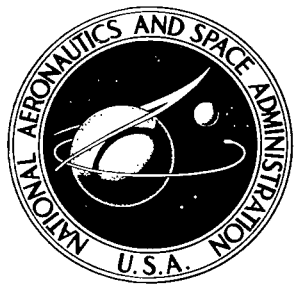
NASA TECHNICAL NOTE

NASA TN D-7934



VARIANTS AND EXTENSIONS OF A FAST DIRECT NUMERICAL CAUCHY-RIEMANN SOLVER, WITH ILLUSTRATIVE APPLICATIONS

E. Dale Martin and Harvard Lomax

Ames Research Center
Moffett Field, Calif. 94035

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • MARCH 1977

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TN D-7934		

4. Title and Subtitle	5. Report Date
VARIANTS AND EXTENSIONS OF A FAST DIRECT NUMERICAL CAUCHY-RIEMANN SOLVER, WITH ILLUSTRATIVE APPLICATIONS	March 1977
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
E. Dale Martin and Harvard Lomax	A-5988
	10. Work Unit No.
9. Performing Organization Name and Address	505-06-12-08
Ames Research Center Moffett Field, California 94035	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Note
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Revised and extended versions of a fast, direct (noniterative) numerical Cauchy-Riemann solver are presented for solving finite-difference approximations of first-order systems of partial differential equations. Although the difference operators treated are linear and elliptic, one significant application of these extended direct Cauchy-Riemann solvers is in the fast, semidirect (iterative) solution of fluid-dynamic problems governed by the nonlinear mixed elliptic-hyperbolic equations of transonic flow. Different versions of the algorithms are derived and the corresponding FORTRAN computer programs for a simple example problem are described and listed. The algorithms are demonstrated to be efficient and accurate.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Fast elliptic solver Numerical solution algorithms Finite differences	Unlimited STAR Category — 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	98	$4.75

*For sale by the National Technical Information Service, Springfield, Virginia 22161

TABLE OF CONTENTS

VARIANTS AND EXTENSIONS OF A FAST DIRECT NUMERICAL CAUCHY-RIEMANN SOLVER, WITH ILLUSTRATIVE APPLICATIONS

E. Dale Martin and Harvard Lomax

Ames Research Center

SUMMARY

Revised and extended versions of a fast, direct (noniterative) numerical Cauchy-Riemann solver are presented for solving finite-difference approximations of first-order systems of partial differential equations. Although the difference operators treated are linear and elliptic, one significant application of these extended direct Cauchy-Riemann solvers is in the fast, semidirect (iterative) solution of fluid-dynamic problems governed by the nonlinear mixed elliptic-hyperbolic equations of transonic flow. Different versions of the algorithms are derived and the corresponding FORTRAN computer programs for a simple example problem are described and listed. The algorithms are demonstrated to be efficient and accurate.

INTRODUCTION

Highly efficient finite-difference algorithms and programs called "fast direct elliptic solvers" are becoming increasingly more useful for solving partial differential equations in practical problems as the techniques that incorporate the solvers, as well as the solvers themselves, become more highly developed. The fast direct solvers differ from other elliptic solution algorithms in that they take advantage of the sparseness and regularity of the highly ordered coefficient matrix of the set of finite-difference equations to obtain a direct solution in a sequence of simple recursive operations. Thus, they are essentially noniterative, and the consequent high efficiency accounts for their attractiveness in various applications.

The purposes of this report are (a) to develop highly efficient and stable numerical algorithms called Cauchy-Riemann solvers for use in methods that treat the first-order elliptic operators in generalized Cauchy-Riemann equations, (b) to describe the development of several different versions of these algorithms, and (c) to provide a simple example computer program for each version, along with the subroutines for the Cauchy-Riemann solvers.

Cauchy-Riemann solvers have significant applications in semidirect (globally implicit) iteration techniques for rapid solution of transonic flows. It is expected that, as the direct solvers become further developed and generalized, the significance of their applications will increase.

Fast direct elliptic solvers were first developed for solving Poisson's equation on a rectangle (refs. 1-5). Since then, the algorithms have been generalized, extended, and applied in numerous ways: (a) Fast solvers have been extended to three dimensions (e.g., refs. 5 and 6, among others). (b) They have been further developed and extended to treat more general second-order elliptic equations and more general boundary conditions and meshes (refs. 5 and 7-17). (c) They have been extended to include interior conditions and irregular domains (refs. 18-23). (d) They have been extended to fourth-order (biharmonic) partial differential equations in references 22 and 24. (e) They have been extended to apply to sets of first-order elliptic equations in reference 25. Chapter 9 in reference 26 also describes an early direct solver for the Cauchy-Riemann equations. However, reference 26 indicates that the algorithm described there has a very low mesh-number limitation, which would severely restrict the usefulness; in addition, that algorithm would be significantly less efficient than the "fast" solvers of references 1 to 5 and 25 that are based on either fast Fourier transforms or cyclic reduction. (f) Direct elliptic solvers have been applied within iteration schemes (i.e., semidirect methods) to extend their usefulness to nonseparable elliptic equations (refs. 27 and 28) and either to Poisson equations as part of a system of nonlinear equations (refs. 29 and 30; see also ref. 31) or as the total driving algorithm in a semidirect iteration of nonlinear equations (refs. 32-34) and of equations that are both nonlinear and nonelliptic in some regions of the solution domain (refs. 35-38). (g) In addition to other applications in the literature (now too numerous to attempt to list here), a notable application of a direct Poisson solver is as an iterative-acceleration device for finite-difference, line-relaxation solutions of the nonlinear, mixed-type equations of transonic flow (refs. 39-41).

The fast Cauchy-Riemann solver developed in reference 25 and used in reference 35, and the more recent variant and extension of it used in references 36 to 38 (to be described below as version C), can be used in problems where a Poisson solver or other second-order elliptic solver could also be used. In addition, however, the direct solution of the corresponding set of first-order equations can be especially useful, for example, in fluid dynamics, because (a) the boundary conditions, including solid-surface conditions and and internal jump (e.g., shock-wave discontinuity or contact discontinuity) conditions, are given most naturally in terms of components of the velocity vector; and (b) the formulation in terms of velocity components (generalized Cauchy-Riemann equations) allows both point sources and point vortices to be included simultaneously. This is contrasted with stream-function and velocity-potential formulations, which are second-order partial differential equations (where velocity components are defined in terms of derivatives of either stream function or velocity potential). The stream-function formulation allows specification of point vortices but not point sources of mass, and the velocity-potential formulation allows specification of point sources but not point vortices.

In the sections below, after description of the computational meshes and mathematical definitions, five versions of the Cauchy-Riemann solver, denoted A to E, are developed. In addition, computer programs including all subroutines for each version are listed in appendices for a simple example problem that illustrates use of the solvers. Version A is the original version of

the Cauchy-Riemann solver derived in reference 25 and used for the slightly supercritical transonic flow problem in reference 35. The present matrix derivation of version A is more concise than the previous derivation, and more details including the program listing can be given here. A significant step for improving the efficiency resulted in the development of version B. Versions B and C are essentially the same except that additional terms (with constant coefficients) needed to stabilize iterations in a semidirect method are included in C. Version C is the solver used for the transonic-flow calculations in references 36 to 38. Versions D and E are developed for future use. They use a mesh that is symmetrical about the x axis, whereas the previous versions used different configurations for upper and lower boundaries on the mesh and different conditions applied there. These versions are especially desired for future treatment of airfoils represented by interior conditions specified on a slit in the center of the mesh. Version D uses constant coefficients on the extra stabilizing terms, whereas version E uses arbitrarily specified, variable (in x) coefficients on the extra terms. Although versions B and C together and D and E together are derived as single algorithms, of which the pairs of versions are special cases, the separate sample programs are provided for all five versions because of the differences in efficiency that may be significant in future applications. The distinguishing features of each version are further described in the sections dealing with the derivations (beginning on pages 18, 21, and 24). A guide for the prospective user's choice of algorithm is given at the end of the section preceding the Concluding Remarks.

GENERAL DIFFERENTIAL AND FINITE-DIFFERENCE EQUATIONS

The algorithm and programs to be developed are for numerical solution of the nonhomogeneous Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = s(x,y) \tag{1a}$$

$$\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = -\omega(x,y) \tag{1b}$$

or of more general extensions of equations (1), which have quite general physical applications. Throughout this report, the terminology of fluid mechanics is used. The dependent variables u and v then frequently represent velocity components. Generally, however, for any two-dimensional vector[1] (u,v), equation (1a) is the definition of the divergence (s) of the vector, and equation (1b) is the definition of the curl (ω) of the vector or, equivalently, the "vorticity" of the vector (u,v). Any point where the divergence s is not zero is called a "source" of the vector (u,v), and any point where ω is not zero is called a "vortex" of the vector (u,v). In any region where

[1]All equations and definitions in this section have three-dimensional counterparts, but the present scope is limited to two dimensions.

s is zero, the vector (u,v) is "solenoidal," and in any region where ω is zero, the vector (u,v) is "irrotational." Consistent with the terminology of fluid mechanics and the above general definitions, then, $s(x,y)$ is here called a "source function," $\omega(x,y)$ is called a "vorticity function," and equations (1) are referred to, respectively, as the "continuity equation" and the "rotationality equation."

The algorithms to be developed can be useful in the efficient numerical solution of generalized Cauchy-Riemann equations; that is, this work is relevant in certain ways to the following more general form of the set of differential equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = s(x,y,u,v,\partial_x,\partial_y) \tag{2a}$$

$$\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = -\omega(x,y,u,v,\partial_x,\partial_y) \tag{2b}$$

where the notation on the right side indicates that s and ω can be arbitrarily specified functions not only of x and y but also of the dependent variables, u and v, and derivatives of arbitrary order of u and v with respect to x and y, with ∂_x being the partial-derivative operator, etc. Thus the equations can be both nonlinear and nonelliptic, even though the left side of equations (2) is a linear elliptic operator.

Iterative schemes for the solution of equations (2) treat the right side as known from a previous iteration. At each iteration, then, the "iteration equations" for solving the system of equations (2) are actually of the form of equations (1), in the simplest case, or they may be of a slightly more general linear form.

The algorithms to be derived for solving equations (1) use central finite differences on the left side. For use in iteration schemes, it has been found (refs. 36-38) that, in some cases, certain stabilizing terms need to be added to both sides of the equations. Then the difference operator on the left will contain terms in addition to those representing the derivatives in equations (1). The most general form of the approximate set of finite-difference equations to be considered here for representing equations (1) is

$$\frac{u^+ - u^-}{h_x} + \frac{v^+ - v^-}{h_y} + a^+u^+ + a^-u^- = a^+u_o^+ + a^-u_o^- + s(\cdot) \tag{3a}$$

$$\equiv s'(\cdot) \tag{3a'}$$

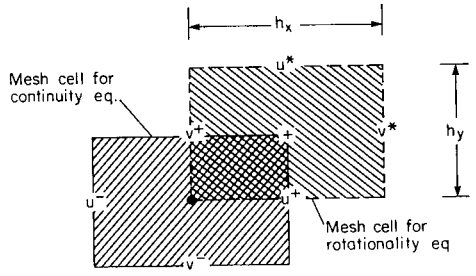
$$\frac{u^* - u^+}{h_y} - \frac{v^* - v^+}{h_x} = -\omega(+) \tag{3b}$$

These equations are written for a staggered mesh, which has been found to have numerous advantages. Refer to figure 1(a), on which: the point at which equation (3a) is being solved is represented by $(\cdot)$; the point at which equation (3b) is being solved is represented by $(+)$; the values of u and v at

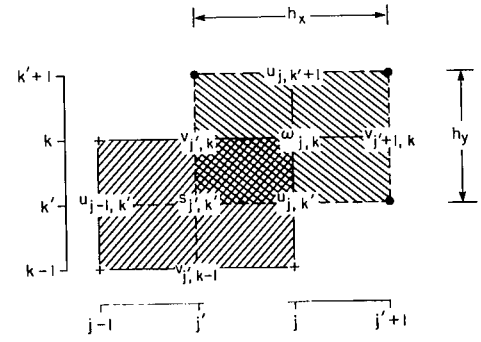
the centers of the boundaries of the mesh cell for the continuity equation (centered at the dot) are u^+, u^-, v^+, and v^-, and the values of u and v at the centers of the boundaries of the mesh cell for the rotationality equation (centered at the cross) are u^*, u^+, v^*, and v^+. The notations $s(\cdot)$ and $\omega(+)$ in equations (3) indicate that the source function s is evaluated at the dot and the vorticity function ω is evaluated at the cross. The quantities a^+ and a^- in equation (3a) may generally vary with x. The symbols u_o^+ and u_o^- in equation (3a) denote some known representations of u^+ and u^-, such as an analytical solution or results from a previous iteration in an iterative solution. The quantity $s'(\cdot)$ is defined to equal the right side of equation (3a). The x and y mesh intervals, h_x and h_y, are constants.

COMPUTATIONAL MESHES AND DISCRETIZED VARIABLES

For the development of the computational algorithms to solve equations (3), the configuration of mesh cells shown in figure 1(a) is imbedded in a large computational mesh. The different versions of the Cauchy-Riemann solver to be described use different treatments of the upper boundary, and so the mesh configurations differ there. However, since the left and bottom boundaries are the same for all versions, a system of mesh-point indexing can be used that is common to all versions, with origins of coordinate indices at or near the left bottom corner.



(a) Basic configuration for differencing.



(b) Indexing of basic configuration.

Figure 1.- Mesh cells for continuity and rotationality equations.

Consider rectangular coordinates x and y and let the left and right boundaries be $x = x_\ell$ and $x = x_u$, and the bottom and top boundaries be $y = y_\ell$ and $y = y_u$. Let j and k index the x and y directions, respectively, so that discrete coordinates x_j and y_k are given by

$$x_j = x_\ell + jh_x , \quad y_k = y_\ell + kh_y \tag{4a}$$

where j and k are integers. Now consider the indexing of the variables in figure 1(a) and refer to figure 1(b). For integers j and k, the point where ω is defined for equation (3b) is labeled $\omega_{j,k}$ on figure 1(b). However, the point where the source function s is defined is displaced from $\omega_{j,k}$ by half intervals. To avoid the use of indices containing fractions (half intervals), define the discrete coordinates as

$$x'_{j'} = x_\ell + \left(j' - \frac{1}{2}\right)h_x , \quad y'_{k'} = y_\ell + \left(k' - \frac{1}{2}\right)h_y \tag{4b}$$

where j' and k' are integers. In order that the indices on ω have the same values as those on s corresponding to the rotationality equation and the continuity equation, respectively, for the staggered mesh cells in figures 1, let

$$j' = j + \frac{1}{2} , \quad k' = k + \frac{1}{2} \tag{5}$$

The locations where u and v are defined in figure 1(a) make it convenient to then define

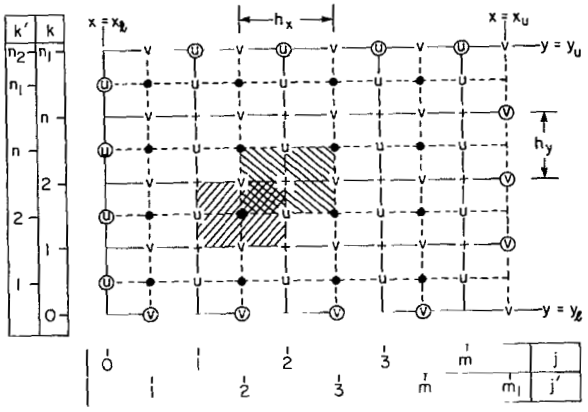
$$u_{j,k'} = u^+ , \quad v_{j',k} = v^+ \tag{6}$$

We thus define values of ω, s, u, and v at discrete points in figure 1(b) as

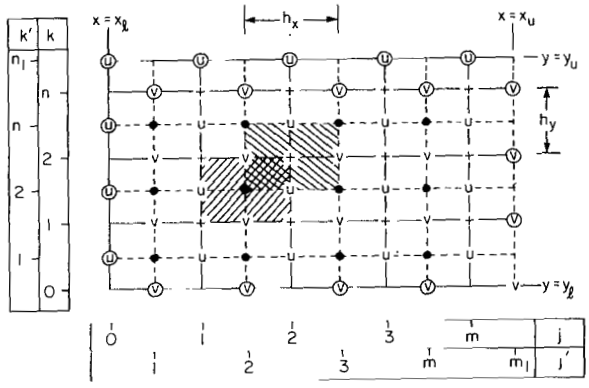
$$\left.\begin{aligned} \omega_{j,k} &= \omega(x_j, y_k) , & s_{j',k'} &= s(x'_{j'}, y'_{k'}) \\ u_{j,k'} &= u(x_j, y'_{k'}) , & v_{j,k} &= v(x'_{j'}, y_k) \end{aligned}\right\} \tag{7}$$

As used in equation (3a), the second-order-accurate, central finite-difference approximations needed for the continuity equation are then

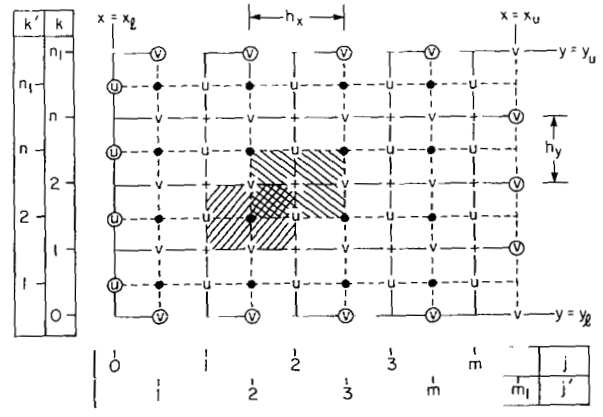
$$\left.\begin{aligned} (\partial u/\partial x)_{j',k'} &\approx (u_{j,k'} - u_{j-1,k'})/h_x \\ (\partial v/\partial y)_{j',k'} &\approx (v_{j',k} - v_{j',k-1})/h_y \end{aligned}\right\} \tag{8a}$$



(a) Version A.



