_0}$	r_1
1	4	1.42E 0		1.71E 0		1.71E 0	
	8	1.36E 0	0.1	2.07E 0	−0.3	2.07E 0	−0.3
	16	3.10E-2	5.5	2.38E-4	13.1	5.07E-7	22.0
	32	2.78E-2	0.2	4.73E-4	−1.0	1.25E-7	2.0
	64	2.00E-2	0.5	6.71E-4	−0.5	6.00E-8	1.1
	128	1.39E-3	3.8	7.68E-4	−0.2	5.23E-8	0.2
2	4	1.90E 1		2.79E 3		2.91E 7	
	8	7.18E 0	1.4	1.18E 3	1.2	1.19E 7	1.3
	16	1.48E 0	2.3	3.45E 2	1.8	3.51E 6	1.8
	32	1.97E-1	2.9	9.03E 1	1.9	9.26E 5	1.9
	64	1.76E-2	3.5	2.26E 1	2.0	2.37E 5	2.0
	128	1.26E-3	3.8	5.49E 1	2.0	5.97E 4	2.0
3	4	8.01E-2		5.45E-4		5.39E-8	
	8	2.31E-1	−1.5	6.04E-4	−0.1	3.69E-8	0.5
	16	9.31E-3	4.6	3.75E-6	7.3	0.0	
	32	8.58E-4	3.4	8.72E-8	5.4	0.0	
	64	6.02E-5	3.8	4.88E-8	0.8	0.0	
	128	3.96E-6	3.9	2.45E-8	1.0	0.0	
1P	4	2.31E 0		1.71E 0		1.71E 0	
	8	4.16E-1	2.5	2.07E 0	−0.3	2.07E 0	−0.3
	16	4.43E-3	6.6	9.57E-3	7.8	9.55E-3	7.8
	32	1.87E-3	1.2	2.00E-3	2.3	2.08E-3	2.2
	64	1.12E-4	4.1	8.03E-4	1.3	6.77E-4	1.6
	128	7.52E-6	3.9	9.36E-5	3.1	1.90E-4	1.8

6. Discussion and conclusion. Based on the results of §§4 and 5, we conclude that splines under tension are most suitable in regions containing boundary and/or interior layers and that collocation with piecewise polynomials is superior elsewhere. In particular, when $\varepsilon/h \ll 1$ Method 3 provides an approximation that converges outside of boundary layer subintervals as $O(h^3)$ when $p(x) \neq 0$ and at least $O(\sqrt{\varepsilon h})$ when $p(x) \equiv 0$ on $[a, b]$. Hemker [15] and de Groen and Hemker [10] reached similar conclusions with their exponentially fitted Galerkin methods.

Partial tension can converge as $O(h^4)$ outside of boundary layer subintervals, but either requires a knowledge of boundary layer locations or a preliminary solution to automatically locate them. The latter procedure may be useful for nonlinear problems where it is necessary to solve a sequence of linear problems to find the solution; however, for linear problems it does not seem to warrant the extra computational effort merely to obtain one order of accuracy more than that available by Method 3.

The use of "one-sided splines under tension," i.e., the selection of basis functions that satisfy

$$(\eta' + \rho\eta)''' = 0, \qquad 0 < s < 1, \tag{6.1}$$

instead of (2.14) would undoubtably improve the results on subintervals where

TABLE 9
Error and order of convergence for Example 3 measured on $\Delta=\{-1,-1+h,-1+2h,\cdots,-1+Nh=1\}$.

Method	N	$\varepsilon=10^{-2}$		$\varepsilon=10^{-4}$		$\varepsilon=10^{-6}$		$\varepsilon=10^{-8}$	
		$\|e\|_{h,\Delta}$	r	$\|e\|_{h,\Delta}$	r	$\|e\|_{h,\Delta}$	r	$\|e\|_{h,\Delta}$	r
1	4	5.26E-1		5.70E-1		5.70E-1		5.71E-1	
	8	2.57E-1	1.0	3.40E-1	0.7	3.41E-1	0.7	3.41E-1	0.7
	16	9.60E-2	1.4	1.83E-1	0.9	1.83E-1	0.9	1.83E-1	0.9
	32	1.81E-2	2.4	9.40E-2	1.0	9.50E-2	0.9	9.50E-2	0.9
	64	1.91E-4	6.6	4.73E-2	1.0	4.83E-2	1.0	4.83E-2	1.0
	128	1.22E-5	4.0	2.33E-2	1.0	2.43E-2	1.0	2.43E-2	1.0
2	4	3.16E-1		6.06E-1		6.25E-1		6.30E-1	
	8	1.00E-1	1.7	4.24E-1	0.5	4.51E-1	0.5	4.53E-1	0.5
	16	1.77E-2	2.5	2.99E-1	0.5	3.66E-1	0.3	3.68E-1	0.3
	32	2.11E-3	3.1	1.82E-1	0.7	3.38E-1	0.1	3.42E-1	0.1
	64	1.84E-4	3.5	6.09E-2	1.6	3.20E-1	0.1	3.35E-1	0.0
	128	1.31E-5	3.8	9.39E-3	2.7	2.88E-1	0.2	3.33E-1	0.0
3	4	5.34E-2		9.22E-2		9.29E-2		9.29E-2	
	8	2.12E-2	1.3	7.25E-3	3.7	7.48E-3	3.6	7.48E-3	3.6
	16	5.68E-3	1.9	1.52E-2	−1.1	8.10E-4	3.2	8.10E-4	3.2
	32	3.02E-4	4.2	4.19E-2	−1.5	6.90E-4	0.2	1.03E-4	3.0
	64	1.89E-5	4.0	1.89E-2	1.1	2.68E-3	−2.0	2.80E-5	1.9
	128	1.19E-6	4.0	1.49E-2	0.3	1.00E-2	−1.9	1.09E-4	−2.0
1P	4	2.74E-2		5.70E-1		5.71E-1		5.71E-1	
	8	1.49E-2	0.9	4.37E-1	0.4	4.45E-1	0.4	4.45E-1	0.4
	16	3.27E-3	2.3	1.34E-1	1.7	1.43E-1	1.6	1.43E-1	1.6
	32	1.93E-4	4.1	2.94E-2	2.2	3.77E-2	1.9	3.78E-2	1.9
	64	1.33E-5	3.9	2.89E-2	0.0	9.50E-3	2.0	9.59E-3	2.0
	128	8.28E-7	4.0	8.06E-3	1.8	2.32E-3	2.0	2.41E-3	2.0

$p(x)\not\equiv 0$. A basis for these approximations would contain either the exponential $\exp(-\rho s)$ when $\rho>0$, or $\exp(-|\rho|(1-s))$ when $\rho<0$, and not both as in the current case. This would more accurately represent the exact solution of problems where $p(x)\not\equiv 0$ on $[a,b]$. The results could possibly be further improved by not restricting the collocation points to be placed symmetrically on each subinterval and by using nonuniform partitions. Another interesting approach would be to try the polynomial taut splines and rational splines of de Boor [9, Chapt. 16] and Pruess [23] as alternatives to the exponential splines in tension. Each of these potential improvements is currently under investigation. Extensions of our methods to higher order scalar and vector systems of two-point boundary value problems as well as second order parabolic partial differential equations are also being studied.

Acknowledgment. The authors would like to thank Professors G. J. Habetler and R. E. O'Malley, Jr., for their valuable comments and suggestions, and Mr. G. M. Heitker for his assistance with some of the programming.

REFERENCES

[1] L. R. ABRAHAMSON, 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] U. ASCHER, J. CHRISTIANSEN, AND R. D. RUSSELL, *Collocation software for boundary value* ODE's, preprint, 1979.

[3] C. M. BENDER AND S. A. ORSZAG, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw Hill, New York, 1978.

[4] A. E. BERGER, J. M. SOLOMON, AND M. CIMENT, *Higher order accurate tridiagonal difference methods for diffusion convection equations*, Proceedings of the Third IMACS Conference on Computer Methods for Partial Differential Equations, Lehigh University, 1979.

[5] A. CLINE, *Curve fitting in one and two dimensions using splines under tension*, Comm. ACM, 17 (1974), pp. 218–223.

[6] J. COLE, *Perturbation Methods in Applied Mathematics*, Blaisdell, Waltham MA, 1968.

[7] J. M. COYLE AND J. E. FLAHERTY, *The solution of boundary value problems having rapidly oscillating solutions*, in preparation.

[8] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582–607.

[9] C. DE BOOR, *A Practical Guide to Splines*, Applied Mathematical Sciences 27, Springer-Verlag, New York, 1978.

[10] 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.

[11] W. ECKHAUS, *Matched Asymptotic Expansions and Singular Perturbations*, North-Holland Mathematics Studies 6, North-Holland, Amsterdam, 1973.

[12] 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.

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

[14] J. C. HEINRICH, AND O. C. ZIENKIEWICZ, *Quadratic finite element schemes for two dimensional convective-transport problems*, Internat. J. Numer. Methods. Engrg., 11 (1977), pp. 1831–1844.

[15] P. W. HEMKER, *A Numerical Study of Stiff Two-point Boundary Problems*, Ph.D. dissertation, Mathematisch Centrum, Amsterdam, 1977.

[16] A. M. IL'IN, *Differencing scheme for a differential equation with a small parameter affecting the highest derivative*, Math. Notes, 6 (1969), pp. 596–602.

[17] H. O. KREISS, *Difference approximations for singular perturbation problems*, in Numerical Solutions of Boundary Value Problems for Ordinary Differential Equations, A. K. Aziz, ed., Academic Press, New York, 1975, pp. 199–212.

[18] R. E. O'MALLEY, JR., *Introduction to Singular Perturbation*, Academic Press, New York, 1979.

[19] C. E. PEARSON, *On a differential equation of the boundary layer type*, J. Math. Physics, 47 (1968), pp. 134–154.

[20] _____, *On nonlinear ordinary differential equations of boundary layer type*, J. Math. Phys., 47 (1968), pp. 351–358.

[21] S. PRUESS, *Solving linear boundary value problems by approximating the coefficients*, Math. Comp., 27 (1973), pp. 551–561.

[22] _____, *Properties of splines in tension*, J. Approx. Theory, 17 (1976), pp. 86–96.

[23] _____, *Alternatives to the exponential spline in tension*, Math. Comp., 33 (1979), pp. 1273–1281.

[24] R. D. RUSSELL, *Collocation for systems of boundary value problems*, Numer. Math., 23 (1974), pp. 119–133.

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

[26] R. D. RUSSELL AND L. F. SHAMPINE, *A collocation method for boundary value problems*, Numer. Math., 19 (1972), pp. 1–28.

[27] D. SCHWEIKERT, *An interpolation curve using splines in tension*, J. Math. Phys., 45 (1966), pp. 312–317.

[28] H. SPÄTH, *Spline-Algorithmen zur Konstruktion glatter Kurven und Flächen*, R. Oldenbourg Verlag, München, 1973; English translation by W. D. Hoskins and H. W. Sager, Spline Algorithms for Curves and Surfaces, Utilitas Mathematica, Winnipeg, 1974.

[29] C. R. STEELE, *Application of the WKB method in solid mechanics*, Mechanics Today, 3 (1976), pp. 243–295.

[30] J. M. VARAH, *Alternate row and column elimination for solving certain linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 71–75.

[31] W. WASOW, *Asymptotic Expansions for Ordinary Differential Equations*, Wiley-Interscience, New York, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

0196-5204/80/0102-0010 $1.00/0

LEAST ABSOLUTE DEVIATIONS CURVE-FITTING*

PETER BLOOMFIELD† AND WILLIAM STEIGER‡

Abstract. A method is proposed for least absolute deviations curve fitting. It may be used to obtain least absolute deviations fits of general linear regressions. As a special case it includes a minor variant of a method for fitting straight lines by least absolute deviations that was previously thought to possess no generalization. The method has been tested on a computer and was found on a range of problems to execute in as little as 1/3 the CPU time required by a published algorithm based on linear programming. More important, this advantage appears to increase indefinitely with the number of data points

Key words. least absolute deviations, linear programming

1. Introduction. The least absolute deviations method of curve-fitting consists of fitting the model

$$y_i = \sum_{j=1}^{k} x_{ij}\theta_j + \varepsilon_i, \qquad i = 1, \ldots, n \tag{1}$$

to data $(x_{i1}, \ldots, x_{ik}, y_i, i = 1, \ldots, n)$ by choosing the parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ to minimize the sum of absolute deviations,

$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{k} x_{ij}\theta_j \right|. \tag{2}$$

According to Eisenhart (1961), the minimization of a quantity like (2) was suggested by Boscovitch in the mid-eighteenth century for fitting lines well before the introduction of the method of least squares. Boscovitch added the condition that the sum of the signed residuals be zero, which constrains the line to pass through the centroid of the $(\mathbf{x}_i, y_i)$ points. He also described an algorithm for finding the slope of the minimizing line, which can of course be used in conjunction with different constraints such as that of a zero intercept.

More than one hundred years later, Edgeworth dropped the constraint and proposed the fitting of lines and of more complicated models by unconstrained minimization of (2). However, the computations are inherently more complex than the solution of the linear equations that arise in the method of least squares, and Edgeworth's method does not seem to have been used widely.

A new method was introduced by Rhodes (1930) and discussed by Singleton (1940), who also proposed an alternative. The development of linear programming and the observation of Harris (1950) that the least absolute deviations fitting problem could be turned into a linear programming problem was the next major advance. This

*Received by the editors October 22, 1979, and in final revised form May 6, 1980.

†Department of Statistics, Princeton University, Princeton, New Jersey 08540. The work of this author was supported in part by the U.S. Energy Research and Development Administration, under Contract EY76-SO2-2310, awarded to the Department of Statistics, Princeton University.

‡Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903. The work of this author was supported in part by the Rutgers University Research Council. Computing facilities were provided by the Rutgers University PDP 20 devoted to computer science research.

line was pursued by Wagner (1959) and many others, including Barrodale and Roberts (1973, 1974) and Narula and Wellington (1977). We comment on these algorithms in §§4 and 5.

The current resurgence of interest in least absolute deviations methods is associated with the development of robust and resistant methods (see Huber, 1973 or Andrews, 1974). That a least absolute deviations fit is less sensitive to extreme errors than is a least squares fit was noted by Bowditch in an English translation of a work by Laplace (see Eisenhart, 1961). Similar remarks have been made by Edgeworth and by Rhodes (1930), who exhibited an example to support the point. While there are techniques that are at the same time statistically more efficient in reasonable circumstances and even less affected by extreme errors, the conceptual simplicity of least absolute deviations estimates and their competitive computational cost makes them well worth considering. We note that these estimates are maximum likelihood and hence asymptotically efficient in the (perhaps uncommon) situation when the errors follow the double exponential Laplace distribution.

In addition to their use for robust estimation of regression equations, least absolute deviations estimates could also be used as starting points for iterative estimation schemes. Although this would be computationally more expensive than the use of least squares estimates as starting points, the resistance of the procedure to outliers would be improved.

Schlossmacher (1973) described a different relationship between iterative and least absolute deviations methods. He pointed out that least absolute deviations estimates could be found by iteratively reweighted least squares. However, numerical tests have shown that this is a computationally expensive way to find an approximate solution to a problem that admits exact solution. Abdelmalek (1971) suggested a different approximate solution, namely minimization of the l_p-norm with p approaching 1 from above. This approach was found by Barrodale and Roberts (1973) to be inefficient.

2. Minimizing the sum of absolute deviations. When there is only one degree of freedom in the fit, such as when $k=1$ or when constraints are imposed as by Boscovitch, the solution is straightforward. For

$$\sum_{i=1}^{n} |y_i-\theta x_i| = \sum_{i=1}^{n} |x_i||y_i/x_i-\theta|,$$

and the minimizing value of θ is thus the weighted median of the ratios y_i/x_i, with respect to weights $|x_i|$. This weighted median may be defined as any value θ such that

$$\sum_{i:\, y_i/x_i=\theta} |x_i| \geqq \left| \sum_{i:\, y_i/x_i<\theta} |x_i| - \sum_{i:\, y_i/x_i>\theta} |x_i| \right|$$

It may be found by ordering the ratios, and then summing the weights from one end until the partial sum first exceeds or equals one half the total of the weights. The corresponding ratio is the weighted median.

This shows that when $k=1$, θ may always be taken to be one of the ratios y_i/x_i, and that at least one of the residuals $y_i-\theta x_i$ vanishes. In the general case there is a solution $\boldsymbol{\theta}$ for which at least r of the residuals vanish, r being the rank of $\mathbf{X}=(x_{ij})$. This is easily seen by a linear programming formulation of (2) (e.g. Wagner (1959)) or

via a simple direct argument: Suppose $0 \leqq m < r$ residuals, $y_i - \sum_{j=1}^{k} x_{ij}\theta_j$, vanish, say for $i = i_1, \ldots, i_m$. Since $m < r$, there is a row, say the p^{th}, not in the space spanned by rows $i_1, \ldots, i_m$, and a vector $\boldsymbol{\delta}$ orthogonal to rows $i_1, \ldots, i_m$ but not row p. Thus the function $f(t) = \sum_{i=1}^{n} |y_i - \sum_{j=1}^{k} x_{ij}(\theta_j + t\delta_j)|$ is a sum with zero terms when $i = i_1, \ldots, i_m$, and $S(\boldsymbol{\theta})) = f(0)$. Finally, write $f(t) = \sum_{i=1}^{n} |w_i - tv_i|$, where $w_i = y_i - \sum_{j=1}^{k} x_{ij}\theta_j$ and $v_i = \sum_{j=1}^{k} x_{ij}\delta_j$. From the $k = 1$ case, f is minimized for $t = \hat{t} = y_q / x_q$, and the q^{th} term of the sum is zero. Now $S(\boldsymbol{\theta} + \hat{t}\boldsymbol{\delta})$ has $m+1$ zero residuals at $i = i_1, \ldots, i_m$ or $i = q$, and $S(\boldsymbol{\theta} + \hat{t}\boldsymbol{\delta}) = f(\hat{t}) \leqq f(0) = S(\boldsymbol{\theta})$, an argument that holds as long as $m < r$.

The problem of minimizing (2) is thus a discrete search problem. One need only search the $\binom{n}{r}$ combinations of r zero residuals, find an appropriate $\boldsymbol{\theta}$ for each, and evaluate $S(\boldsymbol{\theta})$.

This observation motivates a simple method for solving the two-parameter problem of fitting a straight line $y = \theta_1 + \theta_2 x$. The method, described by Rhodes (1930) and Karst (1958), is the basis of a computer algorithm published by Sadovski (1974). First, a line is fitted to the data while constrained to pass through some arbitrary point, such as the origin. As noted above, the fitted line must pass through at least one data point. Next, a line is fitted while constrained to pass through the data point thus identified (an arbitrary one in the case of multiplicity). This identifies a new point to replace the previous one, and the algorithm continues. It terminates when the fitted line does not change.

It is easily seen that the sum of absolute deviations goes down at each step, and that no more than $n-1$ steps can be taken before termination. At some stages the problem may be degenerate in the sense that more than two residuals are zero. Some care must be taken so the algorithm does not cycle endlessly (see Sposito, 1976), or terminate prematurely. Karst (1958) recognized the difficulties introduced by degeneracy. The optimal line through P_1 may pass through P_2 and vice versa, and yet still not be the overall optimum.

Narula and Wellington (1977) described a method based on linear programming (in which, surprisingly, they readopted Boscovitch's constraint that the fit should pass through the centroid). They commented that the Karst/Sadovski technique does not lend itself to regression problems with more parameters. There is, however, a very natural extension that not only leads to an efficient numerical solution of the problem, but also sheds light on the relationship of the least absolute deviations problem to linear programming. This extension, based on the foregoing argument and related to the descent method of Usow (1967), is described in the next section.

3. The method. Assume that $n > k$, and that $\mathbf{X}$ is of full rank. The basis of the method is to search for a set of k data points such that the fit, when constrained to make the corresponding residuals vanish, is optimal. As in the method described above for $k = 2$, this set of data points is found iteratively, by successive improvements. In each iteration one point from the current set is identified as a good prospect for deletion. This point is then replaced by the best alternative. This clearly generalizes the 2-parameter technique. It is also related to the descent technique of Usow (1967). The novel features of our method are

* an efficient method for finding optimal replacement, and
* a heuristic method for identifying the point to be deleted.

(i) *Replacement*. Suppose that the rows of $\mathbf{X}=(x_{ij})$ that correspond to the current set of points are $\mathbf{x}_1^T,\ldots,\mathbf{x}_k^T$, and that $\mathbf{x}_k^T$ has been identified for replacement. There is a one-dimensional set of parameter values $\boldsymbol{\theta}$ that satisfy the remaining constraints

$$y_i=\mathbf{x}_i^T\boldsymbol{\theta}, \qquad i=1,\ldots,k-1, \tag{3}$$

which we may parametrize as

$$\boldsymbol{\theta}=\boldsymbol{\theta}_0+t\boldsymbol{\delta},$$

where $\boldsymbol{\theta}_0$ is any arbitrary member of the set, and $\boldsymbol{\delta}$ satisfies

$$\mathbf{x}_i^T\boldsymbol{\delta}=0, \qquad i=1,\ldots,k-1. \tag{4}$$

Within this set, the optimum may be found by minimizing

$$\sum_{i=1}^{n}|y_i-\mathbf{x}_i^T(\boldsymbol{\theta}_0+t\boldsymbol{\delta})| \tag{5}$$

with respect to the scalar t. Rewriting this objective function as

$$\sum_{i=1}^{n}|(y_i-\mathbf{x}_i^T\boldsymbol{\theta}_0)-t(\mathbf{x}_i^T\boldsymbol{\delta})|,$$

one can minimize (5) by solving the one-dimensional problem of regressing $y_i-\mathbf{x}_i^T\boldsymbol{\theta}_0$ on $\mathbf{x}_i^T\boldsymbol{\delta}$. As was remarked earlier, this minimizing value of t is the weighted median of

$$(y_i-\mathbf{x}_i^T\boldsymbol{\theta}_0)/(\mathbf{x}_i^T\boldsymbol{\delta}), \qquad i=1,\ldots,n,$$

with respect to the weights

$$|\mathbf{x}_i^T\boldsymbol{\delta}|, \qquad i=1,\ldots,n.$$

Thus the minimizing value of t may be found efficiently by using a weighted version of the partial quick-sort procedure (see Chambers, (1971)). The next set of parameter values may then be computed as $\boldsymbol{\theta}_0+t\boldsymbol{\delta}$. Finally, the data point that replaces $\mathbf{x}_k$ is the point corresponding to the weighted median.

For $\boldsymbol{\theta}_0$, we use the parameter values associated with $\mathbf{x}_1,\ldots,\mathbf{x}_k$, for these satisfy (3) and the additional equation with $i=k$. The vector $\boldsymbol{\delta}$ is determined up to scalar multiples by (4).

(ii) *Deletion*. A reasonable goal when deleting a point at some intermediate stage would be to select that point which, when optimally replaced, leads to the largest reduction in the objective function. However, we have found no reliable way to identify this point short of trying all deletions. In tests, this "look ahead" approach was more expensive than the heuristic method described below, and often did not lead to fewer steps before termination.

Our heuristic method is based on gradients. The quantity (5) is a convex, piecewise linear function of t. We examine the larger of its left-hand derivative at 0 and the negative of its right-hand derivative, which may be expressed as

$$\left|\sum_{i:\,r_i<0}w_i-\sum_{i:\,r_i>0}w_i\right|-\sum_{i:\,r_i=0}w_i, \tag{6}$$

where

$$r_i=(y_i-\mathbf{x}_i^T\boldsymbol{\theta}_0)/\mathbf{x}_i^T\boldsymbol{\delta}$$

and

$$w_i = |\mathbf{x}_i^T \boldsymbol{\delta}|.$$

If (6) is negative, then the unique minimum of (5) is at $t=0$, and thus if $\mathbf{x}_k$ were deleted, the same point would immediately reenter (or possibly a different point, but still leading to no improvement in the objective function nor any change in the parameter values). If (6) is zero, $t=0$ is still a minimum, but there is an interval of values all of which also minimize (5). Thus the objective function still cannot be improved, although there are other sets of parameter values that give the same value.

We therefore avoid nonpositive values of (6). To convert the gradient into an estimate of the amount by which (5) may be reduced, we multiply by a rough estimate of the scale of the ratios r_i. Since the numerators are the same in all the possible cases at a given stage, we use the reciprocal of the sum of the denominators as this rough estimate. Thus we use the quantity

$$\rho = \frac{\left| \sum_{r_i<0} w_i - \sum_{r_i>0} w_i \right| - \sum_{r_i=0} w_i}{\sum w_i} \tag{7}$$

to measure the merit of deleting the given point, and we delete the point for which ρ is largest.

We note that $\max(\rho,0)$ has some of the properties of an absolute correlation coefficient between the residuals $y_i - \mathbf{x}_i^T \boldsymbol{\theta}_0$ and the linear compound $\mathbf{x}_i^T \boldsymbol{\delta}$:

* it lies between 0 and 1,
* it is 0 iff the (least absolute deviations) regression of the residuals on the compound is null,
* it is 1 iff the signs of the residuals and the compound are all the same or all opposite.

We make the calculation of (7) for each candidate for deletion economical by the choice of $\boldsymbol{\theta}_0$. For this is the same for each candidate, and hence the residuals need only be computed once. For $\boldsymbol{\delta}$ we use the appropriate column of the inverse of the $k \times k$ submatrix

$$\mathbf{Z} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_k^T \end{bmatrix}. \tag{8}$$

Thus, in evaluating the merit of deleting $\mathbf{x}_j$ from the current set, we compute (7) using the j^{th} column of $\mathbf{Z}^{-1}$ for $\boldsymbol{\delta}$. Clearly this choice satisfies the requirement embodied in (4).

In addition, the partitioning of the residuals into negative, zero, and positive values can be used as the first of the partitioning steps involved in the weighted median calculation. This reduces the series length on the average by almost one half, and leads to a further useful economy.

To start the iteration, any set of k independent rows of $\mathbf{X}$ may be chosen, with the appropriate $\boldsymbol{\theta}_0$. However the following procedure is usually more efficient. Take $\boldsymbol{\theta}=\mathbf{0}$, which corresponds to an empty set of data points. We then add variables in a stepwise fashion, until we have a fit $\boldsymbol{\theta}_0$ and a corresponding set of k data points. At each intermediate stage, the fit involves m variables, $0 \leqq m < k$, and a corresponding set of m data points with zero residuals. We then have the option of including another variable, thus increasing m to $m+1$, or improving the fit with m variables, using the principles described earlier. Both of these alternatives may be carried out by regressing the current residuals on some appropriate linear compound of the variables in the problem.

Suppose variables $v_1, \ldots, v_m$ are in the model corresponding to points $i_1, \ldots, i_m$. Thus the current m variable model has residuals $y_i - \sum_{j=1}^{m} x_{iv_j}\theta_j$ equal to zero when $i = i_1, \ldots, i_m$. If the criterion (7) is largest at variable $p \neq i_1, \ldots, i_m$, p may be added to the model as follows. Choose $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m, \delta_{m+1})$ orthogonal to $(x_{iv_1}, \ldots, x_{iv_m}, x_{ip})$ for $i = i_1, \ldots, i_m$, which, since there are only m of them, can clearly be done. The function $f(t) = \Sigma | y_i - \sum_{j=1}^{m} x_{iv_j}(\theta_j + t\delta_j) - t x_{ip}\delta_{m+1} |$ is a sum with zero terms when $i = i_1, \ldots, i_m$, and $f(0)$ is the sum of absolute deviations at the current fit. If we write $f(t) = \Sigma | w_i - t u_i |$, where $w_i = y_i - \sum_{j=1}^{m} x_{iv_j}\theta_j$ and $u_i = \sum_{j=1}^{m} x_{iv_j}\delta_j + x_{ip}\delta_{m+1}$, it is clear that f is minimized at a value of $t = \hat{t} = w_q / u_q$ and that the q^{th} term of the sum is then equal to zero. Hence we have an $m+1$ variable model $(\boldsymbol{\theta}, 0) + \hat{t}\boldsymbol{\delta}$ based on variables $v_1, \ldots, v_m$, p and points $i_1, \ldots, i_m, q$ corresponding to zero residuals.

On the other hand if (7) is largest for variable v_p, say, the corresponding point i_p is deleted and replaced in the manner already described; an improved m variable model results. In any case it is the criterion (7) that determines whether we step up to a larger model or improve the current one without stepping up.

The mechanism described above for identifying a point to be dropped may give false indications of convergence in the presence of degeneracy. Degeneracy occurs when $k' > k$ residuals vanish at a given stage. In this case, there are $\binom{k'}{k}$ sets of points that all give the same fit. However, it is possible that not all of them can be improved by replacing a single point. To guard against the possibility of arriving at such a set of points and incorrectly concluding that the minimum has been reached, we examine all the sets before allowing the procedure to terminate.

4. Relationship to linear programming. From a computational standpoint two sets of quantities are required for the deletion phase, the residuals $y_i - \mathbf{x}_i^T\boldsymbol{\theta}$ and the weights $|\mathbf{x}_i^T\boldsymbol{\delta}|, i = 1, \ldots, n$. There are k different sets of weights, one for the value of $\boldsymbol{\delta}$ associated with the point in the current set being tested for deletion. Further, the k different $\boldsymbol{\delta}$ values are the columns of $\mathbf{Z}^{-1}$ in (8). The needed weights are thus the inner products of the rows of $\mathbf{X}$ with the columns of $\mathbf{Z}^{-1}$, the inverse of the submatrix of $\mathbf{X}$ corresponding to the data points in the current set. These quantities may be computed and updated easily by using the *pivot* operation of linear programming.

We begin with the bordered matrix $(\mathbf{X}:\mathbf{y})$. By *pivoting* on an entry in the $\mathbf{X}$ portion of this array, we mean first scaling the column in which that entry appears to make its value one, and then subtracting multiples of that column from each other column, to make the remaining entries in that row vanish. If we pivot on one entry in each row of the desired submatrix, with one entry also in each column, we obtain an array in which $\mathbf{X}$ has been replaced by the necessary inner products, and y has been

replaced by the residuals from the corresponding fit. For $\mathbf{X}$ has effectively been post-multiplied by a matrix that has reduced the relevant submatrix to the identity matrix, which must therefore be the desired inverse. Also, y has had multiples of the columns of $\mathbf{X}$ subtracted from it to make certain entries vanish, and hence has been replaced by the residuals from the corresponding fit.

When one point is replaced by another, we can adjust the array by a single additional pivot. If we pivot on that entry in the new row that falls in the same column as the unit entry in the old row, then the rows corresponding to the other points in the current set are unaffected, and hence the resulting array again contains all the values needed in the next step of the solution.

This pivoting operation is, of course, precisely the operation used in updating bases in the simplex method of linear programming. The question arises as to how well one can do using linear programming techniques. Initially, solutions by linear programming required the introduction of additional variables and constraints, as in the formulation:

$$\text{minimize} \quad \sum_{i=1}^{n} e_i$$

$$\text{subject to} \quad e_i \geqq y_i - \sum_{j=1}^{k} x_{ij}\theta_j,$$

$$e_i \geqq -y_i + \sum_{j=1}^{k} x_{ij}\theta_j, \qquad i = 1, \ldots, n.$$

Such solutions must inevitably operate with larger arrays than our method, and are not of comparable efficiency.

Another algorithm by Barrodale and Roberts (1973) seems to have been the best available method for solving (2). It appears to be closely related to linear programming but avoids the extra variables and constraints necessary in the direct formulation, above. However, as the next section strongly suggests, the present algorithm is much more efficient. It seems to be faster than the Barrodale-Roberts algorithm, and the relative advantage increases indefinitely with the size of the problem (2).

5. Computational aspects. In assessing the computational cost of the proposed least absolute deviations algorithm, it will be useful to compare it to ordinary least squares. For this purpose, we take the number of multiplication or division operations as a measure of complexity.

To find $\boldsymbol{\theta}$ in (1) with ordinary least squares, the k normal equations can be obtained at a cost of $(k-1)(k+2)n/2$ operations and solved in $k^3/3 + O(k^2)$ operations. By way of comparison, the computations involved in a single step of our algorithm are

* $2n(k-1)$ comparisons and $n(k-1)$ additions, to find the pivot column,
* finding the weighted median of the n ratios, which takes a number of comparisons and exchanges of the order of n log n and not more than n additions, and
* $n(k+1)$ multiplications and nk additions in the updating operation.

TABLE 1
An illustrative set of data from Karst (1958).

	1	2	3	4	5	6	7
x	12	18	24	30	36	42	48
y	5.27	5.68	6.25	7.21	8.02	8.71	8.42

If the least absolute deviations algorithm terminates at step N, the computational cost would be roughly

$$N(nk + \text{effort for weighted median}), \tag{9}$$

while ordinary least squares would require

$$n(k-1)(k+2)/2 + k^3/3 + O(k^2). \tag{10}$$

It is difficult to carry out the comparison further because we do not know how N depends on n, k, and $\mathbf{X}$. However, it is clear that as long as N is comparable to $k/2$ and $k^2/(2 \log n)$, the costs of ordinary least squares and our least absolute deviations algorithm will be comparable.

The remainder of this section is devoted to the question of how (9) might grow with n, k for certain specific or randomly presented curve fitting problems $\mathbf{X}$. However, instead of counting operations as in (9), we will measure complexity by the CPU time for executing the algorithm and compare this with other procedures for computing or approximating least absolute deviations fits.

To compare our "exact" procedure with that of Schlossmacher, consider the problems of obtaining a $k=2$ dimensional fit for the data of Table 1. In this problem the iterated least squares technique converged to a good approximation in 7 steps, while our least absolute deviations algorithm required 2. Although each of our iterations may be more work than a single step of the iterated least squares procedure, in view of (9) and (10) one expects our algorithm to find the exact fit in less time than that needed for convergence of the iterated least squares approximation.

Finally we performed Monte-Carlo experiments to compare our algorithm with the best available competition, that of Barrodale and Roberts (1974). The results are interesting and bear careful study.

For the model $Y=\theta_0+\theta_1 X_1+\cdots+\theta_k X_k+U, \theta_i=\sqrt{i}$, a sample $\mathbf{X}$ of size n was generated in the following way. For each $i=1,\ldots,n$, successive random numbers[1] $X_{i1}, X_{i2},\ldots, X_{ik}, U_i$ were generated and $Y_i=\theta_0+\theta_1 X_{i1}+,\ldots,+\theta_k X_{ik}+U_i$ formed. From this sample $\mathbf{X}$, LAD estimators $\hat{\theta}$ were computed with the Barrodale-Roberts algorithm and the present one, and the CPU times and numbers of iterations recorded. Since the Barrodale-Roberts iteration also exchanges one set of k zero residuals for another, it is sensible to compare iteration counts. In addition it may help to explain differences in the CPU times.

The above process was repeated for a total of 10 samples of size n. So that the comparisons do not depend too strongly on a particular sequence produced by the random number generator, the total CPU times and total iteration counts are compared.

[1]The FORTRAN function RAN of the DEC system 20 was used in this task.

Finally to explore the effect of the distribution of the point cloud **X** on the complexity of obtaining the LAD fits, three distributions for the X's and U were used. The first two are from the family of Pareto densities $f(t)=1/(1+t)^{1+\alpha}$, $t\geqq 0, \alpha>0$. For $\alpha>1$ the mean exists and equals $\alpha/(\alpha-1)$, and thus the density

$$f(t)=\alpha/[1+(t-c)]^{1+\alpha}, \qquad t\geqq c=\alpha/(\alpha-1),$$

is Pareto and centered at the mean.

We generated X's and U using this density, first when $\alpha=1.2$, and second when $\alpha=2.2$. In the former case the variance is infinite, while in the latter, though the density is long-tailed, the variance is finite. Finally, in a third set of experiments we used X's and U from the unit normal distribution.

The results are in Tables 2, 3, and 4. In each table, each cell, (n, k) has the total CPU time for the 10 samples and the total iteration count used by both the present algorithm, labeled LAD, and the Barrodale-Roberts algorithm, labeled BAR. Incidentally, the experiment reported in each cell of each table employed a different seed for the random number generator.

The results are rather striking. As the size n of the point cloud increases, our algorithm gains relative advantage over the Barrodale-Roberts method, for each k and with all underlying distributions. More specifically, the tables suggest that LAD(n), the CPU time of our algorithm, grows linearly with n, while BAR(n), the CPU time for Barrodale-Roberts grows faster than linearly, perhaps like $n \log n$ or n^2. The three tables would support

$$\text{LAD}(n)=C_k(U)\cdot n,$$

$$\text{BAR}(n)=d_k(U)\cdot n\log n \quad \text{or} \quad d_k(U)\cdot n^2,$$

TABLE 2

Total CPU *time and iteration counts for* 10 *sets of* LAD *estimates, Pareto distribution,* $\alpha=1.2$.

		k = 2		3		6	
		BAR	LAD	BAR	LAD	BAR	LAD
n	100	.76	.79	1.20	1.10	2.77	3.19
		34	42	58	34	119	71
	300	3.08	2.34	4.74	4.10	11.20	10.79
		38	45	68	60	148	83
	600	8.42	4.51	12.04	7.44	25.10	20.07
		44	40	65	49	136	78
	1200	24.33	10.18	34.89	16.27	71.72	44.69
		45	52	71	61	167	109
	1800	47.22	14.34	72.00	22.91	136.14	68.57
		41	48	77	53	163	108

TABLE 3

Total CPU *time and iteration counts for* 10 *sets of* LAD *estimates, Pareto distribution,* $\alpha = 2.2$.

		k = 2		3		6	
		BAR	LAD	BAR	LAD	BAR	LAD
n	100	.81	.83	1.21	1.31	2.91	3.41
		43	51	63	51	140	84
	300	2.83	2.51	4.53	4.15	10.41	11.33
		43	54	71	64	141	112
	600	7.35	5.64	11.18	8.08	25.91	23.87
		43	66	67	61	159	125
	1200	22.57	11.07	33.47	18.17	79.89	56.05
		52	60	87	78	231	165
	1800	40.41	17.41	63.99	24.87	139.81	89.77
		38	69	80	65	214	183

TABLE 4

Total CPU *time and iteration counts for* 10 *sets of* LAD *estimates, Gaussian distribution.*

		k = 2		3		6	
		BAR	LAD	BAR	LAD	BAR	LAD
n	100	.75	.98	1.40	1.55	3.83	4.41
		29	57	75	61	185	124
	300	2.59	3.17	5.21	5.61	15.28	16.04
		33	67	85	96	245	184
	600	6.62	6.50	12.65	10.89	41.49	35.15
		42	70	93	92	322	222
	1200	17.33	13.05	37.06	23.62	105.92	70.27
		47	67	121	104	341	219
	1800	29.63	19.52	68.93	34.66	182.80	121.88
		46	71	120	102	328	274

where $C_k(f)$, $d_k(f)$ are constants that both increase with k and each depends on f, the distribution of the X's and U (but in different ways to be mentioned presently). In fact, it seems reasonable to assert that

$$\frac{\text{BAR}(n)}{\text{LAD}(n)} = a(f) \text{ times an increasing function of } n,$$

where $a(f)$ is a constant depending only on the point cloud's density, f; thus, both $C_k(f)$ and $d_k(f)$ increase with k in the same way.

Finally, it is interesting to observe that f affects the algorithms in different ways:

(1) LAD is best for "spread out" point clouds, $\alpha = 1.2$, and worst for the normal data;

(2) BAR is best for finite variance but long-tailed point clouds, $\alpha = 2.2$, and worst, sometimes in the normal case, sometimes in the infinite variance case, $\alpha = 1.2$. In any event, the relative advantage of LAD is greatest when $\alpha = 1.2$, when the data are most heavy-tailed.

These observations are tentative, as there is little evidence to base such exact assertions upon. Nevertheless, it seems safe to claim that our algorithm has an increasing asymptotic advantage over Barrodale and Roberts as n, the number of points being fit, increases. More strongly,

$$\frac{\text{BAR}(n)}{\text{LAD}(n)} \to \infty$$

as $n \to \infty$. It would be interesting to study the ratio for more values of k and other types of distributions.

REFERENCES

N. N. ABDELMALEK (1971), *Linear L_1 approximation for a discrete point set and L_1 solutions of overdetermined linear equations*. J. Assoc. Comput. Mach., 18, pp. 41-47.

DAVID ANDREWS (1974), *A robust method for multiple linear regression*, Technometrics, 16, pp. 523–532.

I. BARRODALE AND F. D. K. ROBERTS (1973), *An impoved algorithm for discrete l_1 linear approximation*, SIAM J. Numer. Anal., 10, pp. 839–848.

_____ (1974), *Algorithm* 478: *solution of an overdetermined system of equations in the l_1 norm*, Comm. ACM, 17, pp. 319–320.

J. CHAMBERS (1971), *Algorithm* 410: *partial sorting*, Comm. ACM, 14, pp. 357–358.

F. Y. EDGEWORTH (1887), *A new method of reducing observations relating to several quantities*, Phil. Mag. (Fifth Series), 24, pp. 222–223.

_____ (1888), *On a new method of reducing observations relating to several quantities*. Phil. Mag. (Fifth Series), 25, pp. 184–191.

C. EISENHART (1961), *Boscovitch and the combination of observations* in Roger Joseph Boscovitch, L. L. Whyte, eds., Fordham University Press, New York.

T. HARRIS (1950), *Regression using minimum absolute deviations*, Amer. Statistician, 4, pp. 14–15.

PETER J. HUBER (1973), *Robust regression: asymptotics, conjectures, and Monte Carlo*, Ann. Statist. 1, pp. 799–821.

J. OTTO KARST (1958), *Linear curve fitting using least deviations*. J. Amer. Statist. Assoc., 53, pp. 118–132.

S. C. NARULA AND J. F. WELLINGTON (1977), *Algorithm AS*108. *Multiple linear regression with minimum sum of absolute errors*. Applied Stat., 26, pp. 106–111.

E. C. RHODES (1930), *Reducing observations by the method of minimum deviations*, Phil Mag. (Seventh Series), 9, pp. 974–992.

A. N. SADOVSKI (1974), *Algorithm AS*74. *L_1-norm fit of a straight line*. Applied Stat., 23, pp. 244–248.

E. J. SCHLOSSMACHER (1973), *An iterative technique for absolute deviations curve fitting*. J. Amer. Statist. Assoc., 68, pp. 857–865.

R. R. SINGLETON (1940), *A method for minimizing the sum of absolute values of deviations*, Ann. Math. Statist., 11, pp. 301–310.

V. SPOSITO (1976), *Remarks on Algorithm AS*74. Applied Stat., 25, pp. 96–97.

K. H. USOW (1967), *On L_1 approximation II: Computation for discrete functions and discretization effects*, SIAM J. Numer. Anal., 4, pp. 233–224.

HARVEY M. WAGNER (1959), *Linear programming techniques for regressions analysis*, J. Amer. Statist. Assoc., 54, pp. 206–212.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 2, June 1980

ERRATUM: NUMERICAL COMPUTATION OF THE SCHWARZ–CHRISTOFFEL TRANSFORMATION*

LLOYD N. TREFETHEN†

In this paper, pages 87-88 and 89-90 were transposed:

p. 87 should be p. 89;
p. 88 should be p. 90;
p. 89 should be p. 87;
p. 90 should be p. 88.

* This Journal, 1 (1980), pp. 82-102.
† Computer Science Department, Stanford University, Stanford, California 94305.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0007 $01.00/0

THE NUMERICAL STABILITY OF THE LEVINSON-DURBIN ALGORITHM FOR TOEPLITZ SYSTEMS OF EQUATIONS*

GEORGE CYBENKO†

Abstract. The numerical stability of the Levinson-Durbin algorithm for solving the Yule-Walker equations with a positive-definite symmetric Toeplitz matrix is studied. Arguments based on the analytic results of an error analysis for fixed-point and floating-point arithmetics show that the algorithm is stable and in fact comparable to the Cholesky algorithm. Conflicting evidence on the accuracy performance of the algorithm is explained by demonstrating that the underlying Toeplitz matrix is typically ill-conditioned in most applications.

Key words. Toeplitz matrix, Levinson-Durbin algorithm, error analysis

1. Introduction.

Suppose the real sequence

$$\rho_0 = 1, \rho_1, \cdots, \rho_p \tag{1.1}$$

defines a sequence of positive-definite symmetric Toeplitz matrices

$$T_p = \begin{bmatrix} \rho_0 & \rho_1 & \rho_2 & \cdot & \cdot & \cdot & \rho_{p-1} \\ \rho_1 & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \rho_1 \\ \rho_{p-1} & \cdot & \cdot & \cdot & & \rho_1 & \rho_0 \end{bmatrix}. \tag{1.2}$$

The Yule-Walker equations of order p are then defined to be

$$T_p \mathbf{a}_p = -(\rho_1, \rho_2, \cdots, \rho_p)' = -\mathbf{r}_p. \tag{1.3}$$

In 1947, N. Levinson [17] presented an algorithm for solving (1.3) using only $O(p^2)$ arithmetic operations and $O(p)$ storage as opposed to the $O(p^3)$ operations and $O(p^2)$ storage required by general methods such as Gaussian elimination or Cholesky triangularization. Levinson's algorithm was in fact for solving the more general Toeplitz system

$$T_p \mathbf{x} = \mathbf{y}, \tag{1.4}$$

but this involved the explicit computation of the solutions to the Yule-Walker equations of all orders strictly less than p. Durbin [12] subsequently streamlined Levinson's algorithm to solve only (1.3), and his name has been associated with the algorithm in much of the statistical and engineering literature where the Yule-Walker equations are most often encountered. The basic algorithm has been independently discovered, generalized, and modified more recently by a number of authors [1], [10], [16], [26], [27], [29], [35]. In §2, this basic algorithm will be presented and will henceforth be called the Levinson-Durbin algorithm.

Toeplitz systems such as (1.3) arise in a variety of engineering, statistical, and mathematical areas: signal processing and detection, time series analysis and prediction and the theory of orthogonal polynomials to name but a few [4], [13], [14], [18], [21], [24], [25], [27], [30], [33]. The efficiency of the Levinson-Durbin algorithm makes

*Received by the editors July 16, 1979, and in revised form June 20, 1980.
†Department of Mathematics, Tufts University, Medford, Massachusetts, 02155.

it an obvious candidate for solving the Yule-Walker equations, and the method is routinely described in most situations where the system (1.3) arises. Recently, a library of Fortran subroutines named TOEPLITZ, which is fashioned after LINPACK [7], has been developed by a joint US-USSR effort [2]. The library includes routines based on the Levinson-Durbin algorithm for solving (1.4) and inverting positive-definite symmetric Toeplitz matrices.

In spite of the algorithm's importance and relative ubiquity, its stability properties have been a subject of some controversy. Some authors claim the algorithm has poor accuracy performance [4], [13], [22], [30], while others find it to perform well [6], [8], [19]. A full spectrum of opinions may be found in the literature; a popular text [4] on time series analysis cautions that the algorithm is "numerically unstable" and therefore should be avoided in practice, while speech researchers have found the accuracy of computed solutions quite acceptable and have gone so far as to design and build dedicated hardware implementing the algorithm [19]. All points of view are apparently supported by the results of numerical testing. Unfortunately, the results of computational experiments cannot be offered as evidence towards deciding whether an algorithm is numerically stable or not unless the condition of the underlying problem itself is taken into account. It is quite possible for an extremely stable algorithm to produce answers with large relative errors if the problem is ill-conditioned (see [28] for a discussion of numerical stability and conditioning).

In light of these observations, it is important to separate conditioning and stability; these are two distinct issues. To this end, a posteriori upper and lower bounds on the condition numbers of the matrices (1.2) are derived in §3. In particular, it is seen that the circumstances which give rise to ill-conditioned systems (1.3) are precisely the circumstances which are most appropriately modeled by those very systems. This statement will be made precise in §2 and §3 by examining the application in which the Yule-Walker equations are most often encountered; that is, the linear prediction of stationary time series. Thus computational experiments based on "real world" problems are useless in discussions of stability unless the condition of the underlying problem has been taken into account. This has apparently not been the case. To this author's knowledge, there have been no published comparisons between the accuracy performance of the Levinson-Durbin algorithm and known stable algorithms such as the Cholesky method.

The results of a backward-type error analysis [32] are presented and discussed in §4. The analysis shows that residual vectors associated with solutions to (1.3) computed by the Levinson-Durbin algorithm can be bounded by an expression which is comparable to the best analytic bound possible for the residual associated with solutions computed by the stable Cholesky factorization method. The bounds on the Levinson-Durbin and Cholesky residuals are in fact virtually identical for a large class of problems. Thus, it is possible to conclude, on the basis of analytic evidence, that the Levinson-Durbin algorithm is at least as stable as the Cholesky method when attention is restricted to a subclass of problems of the type (1.3). In the general case, the analytic results derived suggest that the residual vector should be of the same order of magnitude as the residual for the Cholesky method. Consequently, it is concluded that the Levinson-Durbin algorithm is numerically stable. It seems that the controversy over the algorithm's stability and the conflicting evidence can be reconciled once the conditioning of the underlying system is taken into account.

Proofs of the results stated in §4 are presented in §5, while §6 contains a summary with concluding remarks.

2. Linear prediction, Yule-Walker equations, and the Levinson-Durbin algorithm. Toeplitz matrices arise naturally in situations where some sort of "shift invariance" occurs. Historically, the Toeplitz system (1.3) was first explicitly used by Yule [34] in the analysis of sunspot data (Walker [31] later encountered the system in his study of periodicity properties of time series). In the context of time series analysis, this shift invariance is manifested as statistical stationarity. Since this context is important for interpreting the main results, a brief development of the relevant ideas is appropriate.

Let

$$\cdots, s(-1), s(0), s(1), \cdots, s(k), \cdots \tag{2.1}$$

be a doubly infinite real sequence such that

$$0 < \sum_n s^2(n) < \infty. \tag{2.2}$$

The "prediction" or "autoregression" coefficients $a_{p,1}, a_{p,2}, \cdots, a_{p,p}$ are determined by the minimization of the "prediction error"

$$E_p = \sum_n \left[s(n) + a_{p,1} s(n-1) + \cdots + a_{p,p} s(n-p) \right]^2, \tag{2.3}$$

where summations over n are understood to be from $-\infty$ to $+\infty$. Taking the partial derivatives of (2.3) with respect to each of the $a_{p,j}$ and setting them equal to zero gives us the normal equations

$$T_p \mathbf{a}_p = T_p (a_{p,1}, \cdots, a_{p,p})' = -\mathbf{r}_p, \tag{2.4}$$

where in this case the entries of T_p,

$$\rho_k = \sum_n s(n) s(n-k), \tag{2.5}$$

are guaranteed finite by (2.2) and the Cauchy-Schwarz inequality. This system is identical to (1.3) once we normalize $\rho_0 = 1$ and observe that the resulting T_p is always positive definite. The quantities ρ_k are the normalized autocorrelations of the underlying time series, while E_p is the mean square prediction error. Upon using (2.4) to expand (2.3) it is seen that

$$E_p = 1 + \rho_1 a_{p,1} + \cdots + \rho_p a_{p,p}, \tag{2.6}$$

in which case it follows that

$$T_{p+1} \begin{pmatrix} 1 \\ \mathbf{a}_p \end{pmatrix} = (E_p, 0, \cdots, 0)', \tag{2.7}$$

and so

$$\begin{pmatrix} 1 \\ \mathbf{a}_p \end{pmatrix} \frac{1}{E_p} = (1, a_{p,1}, \cdots, a_{p,p})' / E_p \tag{2.8}$$

is the first column (and transpose of the first row by symmetry) of T_{p+1}.

Thus apart from being important in the prediction of time series the Yule-Walker equations and their solutions are intricately related to the inverses of positive-definite symmetric Toeplitz matrices. These relations can be exploited to solve the general Toeplitz system (1.4) and compute Toeplitz inverses efficiently [17], [29].

Since the Toeplitz matrix T_p is determined by p parameters, it is reasonable to expect that there are algorithms for solving (1.3) which would use fewer than the $O(p^3)$ operations required by methods not taking advantage of the Toeplitz structure.

The Levinson-Durbin algorithm is only one such efficient algorithm, but it is without doubt the most important in practice.

The Levinson-Durbin algorithm computes quantities E_j, K_j, and vectors $\mathbf{a}_j$ according to the initialization

$$E_0 = 1, \tag{2.9}$$

followed by the recursive computations for $j = 1, 2, \cdots, p$,

$$\begin{aligned} K_j &= -(\rho_j + a_{j-1,1}\rho_{j-1} + \cdots + a_{j-1,j-1}\rho_1)/E_{j-1} \\ &= -\hat{\mathbf{f}}_j' \begin{pmatrix} 1 \\ \mathbf{a}_{j-1} \end{pmatrix} \frac{1}{E_{j-1}}, \end{aligned} \tag{2.10}$$

$$\mathbf{a}_j = \begin{pmatrix} \mathbf{a}_{j-1} + K_j \hat{\mathbf{a}}_{j-1} \\ K_j \end{pmatrix}, \tag{2.11}$$

$$E_j = (1 - K_j^2) E_{j-1}. \tag{2.12}$$

Here $\mathbf{a}_j = (a_{j,1}, a_{j,2}, \cdots, a_{j,j})' \in R^j$, and

$$\hat{\mathbf{a}}_j = (a_{j,j}, a_{j,j-1}, \cdots, a_{j,1})' \tag{2.13}$$

is the symmetric reflection of $\mathbf{a}_j$. It is a simple exercise to verify, by induction, that at the jth stage of the recursive computation $\mathbf{a}_j$ solves the jth order Yule-Walker equations.

This is one of the most attractive aspects of the algorithm: all principal leading subsystems of (1.3) are solved during the intermediate computations. It is useful in applications where a priori knowledge of the appropriate order p is not available. The quantities E_j in (2.12) are precisely the same as in (2.6) (with $p = j$), which may be easily verified by induction on j. Thus (2.6) could be used in place of (2.12) in the algorithm, but the latter clearly uses fewer computations.

It can be shown [14] under the assumptions on T_p that

$$1 = E_0 \geqq E_1 \geqq \cdots \geqq E_p > 0 \tag{2.14}$$

and

$$-1 < K_j < 1 \quad \text{for } 1 \leqq j \leqq p. \tag{2.15}$$

The quantities K_j have important physical and statistical interpretations in many applications. For instance, in the study of wave propagation through concatenated lossless tubes they play a central role as "reflection coefficients," while statistically they are known as "partial correlation coefficients" [23], [25]. Following the current standard nomenclature, they will be referred to as partial correlation coefficients.

Although the partial correlation coefficients are guaranteed to be less than 1 in absolute value, their closeness to 1 reflects very important physical characteristics of the underlying problem being solved. Statistically, if $|K_j| \approx 1$ the underlying time series (2.1) is close to nonstationarity [4], while in the context of linear filtering theory, the polynomial

$$\sum_{k=0}^{p} a_{p,k} z^{p-k}, \qquad a_{p,0} = 1 \tag{2.16}$$

has roots inside, but close to the boundary, of the unit circle $|z| = 1$, and so the filter whose transfer function is the reciprocal of (2.16) is close to "instability" in the filtering sense [21]. Thus in applications the condition $|K_j| \approx 1$ often represents the

fact that the underlying problem which is being modeled by (1.3) is close to some sort of anomalous behavior. It will be seen that the nearness of the partial correlation coefficients to ± 1 also measures the conditioning of the system (1.3) or more generally (1.4).

In any case the Levinson-Durbin algorithm goes beyond being merely an efficient method of solving (1.3); it computes important and significant quantities during the intermediate computations. The algorithm requires $O(p^2)$ arithmetic operations and $O(p)$ storage allocations to solve the system (1.3).

Asymptotically faster methods of solving the general system have been developed recently [3], [5], [20], but these algorithms do not recursively solve increasingly larger subsystems of (1.3), do not have simple control structures, and are faster than the Levinson-Durbin algorithm only for large values of p. Their practical significance has yet to be determined.

3. Condition numbers of Toeplitz matrices. In this section, it will be seen that the closeness of the partial correlation coefficients to ± 1 plays a significant role in determining the conditioning of the Toeplitz systems (1.3) and (1.4). Although the bounds on the condition numbers of the Toeplitz matrix (1.2) are given in terms of a posteriori quantities, namely the partial correlation coefficients, as has been seen those quantities have physical interpretations in the most common applications. Consequently, there is usually empirical evidence for anticipating their relative sizes. The linear condition number of the Toeplitz matrix T_p is defined to be

$$\kappa(T_p) = \|T_p\| \|(T_p)^{-1}\|, \tag{3.1}$$

where $\|\cdot\|$ is some matrix norm. In this paper only the 1-norm will be considered. This quantity measures the sensitivity of solutions to linear systems involving T_p with respect to perturbations in both T_p and the right side of (1.4). Large values of $\kappa(T_p)$ indicate high sensitivity to rounding errors, and so corresponding computed solutions are expected to have large relative errors.

Since $\rho_0 = 1$ and T_p is positive definite, $|\rho_k| < 1$. Thus

$$1 \leqq \|T_p\| \leqq p, \tag{3.2}$$

so that the size of $\|(T_p)^{-1}\|$ will essentially determine the ill-conditioned situations. Attention will therefore be focused on bounding $\|(T_p)^{-1}\|$. As already seen in (2.7), the solution $\mathbf{a}_j$ to the jth order Yule-Walker equations determines the first column of $(T_{j+1})^{-1}$, and this fact can be used to obtain a triangular factorization of $(T_p)^{-1}$.

Define

$$C_p = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 \\ a_{p-1,1} & 1 & 0 & & & \cdot \\ a_{p-1,2} & a_{p-2,1} & 1 & & & \cdot \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & 0 \\ a_{p-1,p-1} & a_{p-2,p-2} & \cdot & \cdot & \cdot & 1 \end{bmatrix},$$

so that by (2.7) the product $T_p C_p$ is upper triangular, with diagonal entries

$E_{p-1}, E_{p-2}, \cdots, E_1, E_0$. Then the product $C_p' T_p C_p$ is upper triangular and symmetric, and hence diagonal. It follows that

$$C_p' T_p C_p = D_p = \begin{bmatrix} E_{p-1} & & & 0 \\ & E_{p-2} & & \\ & & \ddots & \\ 0 & & & E_0 \end{bmatrix},$$

and so

$$(T_p)^{-1} = C_p (D_p)^{-1} C_p'. \tag{3.3}$$

This equation has a striking interpretation: the Levinson-Durbin algorithm explicitly computes a triangular "square-root" factorization of $(T_p)^{-1}$. This contrasts with the Cholesky method which computes a similar factorization for T_p.

The factorization (3.3) shows that the sizes of the solutions $\mathbf{a}_j$ will play a role in determining an upper bound of the condition number of T_p. Recursions of the type (2.11) are central to this goal and to the error analysis itself. That direction will now be explored.

LEMMA 3.1. *If* $\mathbf{a}_j$ *and* K_j *are as in* (2.11) *then*

$$\sum_{i=1}^{j} a_{j,i} = \prod_{i=1}^{j} (1+K_i) - 1. \tag{3.4}$$

Proof. Use (2.11) and induction on j.

Let $\|\mathbf{x}\|_\mu$ represent the standard vector μ-norm for $\mathbf{x} \in R^n$, $\mu \geqq 1$.

LEMMA 3.2. *Suppose* $\mathbf{x}_j \in R^j$ *with* $\|\mathbf{x}_j\|_\mu \leqq r$. *For* $i > j$, $\mathbf{x}_i \in R^i$ *is defined recursively by*

$$\mathbf{x}_i = \begin{pmatrix} \mathbf{x}_{i-1} + K_i \hat{\mathbf{x}}_{i-1} \\ K_i y_i \end{pmatrix}, \tag{3.5}$$

where $|y_i| \leqq s$. *Then*

$$\|\mathbf{x}_i\|_\mu \leqq (r+s) \prod_{m=j+1}^{i} (1+|K_m|) - s. \tag{3.6}$$

Proof. Put

$$B_i = (r+s) \prod_{m=j+1}^{i} (1+|K_m|),$$

and suppose, $\|\mathbf{x}_{i-1}\|_\mu \leqq B_{i-1} - s$. Then

$$\begin{aligned} \|\mathbf{x}_i\|_\mu &\leqq \|\mathbf{x}_{i-1}\|_\mu + |K_i| \|\mathbf{x}_{i-1}\|_\mu + |K_i| |y_i| \\ &\leqq (1+|K_i|)(B_{i-1} - s) + |K_i| s \\ &= (1+|K_i|) B_{i-1} - s \\ &= B_i - s. \end{aligned}$$

Since $B_j = r+s$, the result follows by induction.

Lemmas 3.1 and 3.2 give simple upper and lower bounds on $\mathbf{a}_j$ in terms of the partial correlation coefficients K_i for $i \leqq j$.

LEMMA 3.3. *For $1 \leqq \mu$ define the conjugate exponent ν by $1/\mu + 1/\nu = 1$. Then*

$$n^{-1/\nu}\left|\prod_{j=1}^{p}(1+K_j)-1\right| \leqq \|\mathbf{a}_p\|_\mu \leqq \prod_{j=1}^{p}(1+|K_j|)-1. \tag{3.7}$$

Proof. The lower bound follows from Lemma 3.1 and Holder's inequality, while the upper bound follows from Lemma 3.2 with $r=|K_1|$ and $s=1$.

Notice that if $K_i \geqq 0$ for all i and $\mu = 1$, the bounds are tight. Since the partial correlation coefficients K_j are less than one in absolute value, it is easy to verify that each entry $a_{i,j}$ of $\mathbf{a}_i$ is bounded by the binomial coefficient C_j^i. A tighter bound in terms of the partial correlation coefficients is necessary for the remaining results. It is evident from (2.11) that each $a_{i,j}$ is a multinomial function of the K_m for $m \leqq i$. The exact form of this multinomial is described in the next lemma.

LEMMA 3.4.

$$a_{i,j} = \sum_{M_j^i} \prod_{m \in M_j^i} K_m, \tag{3.8}$$

where the summation is over all l-tuples $M_j^i = (m_1, m_2, \cdots, m_l)$ for which $l \leqq i$ and $i \geqq m_1 > m_2 > \cdots > m_l \geqq 1$, $m_1 - m_2 + m_3 - \cdots + (-1)^{l+1} m_l = j$.

Proof. Induction on i holding j fixed. Trivially, $a_{i,i} = K_i$ is the initialization.

These relations were first observed by the author in [9] and have subsequently proved to be useful in the detection of Gaussian autoregressive processes [11].

The factorization (3.3) can now be used to find upper and lower bounds on $\|(T_p)^{-1}\|_1$.

THEOREM 3.1. *With the quantities E_j, K_j as before,*

$$\max\left\{\frac{1}{E_{p-1}}, \frac{1}{\prod_{j=1}^{p-1}(1-K_j)}\right\} \leqq \left\|(T_p)^{-1}\right\|_1 \leqq \prod_{j=1}^{p-1}\frac{(1+|K_j|)}{(1-|K_j|)}. \tag{3.9}$$

Proof. Since $(1, \mathbf{a}'_{p-1})'/E_{p-1}$ is the first column of $(T_p)^{-1}$, it is seen that

$$\left|\frac{\prod_{j=1}^{p-1}(1+K_j)}{E_{p-1}}\right| \leqq |(1 + a_{p-1,1} + \cdots + a_{p-1,p-1})/E_{p-1}|$$

$$\leqq \left\|(T_p)^{-1}\right\|_1 = \left\|(T_p)^{-1}\right\|_\infty$$

by Lemma 3.3. Clearly,

$$E_{p-1} = \prod_{j=1}^{p-1}(1-K_j^2),$$

and so

$$\frac{\prod_{j=1}^{p-1}(1+K_j)}{E_{p-1}} = \frac{1}{\prod_{j=1}^{p-1}(1-K_j)},$$

while clearly

$$1/E_{p-1} \leqq \left(1 + \sum_{j=1}^{p-1}|a_{p-1,j}|\right)/E_{p-1} \leqq \left\|(T_p)^{-1}\right\|_1.$$

These results give the lower bound.

On the other hand, by (3.3),

$$\left\|(T_p)^{-1}\right\|_1 \leqq \|C_p\|_1 \left\|(D_p)^{-1}\right\|_1 \|C_p'\|_1 \leqq \prod_{j=1}^{p-1} (1+|K_j|)\|C_p\|_1/E_{p-1},$$

where Lemma 3.3 has been used to bound $\|C_p\|_1$ and (2.14) shows that $\left\|(D_p)^{-1}\right\|_1 = 1/E_{p-1}$. To bound $\|C_p'\|_1$, Lemma 3.4 is used. The jth column sum of C_p' satisfies

$$\begin{aligned}\left|1+\sum_{k=1}^{j-1} a_{p-k,j-k}\right| &\leqq 1+\sum_{k=1}^{j-1}\left|\sum_{M_{j-k}^{p-k}} \prod_{m\in M_{j-k}^{p-k}} K_m\right| \\ &\leqq 1+\sum_{k=1}^{j-1}\sum_{M_{j-k}^{p-1}} \prod_{m\in M_{j-k}^{p-1}} |K_m| \\ &\leqq 1+\sum_{k=1}^{p-1}\sum_{M_k^{p-1}} \prod_{m\in M_k^{p-1}} |K_m| \\ &= \prod_{m=1}^{p-1} (1+|K_m|).\end{aligned}$$

Putting this all together gives

$$\|C_p'\|_1 \leqq \prod_{j=1}^{p-1} (1+|K_j|),$$

so that

$$\left\|(T_p)^{-1}\right\|_1 \leqq \frac{\prod_{j=1}^{p-1}(1+|K_j|)^2}{E_{p-1}} = \prod_{j=1}^{p-1} \frac{(1+|K_j|)}{(1-|K_j|)}.$$

Theorem 3.1 shows that the condition of the Toeplitz system is guaranteed large if E_n is small. In turn, E_n is small if any of the partial correlation coefficients is large; that is, close to 1 in absolute value. Now E_n is a measure of how well the prediction scheme (2.3) works, so that a "desirable" situation is for E_n to be very small. Box and Jenkins [4] discourage the use of the Levinson-Durbin algorithm in the cases where E_n becomes small. The rationale is that the algorithm requires division by E_n, thereby allowing for the possibility of significant rounding error magnification. By examining the steps in the algorithm, we see that there is a real possibility that a rounding error committed at the jth stage becomes magnified by a factor of $1/(E_{j-1}E_j\cdots E_{n-1})$ by the nth stage.

Now if errors did indeed propagate as described above, the E_j would not have to be very small for this phenomenon to be noticed. For instance, if E_2 were on the order of 0.1 (which is common in speech applications [25]) rounding errors committed in the first two stages could be magnified by at least 10^{10} by the twelfth stage. This sort of behavior has never been observed in practice, and we shall see in the analysis of the next sections that propagated rounding errors cancel to an extent that such a potential geometric error growth is avoided.

On the other hand, if the values of the E_j do become very small, the underlying problem is ill-conditioned so that large errors in computed answers are expected and essentially unavoidable, regardless of which method is used to solve (1.3). Since

computational experience has revealed serious accuracy problems for small values of the E_j [4], [13], [22], [30], one suspects that a good deal of the error is due to the condition of the underlying system.

Thus past computational experience with the accuracy of the Levinson-Durbin algorithm does not contradict the main result of this paper, namely the stability of the Levinson-Durbin algorithm. As outlined above, the very nature of the context in which the Yule-Walker equations most often arise is one which suggests that the resulting system will usually be ill-conditioned.

4. The numerical stability of the Levinson-Durbin algorithm: Statement of results. In this section, the results of an error analysis of the Levinson-Durbin algorithm, whose derivation is in §5, will be presented.

As mentioned in §1, the stability properties of the Levinson-Durbin algorithm have not been well understood—a naive and gross analysis suggests serious accuracy problems, while in practice this has not been observed except in the case of ill-conditioned problems. The analytic results presented in this section show that the algorithm is stable. By stable, it is meant that errors due to rounding in the solution computed by the Levinson-Durbin algorithm are no worse than perturbations in the solution induced by modest perturbations of the initial data. This will be discussed at length after the results are presented.

First it is necessary to identify the nature of the local rounding errors. Both fixed-point and floating-point arithmetics will be considered. (Although fixed-point computations on general purpose computers and the corresponding error analyses are becoming rare, the fact that the algorithm is commonly implemented by dedicated hardware in fixed-point arithmetic makes a fixed-point analysis of some interest. It should be noted that the a priori dynamic range constraints on the various quantities occurring in the algorithm make the fixed-point realization convenient.)

For fixed-point arithmetic, the rounding error model is

$$\begin{aligned} \mathrm{fx}(a+b) &= a+b, \\ \mathrm{fx}(ab) &= (ab)+\xi, \\ \mathrm{fx}(a/b) &= (a/b)+\zeta, \end{aligned}$$

where a and b are fixed-point numbers and $\mathrm{fx}(\cdot)$ denotes the fixed-point representative of the argument, $\cdot$. Both ξ and ζ are in this case the local rounding errors, and satisfy $|\xi|, |\zeta| \leqq \Delta$, where Δ depends on the wordlength and method of truncation.

For floating-point arithmetic, it is assumed that

$$\begin{aligned} \mathrm{fl}(a+b) &= (a+b)(1+\xi), \\ \mathrm{fl}(a\,b) &= (a\,b)(1+\zeta), \\ \mathrm{fl}(a/b) &= (a/b)(1+\eta), \end{aligned}$$

where a and b are floating-point numbers and $\mathrm{fl}(\cdot)$ denotes the floating-point representative of the argument, $\cdot$. In this case it is assumed that $|\xi|, |\zeta|, |\eta| \leqq \Delta$, and Δ is again implementation dependent.

For a full treatment of the error models and terminology, the reader should consult [15], [28], [32] or any other standard text on numerical computing.

Now suppose that $\mathbf{a}_p$ is the true solution to (1.3). The computed solution is then $\bar{\mathbf{a}}_p = \mathbf{a}_p + \boldsymbol{\alpha}_p$ where $\boldsymbol{\alpha}_p$ is the perturbation due to the accumulated effect of the local

rounding errors. This computed solution will then obviously satisfy a perturbed system of equations,

$$\begin{aligned} T_p \bar{\mathbf{a}}_p &= T_p(\mathbf{a}_p + \boldsymbol{\alpha}_p) \\ &= T_p \mathbf{a}_p + \boldsymbol{\delta}_p \\ &= -\mathbf{r}_p + \boldsymbol{\delta}_p . \end{aligned}$$

The vector $\boldsymbol{\delta}_p$ is called the residual vector and satisfies

$$T_p \boldsymbol{\alpha}_p = \boldsymbol{\delta}_p, \qquad \boldsymbol{\alpha}_p = (T_p)^{-1} \boldsymbol{\delta}_p .$$

One can regard the computed solution $\bar{\mathbf{a}}_p$ as being the exact solution to the problem (1.3) but with the right side perturbed by the term $\boldsymbol{\delta}_p$. Notice that this does not represent a real world perturbation of the problem, since the entries of the coefficient matrix and the right side vector are related. The perturbation does make sense analytically, however.

The main result of this paper is summarized in the following theorem, whose proof is in §5.

THEOREM 4.1. *Suppose the Levinson-Durbin algorithm is used to compute the approximate solution,* $\bar{\mathbf{a}}_p$, *to* (1.3). *If fixed-point arithmetic is used, the residual satisfies*

$$\begin{aligned} (4.1) \qquad \|\boldsymbol{\delta}_p\|_1 &\leqq \sum_{j=1}^{p} (j^2+3j+1)\Delta \prod_{m=j+1}^{p} (1+|K_m|) + O(\Delta^2) \\ &\leqq \Delta \left[\prod_{j=1}^{p} (1+|K_j|) \right] \left(\frac{p^3}{3} + \frac{3}{2}p^2 + p \right) + O(\Delta^2). \end{aligned}$$

If floating-point arithmetic is used, the residual satisfies

$$(4.2) \qquad \|\boldsymbol{\delta}_p\|_1 \leqq \Delta \left(\frac{p^2}{2} + 11p \right) \left[\prod_{j=1}^{p} (1+|K_j|) - 1 \right] + O(\Delta^2).$$

The $O(\Delta^2)$ terms in (4.1) and (4.2) succinctly express the fact that only first order errors are considered important in the analysis. That second and higher order terms cannot become more significant than first order terms is argued in the derivation of the results.

These bounds on the residual vector indicate that they can only be large if the partial correlation coefficients, K_j, are large. Since it is known that $|K_j| < 1$, "large" means relatively close to 1, and the results of §3 show that the underlying problem is in that case close to being ill-conditioned.

These bounds indicate that the algorithm is stable. To see this, the results of Theorem 4.1 will be compared with corresponding results for the Cholesky method of solving (1.3), with floating-point arithmetic. The Cholesky algorithm (triangularization and back-substitution) is known to be stable [28], [32].

Assume that the residual vector corresponding to a solution computed by the Cholesky algorithm is denoted by $\boldsymbol{\delta}^*_p$. It can be shown then that for floating-point arithmetic,

$$(4.3) \qquad \|\boldsymbol{\delta}^*_p\|_1 \leqq O(p^2)\|T_p\|_1 \|\mathbf{a}_p\|_1 \Delta \leqq O(p^3)\|\mathbf{a}_p\|_1 \Delta.$$

Note that the residual must be proportional to the size of the exact solution, $\mathbf{a}_p$; this follows from an obvious scaling argument. Recalling the result of Lemma 3.3,

$$\|\mathbf{a}_p\|_1 \leqq \prod_{j=1}^{p} (1+|K_j|)-1,$$

and Lemma 3.1,

$$\sum_{j=1}^{p} a_{p,j} = \prod_{j=1}^{p} (1+K_j)-1,$$

shows that the Levinson-Durbin and Cholesky residual bounds are of comparable size if the partial correlation coefficients are all positive. Hence, the solutions computed in that case by either the Levinson-Durbin algorithm or the Cholesky algorithm will have residual vectors sharing bounds of the same size essentially. It is therefore analytically established that in this special situation the Levinson-Durbin algorithm provides solutions as accurate as computationally possible since the Cholesky algorithm is universally considered the most stable method of solving (1.3).

Analytically, not as much can be said about the case when the partial correlation coefficients are not all positive. It must be borne in mind that the derivation of (4.1) and (4.2) replaces all occurrences of the coefficients K_j with their absolute values in order to maintain inequality. This is of course analytically necessary, but realistically it is extremely pessimistic. Realistically, it is reasonable to expect (4.1) and (4.2) to remain valid (in the sense of order of magnitude measure) if the absolute values are removed from the partial correlation coefficients. In fact, Lemma 3.2, which is used in the derivation of the bounds, can only yield an equality if all the quantities involved are positive to begin with, and so the first order bounds of Theorem 4.1 cannot be exact unless all of the rounding errors and partial correlation coefficients are positive.

Thus the true sizes of the residuals are somewhat less than the bounds indicate; unfortunately, it seems analytically intractible to give tighter bounds and simultaneously maintain some a posteriori feel for the size of the residual.

These comparisons with the Cholesky method are no longer true for fixed-point arithmetic. In this case, the bound on the residual in a solution computed by the Levinson-Durbin algorithm is comparable to the Cholesky residual if the partial correlation coefficients are consistently large and positive. However, the Levinson-Durbin algorithm is usually implemented in fixed-point arithmetic only for very specialized applications, such as speech signal processing [19], [25]. Empirical studies of the sizes of the partial correlation coefficients and other data characteristics peculiar to such applications have determined acceptable hardware implementations.

Numerical simulations have shown that the bounds as stated in (4.1) and (4.2) are pessimistic. Residuals in actually computed problems are significantly smaller than the bounds indicate. Comparisons with the Cholesky method showed that residuals were always of the same order of magnitude, with Levinson-Durbin residuals actually smaller for the ill-conditioned class of problems where the partial correlation coefficients were consistently close to $+1$.

In light of these analytic and empirical results, it is concluded that the Levinson-Durbin algorithm is a stable algorithm for solving the Yule-Walker equations (1.3).

5. The numerical stability of the Levinson-Durbin algorithm: Details of the error analysis. In this section details of the error analysis which led to the conclusion that the Levinson-Durbin algorithm is stable will be presented. The analysis is for first

order errors; in other words, terms which are the products of errors will be discarded in the equations describing rounding error propagation.

For most of the following results the exact nature of the local rounding errors introduced by any given computation is not important; what is important is the way in which the errors propagate from intermediate quantity to intermediate quantity. Once it is evident how the first order errors propagate, it will be possible to return and identify the exact nature of the local errors. In this way both fixed-point and floating-point arithmetic implementations can be handled together.

The analysis consists of three major steps:

- the propagation of error from stage to stage;
- the recursive description of the first order errors in the residual vector;
- bounding the first order errors in the residual vector.

Let $\bar{K}_j$, $\bar{a}_{i,j}$, and $\bar{E}_j$ denote the computed values corresponding to the exact values in (2.10)–(2.12). They satisfy the following equations:

$$\bar{a}_{i,j}=a_{i,j}+\alpha_{i,j}, \tag{5.1}$$

$$\bar{K}_j=K_j+\kappa_j, \tag{5.2}$$

$$\bar{E}_j=E_j+\varepsilon_j. \tag{5.3}$$

The local errors are then defined through the perturbed versions of (2.10)–(2.12).

$$\bar{K}_i=-(\rho_i+\bar{a}_{i-1,1}\rho_{i-1}+\cdots+\bar{a}_{i-1,i-1}\rho_1)/E_{i-1}+\zeta_i, \tag{5.4}$$

$$\bar{a}_{i,j}=\bar{a}_{i-1,j}+\bar{a}_{i-1,i-j}\bar{K}_i+\xi_{i,j}, \tag{5.5}$$

$$\bar{a}_{i,i}=\bar{K}_i, \tag{5.6}$$

$$\bar{E}_i=(1-\bar{K}_i^2)\bar{E}_{i-1}+\eta_i. \tag{5.7}$$

Of course, if the local errors, ζ, ξ, η, are all identically zero, (2.10)–(2.12) and (5.4)–(5.7) are identical. Bearing in mind that the analysis is for first order errors, we can now derive the error propagation equations.

LEMMA 5.1. *With κ_i, ε_i, and α_{ij} as above,*

$$\kappa_i=-\left(\zeta_i E_{i-1}+K_i\varepsilon_{i-1}+\sum_{j=1}^{i-1}\rho_{i-j}\alpha_{i-1,j}\right)/E_{i-1}, \tag{5.8}$$

$$\alpha_{i,j}=\alpha_{i-1,j}+K_i\alpha_{i-1,i-j}+\kappa_i a_{i-1,i-j}+\xi_{ij}, \tag{5.9}$$

$$\alpha_{i,i}=\kappa_i, \tag{5.10}$$

$$\varepsilon_i=\sum_{j=1}^{i}\rho_j\alpha_{ij}+P_i, \tag{5.11}$$

where P_i is defined recursively by

$$P_i=P_{i-1}-\sum_{j=1}^{i-1}\rho_j\xi_{ij}+K_iE_{i-1}\zeta_i+\eta_i, \tag{5.12}$$

and

$$P_1=\eta_1.$$

Proof. The relations (5.9) and (5.10) are obvious. Using the Taylor expansion of $1/(1+x)$ together with (5.3) and (5.9)–(5.10) establishes (5.8). The only relation which is not entirely trivial, and which is really the heart of the matter since it in a sense precludes the possibility of the geometric error growth alluded to in §3, is (5.11). Expanding (5.7) gives rise to a term $2\,\kappa_i K_i E_{i-1}$ which can be expanded in two ways. From (5.8)

$$\kappa_i K_i E_{i-1} = -K_i E_{i-1}\zeta_i - K_i^2 \varepsilon_{i-1} - K_i \sum_{j=1}^{i-1} \rho_{i-j}\alpha_{i-1,j},$$

while from (2.10)

$$\kappa_i K_i E_{i-1} = -\kappa_i \rho_i - \kappa_i \sum_{j=1}^{i-1} \rho_{i-j} a_{i-1,j}.$$

Induction will then establish (5.11) and the recursive form of P_i.

The equations describing error propagation above show that the errors are intricately related; this is to be expected since the algorithm itself is recursive and relies heavily on the structure of the underlying Yule-Walker equations. Attempting to bound the errors directly at this point would lead to the extremely pessimistic error bounds alluded to in §3. These equations are dramatically decoupled by looking at the residual vector.

Recalling that $T_i \boldsymbol{\alpha}_i = \boldsymbol{\delta}_i$ is the residual for the ith order equations, we get from the above relations a simple recursive description of $\boldsymbol{\delta}_i$.

LEMMA 5.2.

$$\boldsymbol{\delta}_i = \begin{bmatrix} \boldsymbol{\delta}_{i-1} + K_i \hat{\boldsymbol{\delta}}_{i-1} \\ \\ \\ -\zeta_i E_{i-1} - K_i P_{i-1} \end{bmatrix} + T_i \begin{bmatrix} \xi_{i,1} \\ \vdots \\ \xi_{i,i-1} \\ 0 \end{bmatrix}. \tag{5.13}$$

Proof. The symmetries of T_i imply that

$$T_i \hat{\mathbf{a}}_i = -\hat{\mathbf{r}}_i, \qquad T_{i+1}\begin{pmatrix} \hat{\mathbf{a}}_i \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ E_i \end{pmatrix}, \tag{5.14}$$

so that

$$\begin{aligned} \boldsymbol{\delta}_i = T_i \boldsymbol{\alpha}_i &= T_i \begin{bmatrix} \alpha_{i-1,1} + K_i \alpha_{i-1,i-1} + \kappa_i a_{i-1,i-1} + \xi_{i,1} \\ \alpha_{i-1,2} + K_i \alpha_{i-1,i-2} + \kappa_i a_{i-1,i-2} + \xi_{i,2} \\ \vdots \\ \kappa_i \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\delta}_{i-1} + K_i \hat{\boldsymbol{\delta}}_{i-1} \\ \\ \\ E_{i-1}\kappa_i + \sum_{j=1}^{i-1} \alpha_{i-1,j}\rho_{i-j} + K_i(\varepsilon_{i-1} - P_{i-1}) \end{bmatrix} + T_i \begin{bmatrix} \xi_{i,1} \\ \vdots \\ \xi_{i,i-1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\delta}_{i-1} + K_i \hat{\boldsymbol{\delta}}_{i-1} \\ \\ \\ -\zeta_i E_{i-1} - K_i P_{i-1} \end{bmatrix} + T_i \begin{bmatrix} \xi_{i,1} \\ \vdots \\ \xi_{i,i-1} \\ 0 \end{bmatrix}. \end{aligned}$$

Now it should be noticed that the recursion relating $\boldsymbol{\delta}_i$ to $\boldsymbol{\delta}_{i-1}$ is very similar to the recursion in Lemma 3.2, but the lemma cannot be used directly on account of the extraneous terms in (5.13). It is however a simple matter to decompose $\boldsymbol{\delta}_i$ into a sum of vectors which are of the same form as in Lemma 3.2.

Examination of (5.13) shows that $\boldsymbol{\delta}_i$ contains local errors which were already present in $\boldsymbol{\delta}_{i-1}$ together with local errors which were not, namely ζ_i, η_{i-1}, and $\xi_{i,j}$ for $j=1,\ldots,i-1$. Define $\boldsymbol{\psi}_{j,j}$ to be the vector of local errors in $\boldsymbol{\delta}_j$ which did not appear in $\boldsymbol{\delta}_{j-1}$ but which do appear in $\boldsymbol{\delta}_j$. Furthermore, let $\boldsymbol{\psi}_{i,j}$ be the vector of those errors as they then appear in $\boldsymbol{\delta}_i$. By Lemma 5.2,

$$\boldsymbol{\psi}_{j,j}=T_j\begin{bmatrix}\xi_{j,1}\\ \vdots\\ \xi_{j,j-1}\\ 0\end{bmatrix}+\begin{bmatrix}0\\ \\ \\ -\zeta_j E_{j-1}-K_j\eta_{j-1}\end{bmatrix}, \tag{5.15}$$

and for $i>j$

$$\boldsymbol{\psi}_{i,j}=\begin{bmatrix}\boldsymbol{\psi}_{i-1,j}+K_i\hat{\boldsymbol{\psi}}_{i-1,j}\\ K_i\left(\sum_{m=1}^{j-1}\rho_m\xi_{j,m}-K_jE_j\zeta_j-\eta_{j-1}\right)\end{bmatrix}. \tag{5.16}$$

Clearly $\boldsymbol{\delta}_i=\sum_{m=1}^{i}\boldsymbol{\psi}_{i,m}$, and so $\|\boldsymbol{\delta}_i\|_1\leqq\sum_{m=1}^{i}\|\boldsymbol{\psi}_{i,m}\|$ for any vector norm, so all that remains to be done is the bounding of each of the $\boldsymbol{\psi}_{i,j}$. In fact, Lemma 3.2 is now directly applicable and gives the following result.

LEMMA 5.3.

$$\|\boldsymbol{\psi}_{i,j}\|\leqq(r_j+s_j)\prod_{m=j+1}^{i}(1+|K_m|)-s_j,$$

where $r_j\geqq\|\boldsymbol{\psi}_{j,j}\|$ *and*

$$s_j\geqq\left|\sum_{m=1}^{j-1}\rho_m\xi_{j,m}-K_jE_{j-1}\zeta_j-\eta_{j-1}\right|. \tag{5.17}$$

Finally by identifying the exact natures of the local rounding errors, it is possible to derive (4.1) and (4.2) in Theorem 4.1.

Consider floating-point arithmetic first. Clearly, $|\xi_{j,m}|\leqq\Delta|a_{j,m}|+\Delta|K_j||a_{j-1,j-m}|$, so that

$$\sum_{m=1}^{j-1}|\xi_{j,m}|\leqq 2\Delta\prod_{m=1}^{j}(1+|K_m|)-1,$$

and

$$\left\|T_j\begin{bmatrix}\xi_{j,1}\\ \vdots\\ \xi_{j,j-1}\\ 0\end{bmatrix}\right\|\leqq 2\Delta j\left[\prod_{m=1}^{j}(1+|K_m|)-1\right].$$

Furthermore,

$$E_{j-1}|\zeta_j| \leqq 2|K_j|\Delta + j\Delta\left[\prod_{m=1}^{j-1}(1+|K_m|)-1\right]$$

and $|K_j|\cdot|\eta_{j-1}| \leqq 3|K_j|\Delta$. Thus

$$r_j = (3j+5)\Delta\left[\prod_{m=1}^{j}(1+|K_m|)-1\right]$$

bounds $\|\psi_{j,j}\|$. The same reasoning shows that

$$s_j = 3\Delta + (j+3)\Delta\left[\prod_{m=1}^{j}(1+|K_m|)-1\right]$$

works in (5.17). Now

$$\begin{aligned}(r_j+s_j)&\prod_{m=j+1}^{p}(1+|K_m|)-s_j\\ &\leqq\left(3\Delta+(4j+8)\Delta\left[\prod_{m=1}^{j}(1+|K_m|)-1\right]\right)\prod_{m=j+1}^{p}(1+|K_m|)-3\Delta\\ &\leqq(4j+11)\Delta\left[\prod_{m=1}^{p}(1+|K_m|)-1\right],\end{aligned}$$

and (4.2) follows by summing over j.

The result for fixed-point arithmetic follows in essentially the same way, but with $E_{j-1}|\zeta_j| \leqq \Delta$, $|\xi_{i,j}| \leqq \Delta$, $|\eta_i| \leqq 2\Delta$, and

$$\|\psi_{j,j}\| \leqq j^2\Delta + (j+1)\Delta = r_j, \qquad 2j\Delta = s_j.$$

Thus putting these together we get

$$\begin{aligned}\sum_{j=1}^{p}(r_j+s_j)&\prod_{m=j+1}^{p}(1+|K_m|)-s_j \leqq \sum_{j=1}^{p}(j^2+3j+1)\Delta\prod_{m=j+1}^{p}(1+|K_m|)\\ &\leqq\Delta\left(\frac{1}{3}p^3+\frac{3}{2}p^2+p\right)\prod_{m=1}^{p}(1+|K_m|).\end{aligned}$$

A few words are appropriate about the fact that this analysis is for first order error only. At any given stage of the algorithm the new second and higher order error terms can be absorbed by the local error, after which they propagate in exactly the same way as do the local first order errors. It is straightforward but tedious to show that the new second order errors entering in the computation of $\mathbf{a}_j$ are smaller than the corresponding first order local errors so long as all first order relative errors remain less than $\sqrt{\Delta}$. Thus these second order errors contribute only if the first order errors are becoming significant already.

The same argument applies to all computations in the algorithm save the use of only the first term in the Taylor expansion of $1/(1+x)$ in Lemma 5.1. That this is allowable is based on the result of Theorem 3.1; the condition of the underlying problem multiplied by Δ must be small in any case.

This error analysis does not fit the pattern of either of the traditional approaches; namely the forward and backward types. It might seem that a strict backward analysis of the algorithm would be suitable because of the way in which the problem parameters enter the recursive computation. Unfortunately, there seems to be no way to avoid exponential error bounds using that approach. It is evident that the various errors in the computed intermediate quantities are intricately related and the analysis presented in this paper has taken advantage of the relationships to simplify bounds on the residual error. A strict forward analysis would not achieve that goal.

6. Summary. This paper has addressed both the conditioning of Toeplitz linear systems of equations and the numerical stability of the Levinson-Durbin algorithm for solving the special case of the Yule-Walker equations.

Upper and lower a posteriori bounds were derived for the condition number of a positive-definite symmetric Toeplitz matrix. These bounds showed that Toeplitz systems arising in applications where the "prediction error" is known to be very small are guaranteed to be ill-conditioned. The fact that situations most appropriately modeled by such systems are consequently ill-conditioned seems to account for the poor accuracy of computed solutions as observed in past experiments. It seems that the Levinson-Durbin algorithm has been unjustly held responsible for the large errors observed.

An error analysis of the Levinson-Durbin algorithm produced upper bounds on the residual vector for both fixed-point and floating-point arithmetics. The first order error terms in these bounds are virtually identical to the bounds for the Cholesky algorithm residual in a large class of special problems. In the general case it is evident that the residual for both methods should be the same order of magnitude. Since the Cholesky algorithm is known to be extremely stable, it is concluded that the Levinson-Durbin algorithm is numerically stable.

The numerical stability of the Levinson algorithm for the general system (1.4) and the Trench algorithm for inverting positive-definite symmetric Toeplitz matrices can be investigated using the same ideas presented in this paper. Those results will appear in a forthcoming paper.

REFERENCES

[1] H. AKAIKE, *Block Toeplitz matrix inversion*, SIAM J. Appl. Math., 24 (1973), pp. 234–241.

[2] O. B. ARUSHANIAN ET AL., *The* TOEPLITZ *Package: User's Guide*, Technical Report, Argonne National Laboratory, 1979.

[3] R. R. BITMEAD AND B. D. O. ANDERSON, *Asymptotically fast solutions of Toeplitz and related systems of linear equations*, Technical Report No. EE7915, Department of Electrical Engineering, University of Newcastle, Australia, 1979.

[4] G. E. BOX AND G. M. JENKINS, *Time Series Forcasting and Control*, Holden-Day, San Francisco, 1970.

[5] R. P. BRENT, F. G. GUSTAFSON, and D. Y. Y. YUN, *Fast computation of Padé approximants and the solution of Toeplitz systems of equations*, IBM Report, IBM T. J. Watson Research Center, to appear.

[6] R. K. BROUWER, *Particular methods for solving particular systems of linear equations*, CSRR-2046, Department of Applied Analysis and Computer Science, University of Waterloo, Ontario, 1971.

[7] J. R. BUNCH ET AL., *LINPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[8] J. J. CORNYN JR., *Direct methods for solving systems of linear equations involving Toeplitz or Hankel matrices*, Naval Research Laboratory Memorandum Report 2920, Washington, DC, 1974.

[9] G. CYBENKO, *Error Analyses of Some Signal Processing Algorithms*, Ph.D. Thesis, Princeton University, Princeton, NJ, 1978.

[10] B. W. DICKINSON, *Solution of linear equations with rational Toeplitz matrices*, Math. Comp., 34 (1980), pp. 227–233.
[11] B. W. DICKINSON, *Properties and applications of Gaussian autoregressive processes*, preprint.
[12] J. DURBIN, *The fitting of time-series models*, Rev. Int. Inst. Statist., 28 (1960), pp. 233–243.
[13] C. W. J. GRANGER, *Forecasting Economic Time Series*, Academic Press, New York, 1977.
[14] U. GRENANDER and G. SZEGO, *Toeplitz Forms and their Applications*, University of California Press, Berkeley, 1958.
[15] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.
[16] T. KAILATH, S. Y. KUNG, AND M. MORF, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407.
[17] N. LEVINSON, *The Weiner* RMS *(root mean square) error criterion in filter design and prediction*, J. Math. Phys., 25 (1947), pp. 261–278.
[18] J. MAKHOUL, *Linear prediction: a tutorial review*, Proc. IEEE, 63 (1975), pp. 561–580.
[19] J. MARKEL AND A. GRAY JR., *Fixed-point truncation arithmetic implementation of a linear prediction autocorrelation vocoder*, IEEE Trans. Acoust. Speech Signal Process., 22 (1974), pp. 273–281.
[20] M. MORF, *Doubling algorithms for Toeplitz and related equations*, preprint.
[21] A. V. OPPENHEIM, *Applications of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
[22] M. PAGANO, *An algorithm for fitting autoregressive schemes*, J. Roy. Statist. Soc. Ser. C, 21 (1972), pp. 274–281.
[23] M. H. QUENOUILLE, *The joint distribution of serial correlation coefficients*, Ann. Math. Statist., 20 (1949), pp. 561–571.
[24] L. R. RABINER AND B. GOLD, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs NJ, 1975.
[25] L. R. RABINER AND R. SCHAFER, *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs NJ, 1978.
[26] J. RISSANEN, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with applications to factoring positive matrix polynomials*, Math. Comp., 27 (1973), pp. 147–154.
[27] E. A. ROBINSON, *Statistical Communication and Detection*, Hafner, New York, 1967.
[28] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
[29] W. F. TRENCH, *An algorithm for the inversion of finite Toeplitz* matrices, J. Soc. Indust. Appl. Math., 12 (1964), pp. 515–522.
[30] J. W. TUKEY, *An introduction to the calculations of numerical spectrum analysis*, in Spectral Analysis of Time Series, B. Harris, ed., John Wiley, New York, 1967, pp. 25–46.
[31] G. WALKER, *On periodicity in series of related terms*, Proc. Royal Soc. London Ser. A, 131A (1931), p. 518.
[32] J. W. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs NJ, 1963.
[33] A. S. WILLSKY, *Digital Signal Processing and Control: Points of Tangency, Areas of Intersection, and Parallel Directions*, MIT Press, Cambridge MA, 1979.
[34] G. U. YULE, *On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers*, Philos. Trans. Roy. Soc. London Ser. A, 226A (1927), pp. 267–298.
[35] S. ZOHAR, *Toeplitz matrix inversion: the algorithm of W. F. Trench*, J. Assoc. Comput. Mach., 16 (1969), pp. 592–601.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0002 $01.00/0

ANALYTIC SUBTRACTION APPLIED TO THE INCOMPLETE GAMMA AND BETA FUNCTIONS*

MARIETTA J. TRETTER† AND G. W. WALSTER‡

Abstract. The purpose of this paper is to draw attention to a method of performing analytic subtractions that can dramatically improve the numerical stability of a continued fraction (cf) or series expansion. The method is applied to the computation of the incomplete gamma and beta functions.

