

A Fast Cauchy-Riemann Solver

By Michael Ghil* and Ramesh Balgovid**

Abstract. We present a solution algorithm for a second-order accurate discrete form of the inhomogeneous Cauchy-Riemann equations. The algorithm is comparable in speed and storage requirements with fast Poisson solvers. Error estimates for the discrete approximation of sufficiently smooth solutions of the problem are established; numerical results indicate that second-order accuracy obtains even for solutions which do not have the required smoothness. Different combinations of boundary conditions are considered and suitable modifications of the solution algorithm are described and implemented.

1. Introduction. Inhomogeneous Cauchy-Riemann equations appear naturally in many fluid-dynamical problems, as the divergence and the vorticity equations of a two-dimensional steady flow field $(u, v) = (u(x, y), v(x, y))$. The velocity components u, v are usually called in this context *primitive* variables, in contradistinction to the derived variables ψ, ζ in the stream function-vorticity formulation of the flow equations (e.g., Roache [28]). In the latter formulation, the stream function ψ satisfies a Poisson equation; and computations with this formulation have greatly benefited from the rapid development of fast direct methods for the solution of Poisson's equation, or Poisson solvers (Buneman [4], Buzbee, Golub and Nielson [5], Dorr [9], Fischer, Golub, Hald, Leiva and Widlund [12], Golub [18], Hockney [21], [22], Widlund [32]).

Working in the primitive variables, however, permits the treatment of more general flows. Indeed, either nondivergence or irrotationality of the flow are required in order to introduce a stream function ψ or a velocity potential ϕ , and obtain a Poisson equation for them. There are many situations of practical interest in which neither of these assumptions holds. Furthermore, the formulation of boundary conditions is often easier in terms of the primitive variables, by using physical considerations which arise naturally from the problem. On the other hand, a boundary condition on the vorticity ζ for instance is at times hard to formulate (Langlois [23]); the construction of appropriate *discrete* versions of such a boundary condition is often even more difficult (Olinger and Sundström [27]). Hence, the desirability of simple, physically meaningful boundary conditions and, thus, of the use of primitive variables.

Lomax and Martin [24] have developed a fast Cauchy-Riemann solver and

Received April 10, 1978.

AMS (MOS) subject classifications (1970). Primary 65F05, 65N15, 65N20; Secondary 65N04, 65N05, 76B05, 86A10.

Key words and phrases. Fast direct solvers, Cauchy-Riemann equations, elliptic first-order systems, transonic flow.

*The work of this author was supported in part by NASA, Grant No. NSG-5130.

**The work of this author was supported in part by NASA, Grant No. NSG-5034.

© 1979 American Mathematical Society
0025-5718/79/0000-0058/\$13.75

applied it to a quasilinear problem in aerodynamics ([24], [25]). Additional versions of their solver are given in [26]. Our interest in the problem stems from a different application, dealing with a two-dimensional version of the equations of dynamic meteorology (Ghil [14], [15]). The solver we present also differs from those of [24], [26] in a number of ways: the boundary conditions and their numerical treatment, the decoupling of u , v and the reduction to a discrete Poisson problem, and finally the method of solution of the resulting discrete Poisson equation are all different. We substantiate by numerical results that the present solver is second-order accurate, even for solutions which do not have the formally required degree of smoothness. The solvers of [24], [26] seem to be only first-order accurate according to our numerical tests.

The intended application of our solver is to a fully nonlinear first-order system, rather than a quasilinear one. This system is a generalization of a Monge-Ampère equation [15], [17]. Eventually, we hope to apply the solver to cases where the nonlinear equations are of mixed elliptic-hyperbolic type [16], [17]. Preliminary results are encouraging, and we expect to pursue the nonlinear problem in a future publication.

The organization of the article is the following: Section 2 contains the description in continuous and then in discrete form of the model problem of which we seek a fast solution. Section 3 contains the derivation and description of the solution algorithm. Section 4 presents numerical results for test computations with the model problem. Section 5 presents modifications of the model problem arising from changes in boundary conditions. Section 6 contains a comparison of results with the solvers of [24], [26]. Section 7 gives conclusions and a discussion of possible extensions and generalizations. Finally, Appendix A presents an error estimate for the method, and Appendix B contains a listing of the basic program.

Acknowledgements. It is gratifying to acknowledge useful discussions with Professors Eugene Isaacson and Olof Widlund. Numerical calculations were performed in part on the CDC 6600 of the Courant Mathematics and Computing Laboratory, New York University under Contract EY-76-C-02-3077 with the U. S. Department of Energy.

2. The Model Problem.

The Differential Equations. We wish to study the fast numerical solution of the elliptic system of two first-order linear equations in two independent variables,

$$(2.1a) \quad u_x + v_y = d(x, y),$$

$$(2.1b) \quad u_y - v_x = e(x, y).$$

The dependent variables u , v can be thought of as velocity components in the x , y directions, respectively. In this interpretation d , e are the divergence and vorticity of the flow, which are assumed to be known. If $d \equiv 0 \equiv e$, (2.1) are the Cauchy-Riemann equations, and u , v are analytic. We are interested in the inhomogeneous case, $|d| + |e| \neq 0$, and we concentrate on real-valued d , e , u , v , although the method is applicable with minor changes to complex-valued functions as well.

We consider a rectangular domain R , taken without loss of generality to be $R = \{(x, y): 0 < x < 2\pi, 0 < y < \pi\}$. The boundary conditions are that v is given on the

lower and upper side of the rectangle,

$$(2.2a) \quad v = v^{(0)}(x), \quad y = 0,$$

$$(2.2b) \quad v = v^{(1)}(x), \quad y = \pi,$$

and that both u and v are periodic in the x -direction,

$$(2.3a) \quad u(x + 2\pi, y) = u(x, y),$$

$$(2.3b) \quad v(x + 2\pi, y) = v(x, y).$$

The Gauss divergence theorem implies that $d(x, y)$, $v^{(0)}(x)$, and $v^{(1)}(x)$ have to satisfy

$$(2.3c) \quad \int_0^{2\pi} \int_0^\pi d(x, y) dx dy = \int_0^{2\pi} \{v^{(1)}(x) - v^{(0)}(x)\} dx.$$

These conditions, together with (2.1), determine v completely and u up to an additive constant (compare Ghil [15]). The latter indeterminacy in the solution can be eliminated, for instance, by prescribing u at one arbitrary point (x_0, y_0) in the rectangle.

The boundary conditions we use are associated with a standard channel-flow problem in geophysical fluid dynamics (e.g., Elvius and Sundström [10], Gustafsson [20]), from which the nonlinear problem mentioned in Section 1 is derived (Ghil [15], [17]). Extensions to different boundary conditions and to irregular domains will be discussed in Sections 5 and 7.

The Difference Equations. The discretization of the problem we chose is to approximate the derivatives in (2.1) by finite differences. Let U, V, D, E stand for the mesh functions which approximate the continuous functions u, v, d, e ; and let h, k stand for the mesh size in the x, y -directions, respectively. It is natural to use centered differences to replace the corresponding derivatives in (2.1). We write

$$(\Delta) \quad \begin{aligned} U(x + \delta, y) - U(x - \delta, y) &\cong 2\delta u_x(x, y), \\ U(x, y + \epsilon) - U(x, y - \epsilon) &\cong 2\epsilon u_y(x, y), \end{aligned}$$

for u and similar formulas for v ; this yields a second-order accurate approximation of the derivatives and allows us to expect that in some adequate norm $\|\cdot\|$,

$$\|U - u\| + \|V - v\| = O(h^2) + O(k^2).$$

The use of centered differences in a straightforward manner, on an unstaggered mesh $x_i = ih, y_j = jk$, leads, however, to the existence of spurious null vectors (U, V) , i.e., to zero eigenvectors of the discrete matrix operator which approximates the differential Cauchy-Riemann operator. To avoid dealing with these null vectors and to obtain an invertible discrete matrix operator, we used a staggered mesh (Figure 1). Such a mesh, suggested already by Lomax and Martin [24], can be formulated for Eqs. (2.1) in a particularly efficient way.

Let \mathbf{u}, \mathbf{v} denote points at which U, V are defined, and let \cdot, \times (for the vector operators divergence and curl) denote the points at which the discrete versions (2.4a, b) (see below) of equations (2.1a, b) are written and at which D, E are defined. Thus,

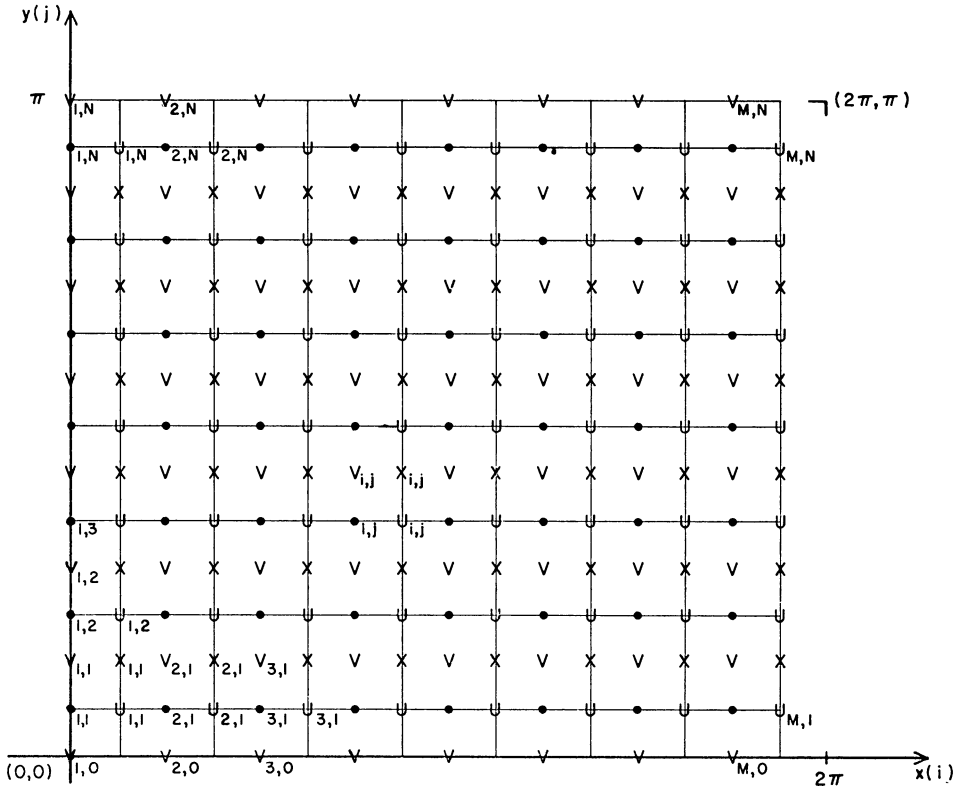


FIGURE 1

u, v alternate on diagonals of the mesh and so do \cdot, \times on diagonals parallel to and alternating with those of u, v ; in other words, v, \times, u, \cdot , in clockwise direction, occupy the corners of an elementary mesh cell of area $hk/4$ (see Figure 1). No averaging of U, V is necessary in writing (2.4a, b) on this staggered mesh. Furthermore, the boundary conditions (2.2) and the periodicity condition (2.3a, b) can be easily handled. Indeed, let U, V be indexed independently, with $y = 0, \pi$ being horizontal V -lines corresponding to V -indices $j = 0, N$, and with $x = 0, 2\pi$ being vertical V -lines with V -indices $i = 1, M + 1$. Then the computational domain includes the points $((i - 1)h, jk)$, at which V_{ij} is defined, and the points $((i - 1/2)h, (j - 1/2)k)$, at which U_{ij} is defined; in other words, V_{ij} approximates $v((i - 1)h, jk)$, $1 \leq i \leq M + 1, 0 \leq j \leq N$, and U_{ij} approximates $u((i - 1/2)h, (j - 1/2)k)$, $1 \leq i \leq M, 1 \leq j \leq N$, with $h = 2\pi/M, k = \pi/N$.

The discrete equations are centered:

$$(2.4a) \quad (U_{i,j} - U_{i-1,j})/h + (V_{i,j} - V_{i,j-1})/k = D_{ij}, \quad 1 \leq i \leq M, 1 \leq j \leq N,$$

$$(2.4b) \quad (U_{i,j+1} - U_{i,j})/k - (V_{i+1,j} - V_{i,j})/h = E_{ij}, \quad 1 \leq i \leq M, 1 \leq j \leq N - 1;$$

here D_{ij} approximates $d((i - 1)h, (j - 1/2)k)$, and E_{ij} approximates $e((i - 1/2)h, jk)$.

Notice that δ, ϵ in (Δ) have been replaced after staggering by $h/2, k/2$, rather than by h, k . The boundary conditions become

$$(2.5a) \quad V_{i,0} = v^{(0)}((i-1)h), \quad 1 \leq i \leq M,$$

$$(2.5b) \quad V_{i,N} = v^{(1)}((i-1)h), \quad 1 \leq i \leq M,$$

and the periodicity condition becomes

$$(2.6a) \quad U_{0,j} = U_{M,j}, \quad 1 \leq j \leq N,$$

$$(2.6b) \quad V_{1,j} = V_{M+1,j}, \quad 0 \leq j \leq N.$$

Conditions (2.5), (2.6) leave $M(2N-1)$ unknowns, to wit, $U_{ij}, 1 \leq i \leq M, 1 \leq j \leq N$, and $V_{ij}, 1 \leq i \leq M, 1 \leq j \leq N-1$, while (2.4) yields $M(2N-1)$ linear algebraic equations for them. We should expect from the situation in the continuous problem that the matrix of the linear system (2.4) has a one-dimensional null space. The resulting indeterminacy can be eliminated either by prescribing the value of U_{ij} at an arbitrary index (i_0, j_0) , or by some alternative procedure which will arise naturally in Section 3; the necessary discrete compatibility condition will also be discussed there.

3. Solution Algorithm. Our plan shall be to rewrite (2.4)–(2.6) in a convenient block matrix form, to eliminate V , then to bring the remaining block matrix operating on U to the form of a discrete five-point Laplacian, and finally to formulate a fast direct algorithm to solve for U . After this, V obtains from U by a straightforward, fast computation.

Block Matrix Equation. We start by rewriting the discrete linear system (2.4) in block matrix form. Let $\mathbf{U}_j = (U_{1,j}, U_{2,j}, \dots, U_{M,j})^*$, $\mathbf{V}_j = (V_{1,j}, V_{2,j}, \dots, V_{M,j})^*$, where $()^*$ denotes (conjugate) transpose, so that $\mathbf{U}_j, \mathbf{V}_j$ are column vectors corresponding to a horizontal mesh line. \mathbf{D}_j and \mathbf{E}_j are introduced in similar fashion. Also let $\rho = k/h$. With this notation, (2.4) can be written as

$$(3.1a) \quad \mathbf{V}_j = \mathbf{V}_{j-1} - T\mathbf{U}_j + k\mathbf{D}_j, \quad 1 \leq j \leq N,$$

$$(3.1b) \quad \mathbf{U}_{j+1} = \mathbf{U}_j - T^*\mathbf{V}_j + k\mathbf{E}_j, \quad 1 \leq j \leq N-1;$$

here $\rho^{-1}T$ is the familiar *backward difference operator*,

$$(3.2) \quad T = \rho \begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & -1 & 1 \end{bmatrix}_{M \times M},$$

acting on the M -periodic vectors $\mathbf{U}_j, \mathbf{V}_j$ and $-\rho^{-1}T^*$ is the corresponding *forward difference operator*.

From (2.5) we have that V_0 and V_N are known; accordingly, we redefine D_1 as $D_1 + k^{-1}V_0$ in the first one, and D_N as $D_N - k^{-1}V_N$ in the last one of the equations (3.1a). After these changes of notation and a change of order, the vector equations (3.1) can be put into the block matrix form

$$(3.3) \quad \begin{matrix} & & \overbrace{\hspace{2cm}}^{N} & & \overbrace{\hspace{2cm}}^{N-1} & & \\ & & \left[\begin{array}{cc|cc} -I & I & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot \\ & & & & -I & I & & \\ \hline T & & & & & T^* & & \\ & \cdot & & & -I & \cdot & & \\ & & & & \cdot & \cdot & & \\ & & & & & & & I \\ & & & & & & & -I \\ & & & & & & & T \end{array} \right] & \left[\begin{array}{c} U_1 \\ \cdot \\ \cdot \\ \cdot \\ U_{N-1} \\ \hline U_N \\ \hline V_1 \\ \cdot \\ \cdot \\ \cdot \\ V_{N-1} \end{array} \right] & = k & \left[\begin{array}{c} E_1 \\ \cdot \\ \cdot \\ \cdot \\ \hline E_{N-1} \\ \hline D_1 \\ \cdot \\ \cdot \\ \cdot \\ D_N \end{array} \right], \end{matrix}$$

where I is the $M \times M$ identity matrix. This form is roughly analogous to writing (2.1a, b) in reversed order,

$$(2.1') \quad \begin{bmatrix} \partial/\partial y & -\partial/\partial x \\ \partial/\partial x & \partial/\partial y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} e \\ d \end{bmatrix}.$$

We notice that the matrix in (3.3) has only four nonzero diagonals of $M \times M$ blocks and that the blocks are at most scalar tridiagonal. Furthermore, all the non-zero entries are ± 1 or $\pm \rho$. In the present form, however, we cannot take full advantage of the extreme sparsity and simplicity of the matrix in order to obtain a solution method comparable to fast Poisson solvers.

Before bringing (3.3) to a more advantageous form we remark that its matrix indeed has a one-dimensional null space: the sum of the MN rows in the lower half of the matrix is zero. The corresponding compatibility condition that $\sum_{ij} D_{ij} = 0$ is the discrete counterpart of (2.3c).

Decoupling of U and V . It is a well-known fact that each of the functions u, v which satisfy (2.1) will also satisfy a Poisson equation obtained from (2.1) by eliminating the other dependent variable by cross-differentiation. This suggests the attempt to eliminate V in system (3.3) in order to obtain a linear algebraic system for U alone. The matrix of this system will be similar to a discrete Laplacian, and to it we shall be able to apply fast solution techniques.

The elimination of V proceeds as follows. In system (3.3) add the N th block row to the $(N + 1)$ st, then the $(N + 1)$ st to the $(N + 2)$ nd and so on. This discrete summation procedure is analogous to integration with respect to y in (2.1a). The lower part of the system thus becomes

$$(3.4) \quad N \left\{ \begin{array}{c} \overbrace{\begin{bmatrix} T & & & \\ T & T & & \\ \vdots & \ddots & \ddots & \\ T & \cdots & T & T \end{bmatrix}}^N \quad \overbrace{\begin{bmatrix} I & & & \\ 0 & \cdot & & \\ & \ddots & \ddots & I \\ & & & 0 \end{bmatrix}}^{N-1} \end{array} \right\} \begin{bmatrix} \mathbf{U}_N \\ \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_{N-1} \end{bmatrix} = k \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{D}_1 + \mathbf{D}_2 \\ \vdots \\ \sum_1^N \mathbf{D}_j \end{bmatrix}.$$

After this transformation, (3.3) can be written as

$$(3.5) \quad \begin{array}{c} N-1 \\ N-1 \\ 1 \end{array} \left\{ \begin{array}{c} \overbrace{\begin{bmatrix} P & Q \\ \vdots & \vdots \\ R & I_{N-1} \\ T \cdots T & 0 \cdots 0 \end{bmatrix}}^N \quad \overbrace{\begin{bmatrix} Q \\ \vdots \\ I_{N-1} \\ 0 \cdots 0 \end{bmatrix}}^{N-1} \end{array} \right\} \begin{bmatrix} \mathbf{U} \\ \vdots \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{E}' \\ \vdots \\ \mathbf{D}' \\ k \sum_1^N \mathbf{D}_j \end{bmatrix};$$

here $\mathbf{U} = (\mathbf{U}_1^*, \dots, \mathbf{U}_N^*)^*$, $\mathbf{V} = (\mathbf{V}_1^*, \dots, \mathbf{V}_{N-1}^*)^*$, I_{N-1} is the $M(N-1) \times M(N-1)$ identity matrix, the definitions of P , Q and \mathbf{E}' are easily read from Eqs. (3.3), while those of R and \mathbf{D}' follow from (3.4). In particular, note that \mathbf{D}' and \mathbf{E}' contain the factor k .

From (3.5) it follows that

$$(3.6a) \quad \mathbf{PU} + \mathbf{QV} = \mathbf{E}',$$

$$(3.6b) \quad \mathbf{RU} + \mathbf{V} = \mathbf{D}',$$

where we omit for the moment the last vector equation. Substituting \mathbf{V} from (3.6b) into (3.6a), we obtain

$$(3.7) \quad (\mathbf{P} - \mathbf{QR})\mathbf{U} = \mathbf{E}' - \mathbf{QD}',$$

where

$$(3.8) \quad \mathbf{QR} = \left. \begin{array}{c} \overbrace{\begin{bmatrix} TT^* & & & \\ \vdots & \cdot & & \\ TT^* & \cdots & TT^* & 0 \end{bmatrix}}^N \right\}^{N-1},$$

since Q is block diagonal with constant equal blocks and $T^*T = TT^*$. Attaching now the last equation of (3.5) to (3.7) yields

$$(3.9) \quad \begin{bmatrix} \mathbf{P} - \mathbf{QR} \\ \vdots \\ T \cdots T \end{bmatrix} \mathbf{U} = \begin{bmatrix} \mathbf{E}' - \mathbf{QD}' \\ \vdots \\ k \sum_1^N \mathbf{D}_j \end{bmatrix}$$

or, changing sign in all but the last equation,

$$(3.9') \quad \left\{ \begin{bmatrix} I & -I & & & & \\ & I & \cdot & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & I & -I \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 \end{bmatrix} + \begin{bmatrix} TT^* & & & & & \\ & \cdot & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ TT^* & & & & TT^* & \\ T & \cdot & \cdot & \cdot & \cdot & T \end{bmatrix} \right\} \mathbf{U}$$

$$= k \begin{bmatrix} T^* \mathbf{D}_1 - \mathbf{E}_1 \\ T^*(\mathbf{D}_1 + \mathbf{D}_2) - \mathbf{E}_2 \\ \vdots \\ T^* \sum_1^{N-1} \mathbf{D}_j - \mathbf{E}_{N-1} \\ \sum_1^N \mathbf{D}_j \end{bmatrix}.$$

Let $S = TT^*$. Notice that

$$(3.10) \quad S = \rho(T + T^*)$$

or, in the familiar notation for tridiagonal matrices,

$$(3.10') \quad T = \rho(-1, 1, 0), \quad T^* = \rho(0, 1, -1), \quad S = \rho^2(-1, 2, -1);$$

this is a slight extension of standard notation: because of periodicity these matrices are actually circulants, rather than tridiagonal proper (cf. (3.2), (3.10)). Thus, S corresponds to a central second difference, carrying further the analogy with the continuous case. However, (3.9') is still not in a form suitable for fast solution.

