JOURNAL OF COMPUTATIONAL PHYSICS **15**, 55–80 (1974)

Fast Direct Numerical Solution of the Nonhomogeneous Cauchy–Riemann Equations

HARVARD LOMAX AND E. DALE MARTIN

Computational Fluid Dynamics Branch, Ames Research Center, NASA, Moffett Field, California 94035

Received August 6, 1973

A fast direct (noniterative) "Cauchy–Riemann Solver" is developed for solving the finite-difference equations representing systems of first-order elliptic partial differential equations in the form of the nonhomogeneous Cauchy–Riemann equations. The method is second-order accurate and requires approximately the same computer time as a fast cyclic-reduction Poisson solver (Buneman's method, but with the cyclic reduction of simple tridiagonal matrices replaced by the Thomas algorithm).

The accuracy and efficiency of the direct solver are demonstrated in an application to solving an example problem in aerodynamics: subsonic inviscid flow over a biconvex airfoil. The analytical small-perturbation solution contains singularities, which are captured well by the computational technique.

The algorithm is expected to be useful in nonlinear subsonic and transonic aerodynamics.

INTRODUCTION

The previously available fast, direct computational algorithms for solving finite-difference equations representing partial differential equations containing elliptic operators have been limited to second-order equations. Those algorithms and the corresponding computer programs are commonly referred to as "fast Poisson solvers" (e.g. see [1–6]). They have been used with significant success in computational physics. In particular they have been useful in computational fluid dynamics (e.g. see discussion and references in Ref. [7], especially pp. 113 ff, 180 ff, and 195), usually in the solution of Poisson's equation for the stream function, or for the pressure, within computational techniques for solving the Navier–Stokes equations. More extensive use of the elliptic solvers in other flow problems has been discussed briefly in Ref. [8], which presented a technique for using the direct solvers in problems with arbitrary interior boundaries.

In some problems in computational fluid dynamics there appear to be advantages in working with the corresponding *first-order* elliptic system of equations [9, 10] in terms of the "primitive variables." In the simplest case, these are the *Cauchy–Riemann* equations for the velocity components, U and V (corresponding to Laplace's equation for either the velocity potential or the stream function). In more general cases, the corresponding equations can be used in the form of the nonhomogeneous Cauchy–Riemann equations. This formulation may be preferred over either the velocity-potential or the stream-function formulation when the right sides are not both identically zero in the problem. This will be of special interest in planned future applications. Therefore it was considered desirable to develop a fast direct algorithm for numerically solving the finite-difference equations representing those first-order elliptic equations. This paper presents such a development, including (a) discussion of appropriate indexing and mesh configurations with resulting orders of accuracy, (b) a procedure for decoupling the algebraic matrix equations for V values from those for U values, (c) the reduction (direct solution) of the matrix equations, (d) the final determination of V and U, and (e) an example problem in subsonic aerodynamics that demonstrates the accuracy and efficiency of the new direct Cauchy–Riemann solver.

The Numerical Problem

The linear set of elliptic, first-order, two-dimensional partial differential equations to be solved numerically can be written in the form

$$\partial U/\partial x + \partial V/\partial y = s(x, y), \tag{1a}$$

$$\partial U/\partial y - \partial V/\partial x = -\omega(x, y). \tag{1b}$$

In applications to inviscid incompressible fluid flow U and V are components of total velocity in the x and y directions, respectively, and s and ω are functions representing source and vorticity distributions, respectively, which may include point sources and point vortices. If s and ω are zero, Eqs. (1) are the Cauchy–Riemann equations.

We wish to replace Eqs. (1) with a set of finite-difference equations that are to some order of approximation their equivalent, and then consider the direct (i.e., noniterative) solution of the resulting coupled algebraic expressions.

The Difference Equations, the Boundary Conditions, and the Mesh

Consider the simplest case of differencing formulas where the derivatives in Eq. (1a) are approximated by backward differences at each j, k and those in Eq. (1b) are approximated by forward differences. We then have the set of difference equations.

$$(U_{j,k} - U_{j-1,k})/\Delta x + (V_{j,k} - V_{j,k-1})/\Delta y = s_{j,k}\,, \tag{2a}$$

$$(U_{j,k+1} - U_{j,k})/\Delta y - (V_{j+1,k} - V_{j,k})/\Delta x = -\omega_{j,k}\,, \tag{2b}$$

where j and k index each point (x, y) at which Eqs. (1a) and (1b) are both approximated. The situation is illustrated schematically in Fig. 1. The computation boundary is indicated by dashed lines. The solid symbols represent positions where the data are given by the boundary conditions, with squares indicating U values and circles indicating V values. The data at the remaining points (open symbols) must be determined from the set of algebraic relations given by Eqs. (2). The dot ($\cdot$) represents a position at which the continuity equation, (1a), is balanced, and the cross ($+$) represents a position at which the vorticity equation, (1b), is balanced. This notation will have more meaning in a later reordering of the mesh. In this example these positions are coincident (parts (a) and (b) of Fig. 1 show

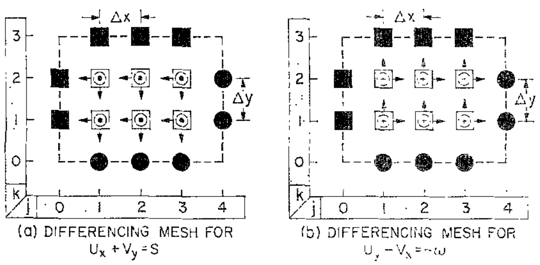
Δx
Δy
k
j
0
1
2
3
4
(a) DIFFERENCING MESH FOR $U_x + V_y = S$
(b) DIFFERENCING MESH FOR $U_y - V_x = -\omega$

FIG. 1. Simplest computation mesh and indexing for (first-order) one-sided finite differences ($\square \sim U$ data; $\circ \sim V$ data; $\cdot \sim U_x + V_y = s$; $+ \sim U_y - V_x = -\omega$; solid symbols $\sim$ prescribed boundary values; dashed lines indicate computation boundary).

identical locations), but the differencing schemes used to approximate the balances are opposite (indicated by arrows on Fig. 1). Note that there are twelve equations (six dots and six crosses) and twelve unknowns (six values of U and six values of V). Equations (2) are a first-order approximation to Eqs. (1); that is, the truncation error in those equations is proportional to the first power of the spacing $(\Delta x, \Delta y)$.

Another approximation of Eqs. (1) can be constructed by staggering in half steps the positions at which the U and V data are carried in the manner illustrated in Fig. 2(a). For the moment, disregard the "U indices" and "V indices" and look at the j and k indices located only in the *shaded areas* (on the lower and left sides) on Fig. 2(a). The symbols on Fig. 2(a) have the same meaning as before but now the positions at which the continuity and vorticity equations are balanced are no longer coincident, and the positions at which the U, V data are stored are different from the balancing positions as well as from one another. The set of difference equations can be written

$$(U_{j+1/2,k} - U_{j-1/2,k})/\Delta x + (V_{j,k+1/2} - V_{j,k-1/2})/\Delta y = s_{j,k}\,, \tag{3a}$$

$$(U_{j+1/2,k+1} - U_{j+1/2,k})/\Delta y - (V_{j+1,k+1/2} - V_{j,k+1/2})/\Delta x = -\omega_{j+1/2,k+1/2}\,, \tag{3b}$$

where j and k index a point $(\cdot)$ at which Eq. (1a) is approximated and $j + 1/2$, $k + 1/2$ is the corresponding point $(+)$ in Fig. 2(a) at which Eq. (1b) is approximated. Notice that each partial derivative has now been replaced by a central difference formula, and Eqs. (3a) and (3b) are, therefore, a second-order-accurate approximation of Eqs. (1). If s and ω are zero, the only difference in the actual

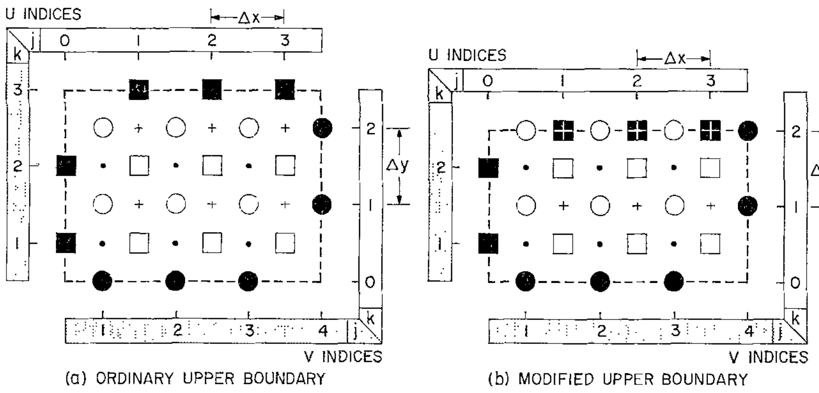

FIG. 2. Staggered computation meshes and indexing for (second-order) central differences with $NX = 3$, $NY = 2$ ($\square \sim U$ data; $\circ \sim V$ data; $\cdot \sim U_x + V_y = s$; $+ \sim U_y - V_x = -\omega$; solid symbols $\sim$ prescribed boundary values; dashed lines indicate computation boundary).

programming of the algorithms represented by Eqs. (2) and Eqs. (3) is in the way the boundary conditions are aligned with respect to the data. Fig. 2(a) shows one way in which the staggered data can be carried in the mesh.