(b) Versions B and C.



(c) Versions D and E.

Figure 2.- Staggered computational meshes.

and, as used in equation (3b), the second-order-accurate, central finite-difference approximations needed for the rotationality equation are

$$\left.\begin{aligned}(\partial u/\partial y)_{j,k} &\approx (u_{j,k'+1} - u_{j,k'})/h_y \\ (\partial v/\partial x)_{j,k} &\approx (v_{j'+1,k} - v_{j',k})/h_x\end{aligned}\right\} \tag{8b}$$

The motivation for using integer values of j, j', k, and k' in equations (4) and (7) is that, for purposes of *indexing* the variables u, v, s, and ω, there need not be any distinction between j and j' or between k and k'. Thus, for example, whenever $u_{j,k}$ is written, $u_{j,k'}$ is implied. The distinction is important only when determining actual locations on the mesh.

Figure 2 shows three different mesh configurations that correspond to the algorithm versions indicated. Figure 1(b) is imbedded in each mesh, as indicated by crosshatching (with $j,k = 2,2$). The three meshes differ only at the upper boundary; the reasons for the differences are explained in later sections. The circled symbols in figure 2(a,b,c) indicate quantities that are specified as boundary conditions. As in figure 1(a), the dots in figure 2 indicate points where the continuity equation (3a) is to be satisfied (values of $s_{j',k'}$ specified) and the crosses indicate points where the rotationality equation (3b) is to be satisfied (values of $\omega_{j,k}$ specified). An exception to this is that on the line $y = y_u$ in figure 2(a) for version A the "circled u" symbols are at points that should also be marked by crosses because the rotationality equation is to be satisfied there and values of ω, as well as u, are to be specified there.

For defining the mesh dimensions and the dimensions of relevant matrices, let m, m_1, n, n_1, and n_2 be integers so that:

m = the number of discrete x values where either the continuity or the rotationality equation is to be satisfied, that is, the number of dots or crosses in a horizontal row in figures 2(a), (b), and (c) (all versions of the solver);

has the value 4 for the special case illustrated in figure 2;

m_1 denoted as the "mesh number for the x direction":

= m + 1;

has the value 5 for the special case illustrated in figure 2;

n = number of discrete y values where the rotationality equation is to be satisfied in figures 2(b) and 2(c), that is, the number of crosses in a vertical column in figures 2(b) and 2(c) (versions B, C, D, and E of the solver);

= also the number of discrete y values where the continuity equation is to be satisfied in figure 2(b), that is, the number of dots in a vertical column in figure 2(b) (versions B and C of the solver);

has the value 3 for the special case illustrated in figure 2;

n_1 denoted as the "mesh number for the y direction";

$= n + 1$;

$= 2^L$, where L is a positive integer;

= number of discrete y values where the rotationality equation is to be satisfied in figure 2(a), that is, the number of crosses in a vertical column in figure 2(a) (version A);

= also the number of discrete y values where the continuity equation is to be satisfied in figures 2(a) and 2(c), that is, the number of dots in a vertical column in figures 2(a) and 2(c) (versions A, D, and E);

has the value 4 for the special case illustrated in figure 2;

$n_2 = n + 2$;

has the value 5 for the special case in figure 2(a).

To summarize the ranges of the discretized functions on the meshes in figure 2, table I is provided.

TABLE I.- RANGES OF DISCRETIZED FUNCTIONS

Function	Version of solver	Ranges of indices	Values of coordinate where function is defined	Discrete coordinate equation
$u_{j,k'}$	All	$j = 0$ to m	$x = x_j$	$x_j = x_\ell + jh_x$
	All	$k' = 1$ to n_1	$y = y_{k'}'$	$y_{k'}' = y_\ell + (k' - \frac{1}{2})h_y$
	A	$k' = n_2$	$y = y_u$	$y_u = y_\ell + n_1 h_y$
$v_{j',k}$	All	$j' = 1$ to m_1	$x = x_{j'}'$	$x_{j'}' = x_\ell + (j' - \frac{1}{2})h_x$
	A,D,E	$k = 0$ to n_1	$y = y_k$	$y_k = y_\ell + kh_y$
	B,C	$k = 0$ to n	$y = y_k$	$y_k = y_\ell + kh_y$
$s_{j',k'}$	All	$j' = 1$ to m	$x = x_{j'}'$	$x_{j'}' = x_\ell + (j' - \frac{1}{2})h_x$
	A,D,E	$k' = 1$ to n_1	$y = y_{k'}'$	$y_{k'}' = y_\ell + (k' - \frac{1}{2})h_y$
	B,C	$k' = 1$ to n	$y = y_{k'}'$	$y_{k'}' = y_\ell + (k' - \frac{1}{2})h_y$
$\omega_{j,k}$	All	$j = 1$ to m	$x = x_j$	$x_j = x_\ell + jh_x$
	A	$k = 1$ to n_1	$y = y_k$	$y_k = y_\ell + kh_y$
	B,C,D,E	$k = 1$ to n	$y = y_k$	$y_k = y_\ell + kh_y$

In terms of the above defined parameters, the mesh intervals h_x and h_y are

$$h_x = (x_u - x_\ell)/[m + (1/2)] \ , \quad \text{all versions} \tag{9a}$$

$$h_y = (y_u - y_\ell)/n_1 \ , \quad \text{versions A,D,E} \tag{9b}$$

$$h_y = (y_u - y_\ell)[n + (1/2)] \ , \quad \text{versions B,C} \tag{9c}$$

It will be convenient for the FORTRAN programs to also define "nominal" values of x_u, x_ℓ, and y_u, denoted as x_{un}, $x_{\ell n}$, y_{un}, by

$$h_x = (x_{un} - x_{\ell n})/m_1 \tag{9d}$$

$$h_y = (y_{un} - y_\ell)/n_1 \tag{9e}$$

The usefulness of these definitions is explained in the later section "FORTRAN Programs."

MATRIX DEFINITIONS AND RELATIONS

With the above definitions of mesh parameters and variables, we can now define some particular matrices and consider some matrix relations that will be of common use in the next four sections for developing the different versions of the Cauchy-Riemann solvers. As described above, we do not distinguish between j and j' or between k and k' for purposes of indexing the dependent variables and specified functions.

First define the column vectors of dimension m:

$$\mathbf{U}_k = \text{col}[u_{1,k}, u_{2,k}, \ldots, u_{m,k}] \tag{10a}$$

$$\mathbf{V}_k = \text{col}[v_{1,k}, v_{2,k}, \ldots, v_{m,k}] \tag{10b}$$

$$\mathbf{f}_k = \text{col}[f_{1,k}, f_{2,k}, \ldots, f_{m,k}] \tag{10c}$$

$$\mathbf{g}_k = \text{col}[g_{1,k}, g_{2,k}, \ldots, g_{m,k}] \tag{10d}$$

where $f_{j,k}$ and $g_{j,k}$ are to be defined later in terms of $s_{j,k}$ and $\omega_{j,k}$. The range of k in equations (10) is slightly different for different versions of the solver and is specified later.

With the vectors in equations (10) as elements, next define the following block column vectors of block dimension p or q:

$$\mathbf{U} = \text{col}[\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_q] \tag{11a}$$

$$\mathbf{V} = \text{col}[\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_p] \tag{11b}$$

$$\mathbf{f} = \text{col}[\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_q] \tag{11c}$$

$$\mathbf{g} = \text{col}[\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_p] \tag{11d}$$

In each application of these vectors, p and q will be specified in terms of n and n_1.

A general tridiagonal matrix of square dimension m is denoted by

$$\mathbf{T}_m(a_j,b_j,c_j) = \begin{bmatrix} b_1 & c_1 & & & \\ \ddots & \ddots & \ddots & & \\ & a_j & b_j & c_j & \\ & & \ddots & \ddots & \ddots \\ & & & a_m & b_m \end{bmatrix}_{m\times m} \tag{12}$$

Denote a general diagonal matrix of square dimension m by

$$\mathbf{D}_m(b_j) = \mathbf{T}_m(0,b_j,0) = \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_m \end{bmatrix}_{m\times m} \tag{13}$$

Special cases are the unit matrix and null matrix, both of dimension m:

$$\mathbf{I} = \mathbf{T}_m(0,1,0) \ , \quad \mathbf{O} = \mathbf{T}_m(0,0,0) \tag{14}$$

Equations (12) and (13) may also be used to define general block-tridiagonal and block-diagonal matrices of block dimension p, for example:

$$\mathbf{T}_p(\mathbf{A},\mathbf{B},\mathbf{C}) = \begin{bmatrix} \mathbf{B} & \mathbf{C} & & & \\ \mathbf{A} & \mathbf{B} & \mathbf{C} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{A} & \mathbf{B} & \mathbf{C} \\ & & & \mathbf{A} & \mathbf{B} \end{bmatrix}_{p\times p \text{ blocks}} \tag{15a}$$

and

$$\mathbf{D}_p(\mathbf{B}) = \mathbf{T}_p(\mathbf{O},\mathbf{B},\mathbf{O}) \tag{15b}$$

where the arguments **A**, **B**, and **C** are dummy $m\times m$ matrices and **O** is defined in equation (14). At this point, it is also convenient to define the following special partitioned square block matrices of block dimension p:

$$\mathbf{H}_p = \left[\begin{array}{ccc|c} \mathbf{O} & \cdots & \mathbf{O} & \mathbf{I} \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{I} \\ \vdots & & \vdots & \vdots \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{I} \end{array}\right]_{p\times p \text{ blocks}} \tag{16}$$

$$\mathbf{J}_p(\mathbf{B}) = \left[\begin{array}{ccc|c} \mathbf{O} & \cdots & \mathbf{O} & \mathbf{O} \\ \vdots & & \vdots & \vdots \\ \mathbf{O} & \cdots & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{O} & \cdots & \mathbf{O} & \mathbf{B} \end{array}\right]_{p\times p \text{ blocks}} \tag{17}$$

and in terms of equations (15b) and (17) then define

$$\mathbf{D}'_p(\mathbf{B}) = \mathbf{D}_p(\mathbf{B}) - \frac{1}{2}\,\mathbf{J}_p(\mathbf{B}) \tag{18}$$

which is a block-diagonal matrix with the last diagonal block equal to half the matrix occupying each of the other diagonal blocks.

A convenient matrix for following developments is denoted by $\mathbf{E}_{pq}(\mathbf{A})$ and is defined as follows. Consider a general matrix $\mathbf{M}_{pq}$ of dimension $p \times q$:

$$\mathbf{M}_{pq} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1q} \\ m_{21} & m_{22} & \cdots & & \vdots \\ \vdots & \vdots & \ddots & & \\ m_{p1} & m_{p2} & \cdots & & m_{pq} \end{bmatrix} \tag{19}$$

Let $\ell = \min\{p,q\}$. Then the *diagonal* is defined to consist of the elements m_{ii}, $i = 1, 2, \ldots, \ell$ (whether or not $p = q$; see, e.g., ref. 42); the *superdiagonal* consists of the elements $m_{i,i+1}$ and the *subdiagonal* consists of the elements $m_{i,i-1}$. Now to define $\mathbf{E}_{pq}(\mathbf{A})$, let each element on its diagonal be a matrix $-\mathbf{A}$ of dimension m and let each element on its superdiagonal be the matrix $\mathbf{A}$. Let all other elements be m-dimensional null matrices. Thus $\mathbf{E}_{pq}(\mathbf{A})$ has p rows and q columns of m-dimensional blocks. Two special cases of interest are for $q = p + 1$:

$$\mathbf{E}_{p,p+1}(\mathbf{A}) = \begin{bmatrix} -\mathbf{A} & \mathbf{A} & & & \\ \mathbf{O} & -\mathbf{A} & \mathbf{A} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{O} & -\mathbf{A} & \mathbf{A} \end{bmatrix}_{p\times(p+1)\text{ blocks}} \tag{20a}$$

and for $q = p$:

$$\mathbf{E}_{p}(\mathbf{A}) \equiv \mathbf{E}_{pp}(\mathbf{A}) = \begin{bmatrix} -\mathbf{A} & \mathbf{A} & & \\ \mathbf{O} & -\mathbf{A} & \ddots & \\ & \ddots & \ddots & \mathbf{A} \\ & & \mathbf{O} & -\mathbf{A} \end{bmatrix}_{p\times p\text{ blocks}} \tag{20b}$$

It is convenient to further define some special matrices that will be of use in the following four sections. From here on, whenever one of these matrix symbols is used, it has the specific definition given in this section. For these definitions, let β, $\beta_{1,j}$, and $\beta_{2,j}$ be quantities that are to be specified later. Then, in terms of the above-defined notation, define the particular m-dimensional square matrices:

$$\mathbf{A} \equiv \mathbf{T}_m(-\beta_{2,j},\beta_{1,j},0) \tag{21a}$$

$$\mathbf{B} \equiv \mathbf{T}_m(0,\beta,-\beta) \tag{21b}$$

and

$$\mathbf{C} \equiv 2\mathbf{I} + \mathbf{AB}$$

$$= \mathbf{T}_m(-\alpha_j,\beta_j,-\gamma_j) \tag{21c}$$

where

$$\left.\begin{aligned} \alpha_j &= \beta\beta_{2,j} \\ \beta_j &= 2 + \beta(\beta_{2,j} + \beta_{1,j}) \\ \gamma_j &= \beta\beta_{1,j} \end{aligned}\right\} \quad (j = 1 \text{ to } m) \tag{21d}$$

with $\beta_{2,1} \equiv 0$; also define the particular block matrices of block dimension p, or q, or $p \times q$, as indicated:

$$\mathbf{D}_A \equiv \mathbf{D}_q(\mathbf{A}) \tag{22a}$$

$$\mathbf{D}_B \equiv \mathbf{D}_p(\mathbf{B}) \tag{22b}$$

$$\mathbf{D}'_B \equiv \mathbf{D}'_p(\mathbf{B}) \tag{22c}$$

$$\mathbf{E} \equiv \mathbf{E}_{pq}(\mathbf{I}) \tag{22d}$$

$$\mathbf{E}^T \equiv \text{Block transpose of } \mathbf{E} \tag{22e}$$

where $\mathbf{A}$ and $\mathbf{B}$ are defined in equations (21) and p and q are to be specified in each application.

The following relationships are then easily found and will be of particular use, for either $q = p$ or $q = p + 1$:

$$\mathbf{E}\mathbf{D}_A = \mathbf{E}_{pq}(\mathbf{A}) \tag{23a}$$

$$[\mathbf{D}_p(\mathbf{A})]\mathbf{E} = \mathbf{E}_{pq}(\mathbf{A}) \tag{23b}$$

$$[\mathbf{D}_p(\mathbf{A})][\mathbf{D}_p(\mathbf{B})] = \mathbf{D}_p(\mathbf{AB}) \tag{23c}$$

$$[\mathbf{D}_p(\mathbf{A})][\mathbf{D}'_p(\mathbf{B})] = \mathbf{D}'_p(\mathbf{AB}) \tag{23d}$$

For $q = p$:

$$\mathbf{E}\mathbf{E}^T = \mathbf{T}_p(-\mathbf{I}, 2\mathbf{I}, -\mathbf{I}) - \mathbf{J}_p(\mathbf{I}) \tag{23e}$$

but, for $q = p + 1$:

$$\mathbf{E}\mathbf{E}^T = \mathbf{T}_p(-\mathbf{I}, 2\mathbf{I}, -\mathbf{I}) \tag{23f}$$

As a consequence of equations (23a) and (23b), we find for either $q = p$ or $q = p + 1$:

$$\mathbf{E}\mathbf{D}_A = [\mathbf{D}_p(\mathbf{A})]\mathbf{E} \tag{24a}$$

and from equations (23c) to (23f), along with (21c), that for $q = p$:

$$\mathbf{E}\mathbf{E}^T + [\mathbf{D}_p(\mathbf{A})]\mathbf{D}_B = \mathbf{T}_p(-\mathbf{I},2\mathbf{I},-\mathbf{I}) - \mathbf{J}_p(\mathbf{I}) + \mathbf{D}_p(\mathbf{AB})$$

$$= \mathbf{T}_p(-\mathbf{I},\mathbf{C},-\mathbf{I}) - \mathbf{J}_p(\mathbf{I}) \tag{24b}$$

$$\mathbf{E}\mathbf{E}^T + [\mathbf{D}_p(\mathbf{A})]\mathbf{D}'_B = \mathbf{T}_p(-\mathbf{I},2\mathbf{I},-\mathbf{I}) - \mathbf{J}_p(\mathbf{I}) + \mathbf{D}'_p(\mathbf{AB})$$

$$= \mathbf{T}_p(-\mathbf{I},\mathbf{O},-\mathbf{I}) + \mathbf{D}'_p(\mathbf{C}) \tag{24c}$$

and for $q = p + 1$:

$$\mathbf{E}\mathbf{E}^T + [\mathbf{D}_p(\mathbf{A})]\mathbf{D}_B = \mathbf{T}_p(-\mathbf{I},\mathbf{C},-\mathbf{I}) \tag{24d}$$

SIMPLEST FORMULATION OF MATRIX EQUATIONS, AND MOTIVATION FOR ALTERNATIVE VERSIONS

Consider as the simplest and most straightforward formulation of equations (3) the case where the mesh is as shown in figure 2(b) *except* that the circles are removed from the row of circled v symbols in the mesh row $k = n$. That is, values of u are specified at $k' = n_1$ ($y = y_u$), but v is not specified at $k = n$.