Discrete Poisson Equation for U. This form is now easily achieved. First, multiply the last block row by T^* . Then subtract the second block row from the first, the third from the second, and so on up to and including the last one; the new last row is obtained by adding all the new rows, from 1 to $(N - 1)$, to the last row. Finally, the new last row is taken as the first row of the system and all the signs are reversed, yielding

$$(3.11) \quad \begin{bmatrix} S + I & -I & & & & \\ -I & S + 2I & -I & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & -I & S + 2I & -I \\ & & & & -I & S + I \end{bmatrix} \mathbf{U}$$

$$= k \begin{bmatrix} T^* \mathbf{D}_1 - \mathbf{E}_1 \\ T^* \mathbf{D}_2 + \mathbf{E}_1 - \mathbf{E}_2 \\ \vdots \\ T^* \mathbf{D}_{N-1} + \mathbf{E}_{N-2} - \mathbf{E}_{N-1} \\ T^* \mathbf{D}_N + \mathbf{E}_{N-1} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \cdot \\ \cdot \\ \mathbf{B}_N \end{bmatrix};$$

the vector \mathbf{B} is introduced for future notational convenience.

We notice that the matrix in (3.11) has almost, but not quite the form of a discrete Laplacian (compare for instance Buzbee, Golub and Nielson [5], for Laplacian with Dirichlet, Neumann and periodic boundary conditions). The matrix is symmetric and nonnegative semidefinite; in fact, $U_{ij} = \text{const}$ satisfies the homogeneous system. The system would be positive definite if any one diagonal element actually exceeded the sum of the off-diagonal elements in the same row (e.g., Collatz [6, pp. 43–47]). This immediately suggests that we make use of the possibility of prescribing $U_{i_0, j_0} = u_0$, i.e., that we increase the element at the position $Mj_0 + i_0$ along the diagonal by an arbitrary amount $\alpha > 0$, and add αu_0 to the right-hand side of that equation at the same time. We shall call the system thus modified (3.11 α) for future reference.

The system (3.11 α) could now be solved by modifying any one of the different methods or combination of methods available for Poisson’s equation, as reviewed for instance by Buzbee, Golub and Nielson [5], Dorr [9], or Widlund [32]. In this sense, the elimination of V is analogous to a single step of odd/even reduction as proposed by Hockney [21], [22], although the role of this step is more crucial in the present context. We describe in the sequel the particular fast algorithm chosen to solve (3.11 α).

Fast Direct Solution for U. Essentially, our algorithm is based on the one described in greater generality by Buzbee, Golub and Nielson [5] as *matrix decomposition*. It relies on the fact that the eigenvalues λ_k and eigenvectors ξ_k of S are known, i.e., that

$$S\xi_k = \lambda_k \xi_k,$$

for

$$(3.12a) \quad \lambda_k = 2\rho^2(1 - \cos(2\pi(k - 1)/M)), \quad 1 \leq k \leq M,$$

$$(3.12b) \quad \xi_k = M^{-1/2}(1, e^{-2\pi i(k-1)/M}, \dots, e^{-2\pi i(M-1)(k-1)/M})^*, \quad 1 \leq k \leq M,$$

where i is the imaginary unit.

Our algorithm deviates, however, from general matrix decomposition because the first and last diagonal blocks in (3.11) differ from all the other blocks and because the matrix (3.11), without suitable modification, is singular. For the sake of simplicity, we shall describe the actual algorithm used directly in a self-contained manner.

Let Q be the matrix whose columns are $\xi_1, \xi_2, \dots, \xi_M$, and let $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_M)$. Then, with our normalization of ξ_k ,

$$(3.13) \quad Q^*Q = I, \quad Q^*SQ = \Lambda.$$

Introduce $\tilde{\mathbf{U}}_j, \tilde{\mathbf{B}}_j$ by

$$(3.14a) \quad \mathbf{U}_j = Q\tilde{\mathbf{U}}_j,$$

$$(3.14b) \quad \mathbf{B}_j = Q\tilde{\mathbf{B}}_j, \quad 1 \leq j \leq N,$$

where \mathbf{B}_j are the subvectors on the right-hand side of (3.11).

discussion of (3.11 α), we can deal with the singularity of A_1 by prescribing any component of $\hat{U}_1, \hat{U}_{1,j_0}$ say; this means prescribing $\hat{U}_{1,j_0} = \tilde{U}_{j_0,1} = \sum_{i=1}^M U_{i,j_0}/\sqrt{M}$, rather than one given mesh value U_{i_0,j_0} . Again we call the suitably modified systems (3.15 α), (3.17 α) and (3.18 α).

The LU factorization of the tridiagonal matrices in (3.18 α) is performed next, say $A_k = L_k U_k$, storing the nontrivial entries of L_k and U_k . The diagonal elements of L_k and the superdiagonal elements of U_k are identically 1, while the subdiagonal elements of L_k and the diagonal elements of U_k are reciprocals of each other. Hence only the diagonal elements of U_k in fact need to be stored.

Having computed $\tilde{\mathbf{B}}$ and thus $\hat{\mathbf{B}}$, from the original data, we can solve for $\hat{\mathbf{U}}$ and hence for $\tilde{\mathbf{U}}$. The computation of $\tilde{\mathbf{B}}$ from \mathbf{B} and of \mathbf{U} from $\tilde{\mathbf{U}}$ involves matrix multiplication by Q^* and by Q , respectively. Since these correspond to a direct and an inverse Fourier transform, they can be performed efficiently by a fast Fourier transform (FFT) algorithm. The algorithm used was the one published by Cooley, Lewis and Welch [7], modified to operate in the real.

Summary and Programming Considerations. The solution algorithm developed in this section reduces in practice to the following:

- (i) From the data of the problem \mathbf{D}, \mathbf{E} , compute

$$\begin{aligned} \mathbf{B}_1 &= k(T^* \mathbf{D}_1 - \mathbf{E}_1), \\ \mathbf{B}_j &= k(T^* \mathbf{D}_j + \mathbf{E}_{j-1} - \mathbf{E}_j), \quad 2 \leq j \leq N-1, \\ \mathbf{B}_N &= k(T^* \mathbf{D}_N + \mathbf{E}_{N-1}), \end{aligned}$$

where $\mathbf{D}_1, \mathbf{D}_N$ include the boundary data $\mathbf{V}_0, \mathbf{V}_N$, respectively. The number of operations is $n_1 = 4MN$.*** Notice that, after shifting of the blocks \mathbf{E}_j , \mathbf{B} can overwrite \mathbf{E} , since \mathbf{E} is no longer needed.

- (ii) From the \mathbf{B}_j above compute $\tilde{\mathbf{B}}_j$ (cf. (3.14b)) by

$$\tilde{\mathbf{B}}_j = Q^* \mathbf{B}_j, \quad 1 \leq j \leq N,$$

using an FFT, with $n_2 = 2MN \log_2 M$ real operations. Although FFT algorithms exist which use $MN \log_2 M$ operations for real transforms, we did not feel that in our application actual execution time would be much improved by the utilization of such an algorithm. In this step, $\tilde{\mathbf{B}}$ can be stored in the previous location of \mathbf{B} . Note that $\tilde{\mathbf{B}}_j$ is complex with pairwise complex conjugate components.

(iii) Solve (3.17 α) for the \hat{U}_k by forward elimination and back substitution, using the stored entries of the LU factors. One solution, for fixed k , requires $8N$ real operations, since the complex conjugate components of $\tilde{\mathbf{B}}_j$ enter nonsymmetrically into the LU factorization of (3.17 α), and we have to operate separately on the real and the imaginary parts of $\tilde{\mathbf{B}}$. However, the fact that the components of $\tilde{\mathbf{U}}_j$ are also complex

*** All operation and storage counts are given to highest order. We do not distinguish between addition, multiplication and division, and we do not assume that accumulation of products is particularly efficient. Furthermore, we take $h = k$, or $\rho = 1$.

conjugate allows us to solve only $M/2 + 1$ systems, so that the operation count for this step is $n_3 = 4MN$ real operations. Notice that no reindexing of \mathbf{B} as indicated by (3.16) is actually necessary, and that at this stage $\hat{\mathbf{U}}$ can overwrite $\tilde{\mathbf{B}}$. The shifting of \mathbf{B} elements and \mathbf{U} elements required to use the LU subroutine is relatively fast.

(iv) From the $\hat{\mathbf{U}}_k$ computed in (iii) obtain \mathbf{U} (again with no need for reindexing) by carrying out (3.14a) with another application of the FFT, in $n_4 = 2MN \log_2 M$ real operations, and with \mathbf{U} stored instead of $\hat{\mathbf{U}}$.

(v) Compute \mathbf{V} from \mathbf{U} obtained in (iv), using (3.1a). This requires $n_5 = 4MN$ operations. Notice that in this recursion \mathbf{V} does not depend on $\mathbf{U}_N, \mathbf{D}_N$, since \mathbf{V}_N is known. Using (3.1a) as a backward recursion would yield independence of \mathbf{V} on $\mathbf{U}_1, \mathbf{D}_1$, since \mathbf{V}_0 is also known. This redundancy, however, is only apparent and computations with (3.1a) used as either forward or backward recursion yielded results with the same accuracy, which only depended on the accuracy of \mathbf{U} .

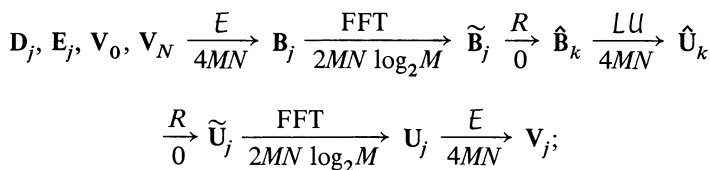
The final operation count is

$$n = \sum_1^5 n_k = 4MN(3 + \log_2 M),$$

to obtain both U and V at all the grid points at which they are defined.

The storage requirement for the L, U factors is $s_1 = MN/2$, using again the fact that we only solve for half the number of components of $\tilde{\mathbf{U}}$; the storage requirement for the right-hand sides \mathbf{D}, \mathbf{E} , is $s_2 = 2MN$; and these locations are successively overwritten until \mathbf{U}, \mathbf{V} are stored in them. Hence, the total storage requirement is $s = s_1 + s_2 = 5MN/2$.

We conclude this section with a diagram of the algorithm:



here E stands for evaluations by linear recursion, FFT for an application of the Fast Fourier Transform, R for reindexing, and LU for factorization. The ranges of the indices are $1 \leq j \leq N, 1 \leq k \leq M$, and the operation counts are given under the corresponding step of the algorithm.

4. Numerical Results. In this section we shall present results of test computations for model problem (2.1)–(2.3). The results will provide evidence for the short computation time required by the method, for its second-order accuracy and for its insensitivity to errors in the data.

The experiments consisted in evaluating analytically d, e in (2.1) for known u, v , and then comparing the numerical solutions U, V with the correct analytical u, v . The computations were carried out on an IBM 360/95 computer, with a FORTRAN IV, level H compiler, the code optimization parameter OPT being set to the value OPT = 2. Double precision arithmetic was used throughout our numerical experiments. Test comparisons on an Amdahl 470V/6 computer and on a CDC 6600

gave entirely similar results; the program is currently being developed into a package and we expect to test it on other computers as well.

Initially, three different versions of the program were run. The first version solved the positive definite modification (3.11 α) of the system (3.11) by Cholesky factorization. It is denoted by C in the tables, and given for orientation purposes only. This version required much longer computer time; moreover, results were less accurate than with the other two versions, since the larger number of arithmetic operations introduced more round-off error.

The second version utilized diagonalization by the discrete Fourier transform as given in (3.14) and the reindexing given in (3.16), but did not use the FFT in steps (ii) and (iv) of the algorithm. It is thus somewhat comparable to Hockney's original method [21] and is also included for comparison purposes. Its numerical results were equal at least to within three significant digits to those of the third version and are listed separately only in Table III. This version is denoted by P in the tables.

TABLE I

CPU execution times for the versions C, P and F on IBM 360/95 with compiler optimization OPT = 2. The number of points is $MN = M^2/2$.

M	C	P	F
16	.07	.09	.01
32	.88	.65	.04
64	14.35	5.17	.13
128			.51
256			2.11

TABLE II

CPU execution times for version F with different optimization levels, OPT = 0, 1, 2, 3, on IBM 360/95, Amdahl 470V/6, and CDC 6600. The compiler used on the CDC computer was FTN 4.6; on the other two computers, a FORTRAN IV, level H compiler was used.

M	360/95				470V/6				6600		
	OPT/0	1	2	3	OPT/0	1	2	3	OPT/0	1	2
16	0.02	0.02	0.01	0.00	0.02	0.02	0.01	0.01	0.059	0.031	0.025
32	0.08	0.07	0.04	0.03	0.09	0.07	0.05	0.06	0.232	0.120	0.093
64	0.35	0.20	0.13	0.13	0.38	0.30	0.23	0.24	0.969	0.494	0.382
128	1.40	0.88	0.51	0.52	1.53	1.25	0.96	0.96	4.065	2.034	1.615
256	5.91	3.70	2.11	2.13	6.52	5.46	4.07	4.14			

The third version is the one actually described in the summary of Section 3 and listed in Appendix B; it represents the proposed method. This version is denoted by F in the tables.

Tables I and II contain timing results. These are essentially independent of the solution, and depend only on the method, or program version, and on the number of points. Table I compares CPU execution times for the three program versions C, P and F on the IBM 360/95 computer with compiler optimization $OPT = 2$. As the number of points increases, the advantages of version F become more and more obvious. We can also see clearly that the execution time for version F is proportional to the number of points MN .

Table II gives CPU execution times for version F with different optimization levels, $OPT = 0, 1, 2, 3$, on the IBM 360/95, Amdahl 470V/6, and CDC 6600 computers. This is meant to give an idea of the order of magnitude of running time improvements which could still be achieved by code optimization. We notice that timing with $OPT = 3$ is not superior to $OPT = 2$.

The other tables are organized as follows: the first column contains the known test solution (u, v) , and for Table V only, its L_2 and L_∞ norms. The norms used for u are the continuous norms

$$(4.1a) \quad \|u\|_2^2 \equiv \frac{1}{2\pi^2} \int_0^\pi \int_0^{2\pi} u^2(x, y) dx dy,$$

$$(4.1b) \quad \|u\|_\infty \equiv \sup_{0 \leq x \leq 2\pi, 0 \leq y \leq \pi} |u(x, y)|,$$

and similarly for v . The corresponding L_2 and L_∞ norms for the vector (u, v) are defined in the obvious way, with $u^2 + v^2$ as the integrand in (4.1a) and $\max\{|u|, |v|\}$ as the maximand in (4.1b). These norms are given in order to compare the corresponding norms of the computational error with them.

The second column contains the version of the program used, specifically version C or version P/F. The third column contains the value of M , the number of points in the x -direction. All reported computations were carried out with equal mesh spacing in the x - and y -directions, i.e., $h = k = 2\pi/M$. Hence, $N = M/2$ always, and the total number of points equals $M^2/2$.

The remaining columns contain the absolute errors in $(u - U, v - V)$. The norms used here are the discrete counterparts of the norms in the first column:

$$(4.2a) \quad l_2^2(U) \equiv \sum_{i=1}^M \sum_{j=1}^N U_{ij}^2 / MN,$$

$$(4.2b) \quad l_\infty(U) \equiv \max_{i=1, \dots, M} \max_{j=1, \dots, N} |U_{ij}|,$$

and similarly for V and (U, V) . For simplicity, in the column headings, as well as in (4.2) above, we used $l_2(U)$, instead of $l_2(u - U)$, and so on. The grid values of (u, v) entering (4.2) are those identified in Section 2 and, therefore, are taken at different locations for u than for v .

Table III shows that for u, v which are combinations of linear and of trigonometric functions, and thus eigenvectors of the discrete operator in (3.11), the results are essentially exact to machine accuracy (in double precision arithmetic).

TABLE III

Numerical results with versions C, P and F of the algorithm for a number of simple test cases. The labelling of rows and columns is explained in the text.

		H	$l_2(U)$	$l_2(V)$	$l_2(U,V)$	$l_\infty(U)$	$l_\infty(V)$
$u = \sin(y), v = y \sin(x)$	C	8	2.6351 -14	2.5457 -15	1.9989 -14	2.8144 -14	6.6613 -15
		16	2.4364 -13	3.2929 -14	1.7934 -13	2.5881 -13	7.6605 -14
		32	2.3308 -12	3.1543 -13	1.6889 -12	2.4685 -12	6.7857 -13
		64	2.1272 -11	2.6296 -12	1.5273 -11	2.2408 -11	5.5140 -12
	P	8	3.5286 -16	3.4851 -16	3.5100 -16	8.7430 -16	6.6613 -16
		16	5.2001 -16	5.0922 -16	5.1500 -16	1.4017 -15	1.5543 -15
		32	2.7386 -15	1.4532 -15	2.2120 -15	6.7862 -15	5.1070 -15
		64	1.4957 -14	5.2021 -15	1.1267 -14	2.8847 -14	3.2488 -14
	F	8	2.7088 -16	3.4397 -16	3.0436 -16	5.8287 -16	6.6613 -16
		16	6.3077 -16	3.3889 -16	5.1554 -16	1.5821 -15	6.6613 -16
		32	2.5606 -15	6.5149 -16	1.8946 -15	4.3438 -15	1.5622 -15
		64	1.4255 -14	2.2937 -15	1.0286 -14	3.9314 -14	7.7934 -15
$u = y \sin(x), v = \sin(y)$	C	8	2.3764 -15	2.5357 -15	2.5870 -15	4.2188 -15	6.4254 -15
		16	1.7252 -14	2.2782 -14	2.0024 -14	3.5083 -14	5.8356 -14
		32	1.4271 -13	1.9268 -13	1.6875 -13	2.9177 -13	5.0736 -13
		64	1.0832 -12	1.5755 -12	1.3481 -12	2.2524 -12	4.2358 -12
	P	8	1.8762 -15	2.5357 -15	2.1834 -15	4.6629 -15	9.7561 -15
		16	1.8400 -15	4.9115 -15	3.6143 -15	6.0091 -15	2.5924 -14
		32	6.1500 -15	3.9430 -14	2.7782 -14	2.7145 -14	3.2709 -13
		64	2.1996 -14	1.7951 -13	1.2690 -13	7.9020 -14	2.0683 -12
	F	8	2.0791 -16	1.9369 -16	2.0194 -16	6.6613 -16	5.6899 -16
		16	3.7749 -16	2.0176 -14	1.3785 -14	8.8818 -16	3.1225 -14
		32	3.6763 -15	1.3516 -13	9.4054 -14	7.1054 -15	2.0578 -13
		64	3.5572 -15	1.0348 -12	7.2592 -13	9.7145 -15	1.4948 -12
$u = y^4 \sin(x) + 0.001, v = y^4 + 0.001$	C	8	2.2678 -14	1.4387 -14	1.9560 -14	5.3290 -14	3.5527 -14
		16	8.0236 -14	8.2059 -14	8.1092 -14	1.9540 -13	2.4158 -13
		32	6.6951 -13	7.8598 -13	7.2820 -13	1.6165 -12	2.3128 -12
	P	8	1.1175 -14	8.9470 -15	1.0279 -14	3.6193 -14	2.1316 -14
		16	1.1179 -14	1.5689 -14	1.3750 -14	5.4622 -14	1.0303 -13
		32	4.1173 -14	1.3525 -13	9.8620 -14	2.5047 -13	1.4211 -12
	F	8	7.3201 -15	1.1531 -14	9.3597 -15	1.4655 -14	3.1974 -14
		16	1.6285 -14	2.5126 -13	1.7206 -15	3.1974 -14	5.2558 -13
		32	8.2406 -14	7.6397 -13	5.3471 -13	1.8474 -13	1.4246 -12

Table IV presents results for slightly perturbed boundary conditions $v^{(0)}(x)$, $v^{(1)}(x)$, $u(x_0, y_0)$ and right-hand sides $d(x, y)$, $e(x, y)$. Notice that errors in the computed V are bounded by the errors in the prescribed boundary data $v^{(0)}$, $v^{(1)}$ in the absence of other errors. Also observe that the l_2 error in U is larger when d, e are perturbed than when $v^{(0)}$ and $v^{(1)}$ are perturbed by a comparable amount.