Key words. analytic subtraction, incomplete gamma function, incomplete beta function, continued fractions

1. Introduction. A series or continued fraction (cf) that converges rapidly must also be numerically stable for it to be computationally useful. The most dramatic source of numerical instability is cancellation due to numerical subtractions (Stoutemyer [11]). Unfortunately, there is neither a general analytic method for uncovering the existence of bad subtractions, nor a single method for eliminating them. Moreover, normalized floating point computer arithmetic gives no indication that bad subtractions have occurred. With the forthcoming hardware implementation of floating point standards and the improved software implementation of interval arithmetic (personal communications with W. Kahan, Intel. Corp. and L. Liddiard, U. of Minnesota), we will at long last be able to monitor automatically the numerical stability of all calculations.

This paper presents one method of resurrecting an otherwise numerically unstable series or cf: "analytic subtraction". Analytic subtraction can be implemented in many ways. For example, rearranging an expression may eliminate the bad subtractions. In all cases, however, the goal of analytic subtraction is to obtain an expression in which the offending terms can be analytically cancelled out, leaving a numerically stable result.

To illustrate with a very simple example, assume we wish to compute $x-y$, where $x=a+b$, $y=a+c$, $a=1\times 10^{50}$, $b=2$ and $c=1$. If we first compute x and y and then subtract, we will get an answer of 0 on any current fixed wordlength computer. However, if we analytically cancel out a, we obtain the correct result, $b-c=2-1=1$.

The analytic subtraction we will present in subsequent sections is slightly more subtle than the above example and has been used elsewhere ([2], [5], [13]). However, we have found it to be so important that it deserves more prominent exposure.

2. Problem. Suppose we wish to compute the function $f(x)=1-g(x)+h(x)$, where x is in that part of the domain of $f(x)$ in which $g(x)\cong 1$ and $f(x)\cong h(x)\ll 1$. Clearly, if we attempt to compute $f(x)$ directly, total loss of significant digits may result. Thus, we need to extract 1 from $g(x)$ so that 1 and -1 can be analytically cancelled out.

3. Solution. If a representation for the natural logarithm in $g(x)$ can be found that will allow its significant digit computation, the following algorithm can be used to

*Received by the editors May 8, 1979, and in final revised form May 5, 1980. This paper acknowledges the use of the MACSYMA system at the Massachusetts Institute of Technology.

†Division of Management Science, Pennsylvania State University, University Park, Pennsylvania 16802.

‡Control Data Corporation, HQNOGV, P.O. Box O, Minneapolis, Minnesota 55440.

evaluate $1-g(x)$:

$$1-g(x)=1-\exp\{\ln g(x)\}=-\sum_{j=1}^{\infty}\frac{u^j}{j!},$$

where

$$u=\ln g(x).$$

The series in u will be strictly decreasing and rapidly converging, as $g(x)\cong 1$ implies $|u|\cong 0$.

Thus,

$$f(x)=-\sum_{j=1}^{\infty}\frac{u^j}{j!}+h(x).$$

4. Example 1. The incomplete gamma function. Suppose we wish to evaluate the incomplete gamma function:

$$\begin{aligned}Q_r(x|\alpha)&=1-P(x|\alpha)\\&=1-\frac{1}{\Gamma(\alpha)}\int_0^x t^{\alpha-1}e^{-t}dt\\&=1-\frac{x^\alpha}{\Gamma(1+\alpha)}-\frac{1}{\Gamma(\alpha)}\sum_{j=1}^{\infty}\frac{(-1)^j x^{\alpha+j}}{(\alpha+j)j!}.\end{aligned}$$

The series is a normalized confluent hypergeometric. If significant digits are required, it is disastrous to perform the subtraction numerically when, for example, $P(10^{-4}|10^{-3})=.9913$, or $P(10^{-10}|10^{-5})=.9997$, etc.

Luke [10, Chapt. 14] devotes several sections to computing $Q_r(x|\alpha)$ with $\alpha\leqq 1$. The cf for $Q_r(x|\alpha)$ given by Luke (essentially the same cf for $Q_r(x|\alpha)$ presented in Wall [14, p. 356]), is slowly converging when x is small. Luke also includes the cf representation of the even and odd parts of the original cf which, he points out, occupy the n and $n-1$ entries in the Padé table (Luke [10, p. 202]). While these even and odd cf's will obviously converge faster than the original cf, the improvement is not enough for small values of x. For the even cf the approximate number of terms, n, required to obtain k decimal digits is $n\sim .33k^2/x$ (Henrici [7, p. 629]). Thus, an alternative method for computing $Q_r(x|\alpha)$ is necessary when x and α simultaneously approach zero.

If we use the power series expansion for $\ln\Gamma(1+\alpha)$, we can write

$$u=\ln\left[\frac{x^\alpha}{\Gamma(1+\alpha)}\right]=\alpha\ln x+a\gamma-\sum_{n=2}^{\infty}(-1)^n\zeta(n)\frac{\alpha^n}{n},$$

where γ is Euler's constant and $\zeta(n)$ is Riemann's zeta function. Thus, we have at once:

$$Q_r(x|\alpha)=-\sum_{n=1}^{\infty}\frac{u^n}{n!}-\frac{1}{\Gamma(\alpha)}\sum_{j=1}^{\infty}\frac{(-1)^j x^{\alpha+j}}{(\alpha+j)j!},$$

which is numerically stable provided both $|u|$ and x are less than $\frac{1}{2}$. A somewhat similar approach is used by DiDonato and Hageman [5] as part of an algorithm for computing the incomplete gamma function to 12 significant digits when $\alpha<\frac{1}{2}$ and $P_r(x|\alpha)>.9$.

5. Example 2. The incomplete beta function. Suppose we want to evaluate the incomplete beta function:

$$I_x(p,q)=B(p,q)^{-1}\int_0^x u^{p-1}(1-u)^{q-1}du.$$

The associated cf derived from Mueller's corresponding cf ([1, §26.5.9], Gautschi [6], Tretter and Walster [13]) is

$$I_x(p,q)=C\left\{1+\frac{k_1w}{|1+l_1w+}\,\frac{k_2w^2}{|1+l_2w+}\,\frac{k_3w^2}{|1+l_3w+}\cdots\right\}, \tag{5.1}$$

where

$$C=\frac{x^p(1-x)^{q-1}}{pB(p,q)},$$

$$w=\frac{x}{(1-x)},$$

$$k_1=\frac{(q-1)}{(p+1)},$$

$$k_s=\frac{(s-1)(p+q+s-2)(p+s-1)(q-s)}{(p+2s-3)(p+2s-2)^2(p+2s-1)},$$

$$l_1=-\frac{(q-2)}{(p+2)},$$

$$l_s=\frac{(p+s-1)(s-q)}{(p+2s-2)(p+2s-1)}+\frac{s(p+q+s-1)}{(p+2s-1)(p+2s)}.$$

This cf has the desirable qualities that allow it to be the basis of a significant digit algorithm for almost any combination of all parameters. However, in its present form, this cf has very limited computational utility.

Even assuming that both x and $(1-x)$ are explicitly available, thereby eliminating the necessity of computing $1-x$, there still exist cases in which the above algorithm is numerically unstable when implemented using the difference equations of the well-known forward recursion algorithm. For example,

$$p=.5\times10^{20},\quad q=.5\times10^4\quad\text{and}\quad x=.99999999999999998863636363636,$$

introduces so much cancellation from numerical subtractions that all significant digits are lost before convergence is achieved. The source of these subtractions is the partial denominators $1+l_sw$. However, if we rewrite the partial denominators in terms of the variable $F=qx/p(1-x)$ (the F random variable of statistics) then expand and simplify, numerous common terms analytically cancel, leaving:

$$1+\frac{l_sx}{(1-x)}=1+\frac{l_spF}{q}$$

$$=\frac{\left[2pFs^2+4qs(s-1)+2p(p-1)Fs+2pq(2s-1)+p^2q(1-F)\right]}{\left[q(p+2s-2)(p+2s)\right]}.$$

These simplifications, if performed by hand, are extremely cumbersome. The results here were easily obtained in one step using the powerful algebraic manipulation system of the MACSYMA system at the Massachusetts Institute of Technology.

With the elimination of these subtractions, the associated cf gives rather astounding results. For the above example, 25 significant digits are obtained with only 21 convergents.

There is still another possible bad subtraction. In computing $I_x(p, q)$, we always have the option of employing the well-known reflection identity: $I_x(p, q) = 1 - I_{(1-x)}(q, p)$. The question then arises, when should we reflect? Two considerations are relevant: efficiency and cancellation.

It can be shown that the absolute value of the difference in successive convergents of Mueller's *corresponding* cf (Tretter and Walster [13]) form a bound on the error of Mueller's *associated* cf. The terms of the corresponding cf are in a form appropriate for the application of a theorem due to Thron [12]. Thron's theorem can be used to obtain an overall measure, M, of the convergence rate of the reflected and unreflected forms.

Thron's theorem [12] states:

The continued fraction $K(a_n/1)$ *converges to a value* u *which satisfies* $\Re(u) \geqq -\frac{1}{2}$ *provided that* $\Im^2(a_n) \leqq \Re(a_n) + \frac{1}{4}$ *and* $|a_n| < M$ *for all* $n \geqq 1$, *where* $K(a_n/1)$ *is the continued fraction* $\dfrac{a_1}{1+}\ \dfrac{a_2}{1+}\ \dfrac{a_3}{1+}\ \cdots$.

Applying Thron's theorem to Mueller's corresponding cf, the following rules are obtained for determining M for the unreflected form:

$$\text{When } p > q, \qquad M = \frac{Cx}{1-x},$$

$$p < q \text{ and } q \geqq 2, \qquad M = \frac{Cx(q-1)}{(1-x)(p+1)},$$

$$p < q \text{ and } q < 2, \qquad M = \frac{Cx(p+q)}{(1-x)(p+1)(p+2)}.$$

The rule for obtaining the reflected M, M', say, can be obtained from the rules for M by interchanging p, q, x, $(1-x)$ in the above (including the coefficient C). It is appropriate to consider convergence of the corresponding cf, as it essentially converges at half the rate of the associated cf. From Thron's theorem, we obtain a bound on the value region of the cf, R_n, where the value region is in the form of nested circles within which the value of the cf lies:

$$R_n \leqq \frac{1}{\prod_{v=1}^{n} \left(1 + \dfrac{1}{4Mv}\right)}.$$

Relatively larger values of M are associated with overall slower convergence.

The ratio $M'/M = R$ can then serve as a test for when to reflect; i.e., if $R < 1$ then reflect. For example, let $p = 1{,}000$, $q = 100$ and $F = 2.0$. Then $R = .012$, which implies we should reflect. The reflected form requires about 36 convergents, while the unreflected requires about 100 convergents, for a given number of digits.

Now, there remains a potential problem. Suppose that the reflected form is more efficient, but that $I_{(1-x)}(q, p) \cong 1$. Then we will have gained efficiency at the cost of losing digits due to subtraction. The only time this event can occur is when the beta density function is "J" shaped; i.e., $p > 1$ and $q < 1$. For example, let $p = 999.5$, $q = .1 \times 10^{-21}$ and $F = .9$. Then $R \approx 10^{-47}$, implying that we should most surely reflect. But $I_x(p, q) = .55 \times 10^{-22}$. Using the unreflected cf is not possible because all digits

are lost to subtraction in the forward recursion algorithm. Fortunately, in those cases where reflection is required and $I_x(p, q)$ is small, the coefficient of the reflected form $C' \cong 1$. Thus, evaluating $u = \ln C'$ we can perform the subtraction analytically, yielding

$$I_x(p,q) = -\sum_{j=1}^{\infty} \frac{u^j}{j!} - C' \left\{ \frac{k_1' w'}{|1 + l_1' w' +} \; \frac{k_2' w'}{|1 + l_2' w' +} \cdots \right\},$$

where k_s', l_s' and w' have the same definition as k_s, l_s and w in (5.1) with x, $(1-x)$ and p, q interchanged. For the above example, 23 significant digits were obtained with only 6 convergents of the cf.

It must be noted that extreme care is necessary in evaluating $\ln C'$ to eliminate further sources of bad subtractions. One cannot simply use a computer library program to compute $\ln C'$ because this may conceal subtraction error in the logarithm computation. A better method of computing $\ln C'$ is to use a combination of series expansions, i.e., writing

$$C' = \frac{(1-x)^q x^{p-1} \Gamma(p+q)}{\Gamma(q+1)\Gamma(p)};$$

then computing $\ln C'$ can be viewed as the problem of computing $\ln[\Gamma(p+q)/\Gamma(p)]$, $\ln(x^{p-1})$, $\ln[(1-x)^q]$, and $\ln \Gamma(q+1)$. Computing $\ln \Gamma(q+1)$ was discussed in § 4.

In computing $\ln[\Gamma(z+a)/\Gamma(z)]$, in general, it is necessary once again to use analytic subtraction. If Stirling's well-known asymptotic formula is used separately to compute $\ln \Gamma(z+a)$ and $\ln \Gamma(z)$, subtraction of the two results can be devastating when z is large relative to a. Thus, the formulas must be combined to perform the subtraction analytically.

To compute $\ln x$ and $\ln(1-x)$, we can transform the beta random variable x, and $(1-x)$, into an F random variable ([1], §26.5.28) and consider the expansion of the form $\ln[a/(a+b)]$, which allows computation of $\ln x$, $\ln(1-x)$ with relative error.

The details of deriving the several expansions, and corresponding error bounds, used in computing $\ln C'$ are not particularly difficult; however, they are rather lengthy and thus are not presented here. The expansion approach outlined above does eliminate most bad subtractions. A perfect expansion which eliminates all bad subtractions, even in very extreme cases, remains to be found.

6. Conclusions. Without analytic subtraction, it is difficult to obtain robust algorithms for the significant digit computation of the incomplete gamma and beta functions, especially when a great number of digits are required. In the absence of robust formulas with all positive terms, analytic subtraction should be considered as a possible means of salvaging an otherwise useless formula. This technique has proved to be especially useful in statistical computations involving cumulative distribution functions for extreme values.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, eds., *Handbook of Mathematical Functions*, Dover, New York, 1968.

[2] D. E. AMOS, *Significant digit incomplete beta ratios*, Sandia Laboratory Report SC-DR-69-591, 1969.

[3] L. A. AROIAN, *Continued fractions for the incomplete beta function*, Ann. Math. Statist., 12 (1941), pp. 218–233.

[4] _____, *Corrections to continued fractions for the incomplete beta function*, Ann. Math. Statist., 30 (1959), p. 1265.
[5] A. R. DIDONATO AND R. K. HAGEMAN, *Computation of the incomplete gamma function ratios*, Naval Surface Weapons Center Tech. Report No. TR-3482, April, 1976.
[6] W. GAUTSCHI, *Computational aspects of three-term recurrence relations* SIAM Rev., 9 (1967), pp. 24-82.
[7] P. HENRICI, *Applied and Computational Complex Analysis*, Vol. II, John Wiley, New York, 1977.
[8] N. L. JOHNSON AND S. KOTZ, *Continuous Univariate Distributions*-1, John Wiley, New York, 1970.
[9] Y. L. LUKE, *The Special Functions and their Approximations, Vol. I*, Academic Press, New York, 1969.
[10] _____, *The Special Functions and their Approximations, Vol. II*, Academic Press, New York, 1969.
[11] D. R. STOUTEMYER, *Automatic error analysis using computer algebraic manipulation*, ACM Trans. Math. Software, 3 (1977), pp. 26-43.
[12] W. J. THRON, *Convergence regions for continued fractions and other infinite processes*, Amer. Math. Monthly, 68 (1961), pp. 734-750.
[13] M. J. TRETTER AND G. W. WALSTER, *Continued fractions for the incomplete beta function: Additions and corrections*, Ann. Statist., 7 (1979), pp. 462-465.
[14] H. S. WALL, *Analytic Theory of Continued Fractions*, Chelsea, New York, 1948.
[15] J. W. WRENCH, *Concerning two series for the gamma function*, Math. Comp., 22 (1968), pp. 617-626.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0003 $01.00/0

ERROR GROWTH IN USING THE RANDOM CHOICE METHOD FOR THE INVISCID BURGERS EQUATION*

JAMES LAVITA†

Abstract. This note describes some numerical experiments assessing the rate of error growth when using the Random Choice Method (RCM) to compute solutions to the inviscid Burgers equation. The RCM was developed and used to study solutions of various gas dynamical systems by Chorin and his collaborators (J. Comp. Phys., 22 (1976), pp. 517–533; 25 (1977), pp. 252–272; P. Colella, Ph.D. dissertation, U. California, Berkeley, 1979). Recently Colella has derived estimates for the error behavior of the Burgers equation. We assess the error numerically and see whether the bounds derived by Colella are achieved. Our conclusion is that the RCM does much better than expected and, indeed, approaches the best limits obtainable.

Key words. Random Choice Method, Burgers equation, gas dynamics

1. Introduction. The purpose of this note is to describe some numerical experiments assessing the rate of error growth when using the Random Choice Method (RCM) to compute solutions to the inviscid Burgers equation

$$u_t+\left(\frac{u^2}{2}\right)_x=0. \tag{1.1}$$

The RCM was developed and used to study solutions of various gas dynamical systems by Chorin and his collaborators [1], [2], [3]. Recently Colella has derived estimates for the error behavior of the Burgers equation as a model for the larger systems which he studies. We are concerned with assessing the error numerically and seeing whether the bounds derived by Colella are achieved. Our conclusion is that the RCM does much better than expected, and, indeed, approaches the best limits obtainable.

2. Background. The RCM has been described in detail in several publications [1], [2], [3], and we refer the interested reader to these more extensive and generally excellent references, particularly the essay of Sod [7]. A brief description is given here to stimulate the interest of those unfamiliar with the method.

In Fig. 1, observe the set-up for a 2-step scheme, $nk,(n+1)k$, $ih,(i+1)h$ being the full step mesh points in time and space respectively, and $(n+\frac{1}{2})k$, $(i+\frac{1}{2})h$ being intermediate steps. We are given or have computed values at the nth step. Now proceed as follows.

Construct the step function $u(x, nk)$:

$$u(x,nk)=\begin{cases} u_{i+1}^n, & x\geqq\left(i+\frac{1}{2}\right)h, \\ u_i^n, & x<\left(i+\frac{1}{2}\right)h, \end{cases}$$

u_j^n being known values. Solve the Riemann problem with this step function as data, getting a function $u(x,(n+\frac{1}{2})k)$. Now set

$$u_{i+\frac{1}{2}}^{n+\frac{1}{2}}=u\left(x+\theta_n,\left(n+\tfrac{1}{2}\right)k\right),$$

*Received by the editors December 5, 1979, and in revised form August 5, 1980. This research was partially supported by the Engineering, Mathematical, and Geosciences Division of the U.S. Department of Energy under contract W-7405-ENG-48, and by the Office of Naval Research under contract N00014-76-0-0316.

†Department of Mathematics, University of Denver, Denver, Colorado 20208. This research was carried out while the author was on leave at the Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720.

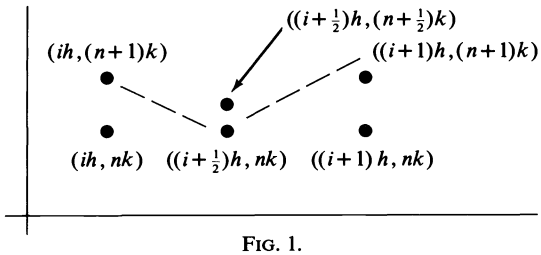


FIG. 1.

where $\theta_n \in [0,1]$ is an equidistributed quasirandom sequence of real numbers. Then one goes onto the $(n+1)th$ step in a similar fashion. One must avoid interacting shocks by invoking the Courant–Friedrichs–Lewy condition, and one should also pick the sequence θ_n wisely.

Colella derives the error bounds in his paper [3] by using the van der Corput sequence, and it can be shown that other sequences give less accurate results. The van der Corput sequence, a_n, is found as follows: Let

$$\sum_{k=0}^{m} i_k 2^k = n$$

be the binary expansion of the integer n. Then

$$a_n = \sum_{k=0}^{m} i_k 2^{-(k+1)}.$$

This sequence is quasirandom.

The method is applicable to the equation

$$u_t + (f(u))_x = 0, \tag{2.1}$$

where f need not be convex as long as one can solve the Riemann problem. We are using the model equation $f(u)=u^2/2$ because some analytical error bounds exist for certain specific types of interactions and the exact solutions can be calculated with reasonable efficiency. Examples of error estimates are (see [3]):

Shock interacting with compression wave:

$$\varepsilon_{L_1}(h,t) \sim C_1 e^{K_1(t-t_0)} h^{\frac{1}{2}} |\log h| + C_2 h |\log h|, \tag{2.2.1}$$

ε_{L_1} being the L_1-error and t_0 being the time when the shock strikes the wave.

Shock interacting with a rarefaction:

$$\varepsilon_{L_1}(h,t) \sim C_1 (t-t_0) h^{\frac{1}{2}} |\log h| + C_2 |\log h|. \tag{2.2.2}$$

Smooth region:

$$\varepsilon_{L_1}(h,t) \sim Ch |\log h|. \tag{2.2.3}$$

Here $\varepsilon_{L_1}(h,t)$ means $\Sigma_{\text{all } i} |u(ih,t) - u_i^n| \cdot h$ where $u(ih,t)$ is the true solution at $x=ih$, t and u_i^n is the computed solution at ih, nk, $k=t/n$.

Other measures of error are also useful. One such is the error in the left- and right-hand limits of the solution value along the shock. These are also used in Colella [3]. We also will experiment with a case not treated analytically, the case of shock formation.

3. Experiments and data. Guided by the ideas outlined above, we measured the error (in the norms described below) at 40 time steps and x running from 0 to 20,

taking $h=1/2^m$, $m=0$ to 4. h/k is constant at 0.4. We deal with five cases: (a) rarefaction alone; (b) compression wave with shock formation; (c) shock alone; (d) shock interacting with a rarefaction; and (e) shock interacting with a compression wave.

The various measures of accuracy used are:

(i) max norm, called M on the graphs.
(ii) L_1- and L_2-norms, called L_1, L_2 on the graphs.
(iii) distance from the computed shock to the true shock for fixed time, called S on the graphs.
(iv) error between the computed value of the right-hand limit at a fixed time and the true value.

The initial data are:

		$0 \leqq x \leqq 4$	$4 < x \leqq 20$
(a)	$\phi(x)=$	0	2
(b)	$\phi(x)=$	2	$1+\dfrac{\cos(\pi/32(x+4))}{\cos(\pi/4)}$
(c)	$\phi(x)=$	2	0
(d)	$\phi(x)=$	2	$-\dfrac{x}{32}+\dfrac{5}{8}$
(e)	$\phi(x)=$	2	$\dfrac{x}{32}-\dfrac{5}{8}$

We have chosen initial data that allow for the easy derivation of the true solutions, as presented below.

(a) is a rarefaction wave, i.e.,

$$u(x,t)=0, \quad 0 \leqq x \leqq 4 \text{ and all } t,$$
$$u(x,t)=2, \quad x \geqq 2t+4,$$
$$u(x,t)=\frac{x}{t}, \quad x \text{ between 4 and } 2t+4.$$

(b) is a compression wave with a shock forming as shown in Fig. 2. It is possible to calculate that the shock is a straight line with slope 1 and to determine its starting point. Since this shock, within the triangle in Fig. 2, is the interaction of many other shocks, this is not evident at first glance. But we can argue as follows.

Consider the data $\hat{\phi}(x)=1$ $(0 \leqq x \leqq 4)$ and $\hat{\phi}(x)=\cos(\pi(x+4)/32)$ $(4<x \leqq 20)$. In the interval $(4,20)$, $\hat{\phi}(x)$ is odd about the point $x=12$, and a symmetry argument shows that the shock formed by interacting characteristics emanating from the initial line is a perpendicular line starting somewhat above the x-axis. Further, the final resulting shock emanating from the apex of the triangle also has slope $dx/dt=0$, and thus the entire shock front consists of a single straight line with slope $dx/dt=0$. Now the initial data in case (b) is thus $\hat{\phi}+1$.

A direct calculation shows that if $u(x,t)$ solves the Burgers equation with data $\hat{\phi}$, then $u(x-at,t)+a$ solves the Burgers equation with data $a+\hat{\phi}$, and so we arrive at the solution described.

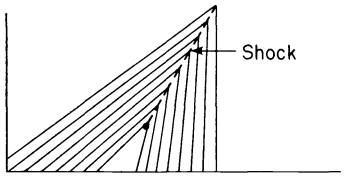


FIG.2.

Using the equation

$$u(x,t)=\phi(x-u(x,t)t) \tag{3.1}$$

which the solution satisfies in smooth regions near the initial line (see [6]), it is easy to use a rapidly converging bisection routine to plot the exact solution with great accuracy.

(c) is a shock,

$$u(x,t)=2, \qquad x\leqq t+4,$$
$$u(x,t)=0, \qquad x\geqq t+4,$$

propagating with speed 1 as required by the jump condition

$$s=\frac{dx}{dt}=\frac{\frac{1}{2}(U_R)^2-\frac{1}{2}(U_L)^2}{U_R-U_L}=\frac{U_R+U_L}{2},$$

where $U_R(U_L)$ is the right- (left-) hand constant value.

(d) As shown in Fig. 3, a compression wave causes the shock to bend into the wave. To one side the solution is identically 2; to the other side of the shock the solution is the compression solution and the shock front itself satisfies the ordinary differential equation

$$\frac{dx}{dt}=s=\frac{U_R+U_L}{2}=\frac{1}{2}\left[\frac{20-x}{32-t}+2\right]$$

whose solution is

$$x(t)=\frac{12}{\sqrt{2}}(32-t)^{\frac{1}{2}}+2t-44.$$

(e) Similarly, a rarefaction causes bending of the shock outward (see Fig. 4). Here we have $U=2$ on the left, $u=(x-20)/(t+32)$ on the right and the shock

$$x(t)=-\frac{20}{\sqrt{2}}(t+32)^{\frac{1}{2}}+2t+84.$$

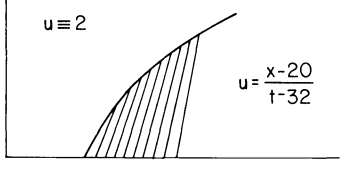


FIG. 3.

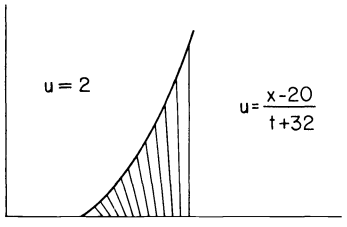


FIG. 4.

One can verify that all these solutions satisfy the entropy conditions guaranteeing that they are the correct weak solutions to the Burgers equation.

Each set of initial data was run on a grid 20 units in x and 16 units in t with $h=1/2^{m-1}$, $m=1$ to 5 and k fixed at $.4h$. Thus the Courant–Friedrichs–Lewy condition is satisfied for all times.

4. Results. As Fig. 5 shows, the errors in the max, L_1-, and L_2-norms appear to be nearly perfect linear functions of h in case (a), verifying the best theoretical estimates and mirroring the first order accuracy expected of Glimm's method. The above is true at various times and for several sets of data which are not included here.

For case (b), shock formation (Fig. 6), all three norms seem to show excellent linear behavior, even though we have no theoretical result after the onset of the shock. In general having a shock makes the max norm useless, but here the shock starts at zero strength and appears not to affect the error for some time. The time when the shock begins can be determined by differentiating the relation (3.1) and solving for u_x; thus,

$$u_x = \frac{\phi'}{1+t\phi'}.$$

For our choice of initial data (b), ϕ' is negative at some points, and thus u_x will surely become unbounded as t approaches $|1/\min\phi'|$. (ϕ' is $-(\pi/(32\cos(\pi/4)))\cdot \sin(\pi(x-4)/32)$ whose minimum is $-\sqrt{2}\ \pi/32$.) In cases (d) and (e), Figs. 7 and 8 show the error in the shock location, S, and in the value R computed for the solution immediately adjacent to the right side of the shock front. The error appears to oscillate about a line drawn in each graph as shown. Such oscillation would be expected in any but the simplest cases because of the random nature of the scheme. Nonetheless, a certain linearity for the asymptotic behavior of the error appears evident.

It is also possible to test the order of asymptotic error growth by plotting the log (error) against $\log h$ and finding the best linear fit to this set of points. The slope of the linear function then gives a measure of the exponent of h in the asymptotic error. This has been done; the result for Fig. 7 is 1.19, which is a bit too high since the best possible result is 1, and for Fig. 8 is 1.2.

Because of the random nature of the scheme and the fact that the slope of the linear functions is only truly correct in the asymptotic limit, these numbers are only approximations to the exponent of h. However, they more nearly support an exponent of 1 than an exponent of $\frac{1}{2}$.

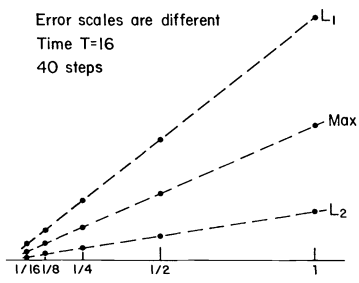


FIG.5.

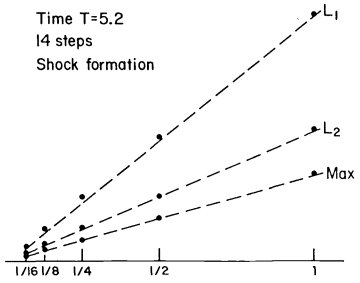


FIG.6.

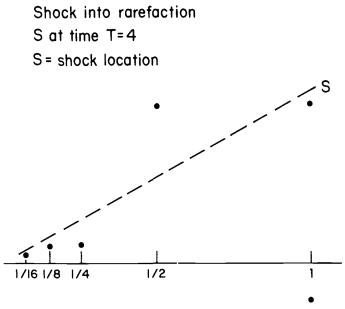


FIG. 7.

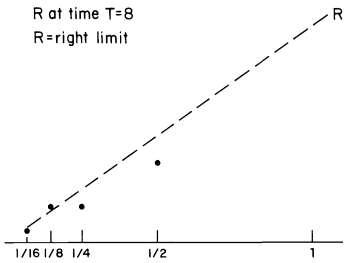


FIG. 8.

5. Further results and conclusion. Observing the terms in $h^{\frac{1}{2}}$ that appear in (2.2.1) and (2.2.2), we would expect to see growth of the error in time if these terms were truly contributing significantly. But examination of all runs (not reproduced here) seems to show little if any growth in time for fixed h. This further supports the contention that we see only h-order error, and that the coefficients of $h^{\frac{1}{2}}$ order terms are negligibly small even for the rather long time spans in our situation.

Thus the conclusion of our results is to show that in practice the theoretical bounds for errors in RCM calculations for our model equation (1.1) are too large by the factor $h^{\frac{1}{2}}$ on the time scales involved here. That is, the error is actually behaving like $O(h)$ independently of time, even in the presence of shocks.

Reference to the articles of Majda and Osher [6] and Lax [5], moreover, indicates that even for the linear case $O(h)$ error in the presence of discontinuities is very good and approaches the best obtainable theoretical possibilities. This lends support for the use of RCM in more difficult and sophisticated situations (for example Chorin's method for gas dynamical equations and others [1], [2], [3]).

Research is presently underway to examine similar situations for the more complicated case of the Buckley–Leverett model equations studied in [4].

Acknowledgment. I would like to thank the referees for many helpful suggestions and Alexandre Chorin for much advice and many illuminating discussions.

REFERENCES

[1] A. J. CHORIN, *Random choice solution of hyperbolic systems*, J. Comp. Phys., 22 (1976), pp. 517–533.

[2] _____, *Random choice method with application to reacting gas flow*, J. Comp. Phys., 25 (1977), pp. 252–272.

[3] P. COLELLA, *An Analysis of the Effect of Operator Splitting and of the Sampling Procedure on the Accuracy of Glimm's Method*, PhD dissertation, University of California, Berkeley, 1979.

[4] P. CONCUS, AND W. PROSKUROWSKI, *Numerical solution of a nonlinear hyperbolic equation by a random choice method*, J. Comp. Phys., 30 (1979), pp. 153–166.

[5] P. D. LAX, *Nonlinear partial differential equations and computing*, SIAM Rev., 11 (1969), pp. 7–19.

[6] A. MAJDA, AND A. OSHER, *Propagation error into regions of smoothness for accurate difference approximations to hyperbolic equations*, Comm. Pure Appl. Math., 30 (1977), pp. 671–705.

[7] G. A. SOD, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comp. Phys., 27 (1978), pp. 1–31.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0004 $01.00/0

CONSTRAINED INTERPOLATION*

JOHN A. ROULIER†

Abstract. The development of theory and algorithms relating to interpolation to data by functions which preserve the monotonicity and/or convexity of the data is presented. The functions used for interpolation are polynomials, piecewise polynomials, polynomial splines and exponential splines. The techniques emphasized are the shape-preserving splines of the author, although some discussion of alternate techniques is given.

Key words. Splines, approximation, interpolation, polynomials, shape preservation

1. Introduction. It is well known by anyone who has used polynomials and splines to interpolate functional data that the interpolant may fail to preserve the shape of the data. That is, if the data set is increasing and/or convex, there is no guarantee that the interpolant will share these properties. It is desirable in many situations to eliminate or at least minimize this difficulty.

Investigations into the possibility of interpolation by polynomials which share the monotonicity and/or convexity of the data have been conducted by Young [23], Wolibner [22], Kammerer [8], Ford and Roulier [6], Passow and Raymon [17], and Rubinstein [21]. In general, these papers show such interpolation can be done, and some [17], provide estimates of the degree of the polynomial required. None provide computational algorithms for accomplishing these goals.

Investigations into the possibility of such interpolation by splines and piecewise polynomials center mainly on the notions of splines under tension [3], [15], [19], [20], Bezier methods [1] and the more recent methods due to the present author and others [2], [4], [5], [7], [9]–[13], [16]. These latter methods are more related to the Bezier techniques than the splines under tension, but are essentially different from both, although the algorithm presented by de Boor [2] can be considered a tension spline technique (See Pruess [20].) Indeed, the manuscript by McAllister and Roulier [13] presents an algorithm which produces a monotonicity and convexity preserving quadratic interpolatory spline with variable knots which is local, and which requires no user judgment as to tension parameters or polygonal arcs as do the above mentioned techniques. In addition, this algorithm allows osculatory interpolation if this is desired.

In the sequel, we will discuss the developments of these latter methods and present examples. The emphasis however, will be on the work of the author.

2. Notation and background. We begin with definitions of terms and notation to be used throughout. Let $\Delta=\{t_0, t_1,\ldots, t_M\}$ be a fixed set of real numbers with $t_j<t_{j+1}$ for $j=0,1,\ldots, M-1$. For given nonnegative integers k, n with $k\leqq n$, we define the set of *polynomial splines of degree n and deficiency n−k on* Δ by

$$S_n^k(\Delta)=\{f\in C^k[t_0, t_M] \mid f \text{ is a polynomial of degree } n \text{ or less on } [t_j, t_{j+1}], j=0,1,\ldots, M-1\}.$$

*Received by the editors May 15, 1980. This research was supported in part by the National Science Foundation under Grant MCS76-04033 and the National Aeronautics and Space Administration under Grant NSG 1549.

†Department of Mathematics, University of Connecticut, Storrs, Connecticut 06268.

The numbers in Δ are called *knots* for the elements of $S_n^k(\Delta)$. $S_n^{n-1}(\Delta)$ is commonly referred to as the *polynomial splines of degree n with knots* Δ, and $S_n^n(\Delta)$ is simply the set of all *polynomials of degree n or less and will be denoted by* Π_n.

Now let a data set $\{(x_i, y_i)\}_{i=0}^N$ be given with $x_i < x_{i+1}$ for $i=0,1,\ldots,N-1$. Define slopes $S_i=(y_i-y_{i-1})/(x_i-x_{i-1})$ for $i=1,2,\ldots,N$. The data set is said to be *increasing* if $S_i>0, i=1,2,\ldots,N$ and *convex* if $S_1<S_2<\ldots<S_N$. If equality is allowed in the above inequalities, then we use the terms nondecreasing and nonconcave, respectively. Similarly, one may define the terms decreasing, concave, nonincreasing, and nonconvex data. In general, data may be segmented into nondecreasing and nonincreasing segments and further subdivided into nonconcave and nonconvex segments. Given $f\in C^1[x_0, x_N]$ for which $f(x_i)=y_i$, $i=0,1,\ldots,N$, we say that f is *monotonicity preserving* if the sign of $f'(x)$ agrees with the sign of S_i when $x\in(x_{i-1}, x_i)$. f is said to be *convexity preserving* if f' is increasing in intervals in which the data are nonconcave and f' is decreasing in intervals in which the data are nonconvex. Thus, f' can only change sign at a data point which is a local extremum if f is monotonicity preserving and f' can only change monotonicity once in any interval between convex and concave regions if f is convexity preserving. If f is both monotonicity and convexity preserving, we say that f is *shape preserving.*

3. Shape-preserving polynomial interpolation. The notion of monotonicity and/or convexity preserving interpolation by polynomials has been studied by many authors. The existence of polynomials which share the monotonicity of the data has been studied by Wolibner [22], Kammerer [8], Rubinstein [21], Young [23], and Passow and Raymon [17]. In the last paper, an estimate on the degree required of the polynomial to preserve monotonicity of the data is obtained.

In [6], Ford and Roulier show the existence of interpolation polynomials which preserve both the monotonicity and convexity of convex increasing data. The main result is given below.

THEOREM 3.1. *Let the data* $\{(x_i, y_i)\}_{i=0}^N$ *be increasing and convex. There are polynomials P and Q satisfying*

$$P(x_i)=y_i, \quad i=0,1,\ldots,N, \tag{3.1}$$

$$P'(x_i)\geqq 0 \quad \text{on } (-\infty,\infty), \tag{3.2}$$

$$P''(x_i)\geqq 0 \quad \text{on } [x_0, x_N], \tag{3.3}$$

$$Q(x_i)=y_i, \quad i=0,1,\ldots,N, \tag{3.4}$$

$$Q'(x_i)\geqq 0 \quad \text{on } [x_0, x_N], \tag{3.5}$$

$$Q''(x_i)\geqq 0 \quad \text{on } (-\infty,\infty). \tag{3.6}$$

It should be noted that no polynomial p of degree 2 or higher can satisfy both $p'(x)\geqq 0$ and $p''(x)\geqq 0$ on $(-\infty,\infty)$.

It is also shown in [6] that if certain information is available about the function which the data represents, then one can prove much more.

THEOREM 3.2. *Let* $k_1<k_2<\ldots<k_p$ *be fixed positive integers and let* $\varepsilon_1,\ldots,\varepsilon_p$ *be fixed signs (i.e.,* $\varepsilon_j=\pm 1$*). Suppose* $f\in C^k[a,b]$ *and* $k_p\leqq k$*. Assume that*

$$\varepsilon_j f^{(k_j)}(x)>0 \quad \text{for } a\leqq x\leqq b \text{ and } j=1,2,\ldots,p,$$

and that $N+1$ *points are given so that*

$$a\leqq x_0<_1<\ldots<x_N\leqq b.$$

Then for n sufficiently large there are polynomials P_n of degree less than or equal to n for which

$$\varepsilon_j P_n^{(k_j)}(x) > 0 \quad \text{on } [a,b], \quad j=1,2,\ldots,p, \tag{3.7}$$

$$P_n(x_i) = f(x_i) \quad \text{for } i=0,1,\ldots,N \quad \text{and} \tag{3.8}$$

$$\max_{a \leqq x \leqq b} |f(x) - P_n(x)| \leqq \left(\frac{C}{n^k}\right)\omega\left(\frac{1}{n}\right), \tag{3.9}$$

where c is a positive constant depending only on $x_0, x_1, \ldots, x_n$ and ω is the modulus of continuity of $f^{(k)}$ on $[a,b]$.

Thus, $\omega(h) = \max\{|f^{(k)}(x) - f^{(k)}(y)|;\ |x-y| \leqq h,\ x, y \in [a,b]\}$.

It should be noted that none of the above mentioned papers present algorithms for the computation of such polynomials, although the proofs of the theorems presented suggest possible algorithms. In any event, for most computational purposes, it would seem more logical to use splines. This is the subject of the remaining sections.

4. Splines under tension. The main purpose of this section is to point out the main features of general splines under tension. For a more detailed treatment, history and bibliography, the reader is referred to Pruess [19], [20] and to Nielson [15] and de Boor [2]. Indeed, Pruess [20] generalizes the notion of tension splines in such a way as to include the notions of exponential splines, piecewise polynomial tension splines and rational splines.

A common feature of all of these notions is that of a tension parameter which may depend on the knot interval. As the tension parameters increase, the graph of the spline tends to pull closer to the shape of the polygonal segments connecting the data. Thus, for sufficiently high tension parameters the spline will be (or very nearly be) shape preserving. Typically, one knows from experience how to choose the tension parameters so that shape preservation is achieved in two or three attempts. Each time a change is made in the tension parameters, the complete spline must be recalculated by solving a system of linear equations.

Cline [3] has produced a subroutine package for interpolation using exponential splines under tension.

In general, these techniques are satisfactory for many purposes, but they have some obvious drawbacks:

(4.1) They are dependent on the user's judgment for a choice and subsequent adjustment of the tension parameters.

(4.2) They are not local. Therefore, each change in tension parameters affects the whole spline.

(4.3) The exponential splines have the added disadvantage that exponential functions must be estimated for each evaluation.

5. Bezier curves. Bezier curves are essentially parametric piecewise polynomial arcs constructed by using Bernstein polynomials of piecewise linear curves. The vertices of the piecewise linear segments are subsequently adjusted to give the curve the desired shape. These curves have been useful in computer-aided geometric design in an interactive setting. Several papers and references involving these are found in Barnhill and Riesenfeld [1]. Here, as for the tension splines, the user must make adjustments to the parameters of the curve in order to attain the desired shape, although these methods are more local.

6. Shape-preserving interpolation with fixed knots. Several papers on this subject have been written recently. The first, by Passow and Roulier [18], provides necessary and sufficient conditions for shape-preserving interpolation to convex, increasing data by splines with fixed knots at the data points. The next paper, by McAllister, Passow and Roulier [9], provides an algorithm for producing the spline discussed in [18]. The algorithm has serious difficulties in that for a given degree of continuity and given convex, increasing data the degree of the spline required can be very high. The problem occurs in the attempt to preserve convexity. If only monotonicity needs to be preserved, then Passow [16] shows that given n there is $f\in S_{2n+1}^n(\Delta)$ which interpolates the data and shares the monotonicity. Here $\Delta=\{x_0, x_1,\ldots, x_N\}$. Furthermore, in [18], Passow and Roulier show that this may be done with $f\in S_{2n}^n(\Delta)$ if Δ is expanded to include the points $\bar{x}_i=x_{i-1}+x_i/2$, $i=1,\ldots, N$ as well as the points $x_0,\ldots, x_N$.

On the other hand, Passow and Roulier [18] and McAllister, Passow and Roulier [9] show that for convexity preservation, no such bound on the degree for a given continuity can be given for fixed knots. We present here the latter and most general result from [9].

THEOREM 6.1. *Let the integer $n\geqq 1$, real numbers $x_0<x_1<x_2<x_3$ and knots Δ be given. There exist numbers y_0, y_1, y_2, y_3 such that the data (x_i, y_i), $i=0,1,2,3$, are increasing and convex and such that no $f\in S_n^1(\Delta)$ satisfies*

$$f(x_i)=y_i, \qquad i=0,1,2,3, \tag{6.1}$$

and

$$f \text{ is convex and increasing on } [x_0, x_3]. \tag{6.2}$$

This theorem is a simple corollary of the following lemma:

LEMMA 6.1. *Let x_0, x_1, x_2, x_3 and Δ be as in Theorem* 6.1, *and let (x_i, y_i), $i=0,1,2,3$, be convex and increasing. Let $f\in S_n^1(\Delta)$ satisfy* (6.1) *and* (6.2) *and let $l=\max\{x\in\Delta \mid x<x_1\}$ and $\mu=\min\{x\in\Delta \mid x>x_1\}$.*

$$n^2\geqq \frac{S_2}{\dfrac{(x_2-x_1)(S_3-S_2)}{\mu-x_1}+\dfrac{(x_1-x_0)S_1}{x_1-l}}. \tag{6.3}$$

We note that if $\Delta=\{x_0, x_1, x_2, x_3\}$, then (6.3) becomes

$$n^2\geqq \frac{S_2}{S_3-S_2+S_1}. \tag{6.4}$$

It is shown by Myers and Roulier [14] that the estimates (6.3) and (6.4) are essentially best possible. That is, one can not replace n^2 by some lower power of n.

On the other hand, the algorithm in [9] allows one to select for given convex, increasing data $\{(x_i, y_i)\}_{i=0}^N$ and positive integer m, a degree n and spline $f\in S_n^m(\Delta)$ with $\Delta=\{x_0, x_1,\ldots, x_N\}$, so that

$$f(x_i)=y_i, \qquad i=0,1,\ldots, N$$

and

$$f \text{ is convex, increasing on } [x_0, x_N].$$

This algorithm works but has some major drawbacks. First, the continuity up to the mth derivative is artificial for $m \geqq 2$ in that $f^{(j)}(x_i)=0$ for $j=2,\ldots,m$ and $i=0,1,\ldots,N$. Thus, the choice of $m=1$ is the only realistic choice for graphical purposes. The second major drawback is that the degree n grows faster for this algorithm than the estimate (6.4) indicates. Indeed, it grows as if the left side of (6.4) were replaced by n. This is, due to the special nature of the splines used in the algorithm, and in fact an inequality like (6.4) can be shown to hold with n^2 replaced by n for the special splines of this algorithm. Examples of just how high n can get for some data appear in [10]. For this reason, it was decided that an investigation of the possibility of using variable knots was worthwhile. For this purpose, we need some of the specific results regarding fixed knots.

In particular, we will give necessary and sufficient conditions on nondecreasing and nonconcave data $\{(x_i, y_i)\}_{i=0}^N$ to insure the existence of a quadratic spline $f \in S_2^1(\Delta)$ which interpolates the data and is convex and increasing on $[x_0, x_N]$, and with $\Delta=\{x_0, x_1,\ldots, x_N\}$. We first present some additional notation as in [10].

DEFINITION 6.1. Suppose that the data $\{(x_i, y_i)\}_{i=0}^N$ are nondecreasing and nonconcave. Let $\bar{x}_i = x_{i-1} + \Delta x_i/2$ where $\Delta x_i = x_i - x_{i-1}$. The set of numbers $\{t_i\}_{i=1}^N$ is said to be $\frac{1}{2}$-*admissible* if the piecewise linear function $L(x)$ generated by the points

$$(x_0, y_0),(\bar{x}_1, t_1),(\bar{x}_2, t_2),\ldots,(\bar{x}_N, t_N),(x_N, y_N)$$

satisfies $L(x_i)=y_i$ for $i=1,\ldots,N-1$ and is nondecreasing and nonconcave.

We now describe a special case of an algorithm found in [9] (see [10]).

The $\frac{1}{2}$*-algorithm.* Define $m_0=0$ and $M_0=S_1$. Now for $i=1,2,\ldots,N-1$ define

$$m_i = 2S_i - M_{i-1} \quad \text{and} \quad M_i = \min(S_{i+1}, 2S_i - m_{i-1}).$$

We now present necessary and sufficient conditions for convex quadratic spline interpolation with knots $\Delta=\{x_0, x_1,\ldots, x_N\}$. This theorem is in part a special case of results in [9] and [10] and appears as two separate theorems in [9].

THEOREM 6.2. *Let* $\{(x_i, y_i)\}_{i=0}^N$ *be nondecreasing nonconcave data, and let* $\Delta=\{x_0, x_1,\ldots, x_N\}$. *The following are equivalent*:

(6.5) *There exists a quadratic spline* $f \in S_2^1(\Delta)$ *satisfying* $f(x_i)=y_i$ *for* $i=0,1,\ldots,N$ *and* f *is convex, increasing on* $[x_0, x_N]$.

(6.6) *There exists a* $\frac{1}{2}$*-admissible set for this set of data.*

(6.7) *The* $\frac{1}{2}$*-algorithm produces* $m_i, i=1,\ldots,N-1$ *satisfying* $m_i \leqq S_{i+1}$ *for* $i=1,2,\ldots,N-1$.

In the next section, we will present theory and algorithms for variable knot quadratic spline interpolation with shape preservation. Theorem 6.2 is the fundamental tool for these results.

Once (6.7) is satisfied one can then produce the $\frac{1}{2}$-admissible set of (6.6) as follows:

Let

$$(6.8) \quad a_N = y_{N-1} + m_{N-1}\frac{(x_N - x_{N-1})}{2} \quad \text{and} \quad b_N = y_{N-1} + M_{N-1}\frac{(x_N - x_{N-1})}{2}.$$

Let t_N be any number between a_N and b_N. The choice of t_N uniquely defines $t_{N-1}, t_{N-2},\ldots, t_1$.

Once the $\frac{1}{2}$-admissible set $\{t_i\}_{i=1}^N$ is obtained, the piecewise linear function L of Definition 6.1 is determined. Now, using L, we are able to construct the spline $f \in S_2^1(\Delta)$ of (6.5) by

$$(6.9)\quad f(x)=\frac{1}{(\Delta x_i)^2}\left[y_{i-1}(x_i-x)^2+2L(\bar{x}_i)(x-x_{i-1})(x_i-x)+y_i(x-x_{i-1})^2\right],$$

$$\text{for } x\in[x_{i-1}, x_i] \text{ and } i=1,2,\ldots,N.$$

The above expression is the second degree Bernstein polynomial for L on each interval $[x_{i-1}, x_i]$. (See [10] as well as [12] and [18]). The quantities $\bar{x}_i$ and Δx_i are as in Definition 6.1. The fact that f is in $S_2^1(\Delta)$ and shares the monotonicity and convexity of the data follows from the well-known properties of Bernstein polynomials and the form of L on $[x_{i-1}, x_i]$. That is, on $[x_{i-1}, x_i]$, L is a piecewise linear function consisting of two linear segments passing through (x_{i-1}, y_{i-1}) and (x_i, y_i), respectively and intersecting at $\bar{x}_i = x_{i-1} + \frac{1}{2}\Delta x_i$, the midpoint of $[x_{i-1}, x_i]$. In [18], it is shown that the second degree Bernstein polynomial B_2 of such a function shares the monotonicity and convexity of the function as well as

$$(6.10)\qquad \begin{aligned} &B_2(x_{i-1})=L(x_{i-1}), \quad B_2(x_i)=L(x_i),\\ &B_2'(x_{i-1})=L'(x_{i-1}), \quad B_2'(x_i)=L'(x_i).\end{aligned}$$

Indeed, it is a simple consequence of a theorem in [18] that any $f \in S_2^1(\Delta)$ can be constructed in this way. It is observed there that the abscissa of the intersection point of the tangent lines to the graph of any quadratic q at $(a, q(a))$ and $(b, q(b))$ is $(a+b)/2$, the midpoint of the interval $[a, b]$.

7. Shape-preserving interpolation by quadratic splines with variable knots. We begin by considering nondecreasing, nonconcave data $\{(x_i, y_i)\}_{i=0}^N$ as in the previous section. Theorem 6.2 is used in [10] to provide the technique for choosing a set of knots $\bar{\Delta}$ which contains $\{x_0, x_1, \ldots, x_N\}$ and at most one additional knot between each pair x_{i-1} and x_i to insure the existence of a quadratic spline $f \in S_2^1(\bar{\Delta})$ satisfying (6.5). In fact, we select additional data points which when added to the original set create a set of data satisfying (6.7). This is done in an optimal way, in that knots are only added where necessary. We now present the point insertion algorithm which shows how to select the new points and which is discussed in the next theorem.

Point insertion algorithm. Apply the $\frac{1}{2}$-algorithm until

$$(7.1)\qquad m_k \geqq S_{k+1} \quad \text{for some } k,$$

(note that we are guaranteed that $k \geqq 2$). Consider the four points

$$(x_{k-2}, y_{k-2}),\quad (x_{k-1}, y_{k-1}),\quad (x_k, y_k),\quad (x_{k+1}, y_{k+1}).$$

Now define $S_0 = (m_{k-2} + M_{k-2})/2$ and define the point $(\bar{x}, \bar{y})$ where

$$\bar{x}=x_{k-1}-2(x_{k-1}-x_{k-2})\frac{(S_{k-1}-S_0)}{(S_k-S_0)}$$

(7.2) and

$$\bar{y}=y_{k-2}+S_0(\bar{x}-x_{k-2}).$$

It can be shown that $x_{k-2} < \bar{x} < x_{k-1}$ and $y_{k-2} < \bar{y} < y_{k-1}$. Now insert the point $(\bar{x}, \bar{y})$ between (x_{k-2}, y_{k-2}) and (x_{k-1}, y_{k-1}) and renumber the data. Now proceed as above until (7.1) is satisfied again or until (6.7) is satisfied using this "expanded data set."

The following theorem is found in [10].

THEOREM 7.1. *Given a convex, increasing set of data* $\{(x_i, y_i)\}_{i=0}^{N}$:

(7.3) *The point insertion algorithm will terminate in a finite number* $M \leqq N$ *steps.*

(7.4) *At most one new point will be inserted between any two original data points.*

(7.5) *The final "expanded" set of data* $\{(x_i', y_i')\}_{i=0}^{N+M}$ *thus created will satisfy* (6.5), (6.6) *and* (6.7).

To illustrate the usefulness of this result, the reader is referred to the examples in [10] for which the degree of the splines required using the abscissas as fixed knots and using the algorithm in [9] is $n = 21$, 9650, 1000, respectively.

This algorithm in [10] can be modified to be used for arbitrary data $\{(x_i, y_i)\}_{i=0}^{N}$ by subdividing the data into increasing and decreasing segments and then further subdividing into convex and concave segments. This, however, seems to be more trouble than is desirable in view of the next result. This provides a technique for shape-preserving interpolation to arbitrary data as above by quadratic splines with variable knots. The algorithm provides for exactly one new knot between each pair of the original data points, is local, and does not require a prior subdivision of the data as the previous algorithm would. Furthermore, if osculatory interpolation is desired, this can be done by adding at most two (but usually only one) variable knots between each pair of original data points. This algorithm appears in part in [4] and [5], but the most general form appears in [12] and [13].

The general approach of this algorithm is as follows:

(7.6) If derivative values are provided at each of the data points, then procedures CHOOSE and CASES for choosing the location of the knots between the data points are called. These procedures choose one or two knots with corresponding y-coordinates between each pair of data points. The only information needed in each interval is the data and derivatives at the twoend points. Thus, this technique is local. The resulting spline will be shape preserving unless the given derivatives do not agree with the monotonicity or convexity exhibited by the data.

(7.7) If only data are given with no derivative values, then a procedure called SLOPES for estimating derivative values at each data point is provided. The derivative values provided by SLOPES are in conformity with the local monotonicity and convexity of the data. The above procedures are then called using the derivatives provided. It should be noted that this procedure for providing derivatives is also local in that the value of the derivative at (x_i, y_i) is calculated from the three points (x_{i-1}, y_{i-1}), (x_i, y_i), and (x_{i+1}, y_{i+1}) for $i = 1, \ldots, N-1$. If $i = 0$ or $i = N$, then the value of the derivative provided is calculated using the information at the given data point and the two points to the right if $i = 0$, or to the left if $i = N$. Furthermore, by construction of these derivatives CHOOSE and CASES will always provide exactly one new knot between the data.

Once the additional data points are provided and derivatives given, a piecewise linear function L as in Definition 6.1 is constructed and, because of the placement of the knots and the derivatives, a quadratic spline f is calculated by evaluating the

second degree Bernstein polynomial of L between each pair of knots as in (6.9). The actual evaluation of $f(x)$ at a given value x is accomplished via a master procedure MEVAL and several subprocedures. We avoid the details of the algorithm which chooses the location of the knots or the calculation of derivatives since these are presented in [12].

We present several illustrative examples of the algorithm some of which appear in [9] and [12]. Figs. 1, 2, 3, 4 and 5 are for functional data and Figs. 6, 7, 8 and 9 are for planar curves and require parametrization. To apply the above algorithm to parametric planar data, the data is parametrized by arc length and the separate parametric function data $\{(t_i, x_i)\}_{i=0}^{N}$ and $\{(t_i, y_i)\}_{i=0}^{N}$ are interpolated. The procedure was written to detect closed curves (i.e., $(x_0, y_0)=(x_N, y_N)$) or symmetric data.

It should be noted here, that although the shape of functional data is preserved here, the concavity of parametric data may not be preserved unless there are enough data in the regions of highest curvature. See Fig. 8. The reason for this is obvious if one examines the second derivative in terms of the quadratic splines interpolating $x_i=x(t_i)$ and $y_i=y(t_i)$.

8. Examples. Figs. 1, 2 and 3 are to illustrate the shape preserving, local and osculatory properties of the algorithm described in the previous section. The data given in Table 1 and the resulting shape-preserving quadratic spline curve fit in Fig. 1 clearly shows the outline of a car. In Fig. 2 we see the curve fit to the data in Table 1 but with (1, 1.8) replaced by (1, 2.3). Note the local change. In Fig. 3 the data is the same as in Table 1 but with a slope of 1 imposed at (10,5) on the "roof" of the car.

Figs. 4 and 5 illustrate examples where other methods had difficulty. In Table 2 the data are taken from an example in Pruess [20]. It took two changes in tension parameter for the spline in tension used there to match the monotonicity and three changes to match the convexity. Fig. 4 shows that the algorithm in the previous section matches both monotonicity and convexity immediately. Fig. 5 appeared in [9] and shows the insufficiency of standard Lagrange polynomials and cubic splines for

TABLE 1.

x	y	Slope
0	0	2.4
1	1.8	1.2
2	2.7	.39
3	2.9	.0833...
4	3.0	0.0
5	3.0	0.0
6	3.65	.80294118
7	4.7	.5250...
8	5.05	0.0
9	5.05	0.0
10	5.0	−.050...
11	4.95	−.0750...
12	4.8	−.2400...
13	4.2	−.750...
14	3.2	−.85714286
15	2.45	−.98863636
16	1.0	−2.0209059
16.3	0.0	−4.6457607

TABLE 2.

x	y
22	523
22.5	543
22.6	550
22.7	557
22.8	565
22.9	575
23.0	590
23.1	620
23.2	860
23.3	915
23.4	944
23.5	958
24	986

shape preservation. The shape-preserving spline in this is for fixed knots at the data using the techniques in [9]. The data for this appears in Table 3, and Fig. 5 shows the graphs.

Figs. 6, 7, 8 and 9 are for parametric planar data. The technique used is described at the end of the previous section. In Fig. 6, the spline interpolates the four endpoints of the major and minor axes of an ellipse (Table 4). In Fig. 7, the points (± 4, ± 3.46) are added, and in Fig. 8, the points (± 7, ± 1.94) are added to those of Fig. 7. Note the lack of concavity in Fig. 7. If the points (± 7, ± 1.94) had been added to Fig. 6 first, then the concavity would not have been lost. See Fig. 9.

Note that except in Figs. 4 and 5 the x and y values are scaled to $\frac{1}{2}$ the tabular values.

TABLE 3.

x	y
-2	$\frac{1}{4}$
-1	1
$-.3$	11.11...
$-.2$	25

TABLE 4.

x	y
8	0
0	-4
-8	0
0	4
8	0

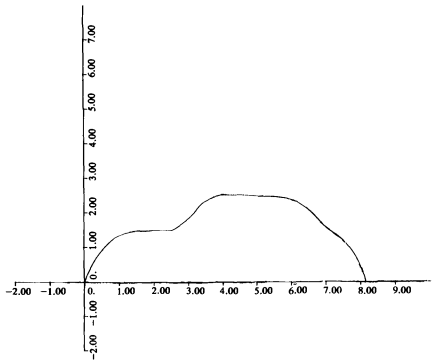


FIG 1

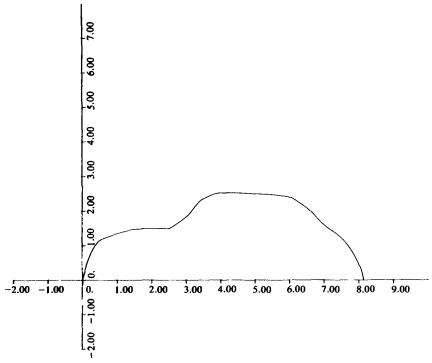


FIG 2

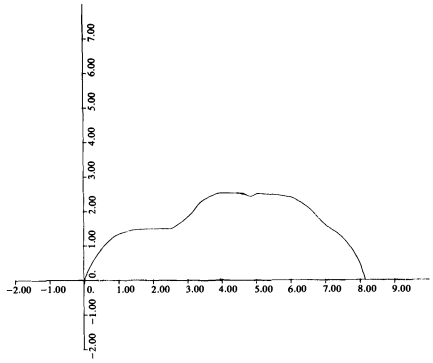


FIG 3

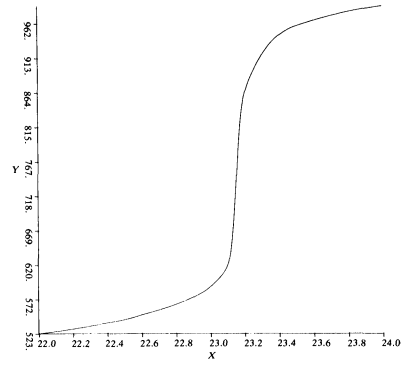


FIG 4

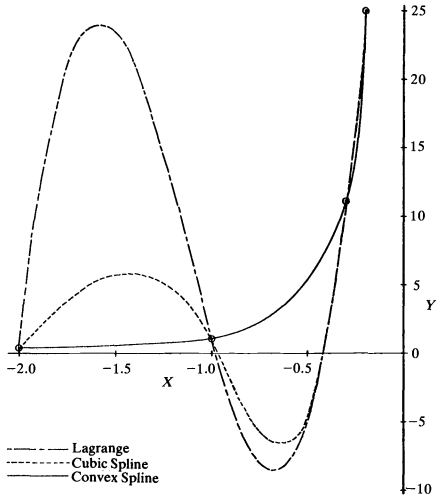


FIG 5

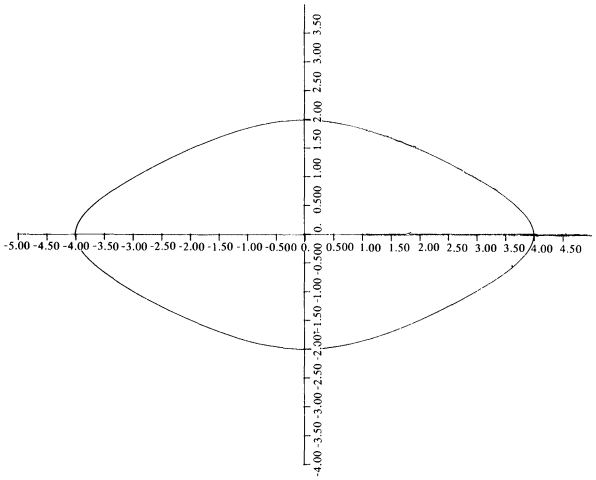


FIG 6

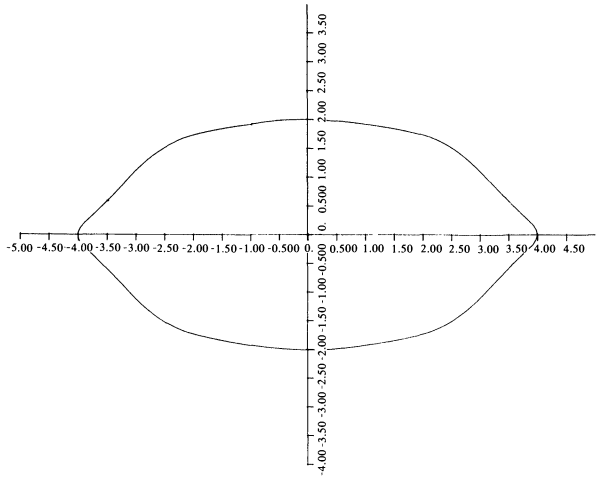


FIG 7

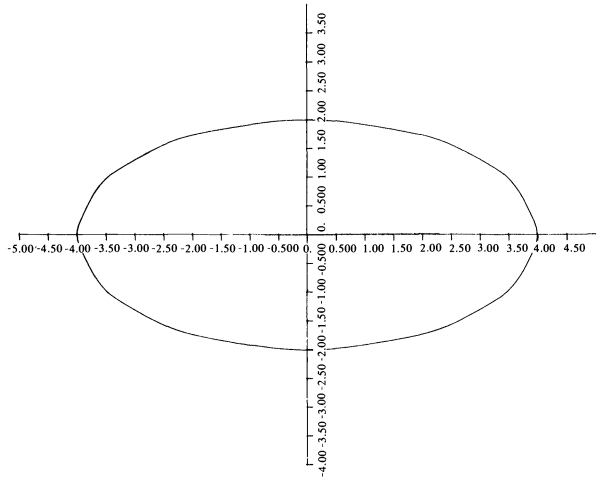


FIG 8

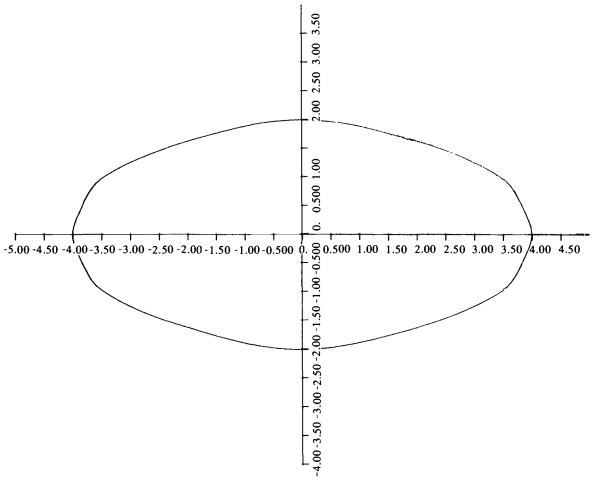


FIG 9

9. Conclusions and remarks. The above-mentioned algorithms for shape-preserving quadratic spline interpolation with variable knots has the double advantage over splines in tension in that no user-dependent choice of tension parameters is necessary for shape preservation, and yet the method is local so that the user may modify the shape of the resulting curve as desired in one area without changing the shape elsewhere. This is also an advantage over Bezier methods in that no user-dependent adjustments are necessary for shape preservation of functional data.

For parametric data, the application of the algorithm always preserves monotonicity, but concavity may not be preserved if points are very sparse. See Fig. 7 once again. This difficulty can be dealt with and will be the subject of a subsequent paper.

The approximation, differentiation, and quadrature properties of these splines are currently under study. The results will appear shortly in a subsequent publication.

REFERENCES

[1] R. E. BARNHILL AND R. F. RIESENFELD (eds.), *Computer Aided Geometric Design*, Academic Press, New York and London, 1974.

[2] C. DE BOOR, *A Practical Guide to Splines*, Springer, New York, 1979.

[3] A. K. CLINE, *Curve fitting in one and two dimensions using splines under tension*, Comm. ACM, 17(1974), pp. 213–218.

[4] L. E. DEIMEL, C. L. DOSS, R. J. FORNARO, D. F. MCALLISTER AND J. A. ROULIER, *Applications of shape-preserving spline interpolations to interactive editing of photogrammetric data*, Proc. SIGGRAPH 1978, 12(1978), pp. 92–99.

[5] L. E. DEIMEL, D. F. MCALLISTER AND J. A. ROULIER, *Smooth curve-fitting with shape preservation using osculatory quadratic splines*, Proc. of 11th Annual Conference on the Interface Between Statistics and Computer Science, 1978, pp. 343–347.

[6] W. T. FORD AND J. A. ROULIER, *On interpolation and approximation by polynomials with monotone derivatives*, J. Approx. Theory, 10(1974), pp. 123–130.

[7] F. N. FRITSCH AND R. E. CARLSON, *Piecewise cubic interpolation methods*, presented at SIAM 1978 Fall meeting in Knoxville, Tennessee.

[8] W. J. KAMMERER, *Polynomial approximations to finitely oscillating functions*, Math. Comp., 15(1961), pp. 115–119.

[9] D. F. MCALLISTER, E. PASSOW AND J. A. ROULIER, *Algorithms for computing shape-preserving spline interpolations to data*, Math. Comp., 31(1977), pp. 717–725.

[10] D. F. MCALLISTER AND J. A. ROULIER, *Interpolation by convex quadratic splines*, Math. Comp., 32(1978), pp. 1154–1162.

[11] D. F. MCALLISTER, J. A. ROULIER AND M. EVANS, *Generation of random numbers using shape-preserving quadratic splines*, Proc. of 16th Annual Southeast Regional ACM Conference, April, 1978, pp. 216–218.

[12] D. F. MCALLISTER AND J. A. ROULIER, *An algorithm for computing a shape-preserving osculatory quadratic spline*, submitted for publication to TOMS.

[13] _____, *Algorithm 5xx, shape-preserving osculatory quadratic splines*, submitted to TOMS.

[14] D. C. MYERS AND J. A. ROULIER, *Markov-type inequalities and the degree of convex spline interpolation*, J. Approx. Theory, 28(1980), pp. 267–272.

[15] G. NIELSON, *Some piecewise polynomial alternatives to splines under tension*, in Computer Aided Geometric Design, R. E. Barnhill and R. F. Riesenfeld (eds.), (see [1]).

[16] E. PASSOW, *Piecewise monotone spline interpolation*, J. Approx. Theory, 12(1974), pp. 240–241.

[17] E. PASSOW AND L. RAYMON, *The degree of piecewise monotone interpolation*, Proc. Amer. Math. Soc., 48(1975), pp. 409–412.

[18] E. PASSOW AND J. A. ROULIER, *Monotone and convex spline interpolation*, SIAM J. Numer. Anal., 14(1977), pp. 904–909.

[19] S. PRUESS, *Properties of splines in tension*, J. Approx. Theory, 17(1976), 86–96.

[20] _____, *Alternatives to the exponential spline in tension*, SIAM J. Numer. Anal., to appear.

[21] Z. RUBINSTEIN, *On polynomial δ-type functions and approximation by monotonic polynomials*, J. Approx. Theory, 3(1970), pp. 1–6.

[22] W. WOLIBNER, *Sur un polynome d'interpolation*, Colloq. Math., 2(1951), pp. 136–137.

[23] S. W. YOUNG, *Piecewise monotone polynomial interpolation*, Bull. Amer. Math. Soc., 73(1967), pp. 642–643.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0005 $01.00/0

A PROJECTED LAGRANGIAN ALGORITHM FOR NONLINEAR MINIMAX OPTIMIZATION*

WALTER MURRAY† AND MICHAEL L. OVERTON‡

Abstract. The minimax problem is an unconstrained optimization problem whose objective function is not differentiable everywhere, and hence cannot be solved efficiently by standard techniques for unconstrained optimization. It is well known that the problem can be transformed into a nonlinearly constrained optimization problem with one extra variable, where the objective and constraint functions are continuously differentiable. This equivalent problem has special properties which are ignored if solved by a general-purpose constrained optimization method. The algorithm we present exploits the special structure of the equivalent problem. A direction of search is obtained at each iteration of the algorithm by solving an equality-constrained quadratic programming problem, related to one a projected Lagrangian method might use to solve the equivalent constrained optimization problem. Special Lagrange multiplier estimates are used to form an approximation to the Hessian of the Lagrangian function, which appears in the quadratic program. Analytical Hessians, finite differencing or quasi-Newton updating may be used in the approximation of this matrix. The resulting direction of search is guaranteed to be a descent direction for the minimax objective function. Under mild conditions the algorithms are locally quadratically convergent if analytical Hessians are used.

Key words. Minimax approximation, Chebyshev approximation, nonlinear programming, projected Lagrangian method

1. Introduction.

The problem of concern is

$$\text{MMP:} \quad \min_{\bar{x}} \{F_M(\bar{x}) \mid \bar{x} \in R^n\}$$
$$\text{where } F_M(\bar{x}) = \max\{f_i(\bar{x}),\ i = 1,2,\ldots,m\},$$

and the functions $f_i: R^n \to R^1$ are twice continuously differentiable. The function $F_M(\bar{x})$ is called the *minimax function* and MMP is usually referred to as the minimax problem. The minimax problem is an unconstrained optimization problem in which the objective function has discontinuous derivatives. Moreover, any solution is usually at a point of discontinuity and consequently it is inappropriate to use any of the known powerful methods for unconstrained minimization to solve MMP. An equivalent problem to MMP is the following nonlinearly constrained problem in which both the objective and constraint functions are twice continuously differentiable:

$$\text{EMP:} \quad \min_{x} \{x_{n+1} \mid x \in R^{n+1}\}$$
$$\text{subject to } c_i(x) \geqq 0, \qquad i = 1,2,\ldots,m,$$
$$\text{where } c_i(x) = x_{n+1} - f_i(\bar{x}), \qquad i = 1,2,\ldots,m,$$
$$\text{and } x^T = (\bar{x}^T, x_{n+1}).$$

We could solve EMP using one of the many methods available for the general constrained optimization problem:

$$\text{NCP:} \quad \min_{x} \{F^{(G)}(x)\}$$
$$\text{subject to } c_i^{(G)}(x) \geqq 0, \qquad i = 1,2,\ldots,m,$$

where $F^{(G)}$ and $\{c_i^{(G)}\}$ are arbitrary twice continuously differentiable functions. It will

*Received by the editors January 3, 1980.

†Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA 94305. The work of this author was supported in part by the U.S. Department of Energy under Contract DE-AT03-76ER72018, the National Science Foundation under Grant MCS76-20019 A01, and the U.S. Army Research Office under Contract DAAG-29-79-C-0110.

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, N Y 10012. The work of this author was supported in part by the U.S. Department of Energy under Contract DE-AS03-76-SF00326 PA#30 and the National Science Foundation under Grants MCS75-13497 and MCS78-11985 at the Computer Science Department, Stanford University.

be shown, however, that a method can be derived that exploits the special structure of EMP.

The primary special feature of EMP from which many other properties follow is that the minimax function F_M is itself a *natural merit function* which can be used to measure progress towards the solution of EMP. For problem NCP in general such a natural merit function is not available, and it is necessary to introduce an artificial one such as a penalty or augmented Lagrangian function to weigh the constraint violation against the decreasing of the objective function, or a barrier function to enforce feasibility. All these merit functions require the definition of a parameter which is to some degree arbitrary, and its selection can prove difficult. In the case of penalty and augmented Lagrangian functions, difficulties may also arise because often the global minimum of the merit function is *not* the solution of the original problem.

The method we adopt to solve MMP essentially consists of two steps at each iteration:

(1) Obtain a direction of search by solving and perhaps modifying an equality-constrained quadratic programming problem (QP), related to one a projected Lagrangian algorithm might use to solve EMP. This procedure is described in full in subsequent sections.

(2) Take a step along the search direction which reduces the minimax function. Because the minimax function is not differentiable, it is important for efficiency to use a special line search algorithm.

Projected Lagrangian algorithms for solving the general problem NCP via successive quadratic programs have been proposed or analyzed by a number of authors including Wilson (1963), Murray (1969a), Robinson (1974), Wright (1976), Han (1977a), Powell (1977), and Murray and Wright (1978). We make further comments on the extent of the implications of the special structure of EMP, and hence the relationship of our algorithm to these algorithms for the general problem, in § 15.

A number of other algorithms have been proposed for solving the nonlinear minimax problem. Our approach is most closely related to those due to Han (1977b) and Conn (1979). We will discuss these further in § 12, after our algorithm has been described in full.

An important special case of MMP is the problem of minimizing the l_∞-norm of a vector function $f(\bar{x}) \in R^m$:

l_∞P: $\min\{F_\infty(\bar{x}) | \bar{x} \in R^n\}$
where $F_\infty(\bar{x}) = \max\{|f_i(\bar{x})|,\ i = 1,2,\ldots,m\}$.

Handling this case in a special manner presents no essential difficulties. However, in order to avoid unnecessarily complicated notation, we postpone discussion of this until § 11.

We note that no convexity assumptions are made about the functions $f_i(\bar{x})$. The difficulties of finding global minima without convexity assumptions are well known—we concern ourselves only with local minima.

1.1 Notation. Define $\overset{*}{x}$ to be a solution of EMP. It follows that $\overset{*}{\bar{x}}$, the vector composed of the first n elements of $\overset{*}{x}$, is a solution to MMP and $\overset{*}{x}_{n+1} = F_M(\overset{*}{\bar{x}})$.

Let $x^{(k)}$ denote the kth approximation to $\overset{*}{x}$ and $\bar{x}^{(k)}$ the kth approximation to $\overset{*}{\bar{x}}$. In general, we will use a $\bar{}$ placed above a vector to denote the vector composed of the first n elements of the vector without the $\bar{}$.

At each iteration of the algorithm, $x^{(k+1)}$ is obtained by setting

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \alpha \bar{p} \quad \text{and} \quad x_{n+1}{}^{(k+1)} = F_M(\bar{x}^{(k+1)}),$$

where $\bar{p}$ is the direction of search and α, a positive scalar, is the steplength. Note that this choice of $x_{n+1}{}^{(k+1)}$ immediately guarantees that all the points $\{x^{(k)}\}$ are *feasible* for problem EMP; i.e., $c_i(x^{(k)}) \geqq 0$, $i = 1, \ldots, m$.

At any point x we define an *active set* of constraints of EMP as those which we think will have the value zero at the solution $\overset{*}{x}$, based on the information at x. This set will usually include all constraints with the value zero at the point x and may also include some with positive values. The exact procedure for initially selecting the active set at each iteration will be discussed in § 10, and procedures for modifying this choice will be described in §§ 5.2 and 6. We define $t(=t(x))$ to be the number of active constraints at x, and write the vector of active constraints as $\hat{c}(x) \in R^t$. We similarly define $\hat{f}(\bar{x})$ as the vector of active functions corresponding to the active constraints, i.e., those functions expected to have the value $F_M(\overset{*}{\bar{x}})$ at $\overset{*}{\bar{x}}$. Let $\hat{V}(\bar{x})$ be the $n \times t$ matrix whose columns $\{\hat{v}_j(\bar{x})\}$ are the gradients of the active functions, and let $\hat{A}(x)$ be the $(n+1) \times t$ matrix whose columns $\{\hat{a}_j(x)\}$ are the gradients of the active constraints. Thus

$$\begin{aligned} \hat{A}(x) &= \begin{bmatrix} -\hat{V}(\bar{x}) \\ \hat{e}^T \end{bmatrix} \quad \text{where } \hat{e} = (1, \ldots, 1)^T \in R^t \\ &= \begin{bmatrix} \hat{a}_1(x) & \ldots & \hat{a}_t(x) \end{bmatrix} \\ &= \begin{bmatrix} -\hat{v}_1(\bar{x}) & \ldots & -\hat{v}_t(\bar{x}) \\ 1 & \ldots & 1 \end{bmatrix}. \end{aligned}$$

We define $Y(x)$ to be a matrix with orthonormal columns spanning the range space of $\hat{A}(x)$, and $Z(x)$ to be a matrix with orthonormal columns spanning the null space of $\hat{A}(x)^T$. Let I_s be the identity matrix of order s. Provided $\hat{A}(x)$ has full rank, we have that $Y(x)$ has dimension $(n+1) \times t$, $Z(x)$ has dimension $(n+1) \times (n+1-t)$, and

$$Y(x)^T Y(x) = I_t, \qquad Z(x)^T Z(x) = I_{n+1-t},$$

$$Y(x)^T Z(x) = \hat{A}(x)^T Z(x) = 0.$$

Let $e_{n+1} = (0, \ldots, 0, 1)^T \in R^{n+1}$. The Lagrangian function for problem EMP is given by

$$L(x, \lambda) = x_{n+1} - \lambda^T \hat{c}(x),$$

where $\lambda \in R^t$ is a vector of *Lagrange multipliers*. The gradient of $L(x, \lambda)$ with respect to x is $e_{n+1} - \hat{A}\lambda$. We define the $(n+1) \times (n+1)$ matrix $W(x, \lambda)$ to be the Hessian of the Lagrangian function with respect to x. Thus

$$W(x, \lambda) = \sum_{i=1}^{t(x)} -\lambda_i \nabla^2 \hat{c}_i(x) = \begin{bmatrix} \overline{W}(\bar{x}, \lambda) & 0 \\ 0 & 0 \end{bmatrix},$$

where

$$\overline{W}(\bar{x}, \lambda) = \sum_{i=1}^{t(x)} \lambda_i \nabla^2 \hat{f}_i(\bar{x}).$$

The term "projected Hessian of the Lagrangian function" is used to indicate projection into the null space of $\hat{A}(x)^T$, i.e., the matrix $Z(x)^T W(x, \lambda) Z(x)$. This matrix may also be written $\bar{Z}(x)^T \overline{W}(\bar{x}, \lambda) \bar{Z}(x)$, where $\bar{Z}(x)$ consists of the first n rows of $Z(x)$.

Often we will omit the arguments from $\hat{c}, \hat{A}, Z$, etc. when it is clear that they are evaluated at $x^{(k)}$. We use the notation $\overset{*}{V}, \overset{*}{A}, \overset{*}{Z}$, etc. to denote $\hat{V}, \hat{A}, Z$, etc. evaluated at $\overset{*}{x}$ with the active set correctly chosen, i.e., consisting of all those constraints with the value zero at $\overset{*}{x}$.

1.2 Necessary and sufficient conditions. In the following we shall refer to the first- and second-order constraint qualifications and the necessary and sufficient conditions for a point $\overset{*}{x}$ to be a local minimum of the general problem NCP as defined in Fiacco and McCormick (1968). The conditions for $\overset{*}{x}$ to be a local minimum of EMP (and hence $\overset{*}{\bar{x}}$ of MMP) are simplifications of these general conditions. The main simplification is that it can be shown that the first-order constraint qualification always holds for EMP. The first-order conditions therefore reduce to the following (see Demyanov and Malozemov (1974) for an alternative derivation applied directly to MMP).

First-order necessary condition. If $\overset{*}{x}$ is a local minimum of EMP, then there exists a vector of Lagrange multipliers $\overset{*}{\lambda} \in R^t$ such that

$$e_{n+1} - \overset{*}{A}\overset{*}{\lambda} = 0 \quad \text{and} \quad \overset{*}{\lambda} \geqq 0. \tag{1.1}$$

Two conditions which are equivalent to (1.1) are that $\overset{*}{x}$ is a stationary point of $L(x, \lambda)$ with respect to x and that $\overset{*}{Z}{}^T e_{n+1} = 0$. Note that (1.1) implies that $\overset{*}{V}$ is rank deficient and that the sum of the multipliers is one.

The second-order constraint qualification does not necessarily hold for EMP (for example at the origin for $f_1 = x_1^3$, $f_2 = -x_1^3$, and $f_3 = -x_1^2$). We therefore include this assumption in the statement of the second-order necessary condition.

Second-order necessary condition. If $\overset{*}{x}$ is a local minimum of EMP and the second-order constraint qualification holds, then $\overset{*}{Z}{}^T W(\overset{*}{x}, \overset{*}{\lambda}) \overset{*}{Z}$, the projected Hessian of the Lagrangian function, is positive semidefinite.

Sufficient condition. If the first-order necessary condition holds at $\overset{*}{x}$, the Lagrange multipliers are all strictly positive, i.e., $\overset{*}{\lambda} > 0$, and $\overset{*}{Z}{}^T W(\overset{*}{x}, \overset{*}{\lambda}) \overset{*}{Z}$ is positive definite, then $\overset{*}{x}$ is a strong local minimum of problem EMP. Thus in terms of problem MMP, $F_M(\overset{*}{\bar{x}}) < F_M(\bar{x})$ for all $\bar{x}$ such that $|\bar{x} - \overset{*}{\bar{x}}| < \delta$, for some $\delta > 0$.

Note that in the case where all the f_i are linear it is well known that a solution must exist with $n+1$ active functions at $\overset{*}{\bar{x}}$ (see Cheney (1966) for the case l_∞P). Then normally $\overset{*}{Z}$ is null and therefore the second-order conditions are also null. The nonlinear problem, however, can have a unique solution with anything from 1 to $n+1$ functions active at $\overset{*}{\bar{x}}$. This relationship is exactly analogous to that between linear and nonlinear programming. For comments on the special case of l_∞-approximation and the meaning of the Haar condition, see § 11.

2. Use of the equivalent problem EMP. Clearly it is desirable that at every iteration the search direction $\bar{p}$ be a descent direction for F_M, i.e.,

$$F'_M(\bar{x}^{(k)}, \bar{p}) < 0,$$

where $F'_M(\bar{x}^{(k)}, \bar{p})$ is the directional derivative $\lim_{h \to 0+} (1/h)(F(\bar{x}^{(k)} + h\bar{p}) - F(\bar{x}^{(k)}))$. An equivalent condition is that $\bar{p}$ is a descent direction for each function f_i for which $f_i(\bar{x}^{(k)}) = F_M(\bar{x}^{(k)})$ (i.e., $c_i(x^{(k)}) = 0$). A second desirable property for $\bar{p}$ arises from

considering the active set which consists of those constraints corresponding to functions we expect to have the value $F_M(\overset{*}{x})$ at $\overset{*}{x}$. We wish to choose $\bar{p}$ so that the first-order change in these functions predicts that they will all have the same value at $\bar{x}^{(k)}+\bar{p}$. An equivalent condition is

$$\hat{A}(x^{(k)})^T p = -\hat{c}(x^{(k)}), \tag{2.1}$$

and hence

$$\hat{f}_i(\bar{x}^{(k)})+\hat{v}_i(\bar{x}^{(k)})^T\bar{p}=F_M(\bar{x}^{(k)})+p_{n+1}, \qquad i=1,\ldots,t,$$

for some value p_{n+1} (the $(n+1)$st component of p). If the active set included all the constraints which are zero at $x^{(k)}$, then the condition

$$p_{n+1}<0 \tag{2.2}$$

also ensures that $\bar{p}$ is a descent direction for F_M. In fact, at every iteration the active set will initially include all such constraints, but it may be desirable to drop one or more of them from the set to move off a constraint. Since the decrease in F_M is limited to the smallest decrease in any of the functions corresponding to $c_i(x^{(k)})=0$, it is also desirable to insist that

$$a_i(x^{(k)})^T p \geqq 0 \quad \text{for all } i \text{ such that } c_i(x^{(k)})=0. \tag{2.3}$$

Conditions (2.2) and (2.3) ensure that p is a *first-order feasible descent direction* with respect to problem EMP. It is straightforward to show the following from the above remarks.

THEOREM 1. *If* (2.2) *and* (2.3) *hold, then* $\bar{p}$ *is a descent direction for* F_M *and hence a sufficiently small step along it must result in a reduction in* F_M.

Note that (2.2) and (2.3) do not guarantee that p is a feasible direction for EMP. This causes no difficulty since $x_{n+1}{}^{(k+1)}$ is set to $F_M(\bar{x}^{(k+1)})$, and hence it is always possible to obtain a lower feasible point for EMP if (2.2) and (2.3) hold. Consequently it is important to look for a reduction of F_M in the line search and not of a penalty function constructed for EMP.

Thus we see that the importance of EMP is as a device to obtain a search direction $\bar{p}$ along which F_M can be reduced in the line search. We emphasize again that we wish (2.2) and (2.3) to hold so that $\bar{p}$ is a descent direction for F_M, and that the active set nature of the algorithm indicates that (2.1) should also hold. It is the case, however, that (2.1), (2.2) and (2.3) will usually not uniquely define $\bar{p}$, and in the next section we utilize properties of EMP to obtain an initial choice of $\bar{p}$ by solving a quadratic program (QP) based on second-order information incorporated in an approximation to the Lagrangian function. The solution to this QP may not always satisfy (2.1), (2.2) and (2.3), and in subsequent sections we discuss how to modify the initial choice to obtain a satisfactory search direction.

3. The QP subproblem. The solution of EMP is at a minimum of the Lagrangian function in the null space of the active constraint Jacobian at $\overset{*}{x}$. The usual method for solving a general linearly constrained problem is to approximate the objective function by a quadratic function and then determine the search direction by solving some appropriate quadratic program (QP). Consider therefore the quadratic program

$$\min_p L(x^{(k)}, \lambda^{(k)})+\left(e_{n+1}-\hat{A}(x^{(k)})\lambda^{(k)}\right)^T p+\tfrac{1}{2}p^T W(x^{(k)}, \lambda^{(k)})p$$

$$\text{subject to } \hat{A}(x^{(k)})^T p=-\hat{c}(x^{(k)}),$$

where $\lambda^{(k)}$ is an approximation to $\overset{*}{\lambda}$.

An equivalent QP is given by

QP1: $\min_p \frac{1}{2} p^T W(x^{(k)}, \lambda^{(k)})p + e_{n+1}^T p$

subject to $\hat{A}(x^{(k)})^T p = -\hat{c}(x^{(k)})$.

Let us drop the arguments $x^{(k)}$ and $\lambda^{(k)}$ from $\hat{A}$ and W, and let Y and Z be the matrices defined in § 1.1. The matrices Y and Z may be determined from the QR factorization of $\hat{A}$,

$$\hat{A} = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Y & Z \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix},$$

where R is an upper triangular matrix of order t. If $\hat{A}$ has full rank and $Z^T W Z$ is positive definite, then the unique solution of QP1 can be expressed as the sum of two orthogonal components:

$$p = Yp_Y + Zp_Z, \quad \text{where } p_Y \in R^t \text{ and } p_Z \in R^{n+1-t}. \tag{3.1}$$

We have

$$\hat{A}^T p = R^T p_Y = -\hat{c}, \tag{3.2}$$

and p_Y is determined entirely by the constraints of QP1. The vector p_Z is given by the solution of

$$(Z^T W Z)p_Z = -Z^T(e_{n+1} + WYp_Y) \tag{3.3}$$

(see Murray and Wright (1978)).

In subsequent sections we will also wish to refer to a related QP and its solution, namely the one with the same quadratic form but homogeneous constraints:

QP2: $\min_p \frac{1}{2} p^T W p + e_{n+1}^T p$

subject to $\hat{A}(x^{(k)})^T p = 0$.

The solution to this is given by $p = Zq_Z$, where

$$(Z^T W Z)q_Z = -Z^T e_{n+1}. \tag{3.4}$$

At every iteration of our algorithm an attempt is made to set the search direction p to the solution of QP1, but for various reasons this may be inadequate (there may not even be a solution). Much of the detailed discussion of the method is concerned with what action to take in these circumstances. It is important to realize that it is only on attempting to solve the QP that these inadequacies are revealed, and if the solution is not sought in a particular manner, certain deficiencies are not ascertained. We are restricted in the action we may take by requiring that the search direction satisfy (2.1), (2.2) and (2.3), and by a need to limit the computational effort when the information on which it is based has proved suspect. We also wish to arrange the computation so that any computational efforts already invested can still be utilized should the initial QP prove inadequate. In particular, provision must be made for the possibility that the wrong active set is identified at $x^{(k)}$. We will always insist that at the beginning of every iteration the potential active set includes all constraints with zero value at $x^{(k)}$ (and it will normally also include constraints with positive values). In § 5.2 and 6 we discuss how a constraint may then, if desired, be deleted from the active set.

For the moment we assume that analytical Hessians are used to compute W, but in § 7 we discuss finite difference and quasi-Newton alternatives.

4. Lagrange multiplier estimates. Lagrange multiplier estimates are needed to define the matrix W and to determine whether constraints should be deleted from the active set. Clearly it is better to use new information obtained at the current point $x^{(k)}$ rather than use the multipliers of the QP solved at the previous iteration.

The most obvious multiplier estimate is the least squares solution to the overdetermined system based on the first-order necessary conditions, i.e., the solution to

$$\min_{\lambda} \| \hat{A}\lambda - e_{n+1} \|_2^2.$$

Let us denote the solution by λ_L, which may be obtained (see Golub (1965)) by using the QR factorization of $\hat{A}$ which we have already introduced to solve QP1. The estimate λ_L has the following property. Suppose for some reason p is unsatisfactory and we wish to delete a constraint from the active set. If $(\lambda_L)_j < 0$ and we delete constraint j, then the steepest descent direction in the null space of the new active constraint Jacobian is guaranteed to be first-order feasible with respect to the deleted constraint. Define $\tilde{A}$ as $\hat{A}$ with $\hat{a}_j$ deleted, and $\tilde{Z}$ by

$$\tilde{A}^T\tilde{Z}=0, \quad \tilde{Z}^T\tilde{Z}=I_{n+2-t}, \quad \tilde{Z}=[Z \quad z]. \tag{4.1}$$

Then the steepest descent step in the new null space is given by

$$\tilde{Z}\tilde{s}_{\tilde{Z}} = -\tilde{Z}\tilde{Z}^T e_{n+1}, \tag{4.2}$$

and we have

$$\hat{a}_j{}^T\tilde{Z}s_{\tilde{Z}} > 0. \tag{4.3}$$

The Newton step in the null space, given by $\tilde{Z}q_{\tilde{Z}}$, where

$$(\tilde{Z}^T W \tilde{Z})q_{\tilde{Z}} = -\tilde{Z}^T e_{n+1}, \tag{4.4}$$

does not in general satisfy

$$\hat{a}_j{}^T\tilde{Z}q_{\tilde{Z}} \geqq 0. \tag{4.5}$$

See Gill and Murray (1979) for the proof of these statements in the context of linearly constrained optimization.