Other alignments also exist. For example, Fig. 2(b) illustrates an especially useful case. Again, disregard "U indices" and "V indices" and look at j and k in the shaded areas only. In this case the derivatives in Eqs. (1) are approximated by the central difference formula everywhere except for the vorticity equation along the upper boundary. For this one line, Eq. (3b) is replaced with

$$(U_{j+1/2,k+1/2} - U_{j+1/2,k})/(\tfrac{1}{2}\Delta y) - (V_{j+1,k+1/2} - V_{j,k+1/2})/\Delta x = -\omega_{j+1/2,k+1/2}\,. \tag{3c}$$

Although Eq. (3c) is only accurate to the first order, its use is quite appropriate for certain applications and will not degrade the overall second-order accuracy. In fact, for reasons given later, the approximation represented by Eqs. (3a), (3b), and (3c) is the one we will consider in detail.

Convenient Indexing and Resulting Matrix Equation

In the section to follow this one we consider in some detail the direct, noniterative solution of Eqs. (3); but first it is convenient to abandon the half-step notation that appears in these equations in favor of the notation illustrated in Fig. 2(a). Use now the j and k values denoted as "U indices" for indexing U and the j and k values denoted by "V indices" for indexing V. One can think of a U mesh and a V mesh as being distinct and displaced by $(\frac{1}{2})\,\Delta x$ and $(\frac{1}{2})\,\Delta y$ from each other. Further, let the j index of s correspond to that of V and the k index of s correspond to that of U; and let the j index of ω correspond to that of U and the k index of ω correspond to that of V. With this convention the second-order-accurate equations in terms of the staggered U indices and V indices for Fig. 2(a) are

$$(U_{j,k} - U_{j-1,k})/\Delta x + (V_{j,k} - V_{j,k-1})/\Delta y = s_{j,k}\,, \tag{4a}$$

$$(U_{j,k+1} - U_{j,k})/\Delta y - (V_{j+1,k} - V_{j,k})/\Delta x = -\omega_{j,k}\,. \tag{4b}$$

Notice that Eqs. (2) and (4) have the same form, although they represent quite different approximations to the basic partial differential equations (1).

Recall that Fig. 2(b) is the same as Fig. 2(a) except that the vorticity equation written for the top row of crosses takes the form of Eq. (3c). In the notation of "U indices" and "V indices," the latter equation is the same as (4b) but with Δy in the denominator of the first term replaced by $(\frac{1}{2})\,\Delta y$, that is, for the top row of crosses in Fig. 2(b), Eq. (4b) is replaced by the following equivalent of (3c):

$$(U_{j,k+1} - U_{j,k})/(\tfrac{1}{2}\Delta y) - (V_{j+1,k} - V_{j,k})/\Delta x = -\omega_{j,k}\,. \tag{4c}$$

For this top row only, the k index of U corresponds to the same y location as the k index of V. In a mesh construction corresponding to Fig. 2(a) or 2(b) let NX be the number of dots (or crosses) along a horizontal row and let NY be the number of dots (or crosses) along a column. Then all of Eqs. (4) apply for $j = 1, 2, \ldots, NX$;

Eqs. (4a) and (4b) apply for $k = 1, 2, \dots, NY - 1$; and Eqs. (4a) and (4c) apply for $k = NY$.

For the case represented by Fig. 2(b), the matrix formulation of Eqs. (4a), (4b), and (4c) is

$$
\left[\begin{array}{ccc|ccc|ccc|ccc}
\mu & 0 & 0 & & & & 1 & & & & & \\
-\mu & \mu & 0 & & 0 & & & 1 & & & 0 & \\
0 & -\mu & \mu & & & & & & 1 & & & \\
\hline
 & & & \mu & 0 & 0 & -1 & & & 1 & & \\
 & 0 & & -\mu & \mu & 0 & & -1 & & & 1 & \\
 & & & 0 & -\mu & \mu & & & -1 & & & 1 \\
\hline
-1 & & & 1 & & & \mu & -\mu & 0 & & & \\
 & -1 & & & 1 & & 0 & \mu & -\mu & & 0 & \\
 & & -1 & & & 1 & 0 & 0 & \mu & & & \\
\hline
 & & & -1 & & & & & & \frac{1}{2}\mu & -\frac{1}{2}\mu & 0 \\
 & 0 & & & -1 & & & 0 & & 0 & \frac{1}{2}\mu & -\frac{1}{2}\mu \\
 & & & & & -1 & & & & 0 & 0 & \frac{1}{2}\mu
\end{array}\right]
\left[\begin{array}{c}
U_{11} \\ U_{21} \\ U_{31} \\ \hline U_{12} \\ U_{22} \\ U_{32} \\ \hline V_{11} \\ V_{21} \\ V_{31} \\ \hline V_{12} \\ V_{22} \\ V_{32}
\end{array}\right]
=
\left[\begin{array}{c}
f_{11} \\ f_{21} \\ f_{31} \\ \hline f_{12} \\ f_{22} \\ f_{32} \\ \hline g_{11} \\ g_{21} \\ g_{31} \\ \hline g_{12} \\ g_{22} \\ g_{32}
\end{array}\right]
\tag{5}
$$

where each equation has been multiplied through by Δy, and where $\mu = \Delta y/\Delta x$, and f and g include $s\,\Delta y$ and $-\omega\,\Delta y$, respectively, together with the terms carrying the boundary conditions and represented by the solid symbols shown in Fig. 2(b). Let $\mathbf{T}(a, b, c)$ be a tridiagonal matrix, and set

$$
\left.\begin{aligned}
\mathbf{I} &= \mathbf{T}(0, 1, 0), \\
\mathbf{A} &= \mu\mathbf{T}(-1, 1, 0), \\
\mathbf{B} &= \mu\mathbf{T}(0, 1, -1) = \mathbf{A}^T, \\
\mathbf{U}_k &= \operatorname{col}(U_{1,k}, U_{2,k}, \dots, U_{NX,k}), \\
\mathbf{V}_k &= \operatorname{col}(V_{1,k}, V_{2,k}, \dots, V_{NX,k}), \\
\mathbf{f}_k &= \operatorname{col}(f_{1,k}, f_{2,k}, \dots, f_{NX,k}), \\
\mathbf{g}_k &= \operatorname{col}(g_{1,k}, g_{2,k}, \dots, g_{NX,k}),
\end{aligned}\right\},
\tag{6}
$$

where k ranges from 1 to NY, and where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{I}$ are square matrices each of which is of order NX. Then Eq. (5) can be extended to form the following matrix of block matrices

$$\left[\begin{array}{cccc|cccc}
\mathbf{A} & & & & \mathbf{I} & \mathbf{O} & & \\
 & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} & \ddots & \\
 & & \ddots & & & \ddots & \ddots & \mathbf{O} \\
 & & & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} \\
\hline
-\mathbf{I} & \mathbf{I} & & & \mathbf{B} & & & \\
\mathbf{O} & \ddots & \ddots & & & \ddots & & \\
 & \ddots & -\mathbf{I} & \mathbf{I} & & & \mathbf{B} & \\
 & & \mathbf{O} & -\mathbf{I} & & & & \frac{1}{2}\mathbf{B}
\end{array}\right]
\left[\begin{array}{c}
\mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{NY} \\ \hline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_{NY}
\end{array}\right]
=
\left[\begin{array}{c}
\mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{NY} \\ \hline \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{NY}
\end{array}\right],
\tag{7}$$

where each of the four quarter blocks of the large matrix in Eq. (7) is of block dimension NY.

The Solution Algorithm

The Cauchy–Riemann solution algorithm to be described here for solving Eqs. (1), in the discrete form of (7), has a direct analogy to a procedure whereby Eqs. (1) would be differentiated first (if differentiable) and then combined to obtain a Poisson equation for V. That analogous procedure, however, is not equivalent to the present algorithm; the two approaches have similarities and differences, which are described below. As we proceed, it will be seen that the development of the present algorithm does contain a matrix equation that is equivalent to a discrete Poisson difference equation. As a result, an available direct Poisson solver (e.g. a variation of Buneman's [1]) can be used within the present solution procedure.

Decoupling **U** *and* **V**

The following simple operation can be used to annihilate all entries in the upper left-hand quarter of the square block matrix in Eq. (7). Consider the upper half of this equation:

(1) Subtract the second row from the first row and add **A** times the first row in the lower half.

(2) Subtract the third row from the second row and add **A** times the second row in the lower half.

(3) Repeat for all but the last row of the upper half, to which is added **A** times the last row from the lower half.

Next consider the lower half of Eq. (7):

(1) Add the bottom row to the one next up.

(2) *Then* add the second from the bottom to the one next up.

(3) Repeat until all the $-\mathbf{I}$ appear only in the diagonal blocks of the lower left quarter block.