Table V contains results for a number of more severe test cases, in which the discretization error is nonnegligible. For comparison, an additional last column is included in this table, which gives a theoretical error bound $e(u)$ on the l_∞ -discretization error in u . This bound is explained and proved in Appendix A; the values of the constants a, b, α and β introduced there were chosen so as to optimize the bound, and the constants C_R, C_Γ in (A.14), (A.15) were computed explicitly. We observe that the numerical errors are indeed lower than the bound in all cases in which u, v are sufficiently differentiable. In fact, the

TABLE IV

Numerical results with the proposed algorithm (version F) for problems with slightly perturbed boundary conditions $v^{(0)}(x)$, $v^{(1)}(x)$, $u(x_0, y_0)$ and right-hand sides $d(x, y)$, $e(x, y)$. The subscript $()_c$ stands for the correct values.

	M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U, V)$	$\ell_\infty(U)$	$\ell_\infty(V)$
$u = \sin(x) \sin(y), v = \sin(4x)$	16	2.1270 -2	7.0594 -2	5.0665 -2	4.1757 -2	1.0958 -1
	32	5.1732 -3	1.6540 -2	1.2091 -2	1.6793 -2	2.6040 -2
	64	1.2952 -3	4.0377 -3	2.9790 -3	5.2345 -3	6.4284 -3
	128	3.2417 -4	9.9953 -4	7.4064 -4	1.4537 -3	1.6021 -3
	256	8.1071 -5	2.4878 -4	1.8473 -4	3.8233 -4	4.0020 -4
$d = 1.1d_c, e = 1.1e_c$	16	6.8127 -2	1.4141 -1	1.0866 -1	1.0417 -1	2.1951 -1
	32	5.6760 -2	8.1393 -2	6.9773 -2	1.0102 -1	1.2814 -1
	64	5.4525 -2	6.6998 -2	6.0982 -2	1.0028 -1	1.0667 -1
	128	5.4003 -2	6.3252 -2	5.8773 -2	1.0009 -1	1.0138 -1
	256	5.3874 -2	6.2204 -2	5.8172 -2	1.0003 -1	1.0007 -1
$v^{(0)} = 1.1v_c^{(0)}, v^{(1)} = 1.1v_c^{(1)}$	16	3.8156 -3	7.7503 -2	5.3018 -2	6.2641 -3	1.1061 -1
	32	1.4438 -2	2.6059 -2	2.0885 -2	4.7086 -2	6.1123 -2
	64	1.8546 -2	1.8548 -2	1.8547 -2	7.5598 -2	6.9934 -2
	128	1.9594 -2	1.8582 -2	1.9099 -2	8.8792 -2	8.2510 -2
	256	1.9858 -2	1.9146 -2	1.9507 -2	9.4717 -2	9.0694 -2
$d = 1.01 d_c, e = 1.01 e_c$	16	2.4564 -2	7.7675 -2	5.6013 -2	4.7747 -2	1.2057 -1
	32	9.1866 -3	2.3026 -2	1.7323 -2	2.4293 -2	3.6250 -2
	64	6.1478 -3	1.0333 -2	8.4702 -3	1.3836 -2	1.6453 -2
	128	5.5561 -3	7.2249 -3	6.4382 -3	1.0738 -2	1.1580 -2
	256	5.4252 -3	6.4446 -3	5.9547 -3	1.0026 -2	1.0367 -2
$u = \sin(x) \sin(y), v = \sin(4x)$ $v^{(0)} = 1.01 v_c^{(0)}, v^{(1)} = 1.01 v_c^{(1)}$	16	1.9396 -2	7.1245 -2	5.0689 -2	3.8097 -2	1.0968 -1
	32	3.2585 -3	1.7228 -2	1.2210 -2	1.0437 -2	2.6091 -2
	64	7.3104 -4	4.9289 -3	3.4965 -3	2.8843 -3	8.8601 -3
	128	1.6720 -3	2.3682 -3	2.0472 -3	7.5755 -3	8.5085 -3
	256	1.9138 -3	2.0042 -3	1.9594 -3	9.1283 -3	9.1034 -3
$u = \sin(x) \sin(y), v = \sin(4x)$ $d = 1.01d_c, e = 1.01 e_c$ $v^{(0)} = 1.01 v_c^{(0)}, v^{(1)} = 1.01 v_c^{(1)}$ $u(x_0, y_0) = 1.01 u_c(x_0, y_0)$	16	2.2784 -2	7.8326 -2	5.6034 -2	4.4085 -2	1.2068 -1
	32	7.7732 -3	2.3701 -2	1.7407 -2	1.7937 -2	3.6301 -2
	64	5.3611 -3	1.1088 -2	8.6658 -3	1.0386 -2	1.6493 -2
	128	5.0612 -3	8.0548 -3	6.7151 -3	1.0096 -2	1.1618 -2
	256	5.0134 -3	7.3146 -3	6.2660 -3	1.0024 -2	1.0404 -2
$u = \sin(x) \sin(y)$ $v = \sin(4x) \sin(4y)$	16	3.2273 -3	5.9183 -2	4.0498 -2	6.2089 -3	1.1072 -1
	32	8.0409 -4	1.3515 -2	9.4190 -3	1.5927 -3	2.6172 -2
	64	2.0085 -4	3.2789 -3	2.3045 -3	4.0074 -4	6.4545 -3
	128	5.0203 -5	8.1045 -4	5.7193 -4	1.0035 -4	1.6082 -3
	256	1.2550 -5	2.0164 -4	1.4258 -4	2.5096 -5	4.0171 -4
$d = 1.01 d_c, e = 1.01 e_c$ $v^{(0)} = 1.01 v_c^{(0)}, v^{(1)} = 1.01 v_c^{(1)}$ $u(x_0, y_0) = 1.01 u_c(x_0, y_0)$	16	8.2597 -3	6.5120 -2	4.4892 -2	1.5891 -2	1.2183 -1
	32	5.8123 -3	1.8814 -2	1.3738 -2	1.1513 -2	3.6434 -2
	64	5.2030 -3	8.3918 -3	6.9572 -3	1.0381 -2	1.6519 -2
	128	5.0508 -3	5.8582 -3	5.4663 -3	1.0096 -2	1.1625 -2
	256	5.0128 -3	5.2234 -3	5.1188 -3	1.0024 -2	1.0406 -2
$d = 1.1 d_c, e = 1.1 e_c$ $v^{(0)} = 1.1 v_c^{(0)}, v^{(1)} = 1.1 v_c^{(1)}$ $u(x_0, y_0) = 1.1 u_c(x_0, y_0)$	16	5.3550 -2	1.1855 -1	8.9935 -2	1.0302 -1	2.2179 -1
	32	5.0885 -2	6.6507 -2	5.8963 -2	1.0079 -1	1.2879 -1
	64	5.0221 -2	5.4407 -2	5.2323 -2	1.0020 -1	1.0710 -1
	128	5.0055 -2	5.1287 -2	5.0670 -2	1.0005 -1	1.0177 -1
	256	5.0014 -2	5.0418 -2	5.0216 -2	1.0001 -1	1.0044 -1

errors become very nearly equal to the bound in some such cases (e.g., $u = y \sin(x), v = y^2 \sin(x)$), which seems to indicate that the bound is rather close to being sharp. The theoretical bound provides also an indication of error magnitude even in some cases where the data are not sufficiently differentiable (e.g., $u = y^4 \sin(8x), v = x(2\pi - x) \sin(8y); u = v = x(\pi - x)(2\pi - x) \sin(y)$).

TABLE IV (Continued)

	M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U,V)$	$\ell_\infty(U)$	$\ell_\infty(V)$
$u = \sin(x) \sin(y)$	128	6.7163 -3	3.5965 -3	5.3989 -3	3.6605 -2	2.7103 -2
$v = \sin(4x) \sin(4y)$	256	6.9041 -3	4.7732 -3	5.9392 -3	6.3567 -2	4.7378 -2
$v^{(0)} = v_c^{(0)} + 0.1 \sin(32x)$						
$v^{(1)} = v_c^{(1)} + 0.1 \sin(32x)$						
$u = \sin(x) \sin(y)$	16	3.2180 -2	7.0895 -2	5.3831 -2	7.2440 -2	1.7657 -1
$v = \sin(4x) \sin(4y)$	32	3.2461 -2	4.3097 -2	3.7981 -2	8.1892 -2	9.5994 -2
$v^{(0)} = v_c^{(0)} + 0.1 \sin(31x)$	64	7.4420 -3	4.5354 -3	6.4849 -3	4.1426 -2	1.8729 -2
$v^{(1)} = v_c^{(1)} + 0.1 \sin(31x)$	128	6.2894 -3	3.7044 -3	5.5055 -3	5.2501 -2	2.7606 -2
	256	7.0221 -3	4.9065 -3	6.0615 -3	6.9534 -2	4.8358 -2
$u = \sin(x) \sin(y)$	16	3.8439 -2	8.0665 -2	6.1843 -2	1.1296 -1	1.5208 -1
$v = \sin(4x)$	32	3.2860 -2	4.4139 -2	3.8780 -2	9.8528 -2	9.2006 -2
$v^{(0)} = v_c^{(0)} + 0.1 \sin(31x)$	64	7.5512 -3	5.1109 -3	6.4665 -3	4.6641 -2	1.8363 -2
$v^{(1)} = v_c^{(1)} + 0.1 \sin(31x)$	128	6.8369 -3	3.7503 -3	5.5256 -3	5.3952 -2	2.7601 -2
	256	7.0226 -3	4.9087 -3	6.0626 -3	6.9553 -2	4.8357 -2
$u = y^4 \sin(3x)$	16	1.1961	5.7240 -1	9.5706 -1	3.6459	1.5927
$v = x(2-x) \sin(3y)$	32	2.9682 -1	1.4061 -1	2.3461 -1	1.0109	4.1008 -1
$v^{(0)} = v_c^{(0)} + 0.1 \sin(31x)$	64	7.3896 -2	3.3020 -2	5.7534 -2	2.6792 -1	1.0273 -1
$v^{(1)} = v_c^{(1)} + 0.1 \sin(31x)$	128	1.9594 -2	8.9053 -3	1.5258 -2	1.0993 -1	3.2782 -2
	256	8.3894 -3	5.3044 -3	7.0244 -3	8.3911 -2	4.8979 -2
$u = y^4 \sin(4x)$	16	2.2889	9.2522 -1	1.7871	5.1558	2.8077
$v = x(2-x) \sin(4y)$	32	5.5349 -1	2.1416 -1	4.2463 -1	1.7679	6.7504 -1
$v^{(0)} = v_c^{(0)} + 0.1 \sin(31x)$	64	1.3716 -1	5.1032 -2	1.0410 -1	4.9607 -1	1.6534 -1
$v^{(1)} = v_c^{(1)} + 0.1 \sin(31x)$	128	3.4841 -2	1.3094 -3	2.6397 -2	1.6413 -1	4.1227 -2
	256	1.1054 -2	5.8169 -3	8.8422 -3	9.8365 -2	4.9620 -2

The formal truncation error of the finite-difference scheme, in the interior as well as on the boundary, is $O(h^2 + k^2)$,[†] hence, we expect that the scheme will produce second-order accurate results. A good way of testing this numerically is by computing the ratio

$$\|U^{(2h,2k)} - u\| / \|U^{(h,k)} - u\|,$$

and similar quantities for v and (u, v) ; here $\| \cdot \|$ is either the l_2 or the l_∞ norm of (4.2). For sufficiently small h, k and twice continuously differentiable (u, v) these ratios should be very close to 4 if the method is indeed second-order accurate, and it should be close to 2 if the method is first-order accurate.

The indicated ratios are computed and entered in additional rows in Table V; the entry in column 3 indicates the values of M and $2M$, rather than of h and $h/2$, to which the norms whose ratio was taken correspond. The interesting result is that for those cases tested the method seems to have second-order accuracy even when v is merely continuous and u once continuously differentiable and that it is first-order accurate when both u and v are merely continuous. The continuity of u, v is given in standard notation as a column to the left of the usual first column; i.e., $u, v \in C^0, C^1, C^2, \dots, C^\infty$, indicate the number of continuous derivatives of u, v . Continuity is understood for u, v and their derivatives as extended x -periodic functions.

[†] A more detailed discussion of accuracy appears in Appendix A.

TABLE V

Numerical results indicating the order of accuracy of the proposed algorithm (version F) and discretization procedure.

		M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U,V)$	$\ell_\infty(U)$	$\ell_\infty(V)$	e(u)	
$u \in C^\infty$ $v \in C^\infty$	$u = y \sin(x)$ $v = y^2 \sin(x)$ $\ u\ _2 = 1.2825$ $\ v\ _2 = 3.1210$ $\ u\ _\infty = 3.1416$ $\ v\ _\infty = 9.8696$	F	8	4.015 -2	2.879 -2	3.573 -2	8.041 -2	4.879 -2	4.455 -1
		16	1.049 -2	7.043 -3	9.049 -3	2.794 -2	1.315 -2	1.114 -1	
		32	2.657 -3	1.722 -3	2.254 -3	8.124 -3	3.414 -3	2.784 -2	
		64	6.664 -4	4.248 -4	5.607 -4	2.193 -3	8.566 -4	6.961 -3	
		8/16	3.826	4.088	3.948	2.878	3.711		
		16/32	3.950	4.090	4.016	3.440	3.851		
		32/64	3.987	4.054	4.019	3.706	3.986		
$u \in C^\infty$ $v \in C^\infty$	$u = y \sin(4x)$ $v = y^2 \sin(4x)$ $\ u\ _2 = 1.2825$ $\ v\ _2 = 3.1210$ $\ u\ _\infty = 3.1416$ $\ v\ _\infty = 9.8696$	F	16	1.7509 -1	2.5573 -1	2.1684 -1	4.9751 -1	5.5634 -1	1.426 +1
		32	4.326 2	6.1881 -2	5.292 -2	1.573 -1	1.4699 -1	3.564	
		64	1.083 -2	1.5133 -2	1.3123 -2	4.6187 -2	3.6885 -2	8.910 -1	
		128	2.7087 -3	3.752 -3	3.268 -3	1.2895 -2	9.2309 -3	2.227 -1	
		16/32	4.066	4.152	4.097	3.805	3.805		
		32/64	3.996	4.069	4.033	3.407	3.964		
		64/128	3.997	4.033	4.015	3.582	3.996		
$u \in C^\infty$ $v \in C^\infty$	$u = y \sin(8x)$ $v = y^2 \sin(8x)$ $\ u\ _2 = 1.2825$ $\ v\ _2 = 3.1210$ $\ u\ _\infty = 3.1416$ $\ v\ _\infty = 9.8696$	F	16	1.0235	1.0575 -1	7.579 -1	1.6347	1.1188 -1	2.103 +2
		32	1.6710 -1	2.973 -1	2.3912 -1	5.6803 -1	7.6216 -1	5.257 +1	
		64	4.0218 -2	7.088 -2	2.739 -2	1.7565 -1	1.856 -1	1.314 +1	
		128	9.9941 -3	1.7449 -2	1.419 -3	5.0463 -2	4.6501 -2	3.286	
		256	2.4956 -3	4.3369 -3	3.835 -3	1.4018 -2	1.1612 -2	8.215 -1	
		16/32	6.125	3.5571 -1	3.141	2.878	1.467 -1		
			4.155	4.194	4.1663	3.234	4.106		
	4.024	4.062	4.045	3.481	3.992				
	4.005	4.023	4.015	3.600	4.004				
$u \in C^\infty$ $v \in C^\infty$	$u = y \sin(16x)$ $v = y^2 \sin(16x)$ $\ u\ _2 = 1.2825$ $\ v\ _2 = 3.1210$ $\ u\ _\infty = 3.1416$ $\ v\ _\infty = 9.8696$	F	16	5.8205 +1	3.1886 +1	4.7763 +1	1.3081 +2	6.0451 +1	3.290 +3
		32	1.0321	5.4585 -2	7.4243 -1	1.7140	5.6038 -2	8.224 +2	
		64	1.5734 -1	3.2060 -1	2.5130 -1	6.0606 -1	8.8298 -1	2.056 +2	
		128	3.7506 -2	7.6097 -2	5.9846 -2	1.8544 -1	2.1426 -1	5.140 +1	
		256	9.2864 -3	1.8749 -2	1.4774 -2	5.2893 -2	5.3174 -2	1.285 +1	
		16/32	5.6397 +1	5.8415 +2	6.4334 +1	7.6319 +1	1.0788 +3		
		32/64	6.5596	1.7026 -1	2.9544	2.8280	6.3465 -2		
64/128	4.1950	4.2131	4.1991	3.2682	4.1210				
128/256	4.0388	4.0587	4.0499	3.5060	4.0294				
$u \in C^\infty$ $v \in C^\infty$	$u = \sin(x) \sin(y)$ $v = \sin(4x) \sin(4y)$ $\ u\ _2 = 0.5$ $\ v\ _2 = 0.5$ $\ u\ _\infty = 1.0$ $\ v\ _\infty = 1.0$	F	8	1.306 -2	2.1429 -16	9.8921 -3	2.2339 -2	5.2730 -16	1.268 -1
		16	3.227 -3	5.198 -2	4.0498 -2	6.2089 -3	1.1072 -1	3.171 -2	
		32	8.0409 -4	1.3515 -2	9.419 -3	1.5927 -3	2.6170 -2	7.927 -3	
		64	2.0085 -4	3.2789 -3	2.3045 -3	4.0074 -4	6.4545 -3	1.982 -3	
		8/16	4.0548	3.628 -15	2.443 -1	3.5980	4.76 -15		
		16/32	4.0135	4.3790	4.2996	3.8982	4.2305		
		32/64	4.0034	4.1219	4.0872	3.9745	4.0548		
$u \in C^\infty$ $v \in C^0$	$u = y^4 \sin(x)$ $v = x(2\pi-x) \sin(y)$ $\ u\ _2 = 22.9595$ $\ v\ _2 = 5.0966$ $\ u\ _\infty = 97.4091$ $\ v\ _\infty = 9.8696$	F	8	3.586 -1	6.33 -1	4.95 -1	5.44 -1	1.168	7.898
		16	9.332 -2	1.509 -1	1.24 -1	1.49 -1	3.14 -1	1.975	
		32	2.36 -2	3.67 -2	3.06 -2	3.821 -2	7.89 -2	4.937 -1	
		64	5.92 -3	9.03 -3	7.61 -3	9.617 -3	1.984 -2	1.234 -1	
		8/16	3.8424	4.1959	4.0077	3.6604	3.7221		
		16/32	3.9541	4.1150	4.0346	3.8922	3.9793		
		32/64	3.9880	4.0603	4.0244	3.9734	3.9761		

Table VI contains a study of the number of grid points per wave length which the method necessitates for given numerical accuracy. The results are given here as relative errors, $\ell_2(u - U)/\|u\|_2$, rather than as absolute errors, $\ell_2(u - U)$, and similarly for v and for ℓ_∞ . It seems that roughly 4 points per wave length will give 10^{-1} relative error, 8 points will give 5×10^{-2} , and 16 will give 10^{-2} . We notice again that if u oscillates less than v , the error in u will be considerably smaller than that in v .