A more appropriate estimate than λ_L can be obtained by considering the special structure of EMP. The least squares solution is motivated by the fact that the overdetermined set of equations $\hat{A}\lambda = e_{n+1}$ is only an approximation to the set of equations which hold at x_M, the minimum of EMP on the manifold defined by $\hat{c}(x)=0$. However, the $(n+1)$st equation $\hat{e}^T\lambda = 1$ is exactly, not approximately, the equation which holds at x_M. We therefore define another multiplier estimate λ_C as the least squares solution to the first n equations subject to the constraint $\hat{e}^T\lambda_C = 1$. Thus λ_C is the solution to the constrained least squares problem

$$\min_{\lambda} \| \hat{V}\lambda \|_2^2 \quad \text{subject to } \hat{e}^T\lambda = 1.$$

In fact we can show that λ_C is exactly λ_L multiplied by a scalar greater than or equal to one.

THEOREM 2. *Assume $\hat{A}$ has full rank and let λ_L and λ_C be defined as above. Then $\lambda_C = \beta\lambda_L$, where $\beta = 1/\hat{e}^T\lambda_L \geqq 1$.*

Proof. We can assume $\hat{V}$ has full rank, since otherwise the result follows trivially. The vector λ_L satisfies

$$\lambda_L = (\hat{V}^T\hat{V} + \hat{e}\hat{e}^T)^{-1}\hat{e}.$$

It follows from the Sherman-Morrison formula (see Householder (1964, p. 123)) that

$$\lambda_L = \frac{1}{1+\hat{e}^T d} d, \quad \text{where } d = (\hat{V}^T \hat{V})^{-1} \hat{e}. \tag{4.6}$$

Note that $\hat{e}^T d > 0$. Since λ_C is the solution to a constrained optimization problem it follows from the first-order necessary conditions that

$$2\hat{V}^T \hat{V} \lambda_C = \beta_1 \hat{e} \quad \text{for some scalar } \beta_1,$$

and therefore, since $\hat{e}^T \lambda_C = 1$,

$$\lambda_C = \frac{1}{\hat{e}^T d} d,$$

from which the result follows. □

Note that it would be highly ill-advised to compute λ_L and λ_C by computing d via (4.6). If $x^{(k)}$ were equal to x_M, the minimum on the manifold, then $\hat{V}$ would be rank-deficient, but $\hat{A}$ would not in general, and hence if $x^{(k)}$ is close to x_M, the condition number of $\hat{V}^T \hat{V}$ may be much bigger than that of $\hat{A}^T \hat{A}$. Since solving the least squares problem using the QR factorization of $\hat{A}$ is already a better conditioned process than explicitly using $\hat{A}^T \hat{A}$ (via the normal equations) it is clear that using the QR factorization of $\hat{A}$ is far preferable to using (4.6).

Using the scaled estimate λ_C instead of λ_L will result in different decisions about whether to delete constraints from the active set since, as will be explained in § 6, the magnitudes as well as the signs of the estimates are used to make the decision. Furthermore, using λ_C instead of λ_L in general increases the magnitude of W and hence affects both the direction (if $\hat{c} \neq 0$) and the magnitude of the solution to QP1. The following example illustrates that it may often be beneficial to use λ_C rather than λ_L. Let $n=m=1$ and $F(\bar{x})=f_1(\bar{x})=\bar{x}^2$. Let the current estimate of the solution be $\bar{x}^{(k)}=2$. Then $\hat{V}=4$ and $\hat{A}[\,{}^{-4}_{1}]$, $\lambda_L = \frac{1}{17}$ and $\lambda_C = 1$. Using λ_C for W results in the exact step to the solution being taken, but using λ_L results in one seventeen times too big. Clearly similar examples can be constructed with a larger number of active constraints. The choice of λ_C over λ_L essentially arises from the fact that problem MMP is in some sense *naturally scaled*—the functions of MMP cannot be individually scaled without changing the solution while the constraints of NCP can be individually scaled in general.

The second-order Lagrange multiplier estimate μ_W is defined as the exact multipliers corresponding to the solution p of QP1, i.e. the solution to the consistent set of equations

$$\hat{A}\mu_W = e_{n+1} + Wp, \tag{4.7}$$

where p is given by (3.1), (3.2) and (3.3) and λ_C is used to define W. The necessity of requiring W to define μ_W implies that second-order estimates can only be useful in determining whether to delete constraints from the active set. If a constraint is deleted corresponding to $(\mu_W)_j < 0$, then (4.3) will not hold in general. If $\hat{c}=0$, then (4.5) will hold.

The system (4.7) is consistent because of the definition of p. Note that because it is consistent there is no question of a second-order estimate analogous to λ_C—the last equation is already satisfied by μ_W.

Both the estimates λ_C and μ_W will be used to decide when to delete constraints from the active set, as will be discussed in § 6. As explained there, a constraint with a negative component of λ_C will not necessarily be deleted from the active set, since the

multiplier estimates may not be reliable. However, we also use λ_C to define W and there is no good reason to include in W a term with a negative component of λ_C. Therefore, we define W to be

$$W(x^{(k)}) = \sum_{i=1}^{t} \lambda_i' \nabla^2 \hat{f}_i(x^{(k)}),$$

where

$$\lambda' = \frac{1}{\hat{e}^T \lambda''} \lambda''$$

and λ'' is defined by $\lambda_i'' = \max(0, (\lambda_C)_i)$.

5. Properties of solution of QP subproblem. In this section we examine the properties of the solution to QP1. Initially we assume that all constraints with zero value are included in the active set and that $\hat{A}$ has full rank and $Z^T W Z$ is positive definite, so that the solution p is given by (3.1), (3.2) and (3.3) and is unique. We would like p to satisfy (2.1), (2.2) and (2.3). Clearly the constraints of QP1 ensure that (2.1) and (2.3) hold. Thus the only question is whether p is a descent direction for EMP, i.e., whether (2.2) holds. If all the active constraints have the value zero then the following applies:

THEOREM 3. *Suppose that $\hat{c}=0$, $\hat{A}$ has full rank and $Z^T W Z$ is positive definite. Then p, the solution of* QP1, *is a descent direction for* EMP *provided it is not zero.*

Proof. Since $\hat{A}$ has full rank and the columns of Y span the range of the columns of $\hat{A}$, we have $p_Y = 0$. Hence $p = Z p_Z$ and

$$\begin{aligned} e_{n+1}^T p &= {p_Z}^T Z^T e_{n+1} \\ &= -{p_Z}^T Z^T W Z p_Z \quad \text{by (3.3).} \end{aligned}$$

Since $Z^T W Z$ is positive definite, $e_{n+1}^T p$ must be negative if $p_Z \neq 0$, i.e., $p \neq 0$. □

If $p=0$, then by (3.3) $Z^T e_{n+1} = 0$ and hence $\hat{A}\lambda_L = e_{n+1}$ is a consistent set of equations, with $\lambda_C = \lambda_L = \mu_W$. Thus either one of the components of λ_C is negative or zero, or the first- and second-order sufficiency conditions are satisfied and $x^{(k)}$ is a solution of problem EMP. If $p=0$ and at least one of the multipliers is negative, then it is necessary to delete a corresponding constraint from the active set to obtain a descent direction. The procedure for doing this is described in § 6. If $p=0$ and the smallest component of λ_C is zero, the point $x^{(k)}$ may or may not be a solution. In this case special techniques such as described in Gill and Murray (1977) must be used to determine whether to treat the corresponding constraint as active or not. These will not be discussed any further here.

In the next three subsections we discuss the actions which it may be necessary to take to obtain a search direction which satisfies (2.1), (2.2) and (2.3) when we drop the assumptions that $\hat{c}=0$, $\hat{A}$ has full rank and $Z^T W Z$ is positive definite.

5.1 Positive active constraints. In practice it will rarely be the case that $\hat{c}=0$, so we now drop this assumption. We note that if we were sufficiently restrictive in the definition of the active set (e.g., choose only one constraint active) then we could force this condition to be true. As will be shown in § 10, however, it is important for the efficiency of the algorithm *not* to be too restrictive in the definition of the active set. This may appear to negate the significance of Theorem 3, but this is not the case. Although dropping the assumption certainly means that Theorem 3 no longer holds, since both $Y p_Y$ and $Z p_Z$ could be ascent directions, we can infer that if $\|\hat{c}\|$ is small

(and it approaches zero near the solution), then p is likely to be a descent direction. If, after QP1 is solved, it transpires that $e_{n+1}^T p > 0$, then, provided either Yp_Y or Zp_Z is a descent direction, a suitable descent direction can be chosen as follows:

$$Yp_Y + \gamma Zp_Z \quad \text{if } e_{n+1}^T Yp_Y < 0,$$
$$\gamma Yp_Y + Zp_Z \quad \text{if } e_{n+1}^T Zp_Z < 0,$$

for some γ satisfying $0 < \gamma \leqq 1$. If neither component is a descent direction, then a further possibility is to replace Zp_Z by the solution of QP2, i.e. Zq_Z where q_Z is given by (3.4). Provided $Z^T e_{n+1} \neq 0$, the vector Zq_Z is a descent direction, and hence so is $\gamma Yp_Y + Zq_Z$ for some γ, $0 < \gamma \leqq 1$.

When $Z^T e_{n+1} = 0$ and Yp_Y is an ascent direction, it is necessary to delete a constraint to obtain a descent direction. One possibility would be to delete constraint j (say) where $\hat{c}_j$ is the largest component of $\hat{c}$, since it is not strictly necessary to be concerned about whether the resulting search direction has a positive inner product with $\hat{a}_j$ ((2.3) applies only to constraints with value zero). However, clearly this may lead to a very small step being taken along the search direction with this constraint being forced immediately back into the active set. The following result shows that when Yp_Y is uphill, a constraint can always be found with a negative multiplier estimate and hence can be deleted more safely. The result is also useful when $Z^T e_{n+1} \neq 0$, since then the fact that Yp_Y is uphill implies there are too many constraints in the active set.

THEOREM 4. *Assume $\hat{A}$ has full rank and let Yp_Y be defined by* (3.2). *If $e_{n+1}^T Yp_Y > 0$, then one of the components of λ_C is negative.*

Proof. We know that λ_C is a positive multiple of λ_L. The vector λ_L satisfies

$$\hat{A}\lambda_L = YY^T e_{n+1}. \tag{5.1}$$

This is a characterization of the least squares solution using the projector matrix YY^T (see Stewart (1973, p. 228)). Thus

$$\lambda_L^T \hat{A}^T Yp_Y = e_{n+1}^T YY^T Yp_Y$$

and hence

$$-\lambda_L^T \hat{c} = e_{n+1}^T Yp_Y > 0.$$

Since $\hat{c} \geqq 0$ it follows that at least one of the components of λ_L and hence λ_C must be negative. □

It also follows from the above proof that if $e_{n+1}^T Yp_Y = 0$, then either $\hat{c} = 0$ (covered by Theorem 3), or the minimum element of λ_C is zero or negative.

It follows from Theorem 4 that if Yp_Y is an ascent direction, we can delete the constraint corresponding to a negative component of λ_C to obtain a first-order feasible descent direction. The procedure for doing this is described in § 6.

5.2 Avoiding rank deficiency in the active constraint Jacobian. In this section we demonstrate how the active set can always be chosen so as to avoid rank deficiency in $\hat{A}$. We first consider the consequences of $\hat{A}$ being rank deficient. If $\hat{A}$ is rank deficient and $\hat{c} \neq 0$, it is not possible in general to satisfy the constraints of QP1 since they may not be compatible. In such circumstances it might be thought that an adequate compromise would be the least squares solution to $\hat{A}^T p \approx -\hat{c}$, but this may not be first-order feasible with respect to EMP, and hence may not be a descent direction for the minimax function even if it is a descent direction for EMP. Clearly it is desirable to restrict the number of constraints in the active set so that $\hat{A}$ has full rank.

The way this is done is as follows. Given a set of candidates for the active set, we determine which are to be actually included in the active set during the QR factorization of $\hat{A}$. Assuming the problem is well-scaled, a reasonable order in which to consider the candidates for inclusion is given by the size of the constraint values. Therefore we order the potential columns of $\hat{A}$ by increasing size of $\{c_j\}$ before proceeding to do a QR factorization of the matrix by columns without column pivoting (except where several columns correspond to the same magnitude of c_j). If it transpires during the factorization that any of the potential columns of $\hat{A}$ is linearly dependent on those already included, then the corresponding constraint is not included in the active set. Clearly such a process results in a matrix $\hat{A}$ that has full column rank.

An example which illustrates this procedure is the following. Suppose the initial candidates for the active set are

$$c=\begin{bmatrix} 0 \\ 10^{-4} \\ 10^{-3} \\ 10^{-2} \end{bmatrix} \quad \text{and} \quad A=\begin{bmatrix} 1 & 1 & 0.5 & 0 \\ 0 & 0 & 0.5 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Then the first and third constraints are selected for the active set, and the second and fourth are ignored.

We must now show that omitting any constraint, say $c_r(x)$, from the active set to avoid rank deficiency in $\hat{A}$ does not cause (2.3) to be violated. Strictly speaking, we need not be concerned with constraints for which $c_r(x)>0$, but for the moment we will not assume $c_r(x)=0$ since the following also serves to illustrate why we put potentially active constraints in increasing order.

Let a_r be the gradient of c_r. We have

$$a_r=\sum_{i=1}^{s} w_i\hat{a}_i \tag{5.2}$$

for some index s and scalars w_i, $i=1,\ldots,s$. Since the constraints were ordered we also have

$$c_r\geqq\hat{c}_s\geqq\hat{c}_{s-1}\geqq\ldots\geqq\hat{c}_1. \tag{5.3}$$

It follows from (3.2) and (5.2) that

$$a_r^Tp=-\sum_{i=1}^{s} w_i\hat{c}_i. \tag{5.4}$$

Since we have no a priori information about how the sizes of the $\{w_i\}$ would change if the columns were ordered differently, we have by putting the constraints in increasing order attempted to prevent $|a_r^Tp|$ from being large and in particular to prevent a_r^Tp from being a large negative number. Ordering the constraints also ensures that the omitted constraint c_r have as large a value as possible, and these two facts combine to make it as unlikely as possible that constraint r will prevent a significant step from being taken along p. It follows from (5.3) and (5.4) that if $c_r=0$, then $a_r^Tp=0$ and hence (2.3) is satisfied. However, if it is necessary to delete a further constraint from the active set as described in § 6, then (2.3) may no longer hold. This difficulty is circumvented by allowing a zero step to be taken along p (see § 8) and then reconsidering the choice of active set in the next iteration.

5.3 Projected Hessian not positive definite. If Z^TWZ is not positive definite, it makes no sense to use the solution of (3.3), even if it exists, to define the direction of search. Such a direction is a step to a stationary point which is not a minimum of the quadratic form on the subspace defined by the constraints and may even be a maximum. It also makes no sense to reduce the number of active constraints (the projected Hessian will still be indefinite), since one means of ensuring that the projected Hessian is positive definite is to increase the number of active constraints. An alternative definition of the search direction which is satisfactory is to utilize the modified Cholesky algorithm of Gill and Murray (1974).

The following numerically stable matrix factorization is computed:

$$Z^TWZ+E=LDL^T.$$

The matrix E is a nonnegative diagonal matrix large enough to make Z^TWZ+E numerically positive definite. The *modified* Newton direction in the null space of $\hat{A}$ is then defined as Zq_Z where

$$LDL^Tq_Z=-Z^Te_{n+1}. \tag{5.5}$$

Recently several more complicated methods have been suggested for handling indefinite Hessians; see Fletcher and Freeman (1977) and Moré and Sorenson (1979). All of these, including the Gill and Murray algorithm, provide ways of obtaining directions of negative curvature. As will be explained shortly, however, these are not so useful in the context of nonlinear constraints; our main concern here is simply to obtain a reasonable descent direction.

Like the Newton direction Zq_Z in the positive definite case, the vector Zq_Z is a descent direction provided that $Z^Te_{n+1}\neq 0$. If $Z^Te_{n+1}=0$ and $e_{n+1}^TYp_Y<0$ we can simply set $p=Yp_Y$; if $Z^Te_{n+1}=0$ and $e_{n+1}^TYp_Y>0$ we have already explained that a constraint may be deleted. Similarly if $e_{n+1}^TYp_Y=0$, but a component of λ_C is negative, a constraint may be deleted.

If Z^TWZ is not positive semidefinite and $e_{n+1}^TYp_Y=0$, $Z^Te_{n+1}=0$ and $\lambda_C\geqq 0$, then the point $x^{(k)}$ is a constrained saddle point (or even a maximum). It is desirable to compute a feasible arc of negative curvature with respect to Z^TWZ, which must exist. For the case of linearly constrained minimization of a differentiable function, Gill and Murray (1974) have shown how to compute a vector of negative curvature using the modified Cholesky factorization. However, it is not possible in general to compute such an arc for the nonlinearly constrained case given only W—it is necessary to know all the individual constraint Hessians. Such a computation would hence require an unreasonable amount of storage as well as computational effort. We therefore suggest the following. If $\hat{c}_j>0$ for some j, delete the jth active constraint and avoid, or at least postpone, the problem. If $\hat{c}=0$, compute the direction of negative curvature assuming the constraints are linear, which is thus feasible to first order, and try stepping along it. If the minimax function is lower at the new point, then take this as $x^{(k+1)}$. Otherwise an alternative is to try to obtain a lower point using a function-comparison method such as described in Swann (1972). There is no reasonable procedure which is guaranteed to work in this situation. However, it should be noted that such a situation is unlikely to occur since the basic modified Newton iteration of solving successive quadratic programs seeks to avoid saddle points and maxima.

6. Deleting constraints from the active set. In § 5 we explained that when $Z^Te_{n+1}=0$ and a multiplier is negative, it is necessary to delete a constraint from the

active set in order to obtain a first-order feasible descent direction. It is well known in the context of constrained optimization that it is ill-advised to wait until $Z^T e_{n+1}$ is zero or very close to zero before deleting a constraint corresponding to a negative multiplier estimate, since doing so is solving the wrong equality-constrained subproblem with unnecessary accuracy (minimizing on a manifold). It is also well known that it is even more ill-advised to delete a constraint too early, when the multiplier estimates are not yet reliable, since this may cause the constraint to be repeatedly added to and dropped from the active set. Gill and Murray (1979) suggest computing both λ_L and μ_W when possible, never considering deleting a constraint unless the estimates have the same sign and agree within a certain tolerance. Here we use λ_C instead of λ_L. Notice that $\lambda_C = \mu_W$ when $Z^T e_{n+1} = 0$. A further possibility is to insist that the following hold before deleting a constraint:

$$\| Z^T e_{n+1} \| < \delta \cdot \min\Big(1, -\min_i (\lambda_C)_i\Big),$$

where $\delta \leqq 1$ is a constant, say $\delta = 1$. In effect, this test ensures that the higher the uncertainty that a multiplier is negative, the greater the accuracy to which the minimum is approximated on the manifold.

In § 5.1 we also pointed out a further situation in which we delete a constraint. This is when $e_{n+1}^T Y p_Y > 0$, which means (as shown in Theorem 4) that $(\lambda_C)_j < 0$ (and $\hat{c}_j > 0$) for some j. A constraint is always deleted in this situation since this is a clear indication that there are too many constraints in the active set.

In the rest of this section we discuss what search direction p to choose after deleting constraint j with $(\lambda_C)_j < 0$. Define $\tilde{A}$, $\tilde{Z}$ and z by (4.1). Define $\tilde{c}$ to be $\hat{c}$ with $\hat{c}_j$ deleted. Since $\tilde{A}$ corresponds to the new active set we wish p to satisfy

$$\tilde{A}^T p = -\tilde{c}, \tag{6.1}$$

corresponding to (2.1), and also $p_{n+1} < 0$, i.e., (2.2). If $\hat{c}_j = 0$ then we must have

$$\hat{a}_j^T p \leqq 0 \tag{6.2}$$

to satisfy (2.3), but as explained in § 5.1, this is desirable even if $\hat{c}_j > 0$.

As we noted in § 4, the steepest descent step $\tilde{Z} s_{\tilde{Z}}$ in the null space of the new set of constraints given by (4.2) must satisfy (6.2), but the Newton step given by (4.4) may not satisfy (6.2). A step satisfying (6.2) which is preferable to the steepest descent step is a combination of the Newton step in the null space of $\hat{A}^T$, i.e., $Z q_Z$, and the steepest descent step in the new direction permitted by deleting the constraint.

THEOREM 5. *Assume $\hat{A}$ has full rank. Suppose $(\lambda_C)_j < 0$. Define $\tilde{A}$, $\tilde{Z}$ and z as in* (4.1). *Let $r_{\tilde{Z}}$ be defined by*

$$r_{\tilde{Z}} = \begin{bmatrix} q_Z \\ -z^T e_{n+1} \end{bmatrix},$$

where q_Z is defined by (5.5). *Then $p = \tilde{Z} r_{\tilde{Z}}$ satisfies* (6.1), (2.2) *and* (6.2).

Proof. The proof may be found in Overton (1979).

Note that there is no guarantee that the corresponding range space step $\tilde{Y} p_{\tilde{Y}}$ to the modified set of constraints satisfies either (2.2) or (6.2). This step is defined by

$$\tilde{A}^T \tilde{Y} p_{\tilde{Y}} = -\tilde{c},$$

where $\tilde{c}$ is $\hat{c}$ with $\hat{c}_j$ deleted and the orthogonal columns of $\tilde{Y}$ span the range of $\tilde{A}$. However, since $\tilde{Z} r_{\tilde{Z}}$ satisfies (2.2) and (6.2) with strict inequality, so does $\gamma \tilde{Y} p_{\tilde{Y}} + \tilde{Z} r_{\tilde{Z}}$ for small enough γ.

Although $\tilde{Z}r_{\tilde{Z}}$ is guaranteed to satisfy the required properties (6.1), (2.2) and (6.2), the Newton step $\tilde{Z}q_{\tilde{Z}}$ may be preferable if it satisfies (6.2). In fact, the step $\tilde{Z}p_{\tilde{Z}}$, defined by

$$\tilde{Z}^T W \tilde{Z} p_{\tilde{Z}} = -\tilde{Z}^T (e_{n+1} + W \tilde{Y} p_{\tilde{Y}}),$$

may be preferable to either, but this is not guaranteed to even be a descent direction. We recommend computing $\tilde{Z}q_{\tilde{Z}}$ and performing the appropriate inner product to check whether it is a feasible descent direction, falling back on $\gamma \tilde{Y} p_{\tilde{Y}} + \tilde{Z} r_{\tilde{Z}}$ if it is not, and substituting $\tilde{Z}q_{\tilde{Z}}$ for $\tilde{Z}r_{\tilde{Z}}$ if it is. This may be done even if $(\mu_W)_j > 0$, which may be the case if we are deleting a constraint when $e_{n+1}^T Y p_Y > 0$, since Theorem 4 does not hold with μ_W substituted for λ_L (even if W is assumed to be positive definite). Although when exact Hessians are available it is normally inadvisable to delete a constraint and take a Newton step when $(\mu_W)_j > 0$, in this case it is desirable to delete a constraint and the only question is what step to take.

In all of the above, when a constraint is deleted from the active set it is not necessary to recompute the factorizations of $\tilde{A}$ and $\tilde{Z}^T W \tilde{Z}$ from scratch. They can be obtained by updating the factorizations already available, as described in Gill, Golub, Murray and Saunders (1974) and Gill and Murray (1974).

Note that we have been able to always obtain a satisfactory choice for p by deleting only one constraint with a negative multiplier estimate.

7. Finite difference and quasi-Newton approximations to the Hessian. It may be that the Hessians $\{\nabla^2 f_i\}$ are unavailable either because they are too expensive to evaluate or too difficult to determine. Recall from § 1.1 that

$$W = \begin{bmatrix} \overline{W} & 0 \\ 0 & 0 \end{bmatrix},$$

so that when analytical Hessians are used a matrix of order n is stored. The two basic alternatives are using a finite difference approximation or a quasi-Newton approximation.

A finite difference approximation to $\overline{W}$, unlike a quasi-Newton approximation, requires extra gradient evaluations to be done at each iteration. However, it is important to note that extra gradient evaluations of only the t active functions are required, where t may be significantly less than m, the number of functions which must be evaluated at each iteration. Furthermore, since the matrix $\overline{W}$ is not explicitly required to compute p we can form a direct finite difference approximation to $\overline{W}\overline{Z}$ by differencing the gradient of the Lagrangian function along the columns of $\overline{Z}$. It is also necessary to difference the gradients along YP_Y to obtain an approximation to WYp_Y if we wish to compute μ_W. Thus the t active gradients must each be evaluated only $n-t+2$ times, which may be considerably less than the $n+1$ required to approximate $\overline{W}$.

When a quasi-Newton method is used, two approaches are possible. Powell (1977) has shown that it is possible to obtain local R-superlinear convergence using a positive definite approximation to the full matrix $\overline{W}$ even though $\overline{W}$ may have negative eigenvalues at the solution. Murray and Wright (1978) have suggested approximating the projected Hessian $\overline{Z}^T \overline{W} \overline{Z}$, which may have much smaller dimension than $\overline{W}$.

One of the prime applications of the minimax algorithm is to data fitting problems in the l_∞-norm (see § 11). These problems typically have a large number of functions f_i (observations), but a relatively small number of variables. Furthermore, if

the functions are not too highly nonlinear the number of active constraints at the solution may be close to $n+1$ (it would be exactly $n+1$ if the problem were linear). Thus it may often be that $t \ll m$ and $n+1-t \ll n$, exactly the situation where the finite difference scheme is most efficient. Since the finite difference scheme will normally produce a better direction of search at each iteration than a quasi-Newton method and also has a higher rate of convergence, it will normally take significantly fewer iterations to reach the solution. Thus the additional work at each iteration may result in a substantial saving overall.

8. The steplength. Given a direction of search $\bar{p}$ which is a descent direction for F_M, we must determine a steplength α to take along it. We use the algorithm described in Murray and Overton (1979a) which is designed for the minimax and related nondifferentiable functions. This includes a facility for varying the limit on the effort to be expended, producing anything from an algorithm which normally takes a single function evaluation to one which does an exact linear search. An initial guess α_0, where F_M is to be evaluated first, is required. We set α_0 to either one, or the estimated step to the nearest inactive constraint using the linear approximations at $x^{(k)}$, if this is less than one. Thus

$$\begin{aligned} &\alpha_0 = \min\{1, \alpha_0'\}, \quad \text{where} \\ &\alpha_0' = \min\left\{ -\left(c_i / a_i^T p\right) \middle| a_i^T p < 0 \text{ and } i \text{ not in active set} \right\}. \end{aligned} \tag{8.1}$$

It is possible that $\alpha_0 = 0$, and hence the algorithm must allow for this possibility. In this case the appropriate inactive constraint must be added to the active set and the iteration repeated.

9. Flowchart of the algorithm. We summarize the basic iteration of the algorithm in the flowchart in Fig. 1. For simplicity we have omitted any tolerances from the flowchart, though clearly in practice these must be included. The parameters γ_1 and γ_2 are optional. For the results presented in § 14, γ_1 is set to 1 when possible but γ_2 is always set to zero. The latter is done because near a saddle point or after deleting a constraint a poor range space direction can swamp a good null space direction if $\gamma_2 > 0$. The notation S is used to mean either W or a finite difference or quasi-Newton approximation to W.

10. Selecting the active set. The success of the algorithm we have described depends on being able to make a reasonable choice of the active set at each iteration. If constraints are required to have very small magnitude to be included in the active set, then the iterates will follow the constraint boundaries very closely and the convergence will be slow. Conversely if too many constraints are selected as active the directions of search may be poor. In this section we describe an active set strategy that has been most successful for the numerical experiments we have carried out.

Clearly one feature required of the active set strategy is that, as the iterates approach $\hat{x}$, it should become successively more difficult for constraints with magnitudes significantly greater than zero to be included. One way to accomplish this is to have the strategy depend on a parameter which is reduced as the solution is approached or if any difficulties arise. We found that reducing a parameter in this way was not satisfactory since it can easily happen that the parameter is reduced too much, making it impossible to ever include all the correct active constraints. Instead we take the following approach. There is always at least one constraint active with value identically zero, so the first decision is whether to include a second constraint. If

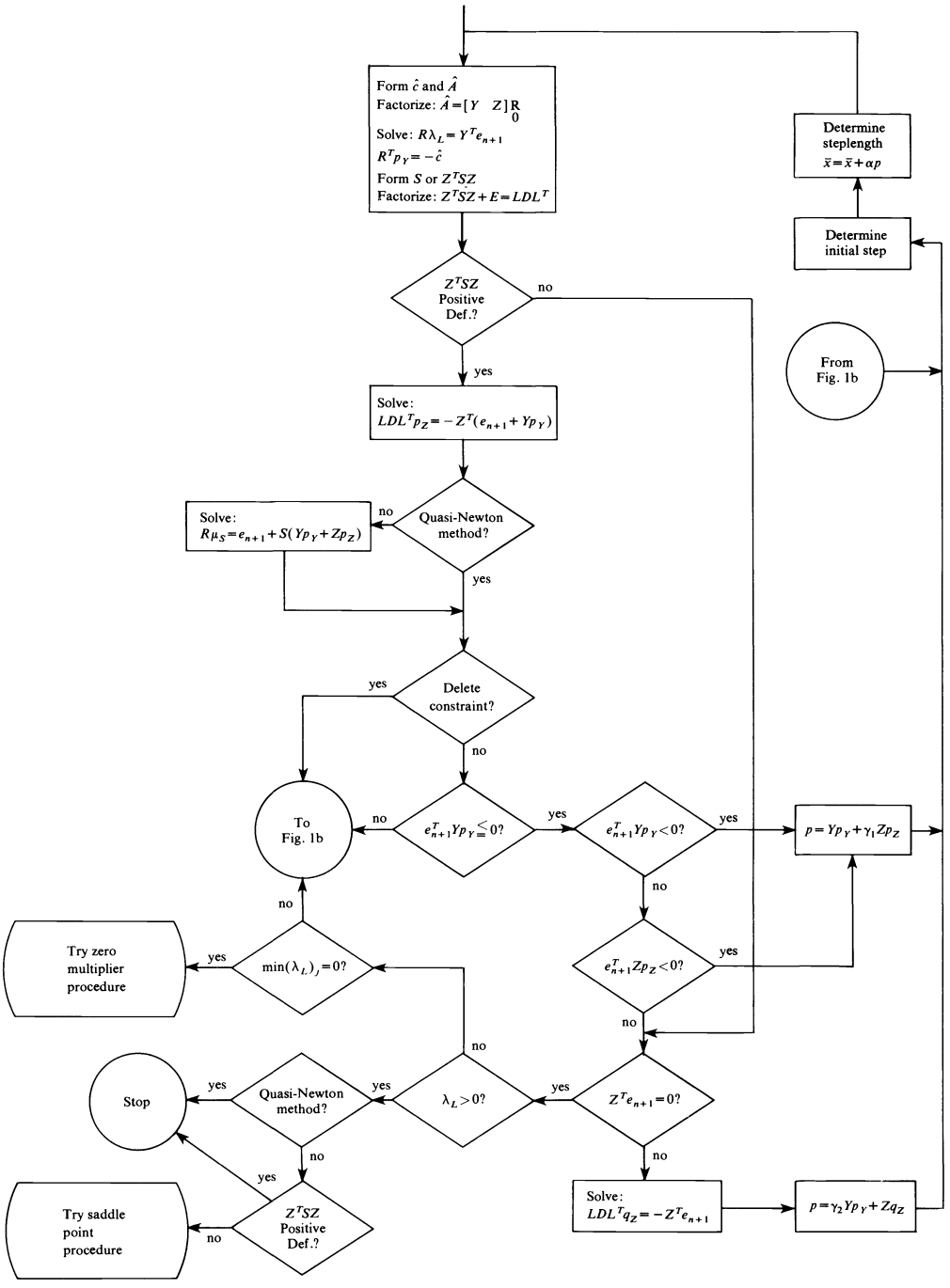

Form $\hat{c}$ and $\hat{A}$
Factorize: $\hat{A}=[Y \quad Z]\begin{matrix}R\\0\end{matrix}$
Solve: $R\lambda_L = Y^T e_{n+1}$
$R^T p_Y = -\hat{c}$
Form S or Z^TSZ
Factorize: $Z^TSZ+E=LDL^T$
Z^TSZ Positive Def.?
no
yes
Solve: $LDL^T p_Z = -Z^T(e_{n+1}+Yp_Y)$
Quasi-Newton method?
no
Solve: $R\mu_S = e_{n+1}+S(Yp_Y+Zp_Z)$
yes
Delete constraint?
yes
no
$e_{n+1}^T Yp_Y \leq 0$?
no
yes
To Fig. 1b
$e_{n+1}^T Yp_Y < 0$?
yes
$p=Yp_Y+\gamma_1 Zp_Z$
no
$e_{n+1}^T Zp_Z < 0$?
yes
no
$\min(\lambda_L)_j = 0$?
yes
Try zero multiplier procedure
no
$\lambda_L > 0$?
yes
no
$Z^Te_{n+1}=0$?
yes
no
Quasi-Newton method?
yes
Stop
no
Z^TSZ Positive Def.?
yes
no
Try saddle point procedure
Solve: $LDL^T q_Z = -Z^Te_{n+1}$
$p=\gamma_2 Yp_Y+Zq_Z$
Determine steplength $\bar{x}=\bar{x}+\alpha p$
Determine initial step
From Fig. 1b

FIG. 1a.

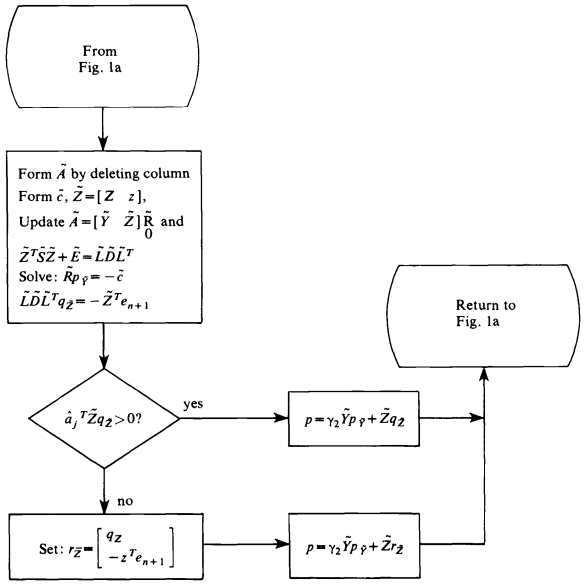


FIG 1b.

this is done the decision of whether to include others is tied to the magnitude of the second constraint. Thus the required objective will be achieved, provided that the first decision is made correctly so that any second active constraint approaches zero as the iterates approach the solution.

Let us order the constraints so that $0=c_1\leqq c_2\leqq \ldots \leqq c_m$, with the $\{f_i\}$ and $\{v_i\}$ correspondingly ordered. We include in the active set all constraints with magnitude less than a very small tolerance, say κ_0. In order to compare the larger constraint magnitudes we must scale them in some way. For problem l_∞P (see § 11) we define the scaled values by $\bar{c}_i=c_i/F_\infty$, since the value of F_∞ can only be very small if all the constraints values are very small. For problem MMP the value of F_M could be zero or negative, so we define the scaled values by $\bar{c}_i=c_i/(1+|f_1|+|f_2|)$. Scaling the values is necessary since two functions with values 999 and 1000 are likely to both be active if one is, while this is not true of functions with values 1 and 2. Notice that the existence of any function in problem MMP with negative values much less than F_M does not affect the definition of $\bar{c}_i$, which is appropriate since it does not affect the solution.

If only one constraint is active, then $\|v_1(\overset{*}{x})\|=0$. Therefore the decision of whether to include a second active constraint c_2 is made as follows. If $\|v_1\|$ is small, say $\|v_1\|<\kappa_1$, then c_2 is included only if $c_2<\kappa_2\|v_1\|^2$, where $\kappa_2>1$, a test which can be justified by Taylor expansions around the solution. If $\|v_1\|\geqq\kappa_1$, then c_2 is included only if $\bar{c}_2<\kappa_3$, where $0<\kappa_3<1$.

It remains to test the other constraint values against c_2. We include c_i if c_{i-1} is included and $\bar{c}_i<\kappa_3$ and $\bar{c}_i<(\bar{c}_{i-1})^{\kappa_4}$, $i=3,4,\ldots,\min(m,n+1)$, where $0<\kappa_4<1$. Thus, for example, if $\kappa_4=0.5$, c_3 is included if c_2 is included, $\bar{c}_2=10^{-6}$ and $\bar{c}_3=10^{-4}$, but not if $\bar{c}_2=0.1$ and $\bar{c}_3=0.5$. Note that we always have $\bar{c}_2\overset{*}{x}1$.

We found that it was also important not to include a constraint in the active set if it was included but subsequently deleted in the previous iteration. It was also helpful to include a constraint requiring only that $\bar{c}_i<\kappa_3$ if the previous line search chose a step to a point of discontinuity involving the corresponding function.

Sometimes some constraints are much smaller than others which should also be in the active set (for example if some of the functions are linear). In such a situation it is not appropriate to compare all the constraints with c_2. The remedy is to let the first constraint with value significantly greater than the machine precision play the role of c_2. However, then v_1 must be replaced by a projected gradient $Z^T v_1$ in the tests, and the factorization of $\hat{A}$ and the accumulation of the transformations which form Z must be performed as the active set selection proceeds. We have not yet implemented this modification.

Our current active set strategy is the above with κ_0 set to the square root of the machine precision, $\kappa_1 = 0.1$, $\kappa_2 = 25$, $\kappa_3 = 0.25$ for l_∞P, $\kappa_3 = 0.1$ for MMP, and $\kappa_4 = 0.5$.

11. The special case of the least l_∞-norm problem. In this section we consider problem l_∞P, defined in § 1. Problem l_∞P is an important special case of MMP. Note that l_∞P must have a solution, whereas MMP may not. Clearly it is preferable to treat l_∞P in a special manner rather than just treat $|f_i|$ as $\max(-f_i, f_i)$. If we eliminate the possibility that F_∞ is zero at the solution, i.e. *all* the $\{f_i\}$ are zero at the solution, then we can observe that only *one* of each pair of constraints $x_{n+1} - f_i(\bar{x}) \geqq 0$, $x_{n+1} + f_i(\bar{x}) \geqq 0$, can be active at the solution. Thus defining $\sigma_i = \operatorname{sgn}(f_i(\bar{x}))$ for any $\bar{x}$ it is straightforward to handle l_∞P by using the algorithm described for MMP, replacing f_i by $\sigma_i f_i$ everywhere. The only places where it is necessary to consider the *inactive* constraints $\{c_i' = x_{n+1} + \sigma_i f_i\}$ are in the determination of the initial guess at the steplength α_0, and in the steplength algorithm itself. Thus α_0 must be set to

$$\alpha_0 = \min\{1, \alpha_0', \alpha_0''\},$$

where α_0' is defined by (8.1) and α_0'' is given by

$$\alpha_0'' = \min\left\{ -\frac{c_i'}{\nabla c_i'^T p} \,|\, \nabla c_i'^T p < 0,\ i = 1, 2, \ldots, m \right\}.$$

The changes which must be made to the steplength algorithm are indicated in Murray and Overton (1979a).

If F_∞ is zero at the solution there is still no difficulty with this approach provided that $m \geqq n+1$, since then of the $2m$ constraints active for the equivalent problem corresponding to l_∞P, at most $n+1$ can be included in the active set to obtain a full rank active constraint Jacobian. Thus including in the active set at most one constraint of each pair causes no difficulty. Such a situation is a highly degenerate one, but the point is that if the situation does arise the algorithm will take care of it efficiently. The usual source of l_∞P is data fitting problems, so we expect almost always to have $m \geqq n+1$. However, if it does happen that we wish to solve l_∞P with $m \leqq n$, then the above technique may be very inefficient since both constraints in a pair of constraints active at the solution cannot be put into the active set. Instead of making a complicated modification to take care of this unlikely possibility, we recommend writing the problem explicitly in the form MMP and solving this directly.

It is straightforward to generalize the above to an algorithm which can be used to minimize a more general function,

$$F_{GM}(\bar{x}) = \max\left\{ \max_{1 \leqq i \leqq m_1} |f_i(\bar{x})|, \max_{m_1+1 \leqq i \leqq m} f_i(\bar{x}) \right\},$$

assuming $m_1 \geqq n+1$ if $m_1 \neq 0$. Since coping with the general case introduces very little

extra overhead, our implementation of the algorithm handles this wider class of functions. In this way one algorithm takes care of both MMP ($m_1=0$) and l_∞P ($m_1=m$).

11.1 The Haar condition. We comment here on the meaning of the Haar condition since this is usually discussed in the context of l_∞-approximation. Let us first consider the case that the f_i are linear. The Haar condition is said to hold for these functions if every $n\times n$ submatrix of V is nonsingular, where V is the $n\times m$ matrix whose columns are the gradients of the $\{f_i\}$. Consider problem EMP, which is now a linear programming problem. A necessary condition for $\overset{*}{x}$ to be a solution to EMP is that $\overset{*}{V}\overset{*}{\lambda}=0$ and $\overset{*}{\lambda}\neq 0$ (since $\hat{e}^T\overset{*}{\lambda}=1$). Thus the requirement that the Haar condition holds implies that there are at least $n+1$ active constraints at the solution with none of the multipliers equal to zero, and hence that the solution is unique. Most algorithms which solve the linear minimax problem do so by solving a linear program related to ELP. Thus whether or not the Haar condition holds is quite irrelevant to the difficulty of solving l_∞P, since zero multipliers cause no real difficulty in solving linear programs. Degeneracy, which occurs when the matrix $\hat{A}$ is rank deficient and can in theory cause problems in solving a linear program, can occur whether or not the Haar condition holds. The significance of the Haar condition is that if it does not hold the solution may not be unique, and hence one may be interested in a "strict" solution to the data approximation problem, i.e., that solution of l_∞P which reduces the inactive functions as much as possible (see Rice (1969) and Brannigan (1978)).

The Haar condition is much stronger than necessary to ensure uniqueness. A slightly more reasonable condition is that there be $n+1$ constraints with nonzero multipliers active at the solution of l_∞P (see Jittorntrum and Osborne (1979, p. 3) for an example showing that this condition is still stronger than necessary to ensure uniqueness). This condition cannot be checked without solving l_∞P, but solving l_∞P is actually much easier to do than checking whether the Haar condition holds.

Let us now consider the nonlinear case. The Haar condition is then said to hold at a point $\bar{x}$ if every subset of gradients of functions with zero constraint values at $\bar{x}$ is linearly independent. As in the linear case we are really only concerned with the situation at $\overset{*}{x}$. Thus the Haar condition holds at $\overset{*}{x}$ if every $n\times n$ submatrix of $\overset{*}{V}$ has full rank. Again it follows that if the Haar condition holds there must be $n+1$ constraints with nonzero multipliers active at $\overset{*}{x}$. This condition is however a much more unreasonable one than in the linear case. There must always exist a solution to the linear problem where $n+1$ constraints are active, but the nonlinear problem can have a unique solution with anything from 1 to $n+1$ constraints active. If the Haar condition does hold at $\overset{*}{x}$ then $\overset{*}{Z}$ is null and the problem can be adequately solved by a method using only first-order information. Thus our algorithm has been designed with the assumption that the Haar condition often does not hold. When l_∞P arises from data fitting problems it may sometimes be the case that the Haar condition can be expected to hold at $\overset{*}{x}$. However, the only way to ascertain whether the Haar condition does indeed hold is to solve the problem. Since we cannot be sure at the outset, it is clearly unsatisfactory to be using an algorithm which uses first order-information only and hence may indicate that the Haar condition does not hold by extremely poor performance. For many data fitting problems the number of active constraints at the solution may be *close* to $n+1$, and as pointed out in § 7 this means that using a finite difference approximation to Z^TWZ, which has order $n+1-t$, may be quite inexpensive.

Cromme (1972) discusses in a broader setting a weaker condition than the Haar condition at $\overset{*}{x}$ called strong uniqueness which still ensures that a particular algorithm using only first-order information converges quadratically. Strong uniqueness implies that $n+1$ constraints are active at $\overset{*}{x}$, but is weaker than the Haar condition since it permits active constraints with zero multipliers. Jittorntrum and Osborne (1979) discuss a still weaker condition which arises from examining the curvature in the null space of the active constraint Jacobian with gradients corresponding to zero multipliers deleted. This weaker condition also ensures that a first-order method has quadratic convergence. The sufficient conditions for a minimum given in § 1.1 are much weaker than any of these conditions since they permit $t(\overset{*}{x})$ to be less than $n+1$, in which case an efficient algorithm must approximate Z^TWZ. These conditions could themselves be weakened if curvature corresponding to zero multipliers were examined.

12. Relationships to other algorithms. A number of other algorithms have been proposed to solve the nonlinear minimax problem. Most of these are first-order methods which are satisfactory if the number of active functions at the solution is $n+1$ but very slow otherwise. It was not until recently that special algorithms for MMP which make use of second-order information appeared. Han (1977b), (1978a), (1978b) suggests methods which solve a sequence of quadratic programming problems using a quasi-Newton approximation to W. These will be discussed further shortly. Watson (1979) and Hald and Madsen (1978) propose two-stage methods which begin by using the first-order methods of Anderson and Osborne (1977) and Madsen (1975), respectively, and switch to solving a system of nonlinear equations using the second-order information in W when it is thought that the active set has been identified. The system is of order $n+1+t$; i.e., the multipliers and variables are all obtained at once. Recall that for problem l_∞P, t is often close to $n+1$, so the systems that Watson and Hald and Madsen solve may be much larger than the ones we solve. The direction of search obtained is not necessarily a descent direction for the minimax function but only for the residual of the nonlinear system. A method related to the second stage of these methods was given by Hettich (1976). Conn (1979) presents a method which is derived from the point of view of a nondifferentiable penalty function. It is related to the algorithm of Charalambous and Conn (1978) but uses second-order information. We discuss this method further below. Other methods which use second-order information and are related to Han's method are discussed by Charalambous and Moharram (1978), (1979) and Wierzbicki (1978).

Our algorithm is most closely related to the methods of Han (1977b), (1978a) and Conn (1979), so we discuss these further here. The primary difference between our method and Han's method is that we attempt to identify the active set at each iteration and then solve an equality-constrained quadratic program (EQP), modifying the resulting direction of search and the active set if necessary, while Han solves an inequality-constrained quadratic program (IQP), thus implicitly selecting the active set associated with the solution of IQP. The IQP has the form:

$$\text{IQP:} \qquad \begin{aligned} &\min \tfrac{1}{2} p^TWp + e_{n+1}^T p \\ &\text{subject to } A^Tp \underset{=}{\geq} -c, \end{aligned}$$

where c and A are the vector and matrix of all the constraints and their gradients.

The dichotomy of whether to solve EQP or IQP occurs at all levels of constrained optimization. Murray (1969a) considered solving the IQP associated with his algorithm for NCP but found an EQP strategy more successful. He also considered a strategy of partially solving IQP. The same question arises in linearly constrained optimization. Brayton and Cullum (1977) report some results which indicate that for the case of minimization subject to bounds on the variables (the simplest possible constrained optimization problem), solving IQP is not in general more efficient than solving EQP.

The motivation for solving IQP is that it makes the fullest use of the information at $x^{(k)}$. Furthermore it simplifies the description of the algorithm and for problem MMP makes it straightforward to get a descent direction since positive constraints are not selected for an active set except in the process of solving IQP. Clearly one disadvantage of solving IQP is that it is more work than solving EQP. If $m \gg n$ and the function evaluations are not too expensive, then an algorithm which solves IQP may be extremely inefficient compared to one which solves EQP. However this is not the main objection to solving IQP. The main motivation for solving EQP rather than IQP is that the linear approximations to the constraints (for NCP and MMP) and the quadratic approximation to the Lagrangian function are unreliable away from the current point $x^{(k)}$. The process of solving IQP involves successively making decisions about which constraints to include in the active set at points which may be quite far from $x^{(k)}$, based on the approximations at $x^{(k)}$. Thus the final point at which this decision is made, the solution of IQP, may be the result of choosing an active set which has no meaning whatsoever. If $x^{(k)}$ is so close to x^* that the approximations are satisfactory then IQP may still have no advantage over EQP since they may well have the same solution.

Most of the differences between Han's method and ours result from the difference between IQP and EQP (he uses the multipliers from the old IQP and requires that the full Hessian $\overline{W}$ is positive definite, while we use λ_C to define $\overline{W}$ and require only that $Z^T W Z$ be positive definite). He discusses only quasi-Newton methods while we also consider a finite difference strategy. Finally, although his line search is used to obtain a reduction in F_M it is not designed specially for nondifferentiable functions as ours is.

Han (1978b) presents another algorithm for MMP which is related to the one discussed above. It is quite different, however, in that the line search takes place in the $(n+1)$-dimensional space. The line search obtains a reduction in the function

$$\theta(\alpha) = x_{n+1} + \alpha p_{n+1} + \sum_{i=1}^{m} \max\left(f_i(\bar{x} + \alpha \bar{p}) - (x_{n+1} + \alpha p_{n+1}), 0\right),$$

instead of $F_M(\bar{x} + \alpha \bar{p})$. The motivation given for this is that θ takes into account some inactive functions while F_M gives bias completely to the active functions. However, our view is that inactive functions should be considered in the line search only if they are likely to become active along the line, and this is exactly what our special line search to reduce the minimax function does.

The problem with reducing θ is that it relies too much on the value of p_{n+1} which gives only a linear approximation to the active functions at $x^{(k)}$. It is easy to construct examples where minimizing θ along the line results in a much smaller reduction of F_M than is possible. We feel that using θ instead of F_M in the line search is discarding one of the most useful tools available to solve MMP.

The analysis in Han (1978b) is concerned with the fact that the quadratic form of IQP is not positive definite in R^{n+1}. Our view, however, is that the only matrix whose positive definiteness should be a concern is the projected Hessian, and this is the same in R^n and R^{n+1} since $\bar{Z}^T\bar{W}\bar{Z}=Z^TWZ$. Recall that we expect the dimension of this matrix to be much smaller than that of W.

We now turn our attention to the algorithm of Conn (1979), which is more closely related to ours in a number of ways. Like us, Conn attempts to identify the active set and solves a related EQP at every iteration. However, unlike ours, his search direction does not include a component in the space spanned by Y unless there is reason to believe that $x^{(k)}$ is near a stationary point. Furthermore, he uses the Hessian of the Lagrangian function to give the quadratic form of the QP only if $x^{(k)}$ is thought to be near a stationary point. Otherwise he uses instead the Hessians of *one* of the active functions at each iteration. His reason for this is that the Lagrange multipliers may be highly inaccurate away from a stationary point. Although this is certainly true, our view is that using the Hessian of only one function is equivalent to using a multiplier estimate with only one component equal to one and the rest zero, and that this is no less arbitrary than using any other vector of nonnegative components which sum to one to define W. As we explained at the end of §4, our algorithm always uses such a vector to define W. Our approach eliminates any need to decide when to switch from one strategy to another, something which it is difficult to do since it is hard to tell how accurate multiplier estimates are. Furthermore, using different Hessians at different iterations makes a quasi-Newton approach difficult.

There are many other differences between Conn's algorithm and ours which follow because of the fact that his approach is via a nondifferentiable penalty function while ours is via a Lagrangian function. For example, he does not factorize the matrix $\hat{A}$ as we do, but instead factorizes a matrix M which is $-\hat{V}$ less one column $\hat{v}_j$ corresponding to the one function whose Hessian is to be computed. This matrix factorization is then updated to give a factorization of the matrix resulting from adding $\hat{v}_j$ to each column of M. It can be shown that this approach restricts $\bar{p}$ to the same null space as our algorithm. Multiplier estimates can also be computed by this approach but they will not be the same as either λ_L or λ_C since they give bias to function j. Conn shows how to take advantage of any explicitly linear functions in his algorithm.

In a way our algorithm treads the middle ground between Han's method and Conn's method. Han relies on the approximation at $x^{(k)}$ so completely that he solves the IQP. Conn distrusts the multiplier estimates and does not use them unless $x^{(k)}$ is near a stationary point. We believe that some multiplier estimates are better than no estimates at all, but we solve the QP which relies as little on the approximations as possible.

13. Convergence properties. In the limit our method becomes a projected Lagrangian method for NCP applied to the special case MMP, and so we can make use of the known asymptotic local convergence results for these methods. Robinson (1974) showed that the method of Wilson (1963) has a quadratic rate of local convergence if analytical Hessians of the objective function and constraints are used, provided that the functions are sufficiently smooth, $\overset{*}{Z}^T\overset{*}{W}\overset{*}{Z}$ is positive definite, $\overset{*}{A}$ has full rank, and $\overset{*}{\lambda}>0$. This last condition also ensures that ultimately solving the IQP and the EQP are the same, so the only difference between our method and Wilson's in the limit is the fact that we use the first-order multiplier estimates at the

new point (instead of the multipliers of the old QP) to define W. It is shown by Fletcher (1974) that using the first-order multiplier estimates does not inhibit the quadratic convergence of a projected Lagrangian method.

Using finite difference approximations to the Hessian or projected Hessian is well known to have essentially the same final rate of convergence (to machine precision) as using analytical Hessians. An R-superlinear rate of convergence for a projected Lagrangian method using a particular quasi-Newton approximation to the full matrix W has been shown by Powell (1977). All of the above convergence results assume that in a neighborhood of the solution, the steplength is always one.

14. Computational results. In this section we illustrate the usefulness of the algorithm by presenting some numerical results for the case where finite difference approximations to the Hessian of the Lagrangian function are used. Six problems are selected from the literature. The definitions of the functions $\{f_i(x)\}$ and the starting points used may be found in the individual references listed in Table 1, and all of the definitions appear together in Murray and Overton (1979b). Not all of the problems were originally posed as minimax problems (for example some of them defined least squares problems). The first four problems are treated as type l_∞P and the last two as type MMP.

The results are summarized in Table 3. The termination conditions required were that $\|\hat{c}\|_2 < 10^{-6}, \|Z^T e_{n+1}\|_2 < 10^{-6}$, $Z^T WZ$ numerically positive semidefinite and $\lambda_C \geqq 0$. The line search accuracy parameter η was set to 0.9 (see Murray and Overton (1979a) for the definition of this parameter). Several other choices of η were tried, but $\eta = 0.9$ was the most efficient, indicating as expected that a slack line search is desirable at least on these problems. The machine used was an IBM 370/168 in double precision, i.e., with 16 decimal digits of accuracy. The column headed NI reports the number of iterations required, which is also the number of times the Hessian was approximated. The column headed NF gives the number of function evaluations (not including gradient evaluations for the Hessian approximation).

The results demonstrate that at least on a limited set of test problems the algorithm fulfills some of its promise. Final quadratic convergence was observed in all cases. The algorithm has been tested on a wider set of problems and results obtained for a variety of choices of the optional parameters. It was clear from these more extensive results that more work needs to be done in developing the active set selection strategy. These results must therefore be regarded as preliminary.

15. Concluding remarks. It should be clear by now how our algorithm is related to the projected Lagrangian algorithms which have been proposed to solve NCP. Wilson (1963), Robinson (1974), Han (1977a) and Powell (1977) all solve successive inequality constrained QP's, so in that sense they are more closely related to the method of Han (1977b), (1978a) than to our method. Murray (1969a), (1969b), Wright (1976) and Murray and Wright (1978) solve successive equality-constrained QP's. However, their methods differ from the others and from ours in the sense that they do not attempt to step to the active constraint boundaries at every step but control how far outside or inside the feasible region the iterates stay by means of penalty and barrier parameters. This type of approach has proved to be very successful for solving NCP because it balances the reduction of the objective function with the reduction of the constraint violation in a satisfactory way. However, this approach is quite unnecessary for solving MMP since it is always trivial to obtain a feasible point for

TABLE 1

Reference	n	m	$n+1-\overset{*}{t}$	NI	NF	Solution Found
1. Bard (1970, p. 170)†	3	15	1	11	11	$F_\infty(\overset{*}{x})=0.77601$ $\overset{*}{x}=(0.17757,-0.94295,5.30796)^T$.
2. Kowalik and Osborne (1968, p. 104), example (v).	4	11	0	11	14	$F_\infty(\overset{*}{x})=0.0080844$ $\overset{*}{x}=(0.18463,0.10521,0.01196,0.11179)^T$.
3. Madsen (1975, p. 326), example 2.	2	3	1	13	19	$F_\infty(\overset{*}{x})=0.61643$ $\overset{*}{x}=(-0.45330,0.90659)^T$.
4. El-Attar et al. (1979, p. 81) example 2.	3	6	2	7	8	$F_\infty(\overset{*}{x})=3.59972$ $\overset{*}{x}=(0.32826,0.00000,0.13132)^T$
5. Charalambous & Bandler (1976) example 1.	2	3	1	6	6	$F_M(\overset{*}{x})=1.95222$ $\overset{*}{x}=(1.13904,0.89956)^T$
6. Charalambous and Conn‡ (1978, p. 182)	4	4	2	7	10	$F_M(\overset{*}{x})=-44.0000$ $\overset{*}{x}=(0.00000,1.00000,2.00000,-1.00000)$.

†This is a different local minimum from that found by Watson (1979).

‡This is the NCP problem of Rosen and Suzuki (1965) transformed to type MMP using a penalty parameter, which is always possible if the parameter is large enough.

EMP. To put it another way, reducing the minimax function in the line search always results in a step towards the constraint boundaries, although we do not usually wish to step exactly to the boundaries by doing an exact line search.

Linearly constrained minimax problems can be handled by the algorithm we have presented, since the constraints can be included in the QP at each iteration. However, nonlinear constraints cannot be handled by the algorithm for MMP in a straightforward way. As soon as nonlinear constraints are introduced the natural merit function is lost and the problem takes on the complexity of the general nonlinear programming problem NCP. Of course nonlinear constraints can still be handled by nonlinear programming methods, but it is important to recognize the increase in difficulty. Clearly the best approach would be one which takes advantage of the minimax structure and introduces an artificial merit function dealing with the genuine nonlinear constraints and not with those of EMP.

Acknowledgments. The authors would like to thank Philip E. Gill, Gene H. Golub and Margaret H. Wright for a number of helpful comments.

REFERENCES

D. H. ANDERSON and M. R. OSBORNE (1977), *Discrete, nonlinear approximations in polyhedral norms: a Levenberg-like algorithm*, Numer. Math., 28, pp. 157–170.

Y. BARD (1970), *Comparison of gradient methods for the solution of nonlinear parameter estimation problems*, SIAM J. Numer. Anal., 7, pp. 157–186.

M. BRANNIGAN (1978), *The strict Chebyshev solution of overdetermined systems of linear equations with rank deficient matrix*, Report, National Research Institute for Mathematical Sciences of the CSIR, Pretoria, South Africa.

R. K. BRAYTON and J. CULLUM (1977), *Optimization with the parameters constrained to a box*, Proc. IMACS Intl. Symp., Simulation Software and Numerical Methods for Stiff Differential Equations.

C. CHARALAMBOUS and J. W. BANDLER (1976), *Nonlinear minimax optimization as a sequence of least p-th optimization with finite values of p*, Internat. J. Systems Sci. 7, pp. 377–391.

C. CHARALAMBOUS and A. R. CONN (1978), *An efficient method to solve the minimax problem directly*, SIAM J. Numer. Anal., 15, pp. 162–187.

C. CHARALAMBOUS and O. MOHARRAM (1978), *A new approach to minimax optimization*, in Large Engineering Systems 2, G. C. Savage and P. H. Roe, eds., pp. 169–172.

_____(1979), *Quasi-Newton methods for minimax optimization*, Department of Systems Design Report 48-0-240179, University of Waterloo, Waterloo, Canada.

E. W. CHENEY (1966), *Introduction to Approximation Theory*, McGraw-Hill, New York.

A. R. CONN (1979), *An efficient second-order method to solve the (constrained) minimax problem*, Department of Combinatories and Optimization Report CORR-79-5, University of Waterloo, Waterloo, Canada.

L. CROMME (1978), *Strong uniqueness: A far reaching criterion for the convergence analysis of iterative procedures*, Numer. Math., 29, pp. 179–194.

V. F. DEMYANOV and V. N. MALOZEMOV (1974), *Introduction to Minimax*, John Wiley, New York and Toronto.

R. A. EL-ATTAR, M. VIDYASAGAR and S. R. K. DUTTA (1979), *An algorithm for L_1-norm minimization with application to nonlinear L_1 approximation*, SIAM J. Numer. Anal., 16, pp. 70–86.

A. V. FIACCO and G. P. MCCORMICK (1968), *Nonlinear Programming: Sequential Unconstrained Optimization Techniques*, John Wiley, New York.

R. FLETCHER (1974), *Methods related to Lagrangian functions*, in Numerical Methods for Constrained Optimization, P. E. Gill and W. Murray, eds., Academic Press, New York and London, pp. 219–240.

R. FLETCHER and T. L. FREEMAN (1977), *A modified Newton method for minimization*, JOTA 23, pp. 357–372.

P. E. GILL, G. H. GOLUB, W. MURRAY and M. A. SAUNDERS (1974), *Methods for modifying matrix factorizations*, Math. Comp., 28, pp. 505–535.

P. E. GILL and W. MURRAY (1974), *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Prog., 7, pp. 311–350.

_____(1977), *The computation of Lagrange multiplier estimates for constrained minimization*, National Physical Laboratory Report, NAC 77, Teddington, England.

_____(1979), *The computation of Lagrange multiplier estimates for constrained minimization*, Math. Prog., 17, pp. 32–60.

G. H. GOLUB (1965), *Numerical methods for solving least squares problems*, Numer. Math. 7, pp. 206–216.

J. HALD and K. MADSEN (1978), *A two-stage algorithm for minimax optimization*, Inst. for Num. Anal. Report NI-78-11, Tech. Univ. of Denmark.

S. P. HAN (1977a), *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., 22, pp. 297–309.

_____(1977b), *Variable metric methods for minimizing a class of nondifferentiable functions*, Department of Computer Science Report, Cornell University, Ithaca, NY.

_____(1978a), *Superlinear convergence of a minimax method*, Department of Computer Science Report, Cornell University, Ithaca, NY.

_____(1978b), *On the validity of a nonlinear programming method for solving minimax problems*, Math. Research Center Report 1891, University of Wisconsin, Madison.

R. HETTICH (1976), *A Newton method for nonlinear Chebyshev approximation*, in Approximation Theory, A. Dold and B. Eckmann, eds., Bonn.

A. S. HOUSEHOLDER (1964), *The Theory of Matrices in Numerical Analysis*, Ginn (Blaidsdell), Boston.

K. JITTORNTRUM and M. R. OSBORNE (1979), *Strong uniqueness and second-order convergence in nonlinear discrete approximation*, working paper, Computing Research Group, Australian National University, Canberra.

J. KOWALIK and M. R. OSBORNE (1968), *Methods for Unconstrained Optimization Problems*, Elsevier, New York.

K. MADSEN (1975), *An algorithm for the minimax solution of overdetermined systems of nonlinear equations*, J. Inst. Math. Applics. 16, pp. 321–328.

J. J. MORÉ and D. C. SORENSON (1979), *On the use of directions of negative curvature in a modified Newton method*, Math. Prog., 16, pp. 1–20.

W. MURRAY (1969a), *Constrained optimization*, Report MA 79, National Physical Laboratory, Teddington, England.

_____(1969b), *An algorithm for constrained optimization*, in Optimization, R. Fletcher, ed., Academic Press, London and New York, pp. 247–258.

W. MURRAY and M. L. OVERTON (1979a), *Steplength algorithms for minimizing a class of nondifferentiable functions*, Computing, 23, pp. 309–331.

_____(1979b), *Projected Lagrangian algorithms for nonlinear minimax optimization*, Systems Optimization Laboratory Report SOL 79-21, Stanford University, Stanford, CA.

W. MURRAY and M. H. WRIGHT (1978), *Projected Lagrangian methods based on the trajectories of penalty and barrier functions*, Systems Optimization Laboratory Report SOL-78-23, Stanford University, Stanford, CA.

M. L. OVERTON (1979), *Projected Lagrangian algorithms for nonlinear minimax and l_1-optimization*, Ph.D. thesis, Computer Science Department Report STAN-CS-79-752, Stanford University, Stanford, CA.

M. J. D. POWELL (1977), *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, Report DAMTP 77/NA3, University of Cambridge, Cambridge, England.

J. R. RICE (1969), *The Approximation of Functions*, Vol. II, Addison-Wesley, Reading, MA.

S. M. ROBINSON (1974), *Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms*, Math. Prog., 7, pp. 1–16.

J. B. ROSEN and S. SUZUKI (1965), *Construction of nonlinear programming test problems*, C.A.C.M. 8, p. 113.

G. W. STEWART (1973), *Introduction to Matrix Computations*, Academic Press, New York and London.

W. H. SWANN (1972), *Direct search methods*, in Numerical Methods for Unconstrained Optimization, W. Murray, ed., Academic Press, New York and London.

G. A. WATSON (1979), *The minimax solution of an overdetermined system of nonlinear equations*, J. Inst. Math. Applics., 23, pp. 167–180.

A. P. WIERZBICKI (1978), *Lagrangian functions and nondifferentiable optimization*, Report WP-78-63, Intl. Inst. Appl. Sys. Anal., Laxenburg, Austria.

R. B. WILSON (1963), *A simplicial algorithm for concave programming*, Ph.D. thesis, Graduate School of Business Administration, Harvard University, Cambridge, MA.

M. H. WRIGHT (1976), *Numerical methods for nonlinearly constrained optimization*, Ph.D. thesis, Computer Science Department Rept. STAN-CS-76-566, Stanford University, Stanford, CA.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0006 $01.00/0

SOLVING THE HELMHOLTZ EQUATION FOR EXTERIOR PROBLEMS WITH VARIABLE INDEX OF REFRACTION: I*

GREGORY A. KRIEGSMANN† AND CATHLEEN S. MORAWETZ‡

Dedicated to R. D. Richtmyer on the occasion of his 70th birthday

Abstract. A new technique for numerically solving the reduced wave equation on exterior domains is presented. The method is basically a relaxation scheme which exploits the limiting amplitude principle. A modified boundary condition at "infinity" is also given. The technique is tested on several model problems: the scattering of a plane wave off a metal cylinder, a metal strip, a Helmholtz resonator, an inhomogeneous cylinder (lens), and a nonlinear plasma column. The results are in good qualitative agreement with previously calculated values. In particular, the numerical solutions exhibit the correct refractive and diffractive effects at moderate frequencies.

Key words. exterior domains, difference equations, Helmholtz equations, Helmholtz resonator, inhomogeneous media, lens, numerical solutions, numerical boundary condition at infinity, relaxation scheme, scattering

1. Introduction. It is well known for dissipative linear ordinary differential equations with a forcing term of period λ that the transients die out and the solution tends, as $t\to\infty$, to solutions of period λ. The same is true of many hyperbolic equations. In particular, solutions of the wave equation in infinite domains with appropriate boundary conditions and with the forcing term $fe^{i\omega t}$ tend, for large times, to solutions of the Helmholtz equation $\Delta u+\omega^2 u=f$. This is known as the limiting amplitude principle [10].

The purpose of this paper is to show that a varied form of a limiting amplitude principle can be used to solve numerically, in a short time and for a rather wide variety of geometries, the Helmholtz equation in the exterior of an obstacle with a variable index of refraction. Furthermore, this can be done at such high frequencies that a great deal of geometric optical behavior can be confirmed even on a relatively coarse mesh.

The Helmholtz equation with variable index of refraction is

$$\Delta u+\omega^2 nu=0, \tag{1.0}$$

where the index of refraction $\sqrt{n}$ is a function of the space variables $(\mathbf{x})$. We also consider situations where $n=n(\mathbf{x},|u|)$, which corresponds to certain models of laser beam propagation. At large distances we assume $n\to$ constant which may be scaled to 1. Then the parameter ω has the dimension of $[\text{Length}]^{-1}$ and the appropriate dimensionless constant is ωa where a is a characteristic length of the scatterer. It may range from zero to infinity.

*Received by the editors August 11, 1980. These results were obtained at the Courant Institute of Mathematical Sciences, New York University, with support from the U.S. Air Force Office of Scientific Research under contract F49620-79-C-0193, and the U.S. Office of Naval Research under contract N00014-76-C-0439. The U.S. Government's right to retain a nonexclusive royalty-free license in and to the copyright covering this paper, for governmental purposes, is acknowledged.

†Department of Engineering Sciences and Applied Mathematics, Northwestern University, Evanston, Illinois 60201.

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012.

The field u is given as a plane wave $e^{i\omega x}$ and its scattered field u_s,

$$u = e^{i\omega x} + u_s. \tag{1.1}$$

However, the source could equally well be located at a finite point in the plane. The scattered field satisfies the outward radiation condition

$$\frac{\partial u_s}{\partial r} - i\omega u_s + \frac{u_s}{2r} \to 0, \tag{1.2}$$

as $r = |x| \to \infty$.

This problem may be studied using geometrical optics for high frequency and expansion in ω for low frequency. Our original objective was to deal with the range of frequency where neither of these approximations is good. We found, in addition, however, that many of the features of geometrical optics emerge from the computations at moderate frequencies. Primarily, though, we are concerned with perturbations in the index of refraction or with disturbing objects where the characteristic length is not large compared to the wavelength.

A wave at frequency ω has the wavelength $2\pi/\omega$ and theoretically, for geometrical optics, one needs $2\pi/\omega \ll a$. Our observations confirm that in many applications $\omega a = 5$ or 10 displays the significant features of geometrical optics and diffraction theory.

In this paper, we restrict ourselves to two-dimensional examples. In a second paper, we plan to demonstrate results in axisymmetric and possibly three-dimensional geometries. A particular advantage in 2D is that every simply-connected object can be studied by noting that it can be mapped conformally onto the exterior of a circle and the transformation induces a new Helmholtz equation for the solution with a new index of refraction. In fact, however, we have also computed objects that are not simply connected but have rectangular boundaries in polar coordinates.

The principal results of interest are:

(i) Confirmation of the scattering cross-section for a cylinder of radius one with Dirichlet boundary condition. Results are compared with those presented by Bowman, Senior and Uslenghi [1]. The error is less than 1(10)% at the frequency $\omega = 5(10)$. The computations also show that the position of the shadow edge, as given by geometrical optics, is within one or two mesh points.

(ii) The diffraction by an infinite strip of halfwidth 2 at various angles to an incident plane wave. There is good correlation with [1] particularly at low frequencies and at an angle of 45° and with geometrical optics [1].

(iii) The computations of the field including the cross-section of a Helmholtz resonator of radius 1 with various apertures and for plane waves at various angles of incidence.

(iv) The refraction of a plane wave by a lens of either constant or variable index of refraction. In particular, we include lenses that produce focusing and caustics such as the Luneberg lens.

(v) The refraction by a model of an overdense plasma.

The method of computation is a modification of the relaxation method used in [2] and is described in § 2. To reduce the number of mesh points involved, a modified radiation condition corresponding to (1.2) is imposed at a finite radius (see § 3). The difference scheme and how to deal with the artificial singularity at the origin created by using polar coordinates is described in § 4. In § 5, we describe in detail the particular examples computed. In each case we give the running times on the CDC

6600 required to achieve convergence. The effects of applying the modified radiation condition at different distances are also discussed.

In § 6, we discuss the limitations of this kind of iterative method and other possible applications.

2. The iteration method. We consider a time-dependent equation

$$(Q\tilde{U})_t = \Delta\tilde{U} + n\omega^2\tilde{U}, \tag{2.1}$$

where Q is a first-order operator in the space variables, $Q = 2\mathbf{a}\cdot\nabla + b$. The coefficient vector $\mathbf{a}$ and the scalar b are chosen so that the solutions of (2.1) of the form plane wave or source plus outgoing scattered wave will approach a time-independent state. This will be the desired solution.

This method was used in [2]. However, (2.1) was used directly, i.e., in characteristic form. Data were given on a characteristic surface. The convergence was therefore very slow because a very small time step was required for stability with the space differences used.

In the present method, we transform (2.1) to a Cauchy problem by a change of variables given by

$$dt' = dt + \mathbf{a}\cdot d\mathbf{x}, \qquad d\mathbf{x}' = d\mathbf{x}. \tag{2.2}$$

We obtain in the new variables

$$|\mathbf{a}|^2\tilde{U}_{tt} = \Delta\tilde{U} + \tilde{U}_t(\nabla\cdot\mathbf{a} - b) + n\omega^2\tilde{U}, \tag{2.3}$$

where we have dropped the primes. We now solve an initial value problem which by our choice of $\mathbf{a}$ and b will converge to the steady state. For example, the convergence has been improved by more than an order of magnitude in the case

$$\mathbf{a} = \frac{\mathbf{x}}{|\mathbf{x}|}, \qquad b = -2i\omega + \nabla\cdot\mathbf{a}. \tag{2.4}$$

In two dimensions, with $\mathbf{a}$, b given by (2.4), we introduce in (2.3) the change of variable

$$\tilde{U} = e^{i\omega x} + \frac{\tilde{W}e^{i\omega t}}{\sqrt{r}}, \tag{2.5}$$

with $|\mathbf{x}| = r$. Equation (2.3) reduces in polar coordinates to

$$\tilde{W}_{tt} = \tilde{W}_{rr} + r^{-2}\left[\tilde{W}_{\theta\theta} + \tfrac{1}{4}\tilde{W}\right] + \omega^2(n-1)\tilde{W} + \omega^2\sqrt{r}\,(n-1)e^{i\omega' x - t}, \tag{2.6}$$

which for $n = 1$ is the wave equation. However, in general, it has fixed characteristics that are independent of n. Thus, if $n = 1$, we are using the standard limiting amplitude principle, but since in some of our examples n changes dramatically it is a great advantage that the characteristics are fixed. For $n \neq 1$, we are solving a wave equation with potential.

From [2] we have the growth restriction to obtain decay,

$$\frac{\partial}{\partial r}\left[r(n-1)\right] > 0. \tag{2.7}$$

If this condition is not satisfied, we are dealing with a potential which has bound states that give rise to exponentially growing solutions of (2.6). This showed up numerically in a very dramatic fashion when we tried lenses with $n > 1$. The inequality (2.7) was proved by J. Weidmann [4] as a necessary condition to prevent these growing modes.

In order to handle this situation, we note that instead of (2.6) we could use

$$n_1\tilde{W}_{tt} = \tilde{W}_{rr} + r^{-2}\left[\tilde{W}_{\theta\theta} + \tfrac{1}{4}\tilde{W}\right] + (n_2 - 1)\omega^2\tilde{W} + \omega^2\sqrt{r}\,(n-1)e^{i\omega(x-t)}, \tag{2.8}$$

where $n_1 + n_2 = n + 1$. If the limiting amplitude principle holds, i.e., the potential $(n_2 - 1)\omega^2$ does not give rise to growing modes and $n_1 > 0$, then the solutions of (2.8) also approach a time-harmonic steady state and the space-dependent coefficient satisfies the desired Helmholtz equation.

If we make $n_1 > 1$ in such a way that $[r(n_2 - 1)]_r \geqq 0$, then (2.8) can be used for the iteration. This worked for cases with $n_1 = n \geqq 1$, $n_2 = 1$ and $n \leqq 4$ inside a circle. Note that the Courant-Friedrichs-Lewy condition for the time step continues to be satisfied over $n_1 > 1$ if it holds for $n_1 = 1$. On the other hand, if there are trapped rays produced by the artificial index n_1, then the approach to steady state may be exceedingly slow as the higher frequencies have poorer exponential decay.

3. Improving the radiation condition and finding the scattering cross-section. Returning to the original radiation condition (1.2), we note that if a solution satisfies the radiation condition, then by using the fundamental solution representation for the reduced wave equation, we have with $u_s = W_r^{-(N-1)/2}e^{i\omega r}$ the expansion

$$W = \sum_{j=0}^{\infty} A_j r^{-j}, \tag{3.1}$$

in any number of dimensions. Here A_j is a function of ω and the angular variables. The differential equation for W at large distances is

$$2iW_r = -W_{rr} - \frac{1}{r^2}LW, \tag{3.2}$$

where L is a differential operator acting on the angular variables (the Laplace-Beltrami operator). We have rescaled r so that $\omega = 1$. On substituting the expansion for W, we find the recursion formula

$$A_{n+1} = \tilde{P}(n+1, L)A_n, \qquad n \geqq 0, \tag{3.3}$$

where $\tilde{P}(n+1, L) = -\frac{1}{2}i(n+1)^{-1}\{L + n(n+1)\}$. Note that $\tilde{P}(1, L) = -(i/2)L$. Next we set

$$P(n, L) = \prod_{j=1}^{n} \tilde{P}(j, L), \qquad P(0, L) = 1,$$

and obtain

$$W = \sum_{0}^{\infty} P(n, L)r^{-n}A_0, \qquad W_r = -\sum_{0}^{\infty} P(n, L)nr^{-n-1}A_0, \tag{3.4}$$

which we invert to find

$$A_0 = -\left(\sum_{1}^{\infty} nP(n, L)r^{-n-1}\right)^{-1} W_r$$

and thus

$$W_{rr} = -\left(\sum_{1}^{\infty} n(n+1)P(n, L)r^{-n-2}\right)\left(\sum_{1}^{\infty} nP(n, L)r^{-n-1}\right)^{-1} W_r. \tag{3.5}$$

The boundary condition on W is then found by substituting in the differential

equation (3.2),

$$2iW_r = -\frac{1}{r^2}LW + \frac{1}{r}\left[2P(1,L) + \sum_{n=2}^{\infty} n(n+1)P(n,L)r^{-n+1}\right] \cdot \left[P(1,L) + \sum_{n=2}^{\infty} nP(n,L)r^{-n+1}\right]^{-1} W_r, \tag{3.6}$$

and solving for W_r as a function of LW.

We simply expand the formula in powers of r, but it may be better to use the approach of Engquist and Majda [5] and use other representations of the operators. What is needed is a good representation of $\Sigma_{n=2}^{\infty} nP(n,L)r^{-n+1}$. The first approximation is $W_r = 0$, and from (3.6) the next approximation is

$$\left(2i - \frac{2}{r}\right)W_r = -\frac{LW}{r^2} + O(r^{-4}). \tag{3.7}$$

Note that as $\omega \to \infty$, $LW \to \infty$ in general, and there are problems of convergence.

In transforming to the Cauchy problem, the derivative $W_r \to \tilde{W}_r + \tilde{W}_t$ and the right-hand side is unchanged. Thus, the radiation condition which we used was

$$\tilde{W}_r + \tilde{W}_t = \frac{L\tilde{W}}{2\omega r^2(i - 1/\omega r)}, \tag{3.8}$$

where, for the two-dimensional problem, $L = \partial^2/\partial\theta^2 + \frac{1}{4}$.

To obtain the far field, that is the scattering cross-section A_0, we invert (3.4) and obtain

$$A_0 = \left[1 - \frac{P(1,L)}{\omega r}\right]W, \tag{3.9}$$

which is accurate to $O(1/\omega^2 r^2)$; see also [3].

4. The difference scheme. To solve the time-dependent differential equation (2.7) imposing the far field condition (3.9), we have used a standard backward difference scheme for the initial value problem of a second-order hyperbolic equation with second-order accuracy. Let the solution to the difference scheme be $\tilde{W}(j, m, n)$, where (j, m, n) are evaluated on a grid

$$r = j\Delta r + r_0, \quad \theta = m\Delta\theta, \quad t = n\Delta t, \tag{4.1}$$

with

$$0 \leqq j \leqq N, \quad 0 \leqq m \leqq M.$$

Then for an interior point,

$$W(j, m, n+1) = T(j \pm 1, j, m \pm 1, m, n, n-1) \tag{4.2}$$

is determined from the values of W at $(j \pm 1, m \pm 1, n), (j, m, n), (j, m, n-1)$. On the outer boundary $r = N\Delta r + r_0$ we use the differenced form of (3.8),

$$\frac{1}{2\Delta t}\{\tilde{W}(N, m, n+1) - \tilde{W}(N, m, n-1)\} + \frac{1}{2\Delta r}\{\tilde{W}(N+1, m, n) - \tilde{W}(N-1, m, n)\} = S(M, m, n), \tag{4.3}$$

where $S(N, m, n)$ denotes the difference approximation for the right-hand side, centered differences being used in L. The value of W at $j = N+1$, m, n has to be

eliminated by using the difference equation (4.2), and thus one obtains

$$\tilde{W}(N, m, n+1) = B(N, m, m\pm 1, n, n-1). \tag{4.4}$$

It still remains to apply the Dirichlet condition if there is a disturbing object, or to deal with the singularity at the origin which is artificially created by the use of polar coordinates.

(a) To apply the Dirichlet condition ($u=0$) we simply use the given values from $\tilde{W} = -\sqrt{r}\exp[i\omega(x-t)]$ at the mesh point $j=0$ for a circle or at any other boundary point. To simplify the computation, we have only boundaries consisting of sections $\theta=$ constant or $r=$ constant.

(b) To deal with the origin, we note two difficulties. First the Courant-Friedrichs-Lewy condition limits the smallest radius; i.e., we must have $r\Delta\theta > \Delta t$ for all points in the computation. Secondly, there is a singularity from the term $\frac{1}{4}r^{-2}\tilde{W}$. However, we do have $\tilde{W}\sim\sqrt{r}$. Our approach is to keep doubling the mesh size $\Delta\theta$ as $r\to 0$. Thus, in these problems for $r\leqq r_0$, we use the mesh

$$r=j'\Delta r, \quad \theta = ms(j'), \quad t=n\Delta t, \tag{4.5}$$

where $s(j')$ is chosen so that $r\Delta\theta = j'\Delta r s(j') > \Delta t$ and so that for each j' where the mesh size $s(j')$ changes, the θ-mesh is half as dense as before. The behavior of the equation at the origin itself has to be taken into account. Various integral forms of the equation could be used, but it is sufficiently effective to use the equation for $\tilde{U}$ at $r=0$ using for $\Delta\tilde{U}$ the values of $\Phi = \tilde{W}/\sqrt{r}$ at $\theta = 0, \pm\pi/2, \pi$ and $r=2\Delta r$. By the standard difference formula along with $\Phi(n)$ for the value of $\tilde{U}$ at $r=0$ and time $t=n\Delta t$, this yields, for the symmetric case,

$$\frac{(\Phi(n+1)-2\Phi(n)+\Phi(n-1))}{(\Delta t)^2} = \frac{\tilde{W}(2, m_1, n)+2\tilde{W}(2, m_2, n)+\tilde{W}(2, m_3, n)-4\Phi(n)}{(2\Delta r)^{3/2}} + \omega^2 n(0)\Phi(n) + \omega^2(n(0)-1)e^{-\omega n\Delta t}.$$

Here m_1, m_2, m_3 correspond to $\theta = 0, \pi/2, \pi$. Note $n(0)$ refers to the value of the index of refraction at the origin. In the nonsymmetric case, there is a similar formula.

Finally we obtain $\tilde{W}(1, m, n+1)$ by interpolating the values of $\Phi^{(n+1)}$ and $\tilde{W}(2, m, n+1)/\sqrt{2\Delta r}$ to obtain $\tilde{W}(1, m, n+1)/\sqrt{\Delta r}$. The error is $O(\Delta r^2)$ but turns out to be larger than in the rest of the computation.

In closing this section, we shall describe our numerical method for determining when $\tilde{W}$ has reached its time harmonic steady state. Recall that the scattered field, u_s, is given by

$$u_s = \frac{[\tilde{W}e^{i\omega t}]}{\sqrt{r}}.$$

The bracketed term must approach $We^{i\omega r}$ as $t\to\infty$. Thus, for large time, $|\tilde{W}|$ becomes independent of t. We terminate our computations when

$$\max_{\substack{0\leqq i\leqq N \\ 0\leqq j\leqq M}} ||\tilde{W}(n+1, i, j)| - |\tilde{W}(n, i, j)|| < \varepsilon, \tag{4.6}$$

for some prescribed $\varepsilon > 0$.

5. Results for model problems.

(i) *The metal cylinder.* This is a benchmark problem to test the method. Solutions have been determined by separation of variables, integral equations and geometrical optics (see [1] for a fairly comprehensive bibliography). In [1], the "scaled cross-section" $(\sqrt{\pi\omega/2}\,)S(\theta)$ is presented graphically for various frequencies. Here, $S(\theta)$ is the amplitude of the outgoing cylindrical wave and θ is the polar angle, with $\theta=\pi$ the direction of the incident plane wave. These results were carefully converted to tabular form for the case $\omega a=5$. They are shown in column C of Table 1, while the results of our present calculation are listed in column B. The agreement is excellent. The relevant parameters used in the computation are $\omega a=5$, $\Delta r=0.1$, $\Delta\theta=\pi/40$, $\Delta t=0.05$, and $\varepsilon=0.01$.

For the sake of comparison, the results of [2] are listed in Column A. The gain over our old method is in the running time, which is now approximately 1 minute on the CDC 6600. The old calculations took roughly 15 minutes on the same machine. Both methods used the same amount of core, 138K.

The effect of applying the modified radiation condition (3.8) at various distances to gain in mesh refinement was done by increasing the radius of the cylinder and applying (3.8) at a fixed radius. The relevant parameter is ωa, where a is the radius of the object [see Table 1, columns D–F]. This does not, of course, refine the θ-mesh.

Since our numerical method gives the total field at each grid point, the cross-sections given in Table 1 represent a small fraction of the generated information. Instead of just listing these numbers, an alternate method was devised to visually convey the results. First, the polar output was converted into a rectangular grid of numbers using straightforward interpolation. (This unfortunately introduces errors which tend to smear out the optical features that will be described shortly.) Then seven weights of shade were used, with the darkest color corresponding to the smallest total field while the lightest gives the largest field. The amplitude ranges are $(0,2)$, $(2,6)$, $(6,8)$, $(8,10)$, $(10,14)$, $(14,17)$ and $(17,\infty)$. This process gives the interference pattern shown in Fig. 1 when $\omega a=10$. The light regions correspond to constructive interference, the dark ones to destructive. Since the incident wave enters from $x=-\infty(\theta=\pi$ or from the left of the figure), the total field is symmetric about the x-axis and only half the pattern is shown in Fig. 1. The dark semicircle is the metal cylinder. The wave patterns are readily seen in this picture. Moreover, the shadow cast by the cylinder is quite apparent. The width of the transition region which connects the deep shadow and the illuminated portions of the plane is exaggerated by the interpolation process mentioned above.

(ii) *Infinite strip.* (*Dirichlet case*). The diffraction by an infinite strip of half-width 2, $\{x=0,|y|\leqq 2\}$ was computed at the frequency 5, $\omega a=10$ at the angles of attack $\alpha=90°$ (head-on), 45° and 90° (on edge). Here α is the angle between the incident plane wave and the positive x-axis. The wave again is approaching from $\theta=\pi$. The mesh size is $\pi/20$ in θ and 0.1 in r. A variable mesh was used near the origin.

Since the interference pattern in the x, y coordinates is smeared by interpolation in this and in other cases with sharp boundaries, it is omitted here. However, the results for $\alpha=0$ are presented in polar form in Fig. 2. The numbers printed at the mesh points are ten times the total field. Thus, the dark regions now correspond to constructive interference, the light to destructive. In this picture, one sees the shadow at 0, 2π, the standing wave in the illuminated region along $\theta=\pi$, the singular corners at $\theta=\pi/2$, $3\pi/2$ and the shadow boundaries on $y=\pm 2=r\sin\theta$.

TABLE 1

Columns A–F *present the cross-section* $(\pi\omega/2)^{1/2}S(\theta)$, *for a metal cylinder with* $|u_s| \sim S(\theta)/\sqrt{r}$, $\Delta t = 0.05$, *and* $\Delta\theta = \pi/40$. *Column* A *contains the results given in* [2], B *presents a tabulated version of the data given graphically in* [1], *and columns* C–F *contain the results of our new calculations.*

$\theta(^\circ)$	A $\omega a = 5$	B $\omega a = 5$	C $\omega a = 5$	D $\omega = 10, a = 1$	E $\omega = 5, a = 2$	F $\omega = 5, a = 4$
0	5.67	6.05	6.05	11.33	10.84	12.49
9	5.30	5.25	5.29	6.63	7.20	10.15
18	3.78	3.30	3.34	2.11	3.72	11.15
27	2.10	1.75	1.70	3.21	3.87	13.50
36	1.81	1.75	1.77	2.50	3.55	14.71
45	2.00	1.95	1.99	2.61	3.41	15.04
54	1.89	1.75	1.81	2.54	3.32	14.97
63	1.74	1.68	1.73	2.55	3.23	14.74
72	1.79	1.75	1.80	2.56	3.18	14.41
81	1.86	1.80	1.83	2.56	3.10	13.91
90	1.86	1.82	1.82	2.56	3.04	13.12
108	1.93	1.88	1.86	2.57	2.88	10.60
126	1.97	1.92	1.89	2.55	2.79	7.51
144	1.99	1.93	1.91	2.55	2.76	5.02
162	1.99	1.94	1.93	2.59	2.75	3.97
180	2.00	1.95	1.96	2.55	2.76	3.88

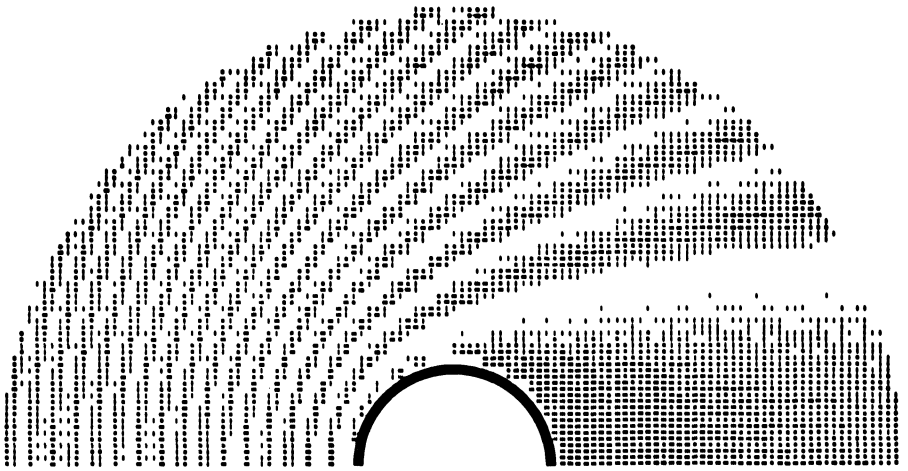


FIG 1. *The interference pattern produced by the total field's amplitude,* $|U|$, *for a metal cylinder. The light regions correspond to constructive interference, the dark to destructive. The dark semicircle is the cylinder. The relevant parameters are* $\omega a = 10$, $\Delta r = 0.1$, $\Delta\theta = \pi/40$, *and* $\Delta t = 0.05$.

To save space, we have not included the polar output for the cases $\alpha=45°$ and 90°. Rather a few words describing these results will be offered instead. A turn through $45°(=\alpha)$ destroys the symmetry and slightly distorts the waves in the illuminated region. The shadow is shifted 45°. When the wave attacks the strip on edge ($\alpha=90°$) there is no shadow. However, in the forward scattered direction the total field is cut in half as it should be; see [6]. The field is again symmetric.

The cross-sections are presented in Table 2. They check well at 45° and qualitatively at 0° with those given in [1]. The deviation between our results and those given by [1] is probably due to the fact that the derivatives of the field become singular at the strip's edges. The coarseness of our mesh masks this problem.

The running time and core requirement for this problem were roughly the same as those needed for the metal cylinder problem.

(iii) *The Helmholtz resonator.* A cylindrical Helmholtz resonator was placed in a plane wave at different angles of attack. The resonator is an infinitely thin cylinder of radius two with a strip aperture centered on the negative x-axis. The angle β which subtends the aperture was taken at different sizes. When $\beta=0$ the resonator is a metal cylinder which was used for comparison. (The frequency was fixed at five.)

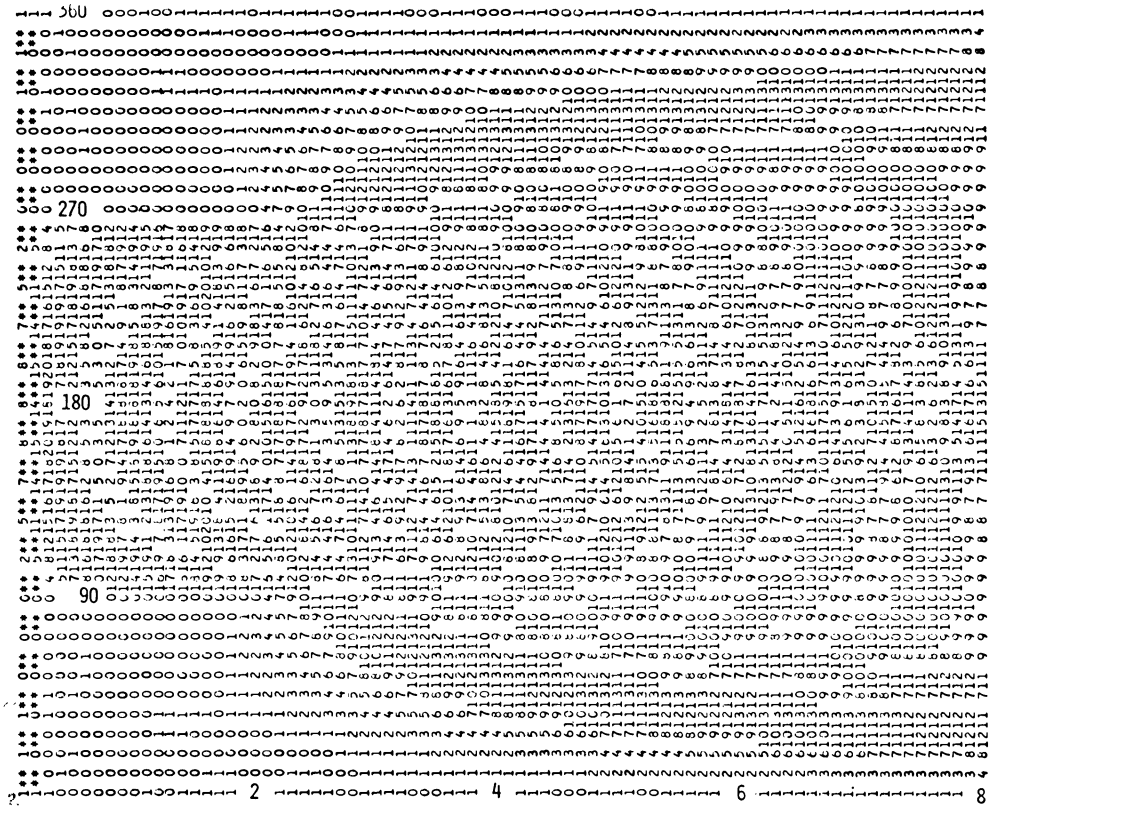


FIG 2. *The total field's amplitude, $|U|$, in polar coordinates for a metal strip with the incident wave normalized to* 10. *The incident wave strikes the strip head on ($\alpha=0$). The set of points $\{r,\theta)|\theta=\pi/2,3\pi/2,0\leqq r\leqq 2\}$ represents the strip in polar coordinates. Since the numbers printed are the actual field values, the dark regions now correspond to constructive interference, the light to destructive. The relevant parameters are the same as in Fig.* 1 *except* $\Delta\theta=\pi/20$.