Eq. (7) now has been reduced to the convenient form

$$\left[\begin{array}{ccccc|ccccc} \mathbf{O} &&&&& 2\mathbf{I}+\mathbf{AB} & -\mathbf{I} &&& \\ & \mathbf{O} &&&& -\mathbf{I} & 2\mathbf{I}+\mathbf{AB} & -\mathbf{I} && \\ && \ddots &&&& \ddots & \ddots & \ddots & \\ &&& \mathbf{O} &&&& -\mathbf{I} & 2\mathbf{I}+\mathbf{AB} & -2\mathbf{I} \\ &&&& \mathbf{O} &&&& -\mathbf{I} & 2\mathbf{I}+\mathbf{AB} \\ \hline -\mathbf{I} &&&&& \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} & \mathbf{B} \\ & -\mathbf{I} &&&&& \mathbf{B} & \cdots & \mathbf{B} & \mathbf{B} \\ && \ddots &&&&& \ddots & \vdots & \vdots \\ &&& -\mathbf{I} &&&&& \mathbf{B} & \mathbf{B} \\ &&&& -\mathbf{I} &&&&& \mathbf{B} \end{array}\right] \left[\begin{array}{c} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{NY-1} \\ \mathbf{U}_{NY} \\ \hline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_{NY-1} \\ \tfrac{1}{2}\mathbf{V}_{NY} \end{array}\right] = \left[\begin{array}{c} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_{NY-1} \\ \mathbf{F}_{NY} \\ \hline \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_{NY-1} \\ \mathbf{G}_{NY} \end{array}\right], \tag{8}$$

where

$$\begin{aligned} \mathbf{F}_1 &= \mathbf{f}_1 - \mathbf{f}_2 + \mathbf{A}\mathbf{g}_1, & \mathbf{G}_1 &= \mathbf{g}_1 + \mathbf{g}_2 + \cdots + \mathbf{g}_{NY-1} + \mathbf{g}_{NY}, \\ \mathbf{F}_2 &= \mathbf{f}_2 - \mathbf{f}_3 + \mathbf{A}\mathbf{g}_2, & \mathbf{G}_2 &= \mathbf{g}_2 + \mathbf{g}_3 + \cdots + \mathbf{g}_{NY}, \\ &\vdots & &\vdots \\ \mathbf{F}_{NY-1} &= \mathbf{f}_{NY-1} - \mathbf{f}_{NY} + \mathbf{A}\mathbf{g}_{NY-1}, & \mathbf{G}_{NY-1} &= \mathbf{g}_{NY-1} + \mathbf{g}_{NY}, \\ \mathbf{F}_{NY} &= \mathbf{f}_{NY} + \mathbf{A}\mathbf{g}_{NY}, & \mathbf{G}_{NY} &= \mathbf{g}_{NY}. \end{aligned} \tag{9}$$

Note that a factor of $\frac{1}{2}$ in the last column of blocks in the large matrix in Eq. (8) has been transferred to the last vector, $\mathbf{V}_{NY}$, in the column of $\mathbf{U}_k$, $\mathbf{V}_k$ vectors.

If we define $\mathbf{A}_1$ to be the block matrix consisting of the upper right quarter of the square block matrix in Eq. (8); $\mathbf{A}_2$ to be the lower right quarter; and

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{NY} \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \frac{1}{2}\mathbf{V}_{NY} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_{NY} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_{NY} \end{bmatrix}, \tag{10}$$

Eq. (8) can be written

$$\mathbf{A}_1\mathbf{V} = \mathbf{F}, \tag{11a}$$

$$-\mathbf{U} + \mathbf{A}_2\mathbf{V} = \mathbf{G}. \tag{11b}$$

Significant results accomplished at this point are that **V** has been decoupled from **U**, so **V** is determined solely by (11a) and then **U** is determined by (11b); and that all the diagonal blocks of $\mathbf{A}_1$ are identical and one off-diagonal block has a factor of 2. The significance of this will be apparent in the subsection "Cyclic Reduction of Particular Block Matrices."

The operations required to generate **F** and **G** and to compute **U** given **V** are simple to program and require relatively small amounts of computer time. (A table of computing-time measurements is given in a later section.) The major task is to solve Eq. (11a) for **V**, where $\mathbf{A}_1$ is a very large but sparse matrix. This can be carried out by cyclic reduction as explained in the following sections.

The Efficient Reduction of Block Tridiagonal Matrices

Before considering the direct solution of Eq. (11a), let us investigate some basic concepts in the efficient solution of block matrix equations. Let us examine solution procedures for the following matrix equation, which is identical to (11a) except for the last column of blocks in the large block matrix $\mathbf{A}_1$. For illustration we consider the special case where the block dimension is $NY = 5$ and the dimension of each of the vectors $\mathbf{V}_k$ and $\mathbf{F}_k$ is an arbitrary integer NX:

$$\begin{bmatrix} \mathbf{C} & -\mathbf{I} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ -\mathbf{I} & \mathbf{C} & -\mathbf{I} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & -\mathbf{I} & \mathbf{C} & -\mathbf{I} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & -\mathbf{I} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \mathbf{V}_4 \\ \mathbf{V}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \mathbf{F}_3 \\ \mathbf{F}_4 \\ \mathbf{F}_5 \end{bmatrix}, \tag{12}$$

where $\mathbf{C}$ is $2\mathbf{I} + \mathbf{AB}$, or the square tridiagonal matrix of scalar elements (of order NX) given by

$$\mathbf{C} = (\Delta y/\Delta x)^2 \begin{bmatrix} c-1 & -1 & & & \\ -1 & c & -1 & & \\ & -1 & c & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & c \end{bmatrix}, \qquad (13a)$$

in which

$$c = 2[1 + (\Delta x/\Delta y)^2]. \qquad (13b)$$

(a) *Gaussian elimination.* Consider first a form of Gaussian elimination for reducing Eq. (12). In a forward sweep one can reduce (12) to the form

$$\begin{bmatrix} \mathbf{B}_1 & -\mathbf{B}_0 & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{B}_2 & -\mathbf{B}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{B}_3 & -\mathbf{B}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_4 & -\mathbf{B}_3 \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_5 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \mathbf{V}_4 \\ \mathbf{V}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{B}_0\mathbf{F}_1 \\ \mathbf{B}_1\mathbf{F}_2 + \hat{\mathbf{F}}_1 \\ \mathbf{B}_2\mathbf{F}_3 + \hat{\mathbf{F}}_2 \\ \mathbf{B}_3\mathbf{F}_4 + \hat{\mathbf{F}}_3 \\ \mathbf{B}_4\mathbf{F}_5 + \hat{\mathbf{F}}_4 \end{bmatrix} \equiv \begin{bmatrix} \hat{\mathbf{F}}_1 \\ \hat{\mathbf{F}}_2 \\ \hat{\mathbf{F}}_3 \\ \hat{\mathbf{F}}_4 \\ \hat{\mathbf{F}}_5 \end{bmatrix}, \qquad (14a)$$

where

$$\mathbf{B}_0 = \mathbf{I},\ \mathbf{B}_1 = \mathbf{C}, \qquad \text{and} \qquad \mathbf{B}_{N+2} = \mathbf{C}\mathbf{B}_{N+1} - \mathbf{B}_N, \qquad (14b)$$

from which the solution follows (to be evaluated in reverse order):

$$\left.\begin{aligned} \mathbf{V}_1 &= \mathbf{B}_1^{-1}[\hat{\mathbf{F}}_1 + \mathbf{B}_0\mathbf{V}_2] \\ \mathbf{V}_2 &= \mathbf{B}_2^{-1}[\hat{\mathbf{F}}_2 + \mathbf{B}_1\mathbf{V}_3] \\ \mathbf{V}_3 &= \mathbf{B}_3^{-1}[\hat{\mathbf{F}}_3 + \mathbf{B}_2\mathbf{V}_4] \\ \mathbf{V}_4 &= \mathbf{B}_4^{-1}[\hat{\mathbf{F}}_4 + \mathbf{B}_3\mathbf{V}_5] \\ \mathbf{V}_5 &= \mathbf{B}_5^{-1}[\hat{\mathbf{F}}_5] \end{aligned}\right\}. \qquad (14c)$$

Now *if the B_N were scalars*, the construction of each $\hat{\mathbf{F}}_k$ would require four multiplications and four additions, and the final recursion for the $\mathbf{V}_k$ would require three multiplications, four additions, and five divisions; a total of eight additions, seven multiplications, and five divisions. However, *when the $\mathbf{B}_N$ are matrices*, the situation is entirely different, and this fact motivates the following discussion.