Define

$$\beta \equiv h_y/h_x \tag{25}$$

and assume for the formulation in this section that a^+ and a^- in equations (3) are zero. Then, with the definitions in equations (4) to (8), equations (3) become

$$\beta(u_{j,k} - u_{j-1,k}) + v_{j,k} - v_{j,k-1} = h_y s_{j,k} \tag{26a}$$

$$-u_{j,k} + u_{j,k+1} + \beta(v_{j,k} - v_{j+1,k}) = -h_y \omega_{j,k} \tag{26b}$$

for all $j = 1$ to m and $k = 1$ to n. (The value of m is arbitrary, but $n_1 = n + 1 = 2^L$ where L is an integer. This is important for the discussion of the most efficient cyclic reduction of the equations that result later.) Equations (26) are second-order accurate. With all specified boundary values of $u_{j,k}$ and $v_{j,k}$ (as indicated in fig. 2(b), except as noted above) moved to

the right sides of equations (26), denote the resulting right side of equation (26a) by $f_{j,k}$ and the resulting right side of equation (26b) by $g_{j,k}$. Then with the vector and matrix definitions in the above section and with

$$\beta_{1,j} = \beta_{2,j} = \beta \qquad (\text{except} \quad \beta_{2,1} \equiv 0) \tag{27}$$

and

$$p = q = n \tag{28}$$

the complete set of finite-difference equations (26) for all j,k can be written as

$$\begin{array}{c} \overbrace{\qquad\qquad}^{n \text{ blocks}} \quad \overbrace{\qquad\qquad}^{n \text{ blocks}} \\ \left[\begin{array}{cccc|cccc} \mathbf{A} & & & & \mathbf{I} & \mathbf{O} & & \\ & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} & \ddots & \\ & & \ddots & & & \ddots & \ddots & \mathbf{O} \\ & & & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} \\ \hline -\mathbf{I} & \mathbf{I} & & & \mathbf{B} & & & \\ \mathbf{O} & -\mathbf{I} & \ddots & & & \mathbf{B} & & \\ & \ddots & \ddots & \mathbf{I} & & & \ddots & \\ & & \mathbf{O} & -\mathbf{I} & & & & \mathbf{B} \end{array}\right] \left[\begin{array}{c} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_n \\ \hline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_n \\ \hline \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_n \end{array}\right] \end{array} \tag{29}$$

or, more concisely,

$$\mathbf{D}_A\mathbf{U} - \mathbf{E}^T\mathbf{V} = \mathbf{f} \tag{30a}$$

$$\mathbf{E}\mathbf{U} + \mathbf{D}_B\mathbf{V} = \mathbf{g} \tag{30b}$$

Thus, equation (30a) represents the entire set of continuity equations (26a), and equation (30b) represents the entire set of rotationality equations (26b).

Equations (30) can be reduced by eliminating $\mathbf{U}$. Premultiply equation (30a) by $-\mathbf{E}$, premultiply equation (30b) by $\mathbf{D}_n(\mathbf{A})$, add the two resulting matrix equations, and use the relations (24a) and (24b) to obtain

$$[\mathbf{T}_n(-\mathbf{I},\mathbf{C},-\mathbf{I}) - \mathbf{J}_n(\mathbf{I})]\mathbf{V} = \mathbf{F} \tag{31a}$$

where

$$\mathbf{F} \equiv \mathrm{col}[\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_p] \tag{32a}$$

$$= [\mathbf{D}_p(\mathbf{A})]\mathbf{g} - \mathbf{Ef} \tag{32b}$$

or

$$\mathbf{F}_k = \mathbf{A}\mathbf{g}_k + \mathbf{f}_k - \mathbf{f}_{k+1}, \quad k = 1 \text{ to } p \tag{32c}$$

where, in this case, $p = n$, and $\mathbf{f}_{p+1}$ must be defined to be zero when $p = q$ for equation (32c) to apply; and where the j component of the vector represented by each product $\mathbf{A}\mathbf{g}_k$ is just $\beta_{1,j}g_{j,k} - \beta_{2,j}g_{j-1,k}$ ($j = 1$ to m, $k = 1$ to p) in which $\beta_{2,1} \equiv 0$ as in equations (27).

Equation (31a) is a block-tridiagonal equation to solve for $\mathbf{V}$, with the right side, $\mathbf{F}$, easily obtained from the simple operations indicated in equation (32c). In an expanded form, the block-tridiagonal equation (31a) is

$$\begin{bmatrix} \mathbf{C} & -\mathbf{I} & & \\ -\mathbf{I} & \ddots & \ddots & \\ & \ddots & \mathbf{C} & -\mathbf{I} \\ & & -\mathbf{I} & \mathbf{C}-\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_n \end{bmatrix} \tag{31b}$$

Equation (31b) could be solved (ref. 25, p. 68) by a fast, direct elliptic solver for each $v_{j,k}$, and then the bottom half of equation (29) could be used to solve for each $u_{j,k}$. However, the direct solution of equation (31b) is significantly more difficult than it would be if the entry in the nth column of the nth row of the block coefficient matrix were $\mathbf{C}$, like all the other diagonal entries, rather than $\mathbf{C} - \mathbf{I}$.

There are two approaches that one can now take. Equation (31b) is in a form equivalent to that obtained for a Poisson equation with a Neumann condition specified in a certain way at one boundary (which would result in the last diagonal entry, $\mathbf{C} - \mathbf{I}$). Therefore, one possible approach is that used by Sweet (ref. 9) and by Schumann and Sweet (ref. 17) in solving Poisson's equation: to deal directly with the irregular element of the block coefficient matrix by developing a separate and more complicated factorization for calculations during the reduction process involving that element. A different point of view avoids the additional complication both in the algorithm development and in the programming. This approach, which had been taken in reference 25 and also is taken here, is to look for simple ways of modifying the problem formulation so as to obtain a very regular coefficient matrix, with all block-diagonal elements the same, so that simpler algorithms and programs can be used. The algorithm versions described in the next three sections deal with the alternative formulations.

VERSION A — ORIGINAL FAST DIRECT CAUCHY-RIEMANN SOLVER

Version A described here is the original version of the Cauchy-Riemann solver developed in reference 25 and used for fast iterative calculations in a nonlinear problem in reference 35. The derivation is given here, not only for completeness in the sequence of development both of algorithms and of programs, but also because the matrix derivation here is simpler than that given in reference 25. We also wish to present slightly more detail in the algorithm and to give the corresponding FORTRAN program for an example problem.

As explained in reference 25, a modified treatment of the upper boundary results in a regular block coefficient matrix with all the block-diagonal elements the same (rather than being different as in eq. (31b)). The modified treatment changes the last column of the matrix from the form obtained in equation (31b) (equivalent to a Neumann condition applied to Poisson's equation in a certain way, as done in refs. 9 and 17) to another form that is equivalent to a Neumann condition applied to Poisson's equation in a different way (as done in ref. 5), which is simpler to treat by cyclic reduction. The modification of the upper boundary to accomplish this was originally derived by working backward from the desired result. Therefore, the motivation is not clear at the beginning of the derivation but becomes clear when a crucial step in the algorithm development is reached.

Consider the mesh shown in figure 2(a), on which the value of m is arbitrary and on which the number of y values both for the continuity equation and for the rotationality equation is $n_1 = 2^L$, where the exponent L is an integer. Note that this is one greater than the dimension n used in the previous section where figure 2(b) was used for the illustration. The rotationality equation is to be applied at the points where u is specified on the upper boundary, as well as all the points indicated by crosses. With β defined by equation (25), the assumption of zero values for α^+ and α^-, and with the definitions in equations (4) to (8), equations (3a) and (3b) become

For j = 1 to m and k = 1 to n_1:

$$\beta(u_{j,k} - u_{j-1,k}) + v_{j,k} - v_{j,k-1} = h_y s_{j,k} \tag{33a}$$

For j = 1 to m and k = 1 to n:

$$-u_{j,k} + u_{j,k+1} + \beta(v_{j,k} - v_{j+1,k}) = -h_y \omega_{j,k} \tag{33b}$$

However, for the rotationality equation at the upper boundary only ($k = n_1$), the difference expression for $(\partial u/\partial y)_{j,k}$ given in equation (8b) is replaced by the first-order-accurate expression:

$$(\partial u/\partial y)_{j,k} \approx (u_{j,k'+1} - u_{j,k'})/\left(\tfrac{1}{2} h_y\right) \tag{34}$$

where, as indicated in figure 2(a), $u_{j,k'+1}$ is evaluated on $y = y_u$. Thus, as before, ignoring the distinction between j and j' or k and k' for indexing, we write

For $j = 1$ to m and $k = n_1$:

$$-u_{j,k} + u_{j,k+1} + \frac{1}{2}\beta(v_{j,k} - v_{j+1,k}) = -\frac{1}{2}h_y\omega_{j,k} \tag{33c}$$

Although equations (33a) and (33b) are second-order-accurate representations of equations (1) at all points, and equation (33c) is only first-order-accurate at the upper boundary, the applications for which the algorithm is intended are not significantly affected by the lower accuracy at the upper boundary, which is assumed to be sufficiently far from regions where any large gradients of u and v occur. The factor $1/2$ in the denominator in equation (34), with the resulting factor of $1/2$ multiplying β in equation (33c), is crucial for obtaining the desired result. With all specified boundary values of $u_{j,k}$ and $v_{j,k}$, as indicated in figure 2(a), moved to the right sides of equations (33a), (33b), and (33c), denote the resulting right side of (33a) by $f_{j,k}$ ($j = 1$ to m, $k = 1$ to n_1) and the resulting right sides of equations (33b) and (33c) by $g_{j,k}$ ($j = 1$ to m, $k = 1$ to n_1). Then with the definitions in the section Matrix Definitions and Relations, and with

$$\beta_{1,j} = \beta_{2,j} = \beta \quad (\text{except } \beta_{2,1} = 0) \tag{35}$$

and

$$p = q = n_1 \tag{36}$$

the complete set of finite-difference equations (33) for all j,k can be written as

$$\begin{array}{c} \overbrace{\qquad\qquad}^{n_1 \text{ blocks}} \quad \overbrace{\qquad\qquad}^{n_1 \text{ blocks}} \\ \left[\begin{array}{cccc:cccc} \mathbf{A} & & & & \mathbf{I} & \mathbf{O} & & \\ & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} & \ddots & \\ & & \ddots & & & \ddots & \ddots & \mathbf{O} \\ & & & \mathbf{A} & & & -\mathbf{I} & \mathbf{I} \\ \hdashline -\mathbf{I} & \mathbf{I} & & & \mathbf{B} & & & \\ \mathbf{O} & -\mathbf{I} & \ddots & & & \ddots & & \\ & \ddots & \ddots & \mathbf{I} & & & \mathbf{B} & \\ & & \mathbf{O} & -\mathbf{I} & & & & \tfrac{1}{2}\mathbf{B} \end{array}\right] \left[\begin{array}{c} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{n_1} \\ \hdashline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_{n_1} \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{n_1} \\ \hdashline \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{n_1} \end{array}\right] \end{array} \tag{37}$$

or, more concisely,

$$\mathbf{D}_A\mathbf{U} - \mathbf{E}^T\mathbf{V} = \mathbf{f} \tag{38a}$$

$$\mathbf{E}\mathbf{U} + \mathbf{D}_B'\mathbf{V} = \mathbf{g} \tag{38b}$$

Equations (38) can be reduced by eliminating $\mathbf{U}$. Premultiply equation (38a) by $-\mathbf{E}$, premultiply equation (38b) by $\mathbf{D}_{n_1}(\mathbf{A})$, add the two resulting matrix equations, and use the relations (24a) and (24c) to obtain

$$\left[\mathbf{T}_{n_1}(-\mathbf{I},\mathbf{O},-\mathbf{I}) + \mathbf{D}'_{n_1}(\mathbf{C})\right]\mathbf{V} = \mathbf{F} \tag{39a}$$

where $\mathbf{F}$ is defined as in equations (32); but, in this case, $p = n_1$. Again, since $p = q$, $\mathbf{f}_{p+1}$ must be defined to be zero for equation (32c) to be valid.

Equation (39a) is a block-tridiagonal equation for $\mathbf{V}$. If we write it in an expanded form:

$$\begin{bmatrix} \mathbf{C} & -\mathbf{I} & & \\ -\mathbf{I} & \ddots & \ddots & \\ & \ddots & \mathbf{C} & -\mathbf{I} \\ & & -\mathbf{I} & \tfrac{1}{2}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_{n_1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_{n_1} \end{bmatrix} \tag{39b}$$

we can now see the motivation for using equation (33c) containing the factor 1/2 at the upper boundary. By multiplying the last column of m-dimensional matrices in the block coefficient matrix in equation (39b) by 2 and at the same time multiplying the last vector element, $\mathbf{V}_{n_1}$, by 1/2, we obtain

$$\begin{bmatrix} \mathbf{C} & -\mathbf{I} & & & \\ -\mathbf{I} & \mathbf{C} & \ddots & & \\ & -\mathbf{I} & \ddots & -\mathbf{I} & \\ & & \ddots & \mathbf{C} & -2\mathbf{I} \\ & & & -\mathbf{I} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \\ \tfrac{1}{2}\mathbf{V}_{n_1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_n \\ \mathbf{F}_{n_1} \end{bmatrix} \tag{40}$$

The block-tridiagonal coefficient matrix now has a form for which the cyclic reduction is relatively simple for $n_1 = 2^L$, L being an integer. The cyclic reduction of equation (40) to obtain all values of $v_{j,k}$ is adequately described in reference 25 (pp. 68-72). The values of $u_{j,k}$ are then obtained directly from the lower half of equation (37) (i.e., the rotationality equation for each j,k), from which

$$\mathbf{U}_{n_1} = \frac{1}{2}\,\mathbf{BV}_{n_1} - \mathbf{g}_{n_1} \tag{41a}$$

and

$$\mathbf{U}_k = \mathbf{U}_{k+1} + \mathbf{BV}_k - \mathbf{g}_k\,, \qquad (k = n \text{ to } 1) \tag{41b}$$

where (see eq. (21b)) the j component of the vector represented by each product $\mathbf{BV}_k$ is just $\beta(v_{j,k} - v_{j+1,k})$, where $v_{m_1,k}$ is set equal to zero for all $k = 1$ to n_1 (since it is already included in $g_{j,k}$).

A FORTRAN program to solve an illustrative example problem, using subroutines corresponding to version A, is described in a later section.

VERSIONS B AND C — FIRST REVISED AND EXTENDED CAUCHY-RIEMANN SOLVER

The motivation for developing version B was the prospect of increased efficiency over version A. Because version A (as described above) contains a block-tridiagonal matrix equation to solve that is equivalent to a discrete Poisson equation with a Neumann condition, the cyclic reduction is less efficient than would be the cyclic reduction of the corresponding equivalent matrix equation derivable from Poisson's equation with all Dirichlet conditions. The latter is known to require one less cycle of reduction containing a significant number of operations, in both the forward and backward recursions. Therefore, the objective was to change the condition application at the upper boundary so that the block-tridiagonal matrix obtained would have all diagonal blocks the same, as in equation (40), but also have all super- and subdiagonal blocks the same, as in equation (31b). The revised method for treating the upper boundary could also alleviate effects of the decreased accuracy at the upper boundary in version A in problems where those effects could be significant.

At the time the revised version B was being developed to increase the efficiency, it was also found, in the studies that were to be reported later in references 36 and 37, that certain terms needed to be added to the solver to stabilize the iterations in the fast semidirect iterative method for the nonlinear, mixed elliptic-hyperbolic problem of transonic flow. Therefore, the algorithm for version B was extended to include the terms $a^+u^+ + a^-u^-$ in equations (3), with constant values of a^+ and a^-. This extended algorithm, version C, is the one used for all the calculations made in references 36 to 38. Since version B can be regarded as a special case of version C, the derivations are combined. However, the subroutines for version B are slightly more efficient than those for C and can be used for the same problems as version A, where the extra terms are not required.

Consider figure 2(b), on which m is arbitrary and $n_1 = n + 1 = 2^L$ with L an integer for the simplest and most efficient cyclic reduction of resulting equations. Note that on the upper boundary, at $k' = n_1$ ($y = y_u$), values of u (circled symbols) are specified. It is also indicated that,

at a half interval below the upper boundary, at $k = n$, values of v are also to be specified. (Such conditions in most physical problems are not difficult to obtain. For example, in refs. 36 to 38, analytical asymptotic conditions valid sufficiently far from $y = y_\ell$ were specified for u at $k' = n_1$ and for v at $k = n$. In the event that values of u and v would be known only at a single boundary line, $y = y_u$, the discretized continuity equation could be used to obtain corresponding values of v on $k = n$ to specify as input.) As was the case in the previous section for version A, the motivation for this particular modification of the upper-boundary treatment is not obvious at the beginning since the necessary modification was originally obtained by working backward from a desired result. After a certain point is reached in the derivation, the reason for specifying v at $k = n$ will become clear.