TABLE 2

Columns A–D *present the cross-section* $(\pi\omega/2)^{1/2}S(\theta)$, *for a metal strip with* $|u_s| \sim S(\theta)/\sqrt{r}$, $\Delta t = 0.05$, $\Delta r = 0.1$, *and* $\Delta\theta = \pi/20$. *Columns* A *and* D *present tabulated versions of data given graphically in* [1]. *Columns* B *and* C *contain results of our new calculations. The parameter* α *is the angle between the incident plane wave and the normal to the strip;* $2a$ *L is the length of the strip.*

$\theta(°)$	A $\alpha=45°, \omega a=10$	B $\alpha=45°, \omega a=10$	C $\alpha=0°, \omega a=10$	D $\alpha=0°, \omega a=10$
−90	.89	.41	.32	.00
−81	.89	.85	.96	.00
−72	.89	.19	.09	.00
−63	.89	1.31	1.0	.83
−54	.89	1.72	1.9	1.8
−45	1.20	1.62	2.6	1.5
−36	.90	1.60	2.8	.9
−27	1.20	1.64	3.3	2.6
−18	1.20	1.52	3.6	0.0
−9	1.20	1.49	6.2	7.0
0	1.50	1.88	8.1	9.9
9	1.66	1.71	6.2	7.0
18	1.80	2.08	3.6	0.0
27	2.10	2.40	3.3	2.6
36	5.96	5.40	2.8	.9
45	7.20	7.30	2.6	1.5
54	5.37	5.61	1.9	1.8
63	2.98	2.50	1.0	.83
72	1.49	1.30	.09	.00
81	1.05	1.05	.96	.00
90	.70	.68	.32	0.0

cross-sections are shown in Table 3, where α is again the angle between the incident plane wave and the positive x-axis. The other relevant parameters are $\Delta t = 0.05$, $\Delta r = 0.1$ and $\Delta\theta = \pi/20$. There are no data for comparison. In iterating on the closed resonator, the method generated some eigenmodes in the interior. These exist as slowly damped modes as the aperture opens.

At an angle of attack of $\alpha = 0°$ and an aperture of 36°, the exterior field is very similar to that generated by a metal cylinder. The interior field contains a considerable amount of energy. The amplitudes focus on the axis in two places. The strongest (maximum amplitude 5) at $x = -0.5$ is a portion of caustic caused by the second reflections off the interior. The weaker focus (maximum amplitude 3) at $x = +0.5$ probably corresponds to the peak of a nephroid-like caustic formed by the first reflections. The difference in strength is probably due to constructive interference.

When the wave strikes the resonator from directly behind (angle of attack = 180°), it acts like a metal cylinder. The only energy inside is diffractive and weak.

At an aperture of 180° and an angle of attack of 0° there should be a nephroid-like caustic, but the edge diffraction obliterates this feature. There is marked focusing as expected, for $-2 \leqq x \leqq -1$. The shadow is not so sharp as with a metal cylinder. The aperture edges act like slits and smear the geometrical effects.

The core requirements for these problems were the same as those needed for the various metal cylinders. However, the running time depends upon the aperture size

TABLE 3

Columns A–C *present the cross-section* $(\pi\omega/2)^{1/2}S(\theta)$, *for various Helmholtz resonators with* $|u_s| \sim S(\theta)/\sqrt{r}$, $\Delta t = 0.05$, $\Delta\theta = \pi/20$, *and* $\Delta r = 0.1$. *The parameter* α *is the angle between the incident plane wave and the positive x-axis while* β *is the aperture angle of the resonator. The parameter* $\omega a = 10$, *where* a = *radius of the resonator.*

$\theta(°)$	A $\beta = 36°, \alpha = 0$	B $\beta = 36°, \alpha = 180°$	C $\beta = 180°, \alpha = 0°$
0	8.86	2.70	8.55
9	6.81	2.71	6.83
18	4.10	2.73	4.11
27	4.11	2.75	4.01
36	3.99	2.79	3.73
45	4.06	2.85	3.39
54	4.01	2.96	2.58
63	4.01	3.11	1.38
72	3.97	3.30	.84
81	3.86	3.50	1.39
90	3.71	3.70	1.09
99	3.53	3.85	1.86
108	3.42	3.99	2.15
117	3.50	3.95	3.12
126	3.72	4.15	3.73
135	3.83	3.83	2.85
144	3.66	4.25	3.55
153	3.28	3.87	5.55
162	2.48	3.77	3.78
171	1.10	7.60	2.99
180	1.36	8.48	5.32

which determines the complex eigenfrequencies of the resonator. Several eigenfrequencies for various resonators are computed numerically and presented in [7]. For an aperture of 36° and 800 iterations, we could satisfy (4.6) only for $\varepsilon \geqq 0.1$. However, when the aperture was 180° we could satisfy (4.6) with $\varepsilon = 0.05$ at 400 iterations. These two numerical examples show how the iteration scheme depends upon the eigenfrequency with the smallest imaginary part. In the first case, the imaginary part is roughly 0.015 while in the second case, it is about 0.3. If we assume that the transients decay like $\exp(-0.015t)$ in the first situation, then $t \sim 200$ would make this factor $O(1/100)$. Thus, for $\Delta t = 0.05$ (the number used in our program) we would need in the neighborhood of 5000 iterations to obtain convergence. The same crude argument shows that about 310 iterations are needed when the aperture is 180°.

All of these problems could be solved with the addition of a variable index of refraction depending on all variables including the amplitude of the total field provided the focusing effects are weak.

(iv) *Lens with variable index of refraction.* For a lens of constant index of refraction less than 1, we used (2.6), i.e., (2.8) with $n_1 = 1$, $n_2 = n$. There are no particular difficulties and the fields are qualitatively correct. The basic lens was of radius 3 and the frequency $\omega \leqq 5$ so that the relevant parameter $\omega a = 15$. For constant n less than 1 in the lens there is rapid convergence and typical lens patterns emerge. In Fig. 3, the interference pattern (in the x,y-plane) is shown for a lens with $a = 1$, $\omega = 5$,

and $n=0.4$. For $1 \leqq n \leqq 2$, there are focusing effects; e.g., when $n=2$ the maximum amplitude is 2.6 and occurs on the axis $\theta=0$. The full limiting amplitude principle with $n_1=n$ and $n_2=1$ works well for these cases. However, $n_1=n$ cannot be taken too large. A rescaling of time in (2.8) would introduce a large effective frequency into the exponential term. This would generate a large truncation error for a fixed Δt and cause numerical instabilities. Numerical experiments confirmed this observation for $n_1=n \geqq 8$ and $\Delta t=0.05$. On the other hand, taking $n_1=1$, $n_2=n$ brings immediate instability into the computation.

More interesting effects occur if $n=n(r)$ or $n=n(r,\theta)$. For example, geometrical optics predicts [8] that for $n=r^2/9$, $r \leqq 3$, $n=1$, $r \geqq 3$ that there will be focusing at $(3,\pi/2)$. At $\omega=5$, we obtained an amplification of 2.5 in the total field at that point. Also there is a geometrical shadow on $\theta=\pi/2$. This shadow (for a finite ω) is not sharp because a single ray on $\theta=0$, $\theta=\pi$ passes through it.

A second case is a Luneberg lens with $n=2-r^2/9$ for $r \leqq 3$, $n=1$ for $r \geqq 3$ that focused at (3,0) with a magnitude of 3. This point is a focus for the geometrical optics rays.

The general case $n=n(r,\theta)$ was tried with $n=(r^2-r_0^2)/(9-r_0^2)$ for $r \leqq 3$, $n=1$ for $r \geqq 3$, where $r_0=2-0.5\cos^2(\theta/2)$. The particular effects are not of interest. Obviously the oscillations in n cannot be too large without destroying accuracy. What is important is that this problem cannot be solved by separation of variables. Geometrical optics is complicated and an integral method would involve inverting a kernel in four variables. No difficulties are encountered here. There is some increase in memory since n depends on two variables. The running time was again about a minute. Condition (4.6) was satisfied for $\varepsilon=.01$ with $\omega=5$, $a=3$.

(v) *A model of an overdense plasma.* An overdense plasma column can be modeled by an index of refraction which becomes negative, such as the example used, $n=(r^2-4)/5$ for $r \leqq 3$, $n=1$ for $r \geqq 3$. Geometrical optics predicts [9] a caustic on the ellipse $x^2/4+y^2/9=1$.

Fig. 4 shows the calculated field in polar coordinates with $\omega=5$, $a=2$, $\Delta t=0.05$, $\Delta r=0.1$, and $\Delta\theta=\pi/40$. Recalling that the larger (hence darker) numbers correspond

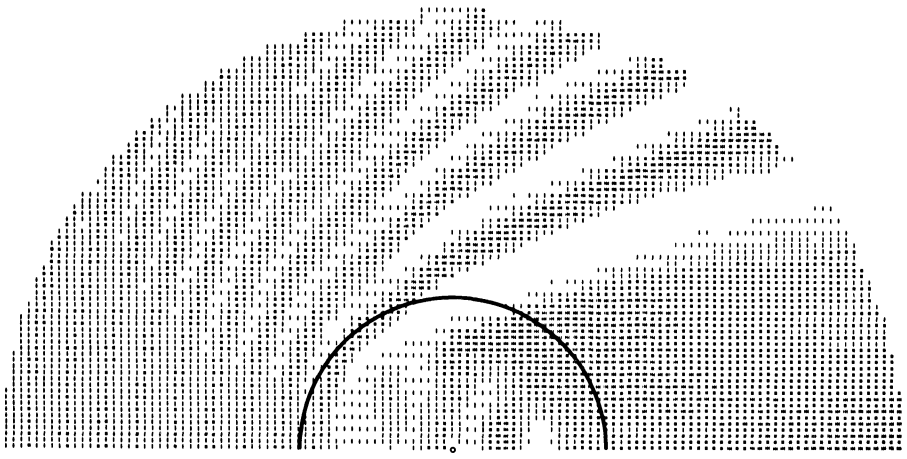


FIG 3. *The interference pattern produced by the total field's amplitude,* $|U|$, *for a dielectric lens with* $n=0.4$. *The relevant parameters are* $\omega a=15$, $\Delta r=0.1$, $\Delta\theta=\pi/20$, *and* $\Delta t=0.05$.

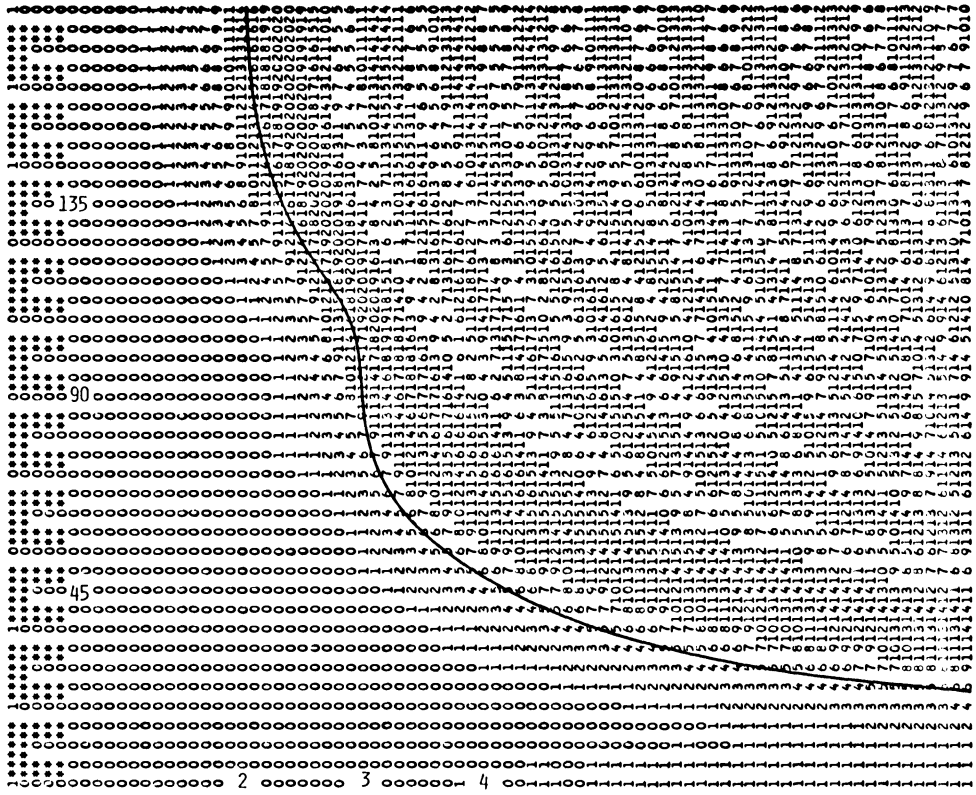


FIG 4. *The total field's amplitude, $|U|$, in polar coordinates for an overdense plasma column. The incident wave was normalized to* 10. *The heavy curve represents the caustic $x^2/4+y^2/9=1$, when $r\leqq 3$ and the geometric shadow $y=3$ when $r\geqq 3$. The relevant parameters are $\omega a=15$, $\Delta r=0.1$, $\Delta\theta=\pi/40$, and $\Delta t=0.05$.*

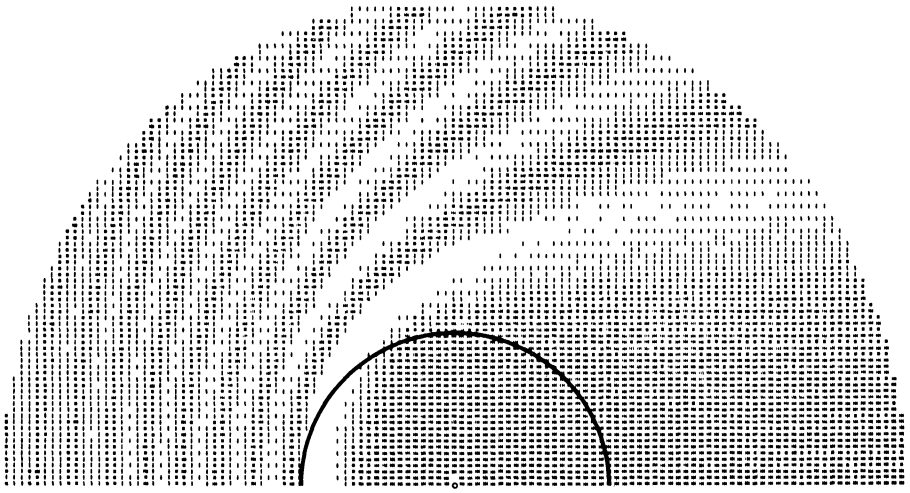


FIG 5. *Same physical problem and relevant parameters as Fig.* 4. *However, this is now the interference pattern in x-y coordinates with the light regions corresponding to constructive interference, the dark to destructive.*

to constructive interference, note that for $\pi/2 \leqq \theta \leqq \pi$ the heavy curve (ellipse) lies very close to the edge of the constructive interference. The rest of the shadow is cast by the caustic along the line $y = r\sin\theta = 3$ (remaining portion of the heavy curve for $r \geqq 3, \theta \leqq \pi/2$). Fig. 5 shows the x-y plot of the interference pattern where the lighter regions now correspond to constructive interference. The amplitude ranges for the various shades are again (0,2), (2,6), (6,8), (8,10), (10,14), (14,17), and $(17,\infty)$. Unfortunately, the interpolation required for the rectangular output smears out the caustic.

6. Conclusions. There are four obvious dimensionless parameters in the computed problem, $\omega\Delta r$, $\omega r\Delta\theta$, ωa and ωr_0, where a is a relevant length and r_0 is the value of r where the radiation condition is imposed. The square of the first two enters the errors produced by the second-order difference scheme and in most of our calculations was approximately 0.2. The third measures the relevance of geometrical optics and ranged up to 15. The last one, usually about 50, arises from an expansion at ∞ for fixed ω. The error terms are of order $(\omega r_0)^{-3}$ but the coefficients are singular as $\omega \to \infty$.

In many of the problems, there were discontinuities, infinitely thin objects or discontinuities in the index of refraction. These induce discontinuities in the second derivatives of the time-dependent solution and in the steady state. In spite of this source of inaccuracy, there was remarkable agreement with known results and there is probably some cancellation of error.

The method could be applied to higher dimensional phenomena where the reflecting object as well as variable index of refraction are not amenable to solution by integral equations. Since the problem is solved by a very straightforward difference scheme, it should be easily effected by vector computation. Other problems where variations of the limiting amplitude principle could be used involve Maxwell's equations, and wave propagation in water.

Finally, the method has been modified slightly and applied to the interesting and important case of a nonlinear medium [12]. In that report

$$n = n_0 + \gamma(1 - n_0)|U|^2,$$

where n_0 is the index of refraction used in our overdense plasma model and γ is a constant which controls the strength of the nonlinearity. The results were excellent; the effects of refraction and self-focusing could be discerned and controlled by varying γ. In particular, self-focusing amplifies the fields near the origin as was first observed by F. Tappert [unpublished results].

Acknowledgments. The authors are grateful to F. Tappert for much advice and many suggestions. The work of A. Bayliss and E. Turkel [3] on similar problems with obstacles treats the radiation condition using the same ideas. Turkel pointed out to the authors that the modified radiation condition is well-posed in the sense of Kreiss-Lobatchinskii [11].

Note added in proof. The authors have learned of the work of Taflove and Brodwin [13], [14], who used the direct form of the limiting amplitude principle to study certain electromagnetic scattering problems.

REFERENCES

[1] J. J. BOWMAN, T. B. A. SENIOR, and P. L. E. USLENGHI, *Electromagnetic and Acoustic Scattering by Simple Shapes*, North-Holland, Amsterdam, 1969.

[2] G. A. KRIEGSMANN and C. S. MORAWETZ, *Numerical solutions of exterior problems with the reduced wave equation*, J. Comput. Phys. 28(1978), pp. 181–197.

[3] A. BAYLISS and E. TURKEL, *Boundary Conditions for Exterior Acoustic Problems*, Report No. 79-7, ICASE, NASA Langley Research Center, 1979.

[4] J. WEIDMANN, *A virial theorem and its application to the spectral theory of Schrödinger operators*, Bull. Amer. Math. Soc., 73(1967), pp. 452–456.

[5] B. ENGQUIST and A. MAJDA, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp. 31(1977), pp. 629–652.

[6] R. N. BUCHAL and J. B. KELLER, *Boundary layer problems in diffraction theory*, Comm. Pure Appl. Math. 13(1960), pp. 104–105.

[7] T. B. A. SENIOR, *Electromagnetic penetration into cavities*, IEEE J. Electromagnetic Compatability, 18(1976), pp. 71–73.

[8] N. G. ALEXOPOULOS, *On the refractive properties of media with poles or zeroes in the index of refraction*, IEEE Trans. Antennas and Propagation, 22(1974), pp. 242–251.

[9] G. A. KRIEGSMANN, *An application of the method of geometrical optics to the scattering of plane electromagnetic waves off cylindrically confined cold plasmas*, J. Math. Phys. 17(1976), pp. 112–120.

[10] C. S. MORAWETZ, *The limiting amplitude principle*, Comm. Pure Appl Math., 15(1962), pp. 349–362.

[11] H. O. KREISS, *Initial boundary value problems for hyperbolic systems*, Comm. Pure Appl. Math., 23(1970), pp. 277–298.

[12] G. A. KRIEGSMANN and C. S. MORAWETZ, *Computations with the nonlinear Helmholtz equation*, submitted to J. of Applied Optics.

[13] A. TAFLOVE and M. E. BRODWIN, *Numerical solution of steady state electromagnetic scattering problems using the time-dependent Maxwell's equations*, IEEE Trans. Microwave Theory Tech., MTT 23(1975), pp. 886–896.

[14] A. TAFLOVE, *Application of the finite-difference time domain method to sinusoidal steady state electromagnetic penetration problems*, IEEE Trans. Electromagnetic Compatability, EMC 3(1980), pp. 191–202.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 3, September 1980

0196-5204/80/0103-0007 $01.00/0

A NUMERICAL METHOD FOR CONFORMAL MAPPINGS*

BENGT FORNBERG†

Abstract. A numerical technique is presented for calculating the Taylor coefficients of the analytic function which maps the unit circle onto a region bounded by any smooth simply connected curve. The method involves a quadratically convergent outer iteration and a super-linearly convergent inner iteration. If N complex points are distributed equidistantly around the periphery of the unit circle, their images on the edge of the mapped region, together with approximations for the $N/2$ first Taylor coefficients, are obtained in $O(N \log N)$ operations. A calculation of time-dependent waves on deep water is discussed as an example of the potential applications of the method.

Key words. conformal mapping, Fast Fourier transform, conjugate gradients, water waves

1. Introduction. A large number of numerical methods have been proposed for calculating approximations to the unique mapping which is described in the Riemann mapping theorem. We will consider the problem of finding the leading Taylor coefficients in a mapping from the unit circle to a given simply connected domain bounded by a smooth curve J.

A book by Gaier [3] gives detailed descriptions of the methods known in 1964. The introduction of the Fast Fourier Transform algorithm (FFT) shortly afterwards (Cooley and Tukey [2], 1965) made possible dramatic increases in the efficiency of some of these methods (Henrici [7]). The most important of the currently used methods seem to be different approximations of Theodorsen's integral equation (Henrici [7], Gutknecht [4], [5]), Symm's method (Symm [10], Henrici [7], Hayes et al. [6]), a method based on numerical solution of the Cauchy-Riemann equations in conjunction with optimization techniques (Chakravarthy and Anderson [1]) and a method based on some new integral equations (Menikoff and Zemack [9]). Theodorsen's method is limited to regions which have single-valued representations in polar coordinates. For almost all such regions, good performance also requires estimates for certain relaxation parameters. The position of the N points on a fixed curve J, which correspond to the N roots of unity in the mapping from the unit circle, can be found in $O(N \log N)$ operations. However, the proportionality constant depends strongly on the shape of J. Symm's method has a similar operation count for simple regions, but in addition, allows general regions with an increase in operation count to $O(N^3)$ (Hayes et al. [6]). The method of Chakravarthy and Anderson [1] requires $O(N^3)$ operations if a Newton optimization technique is used, but may go somewhat faster with an alternative conjugate gradient procedure. The method by Menikoff and Zemach also costs $O(N^3)$ operations, but it allows an arbitrary distribution of the computational points on the boundary.

Most of the methods address the problem of finding a mapping to or from a unit circle. They establish first the mapping of the boundaries. From this follows then the complete mapping function. The method we will present produces approximations to the Taylor coefficients in the mapping from the unit circle onto the given region at the same time as it finds the boundary correspondence. For this reason, we will describe it as a method to map the unit circle onto a given region rather than a method for the

*Received by the editors January 24, 1980. This research was supported by Control Data Corporation and by the U.S. Department of Energy Office of Basic Energy Sciences.

†Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

inverse mapping. The method has an operation count of $O(N\log N)$ for the general case, i.e., without any restrictions on the region to be "near-circular" or "starshaped." The convergence rate appears to be only weakly affected by the complexity of the region and no parameters are required to optimize the performance. Before we describe the idea of the method, we will briefly illustrate some possible ill-conditioning of the mapping problem and mention the application in fluid mechanics for which this method was originally developed.

Fig. 1 illustrates a typical case of mapping from the unit circle to a simple region (described in the last section as test case 1 with $\alpha=.5$). It transpires in this case that more than 8000 Taylor coefficients of the mapping function are needed to obtain the mapping with a 10^{-8} accuracy. This in turn requires N, the number of points on the periphery, to exceed 16,000. The density of the mapped points (uniform on the unit circle) varies by as much as a factor of 400 along different parts of the edge. Very small changes in the shape of the region will quite dramatically change the position of some boundary points. It is clear that conformal mapping is an ill-conditioned problem. One might try to avoid part of this difficulty by looking for a more economical functional representation of the mapping function than a Taylor series. Simply moving the origin would help the economy in this case, but would not help in many others. For example, in cases of fixed shapes like airfoils, initial explicit transforms can be used to first remove a corner or a cusp. A preliminary sequence of transformations, for example with square root branch points just outside indentations in J, may be useful to make the region closer to a circle. Even with the implementation of such preliminary steps we must at some stage find the remaining mapping between a near-circular region and a perfect circle.

Applications of conformal mappings include generation of computational grids and simplifications of geometries for analytical work (for, example, to find electrical fields or potential flows around bodies). Which method is most suitable depends on the application, in particular on the type of geometry (with or without corners, near-circular shape or not, etc.) and the importance of computational efficiency (mapping performed only once or performed repeatedly). The mapping method described in this paper was developed particularly for the calculation of time-dependent waves on inviscid and irrotational deep water. We consider a periodic section of deep water, as shown to the left in a complex $z=x+iy$ -plane in Fig. 2, and introduce a velocity potential $\phi(x, y, t)$ such that ϕ_x and ϕ_y are the x- and y-velocities

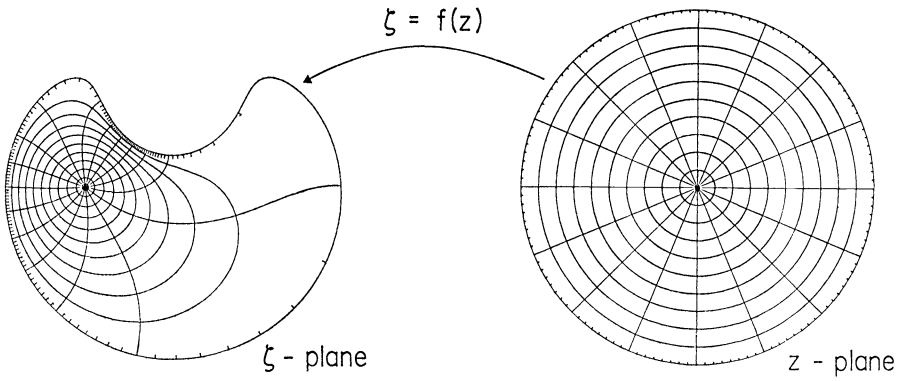


FIG 1. *Example of conformal mapping.*

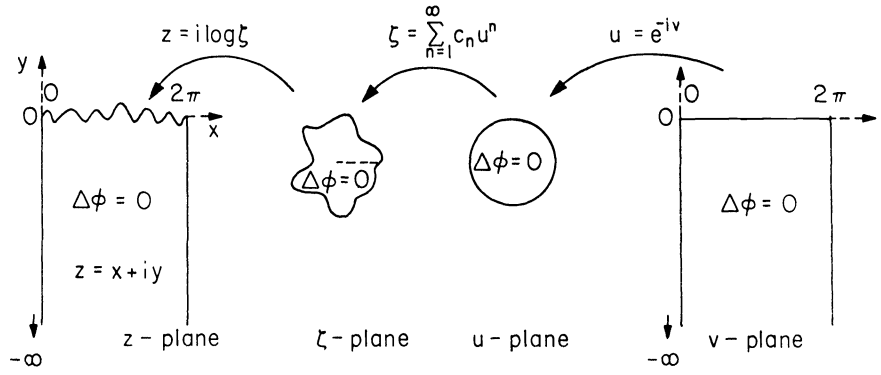


FIG 2. *The steps in a conformal mapping of a wavy surface on deep water to a fixed, flat surface.*

of the fluid elements. The governing equation is

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\phi=0, \tag{1}$$

inside the fluid, with two conditions on the free surface. One is

$$\frac{\partial \phi}{\partial t}=-gy-\tfrac{1}{2}\left(\phi_x^2+\phi_y^2\right), \tag{2}$$

(where g is a gravity constant) and the other one expresses the fact that fluid elements on the surface remain on the surface. Fig. 2 illustrates how, at any step in a numerical time marching, the wavy surface can be mapped to a flat one, leaving the form of (1) unchanged. The derivatives ϕ_x and ϕ_y, required for a time step, can now be obtained easily. In one test calculation, a second order modified Euler scheme was used to advance a uniformly traveling periodic wavetrain five periods in space. The initial condition was a Stokes' wave with height over wavelength .09083 (cf. maximal wave .14107) and a theorectical speed of 1.04155. Fig. 3 shows the solution at time $t=30.1628$ superposed on the initial wave. The very small discrepancy between the curves at the top of the wave is caused by graphical straight line interpolation between the 32 poiints used for the spatial discretization.

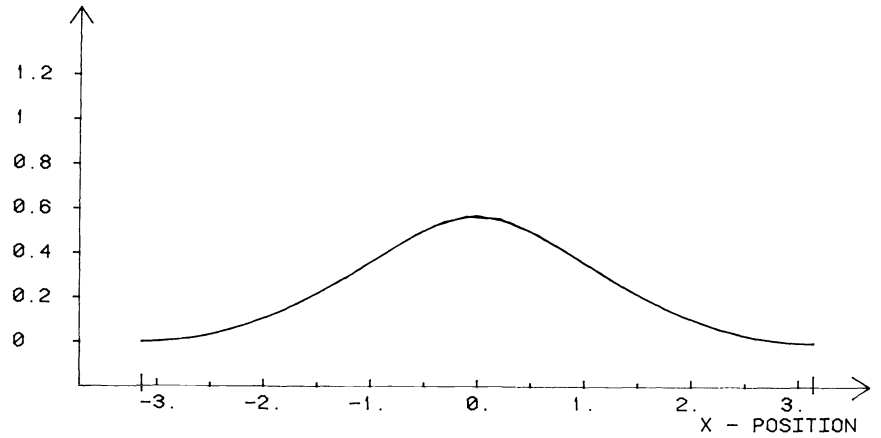


FIG 3. *The initial shape of a Stoke's wave compared with the calculated shape and position after five periods. (Spatial resolution 32 points.)*

The idea of the mapping method can be described as follows: we introduce N complex points ζ_i ordered monotonically along the boundary curve J. The problem is to find a strategy for moving these points along J so that, through the unique mapping (which is unknown to us), they will come to correspond to the N roots of unity z_i on the unit circle. For each guess of the positions of the points ζ_i, an analytic function

$$\zeta(z)=\sum_{\nu=-(N/2)+1}^{N/2} d_\nu z^\nu \tag{3}$$

is introduced. The coefficients d_ν are obtained from a complex discrete Fourier transform applied to the numbers ζ_i. The function $\zeta(z)$ becomes one particular case of an analytic function which maps the points z_i into the points ζ_i on J. However, this function $\zeta(z)$ is in general singular for $z=0$. This is an unacceptable property of a mapping function which is required to satisfy $\zeta(0)=0$. We will describe a quadratically convergent strategy to move all the points ζ_i along the curve J in such a way that the function $\zeta(z)$ loses its singularities inside the unit circle and will satisfy $\zeta(0)=0$. In other words, we will make $d_\nu=0$ for $\nu=0,-1,-2,\ldots,-N/2+1$. The coefficients d_ν for $\nu=1,2,3,\ldots,N/2$ will then approximate the Taylor coefficients of the mapping function. Each step in this process gives rise to a linear problem which is solved by a (super-linearly converging) conjugate gradient iteration.

This mapping method has been coded in vectorized Fortran on a CDC STAR–100 computer (located at the Control Data Corporation Service Center in Arden Hills, Minnesota). We wish to express our gratitude to Control Data Corporation for making their STAR–100 computer system available for this work.

2. Description of the outer iteration. The boundary curve J is assumed to be smooth, simply connected and enclosing the origin in a complex plane. Since J is smooth, the true mapping function $\xi(z)$ can be represented by a convergent Taylor series as

$$\xi(z)=\sum_{\nu=1}^{\infty} c_\nu z^\nu, \qquad |z|\leqq 1. \tag{4}$$

(We denote by $\zeta(z)$ the approximate mapping function we will determine). One more condition than $\xi(0)=0$ (and $\zeta(0)=0$) is needed to uniquely determine the mapping. The condition $\partial\xi/\partial z>0$ is often used. We will instead require that $\zeta(1)$ lies at a specified point on J. For values of z on the unit circle, i.e., $z(\theta)=e^{2\pi i\theta}$, $0\leqq\theta\leqq 1$, (4) becomes

$$\xi(z(\theta))=\sum_{\nu=1}^{\infty} c_\nu e^{2\pi i\nu\theta}. \tag{5}$$

This is called the "boundary correspondence function," a uniquely determined periodic function of θ. We consider now (5) at the θ–values $\theta_k=k/N$, $k=0,1,\ldots,N-1$, and suppose that N is even (N a power of 2 is most efficient and will be assumed in our operation count). We get

$$\xi_k=\xi(z(\theta_k))=\sum_{\nu=-(N/2)+1}^{N/2} g_\nu e^{2\pi i\nu k/N}=\sum_{\nu=-(N/2)+1}^{N/2} g_\nu w^{\nu k}, \tag{6}$$

where $w=e^{2\pi i/N}$ and

$$g_\nu=\sum_{j=0}^{\infty} c_{\nu+jN}, \tag{7}$$

defining $c_\nu = 0$ for $\nu \leqq 0$. The error in accepting g_ν as an approximation for c_ν is

$$g_\nu - c_\nu = \sum_{j=1}^{\infty} c_{\nu+jN}. \tag{8}$$

N can now be chosen so large that these errors for $\nu = 1,2,\ldots,N/2$ are within our tolerance.

The discrete Fourier transform in (6) can be inverted

$$g_\nu = \frac{1}{N} \sum_{k=0}^{N-1} \xi_k w^{-k\nu}, \qquad \nu = -\frac{N}{2}+1,\ldots,\frac{N}{2}. \tag{9}$$

Hence, given the positions for the N points ξ_k, we obtain all the coefficients g_ν by applying one FFT. By choosing N sufficiently large, the values of g_ν, $\nu = -N/2+1,\ldots,0$, become arbitrarily small, and those of g_ν, $\nu = 1,\ldots,N/2$ arbitrarily close to the corresponding c_ν. This leads us to consider the following approximate analogue of (9):

$$d_\nu = \frac{1}{N} \sum_{k=0}^{N-1} \zeta_k w^{-k\nu}, \qquad \nu = -\frac{N}{2}+1,\ldots,\frac{N}{2}. \tag{10}$$

Here, the points ζ_k lie monotonically along J and represent guesses for the numbers ξ_k. We wish to move these points ζ_k on J in such a way that d_ν becomes equal to zero for $\nu = -N/2+1,\ldots,0$. With N free real parameters, we wish to make $N/2$ complex numbers zero. This count of equations and unknowns appears correct, but we have not yet prescribed a position to one of the points. It will transpire that the N equations we obtain after linearization will form a system with rank only $N-1$ (to within truncation errors).

We move the points ζ_k in a two-step process. Given the tangential directions e_k (with $|e_k| = 1$) at the points ζ_k on J, we can try to move these points in the tangential directions by distances t_k in such a way that $d_0, d_{-1},\ldots,d_{-N/2+1}$ become zero:

$$0 = \frac{1}{N} \sum_{k=0}^{N-1} (\zeta_k + t_k e_k) w^{-k\nu}, \qquad \nu = -\frac{N}{2}+1,\ldots,0. \tag{11}$$

Afterwards, the points $\zeta_k + t_k e_k$ are moved back to the curve J. Since J is smooth, the distance from the curve is $O(t_i^2)$. From this follows the quadratic convergence of this outer iteration.

By subtracting the νth equation in (11) from the νth equation in (10) we get $N/2$ complex linear equations for the N real unknowns t_k:

$$d_\nu = -\frac{1}{N} \sum_{k=0}^{N-1} t_k e_k w^{-k\nu}, \qquad \nu = -\frac{N}{2}+1,\ldots,0, \tag{12a}$$

or, using matrix notation:

(12b)

$$\begin{bmatrix} d_0 \\ d_{-1} \\ d_{-2} \\ \vdots \\ d_{-N/2+1} \end{bmatrix} = -\frac{1}{N} \left[\begin{array}{ccccc|c|c} 1 & 1 & 1 & 1 & \cdots & \cdots & 1 \\ 1 & w & w^2 & w^3 & & & w^{N-1} \\ 1 & w^2 & w^4 & w^6 & & & w^{2N-2} \\ \vdots & & & & & & \vdots \\ 1 & w^{N/2-1} & w^{N-2} & w^{3N/2-3} & \cdots & \cdots & w^{(N/2-1)(N-1)} \end{array}\right] \begin{bmatrix} e_0 t_0 \\ e_1 t_1 \\ e_2 t_2 \\ \\ \\ \vdots \\ \\ \\ e_{N-1} t_{N-1} \end{bmatrix}.$$

Two questions must now be investigated.

1. Can the special structure of the coefficient matrix in the linear system (12) be exploited to give a very fast method of solution?

2. The position of one point, for example ζ_0, should be arbitrary. How is this freedom present in (12)? These questions are discussed in the next three sections. A practical implementation of the method and its performance on some test cases is described in the final section.

3. Reformulation of the linear system of equations. Collecting the odd- and even-numbered columns of the matrix in (12b) gives

$$
-\begin{bmatrix} d_0 \\ d_{-1} \\ d_{-2} \\ \vdots \\ d_{-N/2+1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^2 & w^4 & & w^{N-2} \\ 1 & w^4 & w^8 & & w^{2N-4} \\ \vdots & & & & \vdots \\ 1 & w^{N-2} & w^{2N-4} & \dots & w^{2(N/2-1)^2} \end{bmatrix} \times \begin{bmatrix} e_0 & & & & \\ & e_2 & & & \\ & & e_4 & & \\ & & & \ddots & \\ & & & & e_{N-2} \end{bmatrix} \times \begin{bmatrix} t_0 \\ t_2 \\ t_4 \\ \vdots \\ t_{N-2} \end{bmatrix}
$$

$$
+\frac{1}{N} \begin{bmatrix} w^0 & & & & \\ & w^1 & & & \\ & & w^2 & & \\ & & & \ddots & \\ & & & & w^{N/2-1} \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^2 & w^4 & & w^{N-2} \\ 1 & w^4 & w^8 & & w^{2N-4} \\ \vdots & & & & \vdots \\ 1 & w^{N-2} & w^{2N-4} & \dots & w^{2(N/2-1)^2} \end{bmatrix} \tag{13}
$$

$$
\times \begin{bmatrix} e_1 & & & & \\ & e_3 & & & \\ & & e_5 & & \\ & & & \ddots & \\ & & & & e_{N-1} \end{bmatrix} \times \begin{bmatrix} t_1 \\ t_3 \\ t_5 \\ \vdots \\ t_{N-1} \end{bmatrix},
$$

or

$$
-\mathbf{d} = \frac{1}{N} FE_0\mathbf{t}_0 + \frac{1}{N} WFE_1\mathbf{t}_1. \tag{14}
$$

Here, F is the discrete Fourier transform matrix of order $N/2$. The matrices $1/\sqrt{N/2}\ F, E_0, E_1$ and W are all unitary. Multiplication of any vector by any of these matrices or their inverses will require at most $O(N\log N)$ operations.

Let us consider the following iteration: Given any real vector $\mathbf{t}_0{}^{(0)}$, solve (14) for the uniquely determined complex vector $\mathbf{t}_1{}^{(0)}$, remove the imaginary parts from $\mathbf{t}_1{}^{(0)}$ and then solve (14) for $\mathbf{t}_0{}^{(1)}$ where the imaginary parts that are obtained are again removed.

The real vector $\mathbf{t}_0{}^{(1)}$ that results from one step of this iteration will depend linearly on the initial vector $\mathbf{t}_0{}^{(0)}$. There must therefore be a relation

$$
\mathbf{t}_0{}^{(1)} = A\mathbf{t}_0{}^{(0)} + \mathbf{b}. \tag{15}
$$

Straightforward algebra gives now

$$A = R^T R, \quad \text{where } R = \operatorname{Re} C, \quad C = \frac{2}{N} E_0^H F^H W F E_1. \tag{16}$$

A is symmetric and positive semidefinite and C is a unitary complex matrix. Since $\|A\|_2 = \|R\|_2^2 \leqq \|C\|_2^2 = 1$, the eigenvalues λ_i of A satisfy

$$0 \leqq \lambda_i \leqq 1, \qquad i = 0, 1, \ldots, \frac{N}{2} - 1. \tag{17}$$

If there is a real solution $\mathbf{t}_0, \mathbf{t}_1$ of (14), $\mathbf{t}_0^{(0)} = \mathbf{t}_0$ must imply $\mathbf{t}_0^{(1)} = \mathbf{t}_0$ in (15). Hence, $\mathbf{t}_0$ must then satisfy

$$G\mathbf{t}_0 = \mathbf{b}, \tag{18}$$

where $G = I - A$ is again positive semidefinite. Once $\mathbf{t}_0$ has been computed, $\mathbf{t}_1$ follows from (14). The calculation of $\mathbf{t}_0{}^{(1)}$ from $\mathbf{t}_0{}^{(0)}$, in particular the calculation of $\mathbf{b}$ by starting with $\mathbf{t}_0{}^{(0)} = 0$ in (15), and the multiplication of any real vector by G is performed in $O(N \log N)$ operations if N is a power of 2.

We will now further investigate the eigenvalues of G and describe how the conjugate gradient method can be applied very efficiently to solve a modification of the system (18).

4. Eigenvalues of the *G*-matrix. The outer iteration was designed to force to zero not only the coefficients $d_{-1}, d_{-2}, \ldots, d_{-N/2+1}$ but also d_0, thereby ensuring $\zeta(0) = 0$. The mapping is still arbitrary with respect to a rotation. In particular, we should be able to require $t_0 = 0$, where t_0 is the first component of the vector $\mathbf{t}_0$. That means that the point ζ_0 is not moved during the mapping process. Imposing $t_0 = 0$ is only consistent with (18) if G is singular and if $\mathbf{b}$ lies in the subspace spanned by the columns of G except the first one. This was true (to within truncation accuracy) in all cases we tested. The following examples illustrate this result, and show typical distributions for the remaining eigenvalues.

Example 1. Mapping of the unit circle onto itself. The points $\zeta_j, j = 0, 1, \ldots, N-1$, become equidistantly spaced in this trivial mapping. Omitting a complex factor of unit magnitude (corresponding to a rotation), the tangential direction at the point ζ_j is

$$e_j = e^{2\pi i j/N}. \tag{19}$$

The elements of C become in general

$$c_{m,n} = \frac{1}{N} \{1 + i \cot p\pi\} \frac{e_{2n-1}}{e_{2m-2}}, \tag{20}$$

where $p = [2(n-m)+1]/N$. In this case, this simplifies to

$$c_{m,n} = \frac{1}{N} \{-1 + i \cot p\pi\}. \tag{21}$$

The elements of the matrix R are

$$r_{m,n} = -\frac{1}{N}, \tag{22}$$

independently of m and n. The matrix A has one eigenvalue equal to one and all the others equal to zero. Therefore, the matrix G has one eigenvalue equal to zero and the others equal to one.

The bottom line in (13) gives in this case

$$-d_{-(N/2)+1}=(t_0+t_2+\cdots+t_{N-2})-(t_1+t_3+\cdots+t_{N-1}). \tag{23}$$

Since all the t_i are real, the requirement for a solution is that $\operatorname{Im} d_{-(N/2)+1}=0$. This is satisfied since, in the mapping, all coefficients except the first one are absent. It is possible to add the same constant to all t_i in (13) and (18). This corresponds to a rotation of the mapping. Fixing $t_0=0$ removes this ambiguity.

Example 2. We consider regions bounded by curves given in the complex plane $\zeta=x+iy$ by

$$f_1(x,y,\alpha)\equiv\big((x-.5)^2+(y-\alpha)^2\big)\big(1-(x-.5)^2-y^2\big)-.1=0. \tag{24}$$

For $\alpha=\infty$, this defines a circle with center at $x=.5$, $y=0$. Fig. 4 shows the curves for some different values of α down to $\alpha\approx.2746687749$, at which point the region ceases to be simply connected. Fig. 5 illustrates, for different α, the distribution of the eigenvalues of G (and of the matrix $\hat{G}$ which we will introduce below).

In every example we have studied, the eigenvalues of the G show the same pattern. One eigenvalue is zero to truncation error accuracy and all others lie in a heavy cluster around $\lambda=1$. A few double eigenvalues gradually move toward smaller values of λ as the complexity of the curve increases. Increasing the number of points N for a fixed α made (to within truncation accuracy) no difference in this picture or in the positions of the eigenvalues pairs. All additional eigenvalues simply joined the cluster at $\lambda=1$.

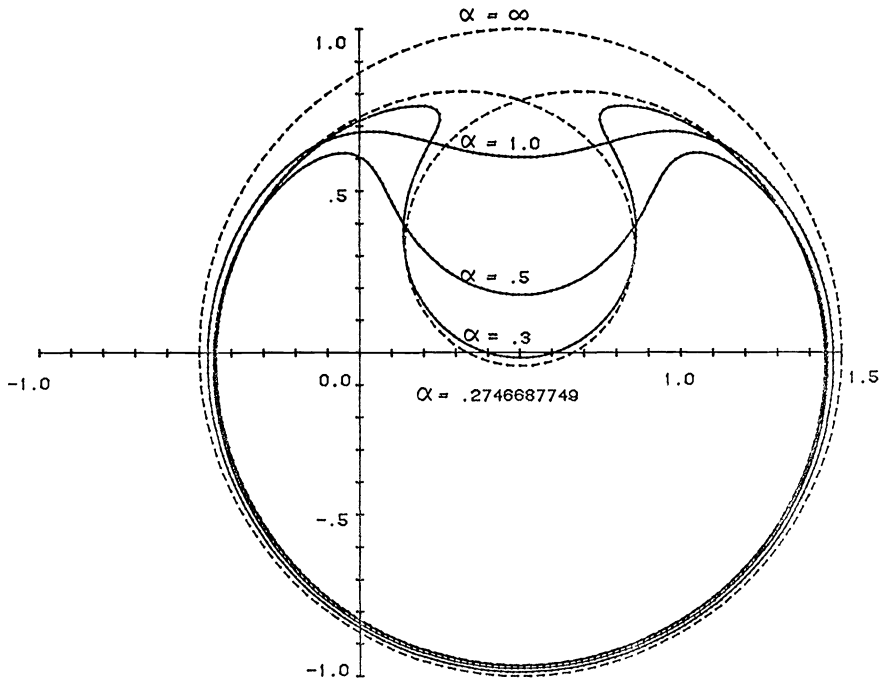


FIG 4. *The curves* $f(x,y,\alpha)\equiv((x-.5)^2+(y-\alpha)^2)(1-(x-.5)^2-y^2)-.1=0$ *for different values of* α.

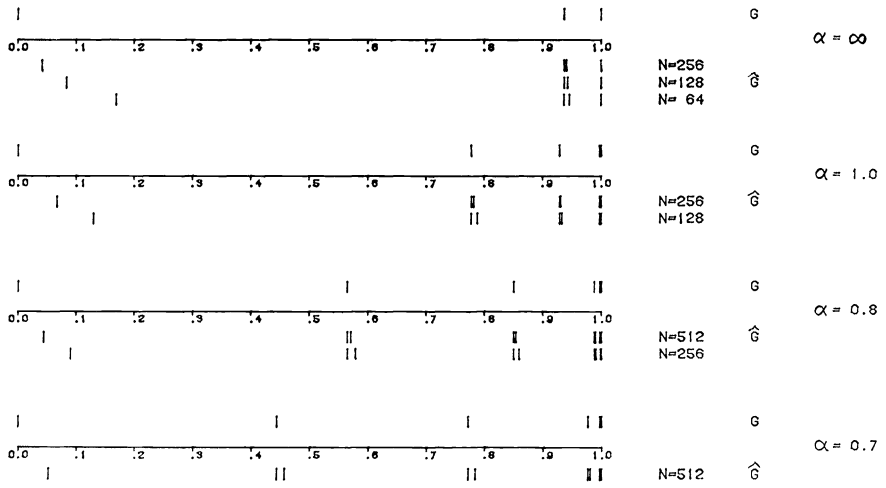


FIG 5. *Eigenvalues of G (virtually independent of N) and of $\hat{G}$ (every second dependent on N) for different values of α in the test case described in the text.*

In the case of extremely low values of N, such as $N=4$ or $N=8$, these eigenvalue properties fail to hold to within truncation accuracy. (In particular, the matrix G is no longer exactly singular.) This indicates that they are not exact properties of the discrete systems. These observations suggest that a smooth curve has some kind of spectrum with the same properties (but with $\lambda=1$ a limit point) and that the discrete method provides exponentially accurate approximations to it for increasing values of N. We have not been able to find any theoretical support for these observations.

The previous discussion has suggested that we can require $t_0=0$ and also consider one equation, for example, the first one, redundant. We write, therefore, (18)

$$\left[\begin{array}{c|c} g & \hat{g}^T \\ \hline \hat{g} & \hat{G} \end{array}\right]\left[\begin{array}{c} t_0 \\ \hline \hat{t} \end{array}\right]=\left[\begin{array}{c} b_0 \\ \hline \hat{b} \end{array}\right]. \tag{25}$$

This leaves us to solve

$$\hat{G}\hat{t}=\hat{b}. \tag{26}$$

The matrix G is positive semidefinite with only one eigenvalue equal to zero. Therefore, $\hat{G}$ is strictly positive definite, again with a cluster at $\lambda=1$. In the case in Example 1, $\hat{G}$ has one eigenvalue equal to $1/N$ and all the others equal to one. Fig. 5 shows the eigenvalues of $\hat{G}$ corresponding to those of G described earlier in Example 2.

In every case we have studied, we have also noticed the same trend in the eigenvalues of $\hat{G}$. As we mentioned above, G was found to have an eigenvalue $\lambda_1=0$, then double eigenvalues $\lambda_{2,3}$, $\lambda_{4,5}$, etc., clustering at $\lambda=1$. For each double eigenvalue of G, $\hat{G}$ must have a single eigenvalue and the remaining eigenvalues of $\hat{G}$ must lie between the pairs for G. We have noticed that these eigenvalues converge from above to $\lambda_1=0$, to $\lambda_{2,3}$, to $\lambda_{4,5}$, etc., with a rate which seems to be proportional to $1/N$ as N increases.

5. Practical implementation of the mapping method. Conjugate gradient iterations are applied to (26) in a straightforward manner (see for example Luenberger [8] for a description of the method). We simplify the notation of (26) to

$$Hx = b. \tag{27}$$

The method is initialized by the steps

$$\begin{aligned} &\text{choose} && x_0 = 0, \\ &\text{compute} && r_0 = b - Hx_0, \\ &\text{set} && p_0 = r_0, \end{aligned} \tag{28}$$

and then iterated for $i = 0, 1, 2, \ldots,$

$$\begin{aligned} a_i &= \frac{r_i^T r_i}{p_i^T H p_i}, \\ x_{i+1} &= x_i + a_i p_i, \\ r_{i+1} &= r_i - a_i H p_i, \\ b_i &= \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}, \\ p_{i+1} &= r_{i+1} + b_i p_i. \end{aligned} \tag{29}$$

We stop iterating when the changes in the x_i are sufficiently small. The level of errors that we desire to reach depends on the current accuracy in the quadratically convergent outer iteration. The ith approximation in the conjugate gradient method can be shown to minimize

$$\|x - x_i\|_H^2 = (x - x_i)^T H (x - x_i), \tag{30}$$

over all approximations of the form

$$x_i = x_0 + P_{i-1}(H) \cdot H \cdot (x - x_0), \tag{31}$$

where P_{i-1} is any polynomial of degree $i-1$. This result is very favorable in cases with eigenvalue distributions like the one in Fig. 5 but with their exact positions unknown. Complete convergence is assured in the same number of steps as there are distinct groups of eigenvalues. Convergence to sufficient accuracy may take still fewer steps.

The computational cost of the inner iteration is dominated by the fast Fourier transforms. Finding **d** (10) costs one transform over N points, or equivalently two transforms over $N/2$ points (denote for simplicity 2 FFTs). Initializing the conjugate gradient method adds 3 FFTs (to find **b**) and each iteration costs another 4 FFTs (to evaluate Hp_i). When the iterations are finished, 2 FFTs are needed to find $\mathbf{t}_1$. One outer iteration with K inner iterations will therefore cost $7+4K$ FFTs. In the last section, we will see that K is usually about six, which means that each outer iteration typically requires about 30 FFTs (complex transforms over $N/2$ points).

The mapping method requires information about the boundary curve J in two connections:

i) To find the tangent directions at each point.

ii) To move a point back to the curve after it has been moved in the tangential direction.

Any parameter representation of J can be used (for example polar coordinates if the region is "starshaped", spline representation between discrete points, etc.). For the test

runs described below, we implemented the method assuming the curve J was given in the form

$$f(x,y,\alpha)=0, \tag{32}$$

with fixed α. (The parameter α allowed us to modify the curve.) Routines were provided for evaluating $f, f_x=\partial f/\partial x$ and $f_y=\partial f/\partial y$ at any point $\zeta_0=x_0+iy_0$ near J. A unit tangential vector at ζ_0 on J is obtained as

$$\frac{(-f_y,f_x)}{(f_x^2+f_y^2)^{\frac{1}{2}}}, \tag{33}$$

and a point ζ_0 off the curve J can, to within sufficient accuracy, be brought back to the curve with one step of a quadratically convergent Newton iteration:

$$d=\frac{f}{(f_x^2+f_y^2)}, \tag{34}$$
$$x_1=x_0-d\cdot f_x,$$
$$y_1=y_0-d\cdot f_y.$$

Tests have to be made to determine the number of outer and inner iterations. The rules implemented to obtain an automatic code were as follows:

1. Number of outer iterations. Each time an outer iteration is started, a residual vector is obtained (which later forms the right-hand side of the linear system in the inner iteration). Outer iterations are stopped when the maximal element of this vector has not decreased by more than a factor of 2 since the last iteration. Since the outer iterations are quadratically convergent, an improvement by a factor less than two indicates that the rounding or truncation error level has been reached.

2. Number of inner iterations. The inner iterations use conjugate gradients to approximate the vector $\mathbf{t}_0$; i.e., the distances the even-numbered points (apart from ζ_0 which is held fixed) are to be moved. These iterations were performed until all elements of $\mathbf{t}_0$ had settled to within .001 of the size of the maximal element of $\mathbf{t}_0$. These tests can easily be improved. Since the outer iterations are quadratically convergent, the accuracy in the inner iterations ought to be increased correspondingly. Also, the last outer iteration gives only a small improvement (at most by a factor of two). For each special application, a test should be devised which allows this last iteration to be omitted. For example, in the case of solving a time-dependent problem in a slowly changing geometry, it may be possible to use just one outer iteration for each numerical time step in the main problem.

6. Test results. In this section, the application of the mapping method to the following two one-parameter families of curves is described.

Case 1.

$$f(x,y,\alpha)\equiv\big((x-.5)^2+(y-\alpha)^2\big)\big(1-(x-.5)^2-y^2\big)-.1=0. \tag{35}$$

Fig. 4 showed these curves for some different values of α. The value $\alpha=\infty$ gives a circle with center at (.5,0) and radius 1. The curve ceases to be "starshaped" (i.e., ceases to have a single-valued radius in polar coordinates) at $\alpha\approx.7675275331$ and

ceases to be simply connected at $\alpha \approx .2746687749$. The mapping method was employed for values of α down to $\alpha = .5$.

Case 2. (Cassini's oval).

$$f(x, y, \alpha) \equiv \big((x+\alpha)^2+y^2\big)\big((x-\alpha)^2+y^2\big) - 1 = 0. \tag{36}$$

Fig. 6 shows these curves for some values of α ranging from 0 to 1. In the case $\alpha = 0$, the curve becomes the unit circle. For $\alpha = 1$, it ceases to be simply connected. The mapping function from the unit circle can in this case be found in closed form:

$$\zeta(z) = z\left(\frac{1-\alpha^4}{1-(\alpha z)^2}\right)^{\frac{1}{2}} = (1-\alpha^4)^{\frac{1}{2}} z \sum_{n=0}^{\infty} \frac{(2n)!}{(n!)^2}\left(\frac{z\alpha}{2}\right)^{2n}. \tag{37}$$

Tables 1 and 2 illustrate the performance of the mapping method for these two test cases. Fig. 7 shows some corresponding mappings. For each parameter value, the initial guess on the distribution of the points ζ_i was obtained from the solution for the previous value of α together with two Newton iterations to bring these points to the curve for the new α. If the number of points N was doubled, a fourth-order interpolation was used to find the positions of the new points. In most cases, larger continuation steps in α than those shown would also have worked. At the bottom of Table 1, we give one case in which we tested for the largest possible step. Tables 1 and 2 show the maximal distance any boundary point was moved in each step of the mapping. In these two test cases, an approximate guide for finding the largest allowed continuation step seemed to be that the initial point positions should not be in error by more than about 0.25. If points had to be moved further than that tangentially to the curve in the first outer iteration, they are likely to be returned to the curve in

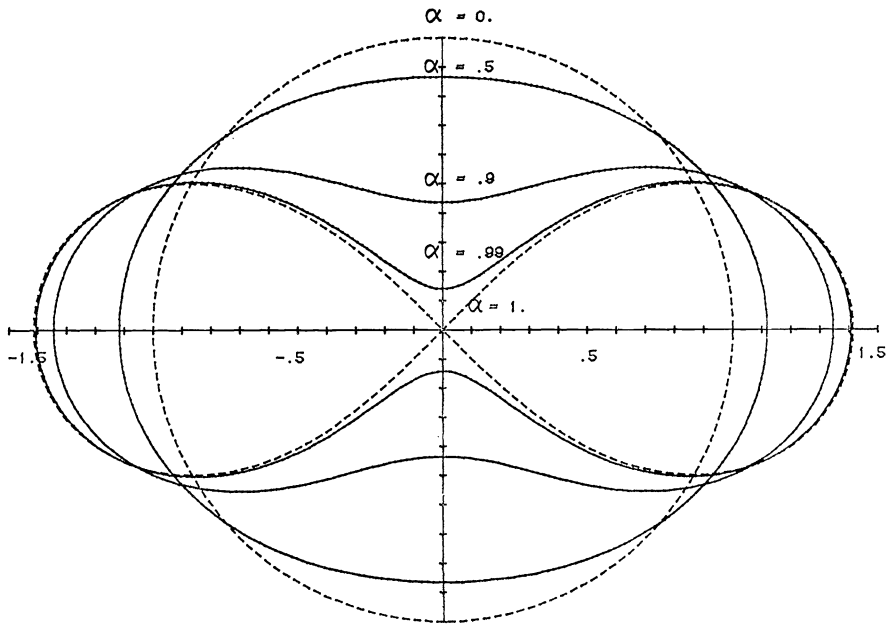


FIG 6. *The curves* $f(x, y, \alpha) \equiv ((x+\alpha)^2+y^2)((x-\alpha)^2+y^2) - 1 = 0$ *for different values of* α.

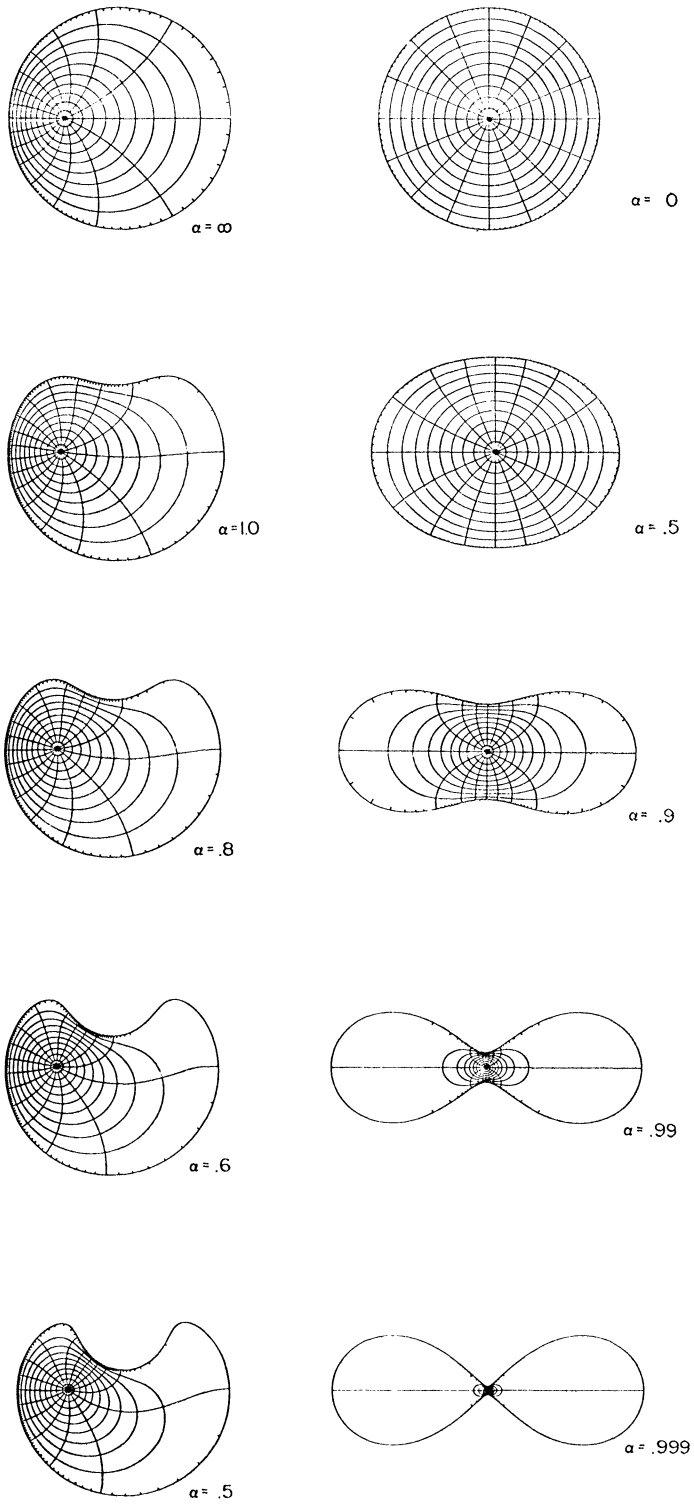


FIG 7. *Examples of conformal mappings in the two test cases.*

places which are no more accurate or may be returned to the curve out of sequential order. If a careful algorithm was developed for returning the point to the curve after each tangential move, still larger continuation steps could probably be used.

We observe in the two test cases that the number of outer iterations seems to depend mainly on the accuracy of the initial guess (which can be read from the column labeled ("Max distance point moved in mapping"). The average number of inner iterations for each outer iteration increased only marginally with increased complexity of the boundary curve. It was found that the method generally produced a uniform absolute accuracy in all the $N/2$ produced Taylor coefficients, and that the error level agreed in size with the first omitted coefficients. The residual vector in the outer iterations also reached this same size. The numbers displayed in the column "Accuracy reached in Taylor coefficients" have been obtained as the maximal element in the final residual vector in the outer iterations. When we keep N fixed and change the parameter α, the changes in the accuracy that was reached only reflect the changes in the decay rates for the leading coefficients.

TABLE 1

Performance of the mapping method in test case 1.

α	N	Nr. of outer iterations	Nr. of inner iterations per outer iterations			Max distance between curves	Max distance point moved in mapping	Accuracy reached in Taylor coefficients	Total comp. time (sec) CDC STAR-100
			Max	Min	Average				
∞	128	Points equidistantly distributed along the curve							
∞	128	7	4	2	3.0	.000	1.000	$.14\cdot10^{-13}$	.08
2.0	128	4	4	4	4.0	.047	.028	$.17\cdot10^{-11}$	.07
1.5	128	4	5	4	4.3	.088	.058	$.14\cdot10^{-7}$	.07
1.2	128	4	5	4	4.5	.140	.118	$.33\cdot10^{-5}$	.07
1.2	256	2	5	4	4.5	.001	.001	$.97\cdot10^{-8}$	.05
1.0	256	4	5	5	5.0	.137	.176	$.17\cdot10^{-5}$	.11
1.0	512	2	5	4	4.5	.000	.000	$.40\cdot10^{-8}$	.09
.9	512	4	6	5	5.5	.076	.149	$.26\cdot10^{-6}$	.19
.9	1024	3	6	5	5.7	.000	.000	$.98\cdot10^{-10}$	.28
.8	1024	5	6	6	6.0	.082	.206	$.42\cdot10^{-7}$	.48
.8	2048	3	6	4	5.0	.000	.000	$.55\cdot10^{-11}$	.53
.75	2048	5	6	3	5.4	.042	.147	$.58\cdot10^{-9}$	.94
.72	2048	5	7	4	5.8	.026	.109	$.40\cdot10^{-8}$	.97
.70	2048	4	7	4	5.8	.018	.083	$.30\cdot10^{-7}$	.78
.70	4096	3	6	4	5.0	.000	.000	$.31\cdot10^{-11}$	1.24
.68	4096	5	7	4	5.8	.018	.091	$.38\cdot10^{-10}$	2.24
.66	4096	5	7	4	5.8	.018	.101	$.13\cdot10^{-9}$	2.24
.64	4096	5	7	4	6.2	.018	.111	$.29\cdot10^{-8}$	2.33
.62	4096	4	7	7	7.0	.018	.122	$.19\cdot10^{-7}$	2.01
.60	4096	4	8	7	7.5	.018	.135	$.84\cdot10^{-7}$	2.10
.60	8192	3	7	5	6.0	.000	.000	$.23\cdot10^{-10}$	3.37
.58	8192	5	8	5	6.6	.018	.149	$.78\cdot10^{-9}$	5.92
.56	8192	5	8	5	6.8	.018	.165	$.90\cdot10^{-9}$	6.05
.54	8192	5	8	5	6.8	.019	.182	$.22\cdot10^{-7}$	6.04
.54	16384	3	7	2	4.7	.000	.000	$.28\cdot10^{-10}$	6.86
.52	16384	6	7	5	5.7	.019	.200	$.99\cdot10^{-9}$	15.20
.50	16384	7	8	5	6.4	.019	.218	$.15\cdot10^{-7}$	19.22
Longest continuation step from $\alpha=.70$, $N=4096$ that worked									
.64	4096	6	7	7	7.0	.053	.248	$.29\cdot10^{-8}$	3.00

TABLE 2
Performance of the mapping method in test case 2.

α	N	Nr. of outer iterations	Nr. of inner iterations per outer iterations			Max distance between curves	Max distance point moved in mapping	Accuracy reached in Taylor coefficients	Total comp. time (sec) CDC STAR-100
0.	128	Points equidistantly distributed. (Exact mapping).							
0.	128	2	3	3	3.0	.000	.000	$.28\cdot10^{-13}$	.03
.5	128	5	5	2	2.6	.133	.127	$.14\cdot10^{-13}$	.06
.7	128	5	6	2	3.0	.151	.154	$.17\cdot10^{-10}$	.06
.8	128	4	3	3	3.0	.114	.139	$.66\cdot10^{-7}$	.05
.9	128	4	3	3	3.0	.163	.257	$.81\cdot10^{-4}$	.05
.9	256	3	3	3	3.0	.006	.004	$.69\cdot10^{-7}$	.06
.93	256	4	3	3	3.0	.068	.143	$.37\cdot10^{-5}$	.07
.95	256	3	3	3	3.0	.055	.137	$.47\cdot10^{-4}$	.06
.95	512	3	3	3	3.0	.006	.004	$.47\cdot10^{-7}$	.10
.97	512	4	3	3	3.0	.069	.212	$.75\cdot10^{-5}$	.13
.97	1024	3	3	3	3.0	.004	.002	$.22\cdot10^{-8}$	.18
.98	1024	4	3	3	3.0	.044	.172	$.34\cdot10^{-6}$	.24
.99	1024	4	4	3	3.5	.058	.291	$.42\cdot10^{-4}$	.26
.99	2048	3	4	3	3.7	.012	.010	$.18\cdot10^{-6}$	.42
.993	2048	3	4	3	3.7	.023	.154	$.32\cdot10^{-5}$	.42
.993	4096	3	4	3	3.7	.004	.003	$.18\cdot10^{-8}$	.97
.995	4096	4	4	3	3.5	.018	.146	$.91\cdot10^{-7}$	1.26
.997	4096	4	4	3	3.8	.022	.220	$.42\cdot10^{-5}$	1.30
.997	8192	3	4	3	3.7	.007	.005	$.65\cdot10^{-8}$	2.38
.998	8192	4	4	4	4.0	.014	.176	$.32\cdot10^{-6}$	3.29
.999	8192	3	4	4	4.0	.018	.293	$.14\cdot10^{-4}$	2.49
.9991	6384	2	4	4	4.0	.021	.020	$.16\cdot10^{-6}$	3.96

REFERENCES

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

[2] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19(1965), pp. 297–301.

[3] D. GAIER, *Konstruktive Methoden der Konformen Abbildung*, Springer Tracts in Natural Philosophy, 3(1964), Springer, Berlin.

[4] M. H. GUTKNECHT. *Solving Theodorsen's Integral Equation for Conformal Maps with the Fast Fourier Transform, Part I, Theory*, Research Report 79–02, Seminar für angewandte Mathematik, Eidgenössische Technische Hochschule, Zürich, 1978.

[5]_____. *Solving Theodorsen's Integral Equation for Conformal Maps with the Fast Fourier Transform, Part II, Practice*, Research Report 79–04, Seminar für angewandte Mathematik, Eidgenössische Technische Hochschule, Zürich, 1979.

[6] J. K. HAYES, D. K. KAHANER AND R. KELLNER, *An improved method for numerical conformal mapping.* Math. Comp., 26 (1972), pp. 327–334.

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

[8] D. G. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[9] R. MENIKOFF AND C. ZEMACH, *Methods for numerical conformal mapping*, J. Comput. Phys., 36(1980), pp. 366–410.

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

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/81/0104-0001 $01.00/0

MINIMUM COVERING ELLIPSES*

B. W. SILVERMAN† AND D. M. TITTERINGTON‡

Abstract. With the aid of a duality relation originally obtained in the theory of statistical experimental design, an exact terminating algorithm is developed for finding the ellipse of smallest area covering a given plane point set. Some applications and related problems are discussed. Empirical timings show the algorithm to be highly efficient, particularly for large sets of points.

Key words. computational geometry, convex hulls, optimal design, D-optimality, spatial data analysis, ranks

1. Introduction. Suppose $V=\{v_1, \cdots, v_n\}$ is a set of points in $\mathbb{R}^k$. Then the *minimum ellipsoid problem* for V is that of finding the k-dimensional ellipsoid, $\mathrm{ME}(V)$, of smallest generalized volume or content that contains V. In algebraic terms we have to choose a $k \times k$ nonnegative definite matrix N and a k-vector c to maximize $\det N$ subject to

$$(v_j - c)^T N (v_j - c) \leqq k, \qquad j = 1, \cdots, n.$$

The existing algorithms for solving this problem are of a nonexact iterative kind; see, for example, [22] and the references therein. In this paper we develop an exact terminating algorithm for the minimum ellipse problem in the plane. This algorithm is very fast and requires only elementary operations, except, in our implementation, for the solution of a cubic equation.

We shall assume throughout that $n \geqq 3$; for n equal to 1 or 2 the problem is trivial.

2. Applications. Minimum ellipsoids are useful in various statistical contexts. They satisfy the property of affine invariance, and can be used as peeling devices in data analysis in a similar manner to the use of convex hulls by Barnett [1] and Green and Silverman [12]. Although, in contrast to the convex hull case, the sequence of minimum ellipsoids may not be nested, they do provide, for most data sets, a finer "peeling". Almost always, the number of points on the surface of a minimum ellipsoid in k dimensions is between $k+1$ and $k(k+3)/2$. In two dimensions this means that 3, 4 or 5 points are peeled off at a time, which is much fewer than, say, for the convex hull results quoted in Green and Silverman [12]; see also Bentley and Shamos [3].

As with convex hulls, the early peels provide a means of detecting outliers, or of trimming off possible outliers. The effect of a single such peel in the context of robust estimation of a correlation coefficient is described by Titterington [22]. There also the use of the minimum ellipsoid itself to estimate the mean and correlation structure of a population is discussed. This is particularly useful if the interior of a data set is somehow obliterated but where the outer crust can be observed.

Minimum ellipsoids can also be used to separate clouds of points, as discussed by Rosen [16]. He takes a different criterion for optimality, choosing to minimize $\mathrm{tr}(N^{-1})$, in the notation of § 1. Even for planar V there seems as yet to be no exact algorithm for this problem.

* Received by the editors September 17, 1980. Part of this research was carried out while the authors were visiting the Department of Statistics, Princeton University, Princeton, New Jersey, supported by the U.S. Department of Energy under contract EI-78-S-01-6540 (BWS) and the U.S. Office of Naval Research under contract N00014-75-C-0453 (DMT).

† School of Mathematics, University of Bath, Bath, BA2 7AY, England.

‡ Department of Statistics, University of Glasgow, Glasgow, G12 8QW, Scotland.

Another application is in operational research. Analogous problems involving spheres and circles have been considered in the context of optimal location of facilities relative to the positions of customers [6], [7], [8], [9], [14], [17] and the ellipsoid problem can perhaps be regarded as modeling "nonisotropic" versions of these.

A final application concerns the calculation of optimal experimental designs for estimating parameters in a linear regression when observations may be made at points in V. Although this application is more esoteric, the link between the two problems is important, particularly in view of a duality relationship given in the next section. This duality leads to an important theoretical result which justifies our algorithm for the minimum ellipse problem.

3. Theoretical considerations; duality and optimal designs. Consider the following optimization problem. Let $\lambda = (\lambda_1, \cdots, \lambda_n)$ be a set of nonnegative numbers summing to 1 and define the nonnegative definite matrix

$$M_0(\lambda) = \sum_{i=1}^{n} \lambda_i (v_i - \bar{v})(v_i - \bar{v})^T,$$

where $\bar{v} = \sum \lambda_i v_i$. A matrix $M_0(\lambda^*)$ is said to be D_0-optimal on V if it minimizes $-\log \det M_0(\lambda)$ over choices of λ.

Direct application of the theory of Lagrange multipliers shows that the D_0-optimal problem for V is the dual of the minimum ellipsoid problem for V; see [21] for details. Further, ME (V) is defined in the terminology of § 1 by $N^{-1} = M_0(\lambda^*)$ and c given by the corresponding $\bar{v}$. Since the dual criterion function $-\log \det M$ is strictly convex on the set of nonnegative definite matrices M, it follows that the minimizing matrix $M = N^{-1}$ and the vector c are unique and hence that ME (V) is unique; see, for example, Sibson [18]. It should be pointed out that the optimal measure λ^* may not be unique.

In the experimental design context, λ_j denotes the proportion of the total number of observations to be taken at the point v_j, and the problem is to choose λ to produce optimal information about the underlying statistical model; see Fedorov [10]. The D_0-optimality criterion reflects one particular possible definition of "optimal".

The duality can be exploited to prove the following theorem which is essential to the development of our exact algorithm for the planar case.

THEOREM. *Given any plane set of points V, containing at least three points, there exists a subset S of V such that*

(1) *S consists of* 3, 4 *or* 5 *points*;

(2) *the points of S lie on* ME (V);

(3) ME (S) = ME (V);

(4) *no subset of S has properties* (1), (2) *and* (3).

Proof. S will contain points in V for which the components of an optimal design λ^* are positive. Existence of an optimal design is guaranteed by the boundedness of V. Conclusions (1) and (2) follow directly from optimal design theory; see [21], [18], [10, Thm. 2.2.3].

The vector λ^* defines a design measure that is D_0-optimal for V and, since S is contained in V, a fortiori for S. By the duality theorem, then, the ellipse corresponding to λ^* is both ME (S) and ME (V), proving conclusion (3). To prove the final part suppose (1), (2) and (3) are satisfied by both S and a proper subset S' of S; replace S by S' and if necessary repeat this step to obtain a subset of V which satisfies the theorem.

A set such as S will be called a *support set* of ME (V). It follows at once from the theorem that S is a support set of ME (V) if and only if

(C1) S is itself a support set of ME (S); and

(C2) ME (S) contains V.

Note that, for points V in general position, the support set of ME (V) will be unique. It is only in certain special cases (for example when V forms a regular hexagon) that ME(V) will have more than one support set.

4. Description of the algorithm. The philosophy of the algorithm is first to use an efficient convex hull algorithm to find the set V_E of extreme points of V. It is clear that ME (V_E) is identical to ME (V) and that, in general, V_E will contain far fewer points than V. Efficient convex hull algorithms are discussed by Eddy [5], Green and Silverman [12] and Bentley and Shamos [3]. The next stage is to step through possible test support sets S_m for ME (V_E) until a correct one is found. We shall presume for the moment that we can reconstruct a minimum ellipse from a support set. The details of this procedure are given in the next section. Note that it follows at once from the theorem that every 3-point set is the support set of its minimum covering ellipse, and so we can find ME (T) for any 3-point set T. The algorithm then proceeds as follows.

Step 1. Using the algorithm PEEL of [12], find the convex hull of V and eliminate all but the extreme points V_E from further consideration.

Step 2. Choose a 3-point subset S_0 of V_E. Find ME (S_0), which must have support set S_0.

Step 3 (*the iterative step*). If the mth test support set is S_m, check whether each of the points of V_E lies outside ME (S_m). If no point lies outside, then S_m is a support set of ME (V); exit. Otherwise choose v in V_E outside ME (S_m) and set $S_m^* = S_m \cup v$.

In our implementation, we choose the v which maximizes the defining quadratic form $(x-c)^T N(x-c)$ of ME (S_m), since this v is in a natural sense "furthest" outside ME (S_m).

Step 4. Find a support set $S_m^\dagger$ of ME (S_m^*). Note first that $S_m^\dagger$ must include v. To prove this, suppose otherwise; then $S_m^\dagger$ will be a subset of S_m and hence ME $(S_m^\dagger) =$ ME (S_m), since ME $(S_m^\dagger) =$ ME (S_m^*) covers S_m and ME (S_m) covers $S_m^\dagger$. Therefore by construction ME $(S_m^\dagger)$ will not contain v, a contradiction. Note also that S_m^* has at most 6 points. Find $S_m^\dagger$ by exhaustive search, as follows.

4a. For each 3-point subset T_3 of S_m^* containing v, find ME (T_3). If ME (T_3) covers S_m^* then $S_m^\dagger = T_3$; go to step 5. Otherwise flag any 4- or 5-point subsets which have minimum ellipses with support set T_3 and hence cannot be possible support sets.

4b. Repeat 4a for unflagged 4-point subsets T_4 of S_m^* containing v, flagging any 5-points subsets covered by ME (T_4).

4c. Repeat 4a for unflagged 5-point subsets of S_m^* containing v.

Step 5. Set $S_{m+1} = S_m^\dagger$. Go to step 3.

Notice that the flagging in step 4 ensures that the subsets T whose minimum ellipses are considered are precisely all those subsets of S_m^* satisfying condition (C1), and hence one of them must be $S_m^\dagger$. This also ensures that each unflagged T is the support set of *some* minimum ellipse, ME (T), so that the procedure described in the next section can be used to find ME (T).

In the flagging, subsets are characterized by their complements in S_m^*, and so the storage requirement in Fortran is at most two LOGICAL arrays of dimension 6 and (5, 6).

Note also that ME $(S_{m+1}) =$ ME $(S_m^*) =$ ME $(S_m \cup v)$, and therefore $S_m \subseteq$ ME(S_{m+1}). Since v is strictly outside ME (S_m), ME $(S_{m+1}) \neq$ ME (S_m), and therefore ME (S_{m+1}) has strictly greater area than ME (S_m). Thus no support set S_m is ever considered twice. Since there are only finitely many possible support sets, the algorithm must terminate.

It is clear that the algorithm will terminate quickly if the initial support set S_0 corresponds to a large ellipse, in some sense. In our implementation the points of S_0 are

chosen to be as equally spaced as possible round the convex hull. This gives a heuristically sensible starting set which is invariant under some affine transformations.

5. Reconstructing an ellipse from its support set. There are three cases to consider, where the support set S has 3, 4 and 5 points respectively. Let the points of S be $s_1, s_2, \cdots, s_k$. If S has 5 points, there is only one conic section passing through the points of S; this must be an ellipse by the definition of support set, and can be found by solving the system

$$s_i^T \begin{bmatrix} 1 & h \\ h & b \end{bmatrix} s_i + 2(f\ g)s_i + c = 0, \qquad i = 1, \cdots, 5$$

of five linear equations in the five unknowns b, c, f, g and h.

If S consists of 3 points it follows by consideration of the dual problem (see [21]) that ME (S) is given by

$$(x-c)^T N(x-c) = 2,$$

where $c = \frac{1}{3}(s_1+s_2+s_3)$ and

$$N^{-1} = \frac{1}{3}\sum_{i=1}^{3}(s_i-c)(s_i-c)^T.$$

The case where S has 4 points is dealt with by a geometrical approach. A series of unitary affine transformations is applied to the plane, as follows. Let Q be the convex quadrilateral with vertices the points of S. The steps are illustrated in Fig. 1, which depicts the diagonals of Q.

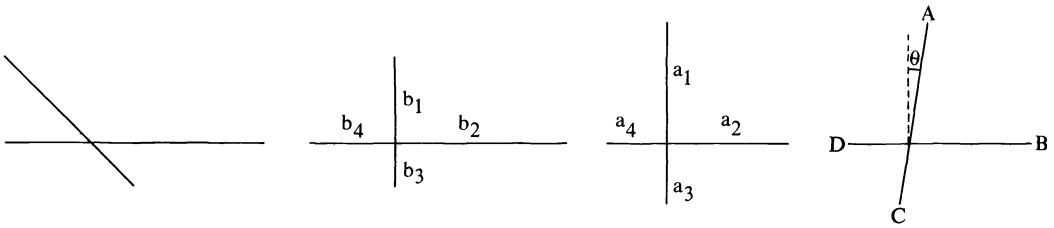


FIG. 1. *Steps in the affine transformation of a convex quadrilateral.*

(a) Rotate so that a diagonal of Q is parallel with the x-axis.

(b) Shear parallel to the x-axis to make the diagonals perpendicular.

(c) Keeping the diagonals perpendicular, make Q cyclic by $x \mapsto d^{-1/4}x$ and $y \mapsto d^{1/4}y$, where $d = b_1b_3/b_2b_4$.

(d) Apply a transformation with matrix

$$D_\theta = \begin{bmatrix} (\cos\theta)^{-1/2} & \sin\theta\,(\cos\theta)^{-1/2} \\ 0 & (\cos\theta)^{1/2} \end{bmatrix};$$

this first "swings" the diagonal AC through an angle θ preserving the lengths a_1, a_2, a_3, a_4 and hence the cyclic nature of $ABCD$, and then rescales by a factor of $(\cos\theta)^{1/2}$ to make the determinant equal to 1. The angle θ is chosen, as described below, to minimize the area of the covering circle.

(e) Find the circle passing through the transformed support set; reversal of steps (d), (c), (b) and (a) transforms this circle into ME (S), as will be shown next.

To justify this procedure, we prove an easy lemma.

LEMMA. *Suppose S is a convex 4-point set. Given any ellipse E through the points of S, there exists a value of θ such that the transformations* (a) *to* (d) *above will transform E to a circle with the same area as E.*

Proof. The fact that the circle has the same area as E follows immediately from the unitary nature of the transformations. After affine transformations (a), (b) and (c), suppose that E has equation

$$(x-c)^T\begin{bmatrix} a & h \\ h & b \end{bmatrix}(x-c)=1.$$

It is easily shown that then applying D_θ with $\theta=\tan^{-1}(h/a)$ transforms E to a conic E_θ with axes parallel to the coordinate axes. Such a conic is uniquely determined by four of its points, and, since the transformed points of S are concyclic, E_θ must be the circle passing through them, completing the proof.

Since every ellipse through the points of S corresponds to some circle of this kind, the minimum ellipse must, because of the area-preserving property, correspond to the circle of minimum area obtainable by transformations (a), (b), (c) and (d). Thus the procedure described above will yield ME (S).

It remains, finally, to determine the optimal value of θ in step (d) above. Consider Fig. 2 and let R be the circum-radius of $ABCD$. Here $\lambda=(\cos\theta)^{1/2}$.

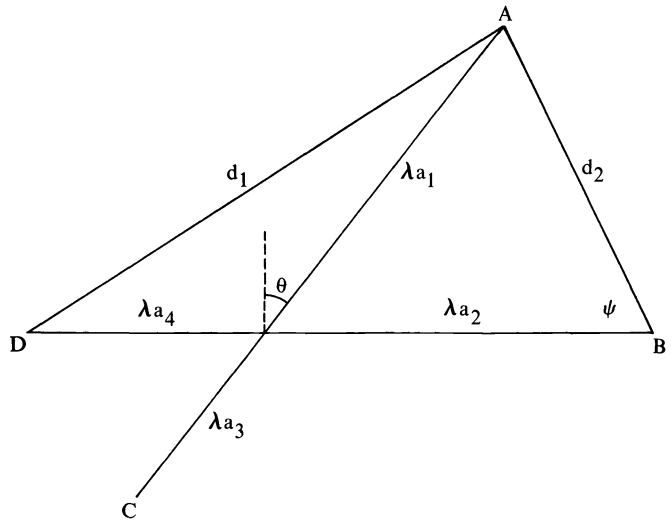


FIG. 2. *Finding the optimal "swing".*

Then $2R=d_1/\sin\psi$, and since $\lambda a_1/\sin\psi=d_2/\cos\theta$, it follows that

$$2R=\frac{d_1d_2}{a_1(\cos\theta)^{3/2}},$$

and hence that the area of the circle is proportional to

$$\frac{d_1^2d_2^2}{\cos^3\theta}=\frac{(a_1^2+a_4^2+2a_1a_4\sin\theta)(a_1^2+a_2^2-2a_1a_2\sin\theta)}{\cos^3\theta}. \tag{1}$$

The condition for stationarity in (1) with respect to θ can be written as

$$2a_1(a_4d_2^2-a_2d_1^2)(1-\sin^2\theta)+3d_1^2d_2^2\sin\theta=0. \tag{2}$$

If d_1^2 and d_2^2 are written in terms of the a_i and $\sin\theta$, then (2) takes the form $f(\sin\theta)=0$, where $f(t)$ is a cubic with leading term $-4a_1^2a_2a_4t^3$. Since (1) tends to infinity as $\sin\theta\to\pm1$, the minimizing value of θ is strictly between -1 and $+1$. Further, $f(-1)\leqq0$

and $f(1) \geqq 0$, while $f(t) \to \infty$ as $t \to -\infty$ and $f(t) \to -\infty$ as $t \to \infty$. It follows that f has a zero in each of $(-\infty, -1]$, $(-1, 1)$ and $[1, \infty)$, and that we are seeking the middle root.

This can be found explicitly, albeit using trigonometric functions; see, for example, [13, p. 23].

6. Theoretical computational complexity. It is straightforward to obtain upper bounds for the computational complexity of the minimum ellipse problem as the number of points in V increases. Suppose V contains n points, of which m lie on the convex hull. It was shown by Graham [11] that the convex hull can be found in $O(n \log n)$ operations. Once the convex hull has been found, the total possible number of support sets that can be considered is $O(m^5)$, because each support set contains at most 5 points and no support set is ever considered twice.

Since at most m points have to be checked for coverage by the current test ellipse, it takes $O(m)$ operations to check the optimality of each support set, while the number of operations involved in updating to the next support set is bounded independently of n and m. Thus, once the convex hull is found, it takes at most $O(m^6)$ operations to find the ellipse. So the total time required is $O(n \log n + m^6)$.

In many cases m increases slowly with n. For example, Raynaud [15] showed that, when the points are drawn from a bivariate normal distribution, $m = O[(\log n)^{1/2}]$. Bentley et al. [2] showed that the expected value of m is $O(\log n)$ if the points are drawn from a continuous distribution with independent components. It follows from Devroye's [4] extensions of this result that $E(m^6)$ has polynomial order in $\log n$ under the same conditions, and hence the expected time to find the minimum covering ellipse once the convex hull has been found is $o(n)$. Furthermore, the expected time to find the convex hull is linear in n under conditions milder than these; see Bentley and Shamos [3] for a theoretical discussion and Green and Silverman [12] for a practical demonstration. Thus the time to find the convex hull will dominate asymptotically and the total expected time to find the minimum covering ellipse will be $O(n)$ under the assumption that the points are independently drawn from a distribution with independent components.

It should be stressed that these calculations and in particular the $O(m^6)$ term provide only a crude description of the behavior of the algorithm in practice; even for moderately large data sets, only a very small number of support sets will be considered. The empirical results given in § 7 below show that the running time of the algorithm increases far more slowly than the number of points, even for quite large values of n.

7. Empirical results and numerical considerations. The method was implemented in a Fortran program, available on request from the authors, on a Honeywell Series 60 Level 68/DPS machine. Results are given for two different point patterns. In the first, pseudo-random samples of different sizes were generated from a bivariate normal distribution, with 100 replications for each sample size. In the second, the pseudo-random samples were generated from the uniform distribution on the annulus whose inner and outer radii are 0.95 and 1. The latter pattern was expected to test the efficiency of the routine severely because the convex hulls would typically have many extreme points, all of which lie approximately on a circle. One difficulty which might then arise is when 6 or more points lie exactly or almost exactly on an ellipse. In that case, as a result of rounding errors, the addition of a further point to a 5-point support set S may fail because none of the test ellipses covers all 6 points. From a practical point of view, however, this would correspond to the algorithm having succeeded, subject to rounding errors. In any case this difficulty was not encountered with this data model.

The average results reported in Tables 1 and 2 indicate the feasibility of the algorithm as far as time is concerned, and also show that the theoretical work of § 6 above is unrealistically pessimistic, and by a long way. Even for the case of points on the annulus (Table 2) the computation is very quick.

TABLE 1

Results for bivariate normal samples, 100 *independent replications for each sample size. Timings on Honeywell Series* 60 *Level* 68/DPS *machine.*

	Sample size				
	10	30	100	300	1000
Average number of extreme points	5.5	7.2	9.0	10.3	12.0
Average number of iterations	2.9	3.7	4.2	4.8	4.9
Average cpu time (secs)	0.05	0.11	0.18	0.31	0.53
Standard deviation of cpu time (secs)	0.09	0.12	0.18	0.20	0.23

TABLE 2.

Results for samples from the annulus $0.95 \leqq radius \leqq 1$, 100 *independent replications for each sample size. Timings on Honeywell Series* 60 *Level* 68/DPS *machine.*

	Sample size				
	10	30	100	300	1000
Average number of extreme points	9.1	20.5	33.6	48.6	72.7
Average number of iterations	4.6	6.6	7.5	8.3	8.6
Average cpu time (secs)	0.27	0.67	0.95	1.17	1.52
Standard deviation of cpu time (secs)	0.21	0.23	0.22	0.25	0.24

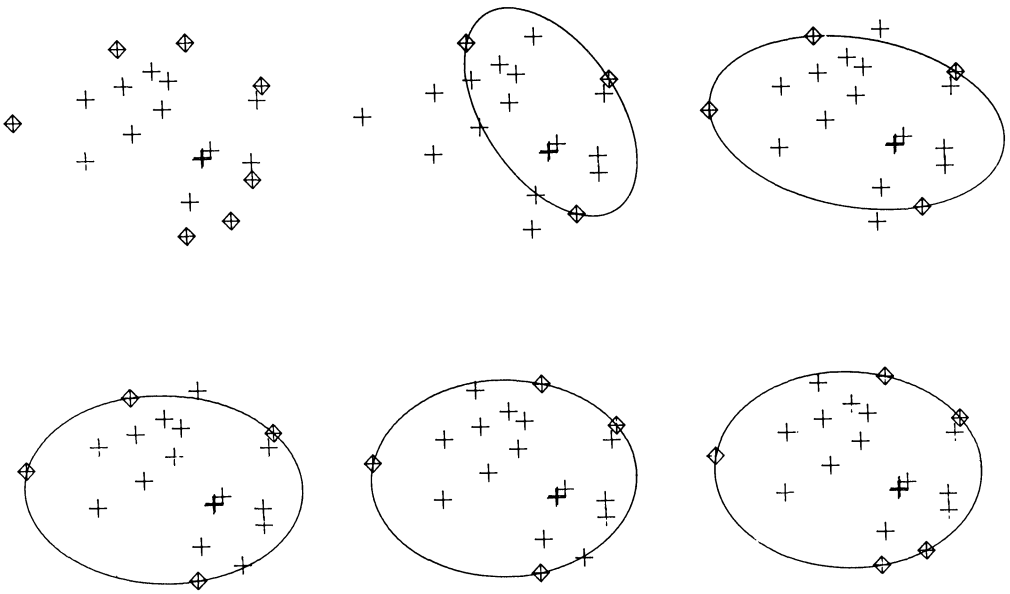


FIG. 3. *Steps in finding the minimum ellipse of a* 20*-point set.*

Fig. 3 shows the progress of the algorithm in a particular example consisting of 20 points. First the extreme points are marked, and then successive support sets and their minimum ellipses are shown. The convex hull has 7 extreme points, and 5 iterations were necessary to find the minimum ellipse. This example was chosen to be somewhat more laborious than is typical for small point sets.

8. Relation to other problems. The two-dimensional problem can be dealt with in this exact way because 5-point ellipses and optimal 3- and 4-point ellipses can be computed exactly. In higher dimensions there is, as yet, no evidence that explicit results are possible unless the optimal support contains $k+1$ or $k(k+3)/2$ points. In these circumstances it seems at present that iterative algorithms offer the only feasible approach; see [22].

Another, related, problem is that of finding the minimum central ellipsoid MCE (v) covering a point set V, in other words, constraining c to be zero in the definition of the ellipsoid. This problem has a dual relationship with the D-optimal experimental design problem; see Fedorov [10] for details of D-optimality and Silvey [19] and Sibson [18] for the duality. An algorithm for finding MCE (V) in the plane case would be similar to that of § 4, but much simpler because only 2 or 3 points are necessary to define the support of a minimum central ellipse. In k dimensions the bounds are k and $k(k+1)/2$; see [10]. Furthermore, the optimal 2-point central ellipse is of the form

$$v^T(v_1v_1^T+v_2v_2^T)^{-1}v=1,$$

where v_1, v_2 are the two support points, corresponding to equal design weights on v_1 and v_2. For three points, the ellipse can be found in the form

$$v^TNv=r \quad \text{with } N_{11}=1.$$

Computation of the convex hull of $V\subset\mathbb{R}^2$ is clearly a sensible first stage in attacking the minimum central ellipse problem for V. It may be even more efficient to construct the convex hull of $V\cup V_R$ or $V\cup\{0\}$, where V_R denotes the reflection of V in the origin.