TABLE V (Continued)

		M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U, V)$	$\ell_\infty(U)$	$\ell_\infty(V)$	e(u)		
$u \in C^\infty$ $v \in C^0$	$u = y^4 \sin(8x)$ $v = x(2\pi-x)\sin(8y)$ $ u _2 = 22.9595$ $ v _2 = 5.0966$ $ u _\infty = 97.4091$ $ v _\infty = 9.8690$	F	16	1.732 +1	3.8587	1.292 +1	3.932 +1	7.2614	3.856 +1	
			32	2.459	7.1378 -1	1.8352	6.2285	2.1761	9.640	
			64	5.853 -1	1.636 -1	4.3267 -1	2.0035	5.130 -1	2.410	
			128	1.446 -1	3.989 -2	1.064 -1	5.3048 -1	1.2653 -1	6.025 -1	
			256	3.604 -2	9.892 -3	2.647 -2	1.3449 -1	3.165 -2	1.506 -1	
			16/32	7.0428	5.406	7.040	6.312	3.337		
			32/64	4.201	4.363	4.242	3.109	4.242		
			64/128	4.048	4.101	4.0656	3.777	4.055		
			128/256	4.012	4.032	4.020	3.944	3.997		
		$u \in C^\infty$ $v \in C^0$	$u = y^4 \sin(16x)$ $v = x(2\pi-x)\sin(16y)$ $ u _2 = 22.9595$ $ v _2 = 5.0966$ $ u _\infty = 97.4091$ $ v _\infty = 9.8690$	F	16	4.66 +1	1.174 +2	8.7105 +1	6.4677 +1	2.055 +2
	32			1.816 +1	2.241	1.3108 +1	4.6798 +1	4.5798	2.017 +2	
	64			2.5178	6.1677 -1	1.8458	6.884	1.783	5.042 +1	
	128			5.964 -1	1.4347 -1	4.353 -1	2.1700	4.106 -1	1.260 +1	
	256			1.471 -1	3.518 -2	1.071 -1	5.7139 -1	1.006 -1	3.151	
	16/32			2.566	5.236 +1	6.630	1.386	4.487		
	32/64			7.212	3.634	7.118	6.781	2.568		
	64/128			4.221	4.299	4.241	3.172	4.343		
	128/256			4.053	4.078	4.061	3.798	4.080		
$u \in C^\infty$ $v \in C^0$	$u = y^4 \sin(32x)$ $v = x(2\pi-x)\sin(32y)$ $ u _2 = 22.9595$ $ v _2 = 5.0966$ $ u _\infty = 97.4091$			F	16	9.32 +1	7.511 +2	5.176 +2	1.2935 +2	1.564 +3
			32	9.176 +1	2.901 +2	2.123 +2	1.277 +2	7.936 +2	3.669 +3	
			64	1.843 +1	1.2121	1.316 +1	5.092 +1	2.7678	9.172 +2	
			128	2.535	5.813 -1	1.8459	7.2431	1.4956	2.293 +3	
			256	5.997 -1	1.3647 -1	4.356 -1	2.2600	3.4708 -1	5.732 +1	
			16/32	1.0157	2.5888	2.4378	1.0128	1.9709		
			32/64	4.9795	2.39 +2	1.613 +1	2.5081	2.867 +2		
			64/128	7.2681	2.085	7.1300	7.030	1.851		
			128/256	4.228	4.259	4.237	3.205	4.309		
		$u \in C^1$ $v \in C^1$	$u = x(\pi-x)(2\pi-x)\sin y$ $v = x(\pi-x)(2\pi-x)\sin y$ $ u _2 = v _2 = 6.0518$ $ u _\infty = v _\infty = 11.9343$		16	1.4431 -1	9.7688 -2	1.2474 -1	3.4316 -1	2.4159 -1
	32			3.6692 -2	2.4499 -2	3.1389 -2	9.5067 -2	6.3250 -2	9.461 -2	
	64			9.2269 -3	6.1387 -3	7.8604 -3	2.4802 -2	1.6099 -2	2.365 -2	
	128			2.3104 -3	1.5368 -3	1.9651 -3	6.3163 -3	4.0366 -3	5.913 -3	
	256			5.7783 -4	3.8452 -4	4.9116 -4	1.5928 -3	1.0101 -3	1.478 -3	
	16/32			3.9330	3.9874	3.9739	3.6097	3.8196		
	32/64			3.9766	3.9910	3.9933	3.8330	3.9287		
	64/128			3.9937	3.9943	4.0000	3.9267	3.9884		
	128/256			3.9984	3.9968	4.0002	3.9656	3.9960		
$u \in C^1$ $v \in C^0$	$u = x(\pi-x)(2\pi-x)\sin y$ $v = x(2\pi-x)\sin y$ $ u _2 = 6.0518$ $ v _2 = 5.0966$ $ u _\infty = 11.9343$ $ v _\infty = 9.8696$				16	1.3883 -1	1.2207 -1	1.3127 -1	3.0675 -1	2.5198 -1
			32	3.4733 -2	1.9011 -2	2.8241 -2	8.3754 -2	6.0616 -2	9.461 -2	
			64	8.6870 -3	4.8190 -3	7.0539 -3	2.1836 -2	1.5363 -2	2.365 -2	
			128	2.1720 -3	1.2133 -3	1.7629 -3	5.5692 -3	3.8529 -3	5.913 -3	
			256	5.4303 -4	3.0443 -4	4.4065 -4	1.4059 -3	9.6442 -4	1.478 -3	
			16/32	3.9970	6.4210	4.6484	3.6625	4.1570		
			32/64	3.9983	3.9450	4.0036	3.8357	3.9456		
			64/128	3.9995	3.9717	4.0014	3.9207	3.9873		
			128/256	3.9999	3.9856	4.0006	3.9613	3.9950		
		$u \in C^0$ $v \in C^0$	$u = x(2\pi-x)\sin y$ $v = x(2\pi-x)\sin y$ $ u _2 = v _2 = 5.0966$ $ u _\infty = v _\infty = 9.8696$		16	4.6542 -1	5.0733 -1	4.8543 -1	1.0241	1.1306
	32			2.3271 -1	2.3676 -1	2.3468 -1	5.6744 -1	5.7633 -1	3.912 -2	
	64			1.1639 -1	1.2030 -1	1.1833 -1	2.9686 -1	2.9230 -1	9.780 -3	
	128			5.8198 -2	6.0629 -2	5.9417 -2	1.5144 -1	1.4679 -1	2.445 -3	
	256			2.9100 -2	3.0434 -2	2.9772 -2	7.6427 -2	7.3474 -2	6.112 -4	
	16/32			2.0000	2.1428	2.0685	1.8047	1.9618		
	32/64			1.9994	1.9680	1.9832	1.9115	1.9717		
	64/128			1.9999	1.9842	1.9916	1.9603	1.9913		
	128/256			2.0000	1.9922	1.9958	1.9814	1.9978		

These conclusions are also supported by some of the results in Table V. It is interesting that experiments with solutions containing odd wave numbers give results which are only slightly worse than those for even wave numbers, if at all; in other words, using M, N which are powers of 2 is not detrimental to accuracy, even when odd wave numbers are present in the solution.

TABLE V (Continued)

		M	$\epsilon_2(U)$	$\epsilon_2(V)$	$\epsilon_2(U,V)$	$\epsilon_\infty(U)$	$\epsilon_\infty(V)$	$e(u)$	
$u \in C^2$ $v \in C^4$	$u = x^2(2\pi-x)^2 \sin(y)$ $v = x^4(2\pi-x)^4 \sin(y)$ $ u _2 = 43.9070$ $ v _2 = 1247.6048$ $ u _\infty = 97.4091$ $ v _\infty = 9488.531$	F	8	1.6926 +1	1.2066 +2	8.0018 +1	3.3808 +1	2.909 +2	2.310 +1
			16	3.5778	2.7330 +1	1.8850 +1	8.311	7.162 +1	5.774
			32	8.674 -1	6.5176	4.6131	2.009	1.783 +1	1.443
			64	2.155 -1	1.614	1.427	4.9799 -1	4.4538	3.609 -1
			8/16	4.7319	4.4148	4.2446	4.0678	4.0618	
			16/32	4.1236	4.1593	4.0867	4.1368	4.0160	
	32/64	4.0258	4.0706	4.0370	4.0344	4.0040			
$u \in C^1$ $v \in C^3$	$u = x(\pi-x)(2\pi-x) \sin(y)$ $v = 0$ $ u _2 = 6.0518$ $ u _\infty = 11.9343$	F	16	2.1348	9.5464 -2	1.5604	4.2730	2.4159 -1	3.784 -1
			32	4.7264 -1	2.3985 -2	3.3996 -1	1.1174	6.3250 -2	9.461 -2
			64	1.0805 -1	6.0150 -3	7.7121 -2	2.8579 -1	1.6099 -2	2.365 -2
			128	2.5536 -2	1.5065 -3	1.8158 -2	7.2238 -2	4.0366 -3	5.913 -3
			256	6.1832 -3	3.7699 -4	4.3888 -3	1.8161 -2	1.0101 -3	1.478 -3
			16/32	4.5167	3.9800	4.5899	3.8241	3.8196	
	32/64	4.3743	3.9877	4.4081	3.9106	3.9287			
	64/128	4.2313	3.9927	4.2471	3.9554	3.9884			
	128/256	4.1299	3.9961	4.1375	3.9777	3.9960			
$u \in C^0$ $v \in C^\infty$	$u = x(2\pi-x) \sin(y)$ $v = y^4 \sin(x)$ $ u _2 = 5.0966$ $ v _2 = 22.9595$ $ u _\infty = 9.8696$ $ v _\infty = 97.4091$	F	8	1.1787	1.0090	1.1091	2.8276	1.9799	1.265 +1
			16	4.9757 -1	4.889 -1	4.936 -1	1.3147	1.1202	3.162
			32	2.3681 -1	2.433 -1	2.3999 -1	6.4005 -1	5.7897 -1	7.904 -1
			64	1.169 -1	1.2179 -1	1.193 -1	3.1481 -1	2.9311 -1	1.976 -1
			128	5.826 -2	6.0988 -2	5.963 -2	1.559 -1	1.4697 -1	4.940 -2
			256	2.9107 -2	3.052 -2	2.982 -2	7.754 -2	7.352 -2	1.235 -2
	16/32	2.1011	2.0093	2.0566	2.0540	1.9349			
	32/64	2.0257	1.9978	2.0110	2.0332	1.9752			
	64/128	2.0065	1.9972	2.0013	2.0194	1.9944			
	128/256	2.0016	1.9982	1.9996	2.0105	1.9990			

In particular, these results also show that the present solver would perform very well on linearized versions of the original geophysical fluid dynamic problem we were interested in ([16], [17]). We shall return to this point in Section 7.

5. Changes in Boundary Conditions. In Section 2 we have formulated the model problem (2.1)–(2.3) which motivated this study. The algorithm of Section 3 has obvious applications to many other situations; it is of interest, therefore, to consider a number of different boundary conditions which could be associated with the Cauchy-Riemann equations (2.1) in a rectangle.

We shall assume throughout this section that the boundary conditions on $y = 0$, π are still (2.2), i.e., v is prescribed there as $v^{(0)}(x)$ and $v^{(1)}(x)$, respectively. The two different combinations of boundary conditions we consider explicitly are: (1) that u is given on the left boundary of the rectangle and v is given on the right boundary, and (2) that u is given on both vertical sides.

It is clear that if v is given on all sides, the problem should be formulated as Poisson's equation for v with Dirichlet boundary conditions; similarly if u is given on all the sides, a Dirichlet problem for u is more suitable. A moment's thought will show that the two situations we shall discuss can easily be transformed into a considerable number of others, by reflections or by interchanging the roles of x and y , and of u and v . In fact, all situations in which u as well as v are prescribed on some of the sides of the rectangle can be handled by slight modifications of the algorithms we present, yielding second-order accurate numerical solutions.

TABLE VI

Numerical results indicating the resolution (mesh points per wave length) required by the proposed discretization procedure and solution algorithm (versions C and F) to obtain prescribed accuracy.

	M	$\ell_2(U)/ u _2$	$\ell_2(V)/ v _2$	$\ell_\infty(U)/ u _\infty$	$\ell_\infty(V)/ v _\infty$
$u = y \sin(x), v = y^2 \sin(x)$	C 64	6.6212 -4	1.3611 -4	8.6548 -4	8.6792 -5
	F 64	5.1958 -4	1.3611 -4	6.9789 -4	8.6788 -5
$u = y \sin(4x), v = y^2 \sin(4x)$ $u = y \sin(8x), v = y^2 \sin(8x)$ $u = y \sin(64x), v = y^2 \sin(64x)$	F 64	8.444 -3	4.849 -3	1.47 -2	3.737 -3
	F 64	3.1358 -2	2.271 -2	5.59 -2	1.88 -2
	F 64	1.9450 +2	4.4180 +1	2.0070 +2	3.00 +1
	128	8.0699 -1	4.4600 -3	5.6440 -1	1.42 -3
	256	1.1422 -1	1.0867 -1	2.0238 -1	1.0347 -1
$u = \sin(x) \sin(y), v = \sin(4x) \sin(4y)$	C 64	4.1000 -4	6.5580 -3	4.4250 -4	6.4550 -3
	F 64	4.0170 -4	6.5578 -3	4.0074 -4	6.4550 -3
$u = y^4 \sin(x), v = x(2\pi-x) \sin(y)$ $u = y^4 \sin(8x), v = x(2\pi-x) \sin(8y)$ $u = y^4 \sin(16x), v = x(2\pi-x) \sin(16y)$ $u = y^4 \sin(32x), v = x(2\pi-x) \sin(32y)$ $u = y^4 \sin(64x), v = x(2\pi-x) \sin(64y)$ $u = y^4 \sin(128x), v = x(2\pi-x) \sin(128y)$ $u = x(2\pi-x) \sin(y), v = y^4 \sin(x)$	C 64	3.0218 -4	1.7716 -3	1.3592 -4	2.0102 -3
	F 64	2.5784 -4	1.7716 -3	9.8728 -4	2.0102 -5
	F 64	2.5490 -2	3.2100 -2	2.0570 -2	5.1980 -2
	F 64	1.097 -1	1.210 -1	7.067 -2	1.807 -1
	F 64	8.027 -1	2.378 -1	5.227 -1	2.085 -1
	F 64	7.962	1.2598 +2	2.617	2.085 +2
	128	8.0577 -1	1.2376 -1	5.4615 -1	1.5496 -1
	256	1.1064 -1	1.1180 -1	7.6292 -2	1.5496 -1
	F 64	1.592 +1	6.956 +2	5.234	9.36 +2
	128	1.5906 +1	2.6429 +2	5.3360	4.6962 +2
	256	1.0707 -1	6.514 -2	5.5526 -1	8.1543 -2
$u = x^2(2\pi-x)^2 \sin(y), v = x^4(2\pi-x)^4 \sin(y)$ $u = x(\pi-x)(2\pi-x) \sin(y), v = 0$	C 64	1.121 -2	1.2937 -3	9.4755 -3	4.694 -4
	F 64	4.9081 -3	1.2937 -3	5.1124 -3	4.6939 -4
	F 64	1.785 -2	---	5.267 -2	---
$u = y^4 \sin(3x), v = x(2\pi-x) \sin(3y)$	F 16	5.2079 -2	1.1205 -1	3.6800 -2	1.6137 -1
	32	1.2851 -2	2.6395 -2	9.6328 -3	4.1549 -2
	64	3.2022 -3	6.4495 -3	2.4339 -3	1.0409 -2
	128	7.9989 -4	1.5969 -3	6.0900 -4	2.6059 -3
	256	1.9993 -4	3.9748 -4	1.5225 -4	6.5159 -4
$u = y^4 \sin(5x), v = x(2\pi-x) \sin(5y)$	F 16	1.6604 -1	2.7604 -1	1.2079 -1	4.2728 -1
	32	3.8882 -2	5.9866 -2	3.1351 -2	9.6724 -2
	64	9.5680 -3	1.4350 -2	7.8489 -3	2.3872 -2
	128	2.3826 -3	3.5362 -3	1.9858 -3	6.0154 -3
	256	5.9508 -4	8.7915 -4	4.9629 -4	1.5020 -3
$u = y^4 \sin(7x), v = x(2\pi-x) \sin(7y)$	F 16	3.7455 -1	5.7315 -1	2.7813 -1	6.7758 -1
	32	8.0104 -2	1.0879 -1	6.5915 -2	1.6856 -1
	64	1.9318 -2	1.9621 -2	1.6605 -3	4.0480 -2
	128	4.7869 -3	6.2087 -3	4.1578 -3	1.0380 -2
	256	1.1941 -3	1.5410 -3	1.0382 -3	2.5881 -3
$u = y^4 \sin(9x), v = x(2\pi-x) \sin(9y)$	F 16	7.4843 -1	1.1403	5.5451 -1	1.3986
	32	1.3893 -1	1.7668 -1	1.1704 -1	2.3986 -1
	64	3.2567 -2	3.9733 -2	2.8390 -2	6.3575 -2
	128	8.0160 -3	9.6458 -3	7.1369 -3	1.5598 -2
	256	1.9963 -3	2.3893 -3	1.7829 -3	3.8817 -3

We proceed now with the description of the algorithm for the two cases mentioned.

Case 1. u Given on the Left Side. The rectangular domain is now taken as $R_1 = \{(x, y) : -h/2 < x < 2\pi, 0 < y < \pi\}$. This is merely done for notational convenience, so as to leave Figure 1 unchanged. The boundary conditions are first (2.2),

which we repeat here as

$$(5.1a) \quad v = v^{(0)}(x), \quad y = 0,$$

$$(5.1b) \quad v = v^{(1)}(x), \quad y = \pi,$$

and also

$$(5.2a) \quad u = u_{(0)}(y), \quad x = -h/2,$$

$$(5.2b) \quad v = v_{(1)}(y), \quad x = 2\pi.$$

Thus Eqs. (5.2) replace the periodicity conditions (2.3a, b). Eqs. (2.1), together with (5.1), (5.2) completely determine u , v , and u need not and should not be prescribed any more at an interior point of R_1 .

The difference equations are still (2.4), which we repeat for convenience as

$$(5.3a) \quad (U_{i,j} - U_{i-1,j})/h + (V_{i,j} - V_{i,j-1})/k = D_{i,j}, \quad 1 \leq i \leq M, 1 \leq j \leq N,$$

$$(5.3b) \quad (U_{i,j+1} - U_{i,j})/k - (V_{i+1,j} - V_{i,j}) = E_{i,j}, \quad 1 \leq i \leq M, 1 \leq j \leq N-1.$$

The boundary conditions become

$$(5.4a) \quad V_{i,0} = v^{(0)}((i-1)h), \quad 1 \leq i \leq M,$$

$$(5.4b) \quad V_{i,N} = v^{(1)}((i-1)h), \quad 1 \leq i \leq M,$$

and

$$(5.5a) \quad U_{0,j} = u_{(0)}((j-1/2)k), \quad 1 \leq j \leq N,$$

$$(5.5b) \quad V_{M+1,j} = v_{(1)}(jk), \quad 0 \leq j \leq N.$$

Hence, there are MN interior U -values to be determined, and $M(N-1)$ interior V -values, or $M(2N-1)$ unknowns altogether. Eqs. (5.3) yield $M(2N-1)$ linear algebraic equations; we shall see that these equations are actually independent and determine U , V completely.

It turns out to be more convenient in this case to form column vectors \mathbf{U}_i , \mathbf{V}_i from the values of U , V along vertical mesh lines; in Section 3 vectors \mathbf{U}_j , \mathbf{V}_j were formed along horizontal mesh lines. Thus

$$\mathbf{U}_i = (U_{i,1}, U_{i,2}, \dots, U_{i,N})^*, \quad \mathbf{V}_i = (V_{i,1}, V_{i,2}, \dots, V_{i,N-1})^*;$$

in particular \mathbf{U}_i , \mathbf{V}_i have now different lengths, the \mathbf{U}_i 's being N -vectors, while the \mathbf{V}_i 's are $(N-1)$ -vectors. In a similar fashion, \mathbf{D}_i is an N -vector, while \mathbf{E}_i is an $(N-1)$ -vector of values along the corresponding vertical mesh lines.

With this notation, (5.3) becomes

$$(5.6a) \quad \mathbf{U}_i - \mathbf{U}_{i-1} + T\mathbf{V}_i = h\mathbf{D}_i, \quad 1 \leq i \leq M,$$

$$(5.6b) \quad \mathbf{V}_{i+1} - \mathbf{V}_i + T^*\mathbf{U}_i = h\mathbf{E}_i, \quad 1 \leq i \leq M;$$

$$(5.10b) \quad Q = \begin{bmatrix} -I_{N-1} & I_{N-1} & & & \\ & \ddots & \ddots & \ddots & \\ & & -I_{N-1} & I_{N-1} & \\ & & & & -I_{N-1} \end{bmatrix},$$

$$(5.10c) \quad R = \begin{bmatrix} T^* & & & & \\ & T^* & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & T^* \end{bmatrix},$$

$$(5.10d) \quad F = \begin{bmatrix} D_1 \\ D_1 + D_2 \\ \vdots \\ \vdots \\ \sum_1^M D_i \end{bmatrix},$$

and I is the $MN \times MN$ identity matrix. From (5.9) we have

$$(5.11a) \quad U = hF - PV,$$

$$(5.11b) \quad (Q - RP)V = h(E - RF),$$

with

$$(5.11c) \quad RP = \begin{bmatrix} S & & & & \\ S & S & & & \\ \vdots & \ddots & \ddots & \ddots & \\ S & \cdots & S & S \end{bmatrix};$$

here $S = T^*T$ is an $(N - 1) \times (N - 1)$ matrix,

$$(5.12) \quad S = \rho^2 \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

Notice that S , in contradistinction to S of Section 3, is nonsingular. We shall return to it later.

Written out explicitly, (5.11b) becomes after a change of sign,

$$(5.13) \quad \begin{bmatrix} S + I & & -I & & & & & & & & \\ & S & & S + I & & -I & & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -I & \\ & S & \cdots & S & & & & & S + I & & \end{bmatrix} \mathbf{V} = h \begin{bmatrix} T^* \mathbf{D}_1 - \mathbf{E}_1 \\ T^* (\mathbf{D}_1 + \mathbf{D}_2) - \mathbf{E}_2 \\ \vdots \\ T^* \sum_1^M \mathbf{D}_i - \mathbf{E}_M \end{bmatrix},$$

where I is now the $(N - 1) \times (N - 1)$ identity. This system can be brought into block-tridiagonal form simply by subtracting the i th row from the $(i + 1)$ st, starting from the top. This produces the system

$$(5.14) \quad \begin{bmatrix} S + I & & -I & & & & & & & & \\ & -I & & S + 2I & & -I & & & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & & & -I & & S + 2I & & -I \\ & & & & & & & & -I & & S + 2I \end{bmatrix} \mathbf{V} = h \begin{bmatrix} T^* \mathbf{D}_1 - \mathbf{E}_1 \\ T^* \mathbf{D}_2 + \mathbf{E}_1 - \mathbf{E}_2 \\ \vdots \\ T^* \mathbf{D}_M + \mathbf{E}_{M-1} - \mathbf{E}_M \end{bmatrix}.$$

We see that the decoupling resulted in this case in the elimination of U , rather than of V . Equation (5.14) is rather similar to (3.11). Some of the differences have already been pointed out; as a result of these differences, the matrix of (5.14) is non-singular. It can be brought to scalar tridiagonal form by diagonalizing S and then re-indexing. We shall comment on the fast solution of (5.14) further at the end of this section, together with the fast solution of the matrix equation obtained in the second case we wish to discuss.