The specification of v on $k = n$, the crucial feature of this algorithm that leads to the desired result, allows us to add an unknown, $v_{j,n}$, to the left side of each continuity equation and to add the corresponding *specified* quantity, denoted by $(v_T)_j$, to the right side. Thus, even though we specify $(v_T)_j$, we also shall determine $v_{j,n}$ as part of the solution algorithm. Also for use in equations (3) with nonzero values of a^+ and a^-, let

$$\bar{\alpha}_1 \equiv -h_x a^+ , \quad \bar{\alpha}_2 \equiv -h_x a^- \tag{42}$$

and in terms of these parameters, along with β defined by equation (25), let

$$\left.\begin{aligned} \beta_{1,j} &= \beta(1 - \bar{\alpha}_1) \\ \beta_{2,j} &= \beta(1 + \bar{\alpha}_2) \quad \text{(except } \beta_{2,1} \equiv 0\text{)} \end{aligned}\right\} \tag{43}$$

Then corresponding to figure 2(b), with equation (3a') applied at all the dots and equation (3b) applied at all the crosses, with the definitions in equations (4) to (8), we write for all $j = 1$ to m and all $k = 1$ to n:

$$\beta(1 - \bar{\alpha}_1)u_{j,k} - \beta(1 + \bar{\alpha}_2)u_{j-1,k} + v_{j,k} - v_{j,k-1} + v_{j,n} = h_y s'_{j,k} + (v_T)_j \tag{44a}$$

$$-u_{j,k} + u_{j,k+1} + \beta(v_{j,k} - v_{j+1,k}) = -h_y \omega_{j,k} \tag{44b}$$

With an appropriate expression for $s'_{j,k}$, as in equation (3a), these equations are second-order-accurate representations of equations (1).

Now, with all specified boundary values of $u_{j,k}$ and $v_{j,k}$ as indicated in figure 2(b) (except $v_{j,n}$) moved to the right sides of equations (44), denote the resulting right sides, respectively, by $f_{j,k}$ and $g_{j,k}$ ($j = 1$ to m, $k = 1$ to n). Then with the definitions in the section Matrix Definitions and Relations, and with $\beta_{1,j}$ and $\beta_{2,j}$ defined by equations (43) and

$$p = q = n \tag{45}$$

the complete set of finite-difference equations (44) for all j,k can be written as

$$\begin{bmatrix} \mathbf{A} & & & & & \mathbf{I} & \mathbf{O} & & & \mathbf{I} \\ & \mathbf{A} & & & & -\mathbf{I} & \mathbf{I} & \ddots & & \mathbf{I} \\ & & \ddots & & & & \ddots & \ddots & \mathbf{O} & \vdots \\ & & & \mathbf{A} & & & & -\mathbf{I} & \mathbf{I} & \mathbf{I} \\ & & & & \mathbf{A} & & & & -\mathbf{I} & 2\mathbf{I} \\ \hline -\mathbf{I} & \mathbf{I} & & & & \mathbf{B} & & & & \\ \mathbf{O} & -\mathbf{I} & \mathbf{I} & & & & \mathbf{B} & & & \\ & \ddots & \ddots & \ddots & & & & \ddots & & \\ & & \mathbf{O} & -\mathbf{I} & \mathbf{I} & & & & \mathbf{B} & \\ & & & \mathbf{O} & -\mathbf{I} & & & & & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{n-1} \\ \mathbf{U}_n \\ \hline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_{n-1} \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_{n-1} \\ \mathbf{f}_n \\ \hline \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{n-1} \\ \mathbf{g}_n \end{bmatrix} \tag{46}$$

(n blocks; n blocks)

With the additional use in this section of the matrix $\mathbf{H}_n$ defined by equation (16), the more concise formulation of equation (46) is

$$\mathbf{D}_A\mathbf{U} + (-\mathbf{E}^T + \mathbf{H}_n)\mathbf{V} = \mathbf{f} \tag{47a}$$

$$\mathbf{E}\mathbf{U} + \mathbf{D}_B\mathbf{V} = \mathbf{g} \tag{47b}$$

Equations (47) can be reduced by eliminating $\mathbf{U}$. Premultiply equation (47a) by $-\mathbf{E}$, premultiply equation (47b) by $\mathbf{D}_n(\mathbf{A})$, add the two resulting equations, and use the relations (24a) and (24b) to obtain

$$[\mathbf{T}_n(-\mathbf{I},\mathbf{C},-\mathbf{I}) - \mathbf{J}_n(\mathbf{I}) - \mathbf{E}\mathbf{H}_n]\mathbf{V} = \mathbf{F} \tag{48}$$

where $\mathbf{F}$ is defined as in equations (32) with $p = n$, and where, since $q = p$, $\mathbf{f}_{p+1}$ in equation (32c) must be defined to be zero. At this point, the motivation for introducing $v_{j,n}$ into equation (44a), which resulted in the term with the matrix $\mathbf{H}_n$ in equation (47a), becomes evident. The identity

$$[\mathbf{E}_n(\mathbf{I})]\mathbf{H}_n \equiv -\mathbf{J}_n(\mathbf{I}) \tag{49}$$

yields, from equation (48),

$$[\mathbf{T}_n(-\mathbf{I},\mathbf{C},-\mathbf{I})]\mathbf{V} = \mathbf{F} \tag{50a}$$

which is the desired simple and very regular form of the block-tridiagonal equation for $\mathbf{V}$:

$$\begin{bmatrix} \mathbf{C} & -\mathbf{I} & & \\ -\mathbf{I} & \mathbf{C} & \ddots & \\ & \ddots & \ddots & -\mathbf{I} \\ & & -\mathbf{I} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_n \end{bmatrix} \tag{50b}$$

Version C uses cyclic reduction of equation (50b), with the matrix $\mathbf{C}$ defined by equations (21c) and (21d), where $\beta_{1,j}$ and $\beta_{2,j}$ are defined by equations (43). Version B is for the special case where $\bar{\alpha}_1$ and $\bar{\alpha}_2$ in equations (43) are zero. The cyclic reduction process for obtaining all values of $v_{j,k}$ from equation (50b) is slightly different from that used in version A and is described in appendix F. The values of $u_{j,k}$ are then obtained directly from the lower half of equation (46), from which

$$\mathbf{U}_k = \mathbf{U}_{k+1} + \mathbf{B}\mathbf{V}_k - \mathbf{g}_k \qquad (k = n,\ n-1,\ \ldots,\ 1) \tag{51}$$

where $\mathbf{U}_{n+1}$ is set equal to zero, and where the j component of the vector represented by each product $\mathbf{B}\mathbf{V}_k$ is just $\beta(v_{j,k} - v_{j+1,k})$ with $v_{m_1,k}$ set equal to zero for all $k = 1$ to n (since it is already included in $g_{j,k}$).

FORTRAN programs to solve the illustrative example problem, using subroutines corresponding to both versions B and C, are described in a later section.

VERSIONS D AND E — SECOND REVISED AND EXTENDED CAUCHY-RIEMANN SOLVER

For current and future applications of a fast direct Cauchy-Riemann solver, it is desired to apply conditions on v on the horizontal centerline of the mesh (by methods beyond the scope of this report). It is therefore desired to have the upper and lower boundaries symmetrically located above and below the centerline and to have the same type of boundary condition applied at both boundaries. We therefore consider the mesh configuration shown in figure 2(c), on which boundary values of v are specified on the upper and lower boundaries, u is specified on the left, and v on the right.

There is a further motivation for pursuing the formulation for the configuration in figure 2(c). One quickly observes that each column of dots

contains one more point for the continuity equation to be solved than the number of crosses in a column of the points for the rotationality equation. Correspondingly, one observes that the number, q, of unknown values of u in a vertical column is also one more than the number, p, of unknown values of v. Thus, in equations (11),

$$q = p + 1 \tag{52}$$

If one then recalls the formulations in previous sections (e.g., eq. (29)), he notes that the quarter partitions of the block coefficient matrix are square because $q = p$ there. Consider that the number of rows of $m \times m$ matrices in the upper partition of equation (29) is q (the number of $\mathbf{U}_k$ vectors), whereas the number of rows in the lower partition is p (the number of $\mathbf{V}_k$ vectors); the number of columns in the left partition of the coefficient matrix must be q, whereas the number of columns in the right partition must be p in order for the indicated matrix multiplication to be defined. Thus the upper left partition has block dimension $q \times q$, the upper right $q \times p$, the lower left $p \times q$, and the lower right $p \times p$. Recalling also the identity (24d), we have an indication that the simplest and most regular coefficient matrix after reduction may be obtained quite naturally for $q = p + 1$, whereas it was not obtained naturally (i.e., without the tricks used for versions A, B, C), for example, in the section Simplest Formulation of Matrix Equations, in which it was assumed that the number of rotationality equations was the same as the number of continuity equations ($p = q$), and all block matrices other than vectors were square. Thus, the simplicity of the matrix in the identity (24d) (with the implied corresponding high efficiency of the algorithm that would result) encourages the pursuit of the formulation corresponding to figure 2(c) with nonsquare matrices.

Versions D and E of the extended Cauchy-Riemann solver are derived here as a single algorithm for the mesh in figure 2(c). Version D is the special case with constant coefficients, a^+ and a^-, in the extra terms $(a^+u^+ + a^-u^-)$ in equation (3a), whereas version E allows coefficients that are arbitrarily variable in x.

Figure 2(c) requires

$$p = n \ , \quad q = n_1 = n + 1 \tag{53}$$

This requirement, with $n_1 = 2^L$ (L, an integer), will result in a matrix equation for which the cyclic reduction is simplest and most efficient. The value of m, as in the other versions, is arbitrary.

For use in equations (3), let a^+ and a^- be defined, respectively, at the same points as u^+ and u^- in figure 1(a). To use the indexing notation of figure 1(b), then, let

$$\alpha_{1,j} \equiv -h_x a^+ \ , \quad \alpha_{2,j-1} \equiv -h_x a^- \tag{54}$$

and in terms of these parameters, along with β defined by equation (25), let

$$\left.\begin{aligned} \beta_{1,j} &= \beta(1 - \alpha_{1,j}) \\ \beta_{2,j} &= \beta(1 + \alpha_{2,j-1}) \quad \text{(except } \beta_{2,1} \equiv 0\text{)} \end{aligned}\right\} \tag{55}$$

Then corresponding to figure 2(c), with equation (3a') applied at all dots and equation (3b) applied at all crosses, with the definitions in equations (4) to (8), we write

For all $j = 1$ to m and all $k = 1$ to n_1:

$$\beta(1 - \alpha_{1,j})u_{j,k} - \beta(1 + \alpha_{2,j-1})u_{j-1,k} + v_{j,k} - v_{j,k-1} = h_y s'_{j,k} \tag{56a}$$

For all $j = 1$ to m and all $k = 1$ to n:

$$-u_{j,k} + u_{j,k+1} + \beta(v_{j,k} - v_{j+1,k}) = -h_y \omega_{j,k} \tag{56b}$$

With an appropriate representation of $s'_{j,k}$ as in equation (3a), these equations are second-order-accurate representations of equations (1).

With the specified boundary values of $u_{j,k}$ and $v_{j,k}$ indicated in figure 2(c) moved to the right sides of equations (56), denote the resulting right sides, respectively, by $f_{j,k}$ ($j = 1$ to m, $k = 1$ to n_1) and $g_{j,k}$ ($j = 1$ to m and $k = 1$ to n). Then, with the definitions in the section Matrix Definitions and Relations, and with $\beta_{1,j}$ and $\beta_{2,j}$ defined by equation (55), the complete set of finite-difference equations (56) for all values of j,k indicated can be written as

$$\left[\begin{array}{c:c} \overbrace{\begin{array}{ccccc} \mathbf{A} & & & & \\ & \mathbf{A} & & & \\ & & \ddots & & \\ & & & \mathbf{A} & \\ & & & & \mathbf{A} \end{array}}^{n_1 \text{ blocks}} & \overbrace{\begin{array}{ccccc} \mathbf{I} & \mathbf{O} & & & \\ -\mathbf{I} & \mathbf{I} & \ddots & & \\ & -\mathbf{I} & \ddots & \mathbf{O} & \\ & & \ddots & \mathbf{I} & \\ & & & -\mathbf{I} & \end{array}}^{n \text{ blocks}} \\ \hdashline \begin{array}{ccccc} -\mathbf{I} & \mathbf{I} & & & \\ \mathbf{O} & -\mathbf{I} & \mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{O} & -\mathbf{I} & \mathbf{I} \end{array} & \begin{array}{ccccc} \mathbf{B} & & & & \\ & \mathbf{B} & & & \\ & & \ddots & & \\ & & & & \mathbf{B} \end{array} \end{array}\right] \left[\begin{array}{c} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_n \\ \mathbf{U}_{n_1} \\ \hdashline \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_n \\ \mathbf{f}_{n_1} \\ \hdashline \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_n \end{array}\right] \tag{57}$$

or, more concisely,

$$\mathbf{D}_A\mathbf{U} - \mathbf{E}^T\mathbf{V} = \mathbf{f} \tag{58a}$$

$$\mathbf{E}\mathbf{U} + \mathbf{D}_B\mathbf{V} = \mathbf{g} \tag{58b}$$

Note that, in this formulation, $\mathbf{E}$ and $\mathbf{E}^T$ are nonsquare and $\mathbf{D}_A$ and $\mathbf{D}_B$ are square but of different block dimensions (see eqs. (22)).

Equations (58) can be reduced by eliminating $\mathbf{U}$. Premultiply equation (58a) by $-\mathbf{E}$, premultiply equation (58b) by $\mathbf{D}_n(\mathbf{A})$, add the two resulting equations, and use equations (24a) and (24d) to obtain directly the simple and very regular form of the block-tridiagonal equation for $\mathbf{V}$:

$$[\mathbf{T}_n(-\mathbf{I},\mathbf{C},-\mathbf{I})]\mathbf{V} = \mathbf{F} \tag{59}$$

where $\mathbf{F}$ is defined as in equations (32) with $p = n$, $q = n_1$. (In this case, there is no special condition on $\mathbf{f}_{p+1}$ as there was in the cases where $\mathbf{E}$ was square.) The expanded form of equation (59) is identical to equation (50b), except that in this case $\mathbf{C}$ has been allowed to have variable elements (eqs. (21c) and (21d) with (55)).

Version E of the solver uses cyclic reduction of equation (59). Version D is for the special case where $\alpha_{1,j}$ and $\alpha_{2,j-1}$ in equation (55) are constants. The cyclic reduction process for obtaining all values of $v_{j,k}$ from equation (59) is described in appendix F.

The procedure for obtaining the $\mathbf{U}_k$ after all $\mathbf{V}_k$ are known is somewhat different from the previous versions since k ranges from 1 to n for $\mathbf{V}_k$ but from 1 to n_1 for $\mathbf{U}_k$. *After* $\mathbf{U}_{n_1}$ is determined, the lower partition of equation (57) (i.e., rotationality equations) can be used to obtain the remaining $\mathbf{U}_k$:

$$\mathbf{U}_k = \mathbf{U}_{k+1} + \mathbf{B}\mathbf{V}_k - \mathbf{g}_k \qquad (k = n,\ n-1,\ \ldots,\ 1) \tag{60}$$

in which the j component of the vector represented by each product $\mathbf{B}\mathbf{V}_k$ is $\beta(v_{j,k} - v_{j+1,k})$ with $v_{m_1,k}$ set equal to zero for all k (since it is already included in $g_{j,k}$). To obtain $\mathbf{U}_{n_1}$ before equation (60) is used, several different approaches can be taken. Let us refer to these as options (a), (b), (c), and (d) and consider them in detail. For option (a) to determine $\mathbf{U}_{n_1}$, we use the bottom row of the upper partition in equation (57), that is, the continuity equation for the top row of dots in figure 2(c):

$$\mathbf{A}\mathbf{U}_{n_1} - \mathbf{V}_n = \mathbf{f}_{n_1} \tag{61a}$$

or

$$u_{j,n_1} = \frac{1}{\beta_{1,j}} \left(\beta_{2,j} u_{j-1,n_1} + v_{j,n} + f_{j,n_1} \right) \tag{61b}$$

for $j = 1$ to m (with u_{0,n_1} set equal to zero since it has been included in f_{1,n_1}). Because some difficulty has been encountered and studied in use of this option in version D of the solver, it has not been included in version E with variable $\beta_{1,j}$ and $\beta_{2,j}$. Therefore, for discussing the nature of the difficulty, we consider only the case (i.e., version D only) where

$$\left.\begin{aligned} \beta_{1,j} &= \beta_1 = \beta(1 - \bar{\alpha}_1) \\ \beta_{2,j} &= \beta_2 = \beta(1 + \bar{\alpha}_2) \quad \text{(except } \beta_{2,1} \equiv 0\text{)} \end{aligned}\right\} \tag{62}$$

with β_1 and β_2 as constants. Further, let

$$\left.\begin{aligned} w_j &= u_{j,n_1} \\ r_j &= v_{j,n} + f_{j,n_1} \end{aligned}\right\} \tag{63}$$

and write equation (61b) as

$$\beta_1 w_j - \beta_2 w_{j-1} = r_j \qquad (j = 1 \text{ to } m) \tag{64a}$$

in which $w_0 = 0$ or, equivalently, as

$$w_j - w_{j-1} = (\lambda - 1) w_{j-1} + \frac{1}{\beta_1} r_j \qquad (j = 1 \text{ to } m) \tag{64b}$$

where

$$\lambda \equiv \beta_2 / \beta_1 \tag{65}$$

This is a common form of difference equation, which can be studied by standard techniques in stability theory to determine that the equation is stable if $|\lambda| \leq 1$, i.e., $|\beta_2/\beta_1| \leq 1$. In fact, the difficulty encountered was traced to an instability in the determination of the u_{j,n_1} by this option only when the values of β_1 and β_2 were chosen so that $|\beta_2/\beta_1| > 1$.