The use of a convex hull algorithm as a preliminary stage in solving the minimum circle problem was suggested by Bentley and Shamos [3] who showed that under mild conditions the minimum circle can be found in linear expected time; this betters the more generally applicable $O(n \log n)$ attained by the method of Shamos and Hoey [17].

Acknowledgments. The authors acknowledge gratefully the facilities provided by the Social Science Research Council Spatial Data Project at the University of Bath; they would like to thank A. Bowyer for his help with the computing and M. A. Sabin and R. Sibson for their comments. The helpful remarks of the referees, especially in relation to § 6, were much appreciated.

REFERENCES

[1] V. BARNETT, *The ordering of multivariate data*, J. Roy. Statist. Soc. Ser. A., 139 (1976), pp. 318–354.

[2] J. L. BENTLEY, H. T. KUNG, M. SCHKOLNICK AND C. D. THOMPSON, *On the average number of maxima in a set of vectors and applications*, J. Assoc. Comput. Mach., 25 (1978), pp. 536–543.

[3] J. L. BENTLEY AND M. I. SHAMOS, *Divide and conquer for linear expected time*, Inform. Process. Lett., 7 (1978), pp. 87–91.

[4] L. DEVROYE, *A note on finding convex hulls via maximal vectors*, Inform. Process. Lett., 11 (1980), pp. 53–56.

[5] W. F. EDDY, *A new convex hull algorithm for planar sets*, ACM Trans. Math. Software, 3 (1977), pp. 398–403.
[6] J. ELZINGA AND D. HEARN, *The minimum covering sphere problem*, Management Sci., 19 (1972), pp. 96–104.
[7] ———, *Geometrical solutions for some minimax location problems*, Transport Sci., 6 (1972), pp. 379–394.
[8] ———, *The minimum sphere covering a convex polyhedron*, Nav. Res. Logist. Q., 21 (1974), pp. 715–718.
[9] J. ELZINGA, D. HEARN AND W. D. RANDOLPH, *Minimax multifacility location with Euclidean distances*, Transport Sci., 10 (1976), pp. 321–336.
[10] V. V. FEDOROV, *Theory of Optimal Experiments*, Academic Press, New York, 1972.
[11] R. L. GRAHAM, *An efficient algorithm for determining the convex hull of a finite planar set*, Inform. Process. Lett., 1 (1972), pp. 132–133.
[12] P. J. GREEN AND B. W. SILVERMAN, *Constructing the convex hull of a set of points in the plane*, Computer J. 22 (1979), pp. 262–266.
[13] G. A. KORN AND T. M. KORN, *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill, New York, 1968.
[14] K. P. K. NAIR AND R. CHANDRASEKAR, *Optimal location of a single service centre of certain types*, Nav. Res. Logist. Q., 18 (1971), pp. 503–510.
[15] H. RAYNAUD, *Sur l'enveloppe convexe des nuages de points aléatoires dans* $\mathbb{R}^n$, J. Appl. Prob., 7 (1970), pp. 35–45.
[16] J. B. ROSEN, *Pattern recognition by convex programming*, J. Math. Anal. Appl., 10 (1965), pp. 123–134.
[17] M. I. SHAMOS AND D. HOEY, *Closest point problems*, 16th Symposium on Foundations of Computer Science, IEEE Conference Record, 1975, pp. 151–162.
[18] R. SIBSON, *Discussion* of a paper by H. P. Wynn, J. Roy. Statist. Soc. Ser. B., 34 (1972), pp. 181–183.
[19] S. D. SILVEY, *Discussion* of a paper by H. P. Wynn, J. Roy. Statist. Soc. Ser. B, 34 (1972), pp. 174–175.
[20] S. D. SILVEY AND D. M. TITTERINGTON, *A geometric approach to optimal design theory*, Biometrika, 60 (1973), pp. 21–32.
[21] D. M. TITTERINGTON, *Optimal design: Some geometrical aspects of D-optimality*, Biometrika, 62 (1975), pp. 313–320.
[22] ———, *Estimation of correlation coefficients by ellipsoidal trimming*, J. Roy. Statist. Soc. Ser. C., 27 (1978), pp. 227–234.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/81/0104-0002 $01.00/0

A FINITE ELEMENT-CAPACITANCE MATRIX METHOD FOR THE NEUMANN PROBLEM FOR LAPLACE'S EQUATION*

WŁODZIMIERZ PROSKUROWSKI† AND OLOF WIDLUND‡

Abstract. Capacitance matrix methods extend the usefulness of fast Poisson solvers to an important family of elliptic problems on arbitrary bounded regions. New algorithms of this kind are introduced for finite element approximations. The use of these methods for the Neumann problem for Laplace's equation and important issues of implementation are discussed in some detail. It is shown that the new methods offer considerable advantages in comparison with finite difference–capacitance matrix methods previously employed.

Key words. capacitance matrix methods, finite element methods, fast Poisson solvers, potential theory, Fredholm integral equations

1. Introduction. In this paper, we explore the use of capacitance matrix techniques for the solution of the large, sparse and very special linear systems of equations which arise when Laplace's equation is discretized by finite element methods. Capacitance matrix methods extend the usefulness of fast Poisson solvers to problems on arbitrary bounded regions, and certain variants of these methods are almost optimal in that the number of arithmetic operations per mesh point required to achieve a given accuracy grows only in proportion to the logarithm of the number of unknowns. The efficiency of such methods has previously been confirmed by several series of numerical experiments with finite difference schemes in two and three dimensions; see O'Leary and Widlund [36], Proskurowski [39], [40] and Proskurowski and Widlund [41]. The almost optimality of certain of these algorithms has also been rigorously established in a number of cases by Astrakhantsev [1], [2] and Shieh [42], [43]. Earlier work on algorithmic design was done by Buzbee and Dorr [8], Buzbee, Dorr, George and Golub [9], George [19], Hockney [23], [25], and Martin [33]. These techniques, known in the Soviet literature as methods of fictitious domains, have also been studied by Il'in and Korotkevich [26], Korneev [29], Kuznetsov and Matsokin [30]; see also the references given in those papers. For a discussion of some of the work prior to 1976, see Proskurowski and Widlund [41]. We also note that capacitance matrix methods can be designed so that they are very suitable for problems with very many degrees of freedom. For a discussion of programs which use much fewer words of storage than degrees of freedom, see O'Leary and Widlund [36] and Proskurowski [39].

Fast Poisson solvers, see for example Banegas [3], Bank [4], [5], Bank and Rose [6], Buneman [7], Buzbee, Golub and Nielson [10], Fischer, Golub, Hald, Leiva and Widlund [16], Hockney [22], [24], and Swartztrauber and Sweet [45], are used as important subroutines in capacitance matrix programs. These powerful algorithms are limited with regard to the choice of meshes and the form of the operators, and work essentially only if the region and the differential operator allow the separation of the variables. Capacitance matrix methods have to our knowledge been developed exclusively for finite difference schemes with uniform meshes in the interior of the

* Received by the editors April 22, 1980.

† Department of Mathematics, University of Southern California, Los Angeles, California 90007. The work of this author was supported by the U.S. Department of Energy under contract No. W-7405-ENG-48 while he was at the Lawrence Berkeley Laboratory.

‡ Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. The work of this author was supported by the U.S. Department of Energy under contract EY-76-C-02-3077 at the Courant Mathematics and Computing Laboratory, and by the National Science Foundation under contract MCS-79-02882.

region in order to match completely the corresponding linear algebraic equations for the fast Poisson solver. In the experiments reported in this paper, we similarly work with regular, right-triangular, finite elements in the interior, while the remaining skin area, which is of a width on the order of the mesh size h, is partitioned into irregular triangles; see §§ 3 and 5. The implementation of our ideas has also been limited to a standard model problem, the Neumann problem for Laplace's operator discretized by using piecewise linear finite element functions. Such finite element problems can equally well be considered finite difference schemes, and in this work, we have indeed tried to combine the most attractive features of finite differences and finite elements.

The introduction of a finite element framework offers very considerable advantages compared to the finite difference methods used previously in capacitance matrix work. As shown, for example, in Ciarlet [11] and Strang and Fix [44], the use of finite elements greatly simplifies the design of accurate methods for general regions. New tools for the analysis of the efficiency of the methods become available within a variational framework and, as we shall see, it also enables us to design more efficient iterative methods for solving the capacitance matrix equations than those previously employed. We note that fast Poisson solvers, based for example on a fast Fourier transform, can be developed for higher order finite element approximations of constant coefficient elliptic problems on rectangular regions subdivided into uniform triangles or quadrilaterals. Our algorithm can therefore be extended to a large family of finite element methods, including those of isoparametric type.

Choosing the triangulation of the region as we have done in our program decreases the cost of the calculation of the stiffness matrix compared to a standard finite element method, since only the contributions to the stiffness matrix from the irregular triangles in the skin region next to the boundary need to be computed and stored; see § 5. At the cost of calculating more elements of the stiffness matrix, some additional storage and a modest increase in the arithmetic per iteration, the number of irregular triangles can be increased and more general and flexible triangulations allowed. Methods further extending those considered in this paper can be obtained by using a finite element method on a triangulation with vertices which are the images under a one-to-one smooth mapping of those of a triangulation as previously described. Similarly, any uniformly elliptic second order Neumann problem can be solved by preconditioned conjugate gradient methods in which the restriction of the solution operator of the discrete Laplace operator on a simple larger region to a suitable subset of mesh points is used as a preconditioning operator. See, for example, Concus, Golub and O'Leary [12] for a discussion of preconditioned conjugate gradient methods. The main theoretical difficulties arise when a general region is considered, while the extension of the proof of almost optimality to cases with variable coefficients and the general meshes described above can be carried out as in the paper just cited.

We have previously written extensively on the connection between classical potential theory and capacitance matrix methods; see, for example, O'Leary and Widlund [36] and Proskurowski and Widlund [41]. We nevertheless discuss this subject again in § 2, since the presentation differs in several important respects from that of our previous papers. We explain in particular how the classical integral operator for the Neumann problem, which is nonsymmetric, can be symmetrized, a fact which enables us to design more efficient conjugate gradient methods; see § 4. We are thus able to avoid the least squares formulation of the capacitance matrix equation previously employed, and limit the number of fast Poisson steps to one per iteration step.

In the third section, a brief review of the finite element method and our capacitance matrix method is given. In particular, we show how to multiply the capacitance matrix

by an arbitrary vector in an efficient way using sparsity. A suitable variant of the standard conjugate gradient method is derived in § 4, where we also show that this iterative method converges rapidly when applied to the integral equation which is the continuous analogue of the capacitance matrix equation.

In the fifth section, we show how a suitable triangulation of a bounded plane region can be generated from similar type of information on the boundary which we have used in our previous work on finite differences. We also discuss how the nonredundant information on the stiffness matrix and the weights for the data is obtained. In the last section, some results of numerical experiments are reported. We conclude that our finite element program performs with an efficiency which is even more favorable than that of our finite difference codes and that an accurate solution can be obtained at an expense comparable to a modest number of calls of the fast Poisson solver subroutine.

2. The continuous problem, a variational formulation and potential theory. We consider the numerical solution of the Neumann problem

$$\begin{aligned} -\Delta u + cu &= f \quad \text{in } \Omega, \\ \frac{\partial u}{\partial n} &= g \quad \text{on } \Gamma, \end{aligned} \tag{2.1}$$

where the region Ω is a bounded open subset of the plane with a sufficiently smooth boundary Γ. The functions f and g are given and sufficiently smooth, c is a nonnegative real constant, Δ the Laplace operator and $\partial/\partial n$ the outward normal derivative. This problem, which often serves as a model problem in the calculus of variations, has a unique solution for any $c > 0$; see, for example, Ciarlet [11]. A solution, unique up to a constant, exists when $c = 0$ if and only if

$$\int_\Omega f\,dx + \int_\Gamma g\,ds = 0, \tag{2.2}$$

which can be interpreted as the condition that the heat liberated from sources in Ω must be balanced by the heat flow across the boundary Γ.

By using a Green's formula, (2.1) can be written in the variational form

$$\int_\Omega \nabla u\, \nabla v\, dx + c\int_\Omega uv\, dx = \int_\Omega fv\, dx + \int_\Gamma gv\, ds. \tag{2.3}$$

This formula has a good meaning for any $v \in H^1(\Omega)$, where

$$H^1(\Omega) = \left\{ w \in L^2(\Omega) : \|w\|_{H^1(\Omega)} = \left(\int_\Omega (\nabla w)^2\, dx + \int_\Omega w^2\, dx \right)^{1/2} < \infty \right\}.$$

Under the conditions stated above, Lax-Milgram's lemma shows that the problem (2.3) has a solution in $H^1(\Omega)$. This solution can be shown to solve (2.1) as well. The variational formulation lends itself to the derivation of finite element and other Galerkin methods; see discussion in § 3.

An alternative, classical method to solve the Neumann problem is provided by the potential theory developed by C. Neumann, Poincaré, Fredholm and others; see for example Courant and Hilbert [14] or Garabedian [18]. In our discussion we specialize to the case of $c = 0$, and by subtracting a particular solution of the Poisson equation we can also reduce our problem to the case when $f \equiv 0$. Following the presentation of Nedelec [34], we state certain identities which provide useful insight. We note that $(1/2\pi) \log|x-y|$ is a fundamental solution of the operator $-\Delta$, where $|x-y|$ is the

Euclidean distance between the points x and y. The formulas (2.4) and (2.5) below are derived by using a Green's formula. We denote by Ω' the open set which is the complement of the closure of Ω.

Let u be harmonic in Ω as well as in Ω', and sufficiently smooth so that the limits u_{int}, u_{ext}, $\partial u/\partial n_{\text{int}}$ and $\partial u/\partial n_{\text{ext}}$ exist, when Γ is approached from Ω and Ω' respectively. Assume further that $|u(x)| = o(1/|x|)$ and $|\nabla u(x)| = O(1/|x|)^2$, for large x. Then

$$u(x) = -\frac{1}{2\pi}\int_\Gamma \left[\frac{\partial u}{\partial n}\right] \log|x-y|\, ds(y)$$

(2.4a)

$$+\frac{1}{2\pi}\int_\Gamma [u]\frac{\partial}{\partial n_y}\log|x-y|\, ds(y) \quad \forall x \in \Omega \cup \Omega'.$$

Here, $[\phi] = \phi_{\text{int}} - \phi_{\text{ext}}$. For $x \in \Gamma$, the left-hand side of (2.4a) is replaced by

$$\frac{(u_{\text{int}} + u_{\text{ext}})}{2}. \tag{2.4b}$$

In the classical theory, a Fredholm integral equation of the second kind is obtained by seeking a harmonic function with the properties just listed for which $[u] = 0$. By (2.4), the solution is thus given in the form of a single layer potential. By using techniques similar to those used to derive (2.4), we obtain for such a function and $x \in \Gamma$,

$$\frac{1}{2}\left(\frac{\partial u}{\partial n_{\text{int}}} + \frac{\partial u}{\partial n_{\text{ext}}}\right) = -\frac{1}{2\pi}\int_\Gamma \left[\frac{\partial u}{\partial n}\right]\frac{\partial}{\partial n_x}\cdot \log|x-y|\, ds(y). \tag{2.5}$$

We introduce the integral operator K by

$$Kq = \frac{1}{\pi}\int_\Gamma q\frac{\partial}{\partial n_x}\log|x-y|\, ds(y). \tag{2.6}$$

It is well known that this operator is compact in $L^2(\Gamma)$. We now obtain straightforwardly from (2.5) the Fredholm integral equation of the second kind,

$$(I-K)\left[\frac{\partial u}{\partial n}\right] = 2\frac{\partial u}{\partial n_{\text{int}}}. \tag{2.7}$$

The operator $I-K$ has a one-dimensional null space, and by the Fredholm theory, (2.7) is solvable if and only if the right-hand side is orthogonal in $L^2(\Gamma)$ to the left eigenfunction associated with the zero eigenvalue. In this case, this means that the right-hand side of (2.7) should have a zero mean value, a condition equivalent to (2.2). Our capacitance matrix equations are discrete analogues of (2.7) with the role of the fundamental solution played by the operator defined by a fast Poisson solver. This analogy suggests that the capacitance matrices should have attractive spectral properties to a large extent inherited from (2.7); see further Astrakhantsev [1], O'Leary and Widlund [36], Proskurowski and Widlund [41] and Shieh [42].

The operator K is nonsymmetric except for very special choices of Γ and so are our capacitance matrices. In the continuous Neumann case, we can obtain a symmetric problem by making $[\partial u/\partial n] = 0$ rather than $[u] = 0$. The resulting equation is a Fredholm integral equation of the first kind, and finite element methods for its solution have been designed and studied by Nedelec [34]. Because of less favorable spectral distributions, these discrete problems and capacitance matrix equations similar in design cannot be solved in optimal time by the conjugate gradient method. See

Proskurowski and Widlund [41] for a detailed discussion. We note, however, that the choice of a solution such that $[\partial u/\partial n] = 0$, i.e., a so called dipole potential, is appropriate for a Dirichlet problem. Equation (2.4) then leads straightforwardly to the equation considered in the famous paper by Fredholm [17],

$$(I + K^T)[u] = 2u_{\text{int}}.$$

Here K^T is the transpose of the operator introduced by (2.6).

Returning to the Neumann case, we show that a symmetric operator can be found which symmetrizes (2.7) (see Pekker [38]) and which enables us to use a conjugate gradient method closely related to the standard one. Denote by $V\rho(x)$ the potential resulting from a charge distribution ρ concentrated on the boundary Γ,

$$V\rho(x) = -\frac{1}{2\pi}\int_\Gamma \rho(y) \log|x-y|\, ds(y).$$

By using a Green's formula, we can show that $\int_\Gamma [\partial u/\partial n]\, ds = 0$, if u is harmonic in Ω and Ω'. It is therefore natural to consider only ρ with a zero mean value. The mapping V can be extended to the space $\mathring{H}^{-1/2}(\Gamma)$, where $\mathring{H}^{-1/2}(\Gamma)$ is the subspace of $H^{-1/2}(\Gamma)$ of elements of zero mean value, $H^{-1/2}(\Gamma)$ the space dual to $H^{1/2}(\Gamma)$, and $H^{1/2}(\Gamma)$ the space of traces of $H^1(\Omega)$; see Nedelec [34] or Nedelec and Planchard [35]. Denote by $Q\rho$ the restriction of $V\rho$ to Γ. For $\rho \in \mathring{H}^{-1/2}(\Gamma)$ the restriction of $V\rho$ to Ω belongs to $H^1(\Omega)$, and thus its trace $Q\rho \in H^{1/2}(\Gamma)$. Let further $\langle \nu, \mu \rangle$ be defined as the $L^2(\Gamma)$ inner product of ν and μ in $L^2(\Gamma)$. This bilinear form is extended in the normal way to $H^{1/2}(\Gamma) \times H^{-1/2}(\Gamma)$. Consider two single layer potentials $u = V\rho$ and $v = V\mu$. By formula (2.4a), $\rho = [\partial u/\partial n]$ and $\mu = [\partial v/\partial n]$. Thus

$$\langle Qu, \rho \rangle = \left\langle v, \left[\frac{\partial u}{\partial n}\right] \right\rangle.$$

By a Green's formula, our definition of the normal direction and the fact that u and v are harmonic,

$$\begin{aligned}\langle Q\mu, \rho \rangle &= \int_\Omega \nabla v\, \nabla u\, dx + \int_{\Omega'} \nabla v\, \nabla u\, dx \\ &= \int_{R^2} \nabla v\, \nabla u\, dx = \int_{R^2} \nabla u\, \nabla v\, dx = \langle Q\rho, \mu \rangle.\end{aligned}$$

Thus, Q is a symmetric, positive definite operator on $\mathring{H}^{-1/2}(\Gamma)$. By using (2.7), we similarly find

$$\langle Q\mu, (I-K)\rho \rangle = 2\left\langle Q\mu, \frac{\partial u}{\partial n_{\text{int}}} \right\rangle = 2\int_\Omega \nabla v\, \nabla u\, dx = \langle Q(I-K)\mu, \rho \rangle, \tag{2.8}$$

which proves that QK is symmetric. It is also easy to conclude that the eigenvalues of K are real and less than or equal to one in absolute value, since

$$\frac{\langle Q\mu, (I-K)\mu \rangle}{\langle Q\mu, \mu \rangle} = 2\int_\Omega (\nabla v)^2\, dx \Big/ \left(\int_\Omega (\nabla v)^2\, dx + \int_{\Omega'} (\nabla v)^2\, dx\right);$$

see also Kellogg [31, p. 309.]

Equation (2.8) can be considered as a reformulation of (2.3). We need only note that, in this context, it is legitimate to restrict the choice of test functions to the subspace of $H^1(\Omega)$ of single layer potentials, since we know from potential theory that the solution sought can be represented in such a form.

3. A finite element–capacitance matrix method. In this section, we review a few facts on a finite element method and introduce our capacitance matrix equation. We discuss only a very simple finite element approximation, but, as we have previously noted, these methods can be extended quite straightforwardly to finite element methods of higher accuracy. A discussion of the details of the triangulation algorithm which has been used in our code is given in § 5. To simplify, we discuss simply connected regions only.

The great majority of our triangles will be regular, i.e., right triangles with sides h, h and $\sqrt{2}\,h$. When triangulating the region, we begin by approximating the boundary Γ to within $O(h^2)$ by a polygon Γ^h. This can be done, for example, by choosing a number of points on Γ, each at a distance of the order of h from its nearest neighbor, and connecting consecutive members of this point set by line segments. The basic strategy is to cover the resulting polygonal region Ω^h by as many regular triangles as possible, leaving a skin area of a width of the order of h which is bounded by the Γ^h and a nearby polygon. The skin area is then partitioned into triangles by connecting suitable pairs of points on the two polygons. In certain cases, such a procedure leads to very thin triangles As shown by Jamet [27], this does not decrease the rate of convergence of the finite element method if the solution is regular enough. We have found, however, in our experiments, that the performance of our iterative method suffers when some of the triangles are very thin. To avoid this problem, we widen the skin area by adding some suitably chosen regular triangles to it. We can then triangulate the resulting area using as vertices only points on the two polygons, thus avoiding the use of any very thin triangles; see further, § 5.

Piecewise linear continuous functions are now introduced in the standard way. They can, for example, be represented in terms of their values at the vertices of the triangles. The corresponding linear space is denoted by V^h; it is a subspace of $H^1(\Omega^h)$. A natural basis for the space V^h is given by the elements which take the value one at one vertex and vanish at all others. A finite element approximation is then obtained by replacing u by $u^h \in V^h$ in (2.3) and by selecting the test functions v from V^h only. If necessary, the integrals in the right-hand side of (2.3) are replaced by suitable numerical quadrature formulas. In its basic form, this finite element method is second order accurate in L^2 and first order accurate in H^1. It is known that the rate of convergence in H^1 is unaffected by changing the boundary from Γ into Γ^h, and the effects of the numerical quadratures are also well understood; see Ciarlet [11] and Strang and Fix [44].

The discrete problem has the form of a linear system of algebraic equations with a positive semidefinite, sparse stiffness matrix A, which for $c = 0$ has a one-dimensional null space of constants. As pointed out by Courant [13], the finite element approximation of the Laplace operator at a vertex, which with its neighbors is located on a uniform square mesh, is the standard five-point formula. This is independent of the orientation of the triangles and, for this reason, no specific choice of triangulation of the interior of the region has to be made. Departing from the textbook finite element model, we approximate the zero order term of (2.1) by a diagonal matrix rather than a multiple of the mass matrix of L^2 inner products of the basis functions of V^h.

In our capacitance matrix method, we use a fast Poisson solver on a uniform mesh which is an extension of the regular, interior part of the finite element triangulation. For each vertex of the triangulation which does not fall on the uniform mesh, we associate a nearby point of the mesh in such a way that we create a one-to-one correspondence between the set of all vertices and a subset of points of the uniform mesh. This set of mesh points naturally divides into two disjoint sets S_r^h and S_{ir}^h, corresponding to the

points where the regular five-point formula applies and remaining points close to the boundary. Point charges, which simulate a single layer charge distribution, are associated with the set S_{ir}^h.

Denote by G the restriction of the solution operator defined by the fast Poisson solver to the set $S_r^h \cup S_{ir}^h$. We seek the solution of the linear system

$$Ax = b. \tag{3.1}$$

For $c = 0$ the matrix A is singular, but this does not create any serious difficulties; see Proskurowski and Widlund [41]. By subtracting, if necessary, a solution obtained from the fast Poisson solver with an arbitrary extension of the vector b as data, we reduce the problem to the case where the right-hand side vanishes except on the set S_{ir}^h. By analogy to the continuous case, we make the Ansatz

$$x = Gy,$$

where the vector y vanishes at all points of S_r^h, and obtain

$$AGy = b. \tag{3.2}$$

The principal submatrix of AG corresponding to the set S_{ir}^h is the capacitance matrix C. The original linear system (3.1) is thus reduced to a much smaller problem, and this reduction in dimensionality, analogous to that of the potential theory of § 2, is accompanied by a very favorable change in the spectral distribution which makes iterative methods quite attractive; see further §§ 4 and 6.

The capacitance matrix can be computed inexpensively for two-dimensional problems if a suitable fast Poisson solver is used, and the resulting linear system of equations can be solved by Gaussian elimination, at least when $c > 0$, or by an iterative method; see Proskurowski and Widlund [41] or O'Leary and Widlund [36] for details. However, in our experiments, we have instead used an iterative method in which access to the capacitance matrix C is needed only in terms of products of C with vectors. It is not necessary to compute or store the capacitance matrix. This idea is due originally to George [19]. To find Cz, charges corresponding to the vector z are placed at the points of S_{ir}^h, the fast Poisson solver is applied to these data and the values of AGz are computed at the points corresponding to the set S_{ir}^h. The data for the Poisson solver is quite sparse and a close examination of the special structure of the matrix A shows that we need the values of Gz only at the points of S_{ir}^h. We have, therefore, chosen to use a special fast Poisson solver similar to that developed by Banegas [3]. See also Proskurowski [39] and § 5 for an operation count and the savings in storage.

4. A conjugate gradient method. We now consider the solution of (3.2) by a special conjugate gradient method. This system has a nonsymmetric coefficient matrix which, however, is similar to a symmetric positive semidefinite matrix. We also consider the solution of (2.7) by the same method to stress further the close connection between the integral and capacitance matrix equations. We note that the use of conjugate gradient methods in the present context has been discussed previously. See O'Leary and Widlund [36] and references therein.

The standard conjugate gradient method applies to symmetric, positive definite problems; see Hestenes and Stiefel [21] or Luenberger [32]. Since it is widely used and well understood, we discuss it only briefly.

Let M be symmetric and positive definite, $Mv = d$ a linear system, v_0 an initial guess, and $r_0 = d - Mv_0$ the corresponding initial residual. Let

$$r_0, \quad Mr_0, \quad \cdots, \quad M^{k-1}r_0$$

be the first k elements of the so-called Krylov sequence, and denote by $S^{(k)}$ the subspace spanned by these vectors. The minimizing element v_k for the problem

$$\min_{u-v_0\in S^{(k)}} \tfrac{1}{2}u^T Mu - u^T d$$

is identical to the standard conjugate gradient approximation. The approximation is thus optimal in a natural sense and, by expanding the initial error $v_0 - v$ in eigenvectors of M, it can be shown that

$$E(v_k) \leqq \min_{P_{k-1}} \max_{\lambda\in\sigma(M)} (1-\lambda P_{k-1}(\lambda))^2 E(v_0). \tag{4.1}$$

Here, $E(u) = (u-v)^T M(u-v)$ is an error functional, $\sigma(M)$ the spectrum of M and P_{k-1} an algebraic polynomial of degree $k-1$. See further Daniel [15], Kaniel [28] or Luenberger [32]. The inequality (4.1) provides a basis for estimates of the rate of convergence of the method. In particular, if the polynomial P_{k-1} is chosen in terms of Chebyshev polynomials, the estimate

$$E(v_k) \leqq \left(\frac{2\left(1-\frac{1}{\kappa}\right)^k}{\left(1+\frac{1}{\sqrt{\kappa}}\right)^{2k} + \left(1-\frac{1}{\sqrt{\kappa}}\right)^{2k}} \right)^2 E(v_0) \tag{4.2}$$

results, where κ is the spectral condition number of M. We note that conjugate gradient methods require no a priori information on the spectrum of the operator and that the acceleration parameters are chosen automatically.

Equation (3.2) has the form

$$AGy = b,$$

where, by assumption, $b \in R(A)$, the range of A. If the coefficient c is strictly positive, the matrix A is positive definite and we can also assume that the fast Poisson solver is chosen such that this is the case for the operator G. For $c = 0$, we assume that the fast solver is chosen such that the restriction of G to $R(A)$ is symmetric positive definite. Since the same condition holds for A and all elements of the Krylov sequence, $r_0, AGr_0, (AG)^2 r_0, \cdots$, belong to $R(A)$, the entire iteration is confined to $R(A)$ if the initial guess is chosen in $R(A)$. We then essentially have a positive definite problem and from now on we consider only that case. Introduce a new variable $z = G^{1/2}y$ and multiply (3.2) by $G^{1/2}$:

$$G^{1/2}AG^{1/2}z = G^{1/2}b.$$

The new operator is symmetric and the standard conjugate gradient method can be used. The following algorithm, which requires five vectors of storage, emerges when we return to the variable y.

Compute

$$r_0 = b - AGy_0,$$

and set

$$p_0 = r_0.$$

For $k = 0, 1, 2, \cdots$:
Update the solution y_k and the residual r_k by

$$y_{k+1} = y_k + \alpha_k p_k,$$

$$r_{k+1} = r_k - \alpha_k AGp_k,$$

where

$$\alpha_k = \frac{r_k^T G r_k}{p_k^T GAGp_k}.$$

Find a new search direction p_{k+1} by

$$p_{k+1} = r_{k+1} + \beta_k p_k,$$

where

$$\beta_k = \frac{r_{k+1}^T G r_{k+1}}{r_k^T G r_k}.$$

This algorithm appears to require the multiplication of the two vectors p_k and r_{k+1} by G, i.e., the use of the fast Poisson solver twice in each iteration step. This can, however, be avoided by introducing additional auxiliary vectors $q_k = Gp_k$ and s_k. The alternative form of the algorithm, which requires no extra storage, has the following form.

Compute

$$r_0 = b - AGy_0,$$

and set

$$p_0 = r_0.$$

Compute

$$q_0 = Gp_0,$$

and set

$$s_0 = q_0.$$

For $k = 0, 1, \cdots$:
Update the solution y_k and the residual r_k by

$$y_{k+1} = y_k + \alpha_k p_k,$$

$$r_{k+1} = r_k - \alpha_k Aq_k,$$

with

$$\alpha_k = \frac{r_k^T s_k}{q_k^T Aq_k}.$$

Compute

$$s_{k+1} = Gr_{k+1}.$$

Find a new search direction p_{k+1} and a new auxiliary vector q_{k+1} by

$$p_{k+1} = r_{k+1} + \beta_k p_k,$$

$$q_{k+1} = s_{k+1} + \beta_k q_k,$$

where

$$\beta_k = \frac{r_{k+1}^T s_{k+1}}{r_k^T s_k}.$$

As pointed out in § 3, the iteration can be carried out using only vectors of an order equal to the number of irregular mesh points and in addition, if a more traditional fast Poisson solver is used, a two-dimensional work array. The possibility of exploiting the sparsity of the vectors y_k, r_k, etc., gives this algorithm an advantage over the generalized conjugate gradient algorithm considered by Concus, Golub and O'Leary [12] and others. In their algorithm the vectors $x_k = Gy_k$ are carried, and they fail to be sparse in this application.

The error estimates (4.1) and (4.2) apply for our algorithm. The relevant spectrum is now that of $G^{1/2}AG^{1/2}$ which coincides with that of AG.

The integral equation (2.7) can also be solved by this conjugate gradient method. The role of AG is then played by the operator $I-K$ and that of G by the symmetrizer Q. Information on the distribution of the nonzero eigenvalues of $I-K$ enables us to estimate the rate of convergence of the conjugate gradient method. We have already shown that all eigenvalues of $I-K$ are less than or equal to two. By elementary properties of compact operators, the eigenvalues have one cluster, at the point 1. The smallest eigenvalue is bounded away from zero by a positive number γ which depends only on Ω. Linear convergence of the iterative method follows directly from (4.2), and more elaborate constructions of the polynomial P_{k-1} show that the convergence is superlinear in this case. See further Hayes [20], Shieh [42], [43] and Widlund [46] for further discussion of superlinear convergence.

5. Implementation of the method. In this section, we describe in some detail how a FORTRAN program can be designed and give additional information on operation counts and storage. Since this is our first program of this kind, we expect that future versions developed inside a framework similar to that outlined in § 3 will represent improvements in several respects when compared to this first effort.

The region Ω is imbedded in a larger region for which a fast Poisson solver is available, and a suitable rectangular mesh covering this larger region is introduced. The coordinates of all mesh points which fail to have all their nearest mesh neighbors in the open set Ω and their signed distances to the boundary Γ along mesh lines provide all the information about the geometry necessary for this simple finite element method. The coordinates of a number of points on Γ are thus available, and a subset of these points is used to construct the polygon Γ^h which approximates Γ. The skin area, briefly described in § 3, is bounded by Γ^h and a second polygon which is the boundary of a union of the regular triangles which covers most of the interior of Ω.

These polygons are constructed as follows. A parameter α, $0 \leqq \alpha \leqq 1$, is introduced to control the minimal thinness of the triangles. For a study of the performance of certain problems for different values of α, see § 6. Similarly, the set of points on Γ can be thinned out to avoid very short edges. The inner polygon is defined in terms of the mesh points in Ω which are within $(1+\alpha)h$ of the boundary in at least one coordinate direction, excluding those which fall within αh of Γ. This set of points, which normally differs from the one originally available but which is easy to compute from that information, is sorted so that consecutive points are within $\sqrt{2}\,h$ of each other and stored in a circular list. For each available point on Γ which does not fall on the mesh we seek a nearby mesh point which is not to be used as a vertex in the triangulation. These mesh points, as well as those defining the inner polygon, carry point charges

during the conjugate gradient iteration. As noted in § 3, no mesh point in the interior of the skin area is used as a vertex, and they can therefore be used for this purpose. Others are found outside Ω. If our rule leads us to assign the same mesh point to several points on Γ, only the point on Γ closest to the mesh point in question is retained in the set, with ties resolved in an arbitrary way. When completed, the list of remaining points on Γ and their associated neighbors on the mesh are sorted and stored in the same way as the point set defining the inner polygon. The construction of the polygons and the set S_{ir}^h is then completed.

The triangulation of the skin area is carried out with the aid of the sorted, circular lists which define the two polygons. By a pair, we denote a pair of points with one element from each of these sets. A pair of nearby points is chosen initially and connected by an edge. The next elements of the lists of vertices are inspected and one of them is chosen to create a new current pair and to complete the first triangle. This process is repeated, always choosing the next vertex of a triangle in the half-space which lies opposite to the half-space defined by the current pair and which contains the previous triangle. If there are two possible ways of completing a triangle, the shortest edge defining a new pair is preferred, resolving a tie in an arbitrary way. When the circular lists are exhausted the skin area is completely covered by triangles, since a closed path can be found through the interior of the union of the triangles constructed and each triangle has exactly one edge which is adjacent to either the exterior of Γ^h or a regular triangle in the interior of Ω.

While this triangulation is carried out, the lengths of the edges of the triangles are computed. Each triangle is stored in terms of a triple of integers, pointing to the lists of coordinates of the vertices, with the first element of the triple associated with the largest angle of the triangle.

With the aid of this information, all nonredundant contributions to the stiffness matrix can be computed by the subassembly method; see, for example, Strang and Fix [44, § 1.10]. The largest angle of any triangle is mapped onto the straight angle of a reference triangle. No list of regular mesh points in the interior of Ω is needed. Suitable quadrature formulas are developed to approximate the right-hand side of (2.3). For a basis function the support of which is a union of regular triangles, only values of the function f at the quadrature nodes and a fixed set of weights are needed. The boundary integral is approximated by integrating exactly the product of basis functions and a linear function which interpolates the boundary data using the values of the Neumann data at the boundary vertices.

Essentially no further new algorithmic ideas, beyond those used previously in our work on finite difference schemes, are needed. See O'Leary and Widlund [36] and Proskurowski [39].

With the exception of the sorting of the lists, all these operations involve only information for relatively few neighboring points at any given time, and the effort required therefore grows only in proportion to p, the number of irregular mesh points. It is well known that sorting can be carried out very efficiently, and we can conclude that for fine meshes the cost of the fast Poisson solver dominates.

In the calculation a total of 15 vectors, of length p, are used to carry geometric information, boundary data and to serve as workspace for the conjugate gradient iteration and the fast Poisson solver which exploits sparsity. By a result from Proskurowski [39], the multiplication count, in our case, for such a fast solver is only $6n \cdot p + 4m \cdot n$, where m and n are the number of mesh points in the two coordinate directions for the region in which Ω is imbedded. In addition, a two-dimensional array of dimension $m \cdot n$ is used to store the data $f(x, y)$ and the final solution; see, however,

Proskurowski [39] for other alternatives which require less storage and which present very attractive space-time tradeoffs on many computer systems.

We note that neither the right-hand side of the Poisson equation nor the solution at the interior mesh points is required during the conjugate gradient iteration which makes up the main part of the calculation. It might appear as if the solution is needed at certain regular mesh points which have neighbors in the set S_{ir}^h in order to calculate the restriction of mesh functions of the form AGy to the set S_{ir}^h. However, the relevant rows of A can be represented as the sum of the standard five-point stencil and stencils which only involve points of S_{ir}^h. No special information on this subset of the regular points is therefore required, since the five-point formula applied to a mesh function of the form Gy returns a component of y.

6. Numerical experiments. In this section, we report on numerical experiments which were carried out on the CDC 6600 at the Courant Institute and a CDC 7600 at the Lawrence Berkeley Laboratory, using FTN (OPT = 2) FORTRAN compilers, and a DEC 10 at the University of Southern California. Single precision was used throughout, even on the DEC 10 which has a short word length.

In our tables, N denotes the number of degrees of freedom in our finite element model and p the number of irregular mesh points. In the cases we tried, $m \times n$, the number of mesh points in the rectangle in which the region is imbedded, exceeded the value of N by a factor of about 2.5 and our fast Poisson solver, which exploits sparsity, required about 15–20 $m \times n$ multiplications and about as many additions.

In our tables, the following norms are used:

$$\|r\|_G = \left(\frac{r^T G r}{p}\right)^{1/2}$$

and

$$\|e\|_2 = \left(\frac{e^T e}{N}\right)^{1/2}.$$

Here, G is the operator defined by the fast Poisson solver. We note that the G-norm is an analogue of the Q-norm introduced in § 2, recalling that during the iteration, the residual r differs from zero only at irregular mesh points. In our computations, the iteration is terminated when $\|r\|_G$ drops below a prescribed tolerance. Such a stopping criterion has been found quite satisfactory, just as for our finite difference codes. For many problems, a condition of the form $\|r_k\|_G \leqq 10^{-3} \times$ (an estimate of the norm of the solution u) provides a satisfactory accuracy.

In an early series of experiments on the CDC 6600, we explored the numerical stability and the rate of convergence of the iteration for a discrete problem for which the exact solution is known. The solution was equal to $x+y$, the region a polygon approximating a circle of radius 0.375, and N varied between 507 and 7481. The parameter α, introduced in § 5, was set equal to zero. The relative l_2-error decreased below 10^{-3} in 5 to 8 iterations, and in all cases it dropped below 10^{-8} in less than 20 iterations. The rate of convergence of the iteration was virtually independent of N. Experiments with other smooth solutions and with solutions generated randomly likewise showed only very minor differences in the rate of convergence. The results reported in the tables below are therefore quite representative of the performance of our iterative method.

For smooth solutions and polygonal approximations of the region with smooth boundaries, we have consistently found the method to be second order accurate provided sufficiently many iteration steps are carried out. As expected, the simplified

approximation of the constant term of the differential operator, which was introduced in § 3, does not degrade the accuracy of the finite element model.

While this first set of experiments shows no ill effects of small triangles, the performance of our algorithm can deteriorate for increasing N if the parameter α is chosen to be zero or quite small. This is illustrated in Table 1, where the region is a circle of radius 0.365. As seen from the table, a choice of $\alpha \geqq 0.2$ is adequate to alleviate this problem. See also Table 3 for detailed study of the iteration for the same region and $\alpha = 0.5$.

TABLE 1a

The effect on t, the number of c.g. *steps, of enlarging the width of the skin area by increasing α. The region is a circle of radius* 0.365. *The solution is* $u = x + y$ *with* $\|u\|_2 = 0.26$. *The iteration was terminated when* $\|r\|_G \leqq 10^{-3}$.

$m = n$	α	t	$\|r\|_G$	$\|e\|_2$
16	0.0	12	.8E−3	.53E−3
	0.1	6	.5E−3	.57E−3
	0.2	6	.5E−3	.49E−3
	0.5	4	.6E−3	.32E−3
32	0.0	10	.7E−3	.32E−3
	0.1	6	.3E−3	.17E−3
	0.2	6	.3E−3	.17E−3
	0.5	5	.4E−3	.17E−3
64	0.0	15	.8E−3	.65E−3
	0.1	8	.3E−3	.18E−3
	0.2	5	.9E−3	.12E−2
	0.5	4	.3E−3	.36E−3
128	0.0	10	.8E−3	.17E−2
	0.1	6	.7E−3	.16E−2
	0.2	4	.1E−2	.25E−2
	0.5	3	.8E−3	.24E−2

TABLE 1b

The effect on t, the number of c.g. *steps, at the tolerances* $\|r\|_G \leqq 10^{-3}$ *and* $\|r\|_G \leqq 10^{-6}$ *when N increases. The problem is the same as in Table* 1a *with* $\alpha = 0.5$. *For the two largest problems the solutions were not computed at all the mesh points. The source of the discretization error is the approximation of the boundary.*

$m = n$	N	p	t	$\|r\|_G$	$\|e\|_2$	DEC 10 CPU sec.
16	145	68	4	.6E−3	.49E−3	.57
			8	.8E−6	.44E−3	.83
32	497	132	5	.4E−3	.17E−3	1.94
			10	.1E−6	.95E−4	2.94
64	1861	268	4	.3E−3	.36E−3	5.96
			9	.5E−6	.26E−4	9.27
128	7001	524	3	.8E−3	.24E−2	20.37
			9	.6E−6	.10E−4	34.94
200	16541	820	3	.6E−3	not avail.	31.63
			8	.8E−6	not avail.	not avail.
400	66541	1644	2	.9E−3	not avail.	101.41
			8	.7E−6	not avail.	not avail.

TABLE 2a

Results for the nonconvex region described in the text. Iteration was terminated when $\|r\|_G \leqq 10^{-3}$. *The solution* $u = x + y$.

$m = n$	N	p	α	t	$\|r\|_G$	$\|e\|_2$
64	1243	266	0.0	22	.9E−3	.30E−2
	1211	262	0.2	12	.6E−3	.18E−2
	1172	260	0.5	5	.6E−3	.86E−3
128	4953	530	0.0	22	.9E−3	.18E−2
	4899	524	0.2	7	.7E−3	.39E−2
	4830	524	0.5	5	.6E−3	.24E−2

TABLE 2b

Results for the nonconvex region and $\alpha = 0.5$. *Termination when* $\|r\|_Q \leqq 10^{-3}$ *and* $\|r\|_Q \leqq 10^{-6}$, *respectively. The* DEC 10 CPU *times for these problems exceeded those recorded for the same m in Table* 1b *by between* 11 *and* 24%.

$m = n$	N	p	t	$\|r\|_G$	$\|e\|_2$
64	1172	260	5	.6E−3	.86E−3
			11	.6E−6	.75E−3
128	4830	524	5	.6E−3	.24E−2
			11	.6E−6	.30E−3
200	11883	820	3	.1E−2	not avail.
			10	.1E−5	not avail.
400	47994	1644	3	.8E−3	not avail.
			10	.9E−6	not avail.

TABLE 3

A detailed study of $\|r\|_G$ *and* $\|e\|_2$ *as a function of t and the mesh size. The discrete solution is a second-order accurate approximation to* $x + y$ *on a circle of radius* 0.365, *cf. Table* 1, *and* $\alpha = 0.5$. *The runs were carried out in single precision on a* DEC 10.

	$m = n = 16$		$m = n = 32$		$m = n = 64$		$m = n = 128$	
t	$\|r\|_G$	$\|e\|_2$	$\|r\|_G$	$\|e\|_2$	$\|r\|_G$	$\|e\|_2$	$\|r\|_G$	$\|e\|_2$
1	.28E−1	.94E−1	.20E−1	.10	.13E−1	.10	.94E−2	.10
2	.89E−2	.10E−1	.13E−1	.25E−1	.40E−2	.61E−2	.28E−2	.60E−2
3	.27E−2	.12E−2	.25E−2	.30E−2	.11E−2	.20E−2	.84E−3	.24E−2
4	.62E−3	.49E−3	.11E−2	.95E−3	.32E−3	.36E−3	.29E−3	.35E−3
5	.14E−3	.44E−3	.35E−3	.17E−3	.66E−4	.34E−4	.62E−4	.58E−4
6	.27E−4	.43E−3	.91E−4	.11E−3	.23E−4	.35E−4	.17E−4	.27E−4
7	.42E−5	.44E−3	.34E−4	.10E−3	.44E−5	.27E−4	.63E−5	.11E−4
8	.77E−6	.44E−3	.10E−4	.95E−4	.12E−5	.27E−4	.12E−5	.11E−4
9	—	—	.35E−5	.95E−4	.45E−5	.26E−4	.56E−6	.11E−4
10	—	—	.97E−6	.95E−4	—	—	—	—

In Table 2, results for a nonconvex, piecewise semicircular region are given. The region Ω is defined by

$$\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3 \backslash \Omega_4,$$

where Ω_1 and Ω_4 are the intersections of the circular disks centered at $(0, 0)$ and $(0, r/2)$ and with radius r and $r/6$, respectively, with the right half-plane, while Ω_2 and Ω_3 are the intersections of the circular disks centered at $(0, -r/3)$ and $(0, 5r/6)$ and with radii $r/6$

with the left half-plane. The parameter $r = 0.365$ in this experiment. We note that it is even more important to use a nonzero value of α for this region. A closer examination of the problems with $\alpha = 0$ show highly nonmonotonic convergence.

In our experience, our code runs about 18 times faster on a CDC 7600 than on a DEC 10. In a typical run, about 70% of the time is spent executing the code of the fast Poisson solver subroutine. The speed of the code could therefore easily be upgraded by replacing that routine by existing faster codes. We finally note that the execution times given include the generation of the finite element model and the data necessary for the run in question.

Acknowledgments. The authors want to thank Paul Concus, Dianne P. O'Leary and Bertrand Mercier for their interest and assistance with this paper. We also thank Maksymilian Dryja and Gene Golub who introduced us to the Soviet literature on methods of fictitious domains. Special thanks are also due to Mark Pekker [37], [38] who provided us with preprints which greatly influenced our presentation in § 2. (We note that Mr. Pekker changed his name to Friedman after leaving the Soviet Union.)

REFERENCES

[1] G. P. ASTRAKHANTSEV, *Method of fictitious domains for a second-order elliptic equation with natural boundary conditons*, U.S.S.R. Computational Math. and Math. Phys., 18 (1978), pp. 114–121.

[2] ———, *On the numerical solution of Dirichlet's problem in an arbitrary region*, Methods of Computational and Applied Mathematics, vol. 2, P. I. Marchuk, ed., Novosibirsk, 1977.

[3] A. BANEGAS, *Fast Poisson solvers for problems with sparsity*, Math. Comp., 32 (1978), pp. 441–446.

[4] R. E. BANK, *Marching algorithms for elliptic boundary value problems. II: The variable coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 950–970.

[5] ———, *A Fortran implementation of the generalized marching algorithm*, ACM Trans. Math. Software, 4 (1978), pp. 165–176.

[6] R. E. BANK AND D. J. ROSE, *Marching algorithms for elliptic boundary value problems. I: The constant coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 792–829.

[7] O. BUNEMAN, *A compact noniterative Poisson solver*, Report SUIPR-294, Inst. Plasma Research, Stanford University, Stanford, CA, 1969.

[8] B. L. BUZBEE AND F. W. DORR, *The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions*, SIAM J. Numer. Anal., 11 (1974), pp. 753–763.

[9] B. L. BUZBEE, F. W. DORR, J. A. GEORGE AND G. H. GOLUB, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.

[10] B. L. BUZBEE, G. H. GOLUB AND C. W. NIELSON, *On direct methods for solving Poisson's equation*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[11] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North Holland, Amsterdam–New York, 1978.

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

[13] R. COURANT, *Variational methods for the solution of problems of equilibrium and vibrations*, Bull. Amer. Math. Soc., 49 (1943), pp. 1–23.

[14] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics*, Vol. 2, Wiley-Interscience, New York, 1953.

[15] J. W. DANIEL, *The conjugate gradient method for linear and nonlinear operator equations*, SIAM J. Numer. Anal., 4 (1967), pp. 10–26.

[16] D. FISCHER, G. GOLUB, O. HALD, C. LEIVA AND O. WIDLUND, *On Fourier-Toeplitz methods for separable elliptic problems*, Math. Comp., 28 (1974), pp. 349–368.

[17] I. FREDHOLM, *Sur une nouvelle méthode pour la resolution du problème de Dirichlet*, Översigt af Kongliga Svenska Vetenskaps-Akademiens Förhandlingar, 57 (1901), pp. 39–46.

[18] P. R. GARABEDIAN, *Partial Differential Equations*, John Wiley, New York, 1964.

[19] J. A. GEORGE, *The use of direct methods for the solution of the discrete Poisson equation on nonrectangular regions*, Computer Science Department Report 159, Stanford University, Stanford CA, 1970.

[20] R. M. HAYES, *Iterative methods of solving linear problems on Hilbert space*, Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues, O. Taussky, ed., Nat. Bur. Standards, Appl. Math. Series, 39, 1954, pp. 71–103.

[21] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[22] R. W. HOCKNEY, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach., 12 (1965), pp. 95–113.

[23] ———, *Formation and stability of virtual electrodes in a cylinder*, J. Appl. Phys., 39 (1968), pp. 4166–4170.

[24] ———, *The potential calculation and some applications*, Methods in Computational Physics, vol. 9, Academic Press, New York, 1970.

[25] ———, *Pot 4 – a fast direct Poisson solver for the rectangle allowing some fixed boundary conditions and internal electrodes*, IBM Research, R. C. 2870, 1970.

[26] V. P. IL'IN AND V. A. KOROTKEVICH, *On the solution of Poisson's equation on nonrectangular regions*, Numerical Methods in Solid Mechanics, Mathematical Modeling (Novosibirsk), 7, 7 (1976), pp. 30–44.

[27] P. JAMET, *Estimation d'erreur pour des elements finis droits presque degeneres*, RAIRO., 10–R (1976), pp. 43–61.

[28] S. KANIEL, *Estimates for some computational techniques in linear algebra*, Math. Comp., 20 (1966), pp. 369–378.

[29] V. G. KORNEEV, *Iterative methods of solving systems of equations of the finite element method*, USSR Computational Math. and Math. Phys., 17 (1977), pp. 109–129.

[30] JU. A. KUZNETSOV AND A. M. MATSOKIN, *A matrix analog of the method of fictitious domains*, preprint, Novosibirsk Computing Center, 1974.

[31] O. D. KELLOGG, *Foundations of Potential Theory*, Springer-Verlag, Berlin, Heidelberg, New York, 1967.

[32] D. G. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading MA., 1973.

[33] E. D. MARTIN, *A generalized capacity matrix technique for computing aerodynamic flows*, Internat. J. Comput. Fluids, 2 (1974), pp. 79–97.

[34] J. C. NEDELEC, *Approximation des equations intégrales en méchanique et en physique*, Centre de Mathématiques Appliquées – Ecole Polytechnique, Palaiseau, France, 1977.

[35] J. C. NEDELEC AND J. PLANCARD, *Une methode variationelle d'elements finis pour la resolution numerique d'un probleme exterieur dans* R^3, RAIRO., 7–R (1973), pp. 105–129.

[36] D. P. O'LEARY AND O. WIDLUND, *Capacitance matrix methods for the Helmholtz equation on general three-dimensional regions*, Math. Comp., 33 (1979), pp. 849–879.

[37] M. PEKKER, *On the numerical solution of the linear magnetostatic problems by the combination of the boundary integral methods and the fast Laplace solver*, to appear.

[38] M. PEKKER (see M. J. FRIEDMAN), *A finite element method for the solution of a potential theory integral equation*, Math. Meth. in Appl. Sci., 1 (1979), pp. 581–587.

[39] W. PROSKUROWSKI, *Numerical solution of Helmholtz's equation by implicit capacitance matrix methods*, ACM Trans. Math. Software, 5 (1979), pp. 36–49.

[40] ———, *Four* FORTRAN *programs for numerically solving Helmholtz's equation in an arbitrary bounded planar region*, Report LBL–7516, Lawrence Berkeley Laboratory, Berkeley, CA, 1978.

[41] W. PROSKUROWSKI AND O. WIDLUND, *On the numerical solution of Helmholtz's equation by the capacitance matrix method*, Math. Comp., 30 (1976), pp. 433–468; appeared also in an NYU Report.

[42] A. S. L. SHIEH, *On the convergence of the conjugate gradient method for singular capacitance matrix equations from the Neumann problem of the Poisson equation*, Numer. Math., 29 (1978), pp. 307–327.

[43] ———, *Fast Poisson solvers on general two-dimensional regions for the Dirichlet problem*, Numer. Math., 31 (1979), pp. 405–429.

[44] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[45] P. SWARZTRAUBER AND R. SWEET, *Efficient Fortran subprograms for the solution of elliptic partial differential equations*, Report NCAR–TN/IA–109, National Center for Atmospheric Research, Boulder, Colorado, 1975.

[46] O. WIDLUND, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/80/0104-0003 $01.00

ON IMPROVING THE 2–4 TWO-DIMENSIONAL LEAP-FROG SCHEME*

SAUL ABARBANEL† AND DAVID GOTTLIEB‡

Abstract. The present paper shows how to modify the Kreiss–Oliger 2–4 two-dimensional leap-frog scheme so that the allowable time step may be doubled while the computational complexity remains about the same.

1. Introduction. In a previous communication [1] (see also [2]) it was shown how the standard multidimensional leap-frog finite difference method for solving linear and quasi-linear systems of partial differential equations can be modified so that the stability condition is substantially improved. In particular, it was shown that in the two-dimensional case one can double the time step. The required change in the algorithm is simple to implement, and the resulting modified leap-frog (MLF) algorithm remains convenient to program and requires no more flux vector evaluations than the original leap-frog scheme.

Recently the question arose whether a similar improvement can be achieved for the Kreiss–Oliger 2–4 scheme [3], which is second order accurate in time and has a fourth order spatial accuracy. This scheme is widely used in meteorology and global circulation studies. Its explicitness imposes, of course, a priori restrictions on the time steps, and attempts have been made [4] to improve running times by resorting to implicit 2–4 methods [5]. Improvements of a factor of 2 in machine time have been reported [4]. In this paper we show how a similar improvement may be obtained by modifying the explicit 2–4 scheme in a manner analogous to that reported in [1].

Here we are considering the hyperbolic system

$$\frac{\partial u}{\partial t}=\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=A\frac{\partial u}{\partial x}+B\frac{\partial u}{\partial y}, \tag{1.1}$$

where u, $F(u)$ and $G(u)$ are m-component vectors and where A and B are the Jacobians of F and G with respect to u.

The 2–4 Kreiss–Oliger finite difference scheme is the following:

$$\begin{aligned} u_{j,k}^{n+1} &= u_{j,k}^{n-1}+\left(\frac{\Delta t}{\Delta x}\right)\left(-\frac{1}{6}F_{j+2,k}^{n}+\frac{4}{3}F_{j+1,k}^{n}-\frac{4}{3}F_{j-1,k}^{n}+\frac{1}{6}F_{j-2,k}^{n}\right)\\ &\quad+\left(\frac{\Delta t}{\Delta y}\right)\left(-\frac{1}{6}G_{j,k+2}^{n}+\frac{4}{3}G_{j,k+1}^{n}-\frac{4}{3}G_{j,k-1}^{n}+\frac{1}{6}G_{j,k-2}^{n}\right)\\ &= u_{j,k}^{n-1}+\left(\frac{\Delta t}{\Delta x}\right)\left[(F_{j+1,k}^{n}-F_{j-1,k}^{n})-\frac{1}{6}(F_{j+2,k}^{n}-2F_{j+1,k}^{n}+2F_{j-1,k}^{n}-F_{j-2,k}^{n})\right]\\ &\quad+\left(\frac{\Delta t}{\Delta y}\right)\left[(G_{j,k+1}^{n}-G_{j,k-1}^{n})-\frac{1}{6}(G_{j,k+2}^{n}-2G_{j,k+1}^{n}+2G_{j,k-1}^{n}-G_{j,k-2}^{n})\right]. \end{aligned} \tag{1.2}$$

* Received by the editors May 8, 1980, and in revised form September 23, 1980. This work was supported in part by the Air Force Office of Scientific Research (NAM), the European Office of Aerospace Research, AFSC, United States Air Force, under Grant AFOSR 78-3651, and in part by NASA Contract NAS1-14101 and Contract NAS1-15810 at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

† Department of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel.

‡ Department of Mathematical Sciences, Tel-Aviv University, Tel-Aviv, Israel, and Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665.

The second term in each of the square brackets serves as a "correction" term to the regular 2–2 leap-frog method and modifies it into a 2–4 scheme, i.e., second order accurate in time and fourth order accurate spatially. The initial value (linear) stability condition for algorithm (1.2) is

$$\Delta t < \frac{1}{\left[\left(\frac{\rho(A)}{\Delta x}\right)+\left(\frac{\rho(B)}{\Delta y}\right)\right]D}, \tag{1.3}$$

where $\rho(A)$ and $\rho(B)$ are, respectively, the spectral radii of the coefficient matrices $A=A(u)=\partial F/\partial u$ and $B=B(u)=\partial G/\partial u$, which are assumed to be simultaneously symmetrizable. The factor D equals 1.372 $(=f((\frac{3}{8})^{1/4})$; see (2.6)).

We now seek to modify (1.2) in order to improve (1.3). The best we can hope for is to achieve the one-dimensional stability conditions, namely

$$\frac{\Delta t}{\Delta x} < \frac{1}{\rho(A)D} \quad \text{and} \quad \frac{\Delta t}{\Delta y} < \frac{1}{\rho(B)D}. \tag{1.4}$$

2. The modified scheme and its stability. If we introduce the differencing and averaging operators, respectively

$$\delta_x q_{j,k} = q_{j+1/2,k} - q_{j-1/2,k}, \qquad \delta_y q_{j,k} = q_{j,k+1/2} - q_{j,k-1/2},$$
$$\mu_x q_{j,k} = (\tfrac{1}{2})(q_{j+1/2,k} + q_{j-1/2,k}), \qquad \mu_y q_{j,k} = (\tfrac{1}{2})(q_{j,k+1/2} + q_{j,k-1/2}),$$

then the scheme (1.2) takes the form

$$u_{j,k}^{n+1} = u_{j,k}^{n-1} + 2\left(\frac{\Delta t}{\Delta x}\right)\delta_x\mu_x\left(1-\frac{1}{6}\delta_x^2\right)F_{j,k}^n + 2\left(\frac{\Delta t}{\Delta y}\right)\delta_y\mu_y\left(1-\frac{1}{6}\delta_y^2\right)G_{j,k}^n. \tag{2.1}$$

Next we review briefly how one establishes the stability limits of (2.1); this will facilitate the treatment of the modified algorithm to be introduced later. If we define a new vector

$$w_{j,k}^n = \begin{pmatrix} u_{j,k}^n \\ v_{j,k}^n \end{pmatrix}, \tag{2.2}$$

then the linearized version of the two-level finite difference equations (2.1) becomes the following single level system:

$$\begin{aligned} u_{j,k}^{n+1} &= v_{j,k}^n + 2\lambda_x\mu_x\delta_x(I-\tfrac{1}{6}\delta_x^2)u_{j,k}^n + 2\lambda_y\mu_y\delta_y(I-\tfrac{1}{6}\delta_y^2)u_{j,k}^n, \\ v_{j,k}^{n+1} &= u_{j,k}^n, \end{aligned} \tag{2.3}$$

or equivalently,

$$w_{j,k}^{n+1} = \begin{bmatrix} u_{j,k}^{n+1} \\ v_{j,k}^{n+1} \end{bmatrix} = \begin{bmatrix} 2\lambda_x\mu_x\delta_x(I-\frac{1}{6}\delta_x^2) + 2\lambda_y\mu_y\delta_y(I-\frac{1}{6}\delta_y^2) & I \\ I & 0 \end{bmatrix}\begin{bmatrix} u_{j,k}^n \\ v_{j,k}^n \end{bmatrix}, \tag{2.4}$$

where $\lambda_x = A\Delta t/\Delta x$ and $\lambda_y = B\Delta t/\Delta x$. Fourier transforming (2.4) we get the amplification matrix

$$M = \begin{bmatrix} 2i(\lambda_x f(\alpha) + \lambda_y f(\beta)) & I \\ I & 0 \end{bmatrix}, \tag{2.5}$$

where

$$f(z) = 2z\sqrt{1-z^2}(1+\tfrac{2}{3}z^2), \qquad -1 \leqq z \leqq 1, \tag{2.6}$$

and $\alpha = \sin(\xi/2)$, $\beta = \sin(\eta/2)$, ξ and η being the dual Fourier variables of the space coordinates x and y.

It may be shown that the stability requirement for M is equivalent to demanding that

$$\rho(\mathcal{C})<1, \tag{2.7}$$

$\rho(\mathcal{C})$ being the spectral radius of $\mathcal{C}=A(\Delta t/\Delta x)f(\alpha)+B(\Delta t/\Delta x)f(\beta)$. We then get

$$\Delta t<\frac{1}{\left[\left(\frac{\rho(A)}{\Delta x}\right)+\left(\frac{\rho(B)}{\Delta y}\right)\right]D}, \tag{2.8}$$

where

$$D=\max|f(z)|=f(z=(\tfrac{3}{8})^{1/4})=1.372. \tag{2.9}$$

The most general modification of (2.1) which maintains the fourth order spatial accuracy and still leaves the algorithm with a compact 5×5 grid support ($j\pm 2$, $k\pm 2$) is:

$$\begin{aligned} u_{j,k}^{n+1}=u_{j,k}^{n-1}&+2\left(\frac{\Delta t}{\Delta x}\right)\mu_x\delta_x\left(1-\frac{1}{6}\delta_x^2-\frac{\gamma}{16}\delta_x^2\delta_y^2-\frac{\varepsilon}{16}\delta_y^4-\frac{\nu}{64}\delta_x^2\delta_y^4\right)F_{j,k}^n\\ &+2\left(\frac{\Delta t}{\Delta y}\right)\mu_y\delta_y\left(1-\frac{1}{6}\delta_y^2-\frac{\gamma}{16}\delta_y^2\delta_x^2-\frac{\varepsilon}{16}\delta_x^4-\frac{\nu}{64}\delta_y^2\delta_x^4\right)G_{j,k}^n. \end{aligned} \tag{2.10}$$

The amplification matrix again has the form

$$G=\begin{bmatrix} 2i\tilde{C} & I\\ I & 0\end{bmatrix},$$

where, now,

$$\begin{aligned}\tilde{C}=&\lambda_x[2\alpha(1-\alpha^2)^{1/2}\cdot(1+\tfrac{2}{3}\alpha^2-\gamma\alpha^2\beta^2-\varepsilon\beta^4+\nu\alpha^2\beta^4)]\\ &+\lambda_y[2\beta(1-\beta^2)^{1/2}\cdot(1+\tfrac{2}{3}\beta^2-\gamma\beta^2\alpha^2-\varepsilon\alpha^4+\nu\beta^2\alpha^4)].\end{aligned}$$

The stability requirement becomes

$$\begin{aligned}&2\rho(\lambda_x)|\alpha|\cdot|1-\alpha^2|^{1/2}\cdot|1+\tfrac{2}{3}\alpha^2-\gamma\alpha^2\beta^2-\varepsilon\beta^4+\nu\alpha^2\beta^4|\\ &+2\rho(\lambda_y)|\beta|\cdot|1-\beta^2|^{1/2}\cdot|1+\tfrac{2}{3}\beta^2-\gamma\beta^2\alpha^2-\varepsilon\alpha^4+\nu\beta^2\alpha^4|\leqq 1.\end{aligned} \tag{2.11}$$

For optimal stability it is required that, in contradistinction to (2.8), $\rho(\lambda_x)\leqq 1/D$ and $\rho(\lambda_y)\leqq 1/D$. We ask whether under these constraints there indeed exists a triplet $(\gamma, \varepsilon, \nu)$ such that inequality (2.11) is still valid for all $|\alpha|, |\beta|<1$. It can be shown that if any member of the triplet $(\gamma, \varepsilon, \nu)$ vanishes one cannot get the stipulated optimal stability (1.4). We found no analytical way to determine a stabilizing triplet. However, we verified numerically that the convenient triplet

$$\gamma=8,\quad \varepsilon=\tfrac{8}{3},\quad \nu=\tfrac{32}{3}$$

is appropriate in that it leaves the inequality (2.11) valid under the stipulated optimal stability conditions (1.4).

Thus, the modified Kreiss–Oliger 2–4 leap-frog scheme takes the form

$$\begin{aligned} u_{j,k}^{n+1}=u_{j,k}^{n-1}&+2\left(\frac{\Delta t}{\Delta x}\right)\delta_x\mu_x\left(1-\frac{1}{6}\delta_x^2-\frac{1}{2}\delta_x^2\delta_y^2-\frac{1}{6}\delta_y^4+\frac{1}{6}\delta_x^2\delta_y^4\right)F_{j,k}^n\\ &+2\left(\frac{\Delta t}{\Delta y}\right)\delta_y\mu_y\left(1-\frac{1}{6}\delta_y^2-\frac{1}{2}\delta_y^2\delta_x^2-\frac{1}{6}\delta_x^4+\frac{1}{6}\delta_y^2\delta_x^4\right)G_{j,k}^n. \end{aligned} \tag{2.12}$$

Writing out explicitly the effect of the differencing and averaging operators δ_x, δ_y, μ_x, μ_y in (2.12), one obtains

$$
\begin{aligned}
u_{j,k}^{n+1} = u_{j,k}^{n-1} &+ \frac{1}{6}\left(\frac{\Delta t}{\Delta x}\right)[-F_{j+2,k+2}^n + F_{j+2,k+1}^n - F_{j+2,k}^n + F_{j+2,k-1}^n - F_{j+2,k-2}^n \\
&+ F_{j+1,k+2}^n + 2F_{j+1,k+1}^n + 2F_{j+1,k}^n + 2F_{j+1,k-1}^n + F_{j+1,k-2}^n \\
&- F_{j-1,k+2}^n - 2F_{j-1,k+1}^n - 2F_{j-1,k}^n - 2F_{j-1,k-1}^n - F_{j-1,k-2}^n \\
&+ F_{j-2,k+2}^n - F_{j-2,k+1}^n + F_{j-2,k}^n - F_{j-2,k-1}^n + F_{j-2,k-2}^n] \\
&+ \frac{1}{6}\left(\frac{\Delta t}{\Delta y}\right)[-G_{j+2,k+2}^n + G_{j+1,k+2}^n - G_{j,k+2}^n + G_{j-1,k+2}^n - G_{j-2,k+2}^n \\
&+ G_{j+2,k+1}^n + 2G_{j+1,k+1}^n + 2G_{j,k+1}^n + 2G_{j-1,k+1}^n + G_{j-2,k+1}^n \\
&- G_{j+2,k-1}^n - 2G_{j+1,k-1}^n - 2G_{j,k-1}^n - 2G_{j-1,k-1}^n - G_{j-2,k-1}^n \\
&+ G_{j+2,k-2}^n - G_{j+1,k-2}^n + G_{j,k-2}^n - G_{j-1,k-2}^n + G_{j-2,k-2}^n].
\end{aligned} \tag{2.13}
$$

If we want to emphasize the "modifying" terms which were added to the regular 2–4 scheme, we may rewrite (2.13) as follows:

$$
\begin{aligned}
u_{j,k}^{n+1} = u_{j,k}^{n-1} &+ \left(\frac{\Delta t}{\Delta x}\right)\left\{F_{j+1,k}^n - F_{j-1,k}^n - \frac{1}{6}[F_{j+2,k}^n - 2F_{j+1,k}^n + 2F_{j-1,k}^n - F_{j-2,k}^n]\right. \\
&+ \frac{1}{6}[-F_{j+2,k+2}^n + F_{j+1,k+2}^n - F_{j-1,k+2}^n + F_{j-2,k+2}^n \\
&+ F_{j+2,k+1}^n + 2F_{j+1,k+1}^n - 2F_{j-1,k+1}^n - F_{j-2,k+1}^n - 6F_{j+1,k}^n \\
&+ 6F_{j-1,k}^n + F_{j+2,k-1}^n + 2F_{j+1,k-1}^n - 2F_{j-1,k-1}^n - F_{j-2,k-1}^n \\
&\left. - F_{j+2,k-2}^n + F_{j+1,k-2}^n - F_{j-1,k-2}^n + F_{j-2,k-2}^n]\right\} \\
&+ \left(\frac{\Delta t}{\Delta y}\right)\left\{G_{j,k+1}^n - G_{j,k-1}^n - \frac{1}{6}[G_{j,k+2}^n - 2G_{j,k+1}^n + 2G_{j,k-1}^n - G_{j,k-2}^n]\right. \\
&+ \frac{1}{6}[-G_{j+2,k+2}^n + G_{j+1,k+2}^n + G_{j-1,k+2}^n - G_{j-2,k+2}^n \\
&+ G_{j+2,k+1}^n + 2G_{j+1,k+1}^n - 6G_{j,k+1}^n + 2G_{j-1,k+1}^n + G_{j-2,k+1}^n \\
&- G_{j+2,k-1}^n - 2G_{j+1,k-1}^n + 6G_{j,k-1}^n - 2G_{j-1,k-1}^n - G_{j-2,k-1}^n \\
&\left. + G_{j+2,k-2}^n - G_{j+1,k-2}^n - G_{j-1,k-2}^n + G_{j-2,k-2}^n]\right\}.
\end{aligned} \tag{2.14}
$$

Note that even though (2.13) and (2.14) look much more complex than (1.2) one still realizes the benefit of the increased stable time step. This is true when the flux vector evaluation is costly in terms of machine time, because all the additional fluxes have already been computed at the neighboring points. This can be best seen by defining the following vectors:

$$
\begin{aligned}
u_{j,k}^{*n} &= (1 - \tfrac{3}{8}\delta_x^2\delta_y^2 - \tfrac{1}{8}\delta_y^4 - \tfrac{1}{8}\delta_x^2\delta_y^4)u_{j,k}^n, \\
u_{j,k}^{**n} &= (1 - \tfrac{3}{8}\delta_y^2\delta_x^2 - \tfrac{1}{8}\delta_x^4 - \tfrac{1}{8}\delta_y^2\delta_x^4)u_{j,k}^n.
\end{aligned} \tag{2.15}
$$

In terms of u^* and u^{**} we can construct a different version of the modified 2–4 scheme so that it takes the same form as the standard one (1.2), namely,

$$
\begin{aligned}
u_{j,k}^{n+1} = u_{j,k}^{n-1} &+ \left(\frac{\Delta t}{\Delta x}\right)\left[-\frac{1}{6}F_{j+2,k}^{n} + \frac{4}{3}F_{j+1,k}^{*n} - \frac{4}{3}F_{j-1,k}^{*n} + \frac{1}{6}F_{j-2,k}^{n}\right] \\
&+ \left(\frac{\Delta t}{\Delta y}\right)\left[-\frac{1}{6}G_{j,k+2}^{n} + \frac{4}{3}G_{j,k+1}^{**n} - \frac{4}{3}G_{j,k-1}^{**n} + \frac{1}{6}G_{j,k-2}^{n}\right],
\end{aligned} \tag{2.16}
$$

where $F_{j,k}^{*n} = F(u_{j,k}^{*n})$, $F_{j,k}^{**n} = F(u_{j,k}^{**n})$ and similarly $G_{j,k}^{*n} = G(u_{j,k}^{*n})$, $G_{j,k}^{**n} = G(u_{j,k}^{**n})$. From (2.15) one can show that (2.16) is equivalent to (2.13) (or (2.14)) to fourth order in space in the nonlinear case. The stability of (2.16) is the same as that of (2.14) since for linear problems they are identical. Notice that the number of flux evaluations is now the same for the modified and the standard 2–4 schemes. The dependent vectors $u_{j,k}^{*n}$ and $u_{j,k}^{**n}$ are evaluated in terms of the neighboring points using only additions of u^n. In particular,

$$
\begin{aligned}
u_{j,k}^{*n} = \tfrac{1}{4}u_{j,k}^{n} + \tfrac{1}{8}[&-u_{j+1,k+2}^{n} + u_{j,k+2}^{n} - u_{j-1,k+2}^{n} + u_{j+1,k+1}^{n} \\
&+ 2u_{j,k+1}^{n} + u_{j-1,k+1}^{n} + u_{j+1,k-1}^{n} + 2u_{j,k-1}^{n} \\
&+ u_{j-1,k-1}^{n} - u_{j+1,k-2}^{n} + u_{j,k-2}^{n} - u_{j-1,k-2}^{n}]
\end{aligned} \tag{2.17}
$$

and

$$
\begin{aligned}
u_{j,k}^{**n} = \tfrac{1}{4}u_{j,k}^{n} + \tfrac{1}{8}[&-u_{j+2,k+1}^{n} + u_{j+1,k+1}^{n} + u_{j-1,k+1}^{n} - u_{j-2,k+1}^{n} \\
&+ u_{j+2,k}^{n} + 2u_{j+1,k}^{n} + 2u_{j-1,k}^{n} + u_{j-2,k}^{n} \\
&- u_{j+2,k-1}^{n} + u_{j+1,k-1}^{n} + u_{j-1,k-1}^{n} - u_{j-2,k-1}^{n}].
\end{aligned} \tag{2.18}
$$

Notice that, from a computational point of view, (2.17) and (2.18) can be carried out by observing that certain neighboring points keep appearing together, and therefore can be lumped together to reduce the number of additions and hence making for a more efficient program.

The original Kreiss–Oliger 2–4 scheme [3] is nondissipative. Our modification leaves it so, since the additional higher order terms all affect only the imaginary part of the amplification matrix. In order to make the scheme dissipative, one may use the same artificial dissipation term they used, namely in equation (2.12) replace $u_{j,k}^{n-1}$ by $[I + (\omega/64)(\delta_x^3\mu_x^3 + \delta_y^3\mu_y^3)]u_{j,k}^{n-1}$. For small ω the stability condition hardly changes.

REFERENCES

[1] S. ABARBANEL AND D. GOTTLIEB, *A note on the leap-frog scheme in two and three dimensions*, J. Comp. Phys., 21 (1976), pp. 351–355.

[2] J. C. WILSON, *A note on the leap frog schemes in any number of space variables*, Ibid., 28 (1978), pp. 433–436.

[3] H. O. KREISS AND J. OLIGER, *Method for the Approximation of Time Dependent Problems*, GARP Publ. Ser. No. 10, 1973.

[4] E. ISAACSON, private communication.

[5] R. W. BEAM AND R. F. WARMING, *An implicit finite-difference algorithm for hyperbolic systems in conservation law form*, J. Comp. Phys., 22 (1976), pp. 87–110.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/80/0104-0004 $01.00/0

GENERATION OF BODY-FITTED COORDINATES USING HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS*

JOSEPH L. STEGER† AND DENNY S. CHAUSSEE‡

Abstract. Grid generation equations formulated as hyperbolic partial differential equations are solved numerically to generate body conforming meshes. This grid generation procedure can be efficiently used to generate smoothly varying grids in which the user has good control of the grid clustering. Two-dimensional results are presented for typical external aerodynamic applications.

Key words. numerical methods, grid generation

Introduction. A procedure for generating body-fitted orthogonal grids has recently been advanced by Graves [1] which has as its origins a scheme developed some years earlier by McNally [2]. In this procedure [1] normals are constructed from the initial distribution of points on the body surface to an adjacent control plane or level line. Normals from this level line are then projected to another adjacent level line, and in such a way normals are ultimately formed between all of the level lines contiguously spaced between the body and the outer boundary. The number and location of the prespecified level lines control the radial-like grid spacing.

At first glance the McNally–Graves procedure may appear to be analogous to generating a grid by marching the Cauchy–Riemann equations away from an initial distribution of points on the body—a procedure that would be improperly posed. Their grid generation scheme, however, requires that grid lines be constructed subject to a constraint of orthogonality; and, in addition, the normal-direction grid increments meet prespecified level curves. Thus, the normal-direction grid increments have an essentially specified length. In this paper we show that the McNally–Graves procedure subject to these constraints can be interpreted as an initial-value problem for two nonlinear partial differential equations for grid generation. Furthermore these equations are shown to be hyperbolic and are thus properly posed for the given initial data.

The interpretation of the grid generation algorithm of McNally and of Graves as a hyperbolic partial differential equation grid generation scheme is advantageous in two ways. For one, it puts this procedure on a comparable theoretical basis to the elliptic grid generation procedures of [3]–[6]. Of more importance, though, the interpretation of the grid generation problem as a set of constraints that forces the solution of hyperbolic partial differential equations can ultimately lead to further generalizations and new algorithms. In this regard we have also developed a new scheme in which the mesh generation equations are governed by the constraints of orthogonality and cell volume. The governing equations are again nonlinear and hyperbolic, and because cell volume is specified the coordinate transformation Jacobian is positive and finite. Thus, a nonsingular grid is ensured in all but extreme cases where "shock-wave-like" discontinuities can occur.

1. Analysis of hyperbolic grid generation schemes.

The task of generating the exterior mesh about an arbitrary closed boundary curve such as the one illustrated in

* Received by the editors March 21, 1980, and in revised form August 15, 1980.
† Department of Aeronautics and Astronautics, Stanford University, Stanford, California 94305.
‡ NASA Ames Research Center, Moffett Field, California 94035.

Fig. 1 is considered. Here the outer circumscribing boundary is not specified; it only need be sufficiently far removed from the inner boundary. Such a grid generation problem is frequently encountered in external aerodynamic applications where a far field solution can be specified.

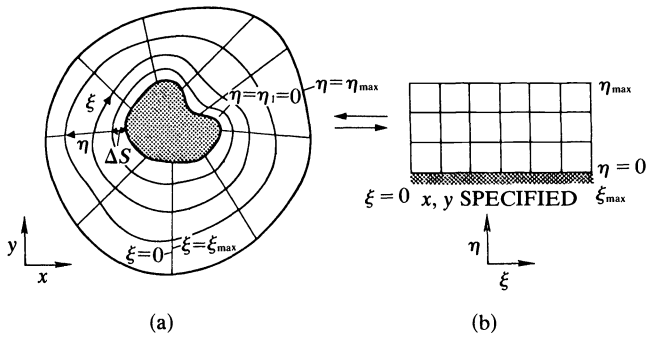


FIG. 1. *Schematic of grid mapping procedure.*

As in [3]–[6], partial differential equations are sought which produce smoothly distributed grid lines of constant ξ and η (see Fig. 1a) such that mesh lines of the same family do not cross or coalesce. In this way a one-to-one mapping will exist between the grid in the physical plane and the specified grid in the computational or transform plane (Fig. 1b). The distribution of grid points on the body is specified, while the outer boundary curve is required to be sufficiently far removed from the interior curve.

A. *Arc length scheme.* Generalizing on the constructive-like grid generation procedure developed in [1] and [2], we propose that the ξ and η coordinate lines satisfy a constraint of orthogonality, i.e.,

(1a) $$\nabla\xi \cdot \nabla\eta = 0,$$

or in the transform plane,

(1b) $$x_\xi x_\eta + y_\xi y_\eta = 0.$$

Moreover, the distance between levels of η = constant lines is constrained by some rule, i.e., in the physical plane

(2a) $$(ds)^2 = (dx)^2 + (dy)^2$$

or

(2b) $$(ds)^2 = (x_\xi\, d\xi + x_\eta\, d\eta)^2 + (y_\xi\, d\xi + y_\eta\, d\eta)^2,$$

where $\Delta s \equiv ds/d\eta$ is specified by the user. For convenience a uniform grid is specified in the computational plane with $d\xi/d\eta = 1$. Expanding (2b) and using (1b) gives

(3) $$x_\xi^2 + y_\xi^2 + x_\eta^2 + y_\eta^2 = \left(\frac{ds}{d\eta}\right)^2 \equiv (\Delta s)^2.$$

Equations (1b) and (3) comprise a system of nonlinear partial differential equations with initial data in η. As shown below, local linearization of this system of equations and subsequent analysis reveals that the equations are hyperbolic and can thus be marched in η from initial data along the body, $\eta = 0$ of Fig. 1.

Let $\tilde{x} = x - x^0$ and $\tilde{y} = y - y^0$, where x^0 and y^0 denote a known near solution state; then term by term linearization of (1b) and (3) proceeds, for example, as

$$
\begin{aligned}
x_\xi x_\eta &= (x^0 + \tilde{x})_\xi (x^0 + \tilde{x})_\eta \\
&\doteq (x^0_\xi x^0_\eta + x^0_\xi \tilde{x}_\eta + x^0_\eta \tilde{x}_\xi) + O(\Delta^2) \\
&= x^0_\xi x^0_\eta + x^0_\xi (x - x^0)_\eta + x^0_\eta (x - x^0)_\xi \\
&= x^0_\xi x_\eta + x^0_\eta x_\xi - x^0_\xi x^0_\eta.
\end{aligned} \tag{4}
$$

The resulting locally linearized system corresponding to (1b) and (3) is then found to be

$$
\begin{bmatrix} x^0_\eta & y^0_\eta \\ x^0_\xi & y^0_\xi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_\xi + \begin{bmatrix} x^0_\xi & y^0_\xi \\ x^0_\eta & y^0_\eta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_\eta = \begin{bmatrix} x^0_\xi x^0_\eta + y^0_\xi y^0_\eta \\ \frac{1}{2}\{(\Delta s)^2 + (x^0_\xi)^2 + (x^0_\eta)^2 + (y^0_\xi)^2 + (y^0_\eta)^2\} \end{bmatrix},
$$

or

$$
A\mathbf{r}_\xi + B\mathbf{r}_\eta = \mathbf{f}. \tag{5}
$$

If $x^0_\xi y^0_\eta - x^0_\eta y^0_\xi$ (which is also the transformation Jacobian) is unequal to zero, then B^{-1} exists. Furthermore, if $B^{-1}A$ has real distinct eigenvalues, the system is hyperbolic and is therefore well posed for initial data in η (cf. [7].) The characteristic equation of $B^{-1}A$ is

$$
\sigma^2 - (x^0_\eta y^0_\xi - x^0_\xi y^0_\eta)^2 = 0, \tag{6}
$$

which clearly has real distinct roots. If $x^0_\xi y^0_\eta - x^0_\eta y^0_\xi = 0$, the mapping from x, y to ξ, η is no longer one-to-one.

Depending on the body shape, the distribution of points along the $\eta = 0$ body boundary and how Δs is specified, good results can be obtained by using (5) to generate x, y grid points in the transformed plane. We can consider the scheme of McNally and Graves' extension of it as a particular solution algorithm of (5), in which Δs is chosen in an indirect way so that certain level line constraints are met. In Graves' work in meshing re-entry aerodynamic shapes, excellent results are obtained for closed bodies without reverse curvature. The same algorithm does a poorer job for bodies with regions of both positive and negative curvature.

B. *Cell volume scheme.* To better ensure a nonsingular mapping from x, y to ξ, η, an alternate set of hyperbolic grid generation equations was formulated. The condition of orthogonality is again imposed. Instead of specifying a grid length increment, however, a grid cell volume (i.e., area in two dimensions) is specified. If a finite grid cell volume is specified, the transformation Jacobian should be nonsingular as

$$
dx\, dy = (x_\xi y_\eta - x_\eta y_\xi)\, d\xi\, d\eta = J^{-1}\, d\xi\, d\eta. \tag{7}
$$

In our numerical implementation $\Delta\xi = \Delta\eta = 1$, so the Jacobian determinate will approximately equal the physical cell volume, $\Delta x\, \Delta y$.

An alternate set of grid generation equations are thus given by

$$
\xi_x \eta_x + \xi_y \eta_y = 0, \tag{8a}
$$

$$
\xi_x \eta_y - \xi_y \eta_x = J, \tag{8b}
$$

or

$$
x_\xi x_\eta + y_\xi y_\eta = 0, \tag{9a}
$$

$$
x_\xi y_\eta - x_\eta y_\xi = \frac{1}{J} \equiv V. \tag{9b}
$$

Here V represents the grid cell volume (actually area) and is user specified.

The linearized form of (9) is found to be

$$\begin{bmatrix} x_\eta^0 & y_\eta^0 \\ y_\eta^0 & -x_\eta^0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}_\xi + \begin{bmatrix} x_\xi^0 & y_\xi^0 \\ -y_\xi^0 & x_\xi^0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}_\eta = \begin{bmatrix} 0 \\ V+V^0 \end{bmatrix}, \tag{10}$$

which is again of the form

$$A\mathbf{r}_\xi + B\mathbf{r}_\eta = \mathbf{f}. \tag{11}$$

Here B^{-1} exists if $x_\xi^2 + y_\xi^2 \neq 0$, while $B^{-1}A$ is found to be a symmetric matrix. A symmetric matrix has real eigenvalues, so again the system is hyperbolic and is suitable for marching in η.

A hyperbolic grid generation algorithm and applications. A grid generation algorithm is developed from (9) by specifying a rule for choosing V and by solving numerically with specified initial data along the body surface. In our procedure a noniterative implicit finite difference scheme is used which is centrally differenced in ξ and which is first order accurate in the marching direction η. An unconditionally stable implicit finite difference scheme is selected, so that any incremental spacing in η can be specified just so long as solution accuracy is maintained.

Let $\Delta\xi = \Delta\eta = 1$ such that $\eta = k-1$ and $\xi = j-1$. To second order numerical accuracy (11) can replace (9) and be differenced as

$$\mathbf{r}_{j,k+1} - \mathbf{r}_{j,k} + B^{-1}A\frac{\mathbf{r}_{j+1,k+1} - \mathbf{r}_{j-1,k+1}}{2} = B^{-1}\mathbf{f}_{j,k+1} + \varepsilon_e(\nabla_j\Delta_j)^2\mathbf{r}_{j,k}, \tag{12}$$

where coefficients x_ξ^0, y_ξ^0, x_η^0, y_η^0, and V^0 of (10) are evaluated at the previous level k; e.g., $(V^0)_{k+1} = V_k$. The term $(\nabla_j\Delta_j)^2\mathbf{r}_{j,k}$ is an added, fourth order numerical dissipation term. The terms x_η^0 and y_η^0 are obtained by solving (9), that is,

$$x_\eta^0 = -\frac{y_\xi^0 V^0}{(x_\xi^0)^2 + (y_\xi^0)^2}, \tag{13a}$$

$$y_\eta^0 = \frac{x_\xi^0 V^0}{(x_\xi^0)^2 + (y_\xi^0)^2}, \tag{13b}$$

while x_ξ^0 and y_ξ^0 are obtained from central differences

$$x_\xi^0 = \frac{x_{j+1,k} - x_{j-1,k}}{2}, \tag{14a}$$

$$y_\xi^0 = \frac{y_{j+1,k} - y_{j-1,k}}{2}, \tag{14b}$$

That the nonlinearity of the equations is maintained is clearly evident from (13a) and (13b).

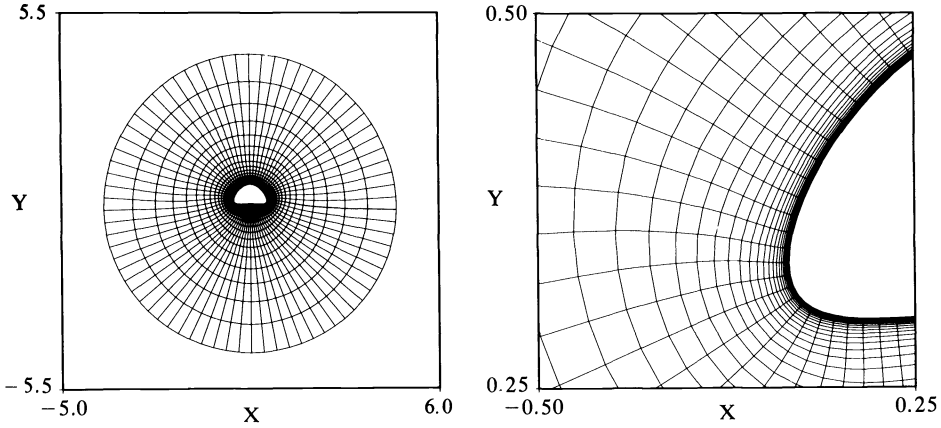
For a specified cell volume V and initial data, the difference equations (12) are easily solved for each new $k+1$ index by inverting a block tridiagonal, with 2×2 blocks, for all points in j. The nonlinear implicit finite difference algorithm used here is more thoroughly described, for example, in [8]. There the interested reader can find extensions for three space dimensions, higher order accuracy, and a discussion as to why numerical dissipation is added to the algorithm.

Various ways of defining the cell volumes are possible, but a simple procedure over which the user has fairly good intuitive control has been devised. Define polar coordinates, θ and R, with variable spacing in $\theta(\xi)$ and $R(\eta)$ about a circle whose circumference is the arc length of the body to be meshed. Distribute points on this circle

with exactly the same arc length spacing with which they are defined on the body. This determines $\theta_j = \theta(\xi)$. Along the radial lines pick a desirable spacing $R_k = R(\eta)$. In many aerodynamic applications, for example, an exponential clustering is used:

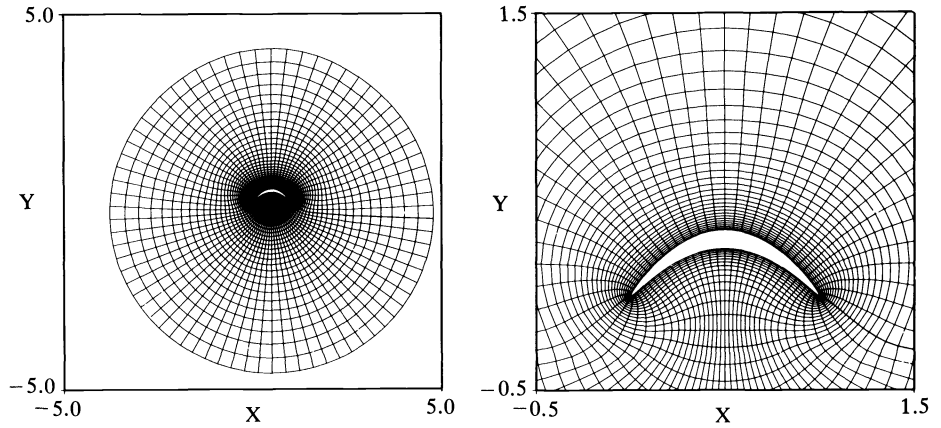
$$R_k = R_{k-1} + (R_2 - R_1)(1+\varepsilon)^{k-2}, \qquad k = 2, 3, 4, \cdots,$$

with ε, R_1 and R_2 specified. The polar grid lines then define reference cell volumes, $V^*_{j,k} = (R^2_{k+1} - R^2_k)(\theta_{j+1} - \theta_{j-1})/4$ which can be used to define $V_{j,k}$.



a) *Overview.* b) *Grid detail near corner.*

FIG. 2. *Viscous grid generated about typical aircraft-fuselage cross section.*



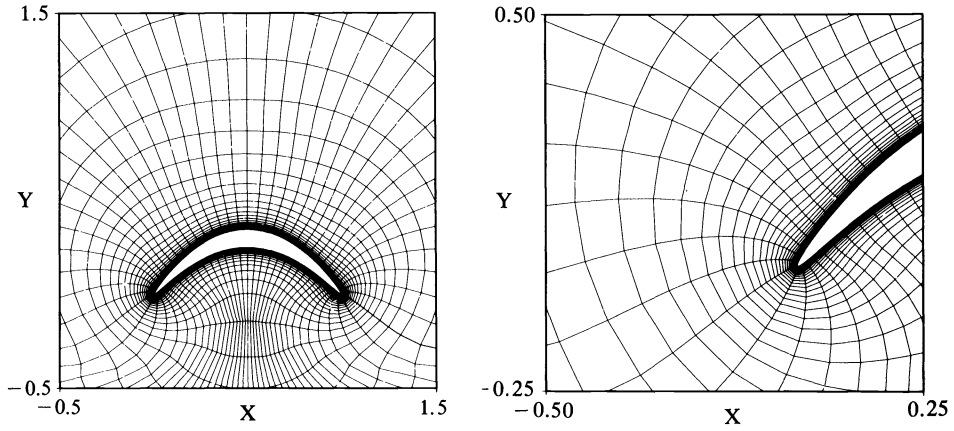
a) *Overview.* b) *Grid detail of blade.*

FIG. 3. *Inviscid grid generated about highly cambered airfoil or turbine blade.*

The grids shown in Figs. 2 to 5 were generated by combining the above cell volume $V^*_{j,k}$ with another reference cell volume, $V'_{j,k}$, which is defined precisely like $V^*_{j,k}$ but with θ_j uniformly spaced. The cell volume is then written as

$$V = \alpha V^* + (1-\alpha)V', \qquad \alpha = (1-\varepsilon)^{k-2}.$$

Grids suitable for both inviscid and viscous flow calculations were generated. The viscous flow grids (Figs. 2 and 4) are distinguished by the use of extremely fine η-grid



a) *Grid detail near body.* b) *Grid detail near leading edge.*

FIG. 4. *Viscous grid generated about highly cambered airfoil or turbine blade.*

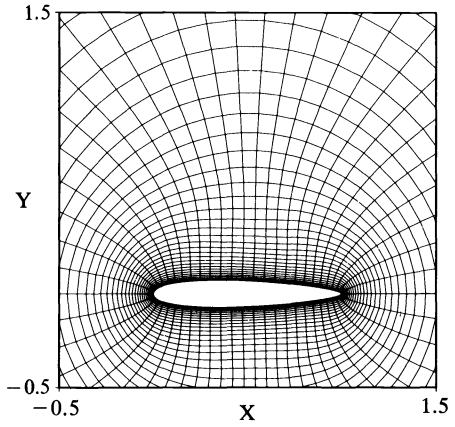


FIG. 5. *Grid generated about symmetric airfoil.*

spacing at the body. In all cases the grid spacing in η near the body is uniform and is essentially equal to the specified value $R_2 - R_1$. Thus, very good control of grid line spacing is maintained near the body. Away from the body, the spacing in η can distort significantly from R_k. In Figs. 3 and 4 the convex curvature on the lower surface tends to force the grid lines to coalesce, but since $V = J^{-1}$ is finite, the spacing in η is forced to grow rapidly and thus a singular grid ($J = 0$) is avoided in all but very severe cases. This feature is inherent to the specified volume scheme and is its most advantageous property. The grid generation procedure here is very fast, and is typically equal to about one iterative sweep of a line relaxation method used to generate a grid with elliptic equations.