The key to the direct reduction techniques is that there exists a *matrix polynomial*, which can be *factored* (cf. Ref. [5]). For example, in Eq. (14), each $\mathbf{B}_N$ is a polynomial in $\mathbf{C}$ of degree N (where N is also the number of roots of the polynomial; see Eqs. (16) below). This follows from the recursion relations, Eq. (14b). Thus, one can show by solving the matrix difference equation for $\mathbf{B}_N$ (along with the two conditions) in Eq. (14b) that

$$\mathbf{B}_N = (\mathbf{C}^2 - 4\mathbf{I})^{-1/2}\{[\mathbf{C} + (\mathbf{C}^2 - 4\mathbf{I})^{1/2}]^{N+1} - [\mathbf{C} - (\mathbf{C}^2 - 4\mathbf{I})^{1/2}]^{N+1}\}/2^{N+1} \quad (15)$$

from which it follows (i.e., the factorization is given by)

$$\mathbf{B}_N = \prod_{n=1}^{N} (\mathbf{C} - \lambda_n \mathbf{I}), \quad (16a)$$

where (see Dorr[4])

$$\lambda_n = 2\cos[n\pi/(N+1)], \quad n = 1, 2, \ldots, N. \quad (16b)$$

It is important to recognize that the factors in a product of the form

$$(\mathbf{C} - \lambda_i \mathbf{I})(\mathbf{C} - \lambda_j \mathbf{I})$$

commute, λ_i and λ_j being any two scalars. Let

$$\mathbf{B}_{\lambda_n} = \mathbf{C} - \lambda_n \mathbf{I} \quad (17a)$$

and notice that $\mathbf{B}_{\lambda_n}$ has the same construction as $\mathbf{C}$, defined in Eq. (13a), except that for $\mathbf{B}_{\lambda_n}$,

$$c = 2 + (\Delta x/\Delta y)^2(2 + \lambda_n). \quad (17b)$$

Thus, by the definition in Eq. (16a), the term $\mathbf{B}_4\mathbf{F}_5$ in Eq. (14a) represents an operation that can be expressed as four consecutive matrix multiplications on a vector of dimension NX that is originally filled with $\mathbf{F}_5$. In our notation

$$\mathbf{B}_4\mathbf{F}_5 = \mathbf{B}_{\lambda_1}\mathbf{B}_{\lambda_2}\mathbf{B}_{\lambda_3}\mathbf{B}_{\lambda_4}\mathbf{F}_5\,. \quad (18)$$

All four of the matrices $\mathbf{B}_{\lambda_n}$ are tridiagonal and their sequence is immaterial because they all commute with one another.

If we let DYX2, ALAM, and NX be FORTRAN variable names representing $(\Delta y/\Delta x)^2$, λ_n, and the rank of $\mathbf{C}$, respectively, the sequence

```
   DO 1 N = 1,4
   ALAM = 2.*COS(N*PI/(N + 1))
 1 CALL SUBROUTINE TRIM(DYX2, ALAM, V, NX)
```

would carry out the actual arithmetic necessary to calculate the product $\mathbf{B}_4\mathbf{F}_5$ in Eq. (18). The subroutine TRIM is constructed to perform a single multiplication of the contents of the first NX elements of the array V by the tridiagonal $\mathbf{B}_{\lambda_n}$ and return the results in the same array V.

Just as $\mathbf{B}_4\mathbf{F}_5$ can be expressed as, and computed by, four consecutive multiplications of a vector by tridiagonal matrices, the term $\mathbf{B}_5^{-1}\hat{\mathbf{F}}_5$ in Eq. (14c) can be expressed as five consecutive tridiagonal inverses operating on a vector originally filled with $\hat{\mathbf{F}}_5$. Thus

$$\mathbf{B}_5^{-1}\hat{\mathbf{F}}_5 = \mathbf{B}_{\lambda_1}^{-1}\mathbf{B}_{\lambda_2}^{-1}\mathbf{B}_{\lambda_3}^{-1}\mathbf{B}_{\lambda_4}^{-1}\mathbf{B}_{\lambda_5}^{-1}\hat{\mathbf{F}}_5 \tag{19}$$

and the program

```
  DO 1 N = 1,5
  ALAM = 2.*COS(N*PI/(N + 1))
1 CALL SUBROUTINE TRIC(DYX2, ALAM, V, NX)
```

would carry out the actual arithmetic if the subroutine TRIC performs a single tridiagonal solution on the array V and returns the results in the same array. (Throughout this paper we refer to the solution of a tridiagonal-matrix equation as a "tridiagonal solution.") Both TRIC and TRIM are extremely simple programs to write. Again because of the commutative property, the sequence of operations in Eq. (19) is immaterial.

On the basis of the above discussion, we now note that the solution expressed by Eqs. (14) in matrix algebra would require 16 tridiagonal multiplications, 15 tridiagonal solutions, and eight vector additions.

(b) *Centralized elimination.* Consider next the solution to Eq. (12) started by

$$\begin{bmatrix} \mathbf{B}_1 & -\mathbf{B}_0 & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{B}_2 & -\mathbf{B}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{B}_3-\mathbf{B}_1 & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & -\mathbf{B}_1 & \mathbf{B}_2 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & -\mathbf{B}_0 & \mathbf{B}_1 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \mathbf{V}_4 \\ \mathbf{V}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{B}_0\mathbf{F}_1 \\ \mathbf{B}_1\mathbf{F}_2 + \hat{\mathbf{F}}_1 \\ \mathbf{B}_2\mathbf{F}_3 + \hat{\mathbf{F}}_2 + \hat{\mathbf{F}}_4 \\ \mathbf{B}_1\mathbf{F}_4 + \hat{\mathbf{F}}_5 \\ \mathbf{B}_0\mathbf{F}_5 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{F}}_1 \\ \hat{\mathbf{F}}_2 \\ \hat{\mathbf{F}}_3 \\ \hat{\mathbf{F}}_4 \\ \hat{\mathbf{F}}_5 \end{bmatrix} \tag{20a}$$

and followed by

$$\left.\begin{aligned} \mathbf{V}_1 &= \mathbf{B}_1^{-1}[\hat{\mathbf{F}}_1 + \mathbf{V}_2], \\ \mathbf{V}_2 &= \mathbf{B}_2^{-1}[\hat{\mathbf{F}}_2 + \mathbf{B}_1\mathbf{V}_3], \\ \mathbf{V}_3 &= (\mathbf{B}_3 - \mathbf{B}_1)^{-1}[\hat{\mathbf{F}}_3], \\ \mathbf{V}_4 &= \mathbf{B}_2^{-1}[\hat{\mathbf{F}}_4 + \mathbf{B}_1\mathbf{V}_3], \\ \mathbf{V}_5 &= \mathbf{B}_1^{-1}[\hat{\mathbf{F}}_5 + \mathbf{V}_4]. \end{aligned}\right\} \tag{20b}$$

(Notice that $\mathbf{B}_3 - \mathbf{B}_1$ factors and $\mathbf{B}_1\mathbf{V}_3$ occurs twice.) Again *if the* $\mathbf{B}_n$ *were scalars,* the arithmetic count would be eight additions, four multiplications, and five divisions. This is not largely different from the number of operations required using Eqs. (14) and scalar arithmetic. When the $\mathbf{B}_n$ are matrices, however, the "operation count" for Eqs. (20) is five tridiagonal multiplications, nine tridiagonal solutions, and eight vector additions. This is about half the arithmetic required for the solution represented by Eqs. (14), even when there are only five $\mathbf{V}_k$ vectors involved.

(c) *Cyclic reduction.* Clearly, block matrices having a form similar to the one in Eq. (12) can be more and more efficiently inverted as one decreases the number of simple repetitive matrix multiplications and tridiagonal solutions required in the process. This amounts to making the largest required value of N in Eq. (16a) as small as possible, and to require that the use of those $\mathbf{B}_N$ with large N that do appear, be as infrequent as possible. These concepts are alien to the theory involving the solution of matrix equations with scalar elements. (Simple Gaussian elimination remains among the most efficient techniques for solving a tridiagonal matrix equation with scalar elements.)

Probably the most widely known technique for obtaining this minimization of operations is the method of recursive cyclic reduction, devised by Professor G. Golub with collaboration of Dr. R. Hockney (see Refs. [1–3]). This method is based on odd/even reduction, which was used extensively by Hockney [2] in direct two-dimensional Poisson solvers and was the basis for the extension by Buneman [1] to his double cyclic reduction algorithm for solving Poisson's equation. Cyclic reduction has been studied extensively by Buzbee, Golub, and Nielson [5] and has been used by Martin [11] in a three-dimensional solver. Fourier transform techniques can also be used to increase the efficiency of direct solutions. In certain problems they appear to be more efficient than cyclic reduction (Ref. [3]) and in other problems less (Ref. [16]). In any event, we concentrate here on cyclic reduction because of its greater generality. Cyclic reduction works best on a block matrix having the form displayed in Eq. (12) when the block dimension lies in the set $2^L - 1$ where L is an integer. In such a case the matrix operators are

$$\mathbf{C}_N = \prod_{n=1}^{N} (\mathbf{C} - \lambda_n \mathbf{I}), \tag{21a}$$

where

$$\lambda_n = 2\cos[(2n-1)\pi/2N], \quad n = 1, 2, \ldots, N, \tag{21b}$$

$$N = 2^l, \quad l = 1, 2, \ldots, L-1. \tag{21c}$$

Notice that the following recursion relation exists

$$\left.\begin{aligned} \mathbf{C}_1 &= \mathbf{C} \\ \mathbf{C}_2 &= \mathbf{C}_1^2 - 2\mathbf{I} \\ \mathbf{C}_4 &= \mathbf{C}_2^2 - 2\mathbf{I} \\ &\vdots \\ \mathbf{C}_N &= \mathbf{C}_{N/2}^2 - 2\mathbf{I} \end{aligned}\right\}. \tag{21d}$$

We make use of these concepts in the following sections.