Case 2. *u Given on Both Vertical Sides.* We fit the grid to the rectangular domain so that $R_2 = \{(x, y): -h/2 < x < 2\pi - h/2, 0 < y < \pi\}$. The boundary conditions are (5.1) on the horizontal sides of the rectangle, and

$$(5.15a) \quad u = u_{(0)}(y), \quad x = -h/2,$$

$$(5.15b) \quad u = u_{(1)}(y), \quad x = 2\pi - h/2,$$

on the vertical sides. Equations (2.1), (5.1) and (5.15) determine u and v completely, subject to the requirement that the data $d(x, y)$, $u_{(0)}(y)$, $u_{(1)}(y)$, $v^{(0)}(x)$, $v^{(1)}(x)$ satisfy the Gauss divergence theorem:

$$(5.16) \quad \iint_{R_2} d(x, y) \, dx \, dy = \int_{-h/2}^{2\pi-h/2} \{v^{(1)}(x) - v^{(0)}(x)\} \, dx + \int_0^\pi \{u_{(1)}(y) - u_{(0)}(y)\} \, dy.$$

The difference equations are (5.3), with (5.3b) only being written for $1 \leq i \leq M - 1$, $1 \leq j \leq N - 1$. The boundary conditions for the mesh variables are (5.4) and

$$(5.17a) \quad U_{0,j} = u_{(0)}((j - 1/2)k), \quad 1 \leq j \leq N,$$

$$(5.17b) \quad U_{M,j} = u_{(1)}((j - 1/2)k), \quad 1 \leq j \leq N.$$

Hence, there remain $(M - 1)N$ U -values to be determined, and $M(N - 1)$ V -values, i.e. $2MN - M - N$ unknowns. In (5.3) we have MN D -equations and $(M - 1)(N - 1)$ E -equations, i.e., $2MN - M - N + 1$ equations in all.

It would seem that the number of equations exceeds the number of unknowns by one. We expect, however, from the continuous case that one compatibility condition has to be imposed on the data, analogous to (5.16), and that the matrix of system (5.3) have rank equal to the number of unknowns. This can be checked directly in the block form (5.18) to which we shall bring this matrix below; the compatibility condition turns out to be the one obtained when computing the integrals in (5.16) by the midpoint rule. A similar statement is true in the case in which u is prescribed on the horizontal sides of the rectangle, and v on the vertical sides; in that case the compatibility condition is the discrete analog of Stokes' curl theorem, involving $e(x, y)$ rather than $d(x, y)$.

We introduce the N -vectors U_i, D_i and the $(N - 1)$ -vectors V_i, E_i as in the previous case. Again, the first and last components of $D_i, D_{i,0}$ and $D_{i,N}$, are modified by the addition of $\rho h^{-1} V_{i,0}$ and of $-\rho h^{-1} V_{i,N}$, respectively. Also, D_1 is modified by the addition of $h^{-1} U_0$ and D_M by the addition of $-h^{-1} U_M$.

After these changes of notation, system (5.3) becomes

$$\begin{aligned}
 & \begin{matrix} M \\ M-1 \end{matrix} \left[\begin{array}{cc|cc} \hline & \overbrace{\quad\quad\quad}^M & & \overbrace{\quad\quad\quad}^{M-1} \\ \hline \tau & & I_N & \\ & \tau & -I_N & I_N \\ & & \cdot & \cdot \\ & & \cdot & \cdot \\ & & \cdot & I_N \\ & & \tau & -I_N \\ \hline -I_{N-1} & I_{N-1} & \tau^* & \\ & \cdot & \cdot & \\ & \cdot & \cdot & \\ & & \cdot & \\ & -I_{N-1} & I_{N-1} & \tau^* \\ \hline \end{array} \right] \begin{matrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ V_M \\ \hline U_1 \\ \cdot \\ \cdot \\ U_{M-1} \end{matrix} \\
 (5.18) \quad & = h \begin{bmatrix} D_1 \\ \cdot \\ \cdot \\ \cdot \\ D_M \\ \hline -E_1 \\ \cdot \\ \cdot \\ -E_{M-1} \end{bmatrix},
 \end{aligned}$$

where T is the $N \times (N - 1)$ matrix defined by (5.7), and I_{N-1}, I_N are identity matrices of the appropriate dimensions. Clearly, the sum of the MN rows in the upper half of the matrix in (5.18) is zero. The corresponding compatibility condition that $\sum_{i,j} D_{ij} = 0$ is exactly the one we expected; we only need to remember that the D_{ij} close to the boundary have been redefined to include the boundary data with appropriate coefficients.

The elimination of V proceeds in a manner analogous to Case 1, by summing the blocks of the upper half of (5.18), in a discrete form of integration with respect to x . The result is

$$(5.19) \quad \begin{array}{c} \left. \begin{array}{c} M-1 \\ \vdots \\ M-1 \end{array} \right\} \left[\begin{array}{c|c} \overbrace{\begin{matrix} T & & & & 0 \\ T & T & & & \cdot \\ \vdots & \cdot & \cdot & \cdot & \cdot \\ T & \cdots & T & T & -0 \end{matrix}}^{M-1} & \begin{matrix} I_N \\ \\ \\ I_N \end{matrix} \\ \hline \begin{matrix} T & \cdot & \cdot & \cdot & T & T & 0 & \cdots & 0 \end{matrix} \end{array} \right] \begin{array}{c} \left[\begin{array}{c} V_1 \\ \vdots \\ V_{M-1} \\ V_M \\ U_1 \\ \vdots \\ U_{M-1} \end{array} \right] = h \left[\begin{array}{c} D_1 \\ D_1 + D_2 \\ \vdots \\ \sum_j^M D_j \\ -E_1 \\ \vdots \\ -E_{M-1} \end{array} \right] \\ \left. \begin{array}{c} M-1 \\ \vdots \\ M-1 \end{array} \right\} \left[\begin{array}{c|c} \begin{matrix} -I_{N-1} & I_{N-1} \\ \cdot & \cdot \\ \cdot & \cdot \\ -I_{N-1} & I_{N-1} \end{matrix} & \begin{matrix} T^* \\ \cdot \\ \cdot \\ T^* \end{matrix} \end{array} \right] \end{array}$$

We rewrite this, with the obvious identifications, as

$$(5.20) \quad \left[\begin{array}{c|c} P & I \\ \hline T \cdots T & 0 \cdots 0 \\ \hline Q & R \end{array} \right] \begin{array}{c} \left[\begin{array}{c} V \\ \\ U \end{array} \right] = h \left[\begin{array}{c} F \\ \sum_j^M D_j \\ -E \end{array} \right] .$$

Notice that P and Q have block dimension $(M - 1) \times M$, P has blocks of dimension $N \times (N - 1)$, and Q has blocks of dimension $(N - 1) \times (N - 1)$.

From (5.20) we obtain

$$(5.21a) \quad U = hF - PV,$$

$$(5.21b) \quad QV + RU = -hE.$$

This allows us to eliminate U and write

$$(5.22a) \quad (Q - RP)V = -h(E + RF),$$

$$(5.22b) \quad T^*(T \ T \ \dots \ T)V = hT^* \sum_1^M D_j;$$

here we introduced the block row missing in (5.21) as (5.22b). The elimination of the single redundant equation in (5.18) was done naturally by multiplication of (5.22b) with the $(N - 1) \times N$ matrix T^* .

System (5.22) becomes, after carrying out the matrix multiplications,

$$(5.23) \quad \begin{bmatrix} S + I & -I & & & & & & & & & \\ & S & S + I & -I & & & & & & & \\ & \vdots & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \\ & S & \cdot & \cdot & \cdot & S & S + I & -I & & & \\ & S & \cdot & \cdot & \cdot & S & S & & & & \end{bmatrix} V = h \begin{bmatrix} T^*D_1 + E_1 \\ T^*(D_1 + D_2) + E_2 \\ \vdots \\ T^* \sum_1^{M-1} D_i + E_{M-1} \\ T^* \sum_1^M D_i \end{bmatrix},$$

with S being the $(N - 1) \times (N - 1)$ matrix defined by (5.12). The matrix of this system is the same as that of (5.13), except for the lower right corner block; also the right-hand sides differ only slightly. Applying the block-tridiagonalization procedure used in Case 1, which corresponds to differencing in x , one obtains

$$(5.24) \quad \begin{bmatrix} S + I & -I & & & & & & & & & \\ & -I & S + 2I & -I & & & & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \\ & & & -I & S + 2I & -I & & & & & \\ & & & & -I & S + I & & & & & \end{bmatrix} V = h \begin{bmatrix} T^*D_1 + E_1 \\ T^*D_2 + E_2 - E_1 \\ \vdots \\ T^*D_{M-1} + E_{M-1} - E_{M-2} \\ T^*D_M - E_{M-1} \end{bmatrix}.$$

We are now prepared to discuss the fast solution of (5.14) and of (5.24).

Fast Sine Transform. The fast solution of (5.14) and of (5.24) involves bringing the corresponding matrices to scalar tridiagonal form. This is done in two steps: the first and crucial step is to diagonalize S ; the second is to bring the two diagonals which are identically -1 from the position of block subdiagonal and block superdiagonal to that of scalar sub and superdiagonal, i.e., immediately adjacent to the main diagonal.

We shall write the procedure for a slightly more general system, which includes (5.14) and (5.24) as special cases, to wit:

$$(5.25) \quad \begin{bmatrix} S_1 + \alpha_1 I & \gamma_1 I & & & & & \\ \beta_1 I & S_2 + \alpha_2 I & \gamma_2 I & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & & \gamma_{M-1} I & \\ & & & & \beta_{M-1} I & S_M + \alpha_M I & \end{bmatrix} \mathbf{W} = \mathbf{B};$$

\mathbf{W} and \mathbf{B} are partitioned to conform with the blocks of the matrix,

$$\mathbf{W} = (\mathbf{W}_1^*, \mathbf{W}_2^*, \dots, \mathbf{W}_M^*)^*, \quad \mathbf{B} = (\mathbf{B}_1^*, \mathbf{B}_2^*, \dots, \mathbf{B}_M^*)^*, \quad \text{and} \quad S_i = \delta_i S.$$

The eigenvalues μ_k and eigenvectors η_k of S ,

$$(5.26a) \quad S\eta_k = \mu_k \eta_k, \quad 1 \leq k \leq N-1,$$

are known:

$$(5.26b) \quad \mu_k = 2\rho^2 \{1 + \cos(\pi k/N)\},$$

$$(5.26c) \quad \eta_{kl} = (2/N)^{1/2} \sin(\pi kl/N).$$

The matrix S differs from S of Section 3 inasmuch as its eigenvectors are generated by the sine function, rather than by the exponential, as in (3.12). Its diagonalization thus corresponds to the fast sine transform, in the same way in which S was connected to the FFT. The differences arise in the continuous problem because of the different boundary conditions.

Let P be the matrix whose columns are $\eta_1, \eta_2, \dots, \eta_{N-1}$, and let $M = \text{diag}(\mu_1, \dots, \mu_{N-1})$. Then

$$(5.27) \quad P^* S P = M, \quad P^* P = I_{N-1}.$$

We introduce $\tilde{\mathbf{W}}_i, \tilde{\mathbf{B}}_i$ by

$$(5.28a) \quad \tilde{\mathbf{W}}_i = P^* \mathbf{W}_i,$$

$$(5.28b) \quad \tilde{\mathbf{B}}_i = P^* \mathbf{B}_i, \quad 1 \leq i \leq M,$$

and $M_i = \delta_i M$. We premultiply (5.25) by a block-diagonal matrix with all the diagonal blocks equal to P^* and use (5.27), (5.28) to yield

$$(5.29) \quad \begin{bmatrix} M_1 + \alpha_1 I & \gamma_1 I & & & & & \\ \beta_1 I & M_2 + \alpha_2 I & \gamma_2 I & & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & & \gamma_{M-1} I & \\ & & & & \beta_{M-1} I & M_M + \alpha_M I & \end{bmatrix} \tilde{\mathbf{W}} = \tilde{\mathbf{B}}.$$

Reindexing $\tilde{\mathbf{W}}$ and $\tilde{\mathbf{B}}$ into $\hat{\mathbf{W}}$ and $\hat{\mathbf{B}}$ by

$$(5.30) \quad \hat{\mathbf{W}}_{k,i} = \tilde{\mathbf{W}}_{i,k}, \quad \hat{\mathbf{B}}_{k,i} = \tilde{\mathbf{B}}_{i,k}, \quad 1 \leq i \leq M, \quad 1 \leq k \leq N-1,$$

the scalar tridiagonalization of (5.25) is completed in the form

$$(5.31) \quad \begin{bmatrix} C_1 & & & & \\ & C_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & C_N \end{bmatrix} \hat{W} = \hat{B};$$

here each C_k is a nonsingular scalar tridiagonal $M \times M$ matrix,

$$(5.32a) \quad C_k = \begin{bmatrix} \delta_1 \mu_k + \alpha_1 & & & & & & \gamma_1 \\ & \beta_1 & & & & & & \gamma_2 \\ & & \delta_2 \mu_k + \alpha_2 & & & & & & \ddots \\ & & & \ddots & & & & & & \ddots \\ & & & & \ddots & & & & & & \gamma_{M-1} \\ & & & & & & & & & & & & \beta_{M-1} \\ & & & & & & & & & & & & & \delta_M \mu_k + \alpha_M \end{bmatrix},$$

$1 \leq k \leq N-1.$

After performing and storing the LU decomposition of the C_k 's, the solution of (5.25) is now carried out by a forward fast sine transform (5.28b) of the data \hat{B} into \tilde{B} , elimination and substitution in each subsystem

$$(5.32b) \quad C_k \hat{W}_k = \hat{B}_k, \quad 1 \leq k \leq N-1,$$

and a backward fast sine transform (5.28a) of the solution \tilde{W} into W ; the reindexing of \tilde{B} into \hat{B} and of \hat{W} into \tilde{W} does not necessitate actual computer operations. Furthermore, in our application,

$$\beta_i = \gamma_i = -1, \quad \delta_i = 1, \quad 1 \leq i \leq M,$$

and all α_i are 2, except for $\alpha_M = 1$ in (5.24).

The operation counts and storage requirements are, therefore, the same as in Section 3; in particular, the same remarks apply to the computation of the sine transform as to the real Fourier transform, i.e., the practical computer time required is essentially determined by twice the number of real data.

Numerical experiments similar to those in Tables I through VI of Section 4 were carried out for Cases 1 and 2, with the algorithm described above. The results were entirely analogous, confirming the fact that the algorithm is second-order accurate and the computational time it requires is essentially proportional to the number of mesh points used in the discretization.

It is remarkable that the numerical tests still indicate second-order accuracy for $u \in C^p$, $v \in C^q$ with $p \geq 1$, $q \geq 0$, and lower-order accuracy for $u \in C^p$, $v \in C^q$ with $p < 1$, $q < 0$. This is true in both Case 1 and Case 2 for jump discontinuities in u and v or their derivatives introduced along either $x = \text{const}$ or $y = \text{const}$. These numerical observations seem to indicate that the method's second-order accuracy even for solutions without the formally required differentiability is not restricted to the case of

periodic boundary conditions. Furthermore, the asymmetry with regard to the continuity requirements on u and v does not appear to depend on whether U or V are eliminated in the algorithm, or on the boundary conditions imposed.

6. Comparison with Existing Solvers. A fast direct solver for the inhomogeneous Cauchy-Riemann equations was published by Lomax and Martin [24]. Some applications and extensions are given in [25], [26]. We carried out a comparison between our solvers and those published previously. The comparison was made first for the test case of [24], [26], then for some of the test cases in our Tables III through VI and similar ones. These computations were carried out on a CDC 6600 computer with a FTN 4.6 compiler; single precision arithmetic was used throughout.

Thin Biconvex Airfoil. The example problem used in [24], [26] to illustrate the use of a Cauchy-Riemann solver in aerodynamics is that of steady, irrotational, subsonic, inviscid flow over a thin symmetrical parabolic-arc biconvex airfoil in the small-perturbation approximation. The linearized Prandtl-Glauert transformation (e.g., [25]) yields for this problem the formulation

$$(6.1a) \quad u_x + v_y = 0,$$

$$(6.1b) \quad u_y - v_x = 0, \quad y > 0, \quad -\infty < x < \infty,$$

with boundary conditions

$$(6.2a) \quad \begin{aligned} v(x, 0) &= -4x, & -0.5 < x < 0.5, \\ &= 0, & 0.5 < |x|, \end{aligned}$$

and

$$(6.2b) \quad u, v \rightarrow 0, \quad x^2 + y^2 \rightarrow \infty.$$

The analytical solution to (6.1), (6.2) is well known and contains a logarithmic singularity at the leading and trailing edges of the airfoil, $x = \pm 0.5, y = 0$. It is most easily written as

$$(6.3) \quad w = \frac{4}{\pi} \left\{ 1 - z \log \frac{z + 0.5}{z - 0.5} \right\},$$

with $w = u - iv$ and $z = x + iy$. The most important quantity one wishes to compute is u on the airfoil $\{(x, y): y = 0, -0.5 < x < 0.5\}$; from it one can obtain the lift.

The exact u there, given by (6.3), is simply

$$(6.4) \quad u(x, 0) = \frac{4}{\pi} \left\{ 1 - x \log \left| \frac{x + 0.5}{x - 0.5} \right| \right\}.$$

In [24] the numerical computation is carried out in a rectangle $R_3 = \{(x, y): 0 < y < 2, -1 < x < 1\}$, which we also did. Computations were performed in this rectangle prescribing the exact, analytic solution as boundary data on the three sides $\{x = \pm 1\}$, $\{y = 2\}$, as well as prescribing homogeneous boundary conditions there. Clearly, it is a matter of choice which function, u or v , is prescribed on the sides of

the rectangle not containing the airfoil. Hence, we used both the solvers of Section 5, Case 1, as well as Case 2.

Among the solvers of [26], we chose for comparison purposes version D, option (d). Version D is the one suggested by the authors themselves for the application at hand, and its option (d) seemed, of all the versions and options offered, to be the only one which could be second-order accurate. This version corresponds to our Case 1 in the choice of boundary conditions.

For the test case (6.1), (6.2) neither the norms of the solution over R_3 , nor the order of accuracy of the solvers is particularly relevant, because of the singularities at the edges of the airfoil, $x = \pm 0.5, y = 0$. As a means of comparison we chose, therefore, the accuracy in computing $u(x, 0)$. To compute U on $y = 0$, which is a V -line and not a U -line in the staggered mesh, [24] suggests the second-order accurate extrapolation formula

$$(6.5) \quad U_{i,1/2} = (1/8)\{9U_{i,1} - U_{i,2} + 3(k/h)(V_{i,0} - V_{i+1,0}) - 3kE_{i,0}\};$$

we used this formula in our program, as well as in theirs.

Because of the singularity at the edges, [24] did computations both with $x = \pm 0.5$ being V -lines, i.e., with computational mesh points *coinciding* with the edges, and with $x = \pm 0.5$ being U -lines, i.e., with the computational mesh *straddling* the edges. In Table VII we give results for the computation with both these meshes.

For each mesh, Table VII contains in successive columns the x -coordinates of the points along $y = 0$ at which U was calculated, the exact u there, the solution by the [26] solver, version D, option (d), and its error, the solution by our solver Case 1, its error, and finally the solution by our solver Case 2, with its error. We were only interested in comparing the different solvers, and did not wish to study the question of computing in a finite domain the solution to a half-plane problem; hence, only the comparison when using exact boundary data is given. The computations with homogeneous boundary data were carried out and gave slightly poorer results for all solvers.

For the mesh points *coinciding* with the edges, our Case 1 solver has a smaller error than the [26] solver at all points except four. The error, in particular, is smaller by a considerable factor over the airfoil $|x| < 0.5$, and it is smaller at the edges, where most of the error occurs. Our Case 2 solver has mostly smaller error than the Case 1 solver, but they are quite comparable.

For the computational mesh *straddling* the edges, our Case 1 solver seems to have mostly larger errors than [26] outside the airfoil, $0.5 < |x|$, but smaller errors inside. The Case 2 solver is still slightly better than the Case 1 solver, except at a few points.

As a conclusion, our solvers of Section 5, Case 1 and Case 2, are quite successful on the example problem of [24]. If anything, they have slightly smaller error than the [26] solver which appeared most promising. A better test would probably be to extract the singularities from the solution analytically (cf. the example in [13], for instance) and compute only the regular part of the solution. We felt, however, that [24] stressed the importance of computing the neighborhood of the singularities as accurately as possible, and made the comparison accordingly.

TABLE VII

Numerical results for the u component of velocity along $y = 0$ in the solution of the example problem (6.1), (6.2). The heading LM indicates results with the solver of [26], version D, option (d); headings GB uv and GB uu indicate results with the solvers of Section 5, Cases 1 and 2, respectively. The solutions were obtained on a 64×64 grid; every second point is given for reasons of space economy, except near the singularity at $x = \pm 0.5$, where every point is given; the "extra" points are marked by stars. The terms "coinciding mesh" and "straddling mesh" are explained in the text.