Once the grid is generated, transformation metrics can be evaluated by a differencing scheme that is compatible with or identical to the differencing scheme used for the physical quantities. As noted in [9], exact evaluation of the metric quantities need not lead to the most accurate numerical solution of the physical problem, but many

numerical schemes are enhanced if the grid has smooth variation. We remark that the use of orthogonal grids does not offer any computational advantage to our particular flow solver technique [9]. Consequently, we prefer to generate the grid by using a low-order accurate numerical scheme, as this enhances smoothness. If orthogonality is an advantageous property, however, one could generate the grid using higher-order accurate differencing in the marching direction than what was used here (cf. [8] for such schemes). As an alternative, one could also ensure orthogonality by using the existing procedure to generate a much finer grid than what is required, and then discard unwanted intermediate grid points. Even with use of the current procedure in a calculation such as shown in Fig. 2, the departure from orthogonality as measured by $\{2|x_\xi x_\eta + y_\xi y_\eta|/[(x_\xi^2 + y_\xi^2)+(x_\eta^2 + y_\eta^2)]\}$ is everywhere less than 4%. The grid shown in Fig. 3 has a similar departure from orthogonality save for those points immediately adjacent to the leading and trailing edges. There the measured error at several points is as high as 20%.

Conclusions. By interpreting previous work, hyperbolic grid generation procedures are formulated in the style of the elliptic partial differential equation schemes and are used to form body-fitted meshes. For problems in which the outer boundary is not constrained, the hyperbolic scheme can be used to efficiently generate smoothly varying grids with good stepsize control near the body. Although only two-dimensional applications are presented, the basic concepts should extend to three dimensions.

Acknowledgments. This work was stimulated by independent discussions with Frank Thames of the NASA Langley Research Center and Harry Dwyer of the University of California, Davis.

REFERENCES

[1] R. A. GRAVES, JR., *Application of a numerical orthogonal coordinate generator to axisymmetric blunt bodies*, NASA TM 80131, 1979.

[2] W. D. MCNALLY, *FORTRAN program for generating a two-dimensional orthogonal mesh between two arbitrary boundaries*, NASA TN D-6766, 1972.

[3] A. M. WINSLOW, *Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh*, J. Comp. Phys., 2 (1967), pp. 149–172.

[4] W. H. CHU, *Development of a general finite difference approximation for a general domain*, J. Comp. Phys., 8 (1971), pp. 392–408.

[5] S. K. GODUNOV AND G. P. PROKOPOV, *The use of moving meshes in gas-dynamical computations*, USSR Comput. Math. Math. Phys., 12, 2 (1972), pp. 182–195.

[6] J. F. THOMPSON, F. C. THAMES AND C. M. MASTIN, *Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies*, J. Comp. Phys., 15 (1974), pp. 299–319.

[7] P. GARABEDIAN, *Partial Differential Equations*, John Wiley, New York, 1964.

[8] R. F. WARMING AND R. M. BEAM, *On the construction and application of implicit factored schemes for conservation laws*, SIAM-AMS Proceedings, 11, American Mathematical Society, Providence, RI, 1978, pp 85–129.

[9] J. L. STEGER, *Implicit finite-difference simulation of flow about arbitrary two-dimensional geometries*, AIAA J., 16 (1978), pp. 679–686.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/81/0104-0005 $01.00/0

DISTRIBUTION OF QUADRATIC FORMS IN NORMAL RANDOM VARIABLES—EVALUATION BY NUMERICAL INTEGRATION*

S. O. RICE†

Abstract. The problem of calculating the distribution function of a general quadratic form in normal random variables is examined. Two numerical integration methods for inverting the characteristic function are presented. Both make use of paths of integration that pass through, or near to, a suitable saddle-point. It is assumed that a computer is available for the calculation of functions of complex variables and for the performance of various matrix computations. Approximations for special cases are stated and examples are given.

Key Words. probability distribution, quadratic form, numerical integration

1. Introduction.

The distribution of the quadratic form

$$y = \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk} z_j z_k, \qquad a_{jk} = a_{kj}, \tag{1}$$

where the z_j's are correlated normal random variables, has been extensively studied. A survey of the subject is given by Johnson and Kotz in [1, Chapt. 29]. Here we shall be concerned with the problem of calculating the distribution of y by using numerical integration to invert the characteristic function.

When we set A equal to the matrix (a_{jk}), (1) can be rewritten as

$$y = z'Az, \tag{2}$$

where z is the column matrix (z_j) and z' is its transpose. Let the probability density of z be

$$\hat{p}(z_1, z_2, \cdots, z_n) = (2\pi)^{-n/2} |V|^{-1/2} \exp\left[-\tfrac{1}{2}(z-\xi)' V^{-1}(z-\xi)\right], \tag{3}$$

where V is the covariance matrix, $|V|$ is its determinant and ξ is the column matrix of the means $\xi_1, \xi_2, \cdots, \xi_n$. By using matrix theory Johnson and Kotz [1] show that the characteristic function $G(it) = E[\exp(ity)]$ is, with $u = it$,

$$G(u) = \exp\left\{ \sum_{j=1}^{n} \tfrac{1}{2}\omega_j^2 [(1-2u\lambda_j)^{-1} - 1] \right\} \prod_{j=1}^{n} (1-2u\lambda_j)^{-1/2}. \tag{4}$$

Here λ_j is the jth eigenvalue of the matrix VA and ω_j depends upon V, A and ξ in the manner shown in Appendix A. The values of λ_j and ω_j are real. Throughout the paper it is assumed that $\lambda_1 \geqq \lambda_2 \geqq \cdots \geqq \lambda_n$ and that no λ_j is zero.

It can be shown that the distribution of y is the same as that of the random variable

$$\sum_{j=1}^{n} \lambda_j (W_j - \omega_j)^2, \tag{5}$$

where the W_j's are mutually independent normal random variables with unit variance and zero mean.

* Received by the editors August 9, 1979, and in revised form September 15, 1980.

† Department of Applied Physics & Information Science, University of California, San Diego, La Jolla, California 92093.

The integrals that we desire to evaluate are

$$p(y)=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty} e^{-uy+\phi(u)}\,du, \tag{6}$$

$$Q(y)=\int_{y}^{\infty} p(y')\,dy'=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\frac{e^{-uy+\phi(u)}\,du}{u}, \tag{7}$$

where $p(y)$ is the probability density of y, the path of integration in (7) is indented towards the right at $u=0$ and we have set

$$\begin{aligned}\phi(u)&=\ln G(u)\\&=\sum_{j=1}^{n}[\tfrac{1}{2}\omega_j^2(1-2u\lambda_j)^{-1}-\tfrac{1}{2}\omega_j^2-\tfrac{1}{2}\ln(1-2u\lambda_j)],\end{aligned} \tag{8}$$

in which $\arg(1-2u\lambda_j)=0$ at $u=0$.

A number of methods of calculating $p(y)$ and $Q(y)$ are discussed in [1]. Recent papers aimed at obtaining numerical results have been published by Davies and by Sheil and O'Muircheartaigh. Davies [2] is concerned with the numerical integration of an integral equivalent to (7), and Sheil and O'Muircheartaigh [3] give an algorithm for the calculation of $p(y)$ and $Q(y)$ that is based on an infinite series of central χ^2 distribution functions. Reference [3] treats the case of positive definite forms and a forthcoming paper by Davies [13] is concerned with the more general case.

When the integrands in (6) and (7) decrease rapidly, numerical integration along a path parallel to the imaginary u-axis becomes feasible. This method is described in § 3. When $Q(y)$ is not close to 0 or 1, the method becomes similar to the one proposed by Davies [2].

If the integrands in (6) and (7) decrease slowly, as when n is small, straightforward numerical integration becomes costly. Grad and Solomon [4] and Slepian [5] overcame this difficulty (for definite quadratic forms and $\xi=0$) by using contour integration to express $p(y)$ as a sum of tractable integrals. Their work was extended by Johnson and Kotz [1] to obtain similar sums when the quadratic form (1) is indefinite and $\xi=0$.

In § 5 we present a different method of dealing with the slow convergence. The path of integration is tilted in such a way that the constant amplitude oscillations of $\exp(-uy)$ are converted into exponentially damped oscillations. This method works well for general quadratic forms when $|y|$ is not too small. An example is given in § 6.

The appendices contain auxiliary information regarding the computations. The items listed are either known or are straightforward extensions of known results.

2. The initial point u_0 and the saddle-point u_1. Let the paths of integration in (6) and (7) be displaced (but not far enough to pass over any of the points $u=1/(2\lambda_j)$) so that the paths run from $u_0-i\infty$ to $u_0+i\infty$, where u_0 is a point on the real u-axis. Then, from the symmetry of $\phi(u)$ about the real u-axis,

$$p(y)=\text{Real}\,\frac{1}{\pi i}\int_{u_0}^{u_0+i\infty} e^{-uy+\phi(u)}\,du, \tag{9}$$

$$Q(y)=\text{Real}\,\frac{1}{\pi i}\int_{u_0}^{u_0+i\infty}\frac{e^{-uy+\phi(u)}\,du}{u}, \qquad u_0>0. \tag{10}$$

When $u_0<0$ the contribution ($=1$) of the pole at $u=0$ must be added to the right-hand side of (10).

A good choice of u_0 is the appropriate saddle-point u_1 of $\exp[-uy+\phi(u)]$, i.e., the appropriate root of

$$y=\frac{d\phi(u)}{du}. \tag{11}$$

Differentiation of the expression (8) for $\phi(u)$ shows that the right-hand side of (11) is the case $l=1$ of

$$\left(\frac{d}{du}\right)^l\phi(u)=\sum_{j=1}^{n}\left[\tfrac{1}{2}\omega_j^2 l!(1-2u\lambda_j)^{-l-1}+\tfrac{1}{2}(l-1)!(1-2u\lambda_j)^{-l}\right](2\lambda_j)^l. \tag{12}$$

If y is equal to $E(y)$, u_1 is zero. When $\lambda_1>0$ and $\lambda_n<0$, u_1 runs from $1/(2\lambda_n)$ to $1/(2\lambda_1)$ as y runs from $-\infty$ to $+\infty$. When both λ_1 and λ_n are positive, u_1 runs from $-\infty$ to $1/(2\lambda_1)$ as y runs from 0 to $+\infty$.

When y is arbitrary the corresponding u_1 can always be obtained by solving (11) numerically. However it is not necessary to know u_0 exactly because (9) and (10) are suited to numerical integration even if u_0 is only close to u_1. One way of obtaining a good value of u_0 is to first calculate a short table of $d\phi(u)/du$ for values of u lying in the range of u_1. A table of this sort is equivalent to a table of solutions of (11) and can be used to obtain an estimate of the u_1 corresponding to a given y. This estimate of u_1 can then be used as the u_0 in (9) and (10).

3. First integration method—path parallel to imaginary *u*-axis. When the integrand decreases rapidly and u_0 is close to u_1, the trapezoidal rule can be applied directly. The first step is to calculate a suitable value of u_0 as discussed in § 2. Then the change of variable $u=u_0+ibv$ converts (9) and (10) (with the help of (7)) into

$$p(y)=\int_0^\infty \text{Real}\left[\left(\frac{b}{\pi}\right)e^{-uy+\phi(u)}\right]dv, \tag{13}$$

$$Q(y)=\begin{Bmatrix}0, & u_0>0\\ \tfrac{1}{2}, & u_0=0\\ 1, & u_0<0\end{Bmatrix}+\int_0^\infty \text{Real}\left[\left(\frac{b}{\pi}\right)u^{-1}e^{-uy+\phi(u)}\right]dv, \qquad u=u_0+ibv. \tag{14}$$

We shall take b to be

$$b=\left[\frac{2}{\phi''(u_0)}\right]^{1/2}, \tag{15}$$

where $\phi''(u_0)$ is the value of $(d/du)^2\phi(u)$ at u_0.

Two forms of the trapezoidal rule are available, one in which the integral of $f(v)$ from $v=0$ to ∞ is approximated by $h[\frac{1}{2}f(0)+f(h)+\cdots]$ and the other in which it is approximated by $h[f(h/2)+f(3h/2)+\cdots]$, $h=\Delta v$ being the spacing. When applied to either (13) or (14) the errors in the two approximations are of about the same magnitude but of opposite sign. Since the second form is more convenient for $Q(y)$, we shall base the numerical integration upon the equations

$$p(y)=h\sum_{n=0}^{\infty}\left[\text{Real}\,\frac{b}{\pi}e^{-uy+\phi(u)}\right]-E_1, \tag{16}$$

$$Q(y)=\frac{1}{1+\exp[2\pi u_0/(bh)]}+h\sum_{n=0}^{\infty}\left[\text{Real}\,\frac{b}{\pi}\frac{1}{u}e^{-uy+\phi(u)}\right]-E_2, \tag{17}$$

$$u=u_0+ibh(n+\tfrac{1}{2}),$$

which follow from (13) and (14). The error terms E_1 and E_2 and the first term on the right in (17) are discussed in Appendix C.

To use (16) and (17) we ignore E_1 and E_2 and compute the sums for several descending values of h. When b has the value given in (15) good first trial values of h are $h = 1.0, 0.5, 0.25$. Selection of a truncation point is aided by the fact that $|\exp[-uy + \phi(u)]|$ usually decreases steadily.

When $u_0 = 0$ (16) and (17) can be used efficiently for values of y such that $Q(y)$ is not close to 0 or 1. Equation (17), with $u_0 = 0$, $b = 1$ and a different method of choosing h, is equivalent to the one used for numerical integration by Davies [2].

4. Example illustrating the method of § 3. To illustrate the numerical integration method discussed in § 3 we consider a characteristic function used by Dillard and Rickard [6] in connection with a detection problem. For their problem, our function (8) for $\phi(u)$ becomes

$$\phi(u) = \sum_{k=1}^{N} [c_k(1-2u\lambda_k)^{-1} - c_k - \ln(1-2u\lambda_k)], \tag{21}$$

where

$$\lambda_k = 2 + 2\cos\left[\frac{k\pi}{(N+1)}\right], \qquad k = 1, 2, \cdots, N. \tag{22}$$

We shall take $N = 25$ and $c_k = 0.2$, $k = 1, 2, \cdots, N$.

The first step is to determine values of u_0 to be used in (16) and (17). We can either set $u_0 = u_1$ where u_1 is the saddle-point obtained by solving $y = \phi'(u)$ (eq. (11)) numerically, or we can obtain $u_0 \approx u_1$ from a short table of $\phi'(u)$. We shall use the second method and calculate a table of the first and second derivatives of $\phi(u)$ for values of u that lie in the range of u_1. From § 2 this range is $-\infty < u < 1/(2\lambda_1) = 0.1255$. Typical entries might be:

(23)

u	0.08	.03	0.0	−0.05	−0.20
$\phi'(u)$	295.679	152.214	120.000	89.811	52.682
$\phi''(u)$	5998.8	1394.5	828.2	441.4	140.3

The entry for $u = 0$ shows that

$$E(y) = \phi'(0) = 120.0, \qquad \text{Var}(y) = \phi''(0) = 828.2. \tag{24}$$

The entry for $u = 0.08$ shows that $u_1 = 0.08$ is the saddle-point of $\exp[-uy + \phi(u)]$ when $y = 295.679$. Furthermore, if we take u_0 to be 0.08 in (16) and (17) for $p(y)$ and $Q(y)$, the parameter b is given by

$$\left[\frac{2}{\phi''(u_0)}\right]^{1/2} = \left[\frac{2}{5998.8}\right]^{1/2} = 0.0183.$$

Substituting these values of u_0, y, and b in (16) and (17) (with E_1 and E_2 deleted) and performing the summations gives the first four columns in Table 1.

TABLE 1

$u_0 = 0.08$, $y = 295.678$				$Q(y)$, $y = 52.682$	
h	M	$p(y)$	$Q(y)$	$u_0 = -0.20$	$u_0 = 0.0$
1.0	12	.4804 5093 (−6)	.5630 9403 (−5)	.9986 7186	.9694 8460
0.5	24	.4813 2785 (−6)	.5639 1427 (−5)	.9986 8994	.9986 8807
0.25	48	.4813 2788 (−6)	.5639 1429 (−5)	.9986 8994	.9986 8994

Here M is the number of terms used in the summations.

In § 3 it was mentioned that numerical integration using (17) with $u_0 = 0$ works well provided $Q(y)$ is not close to 0 or 1. The last two columns in Table 1 show that $Q(52.682) = 0.9986\ 8994$. It is seen that the performance of (17) using $u_0 = 0$ is beginning to deteriorate (compared to that obtained by using $u_0 \approx u_1$), but it is still quite good.

Calculations using $u_0 \approx u_1$ in (16) and (17) for other values of y show essentially the same rate of convergence as in Table 1.

The asymptotic approximations for $p(y)$ and $Q(y)$ stated in Appendix F were computed for the values of $\phi'(u_1) = y$ listed in (23). The values given by the approximations agree with the numerical integration values to within at least three significant figures in all of the cases.

5. Second integration method—tilting the path. When n is small the integrals for $p(y)$ and $Q(y)$ usually converge slowly. In this case, if $y \neq 0$, we can transform the integrals (9) and (10) by setting

$$u = u_0 + sv, \qquad du/dv = s,$$

$$s = \begin{cases} \dfrac{(1+i\sqrt{3})}{y}, & y > 0, \\ \dfrac{(1-i\sqrt{3})}{y}, & y < 0. \end{cases} \tag{25}$$

Jordan's lemma shows that we can take v to run from 0 to ∞. The value of s stated in (25) causes the new path of integration in the u-plane to make an angle of 60° with the real u-axis so that now the absolute values of the integrands decrease exponentially. The further transformation (see [7, § VI]),

$$v = \exp(x - e^{-x}), \qquad \frac{dv}{dx} = v(1 + e^{-x}), \tag{26}$$

and elimination of v puts the integral (9) in a form suited to evaluation by the trapezoidal rule:

$$p(y) = \text{Real}\,\frac{s}{\pi i}\int_{-\infty}^{\infty} \exp\left[-uy + \phi(u) + x - e^{-x}\right](1 + e^{-x})\,dx, \tag{27}$$

$$u = u_0 + s\exp(x - e^{-x}), \tag{28}$$

where s is given by (25).

Dividing the integrand in (27) by u gives the corresponding integral for $Q(y)$ when $u_0 > 0$ (when $u_0 < 0$, the integral is equal to $Q(y) - 1$).

These integrals for $p(y)$ and $Q(y)$ are suited to machine calculation because it is easy to get u from x. When the trapezoidal rule is used for several decreasing values of $h = \Delta x$, the convergence to the exact values is quite rapid if y is not too small. Good first trial values are $h = 0.3, 0.15, 0.075$, with truncation at $x = \pm 3.3$. If $\sum \ln(1 - 2u\lambda_j)$ in the expression (8) for $\phi(u)$ is calculated by first writing it as $\ln P(u)$ and then calculating $P(u) = \prod (1 - 2u\lambda_j)$, care must be taken to get the correct value of $\operatorname{Im}[\ln P(u)]$.

6. Example illustrating the method of § 5. Suppose that we want to calculate the probability density of $y = 2z_1z_2$, where z_1 and z_2 are normal random variables with respective means 3, 5, variances 1, 4 and correlation coefficient $\frac{1}{2}$. Then

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}, \quad \xi = \begin{bmatrix} 3 \\ 5 \end{bmatrix}. \tag{29}$$

The formulas of Appendix A yield the matrices

$$L = \begin{bmatrix} 1 & 0 \\ 1 & \sqrt{3} \end{bmatrix}, \quad L'AL = \begin{bmatrix} 2 & \sqrt{3} \\ \sqrt{3} & 0 \end{bmatrix}, \quad P = \tfrac{1}{2}\begin{bmatrix} \sqrt{3} & -1 \\ 1 & \sqrt{3} \end{bmatrix}, \tag{30}$$

from which we get

$$\begin{aligned} \lambda_1 &= 3, & \lambda_2 &= -1, \\ \omega_1^2 &= \tfrac{121}{12}, & \omega_2^2 &= \tfrac{1}{4}. \end{aligned} \tag{31}$$

The function $\phi(u)$ that we have to work with is obtained by setting these values in (8) with $n = 2$.

The first step in the numerical evaluation of (27) for $p(y)$ and its mate for $Q(y)$ is to calculate a set of values of u_0. As in § 4, we elect to use the method based on a table of $\phi'(u)$ for values of u that lie in the range of the saddle-point u_1. From § 2 this range is $1/(2\lambda_2) = -0.5 < u_1 < 1/(2\lambda_1) = 0.166$. Typical entries might be

u	0.10	0.05	0.0	−0.30	−0.35
$\phi'(u)$	195.6	64.9	32.0	0.87	−1.996,

(32)

where, for instance, $u_1 = 0.10$ is the appropriate saddle-point of $\exp[-uy + \phi(u)]$ when $y = 195.6$.

Suppose that we are interested in calculating $p(y)$ and $Q(y)$ for $y = 100$ and $y = 0.005$. From (32) we see that u_1 for $y = \phi'(u_1) = 100$ lies between 0.10 and 0.05. Since good numerical integration results are obtained even if u_0 is only close to u_1, we arbitrarily choose $u_0 = 0.07$. Similarly for $y = 0.005$ we choose $u_0 = -0.32$. When we use these values of u_0 in (28) and apply the trapezoidal rule to (27) and its mate for $Q(y)$, we get the values shown in Table 2.

TABLE 2

h	N	$p(100)$	$Q(100)$	$p(.005)$	$1 - Q(.005)$
.3	23	.3593 73 (−3)	.4257 12 (−2)	.6894 50 (−2)	.1323 01 (−1)
.15	45	.3594 02 (−3)	.4257 34 (−2)	.6242 53 (−2)	.7094 08 (−2)
.075	89	.3594 02 (−3)	.4257 34 (−2)	.6335 14 (−2)	.7157 60 (−2)
.05	133	–	–	.6335 13 (−2)	.7152 28 (−2)
.0375	177	–	–	.6335 13 (−2)	.7152 28 (−2)

Here we have used the form $h[\cdots+f(-h)+f(0)+f(h)+\cdots]$ of the trapezoidal rule with $h=\Delta x$ and truncation at $x=\pm 3.3$. N is the number of points used.

The values of $h=0.05$ and 0.0375 used for $y=0.005$ in Table 2 show that values of h smaller than the recommended 0.3, 0.15, 0.075 are necessary when y is small. As $|y|$ increases, the larger values of h suffice. For example, at $y=0.87$ the values of $p(y)$ and $Q(y)$ computed by using $h=0.15$ and $h=0.075$ (with $u_0=-0.30$) agree to within six figures, just as for $y=100$ (see the third and fourth columns in Table 2).

When $p(y)$ is computed by numerical integration for a number of values of y and the results plotted, the curve that is obtained resembles the curve of a simple unimodal probability with $E(y)=32$ and Var $(y)=384$ (see Appendix B) except for something peculiar at $y=0$. Reference to Appendix D shows that the peculiarity consists of a logarithmic spike at $y=0$. Calculations based on the equations in Appendix D show that when $|y|$ is small

$$p(y)=-0.000524\ln|y|+\psi(y), \tag{33}$$

where $\psi(y)$ is continuous at $y=0$ and $\psi(0)=0.003537$. Replacing $\psi(y)$ in (33) by $\psi(0)$ and setting $y=0.005$ gives the approximation $p(0.005)\approx 0.006314$. This is to be compared with the "exact" value 0.006335 in Table 2.

Setting $n=2$ and $m=1$ in the equations stated in Appendix E shows that when $|y|\to\infty$

$$p(y)\to 0.000625y^{-1/2}\exp\left[1.833y^{1/2}-\frac{y}{6}\right], \qquad y\to\infty, \tag{34}$$

$$p(y)\to 0.00201(-y)^{-1/2}\exp\left[0.5(-y)^{1/2}+\frac{y}{2}\right], \qquad y\to-\infty. \tag{35}$$

The error in (34) is about ten percent at $y=100$.

The asymptotic approximation for $p(y)$ stated in Appendix F gives three accurate significant figures at $y=20$, and its accuracy increases as y increases beyond 20.

Appendix A—calculation of λ_j and ω_j^2. The values of λ_j and ω_j^2 needed in (4) and (8) can be calculated (see [1]) by first decomposing V into the product $V=LL'$ (Cholesky decomposition) where all of the elements of L lying above the principal diagonal are zero. Then λ_j is the jth eigenvalue of $L'AL$ and ω_j^2 can be obtained by squaring the jth element in

$$\omega=P'L^{-1}\xi. \tag{A.1}$$

Here P' is the transpose of the orthogonal matrix P formed by setting the eigencolumns (eigenvectors because $L'AL$ is symmetric) side by side.

The calculation of L can be avoided at the cost of dealing with unsymmetrical matrices by using the fact that λ_j is also the jth eigenvalue of VA and that

$$\omega_j^2=(j\text{th element in the row }\xi'V^{-1}\hat{P})\times(j\text{th element in the column }\hat{P}^{-1}\xi), \tag{A.2}$$

where $\hat{P}$ is the square matrix formed by putting the eigencolumns of VA side by side.

Appendix B—cumulants. The kth cumulant of the distribution specified by the characteristic function (4) is [1]

$$\kappa_k=2^{k-1}(k-1)!\sum_{j=1}^{n}(k\omega_j^2+1)\lambda_j^k. \tag{B.1}$$

In particular κ_1 is the expected value of y and κ_2 is its variance. We also have

$$\kappa_k = 2^{k-1}k!\,\xi' A(VA)^{k-1}\xi + 2^{k-1}(k-1)!\,\text{Trace}\,(VA)^k, \tag{B.2}$$

which can be expressed as a multiple sum by using the rule for matrix multiplication. Equation (B.2) can be proved with the help of (A.2) and $VA = \hat{P}\Lambda\hat{P}^{-1}$ where $\Lambda = \text{diag}\,(\lambda_1, \lambda_2, \cdots, \lambda_n)$.

Appendix C—the trapezoidal rule in § 3. Equation (17) for $Q(y)$ can be obtained by first assuming $u_0 > 0$ and rewriting (14) as

$$\begin{aligned} Q(y) &= \frac{1}{2}\int_{-\infty}^{\infty}\left[\frac{b}{\pi}\frac{1}{u}e^{-uy+\phi(u)}\right]_{u=u_0+ib(v+\delta)} dv \\ &= \frac{1}{2}\int_{-\infty}^{\infty} f(v)\,dv, \end{aligned} \tag{C.1}$$

where δ is an arbitrary real parameter and $f(v)$ denotes the integrand in the first line. Let

$$\xi_k = y - 2\pi k/(bh),$$
$$q = 2\pi(u_0 + ib\delta)/(bh).$$

Consider the Poisson sum formula in which the $k = 0$ term is the integral (C.1). It can be put in the form

$$\begin{aligned} Q(y) &= \frac{1}{2}h\sum_{h=-\infty}^{\infty} f(nh) - \sum_{k=-\infty}^{\infty}{}' \frac{1}{2}\int_{-\infty}^{\infty} f(v)\exp\left(\frac{i2\pi vk}{h}\right) dv \\ &= \frac{1}{2}h\sum_{n=-\infty}^{\infty} f(nh) - \sum_{k=-\infty}^{\infty}{}' Q(\xi_k)\,e^{-qk}, \end{aligned} \tag{C.2}$$

where the prime on $\sum'$ indicates that the term for $k = 0$ is omitted. When we write $Q(\xi_k)$ as $1 - [1 - Q(\xi_k)]$ in the terms for $k > 0$, we can sum $\exp(-qk)$ from $k = 1$ to ∞ and get

$$Q(y) = \frac{1}{(1-e^q)} + \frac{1}{2}h\sum_{n=-\infty}^{\infty} f(nh) - \sum_{k=1}^{\infty}\left(Q(\xi_{-k})\,e^{qk} - [1 - Q(\xi_k)]\,e^{-qk}\right). \tag{C.3}$$

If we start with $u_0 < 0$ instead of $u_0 > 0$ we again get (C.3) ($Q(y)$ in (C.1) and (C.2) is replaced by $Q(y) - 1$ and $Q(\xi_k)$ in (C.2) by $Q(\xi_k) - 1$).

When $\delta = h/2$, e^q becomes $-\exp[2\pi u_0/(bh)]$ and (C.3) can be put in the form of (17) in which E_2 is equal to the second summation in (C.3).

When $u_0 = 0$ and $b = 1$, the real part of (C.3) becomes an equation given by Davies [2].

An expression for E_1 in (16) for $p(y)$ can be obtained by differentiating E_2 with respect to y, or directly from the Poisson sum formula (see § VII of [8]). The result is

$$E_1 = \sum_{k=-\infty}^{\infty}{}' (-1)^k p\left(y - \frac{2\pi k}{bh}\right) e^{-2\pi u_0 k/(bh)}. \tag{C.4}$$

If we set $\delta = 0$ in (C.3) it becomes

$$\begin{aligned} Q(y) &= \frac{1}{1-e^q} + h\sum_{n=0}^{\infty}{}' \varepsilon_n\,\text{Real}\left[\frac{b}{\pi}\frac{1}{u}e^{-uy+\phi(u)}\right] \\ &\quad - \sum_{k=1}^{\infty}\left(Q(\xi_{-k})e^{qk} - [1 - Q(\xi_k)]\,e^{-qk}\right), \end{aligned} \tag{C.5}$$

where now $q=2\pi u_0/(bh)$, $u=u_0+ibhn$ and $\varepsilon_n=1$ if $n>0$ and $\varepsilon_0=\frac{1}{2}$. When $u_0\to 0$ both $1/(1-e^q)$ and the $n=0$ term in (C.5) become infinite, but in the limit their sum becomes

$$\tfrac{1}{2}+[E(y)-y]bh/(2\pi). \tag{C.6}$$

Equation (C.5) for $Q(y)$ corresponds to the form $h[\frac{1}{2}f(0)+f(h)+\cdots]$ of the trapezoidal rule in the same way as (17) corresponds to the form $h[f(h/2)+f(3h/2)+\cdots]$.

Appendix D—behavior of $p(y)$ near $y=0$. Let E_n (no relation to E_n in Appendix C) be defined by

$$E_n \leqq |\lambda_1\lambda_2\cdots\lambda_n|^{-1/2}\exp\left[-\sum_{j=1}^{n}\tfrac{1}{2}\omega_j^2\right]. \tag{D.1}$$

Then, near $y=0$:

(a) When $n=2$ and $\lambda_1>0$, $\lambda_2<0$,

$$p(y)=-\frac{1}{2\pi}E_2\ln|y|+\psi(y), \tag{D.2}$$

where $\psi(y)$ is continuous at $y=0$ and has the value

$$\psi(0)=\frac{E_2}{2\pi}\left[\ln\frac{4}{B}-0.5772\cdots+\sum_{k=1}^{\infty}\frac{1}{2}\frac{(k-1)!}{(2k)!}(x_+^k+x_-^k)\right] \tag{D.3}$$

at $y=0$. Here $0.5772\cdots$ is Euler's constant, $x_\pm=A_\pm^2/B$, and

$$A_\pm=\omega_1\lambda_1^{-1/2}\pm\omega_2|\lambda_2|^{-1/2},\qquad B=(\lambda_1^{-1}+|\lambda_2|^{-1})/2.$$

One derivation of (D.3) proceeds by: (i) using (5) to express $p(y)$ as a convolution integral, (ii) integrating by parts to bring out the $\ln|y|$ term that appears in (D.2) and (iii) expanding the remaining integral in a series after setting $y=0$ and using the fact that the integral of $\exp(-u)\ln u$ from $u=0$ to ∞ is equal to Euler's constant.

(b) When $n=3$ and $\lambda_1>0$ but λ_2 and λ_3 are <0,

$$p(y)=(2\pi)^{-1/2}E_3\begin{Bmatrix}0, & y<0\\ -y^{1/2}, & y>0\end{Bmatrix}+\psi(y), \tag{D.4}$$

where $\psi(y)$ and its first derivative are continuous at $y=0$.

(c) When all of the λ_j's are positive and y is small and positive,

$$p(y)=2^{-n/2}y^{(n-2)/2}E_n/\Gamma(n/2)+O(y^{n/2}). \tag{D.5}$$

The leading term is the first term in a convergent power series given by Johnson and Kotz [1].

Appendix E—behavior of $p(y)$ near $y=\pm\infty$. Let λ_1 be positive and of multiplicity m so that $\lambda_1=\lambda_2=\cdots=\lambda_m>\lambda_{m+1}\geqq\lambda_{m+2}\geqq\cdots\geqq\lambda_n$. Also let

$$x=y(2\lambda_1)^{-1},\quad M=\sum_{j=1}^{m}\tfrac{1}{2}\omega_j^2,\quad \alpha_j=1-\lambda_j\lambda_1^{-1},$$

$$C=(2\lambda_1)^{-1}\exp\left[-\sum_{j=1}^{n}\tfrac{1}{2}\omega_j^2\right]\prod_{j=m+1}^{n}[\alpha_j^{-1/2}\exp(\tfrac{1}{2}\omega_j^2/\alpha_j)],\qquad m<n \tag{E.1}$$

$$C=(2\lambda_1)^{-1}\exp\left[-\sum_{j=1}^{n}\tfrac{1}{2}\omega_j^2\right],\qquad m=n.$$

Then, as $y \to \infty$,

$$p(y) \to C\left(\frac{x}{M}\right)^{(m-2)/4} e^{-x} I_{(m/2)-1}[2(xM)^{1/2}], \tag{E.2}$$

where $I_{(m/2)-1}[\cdot]$ is a modified Bessel function. Special cases are

$$p(y) \to Cx^{(m/2)-1} e^{-x}/\Gamma(m/2), \qquad M = 0, \quad x \to \infty. \tag{E.3}$$

$$p(y) \to C\left(\frac{x}{M}\right)^{(m-2)/4} \frac{\exp[-x + 2(xM)^{1/2}]}{2\pi^{1/2}(xM)^{1/4}}, \qquad (xM)^{1/2} \to \infty. \tag{E.4}$$

Expression (E.2) can be obtained by noting that the path of integration in the integral (6) for $p(y)$ can be deformed so that when y is large most of the contribution arises from the region around $u = 1/(2\lambda_1)$.

Let λ_n be negative and of multiplicity m. Then as $y \to -\infty$, the behavior of $p(y)$ is given by (E.2), (E.3), and (E.4) with x, M, α_j, and C redefined as

$$\begin{aligned} &x = y(2\lambda_n)^{-1}, \quad M = \sum_{j=n-m+1}^{n} \tfrac{1}{2}\omega_j^2, \quad \alpha_j = 1 - \lambda_j \lambda_n^{-1}, \\ &C = (-2\lambda_n)^{-1} \exp\left[-\sum_{j=1}^{n} \tfrac{1}{2}\omega_j^2\right] \prod_{j=1}^{n-m} [\alpha_j^{-1/2} \exp(\tfrac{1}{2}\omega_j^2/\alpha_j)]. \end{aligned} \tag{E.5}$$

Appendix F—asymptotic-like approximations. When most of the contribution to the integrals (6) and (7) (with paths of integration deformed so as to pass through u_1) comes from the region around u_1, approximations to $p(y)$ and $Q(y)$ can be obtained from asymptotic series given by Daniels [9] and Lugannani and Rice [10].

Let ϕ_l denote the value of $(d/du)^l \phi(u)$ given by (12) with $u = u_1$. Also,

$$\begin{aligned} \theta_l &= \frac{\phi_l}{(l!\phi_2^{l/2})}, \\ \mu &= \frac{1}{(u_1\phi_2^{1/2})}, \\ f_1 &= -u_1 y + \phi(u_1). \end{aligned} \tag{F.1}$$

Then the approximations are

$$\begin{aligned} p(y) &\approx (2\pi\phi_2)^{-1/2} e^{f_1}[1 - (\tfrac{15}{2}\theta_3^2 - 3\theta_4) \\ &\qquad + \tfrac{15}{8}(231\theta_3^4 - 252\theta_3^2\theta_4 + 56\theta_3\theta_5 + 28\theta_4^2 - 8\theta_6)], \\ Q(y) &\approx \tfrac{1}{2}\operatorname{erfc}(\sqrt{-f_1}) + (A_0 - B_0) + (A_1 - B_1), \end{aligned} \tag{F.2}$$

where

$$\operatorname{erfc}(x) = 2\pi^{-1/2} \int_x^\infty e^{-t^2}\, dt,$$

$$\begin{aligned} &A_0 = \mu(2\pi)^{-1/2} e^{f_1}, \qquad B_0 = e^{f_1}/(2\pi^{1/2}\sqrt{-f_1}), \\ &A_1 = -A_0[\mu^2 + 3\mu\theta_3 + (\tfrac{15}{2}\theta_3^2 - 3\theta_4)], \qquad B_1 = B_0/(2f_1), \end{aligned} \tag{F.3}$$

and $\sqrt{-f_1}$ has the same sign as u_1. When $y \to E(y)$, $u_1 \to 0$ and both A_0 and B_0 become infinite. However $A_0 - B_0$ remains finite and we have

$$Q[E(y)] \approx \tfrac{1}{2} - \theta_3/(2\pi)^{1/2}. \tag{F.4}$$

The Chernoff bound for $Q(y)$ can be expressed in terms of f_1 as

$$e^{f_1} \geqq \begin{bmatrix} Q(y) & \text{if } y > E(y) \\ 1-Q(y) & \text{if } y < E(y) \end{bmatrix}. \tag{F.5}$$

Another type of asymptotic approximation suited to the integrals (6) and (7) has been given by Helstrom [11]. His approximation makes use of continued fractions. Grenander, Pollak and Slepian [12] have given an approximation for $p(y)$ when n is large, VA is a Toeplitz matrix and ω_j is zero. Their approximation is in the form of an integral. Its derivation makes use of a result due to Szego regarding the eigenvalues of a Toeplitz matrix.

Acknowledgments. I am grateful to the referees of an earlier version of this paper for many helpful comments and suggestions. I am also indebted to Professor Murray Rosenblatt and Dr. J. T. Rickard for encouragement and helpful comments.

REFERENCES

[1] N. L. JOHNSON AND S. KOTZ, *Distributions in Statistics—Continuous Univariate Distributions*, 2, John Wiley, New York, 1970.

[2] R. B. DAVIES, *Numerical inversion of a characteristic function*, Biometrika, 60 (1973), pp. 415–417.

[3] J. SHEIL AND I. O'MUIRCHEARTAIGH, *The distribution of non-negative quadratic forms in normal variables*, Appl. Stat., 26 (1977), pp. 92–98.

[4] A. GRAD AND H. SOLOMON, *Distribution of quadratic forms and some applications*, Ann. Math. Statist., 26 (1955), pp. 464–477.

[5] D. SLEPIAN, *Fluctuations of random noise power*, Bell System Tech. J., 37 (1958), pp. 163–184.

[6] G. M. DILLARD AND J. T. RICKARD, *Performance of an* MTI *followed by incoherent integration for nonfluctuating signals*, Proc. IEEE 1980 International Radar Conference (Apr. 28–30, 1980), pp. 194–199.

[7] S. O. RICE, *Efficient evaluation of integrals of analytic functions by the trapezoidal rule*, Bell System Tech. J., 52 (1973), pp. 707–722.

[8] S. O. RICE, *Numerical evaluation of integrals with infinite limits and oscillating integrands*, Bell System Tech. J., 54 (1975), pp. 155–164.

[9] H. E. DANIELS, *Saddlepoint approximations in statistics*, Ann. Math. Stat., 25 (1954), pp. 631–650.

[10] R. LUGANNANI AND S. O. RICE, *Saddle point approximation for the distribution of the sum of independent random variables*, Adv. Appl. Prob., 12 (1980), pp. 475–490.

[11] C. HELSTROM, *Approximate evaluation of detection probabilities in radar and optical communications*, IEEE Trans. Aerospace Electro. Systems, AES-14 (1978), pp. 630–640.

[12] U. GRENANDER, H. O. POLLAK AND D. SLEPIAN, *The distribution of quadratic forms in normal variates: a small sample theory with applications to spectral analysis*, J. Soc. Ind. Appl. Math., 7 (1959), pp. 374–401.

[13] R. B. DAVIES (1980), *The distribution of a linear combination of chi-squared random variables*, Appl. Stat. (to appear).

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/80/0104-0006 $01.00/0

A NUMERICAL METHOD FOR COMPUTING THE SHAPE OF A VERTICAL SLENDER JET*

JOHN STRIKWERDA† AND JAMES GEER‡

Abstract. A numerical method is presented for computing the shape of a vertical slender jet of fluid falling steadily under the force of gravity. The problem to be solved is fomulated as a nonlinear free boundary value problem for the cross-sectional shape of the jet. The numerical method of solution treats the boundary conditions of the problem as a pair of nonlinear hyperbolic pseudo-differential equations to be integrated in the stream-wise direction. The original differential equation appears as an auxiliary condition. This formulation is shown to be well-posed. The numerical method is found to be stable and second-order accurate. Computations are presented for jets issuing from several different orifice shapes. The numerical method of solution appears to be new and may be applicable to other nonlinear free boundary value problems.

Key words. free boundary value problem, finite difference method, fluid jet, hyperbolic equations, potential flow

1. Introduction. We present in this paper a numerical method which we have used to determine the shape of the free surface of a slender jet of fluid falling vertically in the presence of gravity. The flow is assumed to be a steady, three-dimensional potential flow. The solution procedure determines the cross-sectional shape given the shape and velocity profile at a particular height (e.g., at an orifice from which the jet emanates). Surface tension and viscous effects are neglected. The mathematical formulation of the problem leads to a fully three-dimensional, nonlinear boundary value problem for Laplace's equation, for which the boundary of the flow is also unknown. For the case of a slender jet, however, Tuck [11] and Geer [2], [3] derived equations to describe the first approximations to the cross-sectional shape and velocities of the jet. The problem of determining the shape is thus reduced to solving a nonlinear two-dimensional problem in the cross-sectional plane of the jet. Both Tuck and Geer gave an exact solution to this problem for a jet with an elliptical cross-sectional shape (see also Green [5].) To date no other exact solutions have been found.

The purpose of this work is to present in some detail the method we have developed to solve numerically the associated nonlinear free boundary value problem for jets which fall vertically from an orifice of a specified shape. The problem is formulated in § 2 and then transformed into a form more suitable for numerical integration. In §§ 3 through 5, we describe the numerical method that we have used to integrate the problem outlined in § 2.

In § 6 we present the results for three of the different orifice shapes for which our calculations were made. These shapes are an ellipse, a rectangle and an equilateral triangle. The accuracy of our method is discussed in § 7, while the well-posedness and stability of the method are discussed in § 8.

The numerical method presented here appears to be new and may be applicable to other three-dimensional free boundary value problems. The usefulness of most existing numerical methods for solving free boundary value problems is restricted to one and two dimensions (see Wilson et al. [12]).

* Received by the editors June 20, 1980.

† Department of Computer Sciences, University of Wisconsin-Madison, Madison, Wisconsin 53706. The work of this author was supported by the National Aeronautics and Space Administration under contract NAS1-15810 while he was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

‡ State University of New York at Binghamton, Binghamton, New York. The work of this author was supported in part by the Research Foundation of SUNY under contract 240-6135A and in part by the National Aeronautics and Space Administration under contract NAS1-15810.

2. Formulation of the problem. Let the velocity potential of the jet be denoted by $\Phi = \Phi(r, \theta, z; \varepsilon)$ and let the shape of the free surface of the jet be described by $r = \mathcal{S}(\theta, z; \varepsilon)$ (see Fig. 1). Here r, θ and z form the usual (nondimensional) cylindrical

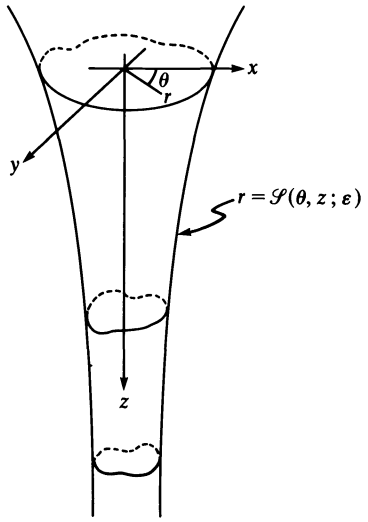


FIG. 1. *Sketch of a vertical slender jet, with an indication of the coordinate system. The locus of centroids of the cross-sections of the jet form a straight line (in the direction of gravity), which we choose to be the z-axis. Then r, θ, and z form the usual cylindrical coordinate system, surface of the jet is denoted by* $r = \mathcal{S}(\theta, z; \varepsilon)$.

coordinate system, with the positive z-axis pointing vertically downward in the direction of gravity. The parameter ε, the slenderness ratio of the jet, is the ratio of a typical radius of the jet to a typical length along the jet and is defined precisely by Geer [2]. The boundary conditions at the free surface are the kinematic condition of no flow through the surface and Bernoulli's equation with constant pressure. For small values of ε, Geer [2] has shown that Φ and $\mathcal{S}$ are given by

$$\Phi = \tfrac{2}{3}(1+z)^{3/2} + \varepsilon^2 \phi(r, \theta, z) + O(\varepsilon^3), \tag{2.1}$$

$$\mathcal{S} = S(\theta, z) + O(\varepsilon), \tag{2.2}$$

where ϕ and S satisfy the conditions

$$\frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r}\frac{\partial \phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \phi}{\partial \theta^2} = -\frac{1}{2}(1+z)^{-1/2}, \qquad z > 0, \quad 0 \leqq r < S(\theta, z), \tag{2.3}$$

with

$$\frac{\partial \phi}{\partial r} - \frac{1}{S^2}\frac{\partial S}{\partial \theta}\frac{\partial \phi}{\partial \theta} = (1+z)^{1/2}\frac{\partial S}{\partial z} \tag{2.4}$$

and

$$\left(\frac{\partial \phi}{\partial r}\right)^2 + S^{-2}\left(\frac{\partial \phi}{\partial \theta}\right)^2 + 2(1+z)^{1/2}\frac{\partial \phi}{\partial z} = 0 \tag{2.5}$$

holding on $r = S(\theta, z)$. Equation (2.3) follows from Laplace's equation for the potential, while (2.4) and (2.5) result from the substitution of the perturbation expansions (2.1)

and (2.2) in the boundary conditions. Thus, we see that ϕ must satisfy the two-dimensional Poisson equation (2.3) in the cross-section of the jet, while (2.4) essentially prescribes the normal derivative of ϕ at the boundary of the cross-section. Equation (2.5) is the additional condition which is needed to determine the free surface. In particular, it is an easy exercise to show that an initially circular jet with a uniform velocity profile has cross-sections which remain circular and decrease in area as the jet accelerates.

To compute ϕ and S, we transform the problem (2.3)–(2.5) into a form that is somewhat easier to deal with numerically. We first note that we can easily find a particular solution to (2.3), and consequently we write ϕ in the form

$$\phi = -\tfrac{1}{8}(1+z)^{-1/2}r^2 + \psi, \tag{2.6}$$

where ψ satisfies the homogeneous version of (2.3), i.e., Laplace's equation. Both ψ and S are presumed known at $z = 0$. We then introduce a new independent radial variable ρ, related to r by

$$\rho = \frac{r}{S(\theta, z)}. \tag{2.7}$$

Thus, r is stretched in a nonuniform manner, but the unknown boundary $r = S(\theta, z)$ is mapped onto the known boundary $\rho = 1$. We also define the new dependent variable $R(\theta, z)$ by

$$R(\theta, z) = \tfrac{1}{2}S(\theta, z)^2(1+z)^{1/2}. \tag{2.8}$$

In terms of the independent variables ρ, θ and z, and the dependent variables $\psi(\rho, \theta, z)$ and $R(\theta, z)$, (2.4) and (2.5) can be written as

$$\frac{\partial R}{\partial z} = (1+\beta^2)\frac{\partial \psi}{\partial \rho} - \beta\frac{\partial \psi}{\partial \theta}, \tag{2.9}$$

$$4R\frac{\partial \psi}{\partial z} = (1+\beta^2)\left(\frac{\partial \psi}{\partial \rho}\right)^2 - \left(\frac{\partial \psi}{\partial \theta}\right)^2 - \frac{3}{4}\frac{R^2}{(1+z)^2}, \tag{2.10}$$

where

$$\beta = \frac{1}{S}\frac{\partial S}{\partial \theta} = \frac{1}{2}\frac{1}{R}\frac{\partial R}{\partial \theta}.$$

These equations hold for $\rho = 1$, $0 \leqq \theta \leqq 2\pi$, and $z > 0$. The differential equation (2.3) then becomes

$$\begin{gathered}(1+\beta^2)\frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho\frac{\partial \psi}{\partial \rho}\right) - \frac{\partial \beta}{\partial \theta}\frac{1}{\rho}\frac{\partial \psi}{\partial \rho} + \frac{1}{\rho^2}\frac{\partial^2 \psi}{\partial \theta^2} - 2\beta\frac{1}{\rho}\frac{\partial^2 \psi}{\partial \rho\, \partial \theta} = 0, \\ 0 \leqq \theta \leqq 2\pi, \qquad 0 \leqq \rho < 1, \qquad z \geqq 0.\end{gathered} \tag{2.11}$$

As a consequence of equations (2.3)–(2.5), we find the integrability condition

$$\int_0^{2\pi} R(\theta, z)\, d\theta = \text{constant} = 2\pi\bar{M}, \tag{2.12}$$

which expresses the constant mass flux in the jet.

Thus, we seek solutions to (2.9)–(2.11) for ψ and R in the region $0 \leqq \rho \leqq 1$, $z > 0$. Once ψ and R have been found, ϕ and S can be recovered using (2.6) and (2.8).

3. Method of solution. In this and the next two sections, we shall describe the method we have devised to solve the problem formulated in § 2. In particular, in this section we will present the underlying motivation for our method as well as the specific finite difference formulas we use. Details of the method we use to solve Laplace's equation will be discussed in the next section, while our treatment of possible discontinuities (e.g., corners) in the jet profile shape will be presented in § 5.

Instead of attempting to solve the differential equation (2.11) subject to the auxiliary conditions (2.9)–(2.10) and (2.12) (as in a classical approach), we proceed in a different manner. To begin, we temporarily think of both ψ and R as functions of z and θ, defined only on the boundary $\rho = 1$. Then, in this context, we may regard equations (2.9)–(2,10) as a system of two nonlinear hyperbolic pseudo-differential equations for ψ and R, with z being the time-like variable and θ the spatial variable. These equations are hyperbolic because the first-order symbol of the linearized system has purely imaginary eigenvalues (see § 8). They are "pseudo" differential equations because the operator $\partial/\partial\rho$ is a nonlocal operator on ψ, when considered as defined only on $\rho = 1$. However, the "auxiliary" condition (2.11) which holds for $\rho < 1$ serves to define $\partial\psi/\partial\rho$ in terms of ψ and R on the boundary. Condition (2.12) is then a conservation law of the system.

In order to obtain a numerical approximation to the solution of our problem formulated in this manner, we use a finite difference scheme defined on the grid points as follows:

$$\begin{aligned} &\theta_i = (i-1)\Delta\theta, && i = 1, \cdots, N, \\ &\rho_j = 1-(j-1)\Delta\rho, && j = 1, \cdots, M, \\ &z_n = n\Delta z, && n = 0, 1, 2, 3, \cdots, \end{aligned} \tag{3.1}$$

where $\Delta\theta = 2\pi/(N-1)$, $\Delta\rho = 1/(M-1)$ and Δz is chosen to satisfy appropriate stability and accuracy criteria (see §§ 7 and 8). Note that $\theta_1 = 0$, $\theta_N = 2\pi$, $z_0 = 0$, $\rho_1 = 1$ and $\rho_M = 0$. We then use the MacCormack scheme [8] to solve (2.9)–(2.10). In particular, if we define the vector $\mathbf{w}(\theta, z)$ by $\mathbf{w} = (R, \psi)^T$, then (2.9), (2.10) can be written as

$$\frac{\partial \mathbf{w}}{\partial z} = \mathbf{F}\left(z, \mathbf{w}, \frac{\partial \mathbf{w}}{\partial \theta}, \frac{\partial \psi}{\partial \rho}\right), \tag{3.2}$$

where the form of the vector $\mathbf{F}$ can be determined from the right-hand sides of (2.9)–(2.10). We employ the forward and backward difference operators D_+ and D_-, respectively defined by

$$\begin{aligned} D_+\mathbf{w}_i^n &= \frac{\mathbf{w}_{i+1}^n - \mathbf{w}_i^n}{\Delta\theta}, \\ D_-\mathbf{w}_i^n &= \frac{\mathbf{w}_i^n - \mathbf{w}_{i-1}^n}{\Delta\theta}, \\ \mathbf{w}_i^n &= \mathbf{w}(\theta_i, z_n). \end{aligned} \tag{3.3}$$

Then the forward-backward MacCormack scheme we use is given by the following two-step formula:

$$\text{(predictor):} \quad \tilde{\mathbf{w}}_i^{n+1} = \mathbf{w}_i^n + \Delta z\,\mathbf{F}(z_n, \mathbf{w}_i^n, D_+\mathbf{w}_i^n, D_\rho\psi_i^n); \tag{3.4}$$

$$\text{(corrector):} \quad \mathbf{w}_i^{n+1} = \tfrac{1}{2}\{\mathbf{w}_i^n + \tilde{\mathbf{w}}_i^{n+1} + \Delta z\,\mathbf{F}(z_{n+1}, \tilde{\mathbf{w}}_i^{n+1}, D_-\tilde{\mathbf{w}}_i^{n+1}, D_\rho\tilde{\psi}_i^{n+1}\}. \tag{3.5}$$

Here $D_\rho\psi_i^n$ is an approximation to $\partial\psi/\partial\rho$ on $\rho = 1$ at $\theta = \theta_i$ and $z = z_n$, which we shall describe below. In order to maintain symmetry, the forward-backward MacCormack scheme is alternated with the backward-forward scheme, which uses backward differences in the predictor step and forward differences in the corrector step. Also, it was found that the conservation law (2.12) was satisfied more closely when the quantity β in (2.9) and (2.10) was approximated as

$$D_\pm R_i^n/(R_i^n + R_{i\pm1}^n)$$

and this form was used in all the calculations given here.

The term $D_\rho\psi_i^n$ in (3.4) and (3.5) is computed by first solving for an approximation to the solution ψ of (2.11), with ψ_i^n specified on the boundary. The approximation is given by

$$\begin{aligned} &A_i^n\rho_j(\rho_{j-1/2}(\psi_{i,j-1}-\psi_{i,j})-\rho_{j+1/2}(\psi_{i,j}-\psi_{i,j+1}))(\Delta\rho)^{-2} \\ &\quad -C_i^n\rho_j(\psi_{i,j-1}-\psi_{i,j+1})(2\Delta\rho)^{-1}+(\psi_{i+1,j}-2\psi_{i,j}+\psi_{i-1,\,j})(\Delta\theta)^{-2} \\ &\quad -\rho_j\{B_{i+}^n[\psi_{i+1,j-1}-\psi_{i,j-1}-\psi_{i+1,j+1}+\psi_{i,j+1}] \\ &\qquad +B_{i-}^n(\psi_{i,j-1}-\psi_{i-1,j-1}-\psi_{i,j+1}+\psi_{i-1,j+1})\}(2\Delta\rho\,\Delta\theta)^{-1}=0. \end{aligned} \tag{3.6}$$

Here, $\psi_{i,j} = \psi_{i,j}^n = \psi(\rho_j, \theta_i, z_n)$, and

$$\begin{aligned} B_{i\pm}^n &= \frac{D_\pm R_i^n}{R_i^n + R_{i\pm1}^n}, \\ A_i^n &= 1+\tfrac{1}{2}((B_{i+}^n)^2+(B_{i-}^n)^2), \\ C_i^n &= \frac{B_{i+}^n - B_{i-}^n}{\Delta\theta}. \end{aligned} \tag{3.7}$$

In (3.6) and (3.7) we have used second-order accurate difference approximations to the derivatives of ψ and R. Equations (3.6) are solved by successive overrelaxation (§ 4). Once $\psi_{i,j}^n$ is determined, the term $D_\rho\psi_i^n$ is computed as

$$D_\rho\psi_i^n = \frac{3\psi_{i,1}^n - 4\psi_{i,2}^n + \psi_{i,3}^n}{2\Delta\rho}, \tag{3.8}$$

which is a second-order one-sided approximation to $\partial\psi/\partial\rho$.

Equations (3.1)–(3.8) describe our numerical scheme to solve the problem of § 2. For each z step, equations (3.6) are solved twice, once corresponding to the predictor step (3.4) and then again for the corrector step (3.5). The fact that our scheme is formally second-order accurate will be shown below. In § 7, the second-order accuracy of our method is confirmed by the results of several numerical experiments.

We conclude this section by showing that the scheme given by (3.4–(3.5) is formally second-order accurate. To do this, we note that if $\mathbf{w}$ is a smooth function of z, then by Taylor's theorem

$$\mathbf{w}(z_{n+1}) = \mathbf{w}(z_n) + \Delta z\,\frac{\partial\mathbf{w}}{\partial z}(z_n) + \frac{(\Delta z)^2}{2}\,\frac{\partial^2\mathbf{w}}{\partial z^2} + O((\Delta z)^3). \tag{3.9}$$

Let $\Delta\theta$ and $\Delta\rho$ be proportional to Δz, and for convenience set

$$\begin{aligned}
\mathbf{w}^n &= \mathbf{w}(1, \theta, z_n),\\
\mathbf{p}_{\pm}^n &= D_{\pm}\mathbf{w}^n,\\
q^n &= D_\rho \psi(1, \theta, z_n),\\
\mathbf{F}_{\pm}^n &= \mathbf{F}(z_n, \mathbf{w}^n, \mathbf{p}_{\pm}^n, q^n).
\end{aligned}$$

Then, from (3.2),

$$\frac{\partial \mathbf{w}}{\partial z}(z_n) = \mathbf{F}\left(z_n, \mathbf{w}^n, \frac{\partial \mathbf{w}^n}{\partial \theta}, D_\rho \psi^n\right) = \tfrac{1}{2}\{\mathbf{F}_+^n + \mathbf{F}_-^n\} + O((\Delta z)^2) \tag{3.10}$$

and

$$\begin{aligned}
\Delta z \frac{\partial^2 \mathbf{w}}{\partial z^2}(z_n) &= \Delta z \frac{\partial}{\partial z}\{\mathbf{F}(z_n, \mathbf{w}^n, \mathbf{p}_-^n, q^n) + O(\Delta z)\}\\
&= \Delta z \frac{\partial \mathbf{F}_-}{\partial z} + \frac{\partial \mathbf{F}_-}{\partial \mathbf{w}} \Delta \mathbf{w}^n + \frac{\partial \mathbf{F}_-}{\partial \mathbf{p}_-} \Delta \mathbf{p}^n + \frac{\partial \mathbf{F}}{\partial q} \Delta q^n + O((\Delta)^2),
\end{aligned} \tag{3.11}$$

where $\Delta \mathbf{w}^n = \mathbf{w}^{n+1} - \mathbf{w}^n$, etc.

If $\tilde{\mathbf{w}}^{n+1}$ is defined by the right side of (3.4), i.e., the predicted value of $\mathbf{w}^{n+1}$, and

$$\tilde{\mathbf{p}}_-^{n+1} = D_- \tilde{\mathbf{w}}^{n+1} \quad \text{and } \tilde{q}^{n+1} = D_\rho \tilde{\psi}(1, \theta, z_{n+1}),$$

then

$$\begin{aligned}
\Delta \mathbf{w}^n &= \tilde{\mathbf{w}}^{n+1} - \mathbf{w}^n + O((\Delta z)^2),\\
\Delta \mathbf{p}_-^n &= \tilde{\mathbf{p}}_-^{n+1} - \mathbf{p}^n + O((\Delta z)^2),\\
\Delta q^n &= \tilde{q}^{n+1} - q^n + O((\Delta z)^2).
\end{aligned} \tag{3.12}$$

Substituting (3.10)–(3.12) into (3.9), we obtain

$$\begin{aligned}
\mathbf{w}(z_{n+1}) &= \mathbf{w}(z_n) + \frac{\Delta z}{2}\mathbf{F}_+^n + \frac{\Delta z}{2}\left\{\mathbf{F}_-^n + \Delta z \frac{\partial \mathbf{F}^n}{\partial z} + \frac{\partial \mathbf{F}}{\partial \mathbf{w}}(\tilde{\mathbf{w}}^{n+1} - \mathbf{w}^n)\right.\\
&\qquad \left. + \frac{\partial \mathbf{F}}{\partial \mathbf{p}}(\tilde{\mathbf{p}}^{n+1} - \mathbf{p}^n) + \frac{\partial \mathbf{F}}{\partial q}(\tilde{q}^{n+1} - q^n)\right\} + O((\Delta z)^3)\\
&= \tfrac{1}{2}\{\mathbf{w}(z_n) + [\mathbf{w}(z_n) + \Delta z \mathbf{F}_+^n] + \Delta z\, \mathbf{F}(z_{n+1}, \tilde{\mathbf{w}}^{n+1}, \tilde{\mathbf{p}}_-^{n+1}, \tilde{q}^{n+1})\} + O((\Delta z)^3),
\end{aligned}$$

which is equivalent to (3.4)–(3.5) and shows that the scheme is formally second-order accurate.

4. Solution of Laplace's equation. To use the difference scheme (3.1)–(3.8) to advance the solution from $z = z_n$ to $z = z_{n+1}$ requires solving the difference approximation (3.6) to Laplace's equation for both the predictor and corrector steps. The values of ψ^n and $\tilde{\psi}^{n+1}$ in the interior (i.e., $\rho < 1$) are used with formula (3.8) to compute $D_\rho \psi^n$ and $D_\rho \tilde{\psi}^{n+1}$, respectively, which are the approximations to the normal derivative of ψ at $\rho = 1$.

The difference approximation (3.6) to Laplace's equation is solved by point successive overrelaxation (SOR) using the natural ordering of grid points. The SOR

algorithm is given by

$$\begin{aligned}\psi_{i,j}^{\cdot,k+1} = \psi_{i,j}^{\cdot,k} + \tilde{\omega}_i\{&A_i\rho_j(\rho_{j-1/2}(\psi_{i,j-1}^{\cdot,k+1} - \psi_{i,j}^{\cdot,k}) - \rho_{j+1/2}(\psi_{i,j}^{\cdot,k} - \psi_{i,j+1}^{\cdot,k}))\,\Delta\rho \\ &- C_i\rho_j(\psi_{i,j-1}^{\cdot,k+1} - \psi_{i,j+1}^{\cdot,k})(2\Delta\rho)^{-1} + (\psi_{i+1,j}^{\cdot,k} - 2\psi_{i,j}^{\cdot,k} + \psi_{i-1,j}^{\cdot,k+1})(\Delta\theta)^{-2} \\ &- \rho_j\{B_{i+}[\psi_{i+1,j-1}^{\cdot,k+1} - \psi_{i,j-1}^{\cdot,k+1} - \psi_{i+1,j+1}^{\cdot,k} + \psi_{i,j+1}^{\cdot,k}] \\ &+ B_{i-}[\psi_{i,j-1}^{\cdot,k+1} - \psi_{i-1,j-1}^{\cdot,k+1} - \psi_{i,j+1}^{\cdot,k} + \psi_{i-1,j+1}^{\cdot,k}]\}(2\Delta\rho\,\Delta\theta)^{-1}\},\end{aligned} \tag{4.1}$$

$$j = 1, \cdots, N-1, \qquad j = 2, \cdots, M-1,$$

where, to simplify notation, we write $\psi_{i,j}^{\cdot,k}$ for the kth iterate for either $\psi_{i,j}^{n}$ or $\tilde{\psi}_{i,j}^{n+1}$. The iteration parameter $\tilde{\omega}_i$ is given by

$$\tilde{\omega}_i = \frac{\frac{1}{2}\omega}{A_i(\rho/\Delta\rho)^2 + (1/\Delta\theta)^2}, \tag{4.2}$$

where

$$\omega = \frac{2}{1 + 2.4\bar{M}^{-1/2}(1+z)^{1/4}\Delta\rho} \tag{4.3}$$

and $\bar{M}$ is defined by (2.12). Formula (4.2) is a normalization, dividing the standard SOR parameter ω by the absolute value of the coefficient of $\psi_{i,j}$ in the approximation (3.6). Formula (4.3) giving the SOR parameter ω will be discussed later in this section. The coefficients A_i, $B_{i\pm}$ and C_i are functions of R_i^n or $\tilde{R}_i^{n+1}$ and are given by formulas (3.7).

The value of $\psi^{\cdot,k+1}$ at the origin, i.e., $\psi_{i,M}^{\cdot,k+1}$, was determined from the values at the neighboring grid points, $\psi_{i,M-1}^{\cdot,k+1}$, by means of the formula

$$\psi_{i,M}^{\cdot,k+1} = \sum_{i=1}^{N-1}(A_i + C_i)\psi_{i,M-1}^{\cdot,k+1} \Big/ \sum_{i=1}^{N-1}(A_i + C_i). \tag{4.4}$$

Formula (4.4) is derived by integrating Laplace's equation (2.11) over a disc of radius ε centered at the origin. This gives

$$\varepsilon \int_0^{2\pi} (1+\beta^2)\frac{\partial\psi}{\partial\rho}(\varepsilon, \theta)\,d\theta + \int_0^{2\pi} \frac{\partial\beta}{\partial\theta}\psi(\varepsilon, \theta)\,d\theta = 0. \tag{4.5}$$

If ε is taken to be $\frac{1}{2}\Delta\rho$ and the integrals are approximated by sums while the integrands are approximated as

$$\frac{\partial\psi}{\partial\rho}(\tfrac{1}{2}\Delta\rho_j, \theta_i) \simeq \frac{\psi_{i,M-1} - \psi_{i,M}}{\Delta\rho}$$

and

$$\psi(\tfrac{1}{2}\Delta\rho_j, \theta_i) \simeq \frac{\psi_{i,M-1} + \psi_{i,M}}{2},$$

then (4.5) yields

$$\sum_{i=1}^{N-1} A_i(\psi_{i,M-1} - \psi_{i,M}) + \sum_{i=1}^{N-1} C_i(\psi_{i,M-1} + \psi_{i,M}) = 0. \tag{4.6}$$

Here, as in (3.7),

$$A_i \simeq 1 + \beta_i^2, \qquad C_i \simeq \frac{\partial}{\partial\theta}\beta(\theta_i).$$

Since $\psi_{i,M}$ is independent of θ_i and

$$\sum_{i=1}^{N-1} C_i = \sum_{i=1}^{N-1} \left(\frac{R_{i+1}-R_i}{R_{i+1}+R_i} - \frac{R_i - R_{i-1}}{R_i + R_{i-1}} \right) \Delta\theta^{-2} = 0,$$

(4.4) follows easily from (4.6). Note that (4.4) is formally second-order accurate, as is the approximation (3.6).

After (4.1) and (4.4) were applied for $i < N$, the periodicity conditions

$$\psi_{N,j}^{\cdot,k+1} = \psi_{1,j}^{\cdot,k+1}, \qquad 1 \leqq j \leqq M \tag{4.7}$$

were imposed. The iterative procedure given by (4.1)–(4.4) was terminated when certain convergence criteria were satisfied. These criteria will be discussed later in this section.

As noted at the beginning of this section, the approximation to Laplace's equation must be solved twice to advance the solution by one z-step. Solving these difference equations is the most time-consuming portion of the algorithm. By using linear extrapolation to obtain the initial iterate for the predictor step, the solution time was reduced dramatically. In particular, for the predictor step, the values of $\tilde{\psi}_{i,j}^{n+1,0}$, the starting values for the iteration, were obtained as

$$\tilde{\psi}_{i,j}^{n+1,0} = 2\psi_{i,j}^{n} - \psi_{i,j}^{n-1} \qquad i = 1, \cdots, N, \quad j = 2, \cdots, M \tag{4.8}$$

for the interior values. The values of $\tilde{\psi}_{i,1}^{n+1}$ on the boundary were given by (3.4). For the corrector step the initial iterates were taken to be the values of the predicted potential in the interior, i.e.,

$$\psi_{i,j}^{n+1,0} = \tilde{\psi}_{i,j}^{n+1}, \qquad i = 1, \cdots, N, \quad j = 2, \cdots, M. \tag{4.9}$$

The values of $\psi_{i,1}^{n+1}$ on the boundary are, of course, given by formula (3.5).

For both the predictor and corrector steps the SOR procedure was terminated when the relative change between iterates measured in the l^2-norm was less than a small parameter ε, i.e., when

$$\|\tilde{\psi}^{n+1,k} - \tilde{\psi}^{n+1,k-1}\| \leqq \varepsilon \|\tilde{\psi}^{n+1,k}\|. \tag{4.10}$$

Then we set

$$\tilde{\psi}_{i,j}^{n+1} = \tilde{\psi}_{i,j}^{n+1,k},$$

and similarly for ψ^{n+1}. For the computations described in this paper ε was taken to be 10^{-5}. The number of iterations to compute either $\tilde{\psi}^{n+1}$ or ψ^{n+1} was restricted to be less than 250. When this limit was achieved, the last iterate was taken to be the solution. This limit was encountered only for $z = \Delta z$ (and sometimes for $2\Delta z$) when $\|\tilde{\psi}\|_2$ and $\|\psi\|_2$ were very small, or for larger values of z when the solution was no longer well-behaved due to lack of resolution (§ 6). Since for our examples the potential ψ is zero at $z = 0$, the condition (4.10) is not very appropriate for small values of n.

Typical values for the number of SOR iterations required to solve for the potential are given in Table 1. Column (a) gives the number of iterations when the linear extrapolation (4.8) was used for the predictor step and column (b) gives the number of iterations when the initial values for the predictor step were the values of potential at the previous values of z, i.e.,

$$\tilde{\psi}_{i,j}^{n+1,0} = \psi_{i,j}^{n}. \tag{4.11}$$

Note that the total number of iterations per step using (4.11) is more than five times that

TABLE 1
SOR *iterations for several values of z for a rectangle* 81×31, $\Delta z = .1$.

	a. Linear extrapolation (4.8)		b. Previous value (4.11)	
z	Iterations Predictor/Corrector		Iterations Predictor/Corrector	
1.0	75	75	>250	48
2.0	22	23	>250	26
3.0	13	13	>250	16
4.0	12	11	247	12
5.0	14	12	217	15
6.0	14	13	181	15
7.0	14	13	147	17

required when the linear extrapolation (4.8) is used. This reduction in time more than justifies the extra storage required to keep the values of ψ^{n-1}.

The formula (4.3) for the iteration parameter ω was obtained in the following way. The difference equations (3.6) are a second-order approximation to Laplace's equation on the region $r \leqq S(\theta, z)$ with a nonuniform grid given by (3.1). For the usual second-order accurate five-point difference approximation for Laplace's equation on a regular mesh, Garabedian [1] showed that the optimal iteration parameter for SQR is given approximately by

$$\omega = \frac{2}{1+k_1 h/\sqrt{2}}, \tag{4.12}$$

where h is the mesh width and k_1 is the first eigenvalue of the Laplacian on the domain being considered. He also pointed out that the value of k_1 can be estimated from below by the Faber–Krahn inequality

$$k_1 \geqq 2.4\left(\frac{\pi}{A}\right)^{1/2} = \tilde{k}_1,$$

where A is the area of the domain.

In the present case, the cross-sectional area A varies as a function of z and is given by

$$A = 2\pi\bar{M}(1+z)^{-1/2},$$

where $\bar{M}$ is defined by (2.12). Thus we have

$$\tilde{k}_1 = (1.2)\sqrt{2}\,\bar{M}^{-1/2}(1+z)^{1/4}.$$

Using this as an estimate for k_1 in (4.11), we find

$$\omega = \frac{2}{1+1.2\bar{M}^{-1/2}(1+z)^{1/4}h}.$$

In the present context it is not clear what value should be given to h. On intuitive grounds it was taken to be proportional to $\Delta\rho$. Moreover, the Faber–Krahn inequality is sharp for circular domains and is less accurate for elongated and nonconvex domains. Thus, the quantity $k_1 h$ in formula (4.12) was estimated by multiples of $\tilde{k}_1\Delta\rho$ and, after some experimentation, it was found that $2\tilde{k}_1\Delta\rho$ or $\tilde{k}_1\Delta\rho$ worked very well in most of the computations considered in this paper.

5. The treatment of corners. The scheme (3.1)–(3.8) was used to compute the shape of jets whose cross-sectional shapes contained corners or cusps. Examples of such jets are those which emanate from rectangular or triangular orifices. In this section we will examine the finite difference approximation to the solution in the vicinity of such corners.

For this purpose, consider a corner such as that illustrated in Fig. 2 and assume that R and ψ are symmetric about the corner, that is,

$$R(\theta_0+\theta)=R(\theta_0-\theta), \qquad \psi(\theta_0+\theta)=\psi(\theta_0-\theta),$$

FIG. 2. *Examples of the placement of gridpoints near a corner.*

where θ_0 is the angular coordinate of the corner. At such a corner $\partial R/\partial\theta$ and $\partial\psi/\partial\theta$ will change sign, i.e.,

$$\frac{\partial R}{\partial\theta}(\theta_0+\theta)=-\frac{\partial R}{\partial\theta}(\theta_0-\theta),$$

$$\frac{\partial\psi}{\partial\theta}(\theta_0+\theta)=-\frac{\partial\psi}{\partial\theta}(\theta_0-\theta),$$

and $\partial R/\partial\theta$ will be discontinuous. (Recall that R is related to the shape function S by (2.8).) However, notice that these discontinuous quantities appear in (2.9)–(2.10) only as products or squares (recall that $\beta=(1/2R)(\partial R/\partial\theta)$ so that the right-hand sides of (2.9)–(2.10) are continuous at a symmetric corner. In order to obtain accurate solutions for jets having such corners, it is essential that the finite difference scheme properly portray this behavior of the differential equations.

Consider, for example, the term $(\partial R/\partial\theta)(\partial\psi/\partial\theta)$ which appears in (2.9). Assume that the grid is as shown in Fig. 2a, with the corner grid point having index i. The discontinuous change in sign of $\partial R/\partial\theta$ at the corner is reflected in the change of sign between D_+R_i and D_-R_i. Similarly, $D_+\psi_i$ and $D_-\psi_i$ are of opposite sign. Thus

$$\frac{\partial R}{\partial\theta}\frac{\partial\psi}{\partial\theta}\simeq\tfrac{1}{2}(D_+R_iD_+\psi_i+D_-R_iD_-\psi_i), \tag{5.1}$$

which is an accurate approximation to the continuous function $(\partial R/\partial\theta)(\partial\psi/\partial\theta)$ on the boundary. Similarly, the squares of β and $\partial\psi/\partial\theta$ are approximated well by the average of the squares of the one-sided differences. In fact, if R and ψ each have one-sided second derivatives at the corner that are continuous and one-sided third derivatives (which may be discontinuous but bounded), then the above approximations are formally second-order accurate. We note, however, that the central difference approximations to $\partial R/\partial\theta$ and $\partial\psi/\partial\theta$ about the corner point vanish and thus give inaccurate approximations. Also, if the grid points are placed symmetrically about the corner

without having a grid point at the corner, as in Fig. 2b, then for those grid points nearest the corner the approximation (5.1) will not be accurate. Central differences will not be accurate in this case either.

Consider now the treatment of these terms in the MacCormack scheme (3.4)–(3.5) when the grid is as in Fig. 2a. As noted above the approximations such as (5.1) are formally second-order accurate at such corners if R and ψ satisfy appropriate conditions on their one-sided higher derivatives. This implies that (3.10) and (3.11) are valid and, hence, that the MacCormack scheme is formally second-order accurate even at such corners (see also § 7).

Laplace's equation in the form of (2.11) also contains the terms β^2 and $(\partial/\partial\rho)(\beta(\partial\psi/\partial\theta))$. The particular form of differencing for these terms in the difference approximation (3.6)–(3.7) was chosen in light of the above considerations. Therefore, one would expect that the approximation (3.6)–(3.7) is more accurate than if centered differences were used for the derivatives with respect to θ.

6. Examples. Several examples of thin streams falling vertically through an orifice of a specified shape were calculated using the scheme outlined in the previous sections. We present here three of these examples. These and other examples are discussed in more detail elsewhere (see Geer and Strikwerda [4]). For each example the initial conditions were $\psi \equiv 0$ and $R(\theta, z)$, i.e., $S(\theta, z)$, specified at $z = 0$. Note that the condition $\psi = 0$ at $z = 0$ corresponds to a jet that is emanating with a uniform velocity profile. Thus, in the notation of § 3, we set $z = 0$, $\psi^0_{i,j} = 0$, and $R^0_i = R^0(\theta_i)$, where $R^0(\theta)$ was specified by one of the following:

1. An ellipse, $R^0 = \frac{1}{2}(.25\cos^2\theta + \sin^2\theta)^{-1}$, where the semi-axes of the ellipse are 2 and 1 (Fig. 3).

2. An equilateral triangle, $R^0 = \frac{1}{2}\min_{l=0,1,2}\sec^2(\theta - 2\pi l/3)$ where the length of the side of the triangle is $2\sqrt{3}$. (Fig. 4).

3. A rectangle, $R^0 = \frac{1}{2}\min(\sec^2\theta, 4\csc^2\theta)$, where 2 and 4 are the lengths of the sides of the rectangle (Fig. 5).

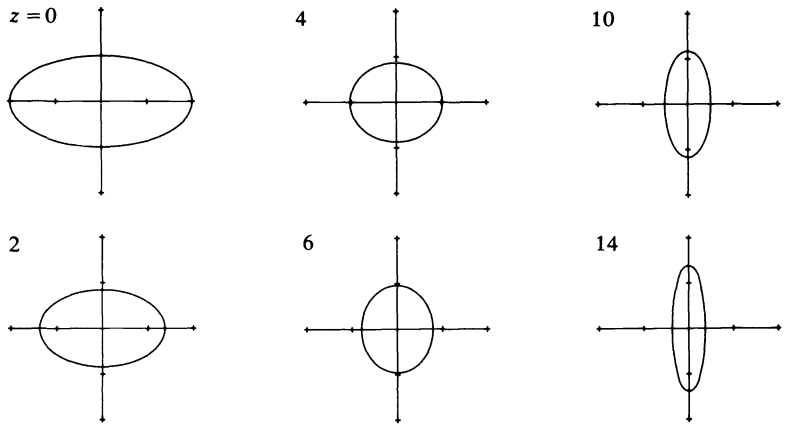


FIG. 3. *Cross-sectional shapes at several values of z for a jet with an initial shape of an ellipse.*

For each example the origin was located at the center of mass of the shape as required in the derivation of the basic equation (2.3)–(2.4) (see Geer [2]). The corresponding figures show cross-sections of the jet as several values of z.

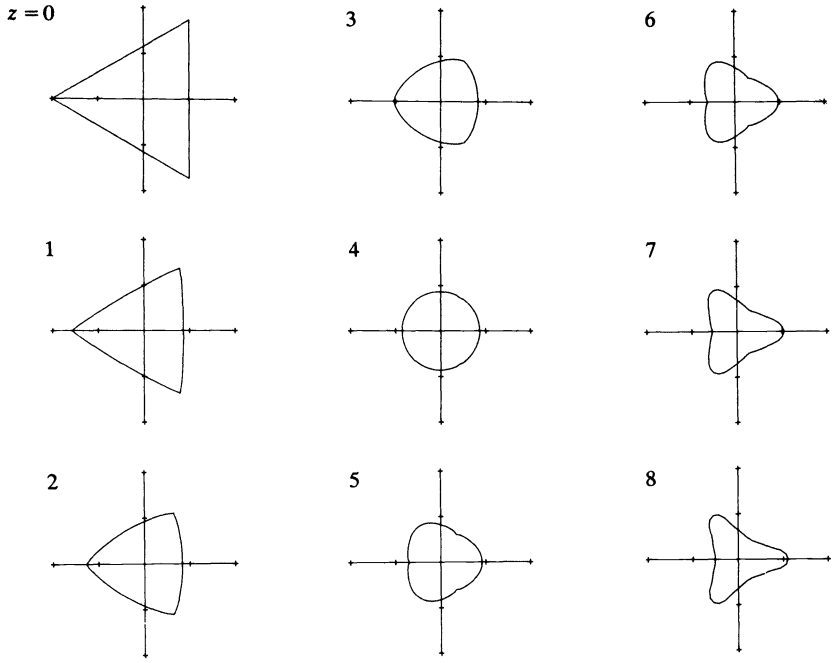


FIG. 4. *Cross-sectional shapes at several values of z for a jet with the initial shape of an equilateral triangle, with side of length* $2\sqrt{3}$.

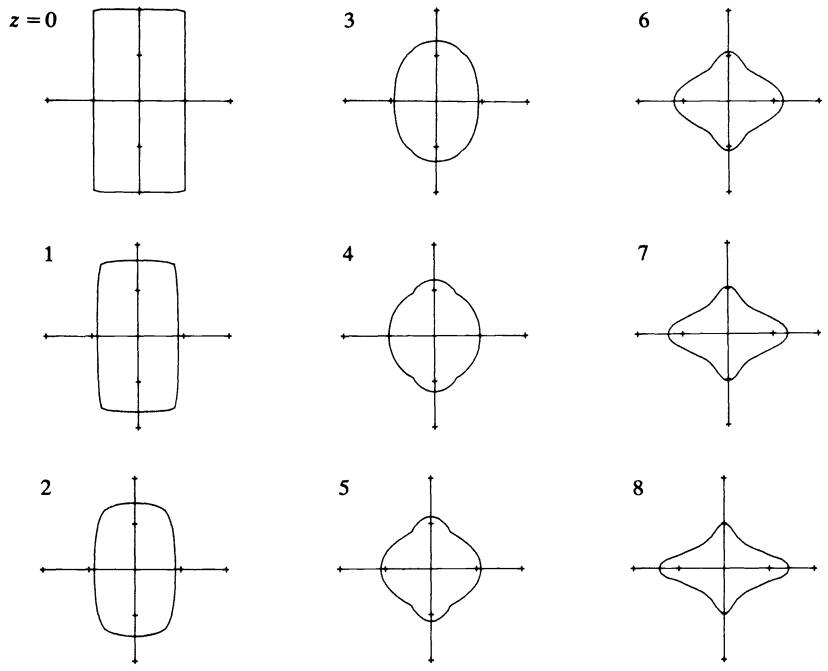


FIG. 5. *Cross-sectional shapes for a jet with the initial shape of a rectangle with sides of length* 2 *and* 4.

The primary purpose of the first example, the ellipse, was to check the accuracy of the numerical scheme. The numerical solution was compared with the analytic solution presented by Geer [2]. The calculation of this analytic solution involved only the straightforward numerical integration of nonlinear ordinary differential equations, and consequently we assumed that this solution is known exactly.

Fig. 3 shows the cross-sectional shape of the jet at various values of z. At $z = 0$, the ellipse had an aspect ratio of 2. As z increased, the shape of the jet became less eccentric, was nearly circular at about $z = 4.9$ and then assumed an elliptical shape with the direction of the major and minor axes exactly interchanged with those of the original axes. The cross-sectional shape became more and more elongated as z increased. At $z = 14.0$, the numerical solution with $N = 101$, $M = 31$ and $\Delta z = .1$ agreed with the analytic solution to within 1% relative error in the l^2- and l^1-norms, and to within 2% relative error in the maximum norm.

In all of our examples the computations terminated when the outward moving portions became sufficiently elongated so that they could no longer be resolved adequately by the uniform grid used for the angular coordinate. The numerical break-up of the solution occurred soon after the last cross-section shown in each case. The conservation law (2.12) was satisfied to within .5% relative error in all the cases shown here.

The other examples had for initial shapes an equilateral triangle (Fig. 4) and a rectangle (Fig. 5). The initial length of a side of the triangle was $2\sqrt{3}$ units, while the sides of the rectangle were 4 units and 2 units. In these examples, for small values of z the cross-sections decreased in area, but maintained essentially the same shape. In particular the discontinuities in the slopes at the corners were propagated for some distance in z. For larger values of z the shape became nonconvex as those portions of the surface that had been corners "buckled-in". Those portions of the surface that had originally been the sides formed the new extremities of the cross-sectional shape. For the case of the equilateral triangle the extremities all extended outward as z increased. For the case of the rectangle the major extremities extended outward and the minor extremities moved slowly inward. The numerical break-up of these cases occurred along these outward moving extremities as noted above.

We point out that the results for the equilateral triangle are consistent with those of Bidone discussed by Rayleigh [9]; i.e., "... a vein issuing from an orifice in the form of a regular polygon, of any number of sides, resolved itself into an equal number of thin sheets, whose planes are perpendicular to the sides of the polygon." (See also Rayleigh [10].)

However, Rayleigh [9] implies by the sketch of the cross-sections of the equilateral triangle that the triangle assumed a hexagonal cross-section. Our calculations did not produce such a shape and we presume that this discrepancy is due to mistaken observations near the point where the cross-section was circular.

The example with the rectangular initial shape was run with $N = 81$, $M = 31$ and $\Delta z = 0.1$. The triangular shape was run with $N = 61$, $M = 31$ and $\Delta z = 0.1$. In these examples, the values of N was chosen so that a grid point would be at or very near the corner of the original shape.

7. Numerical accuracy. As shown in §§ 3 and 5, the finite difference formulas used to approximate the system of (2.9)–(2.11) are all formally second-order accurate. Moreover, in the numerical calculations the parameter Δz and the convergence criteria for the SOR iterations were all chosen with the purpose of maintaining the second-order accuracy. Nonetheless, it must be demonstrated that the overall scheme is in fact second-order accurate.

A series of computations was made to determine the accuracy of the scheme. As noted in § 6, an alternative procedure can be employed to determine the shape of the jet when the jet emanates from an elliptical orifice. This alternative procedure requires solving a nonlinear ordinary differential equation of second order for the aspect ratio. This equation was integrated using a fourth-order Runge–Kutta method with a small step size. Because of the high accuracy employed, the numerical solution of this equation was assumed to be exact for the purposes of this comparison.

The difference scheme given by (3.1)–(3.8) was used to compute the shape of the elliptical jet (with the initial data given in Example 1 of § 6) up to $z = 10.0$ for various values of N and M. The z-step, Δz, was chosen as $10/(N-1)$ so that integrating the finite difference scheme to $z = 10$ required $N-1$ complete steps. The SOR convergence parameter ε was 10^{-5}, while the values of N were 21, 41, 61, 81 and 101 and the values of M were $.3(N-1)+1$.

The results are displayed in Table 2 for the errors measured in the l^2- and maximum norms. The last two columns give the change in the logarithm of the error

TABLE 2
Analysis of the error and accuracy of the method for elliptical jets.

$N-1$	l^2 error	max error	l^2 order	l_∞ order
20	5.36×10^{-2}	2.00×10^{-1}	–	–
40	1.16×10^{-2}	2.93×10^{-2}	−2.21	−2.77
60	5.15×10^{-3}	1.12×10^{-2}	−2.00	−2.37
80	2.89×10^{-3}	5.93×10^{-3}	−2.01	−2.21
100	1.85×10^{-3}	3.74×10^{-3}	−2.00	−2.07

divided by the change in the logarithm of $(N-1)$ for successive values of N; i.e.,

$$\frac{\log(.0116)-\log(.0536)}{\log 40-\log 20} = -2.21, \quad \text{etc.}$$

The closeness of these entries to −2 indicates that the overall method is second-order accurate. The l^2-norm errors listed are relative errors, i.e., the l^2-norm error divided by the l^2-norm of the solution.

Comparisons were also made to study the effect of the placement of grid points on the computation of jets with corners. As discussed in § 5, the placement of grid points as shown in Fig. 2a should be more accurate than that shown in Fig. 2b. In those cases without a grid point at the vertex (Fig. 2b) the discontinuities in the tangents were smeared out almost immediately; otherwise, the solutions are similar to those seen in Figs. 4 and 5. We emphasize that for such shapes we have no means of ascertaining which solution is more accurate. However, in view of the analysis given in § 5, it would appear that the results are most accurate when a grid point is at or very close to the vertex of the corner.

We note also that the symmetry of the results shown in Figs. 3–5 is not imposed on the computations but results only from the initial symmetry at $z = 0$. By alternating the forward-backward MacCormack scheme with the backward-forward scheme, asymmetric discretization errors are presumably minimized. However, in runs with the elliptical jet using only the forward-backward scheme, the accuracy and symmetry were not severely affected.

8. Well-posedness of the problem. We will now discuss the well-posedness of the system of equations (2.9)–(2.11). In particular, we shall derive a necessary condition to insure that the growth rate of small perturbations to the solution of (2.9)–(2.11) is bounded. We will then indicate how, by a similar analysis, we could demonstrate the stability of our numerical scheme.

An analysis of the well-posedness of the system (2.9)–(2.11) is necessary because the original problem, before the perturbation in the slenderness ratio, is an elliptic problem. Solving such a problem by marching in any particular direction is not a well-posed method. It must then be shown that to solve (2.9)–(2.11) by marching in the z-direction is a well-posed problem.

To begin our analysis, let $(\bar{R}, \bar{\psi})$ be a smooth solution of the system (2.9)–(2.11) and consider another solution (R, ψ) of this system of the form

$$R = \bar{R} + 2\eta\bar{R}\tilde{R}, \qquad \psi = \bar{\psi} + \eta\tilde{\psi}. \tag{8.1}$$

Here η is a perturbation parameter, and $(2\eta\bar{R}\tilde{R}, \eta\tilde{\psi})$ represent small perturbations in $\bar{R}$ and $\bar{\psi}$, respectively. Thus, to investigate the well-posedness of the system (2.9)–(2.11), we shall first determine (to first order) the system of equations satisfied by $(\tilde{R}, \tilde{\psi})$. We shall then show that this linear system is well-posed if a certain scalar quantity $\alpha = \alpha(\theta, z)$ is nonpositive.

In order to determine the equations satisfied by $(\tilde{R}, \tilde{\psi})$, we substitute (8.1) into (2.9)–(2.11), expand the resulting expressions in a Taylor series about $\eta = 0$ and then set the coefficient of η in each equation equal to zero. In this way we obtain the following system of equations:

$$2\bar{R}\frac{\partial\tilde{R}}{\partial z} = (1+\bar{\beta}^2)\frac{\partial\tilde{\psi}}{\partial\rho} - \bar{\beta}\frac{\partial\tilde{\psi}}{\partial\theta} + \left(2\bar{\beta}\frac{\partial\bar{\psi}}{\partial\rho} - \frac{\partial\bar{\psi}}{\partial\theta}\right)\frac{\partial\tilde{R}}{\partial\theta} - 2\frac{\partial\bar{R}}{\partial z}\tilde{R}, \tag{8.2}$$

$$2\bar{R}\frac{\partial\tilde{\psi}}{\partial z} = (1+\bar{\beta}^2)\frac{\partial\bar{\psi}}{\partial\rho}\frac{\partial\tilde{\psi}}{\partial\rho} - \frac{\partial\bar{\psi}}{\partial\theta}\frac{\partial\tilde{\psi}}{\partial\theta} + \bar{\beta}\left(\frac{\partial\bar{\psi}}{\partial\rho}\right)^2\frac{\partial\tilde{R}}{\partial\theta} - \left(4\bar{R}\frac{\partial\bar{\psi}}{\partial z} + \frac{3}{2}\frac{\bar{R}^2}{(1+z)^2}\right)\tilde{R}, \tag{8.3}$$

$$\begin{aligned}(1+\bar{\beta}^2)\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\tilde{\psi}}{\partial\rho}\right) - \frac{\partial\bar{\beta}}{\partial\theta}\frac{1}{\rho}\frac{\partial\tilde{\psi}}{\partial\rho} + \frac{1}{\rho^2}\frac{\partial^2\tilde{\psi}}{\partial\theta^2} - 2\bar{\beta}\frac{1}{\rho}\frac{\partial^2\tilde{\psi}}{\partial\rho\,\partial\theta} \\ = -2\left(\bar{\beta}\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\bar{\psi}}{\partial\rho}\right) - \frac{1}{\rho}\frac{\partial^2\bar{\psi}}{\partial\rho\,\partial\theta}\right)\frac{\partial\tilde{R}}{\partial\theta} + \frac{1}{\rho}\frac{\partial\bar{\psi}}{\partial\rho}\frac{\partial^2\tilde{R}}{\partial\theta^2},\end{aligned} \tag{8.4}$$

where $\bar{\beta} = (1/2\bar{R})(\partial\bar{R}/\partial\theta)$. Here (8.2)–(8.3) hold on $\rho = 1$, while (8.4) holds for $0 \leqq \rho < 1$ and $0 \leqq \theta \leqq 2\pi$.

We now wish to examine the behavior of $(\tilde{R}, \tilde{\psi})$ in the neighborhood of a point (θ_0, z_0) on the boundary $\rho = 1$. We are particularly interested in the behavior as a function of z of solutions $(\tilde{R}, \tilde{\psi})$ which have high frequency components in the angular variable θ. Thus, we shall consider solutions whose initial values at $z = z_0$ are of the form

$$\tilde{R}(\theta, z_0) = e^{i\omega l_1(\theta)}R_0(\theta),$$

and similarly for $\tilde{\psi}$, where ω is a positive parameter and l_1 is a function of θ. We will then construct a formal asymptotic series for $(\tilde{R}, \tilde{\psi})$ in positive powers of $1/\omega$. This approach is similar to that of Lax [7].