Cyclic Reduction of Particular Block Matrices

The block matrix $\mathbf{A}_1$ in Eqs. (11) differs from that shown in Eq. (12) by an entry in the right column. For example, given eight rows of unknowns, Eq. (11a) can be written

$$\mathbf{A}_1\mathbf{V} = \begin{bmatrix} \mathbf{C} & -\mathbf{I} & & & & & & \\ -\mathbf{I} & \mathbf{C} & -\mathbf{I} & & & & & \\ & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & & & & \\ & & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & & & \\ & & & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & & \\ & & & & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & \\ & & & & & -\mathbf{I} & \mathbf{C} & -2\mathbf{I} \\ & & & & & & -\mathbf{I} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \mathbf{V}_4 \\ \mathbf{V}_5 \\ \mathbf{V}_6 \\ \mathbf{V}_7 \\ \frac{1}{2}\mathbf{V}_8 \end{bmatrix} = \mathbf{F}. \qquad (22a)$$

Cyclic reduction (multiplying the even rows by $\mathbf{C}$ and adding the adjacent odd rows) leads successively, on the left side, to

$$\begin{bmatrix} \mathbf{C}_2 & -\mathbf{I} & & \\ -\mathbf{I} & \mathbf{C}_2 & -\mathbf{I} & \\ & -\mathbf{I} & \mathbf{C}_2 & -2\mathbf{I} \\ & & -\mathbf{I} & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_2 \\ \mathbf{V}_4 \\ \mathbf{V}_6 \\ \frac{1}{2}\mathbf{V}_8 \end{bmatrix}, \qquad (22b)$$

$$\begin{bmatrix} \mathbf{C}_4 & -2\mathbf{I} \\ -\mathbf{I} & \mathbf{C}_4 \end{bmatrix} \begin{bmatrix} \mathbf{V}_4 \\ \frac{1}{2}\mathbf{V}_8 \end{bmatrix}, \qquad (22c)$$

and

$$[\mathbf{C}_8][\tfrac{1}{2}\mathbf{V}_8], \qquad (22d)$$

where the $\mathbf{C}_N$ are given in terms of $\mathbf{C}$ by Eqs. (21). The solution is found for $\mathbf{V}_8$, from which $\mathbf{V}_4$ follows, then $\mathbf{V}_6$ and $\mathbf{V}_2$, etc.

The simplicity of this algorithm derives from the choice of the particular differencing formulation used for U_y in the vorticity equation along the upper boundary, the top row in Fig. 2(b)(Eq. (4c)). If, instead, the scheme illustrated in Fig. 2(a) had been used, rows 7 and 8 on the left side of Eq. (22a) would have been

$$\begin{bmatrix} \cdot & \cdot & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & -\mathbf{I} & \mathbf{C} & -\mathbf{I} \\ & & & -\mathbf{I} & \mathbf{C}-\mathbf{I} \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{V}_7 \\ \mathbf{V}_8 \end{bmatrix}.$$

The solution to a set of equations ending in this fashion is not so straightforward, [illegible]

For cyclic-reduction to work best on the matrix $\mathbf{A}_1$, the number of rows of open circles in Fig. 2(b) should be in the set 2, 4, 8, 16,..., 2^L. The number of columns of circles in Fig. 2(b) is immaterial, affecting only the size of the tridiagonal $\mathbf{C}$ in Eq. (13a).

Note at this point that Eq. (12) has the same form as is obtained in solving Poisson's equation with Dirichlet conditions. In that case, the values of l indicated in Eq. (21c)($l = 1, 2,..., L - 1$) are appropriate, where the number of rows in the block matrix is $2^L - 1$. However, Eq. (22a) actually has a form that could be obtained in solving *Poisson's* equation with a *Neumann* condition on one boundary (cf. Ref. [5]). In this case, the number of rows of blocks in Eq. (22a) is 2^L, and the values of l used in Eq. (21c) should be $l = 1, 2,..., L$.

Special Treatment to Eliminate Roundoff Difficulties

The discussion in the two previous sub-sections implies the use of both matrix multiplications and tridiagonal solutions. Unfortunately, repeated matrix multiplications can lead to very large numbers, consequent losses of accuracy due to roundoff errors, and eventual breakdown. For this reason Buneman [1] proposed a procedure that completely avoids matrix multiplication in the solution process. The key to this procedure can be appreciated by regrouping the right side of the equations that result from the sequence shown in Eq. (22) under the constraint that (a) $\mathbf{C}_N$ is made to be a factor of one of two sets of terms on the right, and (b) the grouping is recursive. For example, we set

$$\begin{bmatrix} \mathbf{C}_2 & -\mathbf{I} & & \\ -\mathbf{I} & \mathbf{C}_2 & -\mathbf{I} & \\ & -\mathbf{I} & \mathbf{C}_2 & -2\mathbf{I} \\ & & -\mathbf{I} & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_2 \\ \mathbf{V}_4 \\ \mathbf{V}_6 \\ \frac{1}{2}\mathbf{V}_8 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_2\mathbf{q}_2^{(1)} + \mathbf{p}_2^{(1)} \\ \mathbf{C}_2\mathbf{q}_4^{(1)} + \mathbf{p}_4^{(1)} \\ \mathbf{C}_2\mathbf{q}_6^{(1)} + \mathbf{p}_6^{(1)} \\ \mathbf{C}_2\mathbf{q}_8^{(1)} + \mathbf{p}_8^{(1)} \end{bmatrix}, \tag{23b}$$

$$\begin{bmatrix} \mathbf{C}_4 & -2\mathbf{I} \\ -\mathbf{I} & \mathbf{C}_4 \end{bmatrix} \begin{matrix} \mathbf{V}_4 \\ \frac{1}{2}\mathbf{V}_8 \end{matrix}\Bigg]\Bigg[= \begin{bmatrix} \mathbf{C}_4\mathbf{q}_4^{(2)} + \mathbf{p}_4^{(2)} \\ \mathbf{C}_4\mathbf{q}_8^{(2)} + \mathbf{p}_8^{(2)} \end{bmatrix}, \tag{23c}$$

$$[\mathbf{C}_8][\tfrac{1}{2}\mathbf{V}_8] = [\mathbf{C}_8\mathbf{q}_8^{(3)} + \mathbf{p}_8^{(3)}], \tag{23d}$$

and require in proceeding from (23b) to (23c) that

$$\mathbf{C}_4\mathbf{q}_4^{(2)} + \mathbf{p}_4^{(2)} = \mathbf{C}_2[\mathbf{C}_2\mathbf{q}_4^{(1)} + \mathbf{p}_4^{(1)}] + \mathbf{C}_2\mathbf{q}_2^{(1)} + \mathbf{p}_2^{(1)} + \mathbf{C}_2\mathbf{q}_6^{(1)} + \mathbf{p}_6^{(1)},$$

$$\mathbf{C}_4\mathbf{q}_8^{(2)} + \mathbf{p}_8^{(2)} = \mathbf{C}_2[\mathbf{C}_2\mathbf{q}_8^{(1)} + \mathbf{p}_8^{(1)}] + \mathbf{C}_2\mathbf{q}_6^{(1)} + \mathbf{p}_6^{(1)},$$

etc. If such a construction for the $\mathbf{p}_k^{(l)}$ and $\mathbf{q}_k^{(l)}$ is possible without performing any matrix multiplication, then no multiplications are required in the whole process since $\mathbf{C}_N^{-1}\mathbf{C}_N = \mathbf{I}$ and the multiplication implied by $\mathbf{C}_N\mathbf{q}_k^{(l)}$ is cancelled by the inverse. With some straightforward algebra one can show

$$\mathbf{p}_k^{(0)} = \mathbf{F}_k, \qquad (k = 1, 2,..., NY), \tag{24a}$$

$$\mathbf{p}_k^{(1)} = \mathbf{p}_{k-1}^{(0)} + \epsilon\mathbf{p}_{k+1}^{(0)} + \mathbf{C}^{-1}[2\mathbf{p}_k^{(0)}], \qquad (k = 2, 4,..., NY), \tag{24b}$$

$$\begin{aligned}\mathbf{p}_k^{(2)} &= \mathbf{p}_{k-2}^{(1)} - \mathbf{p}_{k-1}^{(0)} + \mathbf{p}_k^{(1)} - \epsilon\mathbf{p}_{k+1}^{(0)} + \epsilon\mathbf{p}_{k+2}^{(1)} + \mathbf{C}_2^{-1}[-\mathbf{p}_{k-3}^{(0)} + \mathbf{p}_{k-2}^{(1)} - \mathbf{p}_{k-1}^{(0)} + 2\mathbf{p}_k^{(1)} \\ &\quad - \epsilon\mathbf{p}_{k+1}^{(0)} + \epsilon\mathbf{p}_{k+2}^{(1)} - \epsilon\mathbf{p}_{k+3}^{(0)}], \qquad (k = 4, 8,..., NY),\end{aligned} \tag{24c}$$

where $\epsilon = 0$ if the subscript is greater than the block dimension NY of the block matrix, $\mathbf{A}_1$ (i.e., the number of rows of open circles in Fig. 2(b)). Otherwise, ϵ is 1. The general term for $l \geqslant 2$ is