Because of the stability difficulty with option (a) in these cases, further options were considered that find the u_{j,n_1} without using the top row of continuity equations on the mesh in figure 2(c). Option (b) is to just specify each u_{j,n_1}, in the step of the algorithm before equation (60) is used, as

$$u_{j,n_1} = (u_T)_j \qquad (j = 1 \text{ to } m) \tag{66}$$

where $(u_T)_j$ is obtained from some analytical representation such as an asymptotic boundary condition evaluated at each j and at $k' = n_1$ in figure 2(c). Options (c) and (d) use, instead, different variants of the rotationality equation to replace equation (56b) for application at points on $y = y_u$ (fig. 2(c)) halfway between the v input points and directly above each of the interior u points (i.e., at $k = n_1$, all j). For these two options, let $(u_T)_j$ be the respective values of u specified at these points on $y = y_u$, and denote $\partial u/\partial y$ at these same points by $(\partial u/\partial y)_{T,j}$. Option (c) will use a first-order-accurate expression for $(\partial u/\partial y)_{T,j}$ to replace the difference approximation given in equation (8b) that resulted in the rotationality difference equation (56b). Similarly, option (d) will use a different second-order-accurate approximation for $(\partial u/\partial y)_{T,j}$. The first-order-accurate expression, to replace $(\partial u/\partial y)_{j,k}$ given by equation (8b), at $k = n_1$, is

$$\left(\frac{\partial u}{\partial y}\right)_{T,j} \approx \left[(u_T)_j - u_{j,n_1}\right] \Big/ \left[\left(\tfrac{1}{2}\right) h_y\right] \tag{67}$$

Thus, for option (c), the replacement for equation (56b) rearranges to

$$u_{j,n_1} = (u_T)_j + \frac{1}{2}\beta\left(v_{j,n_1} - v_{j+1,n_1}\right) + \frac{1}{2} h_y \omega_{j,n_1} \qquad (j = 1 \text{ to } m) \tag{68}$$

For option (d), the second-order-accurate expression, to replace $(\partial u/\partial y)_{j,k}$ given by equation (8b), at $k = n_1$, is

$$\left(\frac{\partial u}{\partial y}\right)_{T,j} \approx \left[8(u_T)_j - 9u_{j,n_1} + u_{j,n}\right] \Big/ (3h_y) \tag{69}$$

The resulting replacement for equation (56b) is

$$8(u_T)_j - 9u_{j,n_1} + u_{j,n} - 3\beta\left(v_{j+1,n_1} - v_{j,n_1}\right) = -3h_y \omega_{j,n_1} \tag{70}$$

Before equation (70) can be solved for u_{j,n_1}, one must eliminate $u_{j,n}$. This can be done by adding equation (70) to (56b) written at $k = n$:

$$-u_{j,n} + u_{j,n_1} + \beta(v_{j,n} - v_{j+1,n}) = -h_y \omega_{j,n}$$

to obtain, after rearranging,

$$u_{j,n_1} = (u_T)_j + \frac{3}{8}\beta\left(v_{j,n_1} - v_{j+1,n_1}\right) + \frac{1}{8}\beta(v_{j,n} - v_{j+1,n}) + \frac{1}{8} h_y\left(3\omega_{j,n_1} + \omega_{j,n}\right) \qquad (j = 1 \text{ to } m) \tag{71}$$

which is the equation used to evaluate u_{j,n_1} for option (d) before equation (60) is used in the algorithm.

FORTRAN programs to solve the illustrative example problem, using subroutines corresponding to both versions D and E, are described later.

EXAMPLE PROBLEM FOR ILLUSTRATION

To illustrate the use of the variants and extensions of the Cauchy-Riemann solver, and as a basis for a complete sample program to be given for each version, consider the same problem used in reference 25. The sample programs can be used to determine such factors as computing times required for significant portions of the calculations and accuracy of the numerical results.

The problem is the small-perturbation formulation for steady, irrotational, incompressible, inviscid flow over a thin symmetrical parabolic-arc biconvex airfoil aligned with a uniform stream far from the airfoil. This problem has a simple mathematical formulation and the analytical solution is available for comparison with numerical results. Also, the problem has the interesting property of containing singularities, which should be captured by the numerical solution. The problem formulation and programs also constitute a basis for extensions to the more complex problems of nonlinear subsonic and transonic aerodynamics (refs. 35 to 38).

In Cartesian coordinates (x,y), let U and V be the respective components of velocity. For the classical small-perturbation approximation, let

$$U = U_\infty(1 + \tau u) \ , \qquad V = U_\infty \tau v \tag{72}$$

where u and v are then dimensionless scaled perturbation velocities, U_∞ is the uniform velocity at infinity, and τ is the small thickness ratio of the airfoil. If these expressions are substituted into the governing gas-dynamic conservation equations, the resulting equations for $u(x,y)$ and $v(x,y)$ are the Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{73a}$$

$$\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = 0 \tag{73b}$$

Let both x and y be normalized by the chord length of the airfoil and let the x axis be along the chord and the y axis be the perpendicular bisector of the chord. Then the biconvex-airfoil surface $y = y_b(x)$ is given by

$$y_b(x) = \pm\tau\left(\frac{1}{2} - 2x^2\right) \ , \qquad \left(-\frac{1}{2} \leqslant x \leqslant \frac{1}{2}\right) \tag{74}$$

In thin-airfoil theory, the condition of flow tangency at the airfoil surface $y = y_b(x)$ is transferred to $y = 0$ by use of Taylor's series (see, e.g.,

ref. 43). The first-order, thin-airfoil result for the boundary condition is then

$$\tau v(x,0\pm) = dy_b/dx \quad \text{in} \quad \left(-\frac{1}{2} < x < \frac{1}{2}\right)$$

or

$$v(x,0^+) = -4x \quad \text{in} \quad \left(-\frac{1}{2} < x < \frac{1}{2}\right) \tag{75a}$$

$$= 0 \quad \text{for} \quad |x| > \frac{1}{2} \tag{75b}$$

The condition of uniform flow at infinity requires that

$$u,v \to 0 \quad \text{as} \quad x^2 + y^2 \to \infty \tag{76}$$

The analytical solution to equations (73) with conditions (75) and (76) (see table A.2 of ref. 44, p. 21), in terms of the complex variable $z = x + iy$, is

$$u - iv = \frac{4}{\pi}\left\{1 - z\ \ell n\left[\frac{z + (1/2)}{z - (1/2)}\right]\right\} \tag{77}$$

from which, at $y = 0$,

$$u(x,0) = \frac{4}{\pi}\left[1 - x\ \ell n\left|\frac{x + (1/2)}{x - (1/2)}\right|\right] \tag{78}$$

Note from conditions (75) that v has a discontinuous jump at both the leading and trailing edges ($|x| = 1/2$) and, from equation (78), that u goes to negative infinity there.

For the computational problem, because of symmetry we consider only the half-plane $y \geqslant 0$ and use conditions (75) at points along the computational boundary $y = y_\ell = 0$. For the outer boundaries, we let $y = y_u$, $x = x_\ell$, and $x = x_u$ define the boundary lines where conditions are to be applied and consider using either $u = u_o(x,y)$ or $v = v_o(x,y)$ as the conditions to be applied there, where u_o and v_o are the analytical expressions obtained from the real and imaginary parts of the exact solution (77). With these exact solutions applied on the outer boundaries, replacing the asymptotic conditions (76), the illustrations will not be affected by errors due to approximate methods for applying conditions at infinity. However, one could also use zero perturbations ($u = v = 0$) on the outer boundaries as approximate conditions, and this option is allowed for in the programs described in the next section. Therefore, let us denote the outer boundaries (x_ℓ, x_u, y_u) as B and denote the conditions there as

$$u = u_B(x,y) \quad \text{or} \quad v = v_B(x,y) \quad \text{on} \quad B \tag{79}$$

where u_B may be either u_O or 0 and v_B may be either v_O or 0. The choice of whether to specify u_B or v_B at points on a section of B depends on which version of the algorithm described previously is used. Furthermore, in versions B and C, where $(v_T)_j$ must be specified at a half-interval inside the upper boundary, $(v_T)_j$ can be obtained from the function $v_B(x,y)$; in versions D and E, in which $(u_T)_j$ must be specified either at a half-interval inside the upper boundary for option (b) or on the upper boundary for options (c) and (d), $(u_T)_j$ can be obtained from the function $u_B(x,y)$.

To test and illustrate all versions of the algorithm, including those with the extra terms in equation (3a), using this example problem governed by equations (73), let us represent equations (73) by the finite-difference equations (3) with the notation defined in equations (4) to (8) and with s and ω equal to zero, so that, in equations (3a) and (3a'):

$$s'(\cdot) = a^+ u_O^+ + a^- u_O^- \tag{80}$$

in those cases for which a^+ and a^- are not zero. Therefore, for version C, the term $s'_{j,k}$ in equation (44a) is

$$h_y s'_{j,k} = -\beta\bar{\alpha}_2 (u_O)_{j-1,k} - \beta\bar{\alpha}_1 (u_O)_{j,k} \tag{81a}$$

and for versions D and E, in equation (56a),

$$h_y s'_{j,k} = -\beta\alpha_{2,j-1} (u_O)_{j-1,k} - \beta\alpha_{1,j} (u_O)_{j,k} \tag{81b}$$

where u_O is the exact solution defined above.

To summarize the numerical example problem for illustration, the difference equations approximating the partial differential equations (73) are equations (3) with (80) represented by equations (81) for the algorithm versions with the extra terms; the boundary conditions are the discretized applications of equations (75) on $y = y_\ell = 0$ and of equations (79) on $x = x_\ell$, $x = x_u$, $y = y_u$, with the choice of whether u or v is specified on any section of B depending on the algorithm version being used.

After the example problem is solved by any version of the algorithm, it is of interest to determine u on $y = 0$ (which approximates the airfoil-surface velocity for $-\frac{1}{2} < x < \frac{1}{2}$ in thin-airfoil theory). Since u on $y = 0$ is not determined directly in the solutions obtained on the meshes in figure 2, a second-order-accurate representation of the rotationality equation can be applied on $k = 0$, with one-sided y differencing, analogous to equation (70) used in versions D and E for $k = n_1$. The rotationality equation can be arranged to obtain (ref. 25):

$$u_{j,0} = (1/8)[9u_{j,1} - u_{j,2} + 3\beta(v_{j,0} - v_{j+1,0}) + 3h_y\omega_{j,0}] \quad (j = 1 \text{ to } m) \tag{82}$$

(in which $u_{j,0}$ is the value at $k = 0$ rather than at $k' = 0$). In this example problem, of course, $\omega_{j,0} = 0$ in equation (82).

FORTRAN PROGRAMS AND RESULTS FOR THE EXAMPLE PROBLEM USING VERSIONS A TO E OF CAUCHY-RIEMANN SOLVER

The purposes of this section are (a) to describe FORTRAN programs, listed in appendices A to E, for solving the simple example problem outlined in the above section, using versions A to E of the Cauchy-Riemann solver and (b) to describe results of calculations for the example problem.

FORTRAN Program Descriptions

The FORTRAN programs are written for use on a Control Data 7600 computer. Slight modifications would be required for use on most other computers that have FORTRAN compilers. Each program consists of a main program and three subroutines, as listed in table II. The last letter of the name of

TABLE II.- PROGRAMS AND SUBROUTINES

UBIC1A	UBIC1B	UBIC1C	UBIC1D	UBIC1E
CALCD1	CALCD	CALCD	CALCD	CALCD
CR2N1	CR2DUV	CR2UVC	CR2UVC	CR2UVE
GE2N1	GE2N1	GE2N2	GE2N2	GEUVE

the main program denotes the version of the Cauchy-Riemann solver used; thus, UBIC1A is the main program to solve the example problem for the biconvex airfoil using version A of the solver. The subroutine CALCD used with versions B to E computes the quantities $d_{\ell,M}$ (eqs. (F7) in appendix F). The corresponding computation in version A is done by subroutine CALCD1. The subroutines CR2DUV, CR2UVC, and CR2UVE perform the cyclic reduction as outlined in appendix F for versions B, C, D, and E. The subroutine CR2N1 performs the version of cyclic reduction described in reference 25 for version A. Within each of the subroutines for cyclic reduction, another subroutine is called to solve tridiagonal equations using the Thomas algorithm (e.g., see ref. 45) for Gaussian elimination. The slightly different subroutine versions for this algorithm are named GE2N1, GE2N2, and GEUVE.

The use of these partially modular forms of the programs, with separate subroutines to perform major steps in the program, is not the most efficient but is best for understanding, modifying, and adapting the programs.

Timer- Each main program includes the use of a timing subprogram from the CDC-7600 program library that returns the CPU time T in seconds (from the start of the job) either by the statement CALL SECOND(T) or, for example, by CPU=SECOND(T). This subprogram is used to measure three times in each main program: T1 is the time for all preliminary computations, excluding input,

before the cyclic reduction; T2 is the time for the cyclic reduction to be completed (determination of **V** after **F** is known); and T3 is the time required to obtain **U** after **V** is known. When the program is used on other computers, all statements containing the FORTRAN variables TT1, TT2, T1, T2, and T3 should be removed or appropriately replaced.

FORTRAN parameters and variables- Before further description of the programs, it is convenient to list in table III some FORTRAN parameters and variables that either are used frequently or are special and that correspond to the indicated equivalent algebraic quantities. For this we note that, since FORTRAN does not permit the index zero in an array, the FORTRAN indices J and K, for example, are shifted by one from the values of j and k. The variable J may represent either JU or JV in table III and the variable K may represent either KU or KV in the same sense as j and k are used above as indices to represent, respectively, either j or j' and either k' or k.

Statement functions- At the beginning of each main program, statement functions are included to determine X = XJU(J) or XJV(J) and Y = YKU(K) or YKV(K) from equations (4) (see table III). Statement functions are also given for the exact analytical solutions for u and v, UEX(X,Y) and VEX(X,Y), which include within the statements the additional statement functions RHO(X,Y) and ANPHI(X,Y). The functions UEX and VEX represent the real and imaginary parts of equation (77). In addition, version E contains the statement functions F1(JU) and F2(JU), which are arbitrarily specified functions to represent $\alpha_{1,j}$ and $\alpha_{2,j}$, respectively, in the test calculation.

Computation of boundaries- Because of the staggered mesh, in some cases it is most convenient to specify nominal values of x_ℓ, x_u, and y_u, and then compute final values of those parameters for which convenient mesh intervals, h_x and h_y, are obtained. For this reason, after the nominal values are input, and m_1 and n_1 are input, h_x and h_y are obtained from equations (9d) and (9e), then equations (9a) and (9b) or (9c) are used to determine the final x_ℓ, x_u, and y_u, taking into account the fact that it is desired to have $x_j = -0.5$ (as given by eq. (4a)) coincide with the leading edge of the airfoil at some J = JLE. This computation is done just after the input is read at the beginning of each main program.

Input- As indicated near the beginning of each main program, versions A, B, and E each require only one input card per case (READ statement labeled 2 with corresponding FORMAT statement labeled 1), and versions C and D require two input cards per case (READ statements labeled 2 and 104 with corresponding FORMAT statements labeled 1 and 103). In each program, *the first (or one only) input card format for each case* requires the following specifications:

NEX Integer in column 5. Specification is arbitrary and superfluous unless NX is specified as some integral power of 2 (NX=2**NEX). In the latter event, specify NEX equal to $\log_2$NX.

NX Integer, right-justified at column 10: Mesh number for x direction, equal to m_1. Specify integer ranging from 4 to 128. (Recommended values either have a factor of 10 or are a power of 2; see table I.)

TABLE III.- FORTRAN SYMBOL DEFINITIONS

FORTRAN	Algebraic	Reference equation	Reference page
JU	$j + 1$	(4a)	9
JV	$j' + 1$	(4b)	9
KU	$k' + 1$	(4b)	9
KV	$k + 1$	(4a)	9
NX	m_1, $m + 1$		8
NY	n_1, $n + 1$		8,9
JM	$m_1 + 1$		8
KM	$n_1 + 1$		9
NEY	L		9
XL, XU	x_ℓ, x_u		6,9
YL, YU	y_ℓ, y_u		6,9
XJU(J)	x_j	(4a)	9
XJV(J)	$x'_{j'}$	(4b)	9
YKU(K)	$y'_{k'}$	(4b)	9
YKV(K)	y_k	(4a)	9
HX, HY	h_x, h_y	(9)	
U(J,K)	$u_{j',k}$	(7)	9
V(J,K)	$v_{j,k'}$	(7)	9
UEX(X,Y)	$u_o(x,y)$		31
VEX(X,Y)	$v_o(x,y)$		31
BETA	β	(25)	
BETA1	$\beta_{1,j}$	(43),(55)	
BETA2	$\beta_{2,j}$	(43),(55)	
ALPHA1	$\bar{\alpha}_1$, $\alpha_{1,j}$	(42),(54)	
ALPHA2	$\bar{\alpha}_2$, $\alpha_{2,j}$	(42),(54)	
A	β_j	(21d)	
A1	β_1	(21d)	
ALFA	β'_j	(F10c)	
ALFA1	β'_1	(F10c)	
GAMMA	$-\alpha_j$	(21d)	
DELTA	$-\gamma_j$	(21d)	
ALF(J)	α_j	(21d)	
BET(J)	β_j	(21d)	
GAM(J)	γ_j	(21d)	
VT(J)	$(v_T)_j$		22
UT(J)	$(u_T)_j$		29

NEY Integer in column 15. Since NY is always an integral power of 2 (NY = 2**NEY), specify NEY equal to $\log_2$NY.