VIIA. Coinciding mesh

x	u exact	LM	LM error	GB uv	GB uv error	GB uu	GB uu error
-0.953125	-1.409-1	-1.7411-1	1.962-4	-1.411-1	1.612-4	-1.411-1	1.595-4
-0.890625	-1.666-1	-1.670-1	4.049-4	-1.670-1	3.346-4	-1.670-1	3.311-4
-0.828125	-2.009-1	-2.016-1	6.647-4	-2.015-1	5.586-4	-2.015-1	5.533-4
-0.765625	-2.486-1	-2.497-1	1.044-3	-2.496-1	9.012-4	-2.496-1	8.942-4
-0.703125	-3.193-1	-3.210-1	1.703-3	-3.208-1	1.522-3	-3.208-1	1.513-3
* -0.640625	-4.341-1	-4.372-1	3.083-3	-4.370-1	2.864-3	-4.370-1	2.853-3
* -0.609375	-5.243-1	-5.286-1	4.290-3	-5.283-1	4.051-3	-5.283-1	4.039-3
* -0.578125	-6.588-1	-6.640-1	5.230-3	-6.637-1	4.970-3	-6.637-1	4.957-3
* -0.546875	-8.895-1	-8.863-1	-3.208-3	-8.860-1	-3.488-3	-8.860-1	-3.503-3
* -0.515625	-1.467	-1.684	2.166-1	-1.684	2.163-1	-1.684	2.163-1
* -0.484375	-1.282	-1.480	1.983-1	-1.480	1.980-1	-1.480	1.980-1
* -0.453125	-4.646-1	-4.560-1	-8.664-3	-4.556-1	-9.009-3	-4.556-1	-9.027-3
* -0.421875	-5.250-2	-5.493-2	2.434-3	-5.456-2	2.066-3	-5.454-2	2.047-3
-0.390625	2.302-1	2.277-1	2.490-3	2.281-1	2.099-3	2.281-1	2.078-3
-0.328125	6.163-1	6.150-1	1.323-3	6.154-1	8.841-4	6.155-1	8.606-4
-0.265625	8.729-1	8.704-1	8.401-4	8.725-1	3.496-4	8.726-1	3.228-4
-0.203125	1.050	1.050	6.658-4	1.050	1.209-4	1.050	9.045-5
-0.140625	1.170	1.169	6.182-4	1.170	1.488-5	1.170	-1.953-5
-0.078125	1.242	1.241	6.331-4	1.242	-3.309-5	1.242	-7.198-5
-0.015625	1.272	1.271	6.869-4	1.272	-4.743-5	1.272	-9.136-5
0.046875	1.262	1.261	7.729-4	1.262	-3.587-5	1.262	-8.551-5
0.109375	1.211	1.210	8.950-4	1.211	4.145-6	1.211	-5.196-5
0.171875	1.116	1.115	1.069-3	1.117	8.743-5	1.116	2.391-5
0.234375	9.698-1	9.684-1	1.338-3	9.695-1	2.539-4	9.696-1	1.818-4
0.296875	7.566-1	7.548-1	1.812-3	7.560-1	6.116-4	7.560-1	5.297-4
0.359375	4.450-1	4.422-1	2.796-3	4.435-1	1.463-3	4.436-1	1.369-3
* 0.390625	2.302-1	2.266-1	3.580-3	2.280-1	2.174-3	2.281-1	2.074-3
* 0.421875	-5.250-2	-5.613-2	3.635-3	-5.464-2	2.149-3	-5.454-2	2.042-3
* 0.453125	-4.646-1	-4.573-1	-7.346-3	-4.557-1	-8.918-3	-4.556-1	-9.033-3
* 0.484375	-1.282	-1.482	1.998-1	-1.480	1.981-1	-1.480	1.980-1
* 0.515625	-1.467	-1.685	2.182-1	-1.684	2.164-1	-1.684	2.163-1
* 0.546875	-8.895-1	-8.880-1	-1.490-3	-8.862-1	-3.367-3	-8.860-1	-3.511-3
* 0.578125	-6.588-1	-6.659-1	7.102-3	-6.639-1	5.103-3	-6.637-1	4.948-3
0.609375	-5.243-1	-5.306-1	6.330-3	-5.285-1	4.197-3	-5.283-1	4.029-3
0.671875	-3.689-1	-3.736-1	4.675-3	-3.711-1	2.231-3	-3.709-1	2.031-3
0.734375	-2.802-1	-2.844-1	4.205-3	-2.816-1	1.374-3	-2.814-1	1.134-3
0.796875	-2.227-1	-2.270-1	4.292-3	-2.237-1	9.723-4	-2.234-1	6.777-4
0.859375	-1.825-1	-1.872-1	4.720-3	-1.833-1	7.678-4	-1.829-1	3.980-4
0.921875	-1.529-1	-1.584-1	5.454-3	-1.536-1	6.693-4	-1.531-01	1.902-4

General Purpose Comparison. Lomax and Martin developed their solvers [24], [26] with certain aerodynamical problems in mind [25], and we developed ours bearing in mind certain problems in geophysical fluid dynamics [15], [17]. On the other hand, the first Poisson solvers were also formulated for specific applications [4], [21], and only later developed into general purpose algorithms and into packages. Therefore, it seemed reasonable to test our solvers on solutions which had an appropriately general character (Tables III through VI, and discussion at the end of Section 5); these tests showed that our solvers are second-order accurate, given rather minimal continuity properties of the solution.

VIII. Straddling mesh

x	u exact	LM	LM error	GB ^{uv}	GB ^{uv} error	GB ^{uu}	GB ^{uu} error
-0.9375	-1.467-1	-1.467-1	-3.986-5	-1.467-1	-6.063-5	-1.467-1	-5.903-5
-0.8750	-1.743-1	-1.742-1	-1.054-4	-1.741-1	-1.474-4	-1.741-1	-1.442-4
-0.8125	-2.114-1	-2.112-1	-2.127-4	-2.111-1	-2.765-4	-2.111-1	-2.717-4
-0.7500	-2.637-1	-2.633-1	-4.073-4	-2.632-1	-4.939-4	-2.632-1	-4.874-4
-0.6875	-3.425-1	-3.417-1	-8.356-4	-3.416-1	-9.463-4	-3.416-1	-9.380-4
-0.6250	-4.753-1	-4.730-1	-2.236-3	-4.729-1	-2.372-3	-4.729-1	-2.362-3
* -0.59375	-5.840-1	-5.793-1	-4.734-3	-5.791-1	-4.884-3	-5.791-1	-4.873-3
-0.5625	-7.559-1	-7.417-1	-1.422-2	-7.415-1	-1.439-2	-7.415-1	-1.437-2
* -0.53125	-1.092	-1.020	-7.206-2	-1.020	-7.224-2	-1.020	-7.223-2
-0.5000							
* -0.46875	-7.763-1	-7.027-1	-7.353-2	-7.025-1	-7.374-2	-7.025-1	-7.373-2
-0.4375	-2.353-1	-2.219-1	-1.338-2	-2.217-1	-1.361-2	-2.217-1	-1.359-2
* -0.40625	9.975-2	1.036-1	-3.796-3	1.038-1	-4.037-3	1.038-1	-4.019-3
-0.3750	3.441-1	3.455-1	-1.406-3	3.458-1	-1.665-3	3.458-1	-1.646-3
-0.3125	6.898-1	6.900-1	-1.765-4	6.903-1	-4.726-4	6.903-1	-4.505-4
-0.2500	9.235-1	9.234-1	1.697-4	9.237-1	-1.669-4	9.237-1	-1.417-4
-0.1875	1.085	1.085	3.290-4	1.085	-5.183-5	1.085	-2.327-5
-0.1250	1.192	1.192	4.268-4	1.192	-2.463-6	1.192	2.980-5
-0.0625	1.253	1.253	4.998-4	1.253	1.715-5	1.253	5.376-5
0.0	1.273	1.273	5.610-4	1.273	1.928-5	1.273	6.068-5
0.0625	1.253	1.253	6.144-4	1.253	6.971-6	1.253	5.376-5
0.1250	1.192	1.191	6.578-4	1.192	-2.301-5	1.192	2.990-5
0.1875	1.085	1.084	6.804-4	1.085	-8.314-5	1.085	-2.327-5
0.2500	9.235-1	9.229-1	6.476-4	9.237-1	-2.095-4	9.237-1	-1.417-4
0.3125	6.898-1	6.894-1	4.365-4	6.903-1	-5.274-4	6.903-1	-4.505-4
0.3750	3.441-1	3.448-1	-6.466-4	3.459-1	-1.733-3	3.458-1	-1.646-3
* 0.40625	-2.353-1	1.027-1	-2.957-3	1.039-1	-4.112-3	1.038-1	-4.019-3
0.4375	-2.353-1	-2.228-1	-1.246-2	-2.216-1	-1.369-2	-2.217-1	-1.359-2
* 0.46875	-7.763-1	-7.037-1	-7.252-2	-7.024-1	-7.383-2	-7.025-1	-7.373-2
0.5000							
* 0.53125	-1.092	-1.021	-7.086-2	-1.019	-7.235-2	-1.020	-7.223-2
0.5625	-7.559-1	-7.430-1	-1.291-2	-7.414-1	-1.450-2	-7.415-1	-1.437-2
* 0.59375	-5.840-1	-5.807-1	-3.304-3	-5.790-1	-5.011-3	-5.791-1	-4.873-3
0.6250	-4.753-1	-4.746-1	-6.781-4	-4.727-1	-2.510-3	-4.729-1	-2.362-3
0.6875	-3.425-1	-3.435-1	1.013-3	-3.414-1	-1.108-3	-3.416-1	-9.380-4
0.7500	-2.637-1	-2.655-1	1.796-3	-2.630-1	-6.830-4	-2.632-1	-4.874-4
0.8125	-2.114-1	-2.138-1	2.432-3	-2.109-1	-4.957-4	-2.111-1	-2.717-4
0.8750	-1.743-1	-1.774-1	3.105-3	-1.739-1	-3.994-4	-1.741-1	-1.442-4
0.9375	-1.467-1	-1.506	3.911-3	-1.464-1	-3.457-4	-1.467-1	-5.903-5

The intent of [24], [25], [26] was explicitly restricted to the solution of specific aerodynamic problems. It appeared worthwhile, however, to consider the more general applicability of their solvers.

We carried out a number of tests with version D, option (d) of [26] on solutions with a generic character. The results are given in Table VIII. This table is organized in the same way as Tables III and V of Section 4, and we refer to the description there.

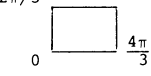
For very simple test cases such as u , v constant or quadratic, the [26] solver is essentially exact to machine accuracy; the round-off error increases slightly with the number of grid points used. The difference between these results and those in Table III is due to the computer used: double-precision arithmetic on an IBM 360/95 is slightly more accurate than single precision on a CDC 6600.

The results with more severe test cases seem to indicate that the [26] solver is only *first-order accurate* in general. This is apparently due to the formulation of the algorithm at the boundaries. An indication that boundary inaccuracies are the cause of lower than second-order accuracy are those exceptional test cases which show high

TABLE VIII

Numerical results for general-purpose test cases using the solver of [26], version D, option (d). The smaller regions, such as $R_4 = \{(x, y): 0 < x < \pi, 0 < y < \pi/2\}$ were used in some of the tests because of the difficulty of fitting computations with $M = 256$ into the core memory of the CDC 6600; M is the actual number of grid points used in the x -direction in each test.

	LM	M	$\ell_2(u)$	$\ell_2(v)$	$\ell_2(u,v)$	$\ell_\infty(u)$	$\ell_\infty(v)$
u, v constant R_4	$u = 1000.0$ $v = -100.0$	16	1.083-11	1.642-12	7.747-12	2.910-11	4.547-12
		32	2.871-11	2.628-11	2.752-11	9.095-11	5.093-11
		64	1.001-10	1.387-10	1.210-10	4.475-10	2.310-10
		128	1.999-10	2.177-10	2.090-10	1.281-9	3.174-10
$u, v \in C^\infty$	$u = x^2 + y^2$ $v = 100$	16	3.076-12	1.537-12	2.431-12	8.640-12	4.547-12
		32	1.360-11	2.533-11	2.033-11	5.571-11	4.957-11
		64	7.589-11	1.376-10	1.111-10	3.583-10	2.287-10
		128	1.706-10	2.164-10	1.949-10	1.122-9	3.156-10
$u, v \in C^\infty$	$u = y \sin(x)$ $v = \sin(y)$	16	4.183-3	4.979-3	4.598-3	9.507-3	1.013-2
		32	1.020-3	1.227-3	1.128-3	2.379-3	2.544-3
		64	2.519-4	3.046-4	2.795-4	5.921-4	6.366-4
		128	6.262-5	7.589-5	6.957-5	1.477-4	1.592-4
		16/32	4.101	4.058	4.076	3.996	3.983
		32/64	4.048	4.023	4.036	4.019	3.996
		64/128	4.023	4.014	4.018	4.009	3.999
$u, v \in C^\infty$	$u = y^2 \sin x$ $v = \sin(y)$	16	1.155-2	8.376-3	1.009-2	2.622-2	1.909-2
		32	2.885-3	2.055-3	2.504-3	6.613-3	4.838-3
		64	7.197-4	5.090-4	6.233-4	1.656-3	1.211-3
		128	1.797-4	1.267-4	1.555-4	4.141-4	3.030-4
		16/32	4.005	4.076	4.029	3.966	3.947
		32/64	4.008	4.037	4.018	3.994	3.993
		64/128	4.005	4.018	4.009	3.999	3.998

	LM	M	$\ell_2(u)$	$\ell_2(v)$	$\ell_2(u,v)$	$\ell_\infty(u)$	$\ell_\infty(v)$
$u, v \in C^\infty$	$u = y^3 \sin(x)$ $v = \sin(y)$	16	2.356-2	3.073-2	2.738-2	8.028-2	6.846-2
		32	5.333-3	7.566-3	6.546-3	2.600-2	1.725-2
		64	1.296-3	1.876-3	1.612-3	7.155-3	4.314-3
		128	3.219-4	4.672-4	4.012-4	1.865-3	1.078-3
		16/32	4.418	4.062	4.183	3.088	3.969
		32/64	4.115	4.032	4.059	3.633	3.998
		64/128	4.025	4.016	4.019	3.836	4.003
$u, v \in C^\infty$	$u = y^4 \sin(x)$ $v = \sin(y)$	16	9.705-2	1.262-1	1.126-1	4.135-1	2.797-1
		32	3.426-2	3.117-2	3.276-2	1.282-1	7.139-2
		64	1.061-2	7.741-3	9.286-3	3.417-2	1.787-2
		128	2.938-3	1.928-3	2.485-3	8.742-3	4.470-3
		16/32	2.832	4.047	3.436	3.224	3.919
		32/64	3.230	4.027	3.527	3.753	3.995
		64/128	3.610	4.014	3.737	3.909	3.998
$u, v \in C^\infty$	$u = y^5 \sin(x)$ $v = \sin(y)$ 	16	6.019-2	5.031-2	5.547-2	1.217-1	1.052-1
		32	3.206-2	1.277-2	2.440-2	6.215-2	2.712-2
		64	1.058-2	3.206-3	7.818-3	1.982-2	6.805-3
		128	2.988-3	8.026-4	2.187-3	5.511-3	1.703-3
		16/32	1.877	3.939	2.273	1.959	3.878
		32/64	3.030	3.983	3.122	3.135	3.985
		64/128	3.542	3.995	3.574	3.597	3.996

accuracy, apparently because their solutions close to the boundary behave in a special way in the variable perpendicular to it. The series of tests with increasing powers of

TABLE VIII (Continued)

	LM	M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U, V)$	$\ell_\infty(U)$	$\ell_\infty(V)$
$u, v \in C^\infty$	$u = y^6 \sin(x) + 0.1$ $v = \sin(y) + 0.1$	16	1.726	1.937	1.835	6.229	4.615
		32	1.002	4.854-1	7.876-1	2.209	1.182
		64	3.696-1	1.210-1	2.750-1	6.000-1	2.969-1
		128	1.087-1	3.018-2	7.981-2	1.750-1	7.427-2
		16/32	1.722	3.990	2.330	2.820	3.904
		32/64	2.712	4.012	2.864	3.682	3.980
		64/128	3.399	4.010	3.446	3.429	3.998
$u, v \in C^\infty$	$u = y^{10} \sin(x)$ $v = \sin(y)$	16	1.099+3	3.209+2	8.115+2	1.944+3	8.179+2
		32	1.740+2	8.664+1	1.374+2	4.049+2	2.312+2
		64	1.065+2	2.186+1	7.685+1	1.687+2	5.875+1
		128	3.603+1	5.470	2.577+1	5.635+1	1.474+1
		16/32	6.320	3.796	5.905	4.800	3.538
		32/64	7.634	3.963	1.788	2.400	3.935
		64/128	2.950	3.997	2.982	2.994	3.985
$u, v \in C^\infty$	$u = y^2 \sin(x)$ $v = \sin(y^2)$	16	5.742-2	1.128-1	8.951-2	2.299-1	3.015-1
		32	1.556-2	2.690-2	2.197-2	8.305-2	7.070-2
		64	3.835-3	6.571-3	5.379-3	2.533-2	1.892-2
		128	9.507-4	1.632-3	1.335-3	7.042-3	4.703-3
		16/32	3.690	4.195	4.074	2.768	4.264
		32/64	4.058	4.094	4.084	3.279	3.737
		64/128	4.033	4.027	4.029	3.597	4.023
$u, v \in C^\infty$	$u = y^4 \sin(x)$ $v = \sin(y^4)$	16	1.261+1	1.892+1	1.608+1	4.736+1	3.853+1
		32	6.517	1.044+1	8.703	3.188+1	2.896+1
		64	4.913	7.542	6.365	2.726+1	1.494+1
		128	2.752	3.623	3.217	1.833+1	1.026+1
		16/32	1.935	1.812	1.847	1.486	1.331
		32/64	1.326	1.384	1.367	1.169	1.939
		64/128	1.785	2.0828	1.978	1.487	1.456

	LM	M	$\ell_2(U)$	$\ell_2(V)$	$\ell_2(U, V)$	$\ell_\infty(U)$	$\ell_\infty(V)$
$u, v \in C^\infty$	$u = x \sin(y)$ $v = y \sin(x)$	16	1.938-1	6.645-2	1.448-1	8.011-1	2.812-1
		32	1.087-1	3.787-2	8.140-2	5.367-1	1.872-1
		64	5.845-2	2.017-2	4.372-2	3.371-1	1.108-1
		128	3.049-2	1.041-2	2.278-2	2.028-1	6.175-2
		16/32	1.782	1.754	1.779	1.493	1.502
		32/64	1.860	1.878	1.862	1.592	1.689
		64/128	1.917	1.939	1.919	1.662	1.795
$u, v \in C^\infty$	$u = \sin(y)$ $v = y^4 \sin(x)$	16	2.733	6.771-1	1.991	1.055+1	4.096
		32	1.787	4.29-1	1.300	8.858	3.456
		64	1.033	2.413-1	7.501-1	6.367	2.401
		128	5.604-1	1.281-1	4.065-1	4.193	1.505
		16/32	1.529	1.578	1.532	1.191	1.185
		32/64	1.730	1.778	1.732	1.391	1.439
		64/128	1.843	1.884	1.845	1.519	1.595
$u, v \in C^\infty$	$u = x^2 \sin(y)$ $v = y^2 \sin(x)$	16	4.381-1	1.319-1	3.235-1	1.738	6.385-1
		32	2.571-1	7.774-2	1.899-1	1.227	4.722-1
		64	1.429-1	4.213-2	1.054-1	8.254-1	3.017-1
		16/32	1.704	1.697	1.704	1.416	1.350
		32/64	1.798	1.845	1.802	1.487	1.565

y in the solution, which lead to decreasing order of accuracy, illustrate this point. The fractional order of accuracy evident in these and in some of the other test cases also points in this direction. Needless to say, we also performed experiments with our solvers of Sections 3 and 5 on the same test cases; they all yielded second-order accurate results.

TABLE VIII (Continued)

	LM	M	$\ell_2(u)$	$\ell_2(v)$	$\ell_2(u,v)$	$\ell_\infty(u)$	$\ell_\infty(v)$	
u, v harmonic R_4	u=cos(x)sinh(y) v=sin(x)cosh(y)	16	7.984-2	2.924-2	6.013-2	3.669-1	1.105-1	
		32	4.464-2	1.647-2	3.365-2	2.559-1	7.177-2	
		64	2.379-2	8.727-3	1.792-2	1.050-1	4.248-2	
		128	1.234-2	4.491-3	9.285-3	1.012-1	2.383-2	
		16/32	1.789	1.775	1.787	1.434	1.539	
		32/64	1.876	1.888	1.878	1.551	1.690	
		64/128	1.928	1.943	1.930	1.630	1.783	
	u, v harmonic $R_4/4$	u=cos(x)sinh(y) v=sin(x)cosh(y)	16	9.154-3	3.840-3	7.019-3	4.605-2	1.348-2
			32	4.861-3	2.110-3	3.747-3	3.087-2	7.809-3
			64	2.515-3	1.105-3	1.943-3	1.938-2	4.181-3
128			1.283-3	5.653-4	9.914-4	1.166-2	2.168-3	
		16/32	1.883	1.820	1.873	1.492	1.726	
		32/64	1.932	1.910	1.929	1.593	1.868	
		64/128	1.961	1.954	1.950	1.661	1.928	
u, v harmonic		u=sin(y)cosh(x) v=cos(y)sinh(x)	16	1.143+1	3.983	8.557	5.535+1	1.776+1
			32	7.065	2.569	5.315	4.188+1	1.457+1
			64	3.951	1.463	2.979	2.760+1	9.492
	128		2.101	7.809-1	1.585	1.696+1	5.490	
		16/32	1.618	1.550	1.610	1.322	1.219	
		32/64	1.788	1.756	1.784	1.517	1.535	
		64/128	1.888	1.873	1.879	1.627	1.745	
	u, v harmonic	u=cos(y)sinh(x) v = - sin(y) cosh(x)	16	1.742+1	8.394	1.367+1	7.865+1	3.313+1
			32	8.911	4.681	7.117	4.542+1	2.071+1
			64	4.500	2.476	3.632	2.443+1	1.165+1
128			2.260	1.274	1.835	1.264+1	6.183	
		16/32	1.955	1.793	1.921	1.731	1.600	
		32/64	1.980	1.891	1.960	1.860	1.779	
		64/128	1.991	1.944	1.970	1.928	1.883	

Subject to further testing, we are forced to conclude at this point that our algorithms, compared to previously published ones, are at least as good when applied to specific problems of interest, and that they are more suitable for development into general-purpose Cauchy-Riemann solvers.

7. Concluding Remarks. The inhomogeneous Cauchy-Riemann equations in a rectangle have been discretized by a finite-difference approximation. A number of different boundary conditions have been treated explicitly, leading to algorithms which have overall second-order accuracy. All boundary conditions with either u or v prescribed along a side of the rectangle can be treated by similar methods. A rigorous proof of the second-order accuracy of the algorithm was given for one combination of boundary conditions, and numerical experiments substantiate this result for all the boundary conditions tested.