$$\begin{aligned}\mathbf{p}_k^{(l)} &= \mathbf{p}_{k-2h}^{(l-1)} - \mathbf{p}_{k-h}^{(l-2)} + \mathbf{p}_k^{(l-1)} - \epsilon\mathbf{p}_{k+h}^{(l-2)} + \epsilon\mathbf{p}_{k+2h}^{(l-1)} + \mathbf{C}_{N'}^{-1}[-\mathbf{p}_{k-3h}^{(l-2)} + \mathbf{p}_{k-2h}^{(l-1)} - \mathbf{p}_{k-h}^{(l-2)} \\ &\quad + 2\mathbf{p}_k^{(l-1)} - \epsilon\mathbf{p}_{k+h}^{(l-2)} + \epsilon\mathbf{p}_{k+2h}^{(l-1)} - \epsilon\mathbf{p}_{k+3h}^{(l-2)}], \qquad (k = 4h, 8h, 12h,..., NY),\end{aligned} \tag{24d}$$

where $h = 2^{(l-2)}$, $N' = 2^{(l-1)}$, and the $\mathbf{C}_{N'}$ are given by Eqs. (21) with N replaced by N'. The expressions for $\mathbf{q}_k^{(l)}$ involve only simple combinations of the $\mathbf{p}_k^{(l)}$, thus

$$\mathbf{q}_k^{(0)} = 0, \tag{25a}$$

$$\mathbf{q}_k^{(1)} = \tfrac{1}{2}(-\mathbf{p}_{k-1}^{(0)} + \mathbf{p}_k^{(1)} - \epsilon\mathbf{p}_{k+1}^{(0)}), \tag{25b}$$

$$\mathbf{q}_k^{(2)} = \tfrac{1}{2}(-\mathbf{p}_{k-2}^{(1)} + \mathbf{p}_k^{(2)} - \epsilon\mathbf{p}_{k+2}^{(1)}), \tag{25c}$$

and, in general, for $l \geqslant 2$,

$$\mathbf{q}_k^{(l)} = \tfrac{1}{2}(-\mathbf{p}_{k-2h}^{(l-1)} + \mathbf{p}_k^{(l)} - \epsilon\mathbf{p}_{k+2h}^{(l-1)}). \tag{25d}$$

Only the $\mathbf{p}_k^{(l)}$ are needed in the forward recursion (to be illustrated below). The $\mathbf{q}_k^{(l)}$ are calculated in the backward recursion but are never stored.

Summary of Procedure for Determining V

The rather ponderous notation used in Eqs. (23)–(25) may lead to the implication that the calculation procedure is complicated. Actually, quite the reverse is true. The computations are easy to program and necessitate only a minimum of computer storage (for example, $\mathbf{f}_k$, $\mathbf{F}_k$, $\mathbf{p}_k^{(l)}$, and $\mathbf{V}_k$ all occupy the same array of dimension NX in memory, as do $\mathbf{g}_k$, $\mathbf{G}_k$, and $\mathbf{U}_k$).

In order to demonstrate the simplicity, we have prepared Fig. 3 (cf. Ref. [3]) to illustrate what actually is required for the direct cyclic reduction for a mesh with eight rows. In the first place, none of the algebra on the left side of Eqs. (23) takes place in the computer. Only the right side is formed and only the values of

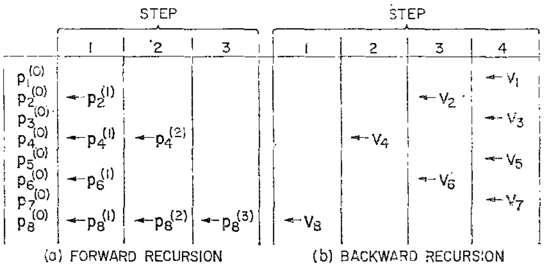

FIG. 3. Illustration of direct cyclic reduction to obtain $\mathbf{V}$ for $N = 8$.

$\mathbf{p}_k^{(l)}$ need be stored. Initially eight vector arrays of dimension NX are filled with data representing the boundary conditions and the source and vorticity distributions according to Eqs. (6) and (9). The preliminary computations of the $\mathbf{F}_k$ for use in Eq. (24a) from Eq. (9) are straightforward. For generality in presentation, $\mathbf{F}_k$ is labeled $\mathbf{p}_k^{(0)}$ (see Eq. (24a)) and appears in the left column of Fig. 3. The next column is formed according to Eq. (24b) operating on vectors to its left and overstoring the old information in the even numbered rows. Successive columns are formed, each time overstoring the information in the arrays indicated in the figure. At any stage the information required to compute $\mathbf{p}_k$ in Eqs. (24) is just that information stored in itself and neighboring vectors from previous calculations. In the formation of each column, only repetitive *tridiagonal solutions* (*no* multiplications) and vector additions are required. The tridiagonal solutions are performed successively (after *factoring* $\mathbf{C}_N$ into the tridiagonal matrices according to Eqs. (21)) in a manner similar to that illustrated by Eq. (19).

The backward recursion is illustrated in the right side of Fig. 3. The procedure would be to compute (from (23d))

$$\tfrac{1}{2}\mathbf{V}_8 = \mathbf{C}_8^{-1}\mathbf{p}_8^{(3)} + \mathbf{q}_8^{(3)}, \tag{26a}$$

where $\mathbf{q}_8^{(3)}$ is given by Eq. (25d) with $l = 3$. Then (from (23c))

$$\mathbf{V}_4 = \mathbf{C}_4^{-1}(\mathbf{p}_4^{(2)} + \mathbf{V}_8) + \mathbf{q}_4^{(2)} \tag{26b}$$

with Eq. (25c) for $l = 2$, etc. The last step ($l = 0$, last column in Fig. 3) is performed using the odd lines in Eq. (22a) with the right sides consisting of $\mathbf{p}_k^{(0)}$; thus

$$\mathbf{V}_1 = \mathbf{C}^{-1}(\mathbf{p}_1^{(0)} + \mathbf{V}_2), \tag{26c}$$

$$\mathbf{V}_3 = \mathbf{C}^{-1}(\mathbf{V}_2 + \mathbf{p}_3^{(0)} + \mathbf{V}_4), \text{ etc.} \tag{26d}$$

Again, (i.e., for the entire backward recursion) only tridiagonal solutions and vector additions are required, and as the overstoring of the $\mathbf{p}_k$ vectors by the newly computed $\mathbf{V}_k$ vectors proceeds, at any given step just that data is available that is necessary for the computation of the next step. At the end of backward recursion, the arrays that initially contained $\mathbf{F}_k$ now contain the solution for $\mathbf{V}_k$ defined in Eq. (6).

Determination of U

With the $\mathbf{V}_k$ known, the $\mathbf{U}_k$ could be determined from Eq. (11b). The simplest procedure, however, is to go back to the lower half of Eq. (7). The results for the same example as in the above section ($N = 8$) are

$$\left.\begin{aligned} \mathbf{U}_8 &= \tfrac{1}{2}\mathbf{B}\mathbf{V}_8 - \mathbf{g}_8, \\ \mathbf{U}_7 &= \mathbf{U}_8 + \mathbf{B}\mathbf{V}_7 - \mathbf{g}_7, \\ &\vdots \\ \mathbf{U}_1 &= \mathbf{U}_2 + \mathbf{B}\mathbf{V}_1 - \mathbf{g}_1, \end{aligned}\right\} \tag{27}$$

where the j component of each product $\mathbf{B}\mathbf{V}_k$ in Eq. (27) is simply $\mu(V_{j,k} - V_{j+1,k})$ and where $V_{j+1,k}$ is zero if $j + 1$ is greater than the number of columns of open circles in Fig. 2(b).

Analogy to Solution of Poisson's Equation

At the beginning of this section there was mentioned an analogous procedure involving a Poisson equation for V. If s and ω were differentiable, Eqs. (1) could be combined to obtain

$$\partial^2 V/\partial x^2 + \partial^2 V/\partial y^2 = \partial s/\partial y + \partial \omega/\partial x.$$

This equation could be integrated numerically using an available direct Poisson solver, and the corresponding U could then be obtained by a quadrature.

The multiplications by $\mathbf{A}$ (or $\mathbf{B}$) in (7) represent $\partial/\partial x$, and the operator $-\mathbf{I}\mathbf{V}_{k-1} + \mathbf{I}\mathbf{V}_k$ represents $\partial/\partial y$; their combination that appears in the $\mathbf{A}_1$ matrix in Eqs. (11) represents a discrete Laplacian operator on V with three-point central differences for the second derivatives. However, the solution algorithm proposed for the nonhomogeneous Cauchy–Riemann equations is not equivalent to the solution of a Poisson equation in terms of V. Two important differences are the following:

(a) The matrix manipulations in the algorithm do not necessarily represent the replacement of derivatives by finite differences. For example, neither these manipulations nor the accuracy of results depend in any way on the differentiability of s and ω. This fact is especially important in planned future applications in which s and ω have values only at isolated discrete points.

(b) The Cauchy–Riemann solution algorithm is to be used in iterative schemes for nonlinear problems in which s and ω represent complicated nonlinear terms that not only vary rapidly but even change form between adjacent mesh points. These varying forms would not be directly adaptable to the Poisson equa-

Example Problem in Aerodynamics: Thin Biconvex Airfoil

As an example to illustrate the accuracy, speed, and other properties of the direct solver in a typical application, we consider the small-perturbation solution for steady, irrotational, subsonic, inviscid flow over a thin, symmetrical parabolic-arc biconvex airfoil. This example problem is chosen because (a) it has a rather simple mathematical formulation, (b) the analytical solution is available for comparison with the numerical results, (c) the analytical solution contains singularities which should be "captured" to a certain degree by the numerical solution, and (d) the problem has some direct extensions that are of high current interest in nonlinear subsonic and transonic aerodynamics.