NY Integer, right-justified at column 20: Mesh number for y direction, equal to n_1. Specify integral power of 2 (NY=2**NEY) with NEY ranging from 2 to 7.

On this same card, as described in the previous subsection (see table III for correspondence between XL and x_ℓ, etc.), nominal values are specified for XL, XU, YL, and YU. The values have E fields of width 10 and are right-justified, respectively, in columns 30, 40, 50, and 60. Recommended values for the example problem are, respectively, -1.E0, 1.E0, 0.E0, and 2.E0. (For the example problem as formulated, YL is always 0.E0.) The same input card in all the program versions also requires specification of the FORTRAN parameter NBC in column 65 as follows:

NBC = 1 results in the outer boundary condition being specified as the exact analytical solution;

= 2 results in zero values being specified for the conditions on the outer boundaries.

In addition to the above parameters, which are all specified on the first (or only) input card for each case, the following parameter options apply as indicated:

LASCAS = 0 causes the program to return, after a case has been run, to the first READ statement for a new case;

= 1 causes the program to end; should be specified for last case in a job.

NUBDRY = 0 in program D, specifies option (a) (eq. (61b)) for determination of u_{j,n_1};

= 1 in programs D and E, specifies option (b) (eq. (66)) for u_{j,n_1};

= 2 in programs D and E, specifies option (c) (eq. (68)) for u_{j,n_1};

= 3 in programs D and E, specifies option (d) (eq. (71)) for u_{j,n_1}.

ALPHA1 and ALPHA2 are arbitrarily specified numbers in programs C and D except that, to keep the difference equations elliptic, require both $(-\infty < \bar{\alpha}_1 < 1)$ and $(-1 < \bar{\alpha}_2 < \infty)$ and, in addition, to keep the calculation stable for the option NUBDRY = 0 in program D, require also $\left|(1 + \bar{\alpha}_2)/(1 - \bar{\alpha}_1)\right| \leq 1$.

In programs A and B, LASCAS is an integer in column 70; in program E, it is in column 75; in programs C and D, it is in column 25 of the *second* input card for each case. The integer parameter NUBDRY is in column 70 of the first input card for each case in programs D and E. The parameters ALPHA1 and ALPHA2 for programs C and D are specified with E fields of width 10 and are right-justified, respectively, at columns 10 and 20 of the second input card

for each case. For convenience, table IV summarizes the input card and column locations for the input parameters.

TABLE IV.- INPUT CARD AND COLUMN LOCATIONS FOR INPUT PARAMETERS
(Column of right-justification)

Input parameter:	NEX	NX	NEY	NY	XL	XU	YL	YU	NBC	LASCAS	NUBDRY	ALPHA1	ALPHA2
Version A column	5	10	15	20	30	40	50	60	65	70			
Version B column	5	10	15	20	30	40	50	60	65	70			
Version C card, column	1,5	1,10	1,15	1,20	1,30	1,40	1,50	1,60	1,65	2,25		2,10	2,20
Version D card, column	1,5	1,10	1,15	1,20	1,30	1,40	1,50	1,60	1,65	2,25	1,70	2,10	2,20
Version E column	5	10	15	20	30	40	50	60	65	75	70		

Note that the arbitrarily specified statement functions F1(J) and F2(J) determine $\alpha_{1,j}$ and $\alpha_{2,j}$ in program E, so these quantities are not input for version E; these specifications are easily changed by simply replacing F1(J) and F2(J) in the main program by suitable statement functions. These specifications should satisfy $(-\infty < \alpha_{1,j} < 1)$ and $(-1 < \alpha_{2,j} < \infty)$ at all j in $(1 \leqslant j \leqslant m)$.

Outline of programs- The FORTRAN programs have the following major steps (see listings in appendices):

1. Specify DIMENSION and COMMON statements.

2. Specify statement-function declarations.

3. Read input as described in above section.

4. Calculate x_ℓ and x_u (and y_u in versions B and C) from the nominal input values.

5. Calculate needed parameters.

6. Calculate $(v_T)_j$ in versions B and C (see p. 22).

7. Initialize interior values of $f_{j,k}$ and $g_{j,k}$ (to zero for version A; to $(v_T)_j$ plus the right side of equation (81a) for versions B and C; to the right side of equation (81b) for versions D and E).

8. Calculate needed boundary values of u and v.

9. Calculate $(u_T)_j$ in versions D and E.

10. Complete the determination of $f_{j,k}$ and $g_{j,k}$ by "modifying the fringes" to include the boundary values of u and v (see definitions of $f_{j,k}$ and $g_{j,k}$ for each version; e.g., p. 19).

11. Set appropriate boundary values of $u_{j,k}$ and $v_{j,k}$ to zero as required by the reduction algorithms.

12. Determine each $F_{j,k}$ (denoted F0 in the program listings) according to equations (32) as described for each program version.

13. Determine needed values of $d_{\ell,m}$ (eqs. (F7)) by calling either CALCD or CALCD1.

14. Determine β_j and β_1 (A and A1; see table III above) in all versions except E (in which these quantities are in the array BET(J), included under step 5 above).

15. Perform cyclic reduction according to the algorithm outlined in appendix F for versions B to E or in reference 25 for version A (call subroutine names beginning with CR in table II) to obtain all unknown values of $v_{j,k}$.

16. Determine all $u_{j,k}$ by procedure described previously for each version.

17. Reload boundary values of $u_{j,k}$ and $v_{j,k}$ that were previously set to zero.

18. Print as output, for appropriate values of j and k: J, K, X and Y (for U), U, X and Y (for V), V, UEX and VEX evaluated at same points as U and V, and ERRU, ERRV, which are the differences between U and UEX and between V and VEX.

19. Print T1, T2, T3.

20. Print values of $u_{j,0}$ calculated from equation (82) along with values of $v_{j,0}$ and the exact solution for comparison.

21. For convenience in plotting the exact solution for u on $y = 0$ near the leading- and trailing-edge singularities, also calculate and print values at small intervals near these edges.

Results

The results of computations for the illustrative example problem of the biconvex airfoil include, in particular, plots of the computed values of u on $y = 0$ versus x, compared with the analytical solution, and the computing times.

For this example problem, there is very little difference in the results for u on $y = 0$ computed by the different program versions, A to E, with the same input conditions, and for any of the options (a) to (d) in version E. The very small differences are virtually indistinguishable on a plot such as shown in figure 3, for which the results were computed with NX = 40, NY = 32, NBC = 1, the nominal values of x_ℓ, x_u, y_ℓ, and y_u as recommended above, and with various values of $\bar{\alpha}_1$ and $\bar{\alpha}_2$ in versions C and D. For these input parameters, the mesh intervals h_x and h_y are, respectively, 0.05 and 0.0625. The outer computation boundaries are at or about 1/2 chord length upstream and downstream from the airfoil edges and 2 chord lengths above the airfoil. Because of the discontinuous conditions on v at the airfoil edges, $(x,y) = (\pm 0.5, 0.0)$, the analytical perturbation solution for u goes to negative infinity there. These singularities are captured well by the numerical algorithms and, even on the relatively coarse mesh used for figure 3, the values very near those points are accurate.

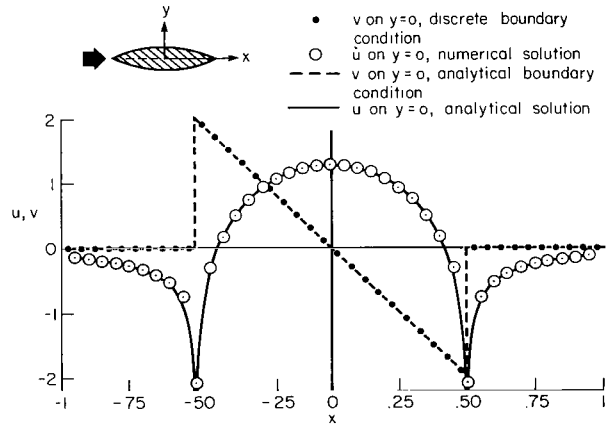


Figure 3.- Perturbation velocities for thin parabolic-arc biconvex airfoil (NX = 40, NY = 32, exact conditions on boundaries).

The computing times T1, T2, and T3, as described under the subsection "Timer," are listed in table V for various meshes and for all program versions A to E. Note that, when used in an iterative calculation such as described in the Introduction and the section that follows it, no significant part of the time T1 would be done more than once (at the beginning of

the program). Thus, since T3 $\ll$ T2, the only time of real significance for practical applications is T2. As one example, we see that the time T2 for the cyclic reduction using version C for NX = 40, NY = 32 is 16 msec. In reference 37, it was noted that the direct solver for this case required only 14 msec. The difference is due to the fact that a special tridiagonal-solution subroutine that takes advantage of the vector capabilities of the CDC-7600 was used for the quoted slightly lower time. (The time saving was much more significant when a previous version of the FTN compiler was being used, before the FTN 4.4 version indicated in table V became available.)

TABLE V.- COMPUTING TIMES FOR BICONVEX AIRFOIL USING VARIANTS OF CAUCHY-RIEMANN SOLVER ON CDC-7600
(FTN 4.4, OPT = 2 Compiler)

Version of program	NX	NY	T1, sec	T2, sec	T3, sec
A	10	8	0.005	0.001	0
	20	16	.006	.005	0
	40	32	.009	.022	.001
	60	64	.013	.076	.004
	100	128	.025	.287	.011
B	10	8	.001	.001	0
	20	16	.002	.004	0
	40	32	.005	.015	.001
	60	64	.010	.056	.003
	100	128	.025	.218	.010
C	10	8	.002	.001	0
	20	16	.009	.003	0
	40	32	.032	.016	.001
	60	64	.092	.058	.003
	100	128	.299	.223	.010
D	10	8	.002	.001	0
	20	16	.009	.003	.001
	40	32	.032	.016	.001
	60	64	.092	.057	.003
	100	128	.296	.223	.010
E	10	8	.003	.001	0
	20	16	.009	.003	0
	40	32	.032	.018	.001
	60	64	.092	.062	.003
	100	128	.299	.240	.011

User's Choice of Algorithm Versions

To aid the prospective user in selecting which version of the algorithm to use, we note the following features:

Version A:

(1) solves the nonhomogeneous Cauchy-Riemann equations;
(2) uses a grid that is nonsymmetrical in both x and y (conditions on u imposed on left and top, v on bottom and right);
(3) has a potential source of errors at the upper boundary due to first-order accuracy of the difference equation there;
(4) is the least efficient of the algorithm versions;
(5) was used for iterative computations in reference 35.

Version B:

(1) solves the nonhomogeneous Cauchy-Riemann equations;
(2) uses a grid that is nonsymmetrical in both x and y (conditions on u imposed on left and top, v on bottom and right);
(3) is the most efficient version for the simplest nonhomogeneous Cauchy-Riemann equations with the nonsymmetrical grid.

Version C:

(1) solves extended nonhomogeneous Cauchy-Riemann equations, with the extra terms having constant coefficients;
(2) uses a grid that is nonsymmetrical in both x and y (conditions on u imposed on left and top, v on bottom and right);
(3) is the most efficient version with extra terms (constant coefficients) and the nonsymmetrical grid;
(4) was used for iterative computations in references 36 to 38.

Version D:

(1) solves extended nonhomogeneous Cauchy-Riemann equations, with the extra terms having constant coefficients;
(2) uses a grid that is symmetrical in y (conditions on u imposed on left, v on top, bottom, and right;
(3) is the most efficient version with the symmetrical grid.

Version E:

(1) solves extended nonhomogeneous Cauchy-Riemann equations, with variable coefficients in the extra terms;
(2) uses a grid that is symmetrical in y (conditions on u imposed on left, v on top, bottom, and right);
(3) is the only version presented having variable coefficients in the extra terms.

CONCLUDING REMARKS

Revised and extended versions of the original Cauchy-Riemann solver of reference 25 have been presented, including detailed derivations of the algorithms, as well as descriptions and listings of FORTRAN computer programs. These programs are believed to be simple and adaptable enough to be useful to nonspecialist users.

The results of the computations for the simple example problem are believed to demonstrate significant efficiency, accuracy, and potential utility of the algorithms.

In addition to the previous applications of the original and the first revised and extended Cauchy-Riemann solvers in semidirect iterative methods for the simplest transonic-flow calculations that have been described in references 35 to 38, it is anticipated that the newest version, version E, will be useful for more general applications in transonic-flow calculations, including lifting airfoils. The Cauchy-Riemann solvers are sufficiently general that further significant applications can also be expected in other fluid-dynamical problems, as well as in broader areas of mathematical physics. It is expected that further extensions of these methods in the future, to include variable meshes and three dimensions, will also be worthwhile.

Ames Research Center
National Aeronautics and Space Administration
Moffett Field, Calif. 94035, December 16, 1976.