The algorithms presented here have nearly minimal time and storage requirements and seem suitable for development into a general-purpose direct Cauchy-Riemann solver for arbitrary boundary conditions. This could be done for instance along the lines of the capacitance matrix methods of discrete potential theory (Widlund [33]); generalizations to nonrectangular domains can also be made by this approach and related ones. More experience with different applications should help in formulating a code which gives a reasonable compromise between efficiency and range of applicability.

It is well known [30], [31], [32] that fast solvers can be formulated for a single separable second-order elliptic equation with variable coefficients. Clearly, the

same generalizations can be carried out for a first-order system (2.1) in which $\partial/\partial x$ and $\partial/\partial y$ are replaced by $a(x)\partial/\partial x$ and by $b(y)\partial/\partial y$, and in which lower-order terms can also be introduced. A special case of such an extension appears in [26]. Efficient algorithms for generalizations of this nature can be based on appropriate modifications and combinations of matrix decomposition, cyclic reduction and Toeplitz factorization [30], [31], [32].

Linear problems with variable coefficients, as well as nonlinear problems, can also be handled by semidirect methods, i.e., by splitting the given operator to be inverted into one whose inverse is easily computed using a fast direct method, and another one which is small in some suitable sense [8], [19], [25], [32]. It is in this direction that we shall seek to extend the work presented here, in order to solve the nonlinear problem of [15], [17] and related ones. The straightforward iterative method of [17] will then be compared to semidirect methods.

Appendix A. An Error Estimate. We saw in Sections 4 and 5 that the proposed algorithm gives a second-order accurate solution to the inhomogeneous Cauchy-Riemann equations under the various boundary conditions considered. The second-order accuracy of the discrete solutions was obtained in our numerical experiments even for cases in which the solution to the continuous problem was merely C^1 , more precisely, $u \in C^1$, $v \in C^0$. We shall give now a rigorous error estimate for the model problem of Section 2, making the stronger and more customary assumption that $u, v \in C^4$, i.e., that the solution to (2.1)–(2.3) has continuous derivatives up to fourth order.

The discrete operator L_h we consider is that of (3.11 α),

$$(A.1) \quad L_h = \begin{bmatrix} S+I & -I & & & & & & & & & & \\ & -I & S+2I & -I & & & & & & & & \\ & & & \cdot & \cdot & \cdot & & & & & & \\ & & & & & & \cdot & & & & & \\ & & & & & -I & S+2I & -I & & & & \\ & & & & & & & -I & S+I & & & \end{bmatrix} + \alpha \epsilon \epsilon^*,$$

where $\alpha > 0$ and ϵ is a vector of length MN with all entries zero but one,

$$\epsilon_l = 0 \quad \text{for } l \neq Mj_0 + i_0, \quad \epsilon_{Mj_0 + i_0} = 1;$$

L_h acts on the grid function U . We wish to show that

$$(A.2a) \quad \|u - U\|_\infty = O(h^2 + k^2).$$

From such an estimate and from (3.1a) it will follow immediately that

$$(A.2b) \quad \|v - V\|_\infty = O(h^2 + k^2)$$

as well; it suffices to observe that (3.1a) is a second-order accurate quadrature formula for

$$(A.3) \quad v(x, y) = v^{(0)}(x) + \int_0^y \{d(x, \eta) - u_x(x, \eta)\} d\eta,$$

namely the midpoint rule. We only need to apply the result of Bramble and Hubbard [2] that (A.2a) implies similar estimates for the difference quotients of u .

The matrix operator L_h of (A.1) is of *monotonic type* (Collatz, [6, p. 42 ff.]) or of *positive type* with diagonal dominance (Forsythe and Wasow [11, p. 181]). To avoid confusion in the terminology, we shall simply state that L_h satisfies the following

LEMMA 1 (MAXIMUM PRINCIPLE). *If $L_h W = H$, and $H \geq 0$ (in the sense that all the components of H are nonnegative), then $W \geq 0$.*

From this, one easily shows that

LEMMA 2 (COMPARISON THEOREM). *If $|L_h W| \leq L_h \Phi$, then $|W| \leq \Phi$.*

We notice that these results would still hold if, instead of $\alpha\epsilon\epsilon^*$, we included in L_h a term αA , with a single diagonal block equal to I_M , and all other blocks zero, I_M being the $M \times M$ identity matrix.

These properties of L_h suggest the familiar estimation procedure first used by Gershgorin [13]. Let u be the solution of

(A.4a)
$$Lu = f,$$

where L and f are defined by the two equations

(A.4b)
$$\Delta u \equiv u_{xx} + u_{yy} = d_x + e_y \quad \text{in } R,$$

(A.4c)
$$u_y = e + v_x \quad \text{on } \partial R \equiv \{(x, y): 0 < x < 2\pi, y = 0, \pi\},$$

with the additional requirements that u be 2π -periodic in x , and given at a point $(x_0, y_0) \in R$. For sufficiently smooth d, e and $v^{(0)}, v^{(1)}$, (A.4) has a unique solution $u \in C^4$, subject to the familiar compatibility condition for the Neumann problem that

(A.4d)
$$\iint_R (d_x + e_y) dx dy = -\oint_{\partial R} (e + v_x) dx.$$

We shall make the necessary smoothness and compatibility assumptions throughout this Appendix.

Let U be the solution of

(A.5a)
$$L_h U = F,$$

where L_h is defined by (A.1) and F is defined as the right-hand side of (3.11a),

(A.5b)
$$F = k \begin{bmatrix} T^*D_1 - E_1 \\ T^*D_2 + E_1 - E_2 \\ \dots\dots\dots \\ T^*D_{N-1} + E_{N-2} - E_{N-1} \\ T^*D_N + E_{N-1} \end{bmatrix} + \alpha\epsilon\epsilon^*U,$$

with $F_j, 1 \leq j \leq N$, corresponding in obvious fashion to the partition of L_h . Clearly, for $2 \leq j \leq N - 1$, (A.5) is an approximation to (A.4b) with second-order local truncation error. This would further encourage us to seek an estimate (A.2a) by first showing that the truncation error satisfies

(A.6)
$$L_h(u - U) = (L_h - L)u + f - F = O(k^2(h^2 + k^2))$$

and then constructing a Gershgorin comparison function Φ which would allow us to

conclude based on Lemma 2 that (A.2) follows from (A.6). We remark at this point that, in order to make the notation of (A.6) transparent, we defined

$$(A.7a) \quad L = -k^2 \Delta \quad \text{in } R, \quad L = -k \partial / \partial y \quad \text{on } \partial R,$$

$$(A.7b) \quad f = -k^2 (d_x + e_y) \quad \text{in } R, \quad f = -k(e + v_x) \quad \text{on } \partial R;$$

furthermore, F is not merely f at the grid points.

A number of slight difficulties arise in carrying out the program above. First, (A.4) is essentially a Neumann problem, rather than a Dirichlet problem as in [13] and in most of the literature on elliptic systems. The work on boundary conditions of the second and third kind most relevant to the estimation which follows is that of Batschelet [1], who gives an $O(h)$ estimate, and that of Bramble and Hubbard [3], who give an $O(h^2 |\log h|)$ estimate, their assumptions being that $u \in C^4$. The latter article also contains further references to estimates for the Neumann problem. The boundary condition approximations in these works are different from ours.

Second, (A.6) actually fails close to the boundary, i.e., for $j = 1, N$, where the local truncation error is of first order only. The work of Bramble and Hubbard ([3] and references therein), combined with our numerical results, led us to expect that (A.2) could still be proved, with some additional effort; this turned out to be the case.

Our method to obtain (A.2) is actually a modification of that in [1], which exploits the fact that the boundary condition we use is second-order accurate, while that in [1] is only first-order. We shall see below that the failure of (A.6) for $j = 1, N$ stems from L_h there being a linear combination of the discrete analog to (A.4b) and of that to (A.4c). Hence the *truncation error*, as defined by (A.6), is of first order there, although both (A.4b) and (A.4c) are separately approximated to second order. In spite of this formally first-order truncation error, the fact that both the equation and the boundary condition are approximated by second-order discrete analogs yields an overall $O(h^2 + k^2)$ error estimate (A.2).

After these observations, we proceed with the business at hand. To start, we derive (A.6) for $2 \leq j \leq N - 1$; let the *discretization error* W be defined as

$$(A.8a) \quad W = u - U,$$

considered as a mesh function. Then

$$(A.8b) \quad \begin{aligned} (L_h W)_j &\equiv -W_{j-1} + (S + 2I)W_j - W_{j+1} \\ &= -\mathbf{u}_{j-1} + (S + 2I)\mathbf{u}_j - \mathbf{u}_{j+1} + k^2(\Delta \mathbf{u})_j \\ &\quad - k^2(d_x + e_y) - k(T^* \mathbf{D}_j + \mathbf{E}_{j-1} - \mathbf{E}_j) \\ &= O(k^2(h^2 + k^2)), \quad 2 \leq j \leq N - 1. \end{aligned}$$

In (A.8b), and in the sequel, \mathbf{u}_j , $(\Delta \mathbf{u})_j$, and similar terms are interpreted as vectors of grid values, in the same way as \mathbf{U}_j . The factor k^2 is convenient in order to keep the coefficients of L_h as $O(1)$. The terms $O(h^2 k^2)$ appear due to differentiation in the x direction, those $O(k^4)$ due to differentiation in the y direction. The constants implied

in writing $O(h^p k^q)$ depend on the derivatives of order $p + q$ for the functions involved; they will not be written down explicitly, since those derivatives are known to be bounded from our assumptions.

Here, as in the derivation of (A.2b) from (A.3) and as in the sequel, it is important to remember the staggering of Figure 1, which essentially guarantees that all differences are centered. Notice also that, for $\epsilon_{j_0} \neq 0$, (A.8b) will only be modified by the term $\alpha \epsilon_{j_0} \epsilon_{j_0}^* \mathbf{u}_{j_0}$ in $L_n u$, and by the term $\alpha \epsilon_{j_0} \epsilon_{j_0}^* \mathbf{U}_{j_0}$ in F ; the condition we chose to eliminate the indeterminacy in the Neumann problem is exactly that these two terms cancel, and hence the estimate is not affected. If instead of $\epsilon \epsilon^*$ we have A , the same observation holds as after Lemma 2; i.e., prescribing the average of \mathbf{U}_{j_0} , rather than one of its components, U_{i_0, j_0} , does not affect the estimate either.

To analyze the situation for $j = 1$, it is convenient to rewrite (A.5) for $j = 1$ as a linear combination of two equations, involving an auxiliary vector \mathbf{U}_0 ; an entirely similar procedure can be carried out for $j = N$, introducing \mathbf{U}_{N+1} , but we shall omit writing out the latter analysis and merely draw the conclusions we need from it. The two equations for $j = 1$ are

$$(A.9a) \quad -\mathbf{U}_0 + (S + 2I)\mathbf{U}_1 - \mathbf{U}_2 = k(T^* \mathbf{D}_1 + \mathbf{E}_0 - \mathbf{E}_1),$$

$$(A.9b) \quad \mathbf{U}_0 - \mathbf{U}_1 = -k\mathbf{E}_0 + T^* \mathbf{V}_0;$$

here we revert to the original definition of \mathbf{D}_1 , which had been replaced by $\mathbf{D}_1 + k^{-1} \mathbf{V}_0$ in writing Eq. (3.3). Equation (A.9a) is now simply the discrete analog of (A.4b) for $j = 1$, or $y = k/2$, with \mathbf{E}_0 defined on $y = 0$ in the usual manner. Equation (A.9b) is the discrete analog of (A.4c) written on $y = 0$. If the auxiliary vector \mathbf{U}_0 is thought of as given on $y = -k/2$, then both (A.9a) and (A.9b) are formally second-order accurate.

With this motivation in mind, it is easy to obtain

$$\begin{aligned} (L_n W)_1 &\equiv (S + I)W_1 - W_2 \\ &= (S + I)\mathbf{u}_1 - \mathbf{u}_2 + k^2 \Delta u|_{y=k/2} + k(\partial/\partial y)u|_{y=0} \\ (A.10a) \quad &-k^2(d_x + e_y)|_{y=k/2} - k(T^* \mathbf{D}_1 + \mathbf{E}_0 - \mathbf{E}_1) \\ &-k(e + v_x)|_{y=0} + k\mathbf{E}_0 - T^* \mathbf{V}_0 \\ &\equiv [(L_n - L)u]_1 + [f - F]_1. \end{aligned}$$

We have from (A.10a) that

$$[(L_n - L)u]_1 = O(k^2(h^2 + k)), \quad [f - F]_1 = O(k^2(h^2 + k^2)) + O(kh^2).$$

We assume throughout that $k/h = O(1)$ and drop the fourth-order terms; thus finally

$$(A.10b) \quad (L_n W)_1 = O(k(h^2 + k^2)).$$

For $j = N$ we obtain in the same way

$$(A.10c) \quad (L_h W)_N = O(k(h^2 + k^2)).$$

At this point we have to exhibit a comparison function ϕ which shall lead us from (A.8) and (A.10) to (A.2). Let ψ be the solution of

$$(A.11a) \quad L\psi = k^2 a \quad \text{in } R,$$

$$(A.11b) \quad L\psi = \begin{cases} -k\alpha b & \text{on } y = 0, \\ k\beta b & \text{on } y = \pi. \end{cases}$$

L is defined by (A.7), and ψ is 2π -periodic in x and fixed at one point; this yields a unique and smooth ψ , which is bounded independently of x, y, h and k . Notice that ψ satisfies inhomogeneous boundary conditions as in [1]. The constants a, b, α and β are positive and chosen so as to satisfy the compatibility condition

$$(A.11c) \quad \iint_R \Delta\psi \, dx \, dy = \int_0^{2\pi} \{\psi_y(x, \pi) - \psi_y(x, 0)\} \, dx,$$

that is, $\pi a = (\alpha + \beta)b$.

We want to find m independent of x, y , but not of k, h , so that

$$(A.12a) \quad \phi = m\psi$$

and

$$(A.12b) \quad |L_h W| \leq L_h \Phi,$$

where W is given by (A.8a) and Φ is the mesh function corresponding to ϕ . Lemma 2 will then provide the desired estimate on W in terms of m .

We consider first the ‘‘interior’’ mesh domain of U -points $R_h = \{(x_i, y_j): 1 \leq i \leq M, 2 \leq j \leq N - 1\}$. There

$$(A.13a) \quad L_h \phi = L\phi + (L_h - L)\phi = mk^2 a + mO(k^2(h^2 + k^2)),$$

as in (A.8b). For h, k sufficiently small, with $h/k = O(1)$,

$$L_h \phi \geq mk^2 a/2.$$

It suffices, therefore, given some sufficiently large constant C_R , to have set

$$(A.14a) \quad m = C_R h^2$$

for (A.12b) to hold in R_h ; C_R will depend on certain bounds for the derivatives of u .

Consider now the ‘‘boundary’’ mesh domain of U -points $\Gamma_h = \{(x_i, y_j): 1 \leq i \leq M, j = 1, N\}$. Here, cf. (A.9, 10), we have

$$(A.13b) \quad \begin{aligned} (L_h \phi)_1 &\equiv (S + I)\Phi_1 - \Phi_2 = k^2 \Delta\phi|_{y=k/2} + k\phi_y|_{y=0} + O(k^2(h^2 + k)) \\ &= -mk^2 a + mk\alpha b + O(k^3); \end{aligned}$$

a similar equation holds for $(L_h \Phi)_N$. Thus, for h, k sufficiently small, we have in Γ_h

$$L_h \phi \geq mk\alpha b/2;$$

remembering again (A.10), it suffices here too that

$$(A.14b) \quad m = C_{\Gamma} h^2,$$

for (A.12b) to hold. The constant C_{Γ} depends only on the derivatives of u , and on the domain R .

In fact, one can write down $\psi = \psi(x, y; a, b, \alpha, \beta)$ explicitly and optimize the parameters a, b, α, β subject to (A.11c). This yields

$$(A.15a) \quad |W| \leq \frac{\pi}{2} h^2 \max \left\{ \frac{\pi}{2} C_R, C_{\Gamma} \right\},$$

where

$$(A.15b) \quad C_R = \frac{1}{24} \sup_{\substack{0 \leq x < 2\pi, \\ 0 < y < \pi}} |2(u_{xxxx} + u_{yyyy}) - (d_{xx} + e_{yy})|,$$

and

$$(A.15c) \quad C_{\Gamma} = \frac{1}{24} \sup_{\substack{0 \leq x < 2\pi, \\ y=0, \pi}} |u_{yy} + v_{xx}|.$$

The bound $e(u)$ in Table V of Section 4 was computed using (A.15).

Having derived bound (A.15) completes the proof that under our assumptions

$$\|u - U\|_{\infty} + \|v - V\|_{\infty} = O(h^2).$$

It might be of interest to notice that the algorithm proposed and the error estimate just given apply not only to the inhomogeneous Cauchy-Riemann equations (2.1), (2.2); they apply directly to the Neumann problem (A.4), with $d_x + e_y, e + v_x$ arbitrarily prescribed as well. In other words, this is also a second-order algorithm for the Poisson equation with Neumann boundary conditions in a rectangle (compare Schumann and Sweet [29]).

Appendix B. The Basic Algorithm. At the end of the paper we present a FORTRAN listing of the program used to compute the results in Tables I through VI. The program implements the algorithm described in Section 3, tested in Section 4, and analyzed in Appendix A. It solves the inhomogeneous Cauchy-Riemann equations for u and v in a rectangle, with v prescribed on the upper and lower side of the rectangle, and with periodicity in the x -direction. The programs for other combinations of boundary conditions, as discussed in Section 5, are very similar.

The program was tested for compatibility with ANSI FORTRAN (U.S.A. Standard X3.9-1966) by a special diagnostic option of the CDC FTN 4.6 compiler; it was further cleaned up to conform to common usage by a program called TIDY, implemented at and available from the Courant Institute. It was run on a CDC 6600 with an FTN 4.6 compiler, and on an IBM 360/95 and an Amdahl 470V/6 with a FORTRAN IV, level H compiler. It is felt that these procedures and the numerical tests reported on in the text are some reasonable steps in the direction of validation and portability; further steps in this direction taken by potential users would be of the greatest interest to the authors.


```

X2=XJV(I)
K=(J-1)*M+I
C
C     SETTING UP R.H.S. OF THE INHOMOGENEOUS CAUCHY-RIEMANN
C     EQUATIONS.
C     E IS THE R.H.S. OF THE VORTICITY EQUATION
C     D IS THE R.H.S. OF THE DIVERGENCE EQUATION
C
E(K)=DUDY(X1,Y1)-DVDX(X1,Y1)
D(K)=DUDX(X2,Y2)+DVDY(X2,Y2)
C
10 CONTINUE
C
C     NEXT STORE THE BOUNDARY VALUES V(TOP) IN VN(I).
C
C     NEXT STORE THE BOUNDARY VALUES V(BOTTOM) IN VO(I).
C
DO 20 I=1,M
X2=XJV(I)
VO(I)=VEX(XJV(I),YU)
VN(I)=VEX(XJV(I),YL)
20 CONTINUE
C
C     NEXT STORE THE FIXED U VALUE IN SOL.
C
C     IROW IS THE ROW WHERE FIXED VALUE U IS ASSIGNED.
SOL=0.0
IROW=N
Y1=G*(2*(IROW-1)+1)+YL
DO 30 J=1,M
C
X1=XJU(J)
SOL=SOL+UEX(X1,Y1)
30 CONTINUE
CALL FASTCR (SOL,H,G,PI,IROW,MPOW,ISAM,ICUDE,N)
C
C     NEXT FINDING NORMS..
C     XN IS THE L2 NORM OF U.
C     YN IS THE L2 NORM OF V.
C     XYN IS THE L2 NORM OF (U+V).
C     XM IS THE MAX NORM OF U.
C     YM IS THE MAX NORM OF V.
C
YM=0.0
YN=0.0
DO 40 J=1,M
X2=XJV(J)
DO 40 I=1,N1
Y2=YKV(I)
K=(I-1)*M+J
DKKK=ABS(D(K)-VEX(X2,Y2))
Y2=YN+DKKK*DKKK
IF (DKKK.GT.YM) YM=DKKK
40 CONTINUE
C
XN=0.0
XM=0.0
DO 50 J=1,M
X1=XJU(J)
DO 50 I=1,N
Y1=YKU(I)
K=(I-1)*M+J
EKKK=ABS(E(K)-UEX(X1,Y1))
XN=XN+EKKK*EKKK
IF (EKKK.GT.XM) XM=EKKK
50 CONTINUE
XYN=(XN+YN)/(IL+ILM)
XN=XN/IL
YN=YN/ILM
XN=SQRT(XN)
YN=SQRT(YN)
XYN=SQRT(XYN)
WRITE (6,120)
WRITE (6,110) XN,YN,XYN,XM,YM
W(1,MF)=XN
W(2,MF)=YN
W(3,MF)=XYN
W(4,MF)=XM
W(5,MF)=YM