It is a result of the analysis that powers of ω to half-integer powers enter the expansion in a natural way. Thus, we look for solutions to (8.2)–(8.4) of the form

$$\tilde{R} = e^{i(\omega l_1 + \omega^{1/2} l_{1/2})}\sum_{j=0}^{\infty} R_{j/2}(\theta, z)\omega^{-j/2}, \tag{8.5}$$

$$\tilde{\psi} = e^{i(\omega l_1 + \omega^{1/2} l_{1/2})} \sum_{j=0}^{\infty} \{\rho^{\omega k_1 + \omega^{1/2} k_{1/2}} \psi_{j/2}(\theta, z, \rho) + \hat{\psi}_{j/2}(\theta, z, \rho)\} \omega^{-j/2}, \tag{8.6}$$

where $\omega \gg 1$ and $l_1(\theta, z)$, $l_{1/2}(\theta, z)$, $k_1(\theta, z)$, $k_{1/2}(\theta, z)$, $R_{j/2}$, $\psi_{j/2}$ and $\hat{\psi}_{j/2}$ are all functions to be determined. The first term in the brackets in the expansion (8.6) represents a solution of the homogeneous version of (8.4) (i.e., (8.4) with $\tilde{R} \equiv 0$), while the second term represents a particular solution of this equation corresponding to $\tilde{R}$ given by (8.5).

Substituting the expansions (8.5) and (8.6) into (8.4) and equating coefficients of like powers of ω results in the relations

$$\hat{\psi}_0 = \rho \frac{\partial \bar{\psi}}{\partial \rho} R_0, \qquad \hat{\psi}_{1/2} = \rho \frac{\partial \bar{\psi}}{\partial \rho} R_{1/2}, \tag{8.7}$$

and

$$k_1 = \frac{1}{1 - i\bar{\beta}} \frac{\partial l_1}{\partial \theta}, \qquad k_{1/2} = \frac{1}{1 - i\bar{\beta}} \frac{\partial l_{1/2}}{\partial \theta}, \quad \text{if } \frac{\partial l_1}{\partial \theta} \geqq 0, \tag{8.8}$$

and

$$k_1 = -\frac{1}{1 + i\bar{\beta}} \frac{\partial l_1}{\partial \theta}, \qquad k_{1/2} = -\frac{1}{1 + i\bar{\beta}} \frac{\partial l_{1/2}}{\partial \theta}, \quad \text{if } \frac{\partial l_1}{\partial \theta} < 0. \tag{8.9}$$

In the following we will assume without loss of generality that $\partial l_1/\partial \theta$ is nonnegative, since by taking the complex conjugate of (8.5) and (8.6) we obtain a solution of similar form with $\partial l_1/\partial \theta$ of opposite sign.

We now substitute the expansions (8.5) and (8.6) into (8.2) and (8.3) and equate like powers of ω, using the relations (8.7) and (8.8). The terms containing first powers of ω give the equations

$$2\bar{R} R_0 \left(\frac{\partial l_1}{\partial z} - \lambda \frac{\partial l_1}{\partial \theta} \right) = \psi_0 \frac{\partial l_1}{\partial \theta},$$

$$2\bar{R} \psi_0 \left(\frac{\partial l_1}{\partial z} - \lambda \frac{\partial l_1}{\partial \theta} \right) = 0,$$

where $\lambda = (\bar{\beta}(\partial \bar{\psi}/\partial \rho) - (\partial \bar{\psi}/\partial \theta))/2\bar{R}$.

From this we conclude that $\psi_0 \equiv 0$ and

$$\frac{\partial l_1}{\partial z} - \lambda \frac{\partial l_1}{\partial \theta} = 0. \tag{8.10}$$

The terms with $\omega^{1/2}$ give the equation

$$\frac{\partial l_{1/2}}{\partial z} - \lambda \frac{\partial l_{1/2}}{\partial \theta} = -\frac{i}{2\bar{R}} \frac{\partial l_1}{\partial \theta} \frac{\psi_{1/2}}{R_0}. \tag{8.11}$$

Equation (8.10) is similar to the eikonal equation of geometrical optics, and in particular it shows that l_1 which is initially real will remain so, being constant along the characteristics given by

$$\frac{d\theta}{dz} + \lambda(\theta, z) = 0.$$

Thus this system has real characteristics as do hyperbolic systems. From (8.11) we see

that $l_{1/2}$ will not be real unless $\psi_{1/2}/R_0$ is purely imaginary. If

$$\operatorname{Im} l_{1/2}(z, \theta) < 0,$$

then from (8.5) we see that the amplitude of the solution will grow as

$$e^{-\omega^{1/2} \operatorname{Im} l_{1/2}},$$

and thus the initial value problem for (8.2) and (8.3) will be ill-posed in the sense of Hadamard (see Kreiss [6]).

From the zeroth order terms in ω we obtain the relation

$$\frac{\partial l_1}{\partial \theta}\left(\frac{\psi_{1/2}}{R_0}\right)^2 = 2\bar{R}\alpha, \tag{8.12}$$

where

$$\alpha = \lambda \frac{\partial^2 \bar{\psi}}{\partial \rho\, \partial \theta} - 2\frac{\partial \bar{\psi}}{\partial z} - \frac{3}{4}\frac{\bar{R}}{(1+z)^2} - \frac{\partial^2 \bar{\psi}}{\partial \rho\, \partial z} + \frac{1}{R}\frac{\partial \bar{R}}{\partial z}\frac{\partial \bar{\psi}}{\partial \rho}.$$

Thus, we see that if α is nonpositive then $\psi_{1/2}/R_0$ will be purely imaginary and the amplitude of the solution will not grow with ω, while if α is positive then the high frequency perturbations will grow exponentially as $e^{\omega^{1/2}|\operatorname{Im} l_{1/2}|}$. Thus, it is a necessary condition for well-posedness that α be nonpositive.

In the numerical experiments the quantity α was approximated using (formally) first-order accurate one-sided differences. It was found that this approximation to α was negative throughout most of the computation. At those values of z for which the thin sheets were not being adequately resolved, α became positive. However, the solution was not smooth, so that the computation of α may have been so inaccurate as to be meaningless.

It appears then that the system of equations (2.9)–(2.11) is well-posed for each of the examples considered here, at least for those values of z for which the solution has been computed. We conjecture that it is well-posed for all values of z. We believe that the break-up of the numerical solution is purely a numerical phenomenon caused by inadequate resolution, and not caused by a loss of well-posedness of the differential equations.

As a demonstration of the similarity of our system to hyperbolic systems, we numerically integrated the system with the initial data of Example 2 from $z = 0$ to $z = 2$ and then integrated back to $z = 0$. The initial conditions were recovered to within the numerical accuracy of the method. Thus this system is reversible as are hyperbolic systems.

Finally, we offer the following comments on the stability of the difference scheme which is described in § 3. As in the above discussion of the well-posedness of the differential equations, all that can be done is to analyze the stability of the linearized problem. The analysis of the stability of the linearized system mimics the above analysis of the well-posedness although it involves more complicated algebraic expressions. The amplification factor of the von Neumann analysis corresponds to the factor $e^{i(\omega l_1 + \omega^{1/2} l_{1/2})}$. The actual details of the derivation are omitted in the interests of brevity.

REFERENCES

[1] P. GARABEDIAN, *Estimation of the relaxation factor for small mesh sizes*, Math Tables and Other Aids to Computation, 10 (1965), pp. 183–185.

[2] J. F. GEER, *Slender streams with gravity: Outer asymptotic expansions* I, Phys. Fluids, 20 (1977), pp. 1613–1621.

[3] ——— *Slender streams with gravity: Outer asymptotic expansions* II, Phys. Fluids, 20 (1977), pp. 1622–1630.
[4] J. F. GEER AND J. C. STRIKWERDA *Vertical slender jets*, J. Fluid Mech., to appear.
[5] A. E. GREEN, *On the steady motion of jets with elliptical sections*, Acta Mech., 26 (1977), pp. 171–177.
[6] H. O. KREISS, *Über Sachgemässe Cauchyprobleme*, Math. Scand., 13 (1963), pp. 109–123.
[7] P. D. LAX *Asymptotic solutions of oscillatory initial value problems*, Duke Math. J., 24 (1957), pp. 627–646.
[8] R. W. MACCORMACK *The effect of viscosity in hypervelocity impact cratering*, AIAA Hypervelocity Impact Conference, AIAA Paper No. 69-354, April, 1969.
[9] LORD RAYLEIGH, *On the capillary phenomena of jets*, Proc. Roy. Soc., 29 (1879), pp. 71–97.
[10] ——— *The Theory of Sound*, 2nd edition, Dover, New York, 1945.
[11] E. O. TUCK, *The shape of free jets of water under gravity*, J. Fluid Mech., 76 (1976), pp. 625 ff.
[12] D. G. WILSON, A. D. SOLOMON AND P. T. BOGGS, *Moving Boundary Problems*, Academic Press, New York, 1978.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/80/0104-0007 $01.00/0

SOLVING FINITE DIFFERENCE APPROXIMATIONS TO NONLINEAR TWO-POINT BOUNDARY VALUE PROBLEMS BY A HOMOTOPY METHOD*

LAYNE T. WATSON†

Abstract. The Chow–Yorke algorithm is a homotopy method that has been proved globally convergent for Brouwer fixed point problems, classes of zero finding, nonlinear programming, and two-point boundary value problems. The method is numerically stable, and has been successfully applied to several practical nonlinear optimization and fluid dynamics problems. Previous application of the homotopy method to two-point boundary value problems has been based on shooting, which is inappropriate for fluid dynamics problems with sharp boundary layers. Here the Chow–Yorke algorithm is proved globally convergent for a class of finite difference approximations to nonlinear two-point boundary value problems. The numerical implementation of the algorithm is briefly sketched, and computational results are given for two fairly difficult fluid dynamics boundary value problems.

Key words. homotopy method, Chow–Yorke algorithm, globally convergent, two-point boundary value problem, finite differences, fixed point computation, nonlinear equations

1. Introduction. For the user of mathematical software, the ideal subroutine is a black box. The user specifies the problem and perhaps an error criterion, and the routine returns an answer and perhaps an indication of whether or not the error criterion was (likely) met. Such software exists for eigenvalues and eigenvectors, linear systems of equations, nonstiff initial value problems, and one-dimensional quadrature. For the first two areas, the sophistication and ease of use of EISPACK and LINPACK are well known. For the latter two areas, relieving the user of the burden of choosing step sizes or mesh points was a tremendous advance. The state of affairs with regard to nonlinear systems of equations is much more primitive, however. Although there is a huge amount of theory on nonlinear systems (for example, [14]), there is not software yet which approaches the black box ideal. Almost all the theory concerns locally convergent methods, which place the (sometimes extremely difficult) burden of choosing a good starting point on the user. Some of the recent quasi-Newton (or least change secant update) methods are robust and efficient, but the inescapable fact is that they still require good starting points (which is often not sufficiently emphasized). For example, with a Brouwer fixed point problem $x = f(x)$, where f maps some ball into itself, a fixed point exists, but the better quasi-Newton methods converge (as they are designed to [31], [32]) to a local minimum of $\|x - f(x)\|$. Typically economics and fluid dynamics problems have many such local minima nowhere near the fixed point, so a good starting point is essential.

A globally convergent algorithm (one that produces the correct answer regardless of its starting point) would seem to be the best foundation for nonlinear equations software. A globally convergent algorithm for nonlinear equations sounds too good to be true. Actually, for Brouwer fixed points and a very large class of nonlinear systems of equations $F(x) = 0$, there are at least three distinct globally convergent algorithms. Because these algorithms are grounded in topology and differential geometry, were not discovered and advocated by numerical analysts and were inefficient in their early implementations, they are not widely known or understood by numerical analysts. The excellent survey by Allgower and Georg [1] is an attempt to remedy that. The three algorithms are based on simplicial approximations (Eaves–Saigal [6], [12], [15]),

* Received by the editors September 4, 1979.

† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061. This work was supported by the National Science Foundation under grant MCS 7821337.

retraction mappings (Kellog–Li–Yorke [9]), and a parameterized Sard's theorem (Chow–Mallet–Paret–Yorke [4], [28]). Details can be found in the original references or [1]. This paper concentrates on the Chow–Yorke algorithm [27], the only one of the three that is both numerically stable and easy to understand.

The idea of the Chow–Yorke algorithm, various aspects of which have been around for a long time [1], [2], [5], [7], [8], [10], [11], [13], [14], is to track the zero curve of the homotopy map

$$\rho_a(\lambda, x) = \lambda f(x) + (1-\lambda)(x-a)$$

emanating from $\lambda = 0$, $x = a$ until a zero of $f(x)$ is reached at $\lambda = 1$. Superficially this resembles standard embedding [5], but there are two important differences. Here λ is *not* the embedding parameter, but is a dependent variable just as x. Hence λ can increase and decrease along the zero curve, and need not increase monotonically from 0 to 1 (as in standard embedding techniques). Furthermore, there are never any "singular points" along the zero curve, and turning points pose no special difficulty [28].

To be useful the Chow–Yorke algorithm must be globally convergent on the types of problems people actually solve. It has been proven globally convergent for Brouwer fixed points [4], the nonlinear complementarity problem [29], convex optimization problems with nonnegativity constraints [26], and some two-point boundary value problems via shooting [30]. Actually, the mathematical theory proves global convergence with probability one (i.e., it can fail only for starting points in a set of Lebesgue measure zero), but that has no practical significance since the exceptional set is nowhere dense. The intent of this paper is to prove the global convergence of the Chow–Yorke algorithm for the finite difference approximations to a class of nonlinear two-point boundary value problems, sketch the essence of the Chow–Yorke algorithm and give some numerical results for two fluid dynamics problems. The intent here is not to prove very general theorems, but merely to justify the application of the Chow–Yorke algorithm to finite difference approximations of simple two-point boundary value problems. The computational results are on problems to which the theory is not obviously applicable.

Some theoretical results are presented in § 2. The proofs rely heavily on theorems and descriptions published elsewhere, but at least the necessary facts are stated here. Section 3 sketches the numerical algorithm, which is described in detail in [28]. Computational results are given in § 4. They are described in detail to make it clear exactly what equations were solved and what the numerical experiments were.

Notation. E^n denotes n-dimensional Euclidean space; x_i denotes the ith component of a vector $x \in E^n$; the inner product $x^t y$ is simply written xy.

2. Theory. Consider first the simple two-point boundary value problem

$$y''(x) = f(x, y(x), y'(x)), \qquad 0 \leqq x \leqq 1, \tag{1}$$

$$y(0) = y(1) = 0, \tag{2}$$

where $y(x)$ is a scalar function and $f(x, u, v)$ is C^2. General boundary conditions and $y(x)$ a vector will be considered later. Partition the interval $[0, 1]$ into $n+1$ equal subintervals of length $h = 1/(n+1)$, let $x_i = ih$, $i = 0, \cdots, n+1$, Y_i be an approximation to an exact solution $y(x)$ at x_i. The following standard finite difference approximations will be used:

$$y''(x_i) = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} + O(h^2), \tag{3}$$

$$y'(x_i)=\frac{y(x_{i+1})-y(x_{i-1})}{2h}+O(h^2), \tag{4}$$

$$y'(x_0)=\frac{-3y(x_0)+4y(x_1)-y(x_2)}{2h}+O(h^2), \tag{5}$$

$$y'(x_{n+1})=\frac{y(x_{n-1})-4y(x_n)+3y(x_{n+1})}{2h}+O(h^2). \tag{6}$$

Substituting (3), (4) into (1)–(2), neglecting terms of order h^2 and replacing $y(x_i)$ by Y_i results in

$$G(Y)=AY+h^2F^h(Y)=0, \tag{7}$$

where

$$Y=\begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ Y_n \end{bmatrix}, \qquad A=\begin{bmatrix} 2 & -1 & 0 & & & \\ -1 & 2 & -1 & & 0 & \\ 0 & -1 & 2 & \cdot & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & 0 & & & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

and $F_i^h(Y)=f(x_i, Y_i, (Y_{i+1}-Y_{i-1})/2h)$, $i=1, \cdots, n$. Thus the two-point boundary value problem (1)–(2) is approximated by the nonlinear system of equations (7). A homotopy method is used to solve (7). The following lemma from [29] will be useful.

LEMMA 1. *Let $F: E^n \to E^n$ be a C^2 map such that for some $r>0$, $xF(x)\geqq 0$ whenever $\|x\|=r$. Then F has a zero in $\{x \in E^n \mid \|x\| \leqq r\}$, and for almost all $a \in E^n$, $\|a\|<r$, there is a zero curve γ of*

$$\rho_a(\lambda, x)=\lambda F(x)+(1-\lambda)(x-a),$$

along which the Jacobian matrix $D\rho_a(\lambda, x)$ has full rank, emanating from $(0, a)$ and reaching a zero $\bar{x}$ of F at $\lambda=1$. Furthermore, γ has finite arc length if $DF(\bar{x})$ is nonsingular.

The function $\rho_a: [0, 1)\times E^n \to E^n$ in Lemma 1 is the homotopy map. It is important to note that: (1) λ need not increase monotonically along the zero curve γ; (2) the Jacobian matrix of the homotopy map has full rank at *every* point along γ (this feature is convenient but not crucial to the success of a numerical method; see [35]); and (3) γ is guaranteed to reach a zero of F with probability one.

THEOREM 1. *Let $F^h(Y)$ in (7) be a C^2 mapping, and suppose that*

$$\overline{\lim_{\|Y\|\to\infty}} \frac{\|F^h(Y)\|}{\|Y\|} \leqq 9. \tag{8}$$

For $W \in E^n$, define $\rho_W: [0, 1)\times E^n \to E^n$ by

$$\rho_W(\lambda, Y)=\lambda G(Y)+(1-\lambda)(Y-W).$$

Then for almost all $W \in E^n$ there exists a zero curve γ of ρ_W, along which the Jacobian matrix $D\rho_W(\lambda, Y)$ has full rank, emanating from $(0, W)$ and reaching a zero $\tilde{Y}$ of G (at $\lambda=1$). Furthermore, if $DG(\tilde{Y})$ is nonsingular, then γ has finite length.

Proof. By Lemma 1, it is sufficient to prove that $YG(Y)\geqq 0$ for all Y sufficiently large. The matrix A in (7) is symmetric and positive definite with smallest eigenvalue

$2(1-\cos \pi/(n+1)) \geqq 9.5/(n+1)^2 = 9.5h^2$ (for $n \geqq 4$) [5]. By hypothesis, there exists $r>0$ such that $\|F^h(Y)\| \leqq 9.4\|Y\|$ for $\|Y\| \geqq r$. Now for $\|Y\| \geqq r$,

$$(9) \quad YG(Y) = YAY + h^2 YF^h(Y) \geqq (9.5h^2)\|Y\|^2 - h^2\|Y\|(9.4\|Y\|) = .1h^2\|Y\|^2 > 0$$

from the symmetry of A and the Cauchy–Schwarz inequality. □

COROLLARY 1. *If the condition* (8) *in Theorem* 1 *is replaced by* $\overline{\lim}_{\|Y\|\to\infty}\|F^h(Y)\|/\|Y\| \leqq C < \pi^2$ *for* $0<h<h_0$, *then there exists* $h_1>0$ *such that Theorem* 1 *holds for* $0<h<h_1$.

Proof. For n large enough, the smallest eigenvalue $\lambda_1 = 2(1-\cos \pi/(n+1))$ of A can be made arbitrarily close to $\pi^2/(n+1)^2 = \pi^2 h^2$ from below. Also for $\|Y\|$ sufficiently large, $\|F^h(Y)\| \leqq (C+\delta)\|Y\| < \pi^2\|Y\|$. Therefore for n and $\|Y\|$ large enough, $\pi^2 h^2 > \lambda_1 > (C+\delta)h^2$, which when used in (9) gives the desired result. n large enough translates to h small enough, so the theorem holds for $0<h<h_1$. □

COROLLARY 2. *The conclusion of Theorem* 1 *holds if* $f(x, u, v)$ *in* (1) *is a* C^2 *mapping and bounded.*

Consider now the differential equation (1) with the general boundary conditions

$$(10) \qquad C\begin{bmatrix} y(0) \\ y'(0) \\ y(1) \\ y'(1) \end{bmatrix} = \begin{bmatrix} \alpha_{11}y(0)+\alpha_{12}y'(0)+\alpha_{13}y(1)+\alpha_{14}y'(1) \\ \alpha_{21}y(0)+\alpha_{22}y'(0)+\alpha_{23}y(1)+\alpha_{24}y'(1) \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b,$$

with rank $C = 2$. Making the finite difference substitutions (3)–(6) in (1) and (10) leads to a system of equations of the form

$$(11) \qquad \mathbf{G}(\mathbf{Y}) = \mathbf{A}^h\mathbf{Y} + h^2\mathbf{F}^h(\mathbf{Y}) = 0,$$

where $\mathbf{Y} = (Y_0, \cdots, Y_{n+1})^t$, $\mathbf{A}^h$ is a matrix of order $n+2$ representing the linear differential operator and boundary conditions, and $\mathbf{F}^h$ is a nonlinear operator arising from the nonlinear term of (1). See Fig. 1.

$$\begin{bmatrix} \alpha_{11}h-\frac{3}{2}\alpha_{12} & 2\alpha_{12} & -\frac{1}{2}\alpha_{12} & 0 & \cdots & 0 & \frac{1}{2}\alpha_{14} & -2\alpha_{14} & \alpha_{13}h+\frac{3}{2}\alpha_{14} \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 2 & -1 \\ \alpha_{21}h-\frac{3}{2}\alpha_{22} & 2\alpha_{22} & -\frac{1}{2}\alpha_{22} & 0 & \cdots & 0 & \frac{1}{2}\alpha_{24} & -2\alpha_{24} & \alpha_{23}h+\frac{3}{2}\alpha_{24} \end{bmatrix} \times \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_{n-2} \\ Y_{n-1} \\ Y_n \\ Y_{n+1} \end{bmatrix} + \begin{bmatrix} -hb_1 \\ h^2 f\left(x_1, Y_1, \frac{Y_2-Y_0}{2h}\right) \\ \cdot \\ \cdot \\ \cdot \\ h^2 f\left(x_n, Y_n, \frac{Y_{n+1}-Y_{n-1}}{2h}\right) \\ -hb_2 \end{bmatrix} = 0$$

FIG. 1. *Detail for equation* (11).

The following two lemmas are from [30].

LEMMA 2. *Let* $g: E^m \to E^m$ *be a* C^2 *map and define* $\rho_a: [0, 1)\times E^m \to E^m$ *by*

$$\rho_a(\lambda, y) = \lambda g(y) + (1-\lambda)(y-a).$$

Then for almost all $a \in E^m$ *there is a zero curve* γ *of* ρ_a *emanating from* $(0, a)$ *along which* $D\rho_a(\lambda, y)$ *has full rank.*

LEMMA 3. *If the zero curve* γ *in Lemma* 2 *is bounded, it has an accumulation point* $(1, \bar{y})$, *where* $g(\bar{y}) = 0$. *Furthermore, if* $Dg(\bar{y})$ *is nonsingular, then* γ *has finite arc length.*

THEOREM 2. *Let* $\mathbf{F}^h(\mathbf{Y})$ *in* (11) *be a* C^2 *mapping, and suppose that* $\mathbf{G}$ *satisfies one of the following*:

1) *there exists* $r>0$ *such that* $\mathbf{Y}-\mathbf{W}$ *and* $\mathbf{G}(\mathbf{Y})$ *do not point in opposite directions for* $\|\mathbf{Y}\|=r$, $\|\mathbf{W}\|<r$;

2) *there exists* $r>0$ *such that* $\mathbf{YG}(\mathbf{Y})\geqq 0$ *for* $\|\mathbf{Y}\|=r$;

3) $\mathbf{A}^h$ *is positive semidefinite* ($\mathbf{YA}^h\mathbf{Y}\geqq 0$ *for all* $\mathbf{Y}$), *and there exists* $r>0$ *such that*

$$\mathbf{YF}^h(\mathbf{Y})\geqq 0 \quad \text{for } \|\mathbf{Y}\|=r.$$

For $\mathbf{W}\in E^{n+2}$, *define* $\rho_{\mathbf{W}}:[0,1)\times E^{n+2}\to E^{n+2}$ *by*

$$\rho_{\mathbf{W}}(\lambda,\mathbf{Y})=\lambda\mathbf{G}(\mathbf{Y})+(1-\lambda)(\mathbf{Y}-\mathbf{W}).$$

Then for almost all $\mathbf{W}\in E^{n+2}$, $\|\mathbf{W}\|<r$, *there exists a zero curve* γ *of* $\rho_{\mathbf{W}}$, *along which the Jacobian matrix* $D\rho_{\mathbf{W}}(\lambda,\mathbf{Y})$ *has full rank, emanating from* $(0,\mathbf{W})$ *and reaching a zero* $\tilde{\mathbf{Y}}$ *of* $\mathbf{G}$ (*at* $\lambda=1$), *Furthermore, if* $D\mathbf{G}(\tilde{\mathbf{Y}})$ *is nonsingular, then* γ *has finite arc length.*

Proof. It is sufficient to prove the theorem under condition 1), since 3) implies 2), which in turn implies 1). Condition 1) says that $\rho_{\mathbf{W}}\neq 0$ for $0\leqq\lambda<1$ on the surface of the ball $\|\mathbf{Y}\|\leqq r$. Therefore γ, if it exists, must lie entirely within the ball $\|\mathbf{Y}\|\leqq r$. The existence of γ for almost all $\mathbf{W}$ and γ reaching $\lambda=1$ follows directly from Lemmas 2 and 3 above. The finite arc length of γ for $D\mathbf{G}(\tilde{\mathbf{Y}})$ nonsingular also follows from Lemma 3. □

COROLLARY 1. *Suppose the matrix* $\mathbf{A}^h$ *in* (11) *is positive definite and* $f(x,u,v)$ *in* (1) *is a bounded* C^2 *mapping. Then the conclusion of Theorem* 2 *holds.*

Proof. f bounded implies $\mathbf{F}^h$ in (11) is bounded, and therefore $h^2\mathbf{YF}^h(\mathbf{Y})=O(\|\mathbf{Y}\|)$. Let $\eta=\min_{\|\mathbf{Y}\|=1}\mathbf{YA}^h\mathbf{Y}>0$. Then $\mathbf{YG}(\mathbf{Y})=\mathbf{YA}^h\mathbf{Y}+h^2\mathbf{YF}^h(\mathbf{Y})\geqq\eta\|\mathbf{Y}\|^2+O(\|\mathbf{Y}\|)>0$ for $\|\mathbf{Y}\|$ large enough, which is condition 2) in Theorem 2. □

COROLLARY 2. *If* $f(x,u,v)$ *in* (1) *is a bounded* C^2 *mapping and the boundary conditions* (10) *are of the form*

$$y(0)=b_1, \qquad y(1)=b_2,$$

then the conclusion of Theorem 2 *holds.*

Proof. By Corollary 1, it is sufficient to show that the matrix $\mathbf{A}^h$ in (11) is positive definite. With these boundary conditions,

$$\mathbf{A}^h=\begin{bmatrix} 1 & 0 & 0 & & & & \\ -1 & 2 & -1 & & & & \\ 0 & -1 & 2 & & & & \\ & & & \ddots & & & \\ & & & & 2 & -1 & 0 \\ & & & & -1 & 2 & -1 \\ & & & & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{YA}^h\mathbf{Y}=Y_0^2-Y_0Y_1+Y_1^2+\sum_{i=1}^{n-1}(Y_{i+1}-Y_i)^2+Y_n^2-Y_nY_{n+1}+Y_{n+1}^2>0 \quad \text{for } \mathbf{Y}\neq\mathbf{0}. \ \square$$

Condition 1) in Theorem 2 is the most general situation in which the homotopy method is guaranteed globally convergent. However, it is virtually impossible to verify in practice, with a few notable exceptions [29]. Condition 3) is easier to verify, but is

frequently not satisfied for practical problems. For most practical problems the homotopy method works very well, even though the above theory is not directly or demonstrably applicable.

Finally, consider the case where $y(x)=(y_1(x), \cdots, y_m(x))$ in (1) is an m-dimensional vector function. Let $Y_i^{(k)}$ be an approximation to the exact solution $y_k(x_i)$ and $Y=(Y^{(1)}, \cdots, Y^{(m)}) \in E^{mn}$. Then the matrix in (7) becomes a block diagonal matrix with each diagonal block the same as A in (7), and $F^h(Y)$ is defined accordingly. The proofs of Theorem 1 and its corollaries are valid for the vector case also, so

THEOREM 3. *Theorem* 1 *and its corollaries are valid for the two-point boundary value problem* (1)–(2), *where* $y(x)=(y_1(x), \cdots, y_m(x))$ *is an m-dimensional vector function and* Y, A, $F^h(Y)$ *in* (7) *are defined as described above.*

The vector case with general boundary conditions (10) is not so easy, because there are many ways of forming the matrix $\mathbf{A}^h$ in (11). The most advantageous way to merge the boundary conditions with the finite difference matrix to form $\mathbf{A}^h$ depends on the problem. Without being more specific, assume $\mathbf{A}^h$ in (11) is formed in some reasonable way, and that $\mathbf{F}^h(\mathbf{Y})$ is defined accordingly. Here $\mathbf{Y} \in E^{m(n+2)}$. The proof of Theorem 2 carries over to the vector case, but not necessarily the corollaries' proofs, since they depended on a specific structure in $\mathbf{A}^h$. Hence:

THEOREM 4. *Theorem* 2 *is valid for the two-point boundary value problem* (1), (10), *where* $y(x)=(y_1(x), \cdots, y_m(x))$ *is an m-dimensional vector function,* C *is a* $2m \times 4m$ *matrix of rank* $2m$, *and* $\mathbf{Y}$, $\mathbf{A}^h$, $\mathbf{F}^h(\mathbf{Y})$ *in* (11) *are defined such that* (11) *is a consistent approximation to* (1), (10).

The boundary condition approximations based on (5), (6) are just one of many possibilities, and there was no particular reason for that choice. For example, adding an extra point to the left of zero or staggering the mesh points around zero and then using a central difference approximation for $y'(0)$ could have been done.

3. Algorithm. The general idea of the algorithm is apparent from Theorems 1 and 2: just follow the zero curve γ emanating from $(0, W)$ until a zero $\tilde{Y}$ of $G(Y)$ is reached (at $\lambda = 1$). Of course it is nontrivial to develop a viable numerical algorithm based on that idea, but at least conceptually, the algorithm is clear and simple. The numerical algorithm was described in detail in [28], and various aspects and applications of it are in [20]–[30]. [27] contains computer code for the algorithm. Since the algorithm has been thoroughly described elsewhere, only a brief outline of it and how it differs from standard continuation will be given here. The homotopy map is

$$\rho_W(\lambda, Y)=\lambda G(Y)+(1-\lambda)(Y-W),$$

which has the same form as a standard continuation or embedding mapping. However, there are two crucial differences. In standard continuation, the embedding parameter λ increases monotonically from 0 to 1 as the trivial problem $Y-W=0$ is continuously deformed to the problem $G(Y)=0$. The present homotopy method permits λ to both increase and decrease along γ with no adverse effect; that is, turning points present no special difficulty. The second important difference is that there are never any "singular points" which plague standard continuation methods. The way in which the zero curve γ of ρ_W is followed and the full rank of $D\rho_W$ along γ guarantee this. Observe that Lemma 2 guarantees that γ cannot just "stop" at an interior point of $[0, 1) \times E^n$.

Parameterize γ by arc length s so $\lambda=\lambda(s)$, $Y=Y(s)$ along γ. Then

$$\rho_W(\lambda(s), Y(s))=0$$

and

$$\frac{d}{ds}\rho_W(\lambda(s), Y(s)) = 0, \tag{12}$$

$$\left\|\left(\frac{d\lambda}{ds}, \frac{dY}{ds}\right)\right\|_2 = 1. \tag{13}$$

If we take

$$\lambda(0) = 0, \qquad Y(0) = W, \tag{14}$$

the zero curve γ is the trajectory of the initial value problem (12)–(14). When $\lambda(\bar{s}) = 1$, the corresponding $Y(\bar{s})$ is a zero of $G(Y)$. Thus all the sophisticated ODE techniques currently available can be brought to bear on the problem of tracking γ [17], [18].

ODE software requires $(d\lambda/ds, dY/ds)$ explicitly, and (12), (13) only implicitly define the derivative $(d\lambda/ds, dY/ds)$. This can be calculated by finding the kernel of the $n \times (n+1)$ matrix

$$D\rho_W(\lambda(s), Y(s)),$$

which has full rank by Lemma 2. It is here that a substantial amount of computation is incurred, and it is imperative that the number of derivative evaluations be kept small. The recommended techniques for these calculations are given in [3], [28].

Remember that tracking γ was merely a means to an end, namely a zero $\tilde{Y}$ of $G(Y)$. Since γ itself is of no interest, one should not waste computational effort following it too closely. On the other hand, since γ is the only sure way to $\tilde{Y}$, losing γ can be fatal. The tradeoff between computational efficiency and reliability, and some practical advice based on computational experience, is also in [28].

4. Numerical results. All the results reported here were obtained on an IBM 370/158 using a double precision version of the code FIXPT in [27]. FIXPT was compiled with the FORTRAN H extended compiler, and the answers were obtained accurate to 8 places unless otherwise mentioned. The execution times are in seconds. The examples here are intended to show the performance of the algorithm on realistic problems, and that the homotopy method works even though the theory in § 2 is not obviously applicable.

The first example concerns the motion of a fluid squeezed between two parallel plates with prescribed normal velocity. The equations are [19], [30]:

$$S(\eta f''' + 3f'' + mf'f'' - ff''') = f^{(4)},$$

$$f(0) = f''(0) = 0, \quad f(1) = 1, \quad f'(1) = 0,$$

$$m = 0 \quad \text{(axisymmetric case)}.$$

With $x_1 = f$, $x_2 = f''$, the second order formulation is

$$x_1'' = x_2,$$

$$x_2'' = S(\eta x_2' + 3x_2 - x_1 x_2'),$$

$$x_1(0) = 0, \qquad x_1(1) = 1,$$

$$x_2(0) = 0, \qquad x_1'(1) = 0.$$

Note that the boundary conditions are unbalanced. This was handled by relating x_1, x_1', x_1'', and x_2 at $\eta = 1$. The nonlinear system $G(Z) = 0$ corresponding to (7) is given in Table 1.

TABLE 1.

Finite difference equations for squeezing problem.

$$
\begin{array}{c|c}
\begin{matrix} \eta_1 & & & & & \eta_n \end{matrix} & \begin{matrix} \eta_1 & & & & & & & & \eta_{n+1} \end{matrix} \\
\left[\begin{array}{cccccc}
2 & -1 & 0 & & & \\
-1 & 2 & -1 & & 0 & \\
0 & -1 & 2 & \ddots & & \\
 & & \ddots & \ddots & & \\
 & & & 2 & -1 & \\
 & 0 & & -1 & 2 &
\end{array}\right] & \left[\begin{array}{ccccccccc}
 & & & & & & & & \\
 & & & & & & 0 & & \\
 & & & & & & & & \\
 & & & & & & & & \\
 & & & & & & & & \\
 & & & & & & & & \\
\hline
0 & 0 & 2 & -1 & 0 & & & & \\
0 & 0 & -1 & 2 & -1 & & & 0 & \\
0 & 0 & 0 & -1 & 2 & \ddots & & & \\
\vdots & \vdots & & & \ddots & \ddots & & & \\
0 & 0 & & & & -1 & 2 & -1 & \\
1 & -8 & & 0 & & 0 & 0 & 2h^2 &
\end{array}\right]
\end{array}
\times
\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \\ z_n \\ z_{n+1} \\ z_{n+2} \\ z_{n+3} \\ \vdots \\ z_{2n} \\ z_{2n+1} \end{bmatrix}
+
\begin{bmatrix}
h^2 z_{n+1} \\
h^2 z_{n+2} \\
h^2 z_{n+3} \\
\vdots \\
h^2 z_{2n-1} \\
h^2 z_{2n} - 1 \\
h^2 S\left(\frac{z_{n+2}}{2} + 3z_{n+1} - z_1 \frac{z_{n+2}}{2h}\right) \\
h^2 S\left(2\frac{z_{n+3}-z_{n+1}}{2} + 3z_{n+2} - z_2 \frac{z_{n+3}-z_{n+1}}{2h}\right) \\
h^2 S\left(3\frac{z_{n+4}-z_{n+2}}{2} + 3z_{n+3} - z_3 \frac{z_{n+4}-z_{n+2}}{2h}\right) \\
\vdots \\
h^2 S\left(n\frac{z_{2n+1}-z_{2n-1}}{2} + 3z_{2n} - z_n \frac{z_{2n+1}-z_{2n-1}}{2h}\right) \\
7
\end{bmatrix}
= 0
$$

The dimension of the nonlinear system is NEQ $= 2n + 1$. Take $S = -4.226$. Starting from $W = 0$ with $n = 4$, the algorithm required 2.25 seconds of CPU time and 118 Jacobian evaluations, with an arc length of 20.0547. Starting from zero for larger n is expensive because most of the components of the solution are large, resulting in a very long zero curve γ, and demonstrating nothing other than the global convergence. The results are shown in Table 2, with these starting points:

$W = (0, 0, 1, 1, -1, -1, -1, -1, 2)$ $(n = 4)$

$W = (3 * 0, 3 * .5, 3 * 1, 3 * -.5, 6 * -1, 2)$ $(n = 9)$

$W = (3 * 0, 4 * .5, 7 * 1, 3 * -1, 3 * -2, 4 * -4, -2, -1, 0, 1, 2)$ $(n = 14)$

$W = (3 * 0, 6 * .5, 10 * 1, 4 * -1, 5 * -3, 5 * -4, -3, -2, -1, 0, 1, 2)$ $(n = 19)$.

TABLE 2.
Squeezing of a fluid between parallel plates.

n	4	9	14	19
NEQ	9	19	29	39
CPU time	2.0	11.37	27.11	63.39
Jacobian evaluations	108	139	118	127
arc length	8.1771	11.8233	6.0622	4.4201
$x_2(1)$	2.7033820	2.1986139	2.1178779	2.0902818
extrapolated values		2.0303579	2.0532891	2.0548011
			2.0561555	2.0553051
				2.0552484

The extrapolation was based on an asymptotic expansion in powers of h^2, which should hold since all the finite difference approximations were $O(h^2)$ accurate [5]. The last extrapolated value compares well to the exact solution $x_2(1) = f''(1) = 2.05514$. (It is known that centered difference methods may have difficulties on such problems [33], [34].)

In [30] this squeezing problem was solved by the same code FIXPT, using instead a nonlinear system, equivalent to the original problem, defined by shooting. The shooting approach produces more accurate solutions than the finite difference method here, and it also yields the first and third derivatives throughout the interval, which the method here does not. The only difficulties with shooting (on this problem) are instability and the magnitude of the solution (which involves higher derivatives), both due to the pronounced boundary layers [19]. The shooting approach took 53 seconds on an Amdahl 470 V6. Allowing for the different machines, the CPU times for the shooting method in [30] and the finite difference method here are comparable. Since the shooting technique in [30] is more accurate, yields more information, is easier to program, and has approximately the same execution time as the finite difference method here, it is clearly preferable (for this squeezing problem).

In many fluid dynamics problems, the boundary layers become more pronounced as certain parameters increase [20], [21], [22]. As these parameters increase, shooting becomes more and more unstable. In extreme cases shooting completely fails, because unless it is started right at the correct initial conditions the solution to the differential equation becomes so large that the ODE solver never reaches the other end of the interval. An example of such a problem is in [20]. Frequently a stretching transformation alleviates the problem, but this doesn't help if there are multiple boundary

layers whose locations change as the parameters change (as in [21], for example). Other possibilities are collocation and multiple shooting [5], but these were not tried. Shooting completely fails on the following problem, for the aforementioned reasons.

A model of the polar ice cap is given by the equations [22]:

$$
\begin{aligned}
&R(f'f''-ff''')=f^{(4)}+\gamma k',\\
&R(f'k-fk')=k''-\gamma f',\\
&R(gf'-fg')=g''+\gamma h+B,\\
&R(gk-fh')=h''-\gamma g,\\
&f(0)=-1,\quad f(1)=-\beta,\quad f'(0)=f'(1)=0,\\
&k(0)=k(1)=g(0)=g(1)=h(0)=h(1)=0.
\end{aligned}
$$

With $x_1=f$, $x_2=f''$, $x_3=g$, $x_4=h$, $x_5=k$, the second order formulation is

$$
\begin{aligned}
&x_1''=x_2,\\
&x_2''=R(x_1'x_2-x_1x_2')-\gamma x_5',\\
&x_3''=R(x_3x_1'-x_1x_3')-\gamma x_4-B,\\
&x_4''=R(x_3x_5-x_1x_4')+\gamma x_3,\\
&x_5''=R(x_1'x_5-x_1x_5')+\gamma x_1',\\
&x_1(0)=-1,\quad x_1(1)=-\beta,\quad x_1'(0)=x_1'(1)=0,\\
&x_3(0)=x_3(1)=x_4(0)=x_4(1)=x_5(0)=x_5(1)=0.
\end{aligned}
$$

The unbalanced boundary conditions are handled as in the previous problem. Let $z=(x_1(h),\cdots,x_1(nh),x_2(0),\cdots,x_2(1),x_3(h),\cdots,x_3(nh),x_4(h),\cdots,x_4(nh),x_5(h),$

TABLE 3.

Finite difference equations for polar ice cap problem (part 1).

$$
\begin{bmatrix}
2z_1-z_2\\
-z_1+2z_2-z_3\\
\vdots\\
-z_{n-2}+2z_{n-1}-z_n\\
-z_{n-1}+2z_n\\
-8z_1+z_2+2h^2z_{n+1}\\
-z_{n+1}+2z_{n+2}-z_{n+3}\\
-z_{n+2}+2z_{n+3}-z_{n+4}\\
\vdots\\
-z_{2n-1}+2z_{2n}-z_{2n+1}\\
-z_{2n}+2z_{2n+1}-z_{2n+2}\\
z_{n-1}-8z_n+2h^2z_{2n+2}\\
2z_{2n+3}-z_{2n+4}\\
-z_{2n+3}+2z_{2n+4}-z_{2n+5}\\
\vdots\\
-z_{3n}+2z_{3n+1}-z_{3n+2}\\
-z_{3n+1}+2z_{3n+2}
\end{bmatrix}
+
\begin{bmatrix}
h^2z_{n+2}+1\\
h^2z_{n+3}\\
\vdots\\
h^2z_{2n}\\
h^2z_{2n+1}+\beta\\
-7\\
(hR/2)((z_2+1)z_{n+2}-z_1(z_{n+3}-z_{n+1}))-(\gamma h/2)z_{4n+4}\\
(hR/2)((z_3-z_1)z_{n+3}-z_2(z_{n+4}-z_{n+2}))-(\gamma h/2)(z_{4n+5}-z_{4n+3})\\
\vdots\\
(hR/2)((z_n-z_{n-2})z_{2n}-z_{n-1}(z_{2n+1}-z_{2n-1}))-(\gamma h/2)(z_{5n+2}-z_{5n})\\
(hR/2)((-\beta-z_{n-1})z_{2n+1}-z_n(z_{2n+2}-z_{2n}))+(\gamma h/2)z_{5n+1}\\
-7\beta\\
(hR/2)(z_{2n+3}(z_2+1)-z_1z_{2n+4})-\gamma h^2z_{3n+3}-Bh^2\\
(hR/2)(z_{2n+4}(z_3-z_1)-z_2(z_{2n+5}-z_{2n+3}))-\gamma h^2z_{3n+4}-Bh^2\\
\vdots\\
(hR/2)(z_{3n+1}(z_n-z_{n-2})-z_{n-1}(z_{3n+2}-z_{3n}))-\gamma h^2z_{4n+1}-Bh^2\\
(hR/2)(z_{3n+2}(-\beta-z_{n-1})+z_nz_{3n+1})-\gamma h^2z_{4n+2}-Bh^2
\end{bmatrix}
=0
$$

$\cdots, x_5(nh)$). The finite difference equations are listed in Tables 3 and 4. The dimension of the nonlinear system is NEQ $= 5n + 2$.

TABLE 4.
Finite difference equations for polar ice cap problem (part 2).

$$\begin{bmatrix} 2z_{3n+3} - z_{3n+4} \\ -z_{3n+3} + 2z_{3n+4} - z_{3n+5} \\ \vdots \\ -z_{4n} + 2z_{4n+1} - z_{4n+2} \\ -z_{4n+1} + 2z_{4n+2} \\ 2z_{4n+3} - z_{4n+4} \\ -z_{4n+3} + 2z_{4n+4} - z_{4n+5} \\ \vdots \\ -z_{5n} + 2z_{5n+1} - z_{5n+2} \\ -z_{5n+1} + 2z_{5n+2} \end{bmatrix} + \begin{bmatrix} Rh^2(z_{2n+3}z_{4n+3} - z_1(z_{3n+4}/2h)) + \gamma h^2 z_{2n+3} \\ Rh^2(z_{2n+4}z_{4n+4} - z_2(z_{3n+5} - z_{3n+3})/2h) + \gamma h^2 z_{2n+4} \\ \vdots \\ Rh^2(z_{3n+1}z_{5n+1} - z_{n-1}(z_{4n+2} - z_{4n})/2h) + \gamma h^2 z_{3n+1} \\ Rh^2(z_{3n+2}z_{5n+2} + z_n(z_{4n+1}/2h)) + \gamma h^2 z_{3n+2} \\ (hR/2)((z_2+1)z_{4n+3} - z_1 z_{4n+4}) + (\gamma h/2)(z_2+1) \\ (hR/2)((z_3 - z_1)z_{4n+4} - z_2(z_{4n+5} - z_{4n+3})) + (\gamma h/2)(z_3 - z_1) \\ \vdots \\ (hR/2)((z_n - z_{n-2})z_{5n+1} - z_{n-1}(z_{5n+2} - z_{5n})) + (\gamma h/2)(z_n - z_{n-2}) \\ (hR/2)((-\beta - z_{n-1})z_{5n+2} + z_n z_{5n+1}) + (\gamma h/2)(-\beta - z_{n-1}) \end{bmatrix} = 0$$

Table 5 lists the results corresponding to these starting points:

$W = 0 \qquad (n = 4)$

$W = (-1, -.5, 0, .5, .5, 3 * 1, 2, 30, 20, 10, 5, 0, -2, -4, -6, -8, -10, -11,$
$18 * 0, 0, 6 * -.5, 2 * 0) \qquad (n = 9)$

$W = (-1, 3 * -.5, 0, 3 * .5, 3 * 1, 2 * 1.5, 2, 20, 17, 12, 7, 2, 0, -1, -2, -4, -6,$
$-8, -9, 4 * -11, 28 * 0, 2 * 0, -.2, 8 * -.5, -.1, 2 * 0) \qquad (n = 14)$

$W = (-1, 4 * -.5, 3 * 0, 3 * .5, 3 * 1, 3 * 1.5, 2 * 2, 23, 3 * 20, 10, 5, 0, -1, -2, -3, -4,$
$-5, -6, -7, -8, -9, 5 * -11, 38 * 0, 3 * 0, 3 * -.5, 3 * -1, 3 * -.6, 3 * -.3,$
$3 * -.1, 0) \qquad (n = 19).$

TABLE 5.
$\gamma = R = 30, \beta = -2, B = .5.$

n	4	9	14	19
NEQ	22	47	72	97
CPU time	18.4	114.8	483.7	1153.9
Jacobian evaluations	155	138	176	179
arc length	38.80661	18.47349	17.41116	20.33562
$f''(0)$	26.924488	23.570686	23.185378	23.068960
extrapolated values		22.452752	22.877132	22.919280
			22.930180	22.933329
				22.933539

For $\gamma = R = 20$ shooting is successful, although with difficulty. For $\gamma = R > 25$ (and $\beta = -2$) shooting completely fails. An interesting note is that shooting succeeds for $\gamma > 25$, $R > 25$, $\beta \geqq 0$, but becomes progressively worse as β decreases. As might be expected, the finite difference approach is impervious to the values of γ, R and β.

Table 6 lists the results for a different set of parameters with starting points:

$W = 0 \qquad (n = 4)$

$W = (-1, -.5, 0, 2 * .5, 3 * 1, 2, 25, 20, 10, 5, 0, -2, -4, -6, -8,$
$2 * -10, 27 * 0) \qquad (n = 9)$

$W = (-1, 3 * -.5, 0, 3 * .5, 3 * 1, 2 * 1.5, 2, 23, 20, 14, 7, 2, 0, -1, -2, -4,$
$-6, -8, -9, 4 * -11, 42 * 0) \qquad (n = 14)$

$W = (-1, 4 * -.5, 3 * 0, 3 * .5, 3 * 1, 3 * 1.5, 2 * 2, 23, 3 * 20, 10, 5, 0, -1, -2, -3,$
$-4, -5, -6, -7, -8, -9, 5 * -11, 57 * 0) \qquad (n = 19).$

TABLE 6.
$\gamma = 1, R = 50, \beta = -2, B = .5.$

n	4	9	14	19
NEQ	22	47	72	97
CPU time	17.4	141.3	546.8	1225.3
Jacobian evaluations	141	165	194	186
arc length	39.40252	13.53256	19.01427	22.85432
$f''(1)$	−10.525653	−11.197161	−11.299751	−11.337634
extrapolated values		−11.420997	−11.381823	−11.386341
			−11.376926	−11.387847
				−11.388575

For $\gamma = 1$, $\beta = -2$ shooting begins to fail around $R = 20$, but the finite difference approach handles $R = 50$ with no difficulty, requiring only slightly more time than for $R = 30$.

5. Conclusion. Three conclusions can be drawn from § 2 and § 4. First, the Chow–Yorke algorithm is theoretically applicable to the finite difference approximations to simple nonlinear two-point boundary value problems, and practically applicable to a much wider class than the present theory indicates. Second, the homotopy method is indeed globally convergent, and the code in [27] requires no jiggling of parameters and starting points to make it work. Third, and this observation has been made before [26], [28], the homotopy method levels the difficulty of problems. Trivial problems take almost as much time as extremely difficult problems. Whenever a quasi-Newton method does converge to the correct answer, the quasi-Newton method is usually at least an order of magnitude more efficient than the homotopy method. The question, of course, is how to know when a quasi-Newton method will work. For nonoscillatory well-conditioned problems Saigal and Todd's accelerated simplicial algorithm [16] is much more efficient than the Chow–Yorke algorithm, but the final verdict is not in yet. One final note: The differential geometry foundation of the Chow–Yorke algorithm is very powerful, and has been used to generate globally convergent nonlinear homotopies for problems on which the simplicial algorithms fail [25].

Acknowledgment. The author is indebted to Gene Golub, Gene Allgower and the referees for their comments and constructive criticisms.

REFERENCES

[1] E. ALLGOWER AND K. GEORG, *Simplicial and continuation methods for approximating fixed points*, SIAM Rev., 22 (1980), pp. 28–85.

[2] P. BOGGS, *The solution of nonlinear systems of equations by A-stable integration techniques*, SIAM J. Numer. Anal., 8 (1971), pp. 767–785.

[3] P. BUSINGER AND G. H. GOLUB, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

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

[5] G. DAHLQUIST, A. BJORCK AND N. ANDERSON, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[6] B. C. EAVES AND R. SAIGAL, *Homotopies for computation of fixed points on unbounded regions*, Math. Programming, 3 (1972), pp. 225–237.

[7] H. B. KELLER, *Numerical Solution of Two-point Boundary Value Problems*, Society for Industrial and Applied Mathematics, Philadelphia, 1976.

[8] ———, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, Academic Press, New York, 1977.

[9] R. B. KELLOGG, T. Y. LI AND J. YORKE, *A constructive proof of the Brouwer fixed-point theorem and computational results*, SIAM J. Numer. Anal., 13 (1976), pp. 473–483.

[10] R. W. KLOPFENSTEIN, *Zeros of nonlinear functions*, J. Assoc. Comput. Mech., 8 (1961), pp. 336–373.

[11] M. KUBICEK, *Dependence of solutions of nonlinear systems on a parameter*, ACM-TOMS, 2 (1976), pp. 98–107.

[12] O. MERRILL, *Applications and extensions of an algorithm to compute fixed points of upper semicontinuous mappings*, Doctoral thesis, I.O.E. Dept., University of Michigan, Ann Arbor, MI, 1972.

[13] G. MEYER, *On solving nonlinear equations with a one-parameter operator embedding*, SIAM J. Numer. Anal., 5 (1968), pp. 739–752.

[14] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.

[15] R. SAIGAL, *On the convergence rate of algorithms for solving equations that are based on methods of complementary pivoting*, Math. Operations Res., 2 (1977), pp. 108–124.

[16] R. SAIGAL AND M. J. TODD, *Efficient acceleration techniques for fixed point algorithms*, SIAM J. Numer. Anal., 15 (1978), pp. 997–1007.

[17] 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.

[18] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations : The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.

[19] C. Y. WANG, *The squeezing of a fluid between two plates*, J. Appl. Mech., 43 (1976), pp. 579–583.

[20] C. Y. WANG AND L. T. WATSON, *Squeezing of a viscous fluid between elliptic plates*, Appl. Sci. Res., 35 (1979), pp. 195–207.

[21] ———, *Viscous flow between rotating discs with injection on the porous disc*, Z. Angew. Math. Phys., 30 (1979), pp. 773–787.

[22] L. T. WATSON, *Numerical study of porous channel flow in a rotating system by a homotopy method*, J. Comput. Appl. Math., to appear.

[23] L. T. WATSON, T. Y. LI AND C. Y. WANG, *Fluid dynamics of the elliptic porous slider*, J. Appl. Mech., 45 (1978), pp. 435–436.

[24] L. T. WATSON AND C. Y. WANG, *Deceleration of a rotating disc in a viscous fluid*, Phys. Fluids, 22 (1979), pp. 2267–2269.

[25] L. T. WATSON AND W. H. YANG, *Optimal design by a homotopy method*, Applicable Anal., 10 (1980), pp. 275–284.

[26] L. T. WATSON, *Computational experience with the Chow–Yorke algorithm*, Math. Programming, 19 (1980), pp. 92–101.

[27] L. T. WATSON AND D. FENNER, *Chow–Yorke algorithm for fixed points or zeros of C^2 maps*, ACM TOMS, 6 (1980), pp. 252–260.

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

[29] ———, *Solving the nonlinear complementarity problem by a homotopy method*, SIAM J. Control Optim., 17 (1979), pp. 36–46.

[30] ———, *An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 394–401.

[31] J. J. MORÉ, *MINPACK Documentation*, Argonne National Lab., Argonne, IL, 1979.
[32] J. E. DENNIS, *private communication*, February, 1980.
[33] F. W. DORR, *The numerical solution of singular perturbations of boundary value problems*, SIAM J. Numer. Anal., 7 (1970), pp. 281–313.
[34] 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.
[35] R. MENZEL AND H. SCHWETLICK, *Zur Lösung parameterabhängiger nichtlinearer Gleichungen mit singulären Jacobi-Matrizen*, Numer. Math., 30 (1978), pp. 65–79.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/81/01/04-0008 $01.00/0

FITTING EMPIRICAL DATA BY POSITIVE SUMS OF EXPONENTIALS

AXEL RUHE†

Abstract. Least squares and maximum likelihood fitting of a positive sum of exponentials to an empirical data series is discussed. A characterization in terms of a convex moment cone is used, to develop a globally convergent self-starting algorithm. The sensitivity of the results to errors in the data and during the computations is also discussed. Numerical tests are reported.

Key words. exponential sum, curve fitting, nonlinear least squares, maximum likelihood, moment problem, semi-infinite programming, Gauss–Newton

1. Introduction. In this contribution, we set out to solve the problem:

Problem 1. Given a series of observations (τ_i, η_i), $i = 1, 2, \cdots, n$, find a set of positive masses $\alpha_k > 0$, $k = 1, \cdots, p$, and decay rates $\alpha \leqq \lambda_1 < \lambda_2 < \cdots < \lambda_p \leqq \beta$, such that

$$\eta_i \sim \sum_{k=1}^{p} \alpha_k \exp(-\lambda_k \tau_i).$$

This is one of the most important, difficult and frequently occurring problems of applied data analysis. Classical examples are radioactive decay [17], [22], compartmental models [4], atmospheric transfer functions [23] and all types of diffusion processes.

We consider two kinds of fitting, weighted least squares and maximum likelihood. In the latter case, it is assumed that the η_i follow Poisson distributions, and maximum likelihood corresponds to least squares weighted by the approximation (see [17] and Appendix).

In several application areas, the restriction to positive coefficients is natural; α_k can, e.g., stand for masses of elements in a mixture. Mathematically, it corresponds to the assumption that the η_i are observations of a completely monotonic function [12], and it has been shown that stronger statements on the unicity of best approximations can be made with an assumption of positive coefficients [2]. Without this assumption, the coefficients can grow beyond bounds and generate polynomials (see [19]).

We continue this contribution in § 2, by discussing a mathematical characterization of the best solution to Problem 1. It is based on a discrete Laplace transform, and was apparently first pointed out by Aigrain and Williams [1]. Its usefulness for giving a self-starting algorithm, which determines the number of terms p itself, was pointed out by Cantor and Evans [4], who developed a sublinearly convergent algorithm which is further discussed in [6] and [23]. Kammler [12] uses the Aigrain–Williams characterization to approximate completely monotonic functions, and treats also the continuous case, when the ordinary Laplace transform enters the characterization. The close relation to the moment problem (see [13]), and to dual pairs of semi-infinite programs [10] is evident. In fact, if the L_1-norm is used instead of least squares, we arrive at a linear semi-infinite program (see Gustafson [11]). Various ways of applying Fourier and Laplace transforms to this problem have been attempted by practitioners (see, e.g., [22]).

In § 3, we discuss perturbation theory. It is well known that the parameters are badly determined by the data; see, e.g., the treatment by Lanczos [14], where it is shown

* Received by the editors April 8, 1980. This work was partially supported by the Swedish Natural Science Research Council under grant F-3471 and the U.S. Public Health Service under grant HL-17731.

† Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden.

how several different exponential sums can approximate the same data series equally well. We show that whenever one of the exponentials can be well approximated by a sum of the others, the matrices occurring in our algorithm become ill-conditioned, as well as the minimum point. Though bad, this is a very straightforward kind of ill-conditioning, which can be detected by means of the closeness of the exponents and the size of the coefficients. A better-conditioned parametrization is obtained if one uses divided differences, as explained by Osborne [16] or Evans et al. [6]; we will, however, not discuss such reparametrizations further here.

In § 4, we describe the algorithm we have used. It contains two phases, one discrete when the exponents λ_k are localized to appropriate intervals, and one continuous when they are adjusted to their optimal values. The discrete phase is treated by a standard routine for nonnegative least squares [15].

In the continuous phase, we can use our knowledge of the derivatives to develop a stable Newton-like method, which converges quadratically towards a global minimum. Note that our algorithm, which is the *perturbation method* of Wedin [18], is in all essentials equivalent to successive Hermite interpolation on the transformed problem [12], or the Prony approach as described by Osborne [16].

We conclude in § 5 by relating results for a few numerical examples, both artificial and arising from practical applications.

Before entering into the details, let us make clear that the characterization we use is a mathematical one. We obtain the maximum possible number of terms p beyond which no improvement can occur for the given data. It is a surprising fact that this number is often quite low, but yet the sum may contain terms of no practical interest. We postpone a discussion of the statistical question, whether a given data series really warrants being approximated by an exponential sum of a certain length, to a later occasion. It is conceivable that techniques like cross validation (see [5], [24]) could resolve such questions.

From now on, we will use the following vector notation to describe our problem. An italic letter will denote a column vector whose elements are the corresponding Greek letter. We will thus have the vectors

$$t = \begin{pmatrix} \tau_1 \\ \tau_n \end{pmatrix}, \quad y = \begin{pmatrix} \eta_1 \\ \eta_n \end{pmatrix}, \quad a = \begin{pmatrix} \alpha_1 \\ \alpha_p \end{pmatrix}, \quad l = \begin{pmatrix} \lambda_1 \\ \lambda_p \end{pmatrix},$$

and introduce the special variable vector

$$e(\lambda) = \begin{pmatrix} \exp(-\lambda\tau_1) \\ \exp(-\lambda\tau_n) \end{pmatrix}$$

and matrix

$$E(l) = (e(\lambda_1), \cdots, e(\lambda_p)).$$

We will use (diagonally) weighted scalar products

$$(x, y)_W := y^T W x = \sum_{i=1}^{n} \eta_i \omega_i \xi_i,$$

$$W = \operatorname{diag}(\omega_i),$$

and weighted Euclidean vector norms

$$\|x\|_W := (x, x)_W^{1/2}$$

Let $\mathscr{E}$ be a closed subset of the real line from which the exponents λ_k may be taken. In Problem 1, it is the interval

$$\mathscr{E} = [\alpha, \beta],$$

but we will allow for more general sets. We denote by $\mathscr{S}_0$ the set of ordered vectors

$$\mathscr{S}_0(\mathscr{E}) := \{x | \xi_k < \xi_{k+1},\ \xi_k \in \mathscr{E}\}.$$

Now we get the following compact formulation of our problem:

$$\underset{a \geqq 0,\, l \in \mathscr{S}_0(\mathscr{E})}{\text{Minimize}} \|y - E(l)a\|_W. \tag{1.1}$$

In the weighted least squares case we have

$$W = \operatorname{diag} (\eta_i)^{-2},$$

while in maximum likelihood we use

$$W = \operatorname{diag} (\eta_i^*)^{-2},$$

where $y^* = E(l)a$ is the approximation.

For l and W fixed, we see that (1.1) is a linear least squares problem and we denote its solution by

$$a = E(l)^+_W y, \tag{1.2}$$

using the notion of a weighted pseudo-inverse.

2. Mathematical characterization. In this section we will start by showing that the set of admissible approximations form a convex cone in R^n. We then get a characterization in terms of supporting hyperplanes, and arrive at a dual formulation by performing a discrete Laplace transform. Our approximation problem is then transformed into a one-sided interpolation problem, and we find a close resemblance to the Gaussian quadrature problem.

Regard (1.1) as a geometrical problem in R^n. y is an arbitrary vector, and the approximation

$$y^* = E(l)a \tag{2.1}$$

is a positive linear combination of the p basis vectors $e(\lambda_1), \cdots, e(\lambda_p)$. y^* is thus a point in the convex cone generated by all $e(\lambda)$, $\lambda \in \mathscr{E}$. In order for y^* to be a best weighted least squares approximation, the difference $y - y^*$ must be the normal to a supporting hyperplane of the cone, that is,

$$(y - y^*, e(\lambda))_W \leqq 0, \qquad \lambda \in \mathscr{E}. \tag{2.2}$$

We see that equality occurs for $\lambda = \lambda_k$, $k = 1, \cdots, p$, since (2.2) is then a row of the normal equations to determine the α coefficients, our task is now to determine l so that (2.2) is negative in the rest of $\mathscr{E}$.

Let us study (2.2) as a function of λ and define

$$\rho(\lambda) = \eta(\lambda) - \eta^*(\lambda), \tag{2.3}$$

$$\eta(\lambda) = (y, e(\lambda))_W,$$

$$\eta^*(\lambda) = (y^*, e(\lambda))_W = (E(l)a, e(\lambda))_W = \sum \alpha_k \varphi_k(\lambda), \tag{2.4}$$

$$\varphi_k(\lambda) = (e(\lambda_k), e(\lambda))_W. \tag{2.5}$$

The ρ, η, η^* and φ_k are the *discrete Laplace transforms* of r, y, y^* and $e(\lambda_k)$, respectively. Now (2.2) means that we seek an η^* that interpolates a given η from above, and that we use basis functions φ_k taken from a family of functions depending on a continuously varying parameter λ_k.

We note that we can pair together properties of our original approximation problem and properties of the transformed problem:

Original problem	*Transformed problem*
Normal equations to determine a given l	Interpolation of η by η^* in $\lambda_1, \cdots, \lambda_p$
Cone behind supporting hyperplane	$\eta^* \geqq \eta$
Limit point of cone	Osculatory interpolation $\eta'(\lambda_k) = \eta^{*\prime}(\lambda_k)$, $k = 1, \cdots, p$

We see immediately the close relation to the moment problem [13]; in fact, we have a semi-infinite programming problem [10] with a quadratic objective function. At most $p \leqq (n+1)/2$ masses are positive; in case of equality our approximation interpolates the data, which are sampled from a completely monotonic function.

3. Perturbation theory. In this section, we will compute the first two derivatives of the least squares objective function, and see how they are expressed in terms of the discrete Laplace transform. These derivatives determine the sensitivity of the solution, l, a, to perturbations in the data, y, and we will relate this sensitivity to a well-studied mathematical approximation problem.

Derivatives of the objective function. Let us denote the (weighted) least squares objective function by

$$\varphi(x) = \tfrac{1}{2}(f(x), \qquad f(x))_W = \tfrac{1}{2} f^T W f,$$

$$x = \begin{pmatrix} a \\ l \end{pmatrix}, \tag{3.1}$$

$$f = y - y^* = y - E(l)a.$$

Expand (3.1) into a Taylor series up to second-order terms;

$$\varphi(x_0 + h) = \varphi(x_0) + h^T J^T W f + \tfrac{1}{2} h^T \left(J^T W J + \sum_{i=1}^{n} f_i w_i G_i \right) h + O(\|h\|^3), \tag{3.2}$$

where J (Jacobian) and G are matrices of first and second derivatives evaluated at $x = x_0$. In our case, they are first derivatives:

$$J = \left[\frac{\partial f_i}{\partial x_j} \right] = [-E(l), -E(l)'A],$$

$$A = \operatorname{diag}(\alpha_k), \tag{3.3}$$

$$E(l)' = (e'(\lambda_1), \cdots, e'(\lambda_p)),$$

$$e'(\lambda) = (-\tau_1 e^{-\lambda\tau_1}, \cdots, -\tau_n e^{-\lambda\tau_n})^T, \tag{3.4}$$

making the gradient,

$$J^T W f = [-(y - y^*, e(\lambda_k))_W, \quad -(y - y^*, e'(\lambda_k))_W \alpha_k]^T$$

$$k = 1, \cdots, p \qquad\qquad k = 1, \cdots, p \tag{3.5}$$

$$= [-\rho(\lambda_1). \cdots, -\rho(\lambda_p), \quad -\rho'(\lambda_1)\alpha_1, \cdots, \rho'(\lambda_p)\alpha_p]^T,$$

where ρ is the discrete Laplace transform of the residual defined in (2.3). Obviously, the gradient is zero whenever η^* interpolates η osculating, as stated in the previous section.

Now study the first term of the Hessian, and get the elements of the four parts as

$$H_1 = J^T W J = \begin{bmatrix} (e(\lambda_j), e(\lambda_k))_W & (e(\lambda_j), e'(\lambda_k))_W \alpha_k \\ (e'(\lambda_j), e(\lambda_k))_W \alpha_j & (e'(\lambda_j), e'(\lambda_k))_W \alpha_j \alpha_k \end{bmatrix}, \tag{3.6}$$

each part having indices j, k ranging from 1 to p. Now the properties of the discrete Laplace transform imply that

$$\begin{aligned} (e(\lambda_j), e(\lambda_k))_W &= \varphi_k(\lambda_j) = \varphi_j(\lambda_k), \\ (e(\lambda_j), e'(\lambda_k))_W &= \varphi_k'(\lambda_j) = \varphi_j'(\lambda_k), \\ (e'(\lambda_j), e'(\lambda_k))_W &= \varphi_k''(\lambda_j) = \varphi_j''(\lambda_k), \end{aligned}$$

in terms of the basis functions φ_k defined in (2.5) and their first two derivatives. The condition of this first part of the Hessian determines the part of the sensitivity of a and l that depends on the parametrization.

The part of the sensitivity that is independent of the parametrization is determined by the curvatures of the f surface, as explained in [18] or [20]. To get them, we study the second term of the Hessian, first by looking at the second derivatives

$$G_i = \frac{\partial f_i}{\partial x_j \, \partial x_k} = \begin{bmatrix} 0 & -\operatorname{diag}_k (e'(\lambda_k))_i \\ -\operatorname{diag}_k (e'(\lambda_k))_i & -\operatorname{diag}_k (e''(\lambda_k))_i \alpha_k \end{bmatrix},$$

with each block a $p \times p$ diagonal matrix, and derivatives of $e(\lambda)$ defined as in (3.4).

Summing up the n components, we get

$$\begin{aligned} H_2 &= \sum_{i=1}^{n} f_i w_i G_i \\ &= \begin{bmatrix} 0 & -\operatorname{diag} (y - y^*, e'(\lambda_k))_W \\ -\operatorname{diag} (y - y^*, e'(\lambda_k))_W & -\operatorname{diag} (y - y^*, e''(\lambda_k))_W \alpha_k \end{bmatrix} \\ &= \begin{bmatrix} 0 & -\operatorname{diag} \rho'(\lambda_k) \\ -\operatorname{diag} \rho'(\lambda_k) & -\operatorname{diag} \rho''(\lambda_k) \alpha_k \end{bmatrix}, \end{aligned} \tag{3.7}$$

in terms of the first and second derivatives of the transformed residual ρ defined by (2.3).

We see that at a stationary point only the lower right block is nonzero, which simplifies the analysis of curvature significantly, as well as the algorithmic development.

Curvatures of surface. At the global optimum, where η^* interpolates from above, it is true that $\rho < 0$, and thus $\rho''(\lambda_k) < 0$, making all the eigenvalues κ_j of the curvature eigenvalue problem

$$(\sum f_i w_i G_i) x = \kappa (J^T W J) x \tag{3.8}$$

nonnegative. This means that the second part of the Hessian dampens the effect of a perturbation on the parameters, and we can be content with a perturbation analysis based on the Jacobian only. On the other hand, note that at a local optimum ($p < p_{\max}$) there will be negative curvatures, and then the nonlinearity of the surface will increase the effect of perturbations.

From the expansion (3.2), we see that if we disregard the second term H_2 of the Hessian, a perturbation δy to y causes a first-order perturbation

$$h = (J^T W J)^{-1} J^T W \delta y$$

to x. In the nonweighted case

$$h = J^+ \delta y,$$

with the first factor only depending on the exponents $\lambda_1, \cdots, \lambda_p$ and the second only on size of h.

Condition of Jacobian. From (3.3), we see that J can be factored

$$J = -[E(l) \quad E(l)'] \begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix},$$

with the first factor only depending on the exponents $\lambda_1, \cdots, \lambda_p$, and the second only on the coefficients $\alpha_1, \cdots, \alpha_p$.

The dependence on α_k is simple enough, small coefficients cause a nearly singular J, those λ_k which correspond to small α_k being exceptionally sensitive.

To see the λ_k influence, we estimate the singular values of the first factor. Let us first study the first half of it, since it is of independent interest in that it determines the sensitivity of a, with l kept fixed.

Condition when decay rates l are fixed. The smallest singular value of $E(l)$ is given by

(3.9)
$$\sigma_p(E) = \min_{\|a\|_2 = 1} \|Ea\|_2.$$

A closer look at the vector $p = Ea$ reveals that it has the elements

$$\pi_i = \sum \alpha_k e^{-\lambda_k \tau_i} = \pi(\tau_i),$$

where π is an exponential polynomial with normalized coefficients. The better we can approximate 0 by such an exponential polynomial, the smaller $\sigma_p(E)$ becomes and the worse-conditioned our approximation problem is. We do not intend to do a complete study of the approximation properties of exponential polynomials, but will limit our discussion to a simple case which we believe to be illustrative.

Assume that the exponents λ_k are equidistant, starting at zero,

(3.10)
$$\lambda_k = (k-1)\,\delta.$$

Then $\pi(\tau)$ will become an ordinary polynomial

(3.11)
$$\pi(\tau) = \sum \alpha_k e^{-\lambda_k \tau} = \sum \alpha_k \xi^{k-1},$$

(3.12)
$$\xi = e^{-\delta \tau_n},$$

where $\xi \in [e^{-\delta \tau_n}, e^{-\delta \tau_1}] = [\gamma, 1] \subset (0, 1]$.

The lower bound $\gamma = e^{-\delta \tau_n}$, determined by the length of the sampling interval τ_n and the spacing between the exponents δ, will be the critical quantity here.

The minimal property of the Chebyshev polynomials implies that,

(3.13)
$$\min_{\|a\|_1 = 1} \|Ea\|_\infty \leqq \frac{1}{T_{p-1}(3+\gamma)/(1-\gamma)}$$

and so by (3.9) and inequalities between different norms,

$$\sigma_p(E) \leqq \frac{\sqrt{np}}{T_{p-1}(3+\gamma)/(1-\gamma)}. \tag{3.14}$$

Since the Chebyshov polynomials grow fast outside $[-1, 1]$, we see that $\sigma_p(E)$ must be small indeed when p is large or $\delta\tau_n$ small. Note that this value is largely independent of the spacing of the sampling points τ_i; an unhappy choice may make it even smaller. The only way to obtain an improvement is to increase the length of the sampling interval τ_n.

We have computed $\sigma_p(E)$ (3.9) for some representative cases and compared to the bound (3.14), and list the result in Table 3.1. Note that (3.14) is not too far away from (3.9) in the equidistant case, while cases where only few λ_k are close are considerably better-conditioned.

TABLE 3.1.
Singular values of Jacobian matrix. Computed values compared to theoretical estimates.

Separation $\delta\tau_n$	Number of terms p	Condition for fixed λ_k computed $\sigma_1(E)$	computed $\sigma_p(E)$	estimated (3.14)	Condition for varying λ_k computed $\sigma_1(EE')$	computed $\sigma_{2p}(EE')$	estimated (3.15)
0.1	2	6.3	$9.56E^{-2}$	$1.58E^{-1}$	7.1	$5.7E^{-6}$	$2.53E^{-4}$
	3	7.6	$1.45E^{-3}$	$2.36E^{-3}$	8.5	$2.5E^{-7}$	$1.21E^{-8}$
	4	8.5	$2.04E^{-5}$	$3.32E^{-5}$	9.5	0	$5.46E^{-13}$
	5	9.4	$4.67E^{-7}$	$4.52E^{-7}$	10.0	0	$2.39E^{-17}$
0.5	2	5.8	$4.28E^{-1}$	$7.07E^{-1}$	6:5	$6.2E^{-4}$	$6.31E^{-3}$
	3	6.6	$2.91E^{-2}$	$4.75E^{-2}$	7.2	$1.2E^{-6}$	$7.53E^{-6}$
	4	7.0	$1.84E^{-3}$	$3.00E^{-3}$	7.6	0	$8.47E^{-9}$
	5	7.4	$1.12E^{-4}$	$1.84E^{-4}$	7.9	0	$9.22E^{-12}$
1.0	2	5.4	$7.34E^{-1}$	1.22	6.0	$4.1E^{-3}$	$2.51E^{-2}$
	3	5.9	$8.63E^{-2}$	$1.42E^{-1}$	6.4	$2.0E^{-5}$	$1.19E^{-4}$
	4	6.1	$9.42E^{-3}$	$1.56E^{-2}$	6.6	$1.1E^{-7}$	$5.30E^{-7}$
	5	6.3	$9.87E^{-4}$	$1.65E^{-3}$	6.7	0	$2.29E^{-9}$
10.0	2	4.6	1.12	1.12	2.16	5.2	$3.5E^{-2}$
	3	4.6	$2.32E^{-1}$	$4.67E^{-1}$	5.2	$1.3E^{-3}$	$3.61E^{-1}$
	4	4.6	$3.83E^{-2}$	$9.26E^{-2}$	5.2	$2.4E^{-5}$	$8.88E^{-2}$
	5	4.7	$4.85E^{-3}$	$1.78E^{-2}$	5.2	$2.5E^{-7}$	$2.12E^{-2}$

Condition, varying decay rates l. Let us now turn our attention to the entire Jacobian, taking care also of the sensitivity of the exponents λ_k. We now seek an estimate of

$$\sigma_{2p}([E(l)\, E(l)']) = \min_{\|x\|_2=1} \|(EE')x\|_2.$$

We make the same assumptions (3.10), (3.11) on λ_k and τ_i, note that

$$e'(\lambda_k)_i = -\tau_i e(\lambda_k)_i$$

holds for the ith components and see that each element of $p = (EE')x$ is now

$$\pi_i = \sum \alpha_k e^{-\lambda_k \tau_i} - \tau_i \sum \beta_k e^{-\lambda_k \tau_i},$$

approximating $f(\tau) \equiv \tau$ by a quotient of two exponential polynomials. With the change of variables (3.12), we note that

$$\tau = \frac{1}{\delta} \ln\left(\frac{1}{\xi}\right),$$

making

$$\pi = \pi_1(\xi) - \frac{1}{\delta} \ln\left(\frac{1}{\xi}\right) \pi_2(\xi),$$

with π_1 and π_2 two ordinary polynomials of degree $p-1$, normalized in the 2-norm. The smaller $\|p\|$ is, the better we can approximate the logarithm by a rational function on the interval $[\gamma, 1]$, $\gamma = e^{-\tau_n \delta}$.

It has been shown (see [7]) that

$$0 < k_1 \leqq r_n(f, I) \exp\left(\frac{2n}{C(I)}\right) \leqq k_2, \tag{3.15}$$

where

$$r_n(f, I) = \inf_{r(x)} \sup_{x \in I} \|f(x) - r(x)\|,$$

$r(x)$ is a rational function of degree n, $C(I)$ is the Green capacity of set I and region where f is not analytic, and k_1, k_2 are suitable constants. In our case, $f(x) = \ln(x)$, $n = p-1$, $I = [\gamma, 1]$, and [9] gives

$$C(I) = \frac{k(1-\gamma)}{\pi k(\gamma)},$$

$$k(m) := \int_0^1 [(1-t^2)(1-mt^2)]^{-1/2}\, dt, \quad \text{a complete elliptic integral.}$$

For a short sampling interval, it is true that

$$C(I) \to \frac{1}{\ln\left(\frac{1}{1-\gamma}\right)}, \qquad \gamma \to 1,$$

so

$$\exp\left(\frac{2n}{C(I)}\right) \to (1-\gamma)^{-2n}.$$

We will thus get the estimate

$$\sigma_{2p}([E(l)E(l)']) \simeq \sqrt{np}\,(1-\gamma)^{2(p-1)}.$$

When γ approaches 1, this gets small much faster than the bound (3.14) for the first half. The sensitivity of the nonlinear problem of determining a, l for p terms is as large as that for determining a only for $2p$ terms. Some sample values of the estimate (3.15) ($k = 1$) compared to actual values of $\sigma_{2p}([EE'])$ are given in Table 3.1.

4. Algorithmic approach. In this section, we will describe the algorithm we have used and discuss its efficiency. Precisely like other semi-infinite programming

algorithms, it consists of three phases; we will start by stating the complete algorithm, and continue by discussing the three phases separately.

Algorithm.

1. Determine number of terms p, and starting value of l by discretization of the set $\mathscr{E}$.
2. Determine the optimal l that gives osculatory interpolation.
3. Verify that residual $\rho \leqq 0$ in $\mathscr{E}$. If unsuccessful, start phase 1 with a finer discretization, or make an error exit.

The main work in the first phase is done by a nonnegative linear least squares solver, and rounded up by pairing together some of the grid points.

Algorithm 1.

1. Form a grid over $\mathscr{E}$:

$$\lambda_1 < \lambda_2 < \cdots < \lambda_m, \qquad \lambda_j \in \mathscr{E},$$
$$l = (\lambda_1, \cdots, \lambda_m)^T.$$

2. Solve the linear problem, with $W = \operatorname{diag}(\eta_i)^{-2}$ in the weighted case,

$$\min_{a \geqq 0} \|y - E(l)a\|_W.$$

At most n of the α_j will be >0.

3. Condense clusters of exponents,

$$\lambda_k^{(0)} := \frac{\sum_{j=r_k}^{s_k} \alpha_j \lambda_j}{\sum_{j=r_k}^{s_k} \alpha_j},$$

where r_k and s_k are defined by

$$\alpha_{r_k-1} = 0, \quad \alpha_{r_k} \alpha_{r_k+1} \cdots \alpha_{s_k} > 0, \quad \alpha_{s_k+1} = 0.$$

The vector

$$l^{(0)} = (\lambda_1^{(0)}, \cdots, \lambda_p^{(0)})$$

will have $p \leqq n$.

The grid may be equidistant or not. Ideally it should have about three points for each final exponent, and thus be finer where the exponents are expected to be clustered, most often in the lower end of the set $\mathscr{E}$. Evans et al. [4], [6], make successive refinements of the grid l in interesting parts of the interval, until the λ_k have been bracketed to sufficient accuracy. Too fine a grid will cause excessive demand on computer time and storage.

For the linear problem, we used the standard approach described in the text book by Lawson and Hanson [15], algorithm NNLS (nonnegative least squares). Upon exit, it makes sure that (2.3)

$$\rho(\lambda_j) \leqq 0, \quad \lambda_j \in G,$$

with equality for at most n j's, where $\lambda_j > 0$. Generally, the interior points occur in pairs; single points and larger clusters are indications that the grid is too coarse. Some plots of the transformed residual $\rho(\lambda)$ are given in Figs. 5.2, 5.4, 5.5 and 5.6 below.

In the normal case, we will get

$$p \leqq \frac{(n+1)}{2},$$

and most often a much smaller p (see the examples in § 5).

The second phase is started at $l^{(0)}$, and gets a starting $a^{(0)}$ by solving a linear least squares problem. We have used both a least squares and a maximum likelihood criterion in phase 2, while phase 1 can only be performed for least squares. However, we assume that the data are at least as good as allowing our ML algorithm to start at a weighted least squares solution. In the least squares case, the problem is linear in a, making a separable approach usable [21], [8], while in ML it is not. We choose to describe the ML variant; the least squares one differs only in trivial details.

Since we expect phase 1 to have delivered a reasonably good starting approximation and have first and second derivatives readily available (3.5), (3.6), (3.7), we could conveniently use the Newton method. However, a close look at the second derivatives (3.7) shows a better alternative. Our concern is to keep the iteration matrix close enough to the Hessian, but still positive definite. The (1, 2) and (2, 1) blocks of H_2 (3.7) will shift the eigenvalues downwards and are zero on convergence, so removing them from the iteration matrix makes it more positive definite. The (2, 2) block will be nonnegative at the global optimum, but can contain large negative elements when we are on the way. Replacing the diagonal elements by their absolute values, we obtain three objectives. First, we can solve each step as an augmented least squares problem, second, we get a Marquardt-type stabilization and third, we get quadratic convergence when approaching the global minimum. It might be of interest to note that removing the (1, 2) and (2, 1) blocks corresponds to the perturbation method, introduced by Wedin [18] as the parametrization independent variant of Newton's method but never tried before in practice, while the other approach of retaining (1, 2) and (2, 1) but removing the second derivatives in (2, 2) was described in [21] but neither tried nor recommended.

Let us now give the algorithmic formulation of phase 2.

Algorithm 2. (Compute a maximum likelihood solution a, l starting at $l^{(0)}$).

1. Start with $l = l^0$ from phase 1, and compute

$$a := E(l)^{+}_{W} y, \quad y^s := E(l)a, \quad x^0 := (a, l)^T.$$

2. *For* $s = 0, 1, \cdots$ *until* convergence:

2.1. *Compute* a) $f := e - y/y^s$ {objective function/elementwise division}.

b) $J := \operatorname{diag}(y/(y^{(s)2}))[E(l)\ E(l)A]$ {Jacobian, see (3.3)}.

c) $G := \operatorname{diag}[(f \times y/(y^{(s)2}))^T E(l)''A]$ {part of Hessian, see (3.7)}.

2.2. *Solve linear least squares problem*

$$\min_h \left\| \begin{bmatrix} J \\ |G|^{1/2} \end{bmatrix} h - \begin{bmatrix} f \\ 0 \end{bmatrix} \right\| \quad \{\text{step h}\}.$$

2.3. *If* $\|Jh\| <$ tolerance *then* convergence.

2.4. *Update* $x^{s+1} := x^s + \mu_s h$ {line search to determine μ_s}.

3. Now $x = (a, l)$ is the solution;

$y^* = y^s$ is the best approximation.

We have chosen not to clutter up the description with details on the constraints. A coefficient α_k which hits zero is a signal of failure, and causes restart of phase 1. An exponent λ_k which is at a limit point of the set $\mathscr{E}$ (see (1.1)) is kept fixed, and the corresponding column of the Jacobian is removed.

The task of the third phase is to verify that the transformed residual (2.3) is nonpositive throughout $\mathscr{E}$. We have simply discretized $\mathscr{E}$ with a grid and tested:

Algorithm 3. (Verify that $\rho(\lambda) \leqq 0$ on $\mathscr{E}$).

1. Form grid $l = (\lambda_0, \cdots, \lambda_j, \cdots, \lambda_m)$.
2. *If* $(y - y^*, e(\lambda_j))_W \leqq 0$ *then* success.

It is a nontrivial task to determine that the grid is fine enough; we refer the reader to Gustafson [10].

5. Numerical examples. We have implemented the algorithm described here as an APL program which we have run on an IBM 5100 computer at the University of California, San Diego and a CDC Cyber 172 computer at the Umeå University Computer Center.

In phase 1, we used a translation of the routine NNLS from [15]. We used both equidistant spacing and a grid with

$$\lambda_j = \alpha + (\beta - \alpha)\, e^{-(m-j)/d}, \qquad j = 1, \cdots, m,$$

starting with $m = 10$, $d = 1$, and continuing if necessary with $m = 20, 40, \cdots, d = 2, 4, \cdots$.

This logarithmically spaced grid is in many cases the most natural one to choose, since the exponents often get clustered around zero. The program for phase 2 was run with tolerance set at 10^{-7} in Algorithm 2 of § 4.

We will report results here from runs on a few data series, all listed in Table 5.1. Two of them are artificial and six empirical.

We start with listing results for the classic data series given by Lanczos [14, p. 276]. The first run was with equal weights $\omega = 1$, which should correspond to the original setup of Lanczos. When we tested the interval $(\alpha, \beta) = (0, 10)$, a grid of $m = 10$ was too coarse to get started on phase 2. A grid of 20 points localized the two pairs of λ values (0.82, 1.35) and (3.68, 6.06). Phase 2 needed 7 steps to converge to seven decimals of accuracy towards

$$(\lambda_1, \lambda_2) = (1.808852, 4.571956).$$

Though the residual is

$$\|y - y^*\| = 1.07 \cdot 10^{-2},$$

the test of phase 3 indicated that this was a local maximum. (Lanczos gets $l = (1.58, 4.45)$ with Prony's method.) The algorithm restarts with $m = 40$ and gets the new intervals (1.35, 1.74), (3.68, 4.72) and the endpoint $\lambda = 10$. Four steps of phase 2 converge towards

$$l = (1.547288\ 4.35606, 10),$$

and weights

$$a = (0.735203, 3.6596791, 0.067805);$$

now the residual is slightly smaller,

$$\|y - y^*\|_2 = 1.03 \cdot 10^{-2},$$

and phase 3 indicates successful termination.

The second run on the Lanczos data used the weights $\omega_i = \eta_i^{-2}$, which is the set most often used for a weighted least squares aproach. Then the endpoint $\lambda = 10$ did not show up, and we got the same starting intervals from phase 1 as before. The final exponents,

$$l = (1.75656, 4.54746),$$

were reached after 3 iterations in phase 2, this time with a program using c-g acceleration [20] instead of the second derivatives as described in § 4.

TABLE 5.1.
Data series for which results are reported.

LANCZOS		BOLIDEN 1		BOLIDEN 3	
0	2.51	0	1	0	1
0.05	2.04	1	0.1543	1	0.1808
0.1	1.67	2.5	0.075	3	0.09973
0.15	1.37	5	0.05417	5	0.07222
0.2	1.12	10	0.03528	10	0.0486
0.25	0.93	15	0.02861	15	0.03611
0.3	0.77	20	0.02472	20	0.02861
0.35	0.64	25	0.02333	25	0.02472
0.4	0.53	30	0.0225	30	0.02111
0.45	0.45	40	0.02122	40	0.01806
0.5	0.38				
0.55	0.32	BOLIDEN 2		BOLIDEN 4	
0.6	0.27				
0.65	0.23	0	1	0	1
0.7	0.2	2.25	0.1087	1	0.1278
0.75	0.17	5	0.056	3	0.05361
0.8	0.15	10	0.03397	5	0.03667
0.85	0.13	15	0.02717	10	0.0208
0.9	0.11	20	0.0216	15	0.01486
0.95	0.10	25	0.01902	20	0.01139
1	0.09	30	0.01821	25	0.01014
1.05	0.08	40	0.01712	30	0.009167
1.1	0.07			40	0.008
1.15	0.06				

EVANS GV		ASTRA 2		ASTRA 3	
0	1	0	29.655	0	22.583
1	0.9399	0.033	45.716	0.03	37.34
2	0.8853	0.083	40.323	0.08	33.961
3	0.8356	0.17	28.868	0.17	25.534
4	0.7904	0.25	24.643	0.25	24.013
5	0.7492	0.333	20.05	0.33	17.405
10	0.5921	0.5	14.81	0.5	13.934
30	0.3518	0.75	10.869	0.75	9.039
60	0.2655	1	7.674	1	6.933
150	0.1655	1.5	4.136	1.5	4.176
300	0.112	2	2.776	2	2.947
400	0.1016	2.5	1.587	2.5	1.886
500	0.09714	3	1.135	3	1.37
1,000	0.0905	4	0.511	4	0.691
1,500	0.08607	5	0.294	5	0.412
2,000	0.08187	6	0.2	6	0.24
3,000	0.07408				
4,000	0.06703				
5,000	0.06065				
6,000	0.05488				

The third run computed the maximum likelihood ($w = E(l)a$) estimate of the exponents as

$$l = (1.74990, 4.54551),$$

after 5 iterations with the algorithm described in § 4.

We plot the Lanczos data together with its LS approximation in Fig. 5.1. The lower curve represents the first term of the sum. We show also the transformed residual (2.2) in Fig. 5.2, first after phase 1 where we see that pairing of points for successive refinements of the grid, and then after phase 2 where we see the osculating interpolation phenomenon.

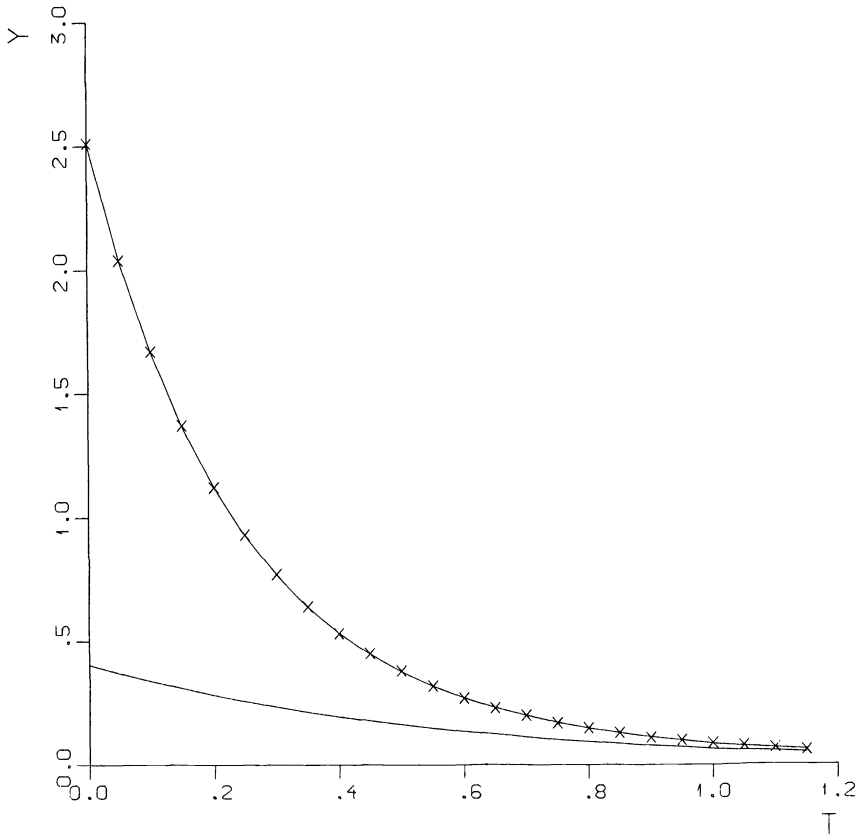


FIG. 5.1. *Data points y, marked as crosses, and approximation y*, drawn as curve, for Lanczos data series.*

Another artificial data series, given by Evans et al. [6], has a rather different appearance, because it has three widely different decay rates. Four iterations of phase 2 produced the ML estimate

$$l = (1{\cdot}0000 \cdot 10^{-4}, 9.9914 \cdot 10^{-3}, 9.9954 \cdot 10^{-2}),$$

$$a = (0.099999, 0.29981, 0.60017)$$

for the exponents and coefficients, which agrees with the function that generated the data to 3 decimals. The transformed residual is now, however, not negative, so the approximation is not optimal. Adding one term corresponding to the right endpoint of the λ interval gives the optimal approximation, differing from this one only in its 5th decimal.

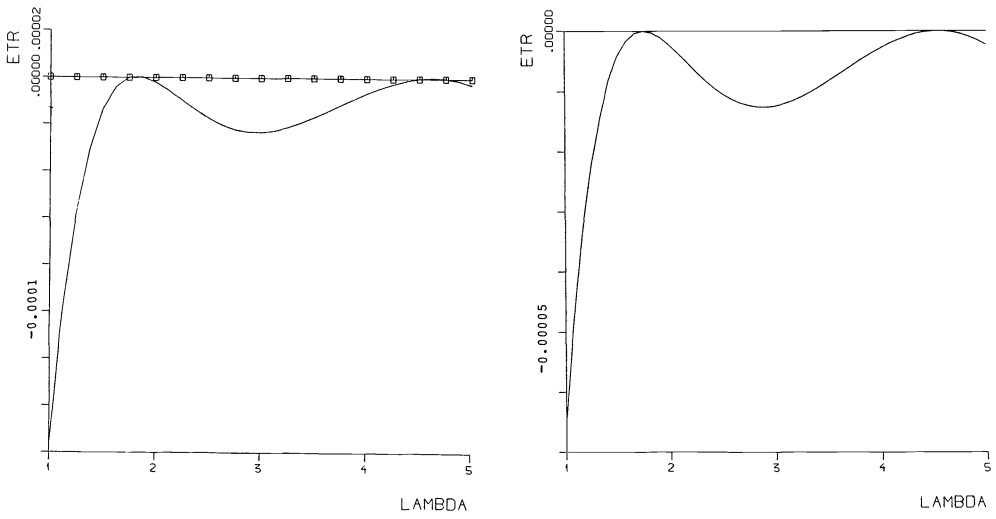


FIG. 5.2. *Transformed residual $\rho(\lambda)$ for Lanczos data.* a) *After discrete phase, weighted least squares. Grid points marked by squares. Interpolation in grid points λ_k with positive coefficients α_k.* b) *After continuous phase, maximum likelihood. Osculatory interpolation in abscissae λ_k.*

The first empirical data series describes retention of penicillin in the blood, and was supplied by the Astra drug company. A compartment model approach leads to a sum of exponentials. Now the model is not entirely adequate, which is indicated by the fact that the first sample, $\eta_1 < \eta_2$, corresponds to the increase in concentration just after administration. This should be represented by a negative term with a fast rate of decay.