APPENDIX A

FORTRAN LISTING OF EXAMPLE PROGRAM AND SUBROUTINES FOR ORIGINAL CAUCHY-RIEMANN SOLVER, VERSION A

```
C     MAIN PROGRAM, UBIC1A76 FOR CDC 7600
C+++  SOLUTION OF CAUCHY-RIEMANN EQS. BY CYCLIC REDUCTION FOR INCOM. FLOW
C     OVER THIN BICONVEX AIRFOIL.  AIRFOIL CHORD IS FROM X=-.5 TO +.5.
C
C     THE METHOD AND THE PROGRAM ARE DESCRIBED IN NASA TN D-7934
C     BY E. D. MARTIN AND H. LOMAX.
C
      DIMENSION U(129,130),V(129,130)
      COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
C
      XJU(JU) = XL + HX*FLOAT(JU-1)
      YKU(KU) = YL + HY*(FLOAT(KU)-1.5)
      XJV(JV) = XL + HX*(FLOAT(JV)-1.5)
      YKV(KV) = YL + HY*FLOAT(KV-1)
      RHO(X,Y) =  SQRT(((X+.5)**2 + Y*Y)/((X-.5)**2 + Y*Y))
      ANPHI(X,Y) = ATAN2(Y,X-.5) - ATAN2(Y,X+.5)
      UEX(X,Y) = FCTR*(1. - X*ALOG(RHO(X,Y)) - Y*ANPHI(X,Y))
      VEX(X,Y) = FCTR*(Y*ALOG(RHO(X,Y))- X*ANPHI(X,Y))
C
      FCTR = 4./3.14159265
 1    FORMAT(4I5,4E10.0,2I5)
 2    READ(5,1) NEX,NX,NEY,NY,XL,XU,YL,YU,NBC,LASCAS
      CALL SECOND(TT1)
C     THE INPUT VALUES OF XL AND XU ARE NOMINAL VALUES, USED TO
C     COMPUTE THE EXACT VALUES.
      HY = (YU - YL)/FLOAT(NY)
      HX = (XU - XL)/FLOAT(NX)
      JLE = 1.005 + (-.5 - XL)/HX
      XL = -.5 - HX*(JLE - 1)
      XU = XL + HX*(NX - .5)
      JM = 1 + NX
      KM = 1 + NY
      KMP2 = KM + 2
      BETA = HY/HX
      BETSQ = BETA**2
C+++  INITIALIZE INTERIOR VALUES OF MASS SOURCES AND VORTICITY (F IS
C     EQUIV. TO V AND G IS EQUIV. TO U)--
      DO 3 K = 1,130
        DO 4 J = 1,129
          V(J,K) = 0.
          U(J,K) = 0.
 4      CONTINUE
 3    CONTINUE
C++++ LOAD BOUNDARY VALUES OF U AND V--
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      K = KM + 1
      DO 5 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1)= -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
 5    CONTINUE
      DO 6 K = 2,KM
        IF (NBC.EQ.2) GO TO 33
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
```

```
        GO TO 6
 33     U(1,K) = 0.
        V(JM,K) = 0.
 6    CONTINUE
C+++  MODIFY FRINGES OF INTERIOR TO INCLUDE BOUNDARY VALUES--
      DO 7 J = 2,NX
        V(J,2) = V(J,2) + V(J,1)
        U(J,KM) = U(J,KM) - U(J,KM + 1)
 7    CONTINUE
      DO 8 K = 2,NY
        V(2,K) = V(2,K) + BETA*U(1,K)
        U(NX,K) = U(NX,K) + BETA*V(JM,K)
 8    CONTINUE
      K = KM
      V(2,K) = V(2,K) + BETA*U(1,K)
      U(NX,K) = U(NX,K) + .5*BETA*V(JM,K)
C+++  ZERO APPROPRIATE BOUNDARY VALUES--
      K = KM + 1
      DO 9 J = 1,JM
        U(J,K) = 0.
        V(J,K) = 0.
        V(J,1) = 0.
 9    CONTINUE
      DO 10 K = 1,KM
        V(JM,K) = 0.
        U(1,K) = 0.
        V(1,K) = 0.
 10   CONTINUE
C+++  NEXT DETERMINE VECTOR F0.
      DO 11 K = 2,KM
        DO 12 J = 2,NX
          V(J,K) = V(J,K) - V(J,K+1) + BETA*(U(J,K) - U(J-1,K))
 12     CONTINUE
 11   CONTINUE
      CALL CALCD1(NEY)
      A = 2.*(1. + BETSQ)
      A1 = 2. + BETSQ
      TT2 = TT1
      T1 = SECOND(TT1) - TT2
      CALL CR2N1
      TT2 = TT1
      T2 = SECOND(TT1) - TT2
C+++  NOW V IS DETERMINED FOR J = 2 TO NX AND K = 2 TO NY+1.
C     NEXT DETERMINE U FOR THE SAME J AND K.
      BETK = .5*BETA
      DO 15 KR = 2,KM
        K = KMP2 - KR
        IF (K.LT.KM) BETK = BETA
        DO 16 J = 2,NX
          U(J,K) = U(J,K+1) - U(J,K) + BETK*(V(J,K) - V(J+1,K))
 16     CONTINUE
 15   CONTINUE
      TT2 = TT1
      T3 = SECOND(TT1) - TT2
C+++  RELOAD BOUNDARY VALUES OF U AND V.
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      K = KM + 1
```

```
      DO 17 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1 ) = -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
17    CONTINUE
      DO 18 K = 2,KM
        IF (NBC.EQ.2) GO TO 34
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
        GO TO 18
34      U(1,K) = 0.
        V(JM,K) = 0.
18    CONTINUE
19    FORMAT(1H1*SOLUTION OF CAUCHY-RIEMANN EQS. BY CYCLIC REDUCTION.*/
     1* INCOMPRESSIBLE FLOW OVER THIN BICONVEX AIRFOIL. CHORD IS FROM X=
     2-.5 TO +.5.*//* NBC=*,I3,*. (IF NBC IS 1, EXACT BC'S ARE IMPOSED
     3FAR FROM AIRFOIL; IF NBC IS 2, U AND V ARE ZERO ON OUTER BOUNDARY.
     4*//
     5* NX=*,I3,*, NY=*,I3,*, XL=*,F10.6,*, XU=*,F10.6,*, YL=*,F10.6,
     6*, YU=*,F10.6,*, HX=*,E15.7,*, HY=*,E15.7,*.*//9X,1HJ,3X,1HK,6X,
     73HXJU,7X,3HYKU,8X,1HU,8X,3HUEX,8X,4HERRU,11X,3HXJV,7X,3HYKV,8X,1HV
     8,8X,3HVEX,8X,4HERRV)
      WRITE(6,19) NBC,NX,NY,XL,XU,YL,YU,HX,HY
20    FORMAT(6X,2I4,F11.6,3F10.6,E15.7,F11.6,3F10.6,E15.7)
      KI = NY/16
      IF (KI.EQ.0) KI = 1
      IF (NX-2**NEX.EQ.0) GO TO 21
      JI = NX/20
      GO TO 22
21    JI = NX/16
22    IF (JI.EQ.0) JI = 1
23    FORMAT(1X)
      DO 24 K = 1,KM,KI
        WRITE(6,23)
        YA = YKU(K)
        YB = YKV(K)
        DO 25 J = 1,JM,JI
          XA = XJU(J)
          XB = XJV(J)
          IF (J.EQ.JM.OR.K.EQ.1) GO TO 26
          UEXA = UEX(XA,YA)
          UA = U(J,K)
          GO TO 27
26        UEXA = 0.
          UA = 0.
27        IF (J.EQ.1) GO TO 28
          IF (K.NE.1) GO TO 35
          IF (ABS(XB*XB-.25).GE.1.E-5) GO TO 35
          VEXB = 0.
          GO TO 36
35        VEXB = VEX(XB,YB)
36        VB = V(J,K)
          GO TO 29
28        VEXB = 0.
          VB = 0.
29        ERRU = UA - UEXA
```

```
          ERRV = VB - VEXB
          WRITE(6,20) J,K,XA,YA,UA,UEXA,ERRU,XB,YB,VB,VEXB,ERRV
25      CONTINUE
24    CONTINUE
30    FORMAT(//* T1=*,E12.4,* SEC., T2=*,E12.4,* SEC., T3=*,F12.4,* SEC.
     1*)
      WRITE(6,30) T1,T2,T3
38    FORMAT(///* VELOCITIES AT Y = 0.*)
39    FORMAT(//26X,1HJ,6X,3HXJU,10X,1HU,13X,3HUEX,11X,3HXJV,10X,1HV,13X,
     13HVEX//)
40    FORMAT(23X,I4,F11.6,2E15.7,F11.6,2E15.7)
      WRITE(6,38)
      WRITE(6,39)
      DO 42 J = 2,NX
        XA = XJU(J)
        XB = XJV(J)
        UEXA = 0.
        VEXB = 0.
        IF (ABS(XA*XA-.25).GE.1.E-5) UEXA = UEX(XA,0.)
        IF (XB*XB.LE..25001) VEXB = -4.*XB
        VB = V(J,1)
        UA =        0.125*(9.*U(J,2)-U(J,3)+3.*BETA*(V(J,1)-V(J+1,1)))
        WRITE(6,40) J,XA,UA,UEXA,XB,VB,VEXB
42    CONTINUE
      WRITE(6,45)
      DELTX = .1*HX
      X = -.5 - 1.1*HX
      N = 0
46    N = N + 1
      IF (N.EQ.2) X = .5 - 1.1*HX
      WRITE(6,23)
      DO 47 I = 1,21
        X = X + DELTX
        UEX1 = 0.
        VEX1 = 0.
        IF (ABS(X*X-.25).GE.1.E-5) UEX1 = UEX(X,0.)
        IF (X*X.LE..25001) VEX1 = -4.*X
        WRITE(6,48) X,UEX1,VEX1
47    CONTINUE
      IF (N.EQ.1) GO TO 46
45    FORMAT(//* EXACT VELOCITIES AT EDGES.*//29X,1HX,11X,3HUEX,12X,3HVE
     1X)
48    FORMAT(23X,F10.6,2E15.7)
      IF (LASCAS.EQ.0) GO TO 2
      STOP
      END



      SUBROUTINE CALCD1(NEY)
C     DIMENSIONS OF D(L,M) ARE LMAX = NEY AND MMAX = 2**LMAX.
      DIMENSION D(7,128)
      COMMON/DC/D
      LMAX = NEY
      D(1,1) = SQRT(2.)
      D(1,2) = -D(1,1)
```

```
      DO 30 L = 2,LMAX
        MMAX = 2**L
        MMAXH = MMAX/2
        MMAXH1 = MMAXH + 1
        DO 10 M = 1,MMAXH
          D(L,M) = SQRT(2.+ D(L-1,M))
10      CONTINUE
        DO 20 M = MMAXH1,MMAX
          MM = M - MMAXH
          D(L,M) = -D(L,MM)
20      CONTINUE
30    CONTINUE
      RETURN
      END




      SUBROUTINE CR2N1
C+++  ROUTINE TO SOLVE BY CYCLIC REDUCTION A BLOCK-TRIDIAGONAL
C     MATRIX EQUATION WITH A NEUMANN-LIKE CONDITION IN 2 DIMENSIONS.
C
      DIMENSION V(129,130),D(7,128),DUM(129)
      COMMON NEX,NX,NEY,NY,RETSQ,A,A1,ALFA,ALFA1,K,V/DC/D
C
      JM = 1 + NX
      KM = 1 + NY
      LMAX = NEY
C+++  FORWARD RECURSION--FIRST LEVEL IS L=1.
      KH = KM
      EPS = 1.
      DO 1 K = 3,KH,2
        DO 2 J = 2,NX
          V(J,K) = 2.*V(J,K)
2       CONTINUE
        ALFA1 = A1
        ALFA = A
        CALL GE2N1
        IF (K.EQ.KM) EPS = 0.
        DO 3 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) + EPS*V(J,K+1)
3       CONTINUE
1     CONTINUE
      DO 4 L = 2,LMAX
        NH1 = 2**(L-2)
        NH2 = 2*NH1
        NH3 = 3*NH1
        NH4 = 4*NH1
        KL = NH4 + 1
        KH = KM
        IEPS = 1
        DO 5 K = KL,KH,NH4
          IF (K.EQ.KM) IEPS = 0
          EPS = FLOAT(IEPS)
          K11 = K - NH1
          K12 = K + NH1*IEPS
```

```
            K21 = K - NH2
            K22 = K + NH2*IEPS
            K31 = K - NH3
            K32 = K + NH3*IEPS
            DO 6 J = 2,NX
              DUM(J) = V(J,K)
              V(J,K) = 2.*V(J,K)-V(J,K11)+V(J,K21)-V(J,K31)
     1                 +EPS*(-V(J,K12)+V(J,K22)-V(J,K32))
 6          CONTINUE
            DO 7 M = 1,NH2
              ALFA1 = A1 + D(L-1,M)
              ALFA = A + D(L-1,M)
              CALL GE2N1
 7          CONTINUE
            DO 8 J = 2,NX
              V(J,K) = V(J,K) + DUM(J) - V(J,K11) + V(J,K21) + EPS*(-V(J,
     1                 K12) + V(J,K22))
 8          CONTINUE
 5        CONTINUE
 4      CONTINUE
C
C+++    BACKWARD RECURSION-- DEFINE NEW INDEX, LB = LMAX-L+1 = NEY-L+1.
        L = LMAX
        K = KM
        NH2 = 2**(L-1)
        NH4 = 2*NH2
        K21 = K-NH2
        DO 9 J = 2,NX
          DUM(J) = V(J,K)
 9      CONTINUE
        DO 10 M = 1,NH4
          ALFA1 = A1 + D(L,M)
          ALFA = A + D(L,M)
          CALL GE2N1
 10     CONTINUE
        DO 11 J = 2,NX
          V(J,K) = 2*V(J,K) + DUM(J) - V(J,K21)
 11     CONTINUE
        DO 12 LB = 2,LMAX
          L = LMAX - LB + 1
          NH2 = 2**(L-1)
          NH4 = 2*NH2
          KL = NH4 + 1
          KH = KM - NH4
          KI = 2*NH4
          DO 13 K = KL,KH,KI
            K21 = K - NH2
            K22 = K + NH2
            K41 = K - NH4
            K42 = K + NH4
            DO 14 J = 2,NX
              DUM(J) = V(J,K)
              V(J,K) = V(J,K) + V(J,K41) + V(J,K42)
 14         CONTINUE
            DO 15 M = 1,NH4
              ALFA1 = A1 + D(L,M)
              ALFA = A   + D(L,M)
```

```
            CALL GE2N1
 15       CONTINUE
          DO 16 J = 2,NX
            V(J,K) = V(J,K) + .5*(DUM(J) - V(J,K21) - V(J,K22))
 16       CONTINUE
 13     CONTINUE
 12   CONTINUE
      DO 17 K = 2,NY,2
        DO 18 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) + V(J,K+1)
 18     CONTINUE
        ALFA1 = A1
        ALFA = A
        CALL GE2N1
 17   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GE2N1
C+++  ROUTINE TO SOLVE BY GAUSSIAN ELIMINATION THE TRIDIAG. EQ.,
C     T(-BETSQ,ALFA,-BETSQ)*V = F (WITH THE FIRST DIAG. ELEMENT REPLACED
C     BY ALFA1), WHERE V AND F ARE 2-DIMENSIONAL.
C     IN THE PROGRAM, V AND F ARE EQUIVALENT.
C
      DIMENSION V(129,130),S(129)
      COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
C
      ALFA1I = 1./ALFA1
      BETSQI = 1./BETSQ
      ALFB = ALFA/BETSQ
      S(2) = -ALFA1I*BETSQ
      V(2,K) = V(2,K)*ALFA1I
      DO 1 J = 3,NX
        S(J) = -1./(ALFB + S(J-1))
        V(J,K) = -S(J)*(BETSQI*V(J,K) + V(J-1,K))
 1    CONTINUE
C     FOR BACKWARD SWEEP, DEFINE NEW INDEX, JB = NX+2-J.
      NXP2 = NX + 2
      DO 2 JB = 2,NX
        J = NXP2 - JB
        V(J,K) = V(J,K) - S(J)*V(J+1,K)
 2    CONTINUE
      RETURN
      END
```