In addition to the conditions mentioned above, the flow is assumed to be uniform at infinity (far from the airfoil) with velocity U_∞ (from left to right), which is aligned with the airfoil chord. Denote the free-stream Mach number as M. Denote by U and V, respectively, the x and y components of total velocity, with the x axis along the airfoil chord and the y axis as the bisector of the airfoil chord. Both x and y are normalized by the chord length. The biconvex airfoil surface is designated by

$$y_b(x) = \pm\, \epsilon(0.5 - 2x^2), \quad (-0.5 \leqslant x \leqslant 0.5), \tag{28}$$

where ϵ is now defined as the ratio of maximum airfoil thickness to chord length.

Small-Perturbation Problem and its Analytical Solution

Use of the classical small-perturbation approximations

$$U = U_\infty(1 + \epsilon u), \quad V = U_\infty \epsilon v, \tag{29}$$

substituted into the governing conservation equations, leads to the approximate thin-airfoil problem (to lowest order in ϵ) for $u(x, y; M)$ and $v(x, y; M)$ (see Refs. [12, 13]), in which the flow tangency condition on $y = y_b(x)$ is transferred to $y = 0$ by use of Taylor's series (see, e.g., Ref. [14]).

The Prandtl–Glauert similarity transformation, which converts the problem for

$0 \leqslant M < 1$ to an equivalent incompressible problem (see, e.g., Ref. [12, pp. 30–31], or Ref. [15, p. 64 ff]), is

$$u(x, y; M) = (1/\beta)\,\bar{u}(x, \bar{y}), \tag{30a}$$

$$v(x, y; M) = \bar{v}(x, \bar{y}), \tag{30b}$$

$$y = (1/\beta)\bar{y}, \tag{30c}$$

where

$$\beta = (1 - M^2)^{1/2}. \tag{31}$$

Then the equivalent problem (for the half plane, $\bar{y} \geqslant 0$) is

$$\bar{u}_x + \bar{v}_{\bar{y}} = 0, \tag{32a}$$

$$\bar{u}_{\bar{y}} - \bar{v}_x = 0, \tag{32b}$$

with the conditions

$$\bar{v}(x, 0^+) = -4x \; (-0.5 < x < 0.5), \tag{32c}$$

$$= 0 \; (|\, x \,| > 0.5), \tag{32d}$$

$$\bar{u}, \bar{v} \to 0 \text{ as } x^2 + \bar{y}^2 \to \infty. \tag{32e}$$

The analytical solution to Eqs. (32) (see Ref. [12, Table A, 2, p. 21]), in terms of the complex variable $z \equiv x + i\bar{y}$, is

$$\bar{u} - i\bar{v} = \frac{4}{\pi}\left[1 - z \ln\left(\frac{z + 0.5}{z - 0.5}\right)\right]. \tag{33}$$

In particular, at $\bar{y} = 0$ (where $\bar{v}$ is given on the upper side by Eqs. (32c) and (32d)),

$$\bar{u}(x, 0) = \frac{4}{\pi}\left(1 - x \ln\left|\frac{x + 0.5}{x - 0.5}\right|\right). \tag{34}$$

Note, from Eqs. (32c) and (32d), that $\bar{v}$ is double-valued at the leading and trailing edges in the perturbation problem, and from Eq. (34) that $\bar{u}$ goes to minus infinity at the leading and trailing edges.

Direct Numerical Solution

To solve Eqs. (32) numerically we note that the differential equations are the same as Eq. (1) with $s = \omega = 0$ and with U, V, and y replaced respectively by $\bar{u}$, $\bar{v}$, and $\bar{y}$. Therefore the direct solver using recursive cyclic reduction described above can be used. The computational algorithm requires $\bar{u}$ to be specified on the left and upper boundary lines and $\bar{v}$ to be specified on the right and lower boundary lines. The condition specified on $\bar{y} = 0$ is given by Eqs. (32c) and (32d). For the other three boundary lines, the infinity condition, Eq. (32e), is treated in different ways as described in the next subsection.

After $\bar{u}$ and $\bar{v}$ are determined at the x and $\bar{y}$ corresponding to the staggered mesh points indicated by open squares and circles in Fig. 2(b)(but with a larger number) it is desired to determine numerical values of $\bar{u}$ on $\bar{y} = 0$ for comparison with the analytical solution (Eq. (34)). For this purpose, the following second-order-accurate formula for $U_{j,0}$ was derived using a combination of differencing schemes and the basic equation (1b),

$$U_{j,0} = (1/8)[9U_{j,1} - U_{j,2} + 3(\Delta y/\Delta x)(V_{j,0} - V_{j+1,0}) + 3(\Delta y)\,\omega_{j,0}], \quad (j = 1, 2,..., NX), \tag{35}$$

in which $k = 0$ denotes the value of the U index, as well as the value of the V index, on $y = 0$ (the bottom dashed line on Fig. 2(b)). The numerical values of $\bar{u}$ are obtained by replacing U, V, and Δy by $\bar{u}$, $\bar{v}$, and $\Delta\bar{y}$ and setting $\omega = 0$ in Eq. (35).

Numerical Results for Different Outer Conditions and Comparison with Analytical Perturbation Solution

Numerical results for $\bar{u}$ and $\bar{v}$ were obtained using the algorithm described above, with the mesh parameters $NY = 32$ (the number of open squares or circles in a *column* in Fig. 2(b), which must be an integer power of 2, i.e., 2^L) and $NX = 39$ ($NX =$ the number of open symbols in a horizontal *row* in Fig. 2(b)). The value of $\bar{y} = (1 - M^2)^{1/2}y$ for the upper boundary for all cases was 2.0, so that $\Delta\bar{y} = 1/16$. The x locations of the left and right boundaries were either equal to or close to ± 1.0 (so that $\Delta x = 1/20$), those values varying according to the desired x locations of input values of $\bar{v}$ on $\bar{y} = 0$. Thus the upstream and downstream computation boundaries were at or about one-half chord length from the airfoil, and the upper boundary was at two chord lengths.

The differences in the results shown in Figs. 4–6 depend on (a) the way the exterior condition (32e) was imposed in the numerical problem, and (b) the x locations of the imposed discrete conditions from Eqs. (32c) and (32d). For both Figs. 4 and 5 the exterior condition at infinity was replaced by imposing the exact values of $\bar{u}$ and $\bar{v}$ given by Eq. (33) along the outer boundaries. This was done so that the first illustrations of this technique would not be affected by errors due to approximate methods for applying conditions at infinity. (A similar procedure was used in Ref. [8] for evaluating results of a technique for second-order equations with elliptic operators.) Then for Fig. 6 the perturbation velocities were set at zero on the exterior computation boundary. This is the most usual approximate way of applying exterior conditions for problems involving no lift when the analytical solution is not known. The differences between Figs. 4 and 5 are in the x locations of the imposed $\bar{v}$ on $\bar{y} = 0$. For Fig. 4 the input x locations for $\bar{v}(x, 0^+)$(solid circles in Fig. 4 and on bottom row of Fig. 2(b)) straddle both the leading and trailing edges of the airfoil. For Fig. 5 the input x locations for $\bar{v}$ $(x, 0^+)$ coincide with both the leading edge and trailing edge. Since in Eqs. (32c) and (32d) $\bar{v}$ is double-valued when those two points ($x = \pm 0.5$) are approached from the left

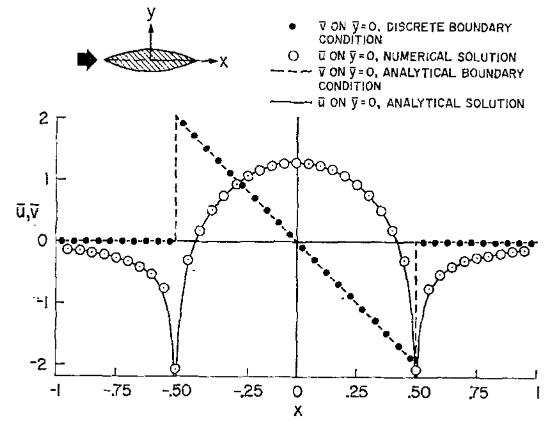

FIG. 4. Transformed Prandtl–Glauert perturbation velocities for thin parabolic-arc biconvex airfoil with prescribed $\bar{v}$ on chord line at points on mesh *straddling* the airfoil edges and with *exact* perturbation conditions on exterior boundaries ($NX = 39$, $NY = 32$).

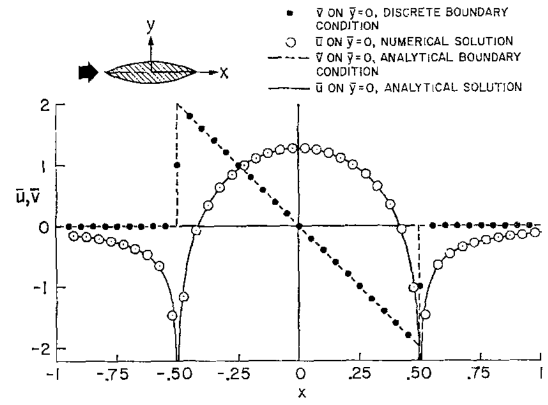

FIG. 5. Transformed Prandtl-Glauert perturbation velocities for thin parabolic-arc biconvex airfoil with prescribed $\bar{v}$ on chord line at points on mesh *coinciding with* the airfoil edges and with *exact* perturbation conditions on exterior boundaries ($NX = 39$, $NY = 32$).