```

```

C
WRITE (6,90) A 129
WRITE (6,90) A 130
WRITE (6,80) N,M,XL,XU,YL,YU A 131
WRITE (6,90) A 132
WRITE (6,90) A 133
WRITE (6,90) A 134
60 CONTINUE A 135
WRITE (6,90) A 136
DO 70 J=1,5 A 137
W(J,3)=W(J,3)/W(J,4) A 138
W(J,4)=W(J,4)/W(J,5) A 139
70 W(J,5)=W(J,5)/W(J,6) A 140
WRITE (6,100) A 141
WRITE (6,110) ((W(J,I),J=1,5),I=3,6) A 142
STOP A 143
C A 144
C A 145
80 FORMAT (5X,*N=*,I3,* M=*,I3,5X,*XL=*,F10.3,* XU=*,F10.3,* YL=*, A 146
1F10.3,* YU=*,F10.3) A 147
90 FOKMAT (1X) A 148
100 FORMAT (7X,9HFOLLOWING,4X,2HIS,4X,5HRATIO,4X,2HOF,4X,5HADOVE,4X,5H A 149
1NORMS) A 150
110 FOKMAT (1X,1P5E20.10) A 151
120 FOKMAT (1X,9HNORM OF U,15X,9HNORM OF V,12X,11HNORM OF U+V,10X,11HM A 152
1AX NORM U,10X,11HMAX NORM V) A 153
END A 154-

SUBROUTINE FASTCR (SOL,DELTA X,DELTA Y,PI,IR0W,MP0W,ISAM,IC0DE,N) B 1
COMMON /HAND/ E(8292),D(8292) B 2
COMMON /DRY/ X(64),Y(64) B 3
COMMON /WET/ A(128,2) B 4
***** B 5
C * B 6
C * SOLVES INHOMOGENOUS CAUCHY-RIEMANN EQUATIONS IN A RECTANGLE * B 7
C * U + V = D(X,Y) * B 8
C * X Y WITH B.C. V(X,0) , V(X,YMAX) * B 9
C * U - V = E(X,Y) * B 10
C * Y X AND A FIXED VALUE OF U ASSIGNED * B 11
C * AT SOME (X,Y). * B 12
C * * B 13
C * ALL FUNCIONS ARE PERIODIC IN X WITH PERIOD 2*DELTA X*M . * B 14
C * * B 15
C * THE R.H.S. E(X,Y) , D(X,Y) ARE STORED IN * B 16
C * E(M*J+I) = E( DELTA X*(2*I-1) , DELTA Y*(2*J+2) ) * B 17
C * FOR J = 0...N-2 , I = 1...M * B 18
C * D(M*J+I) = D( DELTA X*(2*(I-1) DELTA Y*(2*J+1) ) * B 19
C * FOR J = 0...N-1 , I = 1...M * B 20
C * * B 21
C * THE BOUNDARY CONDITIONS ARE STORED IN * B 22
C * A(I,1) = V(DELTA X*I,0) , A(I,2) = V(DELTA X*I,DELTA Y*N) * B 23
C * FOR I = 1...M * B 24
C * * B 25
C * SOLUTIONS U AND V ARE RETURNED IN * B 26
C * E(M*J+I) = U( DELTA X*(2*I-1) , DELTA Y*(2*J+1) ) * B 27
C * FOR J = 0...N-1 , I = 1...M * B 28
C * D(M*J+I) = V( DELTA X*(2*(I-1) , DELTA Y*(2*J+2) ) * B 29
C * FOR J = 0...N-2 , I = 1...M * B 30
C * * B 31
C * N = NUMBER OF MESH POINTS IN Y * B 32
C * 2**MP0W = NUMBER OF MESH POINTS IN X * B 33
C * * B 34
C * ISAM = -1 SOLVES V FROM TOP TO BOTTOM(Y AXIS) * B 35
C * ISAM = 1 SOLVES V FROM BOTTOM TO TOP * B 36
C * ISAM = 0 DOESNT SOLVE V * B 37
C * IR0W = THE ROW IN Y WHERE THE FIXED VALUE OF U IS * B 38
C * ASSIGNED * B 39
C * = (J+1) OF U( X , DELTA Y*(2*J+1) ) * B 40
C * SOL = SUM OVER THE (X) OF THIS ROW * B 41
C * * B 42
C * IC0DE MUST BE SET NOT EQUAL TO 3 WHEN ROUTINE IS * B 43
C * CALLED FOR THE FIRST TIME * B 44
C * IC0DE = 3 FOR SUBSEQUENT CALL IF THE NUMBER OF * B 45
C * MESH POINTS IN X AND Y ARE SAME AS IN THE PREVIOUS CALL * B 46
C * * B 47
C ***** B 48
M=2**MP0W B 49
M*Y2=M/2 B 50
M1=M-1 B 51
A1=1.0/FLOAT(M) B 52
N1=N-1 B 53
NEXT=0 B 54

```

```

MHALFP=MBY2+1
MPLUS2=M+2
IL=M*N
ILM=IL-M
ALAM=DELTAY/DELTAX
ALSO=ALAM*ALAM
C
C      SETTING UP THE R.H.S. OF THE LINEAR SYSTEM
C
G2=2.0*DELTAY
DO 10 K=1,ILM
E(K)=E(K)*G2
10 D(K)=D(K)*G2
DO 20 I=1,M
KP=I+ILM
D(KP)=D(KP)*G2-A(I,2)
D(I)=D(I)+A(I,1)
20 A(I,1)=0.0
IP=1
IS=2
DO 40 J=1,ILM,M
KM=J-1
DO 30 I=1,M1
K1=KM+I
A(I,IS)=E(K1)
30 E(K1)=ALAM*(D(K1+1)-D(K1))+E(K1)-A(I,IP)
KMMP=K1+1
A(M,IS)=E(KMMP)
E(KMMP)=ALAM*(D(KM+1)-D(KMMP))+E(KMMP)-A(M,IP)
IEX=IP
IP=IS
40 IS=IEX
DO 50 I=1,M1
K1=ILM+I
50 E(K1)=ALAM*(D(K1+1)-D(K1))-A(I,IP)
E(IL)=ALAM*(D(ILM+1)-D(IL))-A(M,IP)
C
C      PERFORMING THE FAST FOURIER TRANSFORM BLOCK BY BLOCK
C
SIG=-1.0
DO 80 K1=1,N
JF=(K1-1)*M
DO 60 I=1,M
J=JF+I
A(I,1)=E(J)
60 A(I,2)=0.0
CALL FFT2 (SIG,MPDW,PI)
DO 70 I=1,MBY2
IQF=JF+I*2
E(IQF-1)=A(I,1)
70 E(IQF)=A(I,2)
X(K1)=A(MHALFP,1)
80 Y(K1)=A(MHALFP,2)
C
C      SOLVING EACH SUB-SYSTEM
C
DO 120 J=2,MBY2
IF (ICDDE.EQ.3) GO TO 100
X1=2.0*ALSO*(COS(2.0*PI*(FLOAT(J)-1.0)*AM)-1.0)-2.0
A(1,1)=X1+1.0
A(N,1)=X1+1.0
DO 90 K=2,N1
90 A(K,1)=X1
DO 110 K=1,N
I=(K-1)*M+2*J
IQF=K+N
A(K,2)=E(I-1)
110 A(IQF,2)=E(I)
CALL CONSOL (N,NEXT,ICODE)
DO 120 K=1,N
I=(K-1)*M+2*J
E(I-1)=A(K,2)
IQF=K+N
120 E(I)=A(IQF,2)
IF (ICODE.EQ.3) GO TO 140
X1=-2.0*(ALSO*2.0+1.0)
A(1,1)=X1+1.0
A(N,1)=X1+1.0
DO 130 K=2,N1
130 A(K,1)=X1
DO 150 J=1,N
140 A(J,2)=X(J)
IQF=J+N

```

```

150 A(IQF,2)=Y(J)
CALL CONSOL (N,NEXT,ICODE)
DO 160 J=1,N
IQF=N+J
X(J)=A(J,2)
160 Y(J)=A(IQF,2)
IF (ICODE,EQ,3) GO TO 180
A(1,1)=-1.0
A(N,1)=-1.0
DO 170 K=2,N1
170 A(K,1)=-2.0
180 DO 190 K=1,N
I=(K-1)*M+1
A(K,2)=E(I)
IQF=N+K
190 A(IQF,2)=E(I+1)
A(IROW,1)=A(IROW,1)-1.0
A(IROW,2)=A(IROW,2)-SOL
CALL CONSOL (N,NEXT,ICODE)
DO 200 K=1,N
I=(K-1)*M+1
E(I)=A(K,2)
IQF=N+K
200 E(I+1)=A(IQF,2)
C
C FAST FOURIER TRANSFORM OF SOLUTION U
C
SIG=1.0
DO 230 K1=1,N
JF=(K1-1)*M
DO 210 I=1,M*Y2
IQF=JF+I*2
A(I,1)=E(IQF-1)
210 A(I,2)=E(IQF)
DO 220 I=2,M*Y2
L=MPLUS2-I
A(L,1)=A(I,1)
220 A(L,2)=-A(I,2)
A(MHALFP,1)=X(K1)
A(MHALFP,2)=Y(K1)
CALL FFT2 (SIG,MPOK,PI)
DO 230 I=1,M
J=JF+I
230 E(J)=A(I,1)*AM
C
C SOLUTION V WILL BE IN ARRAY D AND U WILL BE IN E
C
IF (ISAM,EQ,0) GO TO 300
IF (ISAM,EG,-1) GO TO 260
DO 250 K=1,N1
KM=K*M
KMMP=KM-M
A(1,1)=ALAM*(E(KM)-E(KMMP+1))
DO 240 J=2,M
KJ=KMMP+J
240 A(J,1)=ALAM*(E(KJ-1)-E(KJ))
DO 250 I=1,M
J=KMMP+I
D(J)=D(J)+A(I,1)
IF (K.LL.1) GO TO 250
K1I=J-M
D(J)=D(J)+D(K1I)
250 CONTINUE
GO TO 300
260 DO 270 I=1,M
J=I+ILM
270 A(I,1)=D(J)
IS=1
IP=2
DO 290 KD=1,N1
IEX=IS
IS=IP
IP=IEX
KM=M*(N-KD+1)
KMMP=KM-M
A(1,IP)=ALAM*(E(KM)-E(KMMP+1))+A(1,IP)
DO 280 J=2,M
KJ=KMMP+J
280 A(J,IP)=ALAM*(E(KJ-1)-E(KJ))+A(J,IP)
DO 290 I=1,M
J=KMMP+I

```

```

B 138
B 139
B 140
B 141
B 142
B 143
B 144
B 145
B 146
B 147
B 148
B 149
B 150
B 151
B 152
B 153
B 154
B 155
B 156
B 157
B 158
B 159
B 160
B 161
B 162
B 163
B 164
B 165
B 166
B 167
B 168
B 169
B 170
B 171
B 172
B 173
B 174
B 175
B 176
B 177
B 178
B 179
B 180
B 181
B 182
B 183
B 184
B 185
B 186
B 187
B 188
B 189
B 190
B 191
B 192
B 193
B 194
B 195
B 196
B 197
B 198
B 199
B 200
B 201
B 202
B 203
B 204
B 205
B 206
B 207
B 208
B 209
B 210
B 211
B 212
B 213
B 214
B 215
B 216
B 217
B 218

```



```

K1I=J-M
A(I,IS)=D(K1I)
D(K1I)=-A(I,IP)
IF (KD.GT.1) D(K1I)=D(J)+D(K1I)
290 CONTINUE
300 CONTINUE
RETURN
END
SUBROUTINE FFT2 (SIG,M,PI)
COMMON /WET/ X(126),Y(126)
C
C FAST FOURIER TRANSFORM
C
N=2**M
NV2=N/2
NM1=N-1
J=1
DO 30 I=1,NM1
IF (I.GE.J) GO TO 10
IX=X(J)
IY=Y(J)
X(J)=X(I)
Y(J)=Y(I)
X(I)=IX
Y(I)=IY
10 K=NV2
20 IF (K.GE.J) GO TO 30
J=J-K
K=K/2
GO TO 20
30 J=J+K
DO 50 L=1,M
LE=2**L
LE1=LE/2
UX=1.0
UY=0.0
ANG=PI/FLOAT(LE1)
WX=COS(ANG)
WY=SIG*SIN(ANG)
DO 50 J=1,LE1
DO 40 I=J,N,LE
IP=I+LE1
IX=X(IP)*UX-Y(IP)*UY
IY=X(IP)*UY+Y(IP)*UX
X(IP)=X(I)-IX
Y(IP)=Y(I)-IY
X(I)=X(I)+IX
40 Y(I)=Y(I)+IY
US=UX*WX-UY*WY
UY=UX*WY+UY*WX
50 UX=US
RETURN
END
SUBROUTINE CONSOL (N,NEXT,ICODE)
COMMON /KEEP/ Z(8256)
COMMON /WET/ X(126),Y(126)
C
C *****
C * SOLVES A SPECIAL TRI-DIAGONAL LINEAR SYSTEM BY LU *
C * FACTORIZATION . *
C * TWO R.H.S. ARE GIVEN AND STORED IN Y, STARTING AT Y(1) *
C * AND AT Y(N+1) RESPECTIVELY . *
C * THE TWO CORRESPONDING VECTOR UNKNOWN ARE RETURNED IN X, *
C * STARTING AT X(1) AND AT X(N+1) RESPECTIVELY. *
C * *
C * IF ICODE = 3 THEN Z ARRAY MUST BE KEPT RESERVED FOR *
C * USE BY CONSOL. *
C * *****
C
IF (ICODE.EQ.3) GO TO 20
Z(NEXT+1)=X(1)
DO 10 K=2,N
NEXTPL=NEXT+K
K1=K-1
X(K1)=1.0/Z(NEXTPL-1)
10 Z(NEXTPL)=X(K)-X(K1)
20 DO 30 K=2,N
KPL=K+N
KNEXT=NEXT+K-1
Y(K)=Y(K)-Y(K-1)/Z(KNEXT)

```

```

B 214
B 220
B 221
B 222
B 223
B 224
B 225
B 226-
C 1
C 2
C 3
C 4
C 5
C 6
C 7
C 8
C 9
C 10
C 11
C 12
C 13
C 14
C 15
C 16
C 17
C 18
C 19
C 20
C 21
C 22
C 23
C 24
C 25
C 26
C 27
C 28
C 29
C 30
C 31
C 32
C 33
C 34
C 35
C 36
C 37
C 38
C 39
C 40
C 41
C 42
C 43
C 44
C 45-
D 1
D 2
D 3
D 4
D 5
D 6
D 7
D 8
D 9
D 10
D 11
D 12
D 13
D 14
D 15
D 16
D 17
D 18
D 19
D 20
D 21
D 22
D 23
D 24
D 25
D 26
D 27

```

```

30 Y(KPL)=Y(KPL)-Y(KPL-1)/Z(KNEXT)           D 28
NEXTPL=NEXT+N                                 D 29
Y(N)=Y(N)/Z(NEXTPL)                           D 30
Y(Z*N)=Y(Z*N)/Z(NEXTPL)                       D 31
DO 40 J=2,N                                    D 32
K=N-J+1                                        D 33
KPL=K+N                                        D 34
KNEXT=K+NEXT                                   D 35
Y(K)=(Y(K)-Y(K+1))/Z(KNEXT)                   D 36
Y(KPL)=(Y(KPL)-Y(KPL+1))/Z(KNEXT)             D 37
NEXT=NEXTPL                                    D 38
RETURN                                          D 39
END                                              D 40-

```

Courant Institute of Mathematical Sciences
 New York University
 New York, New York 10012

1. E. BATSCHELET, "Über die numerische Auflösung von Randwertproblemen bei elliptischen partiellen Differentialgleichungen," *Z. Angew. Math. Phys.*, v. 3, 1952, pp. 165–193.
2. J. H. BRAMBLE & B. E. HUBBARD, "Approximation of derivatives by difference methods in elliptic boundary value problems," *Contributions to Differential Equations*, v. 3, 1964, pp. 399–410.
3. J. H. BRAMBLE & B. E. HUBBARD, "A finite difference analog of the Neumann problem for Poisson's equation," *SIAM J. Numer. Anal.*, v. 2, 1964, pp. 1–14.
4. O. BUNEMAN, *A Compact Non-Iterative Poisson Solver*, SUIPR Report No. 294, Inst. Plasma Research, Stanford Univ., May 1969, 11 pp.
5. B. L. BUZBEE, G. H. GOLUB & C. W. NIELSON, "On direct methods for solving Poisson's equations," *SIAM J. Numer. Anal.*, v. 8, 1970, pp. 627–656.
6. L. COLLATZ, *The Numerical Treatment of Differential Equations*, 3rd ed., *Mathematische Wissenschaften*, vol. 60, Springer-Verlag, Berlin, 1966, 568 pp.
7. J. W. COOLEY, P. A. W. LEWIS & P. D. WELCH, "The finite Fourier transform," *IEEE Trans. Audio and Electroacoustics*, v. 17, 1969, pp. 77–85.
8. E. G. D'JAKONOV, "On certain iterative methods for solving nonlinear difference equations," in *Conference on the Numerical Solution of Differential Equations* (J. Li. Morris, Ed.), *Lecture Notes in Math.*, vol. 109, Springer, Berlin, 1969, pp. 7–22.
9. F. W. DORR, "The direct solution of the discrete Poisson equation on a rectangle," *SIAM Rev.*, v. 12, 1970, pp. 248–263.
10. T. ELVIUS & A. SUNDSTRÖM, "Computationally efficient schemes and boundary conditions for a fine-mesh barotropic model based on the shallow-water equations," *Tellus*, v. 25, 1973, pp. 132–156.
11. G. E. FORSYTHE & W. R. WASOW, *Finite-Difference Methods for Partial Differential Equations*, Wiley, New York, 1960, 444 pp.
12. D. FISCHER, G. GOLUB, O. HALD, C. LEIVA & O. WIDLUND, "On Fourier-Toeplitz methods for separable elliptic problems," *Math. Comp.*, v. 28, 1974, pp. 349–368.
13. S. GERSCHGORIN, "Fehlerabschätzung für das Differenzverfahren zur Lösung partieller Differentialgleichungen," *Z. Angew. Math. Mech.*, v. 10, 1930, pp. 373–382.
14. M. GHIL, "The initialization problem in numerical weather prediction," in *Improperly Posed Boundary Value Problems* (A. Carasso and A. P. Stone, Eds.), *Research Notes in Math.*, vol. 1, Pitman, London, 1975, pp. 105–123.
15. M. GHIL, *Initialization by Compatible Balancing*, Report 75–16, Inst. Comp. Appl. Sci. Engr., Hampton, Virginia, 1975, 38 pp.
16. M. GHIL & B. SHKOLLER, "Wind laws for shockless initialization," *Ann. Meteor.* (Neue Folge), v. 11, 1976, pp. 112–115.
17. M. GHIL, B. SHKOLLER & V. YANGARBER, "A balanced diagnostic system compatible with a barotropic prognostic model," *Mon. Wea. Rev.*, v. 105, 1977, pp. 1223–1238.
18. G. GOLUB, "Direct methods for solving elliptic difference equations," in *Symposium on the Theory of Numerical Analysis* (J. Li. Morris, Ed.), *Lecture Notes in Math.*, vol. 193, Springer-Verlag, Berlin, 1971, pp. 1–19.
19. J. E. GUNN, "The solution of elliptic difference equations by semi-explicit iterative techniques," *SIAM J. Numer. Anal.*, v. 2, 1965, pp. 24–45.

20. B. GUSTAFSSON, "An alternating direction implicit method for solving the shallow water equations," *J. Computational Phys.*, v. 7, 1971, pp. 239–254.
21. R. W. HOCKNEY, "A fast direct solution of Poisson's equation using Fourier analysis," *J. Assoc. Comput. Mach.*, v. 12, 1965, pp. 95–113.
22. R. W. HOCKNEY, "The potential calculation and some applications," in *Methods in Computational Physics* (B. Adler, S. Fernbach and M. Rotenberg, Eds.), vol. 9 (Plasma Physics), Academic Press, New York, 1969, pp. 135–211.
23. W. E. LANGLOIS, *Vorticity-Stream Function Computation of Incompressible Fluid Flow with an Almost-Flat Free Surface*, IBM Research Report RJ 1794 (#26092), 1976, 8 pp.
24. H. LOMAX & E. D. MARTIN, "Fast direct numerical solution of the nonhomogeneous Cauchy-Riemann equations," *J. Computational Phys.*, v. 15, 1974, pp. 55–80.
25. E. D. MARTIN & H. LOMAX, *Rapid Finite-Difference Computation of Subsonic and Transonic Aerodynamic Flows*, AIAA Paper No. 74–11, 1974, 13 pp.
26. E. D. MARTIN & H. LOMAX, *Variants and Extensions of a Fast Direct Numerical Cauchy-Riemann Solver, with Illustrative Applications*, NASA Tech. Note TN D-7934, 1977, 94 pp.
27. J. OLIGER & A. SUNDSTRÖM, "Theoretical and practical aspects of some initial-boundary value problems in fluid dynamics," *SIAM J. Appl. Math. A*, v. 35, 1978, pp. 419–446.
28. P. J. ROACHE, *Computational Fluid Dynamics*, 2nd ed., Hermosa Publishers, Albuquerque, 1976, 446 pp.
29. U. SCHUMANN & R. A. SWEET, "A direct method for the solution of Poisson's equation with Neumann boundary conditions on a staggered grid of arbitrary size," *J. Computational Phys.*, v. 20, 1976, pp. 171–182.
30. P. N. SWARZTRAUBER, "A direct method for the discrete solution of separable elliptic equations," *SIAM J. Numer. Anal.*, v. 11, 1974, pp. 1136–1150.
31. R. A. SWEET, "A generalized cyclic reduction algorithm," *SIAM J. Numer. Anal.*, v. 11, 1974, pp. 506–520.
32. O. WIDLUND, "On the use of fast methods for separable finite-difference equations for the solution of general elliptic problems," in *Sparse Matrices and Their Applications* (D. J. Rose and R. A. Willoughby, Eds.), Plenum Press, New York, 1972, pp. 121–131.
33. O. WIDLUND, "Capacitance matrix methods for Helmholtz' equation on general bounded regions," in *Numerical Treatment of Differential Equations* (R. Bulirsch, R. D. Grigorieff and J. Schröder, Eds.), Lecture Notes in Math., vol. 631, Springer, Berlin, 1978, pp. 209–219.