The results for the two series ASTRA 2 and ASTRA 3 are summarized in Table 5.2 We see that, although the test on the transformed residual indicated that more terms

TABLE 5.2.
Summary of results for empirical data series.

Series	p	residual	exponents l		
ASTRA 2	1	1.43	0.90517		
	2	0.47	0.59165	2.0986	
	3	0.39	0	0.94446	3.2732
ASTRA 3	1	1.27	0.83700		
	2	0.44	0.59734	2.4285	
	3	0.44	0	0.70868	2.7516
BOLIDEN 1	4	0.019	0.035252	0.15634	0.33062	2.6749
2	4	0.045	0	0.12055	0.70273	8.2892
	3	0.025	0.00548	0.10219	0.72566	10.843
4	4	0.028	0.00961	0.1627	0.72194	3.392

are needed, no significant drop in the residuals occurs after $p = 2$, in spite of the fact that the exponents are changed quite a bit. Some selected plots are given in Fig. 5.3. The transformed residual indicated that the global optimum is not yet obtained, probably depending on the inadequacy of the model. The weird fact is that phase 1 gives no indication of the need for more terms. This phenomenon certainly merits further study.

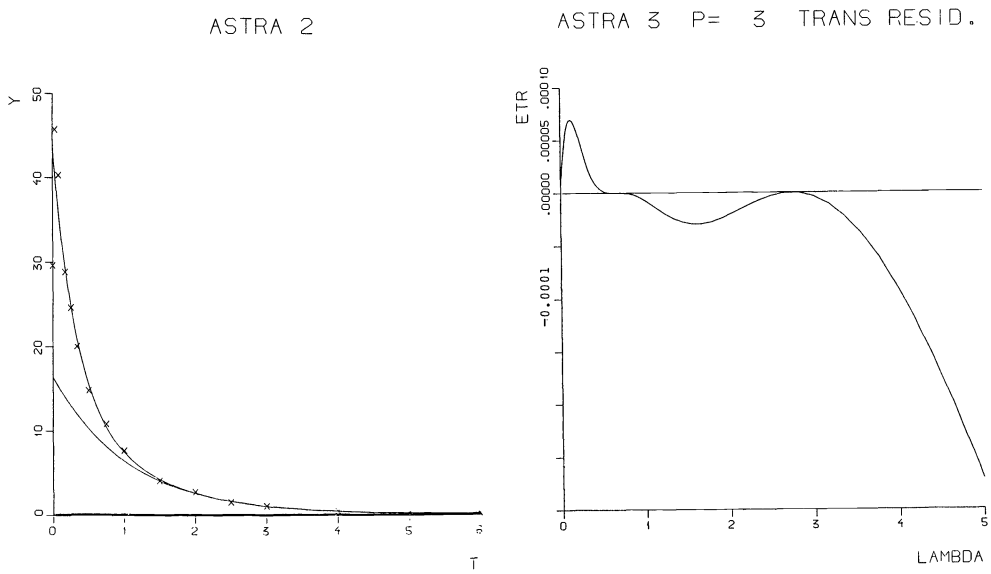


FIG. 5.3. *Selected plots of Astra data series.*

The second set of empirical data describes a batch flotation process and was supplied by the Boliden mining company. Results are given in Table 5.2 and a selected plot in Fig. 5.4. It is evident that these data behave in quite a different way from the Astra series. Now the first point η_1 is large, forcing a fast decaying term the exact position of which is virtually undetermined. A notable fact also is that four terms are needed, forcing the function to nearly interpolate the data. One might speculate over the nature of the errors.

The results of the empirical data indicate that a study of suboptimal approximations is certainly of interest, but then certain precautions have to be inserted into the algorithm to avoid convergence to local minima.

Appendix. Derivation of the maximum likelihood estimates for Poisson distribution.

Observed count rates for radiation quanta, such as X-rays, photons or neutrons, are theoretically distributed as Poisson distributions (see [17])

$$p(y) = \frac{(\exp(-y_0)y_0^y)}{y!},$$

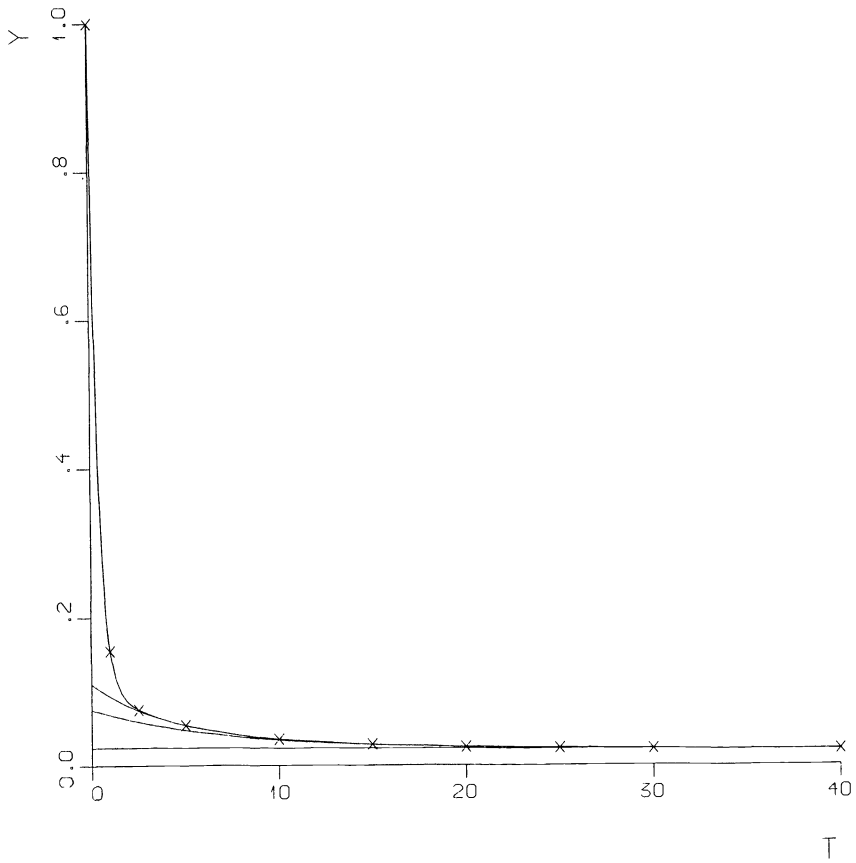


FIG. 5.4. *Selected plot of Boliden data series.*

Suppose y is described by the model

$$y = y_0 + \varepsilon, \qquad \varepsilon \text{ errors},$$

$$y_0 = f_0(\theta), \qquad \theta \text{ parameters } (a, l \text{ in our case});$$

we get the log likelihood function for a series of observations $y = (y_1, \cdots, y_k)$ as

$$\log L(y, \theta) = \sum_k [-f_k(\theta) + y_k \log f_k(\theta) - \log y_k!],$$

(f_k is the model function at time t_k).

The derivative is now

$$\frac{\partial \log L(y, \theta)}{\partial \theta_i} = \sum_k \; -\frac{\partial f_k(\theta)}{\partial \theta_i} + y_k \frac{\dfrac{\partial f_k(\theta)}{\partial \theta_i}}{f_k(\theta)}$$

$$= \sum_k \frac{1}{f_k(\theta)} \{y_k - f_k(\theta)\} \frac{\partial f_k(\theta)}{\partial \theta_i}.$$

Equating this to zero, we get $\hat{y}_{ML} = f(\hat{\theta}_{ML})$ satisfying the equations

$$\sum_k \frac{1}{f_k(\theta)} \left[(y_k - f_k(\theta)) \frac{\partial f_k(\theta)}{\partial \theta_i} \right]_{\theta = \hat{\theta}_{ML}} = 0, \qquad i = 1, \cdots, p.$$

These equations are conditions for a minimum of

$$\|y - f(\theta)\|_W = \left(\sum_k [f_k(\theta)^{-1}(y - f_k(\theta))]^2\right)^{1/2},$$

$$W = \text{diag}\,(f_k(\theta)^{-2}).$$

Noting that $y^* = f(\hat{\theta}_{\text{ML}})$, we get the ML weighting discussed in the paper.

Acknowledgments. This work was started while the author was visiting the University of California, San Diego, and he owes much of the style of the presentation to discussions with W. B. Gragg and John Evans. He has also had clarifying discussions on mathematical and statistical aspects with Johan Karlsson and Jacques de Maré.

REFERENCES

[1] P. R. AIGRAIN AND E. M. WILLIAMS, *Synthesis of n-reactance networks for desired transient response*, J. Appl. Phys., 20 (1949), pp. 597–600.

[2] D. BRAESS, *Approximation mit Exponentialsummen*, Computing, 2 (1967), pp. 309–321.

[3] D. BRAESS, *Die Konstruktion der Tschebyscheff Approximierenden bei der Anpassung mit Exponentialsummen*, J. Approx. Theory, 3 (1970), pp. 261–273

[4] D. G. CANTOR AND J. W. EVANS, *On approximation by positive sums of powers*, SIAM J. Appl. Math., 18 (1970), pp. 380–388.

[5] B. EFRON, *Computers and the theory of statistics: thinking the unthinkable*, SIAM Rev., 21 (1979), pp. 460–480.

[6] J. W. EVANS, W. B. GRAGG AND R. J. LEVEQUE, *On least squares exponential sum approximation with positive coefficients*, Math. Comp., 34 (1980), pp. 203–212.

[7] T. GANELIUS, *Orthogonal polynomials and rational approximation of holomorphic functions*, Paul Turan Memorial Volume, to appear.

[8] G. H. GOLUB AND R. J. LEVEQUE, *Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems*, Tech. Rep. SU 236 P 30–60, Computer Science Dept., Stanford, Univ., Stanford CA, 1979.

[9] A. A. GONČAR, *Estimates of the growth of rational functions and some of their applications*, Math. USSR-Sb, 1 (1967), pp. 445–456.

[10] S.-Å. GUSTAFSON AND K. KORTANEK, *Numerical treatment of a class of semi-infinite programming problems*, Naval Res. Logist. Quart., 20 (1973), pp. 477–504.

[11] S.-Å. GUSTAFSON, *Lectures on semi-infinite programming*, Vorlr SBF72 No. 1, Inst. Angew Math., Bonn, 1979.

[12] D. W. KAMMLER, *Least squares approximation of completely monotonic functions by sums of exponentials*, SIAM J. Numer. Anal., 16 (1979), pp. 801–818.

[13] S. KARLIN AND W. J. STUDDEN, *Tchebycheff Systems with Applications in Analysis and Statistics*, John Wiley, New York, 1966.

[14] C. LANCZOS, *Applied Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1956.

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

[16] M. R. OSBORNE, *Some special nonlinear least squares problems*, SIAM J. Numer. Anal., 12 (1975), pp. 571–592.

[17] P. F. PRICE, *A comparison of the least squares and maximum likelihood estimators for counts of radiation quanta which follow a Poisson distribution*, Acta. Cryst. Sect. A, 35 (1979), pp. 57–60.

[18] H. RAMSIN AND P. Å. WEDIN, *A comparison of some algorithms for the nonlinear least squares problem*, BIT, 17 (1977), pp. 72–90.

[19] J. R. RICE, *The approximation of Functions Vol. 2: Nonlinear and Multivariate Theory*, Addison-Wesley, Reading, MA, 1969.

[20] A. RUHE, *Accelerated Gauss Newton algorithms for nonlinear least squares problems*, BIT, 19 (1979), pp. 356–367.

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

[22] M. R. SMITH, S. COHN-SFETCU AND H. A. BUCKMASTER, *Decomposition of multicomponent exponential decays by spectral analytic techniques*, Technometrics, 18 (1976), pp. 467–482.

[23] W. J. WISCOMBE AND J. W. EVANS, *Exponential sum fitting of radiative transmission functions*, J. Comp. Phys., 24 (1977), pp. 416–444.

[24] G. H. GOLUB, M. HEATH AND G. WAHBA, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.

SIAM J. SCI. STAT, COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/81/01/04-0009 $01.00/0

EQUIDISTRIBUTING MESHES WITH CONSTRAINTS*

J. KAUTSKY† AND N. K. NICHOLS‡

Abstract. Adaptive methods which "equidistribute" a given positive weight function are now used fairly widely for selecting discrete meshes. The disadvantage of such schemes is that the resulting mesh may not be smoothly varying. In this paper a technique is developed for equidistributing a function subject to constraints on the ratios of *adjacent* steps in the mesh. Given a weight function $f \geqq 0$ on an interval $[a, b]$ and constants c and K, the method produces a mesh with points $x_0 = a$, $x_{j+1} \doteq x_j + h_j$, $j = 0, 1, \cdots, n-1$ and $x_n = b$ such that

$$\int_{x_j}^{x_{j+1}} f \leqq c \quad \text{and} \quad \frac{1}{K} \leqq \frac{h_{j+1}}{h_j} \leqq K \quad \text{for } j = 0, 1, \cdots, n-1.$$

A theoretical analysis of the procedure is presented, and numerical algorithms for implementing the method are given. Examples show that the procedure is effective in practice. Other types of constraints on equidistributing meshes are also discussed.

The principal application of the procedure is to the solution of boundary value problems, where the weight function is generally some error indicator, and accuracy and convergence properties may depend on the smoothness of the mesh. Other practical applications include the regrading of statistical data.

Key words. mesh selection, equidistributing mesh, ordinary differential equation, boundary value problem, quasi-uniform and locally quasi-uniform mesh, adaptive method, smoothing constraint, optimal mesh

1. Introduction.

A number of adaptive mesh selection procedures which use equidistribution techniques have recently been developed for solving boundary value problems, (Ablow and Schechter [1], de Boor [3], Denny and Landis [4], Lentini and Pereyra [8], Pearson [9], Pereyra and Sewell [10], Russell and Christiansen [11], White [12]). The objective of such procedures is: given a positive weight function defined on an interval, find a partitioning of the interval such that the integral of the weight function takes a given constant value over each subinterval; that is, such that the weight function is "equidistributed" over the chosen mesh. Various weight functions have been used, such as (i) a measure of the arc length of the solution to the boundary value problem, (ii) an estimate of the truncation error in the discretization of the problem, (iii) an approximation to the error between the discrete and the true solution of the problem, and (iv) various transformations of the independent variable of the problem.

In many practical cases, it is necessary also for the chosen mesh to have certain smoothness properties, and the mesh is therefore required to satisfy certain additional constraints. The two most common constraints are:

(1) The ratio of the maximal to the minimal step in the mesh must be bounded above by a given constant; i.e., the mesh must be *quasi-uniform.*

(2) The ratios of adjacent steps in the mesh must be bounded by given constants; i.e., the mesh must be *locally bounded.*

The problem of equidistributing a function subject to the first constraint has been completely solved (Pereyra and Sewell [10]). However, for many applications constraint (1) is often either too restrictive (for a small constant) or too weak (for a large constant), and then constraint (2) is the more important one to satisfy. The purpose of this paper is to give a theory for equidistributing meshes subject to the second constraint, and to present a numerical algorithm for the explicit construction of such meshes.

* Received by the editors February 8, 1980, and in revised form September 24, 1980. This research was supported in part by the U.S. Department of Energy under contract DE-AS03-76SF00326 PA # 30.

† Flinders University, Bedford Park, South Australia 5042.

‡ Department of Mathematics, University of Reading, Reading, England RG6-2AH.

In § 2 we introduce general notation and the theory for quasi-uniform meshes. In § 3 we extend these results and develop a new theory for locally bounded meshes. In §§ 4 and 5 the numerical procedure is described and applications to several examples are presented, and in § 6 some asymptotic results are discussed. The paper concludes in § 7 with some remarks on further generalizations of the theory.

2. Notation and quasi-uniform meshes. In this section we introduce the necessary notation and terminology. The known results for quasi-uniform meshes are reformulated and an example is given to show that the standard solution need not be minimal (i.e., the one with the least number of steps).

DEFINITION 1. Let[1] $f \in C_+$, the set of nonnegative piecewise continuous functions on $[a, b]$, and $c > 0$ be a constant such that $n = (1/c)\int_a^b f$ is an integer. We say that the mesh

$$\pi: a = x_0 < x_1 < \cdots < x_n = b$$

is *equidistributing* (e.d.) on $[a, b]$ with respect to f and c if

$$\int_{x_j}^{x_{j+1}} f = c \quad \text{for } j = 0, 1, \cdots, n-1.$$

DEFINITION 2. With the same notation as in Definition 1, a mesh π is called *sub-equidistributing* (s.e.d.) with respect to f and c on $[a, b]$ if, for $nc \geqq \int_a^b f$,

$$\int_{x_j}^{x_{j+1}} f \leqq c, \qquad j = 0, 1, \cdots, n-1$$

is satisfied.

Clearly an e.d. mesh is s.e.d., and an e.d. mesh exists only if $\int_a^b f$ is an integer multiple of c. The e.d. mesh is then the unique mesh which is s.e.d. with respect to f and c and is *minimal*. If $(n-1)c < \int_a^b f < nc$, for some integer n, then no e.d. mesh exists, and there are an infinite number of minimal s.e.d. meshes with exactly n steps.

We now have two obvious observations.

LEMMA 1. *If $f_1 \geqq f_2$ on $[a, b]$ and π is* s.e.d. *with respect to f_1 and c, then π is* s.e.d. *with respect to f_2 and c.*

LEMMA 2. *If there are constants m and M such that $0 < m < f < M$ on $[a, b]$ and if π is* e.d. *with respect to f and some c, then*

$$\frac{\max\limits_{j=0,1,\cdots,n-1} \{h_j\}}{\min\limits_{j=0,1,\cdots,n-1} \{h_j\}} \leqq \frac{M}{m},$$

where

$$h_j = x_{j+1} - x_j, \qquad j = 0, 1, \cdots, n-1.$$

Proof. Note that $\max_j h_j \leqq c/m$ and $\min_j h_j \geqq c/M$. The result follows directly.

DEFINITION 3. We call mesh π *quasi-uniform* with respect to constant K if

$$\frac{\max\limits_j h_j}{\min\limits_j h_j} \leqq K.$$

[1] We assume that $f \in C_+ \Rightarrow f(t) = \max(f(t+), f(t-))$.

We may now formulate and solve the equidistribution problem subject to the first constraint.

Problem 1. Given $f \in C_+$ and constants $c > 0$ and $K \geqq 1$, find a mesh which is

(1) s.e.d. on $[a, b]$ with respect to f and c;

(2) quasi-uniform with respect to K.

THEOREM 1. (See Pereyra and Sewell [10].)

A solution to Problem 1 is the mesh π *which is* e.d. *on* $[a, b]$ *with respect to* g *and* d, *where*

$$g(t) = \max(f(t), p) \quad \textit{with} \quad p = \frac{1}{K} \max_{t \in [a,b]} \{f(t)\},$$

and

$$d = \frac{1}{n} \int_a^b g,$$

with n *equal to the smallest integer such that* $nc \geqq \int_a^b g$.

The proof that such a mesh satisfies the conditions of Problem 1 follows directly from Lemma 1 and Lemma 2.

The idea of this solution is to *increase* the given function f in such a way as to avoid excessively large mesh steps which may arise from very small values of the function. We refer to this procedure as "*padding*". Equidistribution with respect to the padded function g, instead of the original function f, may of course increase the number of steps in the mesh.

Solving a constrained problem by using a padded function leads necessarily to a s.e.d. mesh with respect to the original function, rather than an e.d. one. Together with the possible increase in the number of mesh steps, this is the penalty paid to satisfy the smoothness requirement. Furthermore, since the padded function must be equidistributed on the mesh with respect to some $d \leqq c$, in order to guarantee compliance with the constraints, the actual bound on the integrals of the original function over each subinterval of the mesh may be less than required.

Theoretically, it is easy to determine the e.d. mesh π with respect to the padded function g and the constant d. The points of the mesh are simply equidistant points on the inverse function $[\int_a^x g]^{-1}$, separated by distance d. The practical aspects of generating the mesh π are considered in § 4.

The solution given by Theorem 1 is constructive, and is universal in the sense that it can be constructed for any $c > 0$ and any given function $f \in C_+$. However, this solution need not be the unique solution to Problem 1, or even the minimal solution. Indeed, consider the following example:

Example 1. Let f be equal to 1.0 on $[0, 0.05]$, $[\frac{1}{3} - 0.05, \frac{1}{3} + 0.05]$ and $[0.95, 1]$, and equal to 0.0 elsewhere on $[0, 1]$. We require a mesh on $[0, 1]$ which is s.e.d. with respect to f and $c = 0.1$, and is quasi-uniform with respect to constant $K = 2$. If we use the solution of Theorem 1, then the corresponding function g is

$$g(t) = \max(f(t), 0.5), \quad \text{where} \int_0^1 g = 0.6,$$

and we obtain the mesh

$$\pi: \{0, 0.15, \tfrac{11}{30} - 0.05, 0.45, 0.65, 0.85, 1.0\}$$

with six steps. However, a solution is also given by the two step mesh

$$\pi: \{0, \tfrac{1}{3}, 1\}.$$

It is possible to construct other examples which are less artificial and which better illustrate the difficulties, but these are considerably more complicated. This example does suggest, however, that a general method which always produces a minimal solution to the constrained equidistribution problem is unlikely to exist.

3. Locally bounded meshes. We now consider an extension of the concept of padding to the solution of the equidistribution problem subject to the second constraint. We make the following definition.

DEFINITION 4. We call mesh π *locally bounded* with respect to constraint $K \geqq 1$ if

$$\frac{1}{K} \leqq \frac{h_j}{h_{j-1}} \leqq K, \qquad j = 1, 2, \cdots, n-1.$$

Then the equidistribution problem becomes:

Problem 2. Given $f \in C_+$ and constants $c > 0$ and $K \geqq 1$, find the mesh π which is

(1) s.e.d. on $[a, b]$ with respect to f and c;

(2) locally bounded with respect to K.

With suitably chosen constants, a solution to Problem 1 is also a solution of Problem 2; however, this solution would obviously be too far from a minimal one in most cases. We therefore seek a suitable padding of the function f such that when the padded function is equidistributed, the ratio of the consecutive steps in the resulting mesh are bounded as required. Recalling the solution to Problem 1, we observe that there the padding p is chosen so that the steps in the e.d. mesh along the padding are constant and equal to the maximal length permitted by the constraint. It is therefore natural with Problem 2 to seek a padding for which the equidistributing mesh has adjacent steps with constant ratios equal to the maximum allowed. Such a function exists and has the form $p(t) = 1/(l \pm \lambda t)$, where l is an arbitrary parameter and $\lambda > 0$ depends on the required ratio K *and* on the e.d. constant c (contrary to Problem 1, where the padding p depends only on K). The details are given in the following theorem.

THEOREM 2. *Let $x, z > 0$, $K > 1$, $c > 0$ be given constants. Define*

$$\lambda = \frac{\log K}{c}$$

and

$$p(t) = \frac{1}{\lambda |t - x| + z}.$$

Let $x_1 < x_2 < x_3$ be such that

$$\int_{x_1}^{x_2} p = \int_{x_2}^{x_3} p = c$$

and either $x \leqq x_1$ or $x_3 \leqq x$. Then

$$\frac{x_3 - x_2}{x_2 - x_1} = \begin{cases} K & \text{if } x \leqq x_1, \\ \dfrac{1}{K} & \text{if } x_3 \leqq x. \end{cases}$$

Proof. Let $a_i = \lambda(x_i - x) + z$. Consider the case $x \leqq x_1$. By integration,

$$c = \frac{1}{\lambda} \log \frac{a_{i+1}}{a_i}, \qquad i = 1, 2,$$

and from the definition of λ, $K = a_{i+1}/a_i$. Then

$$\frac{x_3 - x_2}{x_2 - x_1} = \frac{a_3 - a_2}{a_2 - a_1} = \frac{Ka_2 - a_2}{a_2 - a_2/K} = K.$$

The case $x_3 \leqq x$ is similar.

The padding required for a solution of Problem 2 is essentially the envelope of all the curves of form $p(t) = 1/(l \pm \lambda t)$ which pass through points on the function to be sub-equidistributed. It is actually defined in the form of a mapping of one nonnegative function onto another, which must be examined in some detail in order to prove that it provides the necessary solution.

DEFINITION 5. Let λ be a given number. For any $f \in C_+$ we define

$$G(f; t, \tau) = f(\tau)/(1 + \lambda|t - \tau| f(\tau))$$

and the padding $P(f)$ of f by

$$P(f)(t) = \max_{\tau \in [a,b]} G(f; t, \tau).$$

We observe that the padding $P(f)$ is strictly positive on $[a, b]$, except in the case $f \equiv 0$. Further properties of $P(f)$ are given as follows:

LEMMA 3. $P(f) \geqq f$ *on* $[a, b]$.

Proof. $f(t) = G(f; t, t)$ and $P(f)(t) \geqq G(f; t, \tau)$ for any $\tau \in [a, b]$.

LEMMA 4. $P(P(f)) = P(f)$ *on* $[a, b]$.

Proof. Consider $g_i(t) = 1/(\lambda|t - \tau_i| + d_i)$, $i = 0, 1$, for some constants τ_0, τ_1, d_0, d_1. If $g_0(\xi) = g_1(\xi)$ for some $\xi \leqq \tau_0$, then either $g_0(t) = g_1(t)$ for all $t \leqq \xi$ or $g_0(t) < g_1(t)$ for all $t < \xi$. Now let us assume that there exists t_0 such that

$$P(P(f))(t_0) > P(f)(t_0).$$

From Definition 5, there exists τ_0 such that $P(P(f))(t_0) = G(P(f); t_0, \tau_0)$. By the assumption, $\tau_0 \neq t_0$. Suppose $\tau_0 > t_0$. Also from Definition 5 there exists τ_1 such that $P(f)(\tau_0) = G(f; \tau_0, \tau_1)$. Now, since $P(f)(\tau_0) = G(P(f); \tau_0, \tau_0) = G(f; \tau_0, \tau_1)$, if we let $g_0(t) \equiv G(P(f); t, \tau_0)$ and $g_1(t) \equiv G(f; t, \tau_1)$, then $g_0(t) = g_1(t)$ at $t = \xi = \tau_0$. From the result above, we then conclude that

$$G(P(f); t, \tau_0) \leqq G(f; t, \tau_1) \quad \text{on } [a, \tau_0],$$

and in particular, since $t_0 < \tau_0$,

$$P(f)(t_0) \geqq G(f; t_0, \tau_1) \geqq G(P(f); t_0, \tau_0) = P(P(f))(t_0),$$

which contradicts the initial assumption. The case $\tau_0 < t_0$ is shown similarly. Therefore we have $P(P(f)) \leqq P(f)$, but by Lemma 3 $P(P(f)) \geqq P(f)$ and the equality follows.

LEMMA 5. *Let* $f \in C_+$ *be such that* $P(f) = f$ *and let* $x \in [a, b]$. *Then*

(1) $f(t) \geqq G(f; t, x)$ *for all* $t \in [a, b]$;

(2) $f(t) \leqq 1/(-\lambda|t - x| + 1/f(x))$ *for all* $t \in [a, b]$ *such that* $|t - x| \leqq 1/\lambda f(x)$.

Proof. (1) is a direct result of the definition (and idempotency). To show (2), let $t \in [a, b]$ be such that $|t - x| \leqq 1/\lambda f(x)$. This is possible since $f = P(f)$ is strictly positive

on $[a, b]$, except in the trivial case. Then, by definition,

$$f(x) \geqq G(f; x, t) = 1/(\lambda|x-t| + 1/f(t)).$$

Solving this inequality for $f(t)$, under the assumption on t, gives the result.

The following theorem now establishes a solution to Problem 2.

THEOREM 3. *Let $\lambda > 0$, $f \in C_+$ and $f^* = P(f)$. Let π be equidistributing on $[a, b]$ with respect to f^* and some $c > 0$. Then*

(1) *π is* s.e.d. *with respect to f and c and*

(2) *for $K = e^{\lambda c}$ we have*

$$\frac{1}{K} \leqq \frac{h_j}{h_{j-1}} \leqq K, \qquad j = 1, 2, \cdots, n-1.$$

Proof.

(1) Since $f^* \geqq f$ by Lemma 3, then π is s.e.d. by Lemma 1.

(2) Let x_{i-1}, x_i, x_{i+1} be any three consecutive points of the mesh, and define

$$\tilde{x}_{i-1} = \max(x_{i-1}, x_i - 1/\lambda f^*(x_i)).$$

Define also

$$\tilde{f}(t) = 1/(\lambda(t - x_i) + 1/f^*(x_i)) \quad \text{for } t > \tilde{x}_{i-1}.$$

Let y_1, y_2, y_3 be such that $y_2 = x_i$ and

$$\int_{y_j}^{y_{j+1}} \tilde{f}(t)\, dt = c \quad \text{for } j = 1, 2.$$

Obviously, since c is finite and $\tilde{f}$ is not integrable at its singularity, $y_1 > x_i - 1/\lambda f^*(x_i)$. Now $f^* = P(f^*)$ by Lemma 4, and hence by Lemma 5

$$f^* \leqq \tilde{f} \quad \text{on } (\tilde{x}_{i-1}, x_i)$$

and

$$f^* \geqq \tilde{f} \quad \text{on } (x_i, x_{i+1}).$$

But, since $\int_{x_{i-1}}^{x_i} f^* = \int_{x_i}^{x_{i+1}} f^* = c$, we therefore have

$$y_2 - y_1 \leqq x_i - x_{i-1} = h_{i-1}$$

and

$$y_3 - y_2 \geqq x_{i+1} - x_i = h_i.$$

Finally, by Theorem 2, $y_3 - y_2 = K(y_2 - y_1)$ and thus

$$K = \frac{y_3 - y_2}{y_2 - y_1} \geqq \frac{h_i}{h_{i-1}}.$$

The other inequality follows similarly by using the function

$$\tilde{\tilde{f}}(t) = 1/(\lambda(x_i - t) + 1/f^*(x_i)),$$

and the theorem is proved.

To obtain the mesh π which sub-equidistributes f with respect to c, we pad f using $\lambda = (\log K)/d$ and equidistribute the padded function $f^* = P(f)$ with respect to d, where $d = (\int_a^b f^*)/n$ and n is the smallest integer such that $nc \geqq \int_a^b f^*$. In practice we cannot know exactly the constant d with which to pad the function before actually performing the padding. However, if we apply the padding using $\lambda = (\log K)/c$, and equidistribute f^* with respect to $d \leqq c$, then the mesh π is locally bounded with respect to a constant L

satisfying

$$\lambda = \frac{1}{c}\log K = \frac{1}{d}\log L,$$

and therefore $L \leqq K$. The resulting mesh is thus s.e.d. with respect to f and c and still locally bounded with respect to K, but the number of points in the mesh may be greater than required to satisfy the criteria of the problem.

The solution given by this procedure is again constructive and universal, but it is not necessarily minimal. Indeed, if f is the weight function of Example 1 (in § 2) and $c = 0.1$, $K = 2$, then the solution obtained has seven steps, while the two-step mesh defined in § 2 also satisfies the necessary constraints. In more realistic cases, where c is sufficiently small with respect to a smooth weight function f, there is evidence, however, that the procedure determines solutions with the minimal or close to the minimal number of steps. Further discussion is given in § 6.

4. Numerical algorithms. To find a s.e.d. mesh with respect to f and c which is locally bounded with respect to K, we require two algorithms. The first produces the padded function $P(f)$, and the second performs the equidistribution.

The padding process is carried out in two sweeps of the mesh. First the interval is searched in the forward direction. If at some point t^* the function f decreases faster than the padding function

$$g(t) = f(t^*)/(1 + \lambda(t - t^*)f(t^*))$$

fitted at that point, i.e., $g > f$ for $t > t^*$, then the function f is replaced by g as long as *this* padding dominates the function, that is, until some point is reached where $g < f$. These steps are repeated for increasing t until $t = b$ is reached. Then the same procedure is performed in the negative direction with

$$g(t) = f(t^*)/(1 - \lambda(t - t^*)f(t^*)).$$

The equidistribution is also done in two steps; for given f and c, we first approximate $g(t) = \int_a^t f(\tau)\, d\tau$ and then find the new mesh by inverse interpolation. The points x_j are given explicitly by

$$x_j = g^{-1}(j \cdot d), \qquad j = 0, 1, \cdots, n,$$

where g^{-1} is the inverse function of g, n is the smallest integer such that $nc \leqq g(b)$ and $d = g(b)/n$.

In practical applications the weight function f which is to be equidistributed is generally given in discrete form; that is, we have the data $y_j = f(t_j)$ given at $m + 1$ points t_j on the mesh

$$\pi_m\colon a = t_0 < t_1 < \cdots < t_m = b.$$

In the numerical procedures for implementing the padding and equidistribution algorithms, we therefore replace the processes described by suitable discrete approximations. Given the data, the steps in the padding procedure are performed for values of $t = t_j$ only, and the padded function $P(f)$ is returned in the form of a discrete data set $(t_j, P(f)(t_j))$, $j = 0, 1, \cdots, m$. In the equidistribution procedure, a trapezoidal rule is

used to determine $g(t_j)$ at each point t_j of the original mesh π_m. The result is exact for piecewise linear functions f. For the inverse interpolation we may use linear or quadratic approximations. The latter is exact for piecewise linear functions f, since g is then piecewise quadratic. Numerical results obtained with both linear and quadratic interpolation indicate that slightly better solutions are given by the quadratic process, as would be expected.

Higher order polynomial approximations may be used in both the integration and interpolation procedures, but the complications introduced do not seem to be justified. Low order rational approximations could also be used, for which the integration and interpolation would be exact along the padding of the function. Another strategy for improving the solutions is to repeat the equidistribution in the manner of an iterative refinement. Both these techniques are investigated in another paper (Kautsky and Nichols [7]).

Two simple Fortran subroutines (of about 30 lines each) have been developed to implement the padding and equidistribution procedures. Numerical results obtained with these routines are given in the next section. (Copies of the program codes are available from the authors.)

5. Examples. Numerical solutions were obtained for test functions of the form

$$f(x) = a_0 e^{-b_0 x} + a_1 e^{b_1(x-1)} + a_2 e^{-b_2(x-0.3)^2}$$

with various values of the coefficients a_i, b_i, $i = 0, 1, 2$, and $x \in [0, 1]$. Results are presented here for two examples.

Example A. $a_0 = 1.0, \quad a_1 = 1.5, \quad a_2 = 0.8,$

$b_0 = 30, \quad b_1 = 20, \quad b_2 = 300;$

Example B. $a_0 = a_1 = a_2 = 1.0,$

$b_0 = b_1 = 80, \quad b_2 = 800.$

The function values $f(x_j)$ were initially specified at 36 points:

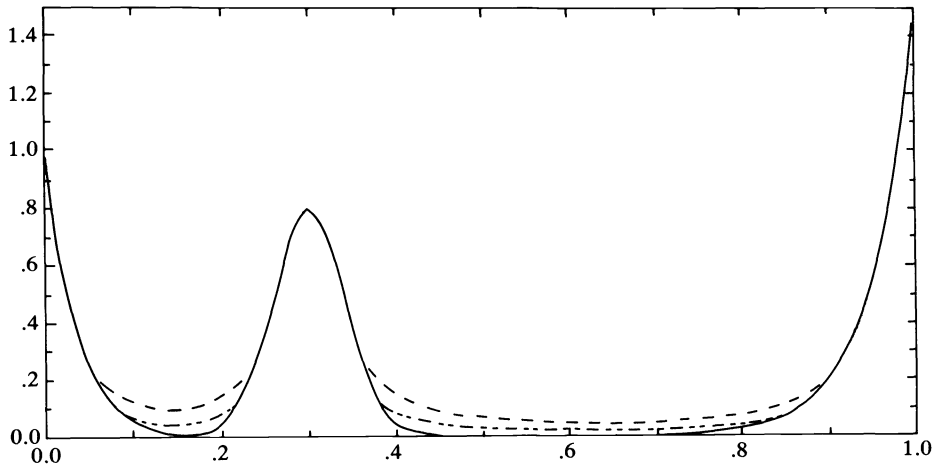
$$x_j = 0.02j, \qquad j = 0, 1, \cdots, 20,$$
$$x_j = 0.4 + 0.05j, \qquad j = 1, \cdots, 10,$$
$$x_j = 0.9 + 0.02j, \qquad j = 1, \cdots, 5.$$

The functions were equidistributed with and without the padding for comparison, and solutions were found for different choices of the equidistribution parameters K and c.

In Figs. 1 and 2, the original function and the padded function are shown for Examples A and B with $c = 0.005$ and two values of K ($K = 3.0$ and $K = 1.5$). Summaries of the results obtained for various equidistribution parameters are given in Tables 1 and 2. The number of points N in the mesh, the maximum step ratio R and the maximum of the integrals of the function over each step,

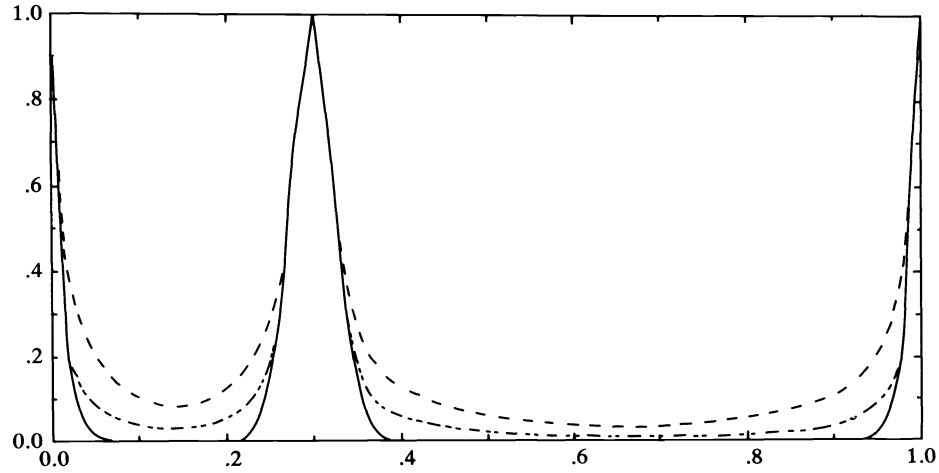
$$\overline{\text{INT}} = \max_{i=0,1,\cdots,n-1} \int_{x_i}^{x_{i+1}} f,$$

are shown for the equidistributing mesh without constraints (E) and for the equidistributing meshes constrained with respect to the given K ($K = 3.0$, $K = 1.5$).



Example A ——— $f(x)$
- · · - · · - $P(f)(x)$ $K = 3.0$
– – – – – – $P(f)(x)$ $K = 1.5$

FIG. 1



Example B ——— $f(x)$
- · · - · · - $P(f)(x)$ $K = 3.0$
– – – – – – $P(f)(x)$ $K = 1.5$

FIG. 2

TABLE 1
Example A

	$c = 0.005$			$c = 0.0025$		
	E	$K = 3.0$	$K = 1.5$	E	$K = 3.0$	$K = 1.5$
N	40	43	49	79	81	87
R	18.6	2.95	1.53	10.6	3.09	1.67
$\overline{\text{INT}}$	0.010	0.506E−2	0.503E−2	0.271E−2	0.257E−2	0.255E−2

TABLE 2
Example B

	$c = 0.005$			$c = 0.0025$		
	E	$K = 3.0$	$K = 1.5$	E	$K = 3.0$	$K = 1.5$
N	20	26	37	39	45	56
R	55.1	3.21	1.57	59.3	3.17	1.786
$\overline{\text{INT}}$	0.613E−2	0.530E−2	0.529E−2	0.053	0.266E−2	0.263E−2

From these results we conclude that the padding procedure successfully produces sub-equidistributing meshes which are locally bounded. Furthermore, in these examples very few extra points are required to achieve this smoothness, and function values can be obtained on the locally bounded mesh with very little extra expense.

We note that in the numerical approximation, the maximum of the local integrals ($\overline{\text{INT}}$) often exceeds c by a small amount. This occurs because the computation of the integral of the function from the discrete data is not sufficiently accurate. We also observe that the local bound K on the mesh ratios is not always satisfied precisely. This is due to the fact that with the discretized data, the correct padding of the function is not determined. In some areas a smaller padding than necessary is used, and at intersections of the computed paddings, larger steps than acceptable are allowed.

The padding and equidistribution procedures were also tested on Example B with initial data $f(x_j)$ given at 26 equally spaced points $x_j = j \cdot h$, $j = 0, 1, \cdots 25$, with $h = 0.04$. Essentially the same results were obtained as with the previously specified initial data.

The use of the constrained equidistribution procedures in specific practical applications, including the solution of boundary value problems and the regrading of statistical data, are described elsewhere (Kautsky and Nichols [6], [7]).

6. Asymptotic results and minimal solutions. In this section we examine the behavior of the solution to Problem 2, given by the procedure of § 3, for limiting values of the parameters K and c, and compare the asymptotic solutions obtained with known minimal solutions. This comparison provides an assurance of the effectiveness of the solution procedure, and at the same time an indication of its limitations.

We observe that the solution of § 3 is generated by equidistributing the padded function $P(f)$ with padding constant $\lambda = (\log K)/c$. Two limiting cases occur:

(i) if either $K \to 1$ for fixed c, or $c \to \infty$ for fixed K, then $\lambda \to 0$;

and

(ii) if either $K \to \infty$ for fixed c, or $c \to 0$ for fixed K, then $\lambda \to \infty$.

In the first case (i), it is easy to show that $f \in C_+$ implies

$$\lim_{\lambda \to 0} P(f) = M \equiv \max_{[a,b]} f.$$

Obviously, equidistributing the constant M gives a mesh with n constant steps, where n is the least integer such that $nc \geqq (b-a)M$. Therefore, if c is fixed and $K = 1$ in the limit, we obtain a uniform mesh as required. If K is fixed, however, then for all finite c sufficiently large ($c \geqq (b-a)M$), the mesh contains only one step, and the limiting solution as $c \to \infty$ is of no particular interest.

In the second case (ii), we have for $f \in C_+$

$$\lim_{\lambda \to \infty} P(f)(t) = f(t) \quad \text{for all } t \in [a, b],$$

and if f is continuous, the convergence is uniform. (For proofs, see Kautsky and Nichols [5].) The limiting solution is, therefore, the same mesh as would be generated by equidistributing the original weight function f without constraints. As $K \to \infty$ for fixed c, this is clearly the mesh we expect to obtain, since for sufficiently large K, the constraint should have no effect on the solution. Similarly, for fixed K, we expect the constraint to be ineffective if c is sufficiently small. However, as $c \to 0$, the number of steps in the mesh becomes infinite. If f is continuous, the ratios of adjacent steps in the solution converge to unity for any fixed K, and the constraint becomes superfluous, (although the mesh does not become uniform in the usual sense); but, if f is discontinuous, the convergence of $P(f)$ to f is not uniform, and there may always be adjacent steps in the solution with fixed ratio >1. (See Kautsky and Nichols [5].)

We now consider minimal solutions to Problem 2 for extreme values of c and K. If c is fixed and $K = 1$, there is a unique minimal solution with constant step size and at most n steps (where n is defined, as before, as the least integer $\geqq (b-a)M/c$). For some functions f, the number of steps in the minimal solution is exactly n, and the uniform mesh obtained by the procedure of § 3 as $K \to 1$ is therefore the minimal solution. (Of course, it is always possible to construct functions f for which a s.e.d. mesh with less than n steps exists for a particular choice of c.)

In the case where c is fixed and K is sufficiently large, the mesh obtained by equidistributing f is itself a solution to Problem 2. This mesh has a finite number of steps and some maximal ratio of consecutive steps, say K_E, and it therefore satisfies the constraints for all $K \geqq K_E$. This solution is also a minimal solution (unique if $\int_a^b f/c$ is an integer). The solution of § 3, obtained by equidistributing $P(f)$, clearly cannot be better since $P(f) \geqq f$. However, we have shown that $P(f) \to f$, as $K \to \infty$, and it is possible for certain f that $P(f) = f$ when $K = K_0$ ($K_0 \geqq K_E$). The procedure of § 3 then gives the same (minimal) solution for all $K \geqq K_0$. On the other hand, it is also possible to construct functions f such that $P(f) \neq f$ for any finite value of K, and in that case the procedure may give solutions with at least one more step than the minimal, even for very large K.

In the case K is fixed and f is *continuous*, the mesh which equidistributes f similarly provides a minimal solution to Problem 2 for sufficiently small c, since there then exists a $c_0 > 0$ such that $0 < c \leqq c_0$ implies $K_E \leqq K$. Again, the method of § 3 may give the same or slightly worse solutions as $c \to 0$, depending on the function f. If f is *not* continuous, however, the behavior of the e.d. mesh as $c \to 0$ is not uniform, and we cannot make any deductions concerning the minimal solution itself.

We conclude that the locally bounded s.e.d. meshes obtained by the procedure of § 3 are asymptotically minimal in certain circumstances, and behave in the limit for extreme values of K and c as desired.

7. Generalizations. We have investigated the problem of finding equidistributing meshes subject to two constraints on the smoothness of the mesh. In § 2, we discussed quasi-uniform meshes where the ratio of the maximum to the minimum step in the mesh was required to be bounded. In § 3 we examined locally bounded meshes where the ratios of *adjacent* mesh steps were bounded by a constant. Numerical methods for implementing the theory were also given and shown to be effective in practice.

Other types of constraint are also of interest. We describe here some generalizations of the techniques developed for the first two problems, which can be used to treat different forms of restrictions.

In Babuska and Rheinboldt [2], the concept of a (μ, γ)-regular mesh was introduced. Such a mesh satisfies

$$\mu(\max_j h_j)^\gamma \leqq \min_j h_j.$$

Although the meshes discussed by Babuska and Rheinboldt are shown to be automatically (μ, γ)-regular for specified γ and some constant μ, it may prove useful in practical applications to impose (μ, γ)-regularity as a restriction with chosen values of μ, γ. A technique analogous to that of § 2 can be employed in that case. A s.e.d. mesh with respect to f and c which is (μ, γ)-regular is obtained by equidistributing a padded function $g(t)$ (in the manner of Theorem 1) where $g(t) = \max(f(t), p)$ and constant p is now such that

$$p^\gamma = \mu c^{\gamma-1} \max_{\tau\in[a,b]} f(\tau).$$

The proof follows again from Lemma 1 and Lemma 2. We note that for $\gamma \neq 1$ the padding function now depends on c. For $\gamma = 1$ this case reduces to Problem 1 with $K = 1/\mu$.

In the case where the weight function f is integrable, but has a singularity (or near singularity) in the domain $[a, b]$, methods of padding with respect to $\max_{[a,b]} f$ may prove impossible or unnecessarily pessimistic. However, the solution methods remain valid if, in the expression for the padding constant p, the factor $\max_{[a,b]} f$ is replaced by c/τ^* where τ^* is such that $\tau^* \leqq \tau$ for any $\tau > 0$ with $\int_x^{x+\tau} f = c$. Thus s.e.d. quasi-uniform or (μ, γ)-regular meshes can be found even in the case of singular (integrable) weight functions.

The locally bounded mesh introduced in § 3 is a special case of the following concept.

DEFINITION 6. Let $s \leqq n$ be an integer and let $\mu_1, \mu_2, \cdots, \mu_s \leqq 1$ be real numbers. We say that a mesh π is $(s; \mu_1, \mu_2, \cdots, \mu_s)$-*locally bounded* if the following conditions are satisfied for $i = 1, 2, \cdots, s$:

$$\mu_i \max(h_j, h_{j+1}, \cdots, h_{j+i}) \leqq \min(h_j, h_{j+1}, \cdots, h_{j+i})$$

for all $j = 0, 1, \cdots, n-i-1$ (assuming the notation of Definition 1).

Obviously the locally bounded meshes of § 3 are $(1; 1/K)$-locally bounded in the above sense. Also the quasi-uniform meshes of § 2 may be regarded as $(n-1; 0, 0, \cdots, 0, 1/K)$-locally bounded meshes. The following lemma then holds.

LEMMA 6. *Let an integer* $s \leqq n$ *and real numbers* $\mu_1, \mu_2, \cdots, \mu_s$ *be given. Set*

$$\mu = \max_{i=1,2,\cdots,s} (\mu_i)^{1/i}.$$

Then any $(1; \mu)$*-locally bounded mesh is* $(s; \mu_1, \mu_2, \cdots, \mu_s)$*-locally bounded.*

As a corollary, an equidistribution problem subject to the mesh being $(s; \mu_1, \mu_2, \cdots, \mu_s)$-locally bounded may be solved by the method introduced in § 3. Depending on $s, \mu_1, \mu_2, \cdots, \mu_s$ and the variations of the weight function f, such a solution may lead to a mesh with more steps than necessary. However, because of its simplicity and low implementation costs, we feel that in most applications this solution would prove sufficient.

Acknowledgment. The authors would like to thank Professor Gene H. Golub of Stanford University for his kind hospitality and generous support during their stay at the new "Serra House".

REFERENCES

[1] C. M. ABLOW AND S. SCHECTER, *Campylotropic coordinates*, J. Comp. Phys., 27 (1978), pp. 351–362.

[2] I. BABUSKA AND W. C. RHEINBOLDT, *Analysis of optimal finite-element meshes in* R^1, Math. Comput., 33 (1979), pp. 435–463.

[3] C. DE BOOR, *Good approximations by splines with variable knots*, II, in Conference on the Numerical Solution of Differential Equations. Lecture Notes in Mathematics, 363 : 12, Springer-Verlag, New York, 1973.

[4] V. E. DENNY AND R. B. LANDIS, *A new method for solving two-point boundary value problems using optimal node distribution*, J. Comp. Phys., 9 (1972), pp. 120–137.

[5] J. KAUTSKY AND N. K. NICHOLS, *Equidistributing meshes with constraints*, Stanford University Computer Science Report Stan-CS-79-766, 1979.

[6] ———, *Smooth regrading of discrete data*, University of Reading, Dept. of Mathematics Numerical Analysis Report, NA 1/80, 1980.

[7] ———, *Adaptive selection of constrained equidistributing meshes using rational approximations*, in preparation.

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

[9] C. E. PEARSON, *On a differential equation of boundary layer type*, J. Math. Phys., 47 (1968), pp. 134–154.

[10] V. PEREYRA AND E. G. SEWELL, *Mesh selection for discrete solution of boundary problems in ordinary differential equations*, Numer. Math., 23 (1975), pp. 261–268.

[11] R. D. RUSSELL AND J. CHRISTIANSEN, *Adaptive mesh selection strategies for solving boundary value problems*, this Journal, 15 (1978), pp. 59–80.

[12] A. B. WHITE, JR., *On selection of equidistributing meshes for two-point boundary problems*, this Journal, 16 (1979), pp. 472–502.

SIAM J. SCI. STAT. COMPUT.
Vol. 1, No. 4, December 1980

0196-5204/80/0104-0010 $01.00/0

A STABLE ALGORITHM FOR SOLVING THE MULTIFACILITY LOCATION PROBLEM INVOLVING EUCLIDEAN DISTANCES*

P. H. CALAMAI† AND A. R. CONN‡

Abstract. There is a rapidly growing interdisciplinary interest in the application of location models to real life problems. Unfortunately, the current methods used to solve the most popular minisum and minimax location problems are computationally inadequate. A more unified and numerically stable approach for solving these problems is proposed. Detailed analysis is done for the linearly constrained Euclidean distance minisum problem for facilities located in a plane. Preliminary computational experience suggests that this approach compares favourably with other methods.

Key words. continuous location problems, nonsmooth optimization, numerical linear algebra

1. Introduction. Facility location problems are generally concerned with finding the optimal location of a set of new facilities in a network of existing facilities. Over the past few decades various forms of the problem have been developed and addressed in the literature (see Francis and Goldstein [11]). In this paper we present a stable method for solving the multifacility location problem involving Euclidean distances. We assume that the feasible region is connected and that the parameters are static with respect to time (for a discussion of the dynamic aspect of these problems see Wesolowsky and Truscott [17]). In addition, we assume that the problem is well formulated (Francis and Cabot [10] formally describe this and many other properties of the problem).

Various approaches have been used in an attempt to efficiently solve this location problem. One such method involves the use of a heuristic algorithm described by Vergin and Rogers [14]. Unlike their approach, most solution techniques guarantee optimality. Included among these are the convex programming approach described by Love [13], the hyperbolic approximation method used by Eyster, White and Wierwille [9], the pseudo-gradient technique described by Calamai and Charalambous [3] and the subgradient algorithm presented by Chatelon, Hearn and Lowe [5].

One property of this problem that causes considerable difficulty is the fact that the objective function is not everywhere differentiable. This nondifferentiability occurs whenever any new facility coincides with any other new facility or with an existing facility. As a result, standard minimization techniques cannot be directly applied. We overcome this difficulty by projecting onto a particular affine space in which the current near zero terms stay unchanged and the remaining well-defined terms can be decreased. Then, when we think we are optimal, we perform a linear refinement step that makes the near zero terms exactly zero. Under mild and suitable conditions we indicate that this technique converges to the solution of our problem.

Another significant property of this technique is that it can be used for the linearly constrained problem. Here we replace the constrained problem by a sequence of unconstrained problems by using a penalty function due to Pietrzykowski

*Received by the editors March 24, 1980. This work was supported in part by NSERC Grant No. A8639. This paper was typeset at the University of Waterloo using the Troff software developed for the Unix operating system. Final copy was produced on a Photon Econosetter.

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

[15] and Zangwill [18]. Using this method we are able to obtain a sequence of points (optimal to the unconstrained penalty function) which converge to the desired constrained optimum.

2. Problem statement and definitions. The multifacility minisum problem involving costs associated with Euclidean distances between facilities in $\mathbf{R}^q$ can be stated as: Find the point $x^{*T} = \{x_1^{*T}, \ldots, x_n^{*T}\}$ in $\mathbf{R}^{qn}$ to minimize

$$f(x) = \sum_{1 \leqslant j < k \leqslant n} v_{jk} \| x_j - x_k \| + \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ji} \| x_j - p_i \| , \tag{2.1}$$

where

$n \triangleq$ number of new facilities (NF's) to be located.

$m \triangleq$ number of existing facilities (EF's).

$x_j^T = (x_{j1} \cdots x_{jq}) \triangleq$ vector location of NF_j in $\mathbf{R}^q$, $j = 1, \ldots, n$.

$p_i^T = (p_{i1} \cdots p_{iq}) \triangleq$ vector location of EF_i in $\mathbf{R}^q$, $i = 1, \ldots, m$.

$v_{jk} \triangleq$ nonnegative constant of proportionality relating the l_2 distance between NF_j and NF_k to the cost incurred, $1 \leqslant j < k \leqslant n$.

$w_{ji} \triangleq$ nonnegative constant of proportionality relating the l_2 distance between NF_j and EF_i to the cost incurred, $1 \leqslant j \leqslant n$, $1 \leqslant i \leqslant m$.

$\| x_j - x_k \| = (\sum_{c=1}^{q} | x_{jc} - x_{kc} |^2)^{1/2} \triangleq l_2$ distance between NF_j and NF_k, $1 \leqslant j < k \leqslant n$.

$\| x_j - p_i \| = (\sum_{c=1}^{q} | x_{jc} - p_{ic} |^2)^{1/2} \triangleq l_2$ distance between NF_j and EF_i, $1 \leqslant j \leqslant n$, $1 \leqslant i \leqslant m$.

In order to simplify the remainder of the discussion we set $q = 2$ (i.e., all facilities are in $\mathbf{R}^2$). Now if we let $\eta = (n^2 - n)/2$ and $\tau = \eta + mn$, and if we define the constants

$$\{\alpha_1, \ldots, \alpha_\tau\} = \{\underbrace{v_{12}, \ldots, v_{1n}}_{n-1}, \underbrace{v_{23}, \ldots, v_{2n}}_{n-2}, \ldots, \underbrace{v_{n-1\,n}}_{1}, \underbrace{w_{11}, \ldots, w_{1m}}_{m}, \ldots, \underbrace{w_{n1}, \ldots, w_{nm}}_{m}\},$$

$$\{\beta_1, \ldots, \beta_\tau\} = \{\underbrace{1, \ldots, 1}_{n-1}, \underbrace{2, \ldots, 2}_{n-2}, \ldots, \underbrace{n-1}_{1}, \underbrace{1, \ldots, 1}_{m}, \ldots, \underbrace{n, \ldots, n}_{m}\},$$

$$\{\gamma_1, \ldots, \gamma_\tau\} = \{\underbrace{2, 3, \ldots, n}_{n-1}, \underbrace{3, 4, \ldots, n}_{n-2}, \ldots, \underbrace{n}_{1}\},$$

and the index set $M = \{1, \ldots, \tau\}$ then (2.1) can be restated more conveniently as

$$\text{minimize } f(x) = \sum_{i \in M} \| A_i^T x - b_i \| , \tag{2.2}$$

where the $2\times 2n$ matrix A_i^T and the 2×1 vector b_i are defined by

$$A_i^T = \begin{cases} [0_1 \quad \alpha_i I \quad 0_2 \quad -\alpha_i I \quad 0_3], & i = 1, \ldots, \eta, \\ [0_1 \quad \alpha_i I \quad 0_4], & i = \eta+1, \ldots, \tau, \end{cases}$$

$$b_i = \begin{cases} 0_5, & i = 1, \ldots, \eta, \\ \alpha_i p_e, & i = \eta+1, \ldots, \tau, \end{cases}$$

and

I is a 2×2 identity matrix,

$0_1, 0_2, 0_3, 0_4$ and 0_5 are zero matrices of dimension $2\times 2(\beta_i - 1)$, $2\times 2(\gamma_i - \beta_i - 1)$, $2\times 2(n - \gamma_i)$, $2\times 2(n - \beta_i)$ and 2×1 respectively,

$e = (i - \eta) \bmod m$ and $p_0 \equiv p_m$.

In a manner analogous to that of Bartels, Conn and Sinclair [2], define for any point x in $\mathbf{R}^{2n}$: the residual vectors

$$r_i(x) = A_i^T x - b_i \quad \forall i \in M,$$

the index set

$$I_\epsilon(x^k) = \{i \in M \mid \| r_i(x^k)\| \leqslant \epsilon f(x^{k-1})/\tau\} = \{i_1, \ldots, i_{t_k}\}, \qquad \epsilon \geqslant 0,$$

the corresponding matrix and vector

$$\mathbf{A} = \mathbf{A}(x^k) = [A_{i_1}, \ldots, A_{i_{t_k}}] \text{ and } r(x^k)^T = [r_{i_1}(x^k)^T, \ldots, r_{i_{t_k}}(x^k)^T], \tag{2.3}$$

and the vector

$$\tilde{\nabla} f_k = \sum_{i \in M \setminus I_\epsilon(x^k)} \nabla(\| r_i(x^k)\|) = \sum_{i \in M \setminus I_\epsilon(x^k)} A_i r_i(x^k)/\| r_i(x^k)\|.$$

(Note that for $\epsilon = 0$ the index set $I_\epsilon(x^k)$ corresponds to those terms which are exactly *active* (binding) at the point x^k. Conn [7] gives an account of the necessity for considering, as we do, "*near-active*" terms.)

Finally, define P_k as the orthogonal projector onto $S_k^\perp$, where S_k is the space spanned by the columns of $\mathbf{A}(x^k)$. *For the present it will be assumed that* $\mathbf{A}(x^k)$ *has full rank.*

3. Optimality conditions. For all points x close enough to x^k in $\mathbf{R}^{2n}$ we have $I_\epsilon(x) = I_\epsilon(x^k)$. Furthermore,

$$\begin{aligned} f(x) &= \sum_{i \in M \setminus I_\epsilon(x^k)} \| r_i(x)\| + \sum_{i \in I_\epsilon(x^k)} \| r_i(x)\| \\ &= h_1(x) + h_2(x). \end{aligned} \tag{3.1}$$

Assume $P_k \nabla h_1(x^k) \neq 0$ (i.e., $P_k \tilde{\nabla} f_k \neq 0$); then for $d^k = -P_k \tilde{\nabla} f_k$ we have

$$\begin{aligned} h_1(x^k + \lambda d^k) &= h_1(x^k) + \lambda (d^k)^T \nabla h_1(x^k) + O(\lambda^2) \\ &= h_1(x^k) - \lambda \| P_k \tilde{\nabla} f_k\|^2 + O(\lambda^2), \end{aligned} \tag{3.2a}$$

and since $A_i^T d^k = 0$ for all $i \in I_\epsilon(x^k)$,

(3.2b) $$h_2(x^k + \lambda d^k) = h_2(x^k).$$

Combining equations (3.2a) and (3.2b) yields the following result:

(3.3) $$f(x^k + \lambda d^k) - f(x^k) = -\lambda \| P_k \tilde{\nabla} f_k \|^2 + O(\lambda^2).$$

Therefore for $d^k = -P_k \tilde{\nabla} f_k \neq 0$ there exists $\delta > 0$ such that $f(x + \lambda d) < f(x)$ for all $0 < \lambda < \delta$. Alternatively, if $P_k \nabla f_k = 0$ then *(under the linear independence assumption)* the corresponding point x^k is called a *(nondegenerate) dead point.* In this case ∇f_k can be expressed *(uniquely)* as a linear combination of the columns of $\mathbf{A}(x^k)$,

(3.4) $$\tilde{\nabla} f_k = \mathbf{A}u = \sum_{i \in I_\epsilon(x^k)} A_i u_i, \qquad u^T = [u_1^T \cdots u_{t_k}^T].$$

Then, for any d^k,

$$h_1(x^k + \lambda d^k) = h_1(x^k) + \lambda (d^k)^T \tilde{\nabla} f_k + O(\lambda^2)$$

(3.5a)
$$= h_1(x^k) + \lambda \sum_{i \in I_\epsilon(x^k)} u_i^T A_i^T d^k + O(\lambda^2),$$

and

$$h_2(x^k + \lambda d^k) = h_2(x^k) + \lambda \left[\sum_{i \in I_0(x^k)} \| A_i^T d^k \| + \sum_{i \in I_\epsilon(x^k) \setminus I_0(x^k)} \nabla^T (\| r_i(x^k) \|) d^k \right] + O(\lambda^2)$$

(3.5b)
$$= h_2(x^k) + \lambda \left[\sum_{i \in I_0(x^k)} \| A_i^T d^k \| + \sum_{i \in I_\epsilon(x^k) \setminus I_0(x^k)} \frac{r_i(x^k)^T A_i^T d^k}{\| r_i(x^k) \|} \right] + O(\lambda^2).$$

Combining equations (3.5a) and (3.5b) yields the following result:

$$f(x^k + \lambda d^k) - f(x^k) = \lambda \left[\sum_{i \in I_0(x^k)} (u_i^T A_i^T d^k + \| A_i^T d^k \|) \right.$$

(3.6)
$$\left. + \sum_{i \in I_\epsilon(x^k) \setminus I_0(x^k)} \left(u_i^T A_i^T d^k + \frac{r_i(x^k)^T A_i^T d^k}{\| r_i(x^k) \|} \right) \right] + O(\lambda^2).$$

Now, if there exists an index $l \in I_\epsilon(x^k)$ such that $\| u_l \| > 1$, then take

(3.7) $$d^k = -P_{k/l} A_l u_l,$$

where $P_{k/l}$ is the orthogonal projector onto $S_{k/l}^{\perp}$ and $S_{k/l}$ is the space spanned by the columns of $\mathbf{A}(x^k)$ with columns A_l deleted. Then, since $A_i^T d^k = 0$ for all $i \in I_\epsilon(x^k) - \{l\}$ and $A_l^T d^k = -A_l^T P_{k/l} A_l u_l = -\rho u_l$ where $\rho > 0$ (see Appendix), we

have, for sufficiently small $\lambda > 0$,

$$f(x^k + \lambda d^k) - f(x^k) = \begin{cases} -\lambda\rho\left[\| u_l \|^2 + \dfrac{r_l(x^k)^T u_l}{\| r_l(x^k)\|}\right] < 0, & l \in I_\epsilon(x^k)\backslash I_0(x^k) \quad (3.8a) \\ -\lambda\rho \| u_l \| (\| u_l \| - 1) < 0, & l \in I_0(x^k) \quad (3.8b) \end{cases}$$

Thus, if there exists $l \in I_\epsilon(x^k)$ such that $\| u_l \| > 1$ then, for $d = -P_{k/l} A_l u_l$, there exists a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ for all $0 < \lambda < \delta$. What happens if x^k is a dead point and $\| u_i \| \leqslant 1 \ \forall i \in I_\epsilon(x^k)$?

Case I. $I_\epsilon(x^k)\backslash I_0(x^k) = \varnothing$.

Under these circumstances we have

$$f(x^k + \lambda d^k) - f(x^k) = \lambda \sum_{i \in I_0(x^k)} (u_i^T A_i^T d^k + \| A_i^T d^k \|) + O(\lambda^2), \quad (3.9)$$

which is nonnegative for all d^k in $\mathbf{R}^{2n}$. Thus x^k is at least a weak local minimum and hence a global minimum of the convex function $f(x)$.

Case II. $I_\epsilon(x^k)\backslash I_0(x^k) \neq \varnothing$.

For any $l \in I_\epsilon(x^k)\backslash I_0(x^k)$ take $d^k = -P_{k/l} A_l r_l(x^k)$. For this choice we have $A_i^T d^k = 0 \ \forall i \in I_\epsilon(x^k) - \{l\}$ and (3.6) becomes

$$f(x^k + \lambda d^k) - f(x^k) = -\lambda\rho(u_l^T r_l(x^k) + \| r_l(x^k)\|) + O(\lambda^2), \quad (3.10)$$

which is nonpositive (strictly negative unless $u_l = -r_l(x^k)/\| r_l(x^k)\|$).The next section explains a method for avoiding this case.

4. The linear refinement. If we are at the point x^k, and $\| P_k \tilde{\nabla} f_k \|$ is "*small*", then we may be approaching a dead point, say $\hat{x}$. We can then obtain estimates $\{\hat{u}_i\}$ of the "*dual*" variables $\{u_i\}$ by finding the least-squares solution to

$$\tilde{\nabla} f_k = \sum_{i \in I_\epsilon(x^k)} A_i u_i. \quad (4.1)$$

Then if $\| \hat{u}_i \| < 1 \ \forall i \in I_\epsilon(x^k)$ we may be near a local minimum (note that if $I_\epsilon(x^k)\backslash I_o(x^k) = \varnothing$ this is especially clear). We therefore wish to satisfy the near-active terms exactly. In [8], Conn and Pietrzykowski define a *vertical component*, based on linearizations, in an attempt to satisfy their "near-activities" exactly. Here we wish to find the solution to the already linear system of equations

$$A_i^T(x^k - v^k) - b_i = 0 \quad \forall i \in I_\epsilon(x^k). \quad (4.2)$$

This is best accomplished by evaluating the least-squares solution of minimal norm to

$$\mathbf{A}(x^k)^T v^k = r(x^k), \quad (4.3)$$

which is given, *algebraically at least*, by

$$v^k = (\mathbf{A}(x^k)^+)^T r(x^k), \quad (4.4)$$

where $\mathbf{A}(x^k)^+$ is the generalized (Moore-Penrose) inverse of $\mathbf{A}(x^k)$. Note that once a refinement step is taken the set $I_\epsilon(x^k)\backslash I_0(x^k)$ is nullified and thus Case II above need not be considered.

5. Remarks on convergence. The optimality conditions (§3) along with the linear refinement (§4) show that either the point x^k is optimal or there exists a descent direction d^k such that, for small $\lambda > 0$,

$$f(x^k + \lambda d^k) - f(x^k) = \lambda (d^k)^T g \quad \text{with } (d^k)^T g < 0. \tag{5.1}$$

We take

$$d^k = -P_k \tilde{\nabla} f_k \quad \text{and} \quad g = \tilde{\nabla} f_k \tag{5.2}$$

if x^k is not a dead point, and

$$d^k = -P_{k/l} A_l u_l \text{ and } g = \begin{cases} \tilde{\nabla} f_k - A_l u_l / \| u_l \|, & l \in I_0(x^k), \\ \tilde{\nabla} f_k + A_l r_l / \| r_l \|, & l \in I_\epsilon(x^k) \backslash I_0(x^k), \end{cases} \tag{5.3}$$

(where $l \in I_\epsilon(x^k)$ is an index giving $\| u_l \| > 1$) when x^k is a dead point. (As in [2], the vector g will be referred to as the *restricted gradient* of the function.)

Given a descent direction one is then able to determine a "sufficient" decrease in f as in [8, Prop. 1] (In practice we use the line search outlined in the next section.) One may thus analogously ([8, Thm. 3]) prove convergence. It is beyond the scope of this paper to give the theoretical details of these proofs. Furthermore, we are ultimately more interested in a second-order version and the subsequent convergence proofs.

6. The stepsize. Suppose we are at a point $x \in \mathbf{R}^{2n}$ and a direction d has been chosen as described. Further, suppose our function is of the form

$$\Phi(x) = \sum_{i \in M} \| \phi_i(x) \|, \tag{6.1}$$

where the functions $\phi_i(x)$ are all linear. Clearly a minimum of Φ, in the direction d, must be at a point $\bar{x}$ on d where, for at least one $i \in M \backslash I_\epsilon(x^k)$, we have $\phi_i(\bar{x}) = 0$. The points $\bar{x}$ are called "*breakpoints*". Therefore we determine the stepsize λ to the breakpoint that gives the minimum function value along d. In other words, *as long as the function is decreasing past the breakpoint we continue to move along d through successive breakpoints.*

To extend this idea to the case where the functions $\phi_i(x)$ are nonlinear is not difficult as long as we are: 1) content with estimating the location of breakpoints and, 2) willing to use a linear approximation in evaluating $d^T g(\bar{x})$.

In the first instance we wish to find the values of λ_i satisfying

$$A_i^T(x + \lambda_i d) - b_i = 0 \quad \forall i \in M \backslash I_\epsilon(x). \tag{6.2}$$

A good approximation to use, in the l_2 sense, comes from the equations:

$$\lambda_i = \frac{-(A_i^T d)^T r_i(x)}{\| A_i^T d \|^2} \quad \forall i \in M \backslash I_\epsilon(x). \tag{6.3}$$

In the second instance we wish to find an expression for $d^T g(x + \lambda d)$ which is linear in λ. Using the first-order Taylor series approximation about the point x, we get

$$d^T g(x + \lambda d) = d^T g(x) + \lambda \sum_{i \in M \backslash I_\epsilon(x)} \left\{ \frac{\| A_i^T d \|^2}{\| r_i(x) \|} - \frac{[(A_i^T d)^T r_i(x)]^2}{\| r_i(x) \|^3} \right\}. \tag{6.4}$$

With these ideas, the following stepsize algorithm (similar in concept to the one given in [4]) can be formulated:

1) Set:

$$\lambda_i \leftarrow \frac{-(A_i^T d)^T r_i(x)}{\| A_i^T d \|^2} \quad \forall i \in M \backslash I_\epsilon(x).$$

$$J_1 \leftarrow \{i \in M \backslash I_\epsilon(x) \mid \lambda_i > 0\}.$$

$$\Delta \leftarrow \sum_{i \in M \backslash I_\epsilon(x)} \left\{ \frac{\| A_i^T d \|^2}{\| r_i(x) \|} - \frac{[(A_i^T d)^T r_i(x)]^2}{\| r_i(x) \|^3} \right\}.$$

$$\hat{\lambda} \leftarrow -\frac{d^T g(x)}{\Delta}.$$

$$\nu \leftarrow 1, \quad l_0 \leftarrow 0, \quad \lambda_0 \leftarrow \hat{\lambda}, \quad \delta \leftarrow -\delta_0 d^T g(x).$$

2) IF ($J_\nu = \varnothing$) THEN go to 5.

3) Determine $l_\nu \in J_\nu$ such that $\lambda_{l_\nu} \leqslant \lambda_i \quad \forall i \in J_\nu$.

4) IF ($\lambda_{l_\nu} \geqslant \hat{\lambda}$) THEN go to 5.
ELSE set $J_{\nu+1} \leftarrow J_\nu - \{l_\nu\}$
$\nu \leftarrow \nu + 1$ and go to 2.

5) Set $\tilde{x} \leftarrow x + \lambda_{l_{\nu-1}} d$.

6) IF ($f(\tilde{x}) < f(x) - \lambda_{l_{\nu-1}} \delta$) THEN set $x \leftarrow \tilde{x}$ and return.
ELSE use x and $\tilde{x}$ in a successive cubic interpolation routine to get a new x, and return.

Notes:

δ_0 is some preassigned positive constant.

The cubic line search used in step 6 need not be (and in our present implementation is not) exact.

7. The minimization algorithm.

1) Choose any $x^1 \in \mathbf{R}^{2n}$ and set $k \leftarrow 1$. Initialize $\epsilon_{ACT}, \epsilon_{ZP1}, \epsilon_{ZP2}$ and ϵ. Set RSOL$\leftarrow$.FALSE.

2) Evaluate all residuals. Identify all index sets, the vector $r(x^k)$ and the matrix $\mathbf{A}(x^k)$. Evaluate ∇f_k and construct P_k.

3) Set $d^k \leftarrow -P_k \tilde{\nabla} f_k$, and $g \leftarrow \tilde{\nabla} f_k$.

4) Consider retaining same ϵ_{ACT} - active set:

IF ($\| d^k \| \geqslant \delta_1$) THEN go to 10.

5) Determine the "duals". Find the least-squares solution $\hat{u}$ to:

$$\mathbf{A}u = \tilde{\nabla} f_k.$$

6) Decide whether or not to drop an activity:

IF ($\| \hat{u}_i \| \leqslant 1 \quad \forall i \in I\epsilon_{\mathrm{ACT}}(x^k)$) THEN go to 8.

7) Drop an activity:

Choose $l \in I\epsilon_{\mathrm{ACT}}(x^k)$ with $\| \hat{u}_l \| > 1$ and set $d^k \leftarrow -P_{k/l} A_l \hat{u}_l$. Set

$$g \leftarrow \begin{cases} \tilde{\nabla} f_k - A_l \hat{u}_l / \| \hat{u}_l \| & l \in I_0(x^k) \\ \tilde{\nabla} f_k + A_l r_l / \| r_l \| & l \in I_{\epsilon_{\mathrm{ACT}}}(x^k) \backslash I_0(x^k). \end{cases}$$

Go to 10.

8) Consider checking for optimality:

IF ($\| d^k \| > \delta_2$) THEN go to 10.

9) Determine optimality or perform the linear refinement:

IF (RSOL .OR. $r_k^{\infty} \leqslant \epsilon$) THEN STOP!

ELSE Solve, in the least-squares sense, the system

$$\mathbf{A}^T v^k = r(x^k).$$

IF ($f(x^k - v^k) < f(x^k)$) THEN set $x^{k+1} \leftarrow x^k - v^k$

RSOL $\leftarrow$ .TRUE.

$k \leftarrow k + 1$ and go to 2.

10) Use the line search algorithm to find x^{k+1}. Set RSOL $\leftarrow$ .FALSE. , $k \leftarrow k + 1$, and go to 2.

Notes:

$$r_k^{\infty} = \underset{i \in I_{\epsilon_{\mathrm{ACT}}}(x^k)}{\text{maximum}} (\| r_i(x^k) \|).$$

$$\delta_1 = \begin{cases} \epsilon_{\mathrm{ZP1}} f(x^k)/\tau, & I_{\epsilon_{\mathrm{ACT}}}(x^k) = \varnothing, \\ \epsilon_{\mathrm{ZP1}} \| g \|, & \text{otherwise.} \end{cases}$$

$$\delta_2 = \begin{cases} \epsilon_{\mathrm{ZP2}} f(x^k)/\tau, & I_{\epsilon_{\mathrm{ACT}}}(x^k) = \varnothing, \\ \epsilon_{\mathrm{ZP2}} \| g \|, & \text{otherwise.} \end{cases}$$

8. Linear constraints and the penalty function. Consider the following constrained form of the minisum multifacility location problem:

$$\begin{aligned} &\text{minimize } f(x) = \sum_{i \in M} \| A_i^T x - b_i \| \\ &\text{such that } r_i(x) = a_i^T x - b_i \geqslant 0, \quad i = \tau + 1, \ldots, \tau', \\ &\qquad\qquad r_i(x) = a_i^T x - b_i = 0, \quad i = \tau' + 1, \ldots, \tau''. \end{aligned} \tag{8.1}$$

We can transform this problem into the penalty function of Pietrzykowski [15] and solve the sequence of unconstrained problems,

$$\text{(8.2)} \qquad \text{minimize } p(x,\mu) = \mu f(x) - \sum_{i\in LI} \min\,[0, r_i(x)] + \sum_{i\in LE} |\, r_i(x)|\,,$$

where μ is a positive parameter, $LI = \{\tau+1, \ldots, \tau'\}$ and $LE = \{\tau'+1, \ldots, \tau''\}$. It has been shown [15] that under *mild and suitable conditions*, an exact minimum of (8.2) coincides with an exact minimum of (8.1) for all values of μ sufficiently small. This suggests the following outer algorithm:

1) Choose $\mu > 0$.
2) Minimize $p(x,\mu)$ over x.
3) IF (we are optimal) THEN STOP!
4) Set $\mu \leftarrow \mu/10$ and go to 2.

The minimization in step 2 can be performed using obvious extensions of the techniques already described (see [1]). Some of the details of these extensions are now given.

For any x^k in $\mathbf{R}^{2n}$ define the index sets

$$I_\epsilon^{AI}(x^k) = \{i \in LI \;\big|\; |\, r_i(x^k)| \leqslant \epsilon\} = \{i_{t_k+1}, \ldots, i_{t'_k}\},$$

$$\text{(8.3)} \qquad I_\epsilon^{VI}(x^k) = \{i \in LI \;\big|\; r_i(x^k) < -\epsilon\} \quad \text{and}$$

$$I_\epsilon^{AE}(x^k) = \{i \in LE \;\big|\; |\, r_i(x^k)| \leqslant \epsilon\} = \{i_{t'_k+1}, \ldots, i_{t''_k}\},$$

and the vector

$$\tilde{\nabla} p_k = \mu \tilde{\nabla} f_k - \sum_{i\in I_\epsilon^{VI}(x^k)} a_i + \sum_{i\in LE\setminus I_\epsilon^{AE}(x^k)} \operatorname{sgn}\,[r_i(x^k)]a_i\,.$$

Redefine the matrix $\mathbf{A}(x^k)$, the vector $r(x^k)$ and the scalar r_k^∞ as

$$\mathbf{A}(x^k) = [A_{i_1} \ldots, A_{i_{t_k}} a_{i_{t_k+1}} \ldots, a_{i_{t'_k}} a_{i_{t'_k+1}} \ldots, a_{i_{t''_k}}],$$

$$r(x^k)^T = [r_{i_1}(x^k)^T, \ldots, r_{i_{t_k}}(x^k)^T, r_{i_{t_k+1}}(x^k), \ldots, r_{i_{t'_k}}(x^k), r_{i_{t'_k+1}}(x^k), \ldots, r_{i_{t''_k}}(x^k)],$$

$$r_k^\infty = \max_{j=1,\ldots,t''_k}(\|\, r_{i_j}(x^k)\|\,).$$

Now, if we replace $\tilde{\nabla} f_k$ by $\tilde{\nabla} p_k$ and $f(\cdot)$ by $p(\cdot,\mu)$, then our algorithm can be used to minimize $p(x,\mu)$ over x if we make the following changes to steps 6 and 7:

6) Decide whether or not to drop an activity:

IF ($\|\hat{u}_i\| \leqslant 1 \;\; \forall i \in I_{\epsilon_{\text{ACT}}}(x^k)$ .AND.

$0 \leqslant \hat{u}_i \leqslant 1 \;\; \forall i \in I_\epsilon^{\text{AI}}(x^k)$) THEN go to 8.

7) Drop an activity:

Choose $l \in I_{\epsilon_{\mathrm{ACT}}}(x^k)$ with $\| \hat{u}_j \| > 1$ and set $d^k = -P_{k/l} A_l \hat{u}_l$

.OR. Choose $l \in I_{\epsilon}^{\mathrm{AI}}(x^k)$ with $\hat{u}_l < 0$ and set $d^k = P_{k/l} a_l$.

$$\text{Set } g \leftarrow \begin{cases} \tilde{\nabla} f_k - A_l \hat{u}_l / \| \hat{u}_l \|, & l \in I_0(x^k), \\ \tilde{\nabla} f_k + A_l r_l / \| r_l \|, & l \in I_{\epsilon_{\mathrm{ACT}}}(x^k) \backslash I_0(x^k), \\ \tilde{\nabla} f_k, & l \in I_{\epsilon}^{\mathrm{AI}}, \end{cases}$$

go to 10.

(It can be shown that if there exists an index $l \in I_{\epsilon}^{\mathrm{AI}}(x^k)$ with $\hat{u}_l > 1$ or an index $l \in I_{\epsilon}^{\mathrm{AE}}(x^k)$ with $|\hat{u}_l| > 1$ then the direction $d = -\mathrm{sgn}(\hat{u}_l) P_{k/l} a_l$ is a descent direction for the penalty function. However, as pointed out by Bartels and Conn [1], constraint r_{j_l} will be violated at $x^k + \lambda d^k$ and, rather than do this, we take an alternative descent direction if one exists. Otherwise, we reduce μ in order to give more emphasis to the constraints.)

9. Implementation. Throughout this derivation it was assumed that the matrix $\mathbf{A}(x)$ was full rank in order to uniquely define the solution to various equations. This can no longer be accomplished when $\mathbf{A}(x)$ is rank deficient because of the inherent degeneracies encountered. Fortunately, this difficulty can be resolved in a straightforward manner computationally by using a method similar to Bartels et al. [2]. This involves treating all degeneracies as if they occurred due to the error introduced through the storage of data on a finite machine. Then the machine accuracy can be artificially reduced and the problem perturbed so that the degeneracies are resolved. Finally, after the perturbation, a unique solution can be attained which satisfies the original problem.

In particular, we do the following. After identifying the current active set, we randomly perturb all residuals associated with degenerate members of this set in such a way that those terms are no longer considered active. We then adjust the restricted gradient appropriately and proceed with the original algorithm. In this way, we either leave this degenerate neighbourhood, identify a solution among these perturbed vertices or enter a new degenerate neighbourhood. It should be noted that cycling cannot occur.

The full rank matrix $\mathbf{A}(x^k)$ can be factored, by forming a product of Givens transformations, into the form

$$\mathbf{A}(x^k) = Q^k \begin{bmatrix} R^k \\ 0 \end{bmatrix} = [Q_1^k \quad Q_2^k] \begin{bmatrix} R^k \\ 0 \end{bmatrix}, \tag{9.1}$$

where Q^k is an orthonormal matrix and R^k is a nonsingular upper-triangular matrix. This numerically stable decomposition can easily be modified to accommodate the necessary additions and deletions of column vectors to the matrix $\mathbf{A}$ when it is updated (see [12]).

Since the columns of Q_2^k form an orthogonal basis for the space $S_k^{\perp}$, the projection matrix P_k can be computed as

$$P_k = (Q_2^k)(Q_2^k)^T, \tag{9.2}$$

In addition, the vector $\hat{u}$ required in step 5 of the minimization algorithm can be obtained efficiently by solving

$$R^k\hat{u} = (Q_1^k)^T\tilde{\nabla}f_k, \tag{9.3}$$

whereas the vector v^k required in step 10 can easily be obtained by first solving

$$(R^k)^T\bar{v}^k = r(x^k) \tag{9.4}$$

and then forming

$$v^k = Q_1^k\bar{v}^k. \tag{9.5}$$

10. Preliminary computational experience. Six small problems were run on a Honeywell 66/60 computer to compare the performance of this projection technique with other approaches. The problems used were chosen because of the availability of comparison data. The other solution methods considered are as follows:

i) Hyperbolic Approximation Procedure (HAP)

ii) Modified HAP (MHAP)

iii) The program of Calamai and Charalambous (MFLPVI)

A description of these methods and references to the data used in the six problems are given in [3]. A summary of the results is given below. The column labeled "NEW" refers to the projection algorithm.

Table 1
Comparative test results

#	MFLPVI	$\epsilon^{(0)} = 10^0$		$\epsilon^{(0)} = 10^{-4}$		NEW
		HAP	MHAP	HAP	MHAP	
1	77+487	1661	1381	2027	1407	64
2	34+114	647	546	4641	2281	17
3	16+47	87	70	770	197	8
4	15+16	45	45	45	45	17
5	40+183	142	114	1763	975	26
6	7+56	242	164	3743	1869	18
TOTAL	1092	2824	2320	12989	6774	150
OPS	112248	386592	318426	1647886	922390	77194

Remarks.

(a) Except for the last row, the figures in the table refer to the number of iterations required to reach the solution. Under the column heading MFLPV1, the first figure refers to the number of successful iterations whereas the second figure refers to the number of unsuccessful iterations.

(b) For our algorithm no attempt was made to choose ideal parameters. The choice, for the results shown, was $\epsilon_{ACT} = 0.1$, $\epsilon_{ZP1} = 0.2$, $\epsilon_{ZP2} = 0.05$ and $\epsilon = 7.45\text{E}{-9}$.

(c) The last row in the table gives an estimate of the total number of addition operations required by each of the techniques in solving the six problems. With the exception of the MFLPV1 approach approximately the same number of multiplications would be required. There are usually more addition than multiplication operations in MFLPV1 due to combinatorial situations which sometimes arise. (For these six problems there are approximately 17 percent more addition than multiplication operations in MFLPV1.)

(d) The following are formulas for evaluating the number of addition operations per iteration for the four techniques (with insignificant terms neglected).

$$\text{HAP and MHAP } \{\ 6\tau + 2n;$$

$$\text{MFLPV1 } \begin{cases} v & \text{if unsuccessful,} \\ v + 3\tau + 4n + 2m & \text{if successful;} \end{cases}$$

$$\text{NEW } \begin{cases} w & \text{if §7 step 4 satisfied,} \\ w + 2t(2t+1) + 8nt & \text{if §7 step 6 satisfied,} \\ w + t(2t+1) + 8n(n-t) + 4(nt+1) & \text{if §7 step 6 not satisfied;} \end{cases}$$

where

$$w = 3\tau + 4(\tau - t) + 16(n - t)^2 + 8t^2 + 2n,$$

$$v = \begin{cases} 4\{\sum_{z \in Z} \sum_{k=2}^{z-1} k\binom{z}{k} + \sum_{k=2}^{z''} k\binom{z'}{k}\} & \text{for combinatorial case,} \\ \tau + 5n + 3m & \text{otherwise;} \end{cases}$$

and (see [3]):

$Z \triangleq$ set containing cardinalities of all clusters unsuccessfully tested,

$z' \triangleq$ cardinality of cluster successfully tested,

$z'' \triangleq$ cardinality of subset moved ($z'' \leqslant z' \leqslant n-1$).

(e) The results and the theory suggest that the new techniques will work well with larger and/or more difficult problems.

11. Conclusions and recommendations.

(a) The purpose of this paper is to provide a simple stable method for solving the multifacility location problem involving Euclidean distances; however, the same ideas can be, and perhaps should be, applied to location problems that use other distance measures (l_1 and l_∞ for example). This would provide a more unified approach to solving this class of problems.

(b) The method derived for handling linear constraints can be modified to handle any continuous functions mapping $\mathbf{R}^{2n} \to \mathbf{R}^1$ by considering local linearizations of those functions.

(c) Instead of this method that essentially uses steepest descent in a subspace, a more sophisticated approach would involve the use of second-order information to derive quasi-Newton steps in a subspace. Such a technique would undoubtedly lead

to an improvement in the rate of convergence. See [6].

(d) For large problems the sparsity and structure of the matrix $\mathbf{A}(x)$ would have to be given attention. This would suggest the use of decomposition techniques as well as the use of methods that maintain sparsity in updating these decompositions whenever columns were added to or deleted from $\mathbf{A}$.

(e) The stepsize algorithm outlined in this paper is not necessarily optimal. Many alternative schemes are available, including:

1) the reordering of breakpoints based on the magnitude of the least-squares error in their estimation,

2) the use of a more exact cubic line search either with or without the breakpoint analysis,

3) the use of a more sophisticated steplength algorithm. See [14].

(f) Step 7 of our algorithm results in the release of an index, associated with an "out-of-kilter" dual, from the activity set. In our implementation we drop the index (and thus the corresponding activity) associated with the first "out-of-kilter" dual encountered. This choice may not result in the optimal descent direction.

All these ideas are currently being investigated.

Appendix. *Prove:*

$$A_l^T P_{k/l} A_l = \rho I, \qquad \rho > 0,$$

where $P_{k/l}$ *is the orthogonal projector onto* $S_{k/l}^{\perp}$ *where* $S_{k/l}$ *is the space spanned by the columns of* $\bar{\mathbf{A}}$. ($\bar{\mathbf{A}}$ is formed by deleting the columns of A_l from the full rank matrix $\mathbf{A}$.)

Proof. First we prove that $b^T W b = \xi > 0$, where b is any n-vector outside the span of B, B is any $n \times t$ full rank matrix and W is the $n \times n$ orthogonal projector onto $B^{\perp}$. Since B is full rank, there exists an $n \times n$ orthogonal matrix Q and a $t \times t$ nonsingular upper-triangular matrix R such that

$$B = [Q_1 Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R,$$

where Q_1 denotes the first t columns and Q_2 denotes the remaining $n-t$ columns of Q. It can then be shown that $W = Q_2 Q_2^T$. Thus

$$b^T W b = (Q_2^T b)^T (Q_2^T b) = \xi > 0.$$

Note that $b^T W b = 0 \Rightarrow Q_2^T b = 0 \Rightarrow$ there exists a $z \in \mathbf{R}^t$ s.t. $b = Q_1 z$

$\Rightarrow b = B\tilde{z}$ where $\tilde{z} = R^{-1} z$

$\Rightarrow b \in \text{span}(B)$

$\Rightarrow$ contradiction.

Now define the transformation $T: \mathbf{R}^{\mu \times \nu} \rightarrow \mathbf{R}^{2\mu \times 2\nu}$ as follows:

$$T \begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix} = \begin{bmatrix} \beta_{11} I & \cdots & \beta_{1\nu} I \\ \vdots & & \vdots \\ \beta_{\mu 1} I & \cdots & \beta_{\mu\nu} I \end{bmatrix}$$

where $\beta_{ij} \in \mathbf{R}$ for $i=1, \ldots, \mu$ and $j=1, \ldots, \nu$ and I represents the standard 2×2 identity matrix. Obviously there exists a vector b and a matrix B such that $A_l = T(b)$ and $\bar{\mathbf{A}} = T(B)$.

Finally, we note that whenever the product XY is defined then $T(XY) = T(X)T(Y)$ and similiarly, whenever the sum $X+Y$ is defined then $T(X+Y) = T(X)+T(Y)$. Furthermore, if U is an upper-triangular matrix then $T(U)$ is also upper-triangular and $T(I) = I$. Thus, if we apply this transformation to the foregoing proof we obtain

$$A_l^T P_{k/l} A_l = \rho I \quad \text{with } \rho > 0.$$

Acknowledgement. The splendid typesetting was accomplished by Anne Tripde-Roche and Brian Finch using the Photon Econosetter and Troff.

REFERENCES

[1] R. H. Bartels and A. R. Conn, *Linearly constrained discrete l_1 problems,* ACM TOMS, 6 (1980), pp. 594-608.

[2] R. H. 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.

[3] P. H. Calamai, and C. Charalambous, *Solving multifacility location problems involving Euclidean distances,* Naval Res. Log. Quart., to appear.

[4] C. Charalambous and A. R. Conn, *An efficient method to solve the minimax problem directly,* SIAM J. Numer. Anal., 15 (1978), pp. 162-187.

[5] J. A. Chatelon, D. W. Hearn, and T. J. Lowe, *A subgradient algorithm for certain minimax and minisum problems,* Math. Prog., 15 (1978), pp. 130-145.

[6] T. F. Coleman and A. R. Conn, *Nonlinear programming via an exact penalty function method: A global method,* Comp. Sci. Tech. Rep. CS-80-31, University of Waterloo, Waterloo, Ontario, 1980 (to appear in Math. Prog.).

[7] A. R. Conn, *Constrained optimization using a nondifferentiable penalty function,* SIAM J. Numer. Anal., 10 (1973), pp. 760-784.

[8] A. R. Conn, and T. Pietrzykowski, *A penalty function method converging directly to a constrained optimum,* SIAM J. Numer. Anal., 14 (1977), pp. 348-375.

[9] J. W. Eyster, J. A. White, and W. W. Wierwille, *On solving multifacility location problems using a hyperboloid approximation procedure,* AIIE Trans., 5 (1973), pp. 1-6.

[10] R. L. Francis and A. V. Cabot, *Properties of a multifacility location problem involving Euclidean distances,* Naval Res. Log. Quart., 19 (1972), pp. 335-353.

[11] R. L. Francis and J. M. Goldstein, *Location theory: A selective bibliography,* Oper. Res., 22 (1974), pp. 400-410.

[12] P. E. Gill, G. H. Golub, W. Murray and M. Saunders, *Methods for modifying matrix factorizations,* Math. Comp., 28 (1974), pp. 505-535.

[13] R. F. Love, *Locating facilities in three-dimensional space by convex programming,* Naval Res. Log. Quart., 16 (1969), pp. 503-516.

[14] W. Murray and M. Overton, *Steplength algorithms for minimizing a class of non-differentiable functions,* Computing, 23 (1979), pp. 309-331.

[15] T. Pietrzykowski, *An exact potential method for constrained maxima,* SIAM J. Numer. Anal., 6 (1969), pp. 299-304.

[16] R. C. Vergin and J. D. Rogers, *An algorithm and computational procedure for locating economic facilities,* Management Sci., 13 (1967), pp. 240-254.

[17] G. O. Wesolowsky and W. G. Truscott, *Dynamic location of multiple facilities for a class of continuous problems,* AIIE Trans., 8 (1976), pp. 76-83.

[18] W. I. Zangwill, *Nonlinear programming via penalty functions*, Management Sci., 13 (1962), pp. 344-350.