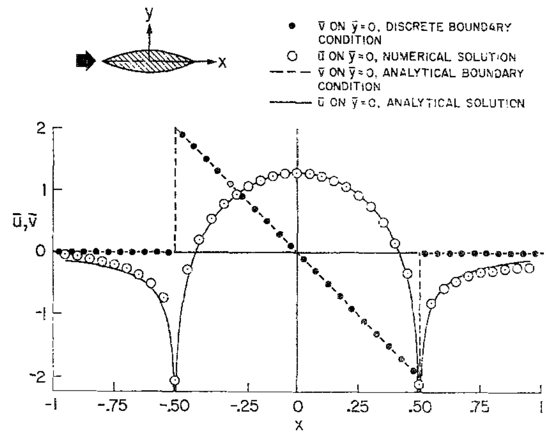

FIG. 6. Transformed Prandtl–Glauert perturbation velocities for thin parabolic-arc biconvex airfoil with prescribed $\bar v$ on chord line at points on mesh *straddling* the airfoil edges and with *zero* perturbation conditions on exterior boundaries ($NX = 39$, $NY = 32$).

and from the right, the average values were used, $\bar v(\pm 0.5, 0^+) = \mp 1.0$ (see solid circles on Fig. 5).

The numerical results for $\bar u$ $(x, 0)$ in Figs. 4 and 5 (for exact conditions specified on the computation boundary) are accurate for both cases. The singularities at the edges are captured well in both figures, but this is most evident in Fig. 4 where $\bar u$ is computed at exactly those points where the analytical solution is infinite. The accuracy is surprising in view of the facts (a) that the numerical values of $\bar u(x, 0)$ are determined from a formula, Eq. (35), that involves numerical differentiation of data already obtained from the numerical solution, and (b) that this is true even

special treatment of those points. Further, combination or superposition of the results of Figs. 4 and 5 would yield a mean solution that would be even more accurate than either figure alone.

Fig. 6 corresponds to Fig. 4 except that zero perturbations are specified on the outer boundaries. Clearly, the numerical results differ significantly from the analytical solution near the outer computation boundaries. At the upstream computation boundary, the computed $\bar u(x, 0)$ goes to zero; at the downstream boundary, where $\bar v$ was specified as zero, the solution for $\bar u$ is also significantly in error but is not zero. In spite of the proximity of the computation boundaries and in spite of these expected errors near the computation boundaries, the numerical results are quite accurate for x locations on the airfoil. Again (as in Figs. 4 and 5) the edge singularities are captured well.

It is noted that the limitation of first-order accuracy at the upper boundary in

Fig. 2(b) did not have any significant effect on the accuracy of the solution near the airfoil because the flow variations at the upper (outer) boundary are very small.

Computing Times

The computation times required for this fast, direct solver are of special interest. Table I lists computing times for significant portions of different cases of a program comparable to that for the airfoil problem using the same fast direct U, V solver. The computing times were measured on the IBM 360/67 Operating System. The computer programs used Gaussian elimination (as in Ref. [11]) for all the simple tridiagonal solutions performed within the block cyclic reduction. For each case,

TABLE I

Computing Times for U, V Solver on *IBM* 360/67 (FORTRAN IV, Level *H*, *OPT* = 2)

NX	NY	t_1 (sec)	t_2 (sec)	t_3 (sec)
9	8	<.01	.02	<.01
19	16	.01	.20	<.01
39	32	.05	.55	.03
59	64	.14	1.95	.11
99	128	.43	7.50	.37

NX and NY are the mesh parameters defined in the section above, with reference to Fig. 2(b). The time t_1 includes: all preliminary calculations of parameters, initializing interior values of s and ω, loading boundary values of U and V, modifying the fringe of the interior to include boundary values (to get $\mathbf{f}_k$ and $\mathbf{g}_k$ in Eq. (7)), zeroing boundary values, manipulating $\mathbf{f}_k$ and $\mathbf{g}_k$ to get $\mathbf{F}_k$ and $\mathbf{G}_k$, and computing and storing all needed values of the λ_n for each required l. The time t_2 is the time required for the cyclic reduction starting with the $\mathbf{p}_k^{(0)}$ and resulting in the $\mathbf{V}_k$ (see Fig. 3). Finally, t_3 is the time required to obtain $\mathbf{U}$ after $\mathbf{V}$ is known.

Note that t_1 and t_3 combined are about 10 % of t_2. Note also that t_2 is just that time which would be required by a direct cyclic-reduction Poisson solver to compute the velocity potential or stream function. If the results of a Poisson solver were used to compute the velocity components throughout the entire field (which are automatically provided by the Cauchy–Riemann solver), the times required by the two algorithms would be even closer.

Concluding Remarks

The above example problem in subsonic aerodynamics, including the comparison of the numerical solutions with the analytical solution, has demonstrated several attractive features of the new direct Cauchy–Riemann solver. First, the results are highly accurate and capture well the singularities of the analytical solution. Thus, whereas generalized relaxation schemes for such problems experience convergence difficulties when these "peaks" occur, the direct solver easily produces an accurate solution in a single step. Second, the measured computing times have also demonstrated the high efficiency of the direct solver.

In future applications, the fast, direct Cauchy–Riemann solver is expected to be useful in aerodynamic flow computations for which the right sides in the set of first-order elliptic equations are not zero (accounted for in the developed algorithm). These anticipated applications include nonlinear equations, with the direct solver being used within an iteration scheme. In addition it may be possible to extend the algorithm to three dimensions as was done for the second-order elliptic (Poisson) solver (Ref. [11]) and also to further generalize the generalized-capacity-matrix technique (Ref. [8]) to first-order equations for applying the direct Cauchy–Riemann solver in problems with interior boundary conditions.

In summary, the results of the example problem have established the potential usefulness of the new fast, direct Cauchy–Riemann solver in computing aerodynamic flows, as well as in the solution of other similar problems in mathematical physics governed by similar equations.

References

1. O. Buneman, "A Compact Non-Iterative Poisson Solver," SUIPR Rept. No. 294, Inst. Plasma Research, Stanford University, Stanford, CA, 1969.
2. R. W. Hockney, *J. Assoc. Comp. Mach.* **12** (1965), 95.
3. R. W. Hockney, *Methods Computational Phys.* **9** (1970), 135.
4. F. W. Dorr, *SIAM Rev.* **12** (1970), 248.
5. B. L. Buzbee, G. H. Golub, and C. W. Nielson, *SIAM J. Numer. Anal.* **7** (1970), 627.
6. R. C. LeBail, *J. Computational Phys.* **9** (1972), 440.
7. P. J. Roache, "Computational Fluid Dynamics," Hermosa Publishers, Albuquerque. NM, 1972.
8. E. D. Martin, "A Generalized-Capacity-Matrix Technique for Computing Aerodynamic Flows," Paper presented at the Symposium on Application of Computers to Fluid Dynamics Analysis and Design, Polytechnic Institute of Brooklyn Graduate Center, Farmingdale, NY, Jan. 3–4, 1973. To appear in *Computers and Fluids.*
9. J. L. Steger and H. Lomax, Generalized relaxation methods applied to problems in transonic flow, *in* "Lecture Notes in Physics, Vol. 8, Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics" (M. Holt, Ed.), p. 193, Springer-Verlag, Berlin/Heidelberg/New York, 1971.

10. J. L. Steger and J. M. Klineberg, *AIAA J.* **11** (1973), 628.
11. E. D. Martin, *Internat. J. Numer. Methods Engr.* **6** (1973), 201.
12. R. T. Jones and D. Cohen, Aerodynamics of wings at high speeds, *in* "Aerodynamic Components of Aircraft at High Speeds" (A. F. Donovan and H. R. Lawrence, Eds.), Vol. VII, Section A, High Speed Aerodynamics and Jet Propulsion, Princeton University Press, Princeton, NJ, 1957. Also available as Princeton Aeronautical Paperback No. 6 entitled "High Speed Wing Theory," 1960.
13. L. Prandtl, "General considerations on the flow of compressible fluids," NACA Tech. Mem. 805, 1936.
14. M. J. Lighthill, Higher approximations, Section E, *in* "General Theory of High Speed Aerodynamics" (W. R. Sears, Ed.), Vol. VI, High Speed Aerodynamics and Jet Propulsion, Princeton University Press, Princeton, NJ, 1954. Also available as Princeton Aeronautical Paperback No. 5, entitled "Higher Approximations in Aerodynamic Theory," 1960.
15. W. R. Sears, Small perturbation theory, Section C, *in* "General Theory of High Speed Aerodynamics" (W. R. Sears, Ed.), Vol. VI, High Speed Aerodynamics and Jet Propulsion, Princeton University Press, Princeton, NJ, 1954. Also available as Princeton Aeronautical Paperback No. 4 entitled "Small Perturbation Theory," 1960.
16. R. A. Sweet, *J. Computational Phys.* **12** (1973), 422.