APPENDIX B

FORTRAN LISTING OF EXAMPLE PROGRAM AND SUBROUTINES FOR FIRST REVISED CAUCHY-RIEMANN SOLVER, VERSION B

```
C       MAIN PROGRAM,UBIC1B76 FOR CDC 7600
C+++    SOLUTION OF CAUCHY-RIEMANN EQS. FOR INCOMPR. FLOW OVER THIN BICONV
C       AIRFOIL USING VERSION B OF CAUCHY-RIEMANN SOLVER.
C       AIRFOIL CHORD IS FROM X=-.5 TO +.5.
C
C       THE METHOD AND THE PROGRAM ARE DESCRIBED IN NASA TN D-7934
C       BY E. D. MARTIN AND H. LOMAX.
C
        DIMENSION U(129,129),V(129,130),VT(129)
        COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
C
        XJU(JU) = XL + HX*FLOAT(JU-1)
        YKU(KU) = YL + HY*(FLOAT(KU)-1.5)
        XJV(JV) = XL + HX*(FLOAT(JV)-1.5)
        YKV(KV) = YL + HY*FLOAT(KV-1)
        RHO(X,Y) =  SQRT(((X+.5)**2 + Y*Y)/((X-.5)**2 + Y*Y))
        ANPHI(X,Y) = ATAN2(Y,X-.5) - ATAN2(Y,X+.5)
        UEX(X,Y) = FCTR*(1. - X*ALOG(RHO(X,Y)) - Y*ANPHI(X,Y))
        VEX(X,Y) = FCTR*(Y*ALOG(RHO(X,Y))- X*ANPHI(X,Y))
C
        FCTR = 4./3.14159265
  1     FORMAT(4I5,4F10.0,2I5)
  2     READ(5,1) NEX,NX,NEY,NY,XL,XU,YL,YU,NBC,LASCAS
        CALL SECOND(TT1)
C       THE INPUT VALUES OF XL,XU, AND YU ARE NOMINAL VALUES, USED TO
C       COMPUTE THE EXACT VALUES.
        HY = (YU - YL)/FLOAT(NY)
        YU = YL + HY*(NY - .5)
        HX = (XU - XL)/FLOAT(NX)
        JLE = 1.005 + (-.5 - XL)/HX
        XL = -.5 - HX*(JLE - 1)
        XU = XL + HX*(NX - .5)
        JM = 1 + NX
        KM = 1 + NY
        NYP2 = NY + 2
        BETA = HY/HX
        BETSQ = BETA**2
C       CALCULATE V=VT(J) AT K=NY.
        Y = YKV(NY)
        DO 101 J = 2,NX
          X = XJV(J)
          IF (NBC.EQ.2) GO TO 102
          VT(J) = VEX(X,Y)
          GO TO 101
 102      VT(J) = 0.
 101    CONTINUE
C+++    INITIALIZE INTERIOR VALUES OF MASS SOURCES AND VORTICITY (F IS
C       EQUIV. TO V AND G IS EQUIV. TO U)--
        DO 3 K = 2,NY
          DO 4 J = 2,NX
            V(J,K) = VT(J)
            U(J,K) = 0.
 4        CONTINUE
 3      CONTINUE
C++++   LOAD BOUNDARY VALUES OF U AND V--
C       IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C       FROM AIRFOIL.
        K = KM
```

```
      DO 5 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1)= -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
 5    CONTINUE
      DO 6 K = 2,NY
        IF (NBC.EQ.2) GO TO 33
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
        GO TO 6
 33     U(1,K) = 0.
        V(JM,K) = 0.
 6    CONTINUE
C+++  MODIFY FRINGES OF INTERIOR TO INCLUDE BOUNDARY VALUES--
      DO 7 J = 2,NX
        V(J,2) = V(J,2) + V(J,1)
        U(J,NY) = U(J,NY) - U(J,KM)
 7    CONTINUE
      DO 8 K = 2,NY
        V(2,K) = V(2,K) + BETA*U(1,K)
        U(NX,K) = U(NX,K) + BETA*V(JM,K)
 8    CONTINUE
C+++  ZERO APPROPRIATE BOUNDARY VALUES--
      K = KM
      DO 9 J = 1,JM
        U(J,K) = 0.
        V(J,K) = 0.
        V(J,1) = 0.
 9    CONTINUE
      DO 10 K = 1,KM
        V(JM,K) = 0.
        U(1,K) = 0.
        V(1,K) = 0.
 10   CONTINUE
C+++  NEXT DETERMINE VECTOR  F0.
      DO 11 K = 2,NY
        DO 12 J = 2,NX
          V(J,K) = V(J,K) - V(J,K+1) + BETA*(U(J,K) - U(J-1,K))
 12     CONTINUE
 11   CONTINUE
      CALL CALCD(NEY)
      A = 2.*(1. + BETSQ)
      A1 = 2. + BETSQ
      TT2 = TT1
      T1 = SECOND(TT1) - TT2
      CALL CR2DUV
      TT2 = TT1
      T2 = SECOND(TT1) - TT2
C+++  NOW V IS DETERMINED FOR J = 2 TO NX AND K = 2 TO NY.
C     NEXT DETERMINE U FOR THE SAME J AND K.
      DO 15 KB = 2,NY
        K = NYP2 - KB
        DO 16 J = 2,NX
          U(J,K) = U(J,K+1) - U(J,K) + BETA*(V(J,K) - V(J+1,K))
 16     CONTINUE
 15   CONTINUE
```

```
      TT2 = TT1
      T3 = SECOND(TT1) - TT2
C+++  RELOAD BOUNDARY VALUES OF U AND V.
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      K = KM
      DO 17 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1 ) = -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
 17   CONTINUE
      DO 18 K = 2,NY
        IF (NBC.EQ.2) GO TO 34
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
        GO TO 18
 34     U(1,K) = 0.
        V(JM,K) = 0.
 18   CONTINUE
 19   FORMAT(1H1*SOLUTION OF CAUCHY-RIEMANN EQS. BY CYCLIC REDUCTION,
     1VERSION B.*/
     1* INCOMPRESSIBLE FLOW OVER THIN BICONVEX AIRFOIL. CHORD IS FROM X=
     2-.5 TO +.5.*//* NBC=*,I3,*. (IF NBC IS 1, EXACT BC'S ARE IMPOSED
     3FAR FROM AIRFOIL; IF NBC IS 2, U AND V ARE ZERO ON OUTER BOUNDARY.
     4*//
     5* NX=*,I3,*, NY=*,I3,*, XL=*,F10.6,*, XU=*,F10.6,*, YL=*,F10.6,
     6*, YU=*,F10.6,*, HX=*,E15.7,*, HY=*,E15.7,*.*//9X,1HJ,3X,1HK,6X,
     73HXJU,7X,3HYKU,8X,1HU,8X,3HUEX,8X,4HERRU,11X,3HXJV,7X,3HYKV,8X,1HV
     8,8X,3HVEX,8X,4HERRV)
      WRITE(6,19) NBC,NX,NY,XL,XU,YL,YU,HX,HY
 20   FORMAT(6X,2I4,F11.6,3F10.6,F15.7,F11.6,3F10.6,E15.7)
      KI = NY/16
      IF (KI.EQ.0) KI = 1
      IF (NX-2**NEX.EQ.0) GO TO 21
      JI = NX/20
      GO TO 22
 21   JI = NX/16
 22   IF (JI.EQ.0) JI = 1
 23   FORMAT(1X)
      DO 24 K = 1,KM,KI
        WRITE(6,23)
        YA = YKU(K)
        YB = YKV(K)
        DO 25 J = 1,JM,JI
          XA = XJU(J)
          XB = XJV(J)
          IF (J.EQ.JM.OR.K.EQ.1) GO TO 26
          UEXA = UEX(XA,YA)
          UA = U(J,K)
          GO TO 27
 26       UEXA = 0.
          UA = 0.
 27       IF (J.EQ.1) GO TO 28
          IF (K.NE.1) GO TO 35
          IF (ABS(XB*XB-.25).GE.1.E-5) GO TO 35
```

```
          VEXB = 0.
          GO TO 36
35        VEXB = VEX(XB,YB)
36        VB = V(J,K)
          GO TO 29
28        VEXB = 0.
          VB = 0.
29        ERRU = UA - UEXA
          ERRV = VB - VEXB
          WRITE(6,20) J,K,XA,YA,UA,UEXA,ERRU,XB,YB,VB,VEXB,ERRV
25      CONTINUE
24    CONTINUE
30    FORMAT(//* T1=*,E12.4,* SEC., T2=*,E12.4,* SEC., T3=*,E12.4,* SEC.
     1*)
      WRITE(6,30) T1,T2,T3
38    FORMAT(///* VELOCITIES AT Y = 0.*)
39    FORMAT(//26X,1HJ,6X,3HXJU,10X,1HU,13X,3HUEX,11X,3HXJV,10X,1HV,13X,
     13HVEX//)
40    FORMAT(23X,I4,F11.6,2E15.7,F11.6,2E15.7)
      WRITE(6,38)
      WRITE(6,39)
      DO 42 J = 2,NX
        XA = XJU(J)
        XB = XJV(J)
        UEXA = 0.
        VEXB = 0.
        IF (ABS(XA*XA-.25).GE.1.E-5) UEXA = UEX(XA,0.)
        IF (XB*XB.LE..25001) VEXB = -4.*XB
        VB = V(J,1)
        UA =        0.125*(9.*U(J,2)-U(J,3)+3.*BETA*(V(J,1)-V(J+1,1)))
        WRITE(6,40) J,XA,UA,UEXA,XB,VB,VEXB
42    CONTINUE
      WRITE(6,45)
      DELTX = .1*HX
      X = -.5 - 1.1*HX
      N = 0
46    N = N + 1
      IF (N.EQ.2) X = .5 - 1.1*HX
      WRITE(6,23)
      DO 47 I = 1,21
        X = X + DELTX
        UEX1 = 0.
        VEX1 = 0.
        IF (ABS(X*X-.25).GE.1.E-5) UEX1 = UEX(X,0.)
        IF (X*X.LE..25001) VEX1 = -4.*X
        WRITE(6,48) X,UEX1,VEX1
47    CONTINUE
      IF (N.EQ.1) GO TO 46
45    FORMAT(//* EXACT VELOCITIES AT EDGES.*//29X,1HX,11X,3HUEX,12X,3HVE
     1X)
48    FORMAT(23X,F10.6,2E15.7)
      IF (LASCAS.EQ.0) GO TO 2
      STOP
      END
```

```
      SUBROUTINE CALCD(NEY)
C     DIMENSIONS OF D(L,M) ARE LMAX=NEY-1 AND MMAX= 2**LMAX.
      DIMENSION D(6,64)
      COMMON/DC/D
      LMAX = NEY - 1
      D(1,1) = SQRT(2.)
      D(1,2) = -D(1,1)
      DO 30 L = 2,LMAX
        MMAX = 2**L
        MMAXH = MMAX/2
        MMAXH1 = MMAXH + 1
        DO 10 M = 1,MMAXH
          D(L,M) = SQRT(2.+ D(L-1,M))
 10     CONTINUE
        DO 20 M = MMAXH1,MMAX
          MM = M - MMAXH
          D(L,M) = -D(L,MM)
 20     CONTINUE
 30   CONTINUE
      RETURN
      END



      SUBROUTINE CR2DUV
C     ROUTINE TO SOLVE BY CYCLIC REDUCTION A BLOCK-TRIDIAGONAL EQUATION
C     FOR USE IN UBIC1B.
C
      DIMENSION V(129,130),D(6,64 ),DUM(129)
      COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V/DC/D
C
      JM = 1 + NX
      KM = 1 + NY
      LMAX = NEY - 1
C+++  FORWARD RECURSION--FIRST LEVEL IS L=1.
      KH = NY - 1
      DO 1 K = 3,KH,2
        DO 2 J = 2,NX
          V(J,K) = 2.*V(J,K)
 2      CONTINUE
        ALFA1 = A1
        ALFA = A
        CALL GE2N1
        DO 3 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) +      V(J,K+1)
 3      CONTINUE
 1    CONTINUE
      DO 4 L = 2,LMAX
        NH1 = 2**(L-2)
        NH2 = 2*NH1
        NH3 = 3*NH1
        NH4 = 4*NH1
        KL = NH4 + 1
        KH = KM - NH4
        DO 5 K = KL,KH,NH4
          K11 = K - NH1
          K12 = K + NH1
```

```
            K21 = K - NH2
            K22 = K + NH2
            K31 = K - NH3
            K32 = K + NH3
            DO 6 J = 2,NX
              DUM(J) = V(J,K)
              V(J,K) = 2.*V(J,K)-V(J,K11)+V(J,K21)-V(J,K31)
     1                      -V(J,K12)+V(J,K22)-V(J,K32)
 6          CONTINUE
            DO 7 M = 1,NH2
              ALFA1 = A1 + D(L-1,M)
              ALFA = A + D(L-1,M)
              CALL GE2N1
 7          CONTINUE
            DO 8 J = 2,NX
              V(J,K) = V(J,K) + DUM(J) - V(J,K11) + V(J,K21)         -V(J,
     1                 K12) + V(J,K22)
 8          CONTINUE
 5        CONTINUE
 4      CONTINUE
C
C+++    BACKWARD RECURSION-- DEFINE NEW INDEX, LB = LMAX-L+1 = NEY-L.
        DO 12 LB = 1,LMAX
          L = NEY - LB
          NH2 = 2**(L-1)
          NH4 = 2*NH2
          KL = NH4 + 1
          KH = KM - NH4
          KI = 2*NH4
          DO 13 K = KL,KH,KI
            K21 = K - NH2
            K22 = K + NH2
            K41 = K - NH4
            K42 = K + NH4
            DO 14 J = 2,NX
              DUM(J) = V(J,K)
              V(J,K) = V(J,K) + V(J,K41) + V(J,K42)
 14         CONTINUE
            DO 15 M = 1,NH4
              ALFA1 = A1 + D(L,M)
              ALFA = A    + D(L,M)
              CALL GE2N1
 15         CONTINUE
            DO 16 J = 2,NX
              V(J,K) = V(J,K) + .5*(DUM(J) - V(J,K21) - V(J,K22))
 16         CONTINUE
 13       CONTINUE
 12     CONTINUE
        DO 17 K = 2,NY,2
          DO 18 J = 2,NX
            V(J,K) = V(J,K) + V(J,K-1) + V(J,K+1)
 18       CONTINUE
          ALFA1 = A1
          ALFA = A
          CALL GE2N1
 17     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE GE2N1
C+++  ROUTINE TO SOLVE BY GAUSSIAN ELIMINATION THE TRIDIAG. EQ.,
C     T(-BETSQ,ALFA,-BETSQ)*V = F (WITH THE FIRST DIAG. ELEMENT REPLACED
C     BY ALFA1), WHERE V AND F ARE 2-DIMENSIONAL.
C     IN THE PROGRAM, V AND F ARE EQUIVALENT.
C
      DIMENSION V(129,130),S(129)
      COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
C
      ALFA1I = 1./ALFA1
      BETSQI = 1./BETSQ
      ALFB = ALFA/BETSQ
      S(2) = -ALFA1I*BETSQ
      V(2,K) = V(2,K)*ALFA1I
      DO 1 J = 3,NX
        S(J) = -1./(ALFB + S(J-1))
        V(J,K) = -S(J)*(BETSQI*V(J,K) + V(J-1,K))
 1    CONTINUE
C     FOR BACKWARD SWEEP, DEFINE NEW INDEX, JB = NX+2-J.
      NXP2 = NX + 2
      DO 2 JB = 2,NX
        J = NXP2 - JB
        V(J,K) = V(J,K) - S(J)*V(J+1,K)
 2    CONTINUE
      RETURN
      END
```

APPENDIX C

FORTRAN LISTING OF EXAMPLE PROGRAM AND SUBROUTINES FOR FIRST REVISED AND EXTENDED CAUCHY-RIEMANN SOLVER, VERSION C

```
C       MAIN PROGRAM, UBIC1C  FOR CDC 7600
C+++    SOLUTION OF CAUCHY-RIEMANN EQS. FOR INCOMPR. FLOW OVER THIN BICONV
C       AIRFOIL USING VERSION C OF CAUCHY-RIEMANN SOLVER (WITH EXTRA TERMS
C       AIRFOIL CHORD IS FROM X=-.5 TO +.5.
C
C       THE METHOD AND THE PROGRAM ARE DESCRIBED IN NASA TN D-7934
C       BY E. O. MARTIN AND H. LOMAX.
C
        DIMENSION U(129,129),V(129,130),VT(129)
        COMMON NFX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
        COMMON/COEF/GAMMA,DELTA,GD,DELTAI
C
        XJU(JU) = XL + HX*FLOAT(JU-1)
        YKU(KU) = YL + HY*(FLOAT(KU)-1.5)
        XJV(JV) = XL + HX*(FLOAT(JV)-1.5)
        YKV(KV) = YL + HY*FLOAT(KV-1)
        RHO(X,Y) =  SQRT(((X+.5)**2 + Y*Y)/((X-.5)**2 + Y*Y))
        ANPHI(X,Y) = ATAN2(Y,X-.5) - ATAN2(Y,X+.5)
        UEX(X,Y) = FCTR*(1. - X*ALOG(RHO(X,Y)) - Y*ANPHI(X,Y))
        VEX(X,Y) = FCTR*(Y*ALOG(RHO(X,Y))- X*ANPHI(X,Y))
C
        FCTR = 4./3.14159265
 1      FORMAT(4I5,4F10.0,I5)
 2      READ(5,1) NFX,NX,NEY,NY,XL,XU,YL,YU,NBC
 103    FORMAT(2E10.0,I5)
 104    READ(5,103) ALPHA1,ALPHA2,LASCAS
        CALL SECOND(TT1)
C       THE INPUT VALUES OF XL,XU, AND YU ARE NOMINAL VALUES, USED TO
C       COMPUTE THE EXACT VALUES.
        HY = (YU - YL)/FLOAT(NY)
        YU = YL + HY*(NY - .5)
        HX = (XU - XL)/FLOAT(NX)
        JLE = 1.005 + (-.5 - XL)/HX
        XL = -.5 - HX*(JLE - 1)
        XU = XL + HX*(NX - .5)
        JM = 1 + NX
        KM = 1 + NY
        NYP2 = NY + 2
        BETA = HY/HX
        BETSQ = BETA**2
        BETA1 = BETA*(1. - ALPHA1)
        BETA2 = BETA*(1. + ALPHA2)
        BETA3 =-BETA*ALPHA1
        BETA4 =-BETA*ALPHA2
        GAMMA = -BETA*BETA2
        DELTA = -BETA*BETA1
        GD = GAMMA/DELTA
        DELTAI = 1./DELTA
C       CALCULATE V=VT(J) AT K=NY.
        Y = YKV(NY)
        DO 101 J = 2,NX
          X = XJV(J)
          IF (NBC.EQ.2) GO TO 102
          VT(J) = VEX(X,Y)
          GO TO 101
 102    VT(J) = 0.
 101    CONTINUE
C+++    INITIALIZE INTERIOR VALUES OF MASS SOURCES AND VORTICITY (F IS
C       EQUIV. TO V AND G IS EQUIV. TO U)--
```

```
      DO 3 K = 2,NY
        Y = YKU(K)
        X1 = XL
        UEX1 = UEX(X1,Y)
        DO 4 J = 2,NX
          X = XJU(J)
          U(J,K) = 0.
          UEXX = UEX(X,Y)
          V(J,K) = VT(J) + BETA3*UEXX + BETA4*UEX1
          X1 = X
          UEX1 = UEXX
 4      CONTINUE
 3    CONTINUE
C++++ LOAD BOUNDARY VALUES OF U AND V--
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      K = KM
      DO 5 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1)= -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
 5    CONTINUE
      DO 6 K = 2,NY
        IF (NBC.EQ.2) GO TO 33
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
        GO TO 6
 33     U(1,K) = 0.
        V(JM,K) = 0.
 6    CONTINUE
C+++  MODIFY FRINGES OF INTERIOR TO INCLUDE BOUNDARY VALUES--
      DO 7 J = 2,NX
        V(J,2) = V(J,2) + V(J,1)
        U(J,NY) = U(J,NY) - U(J,KM)
 7    CONTINUE
      DO 8 K = 2,NY
        V(2,K) = V(2,K) + BETA2*U(1,K)
        U(NX,K) = U(NX,K) + BETA*V(JM,K)
 8    CONTINUE
C+++  ZERO APPROPRIATE BOUNDARY VALUES--
      K = KM
      DO 9 J = 1,JM
        U(J,K) = 0.
        V(J,K) = 0.
        V(J,1) = 0.
 9    CONTINUE
      DO 10 K = 1,KM
        V(JM,K) = 0.
        U(1,K) = 0.
        V(1,K) = 0.
 10   CONTINUE
C+++  NEXT DETERMINE VECTOR  FO.
      DO 11 K = 2,NY
        DO 12 J = 2,NX
          V(J,K) = V(J,K) - V(J,K+1) + BETA1*U(J,K) - BETA2*U(J-1,K)
 12     CONTINUE
```

```
 11   CONTINUE
      CALL CALCD(NEY)
      A1 = 2. + BETA*BETA1
      A = A1 + BETA*BETA2
      TT2 = TT1
      T1 = SECOND(TT1) - TT2
      CALL CR2UVC
      TT2 = TT1
      T2 = SECOND(TT1) - TT2
C+++  NOW V IS DETERMINED FOR J = 2 TO NX AND K = 2 TO NY.
C     NEXT DETERMINE U FOR THE SAME J AND K.
      DO 15 KB = 2,NY
        K = NYP2 - KB
        DO 16 J = 2,NX
          U(J,K) = U(J,K+1) - U(J,K) + BETA*(V(J,K) - V(J+1,K))
 16     CONTINUE
 15   CONTINUE
      TT2 = TT1
      T3 = SECOND(TT1) - TT2
C+++  RELOAD BOUNDARY VALUES OF U AND V.
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      K = KM
      DO 17 J = 2,NX
        V(J,1) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) V(J,1 ) = -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) V(J,1) = .5*V(J,1)
        IF (NBC.EQ.1) U(J,K) = UEX(XJU(J),YU)
        IF (NBC.EQ.2) U(J,K) = 0.
 17   CONTINUE
      DO 18 K = 2,NY
        IF (NBC.EQ.2) GO TO 34
        U(1,K) = UEX(XL,YKU(K))
        V(JM,K) = VEX(XU,YKV(K))
        GO TO 18
 34     U(1,K) = 0.
        V(JM,K) = 0.
 18   CONTINUE
 19   FORMAT(1H1*UBIC1C*//* SOLUTION OF CAUCHY-RIEMANN EQS. (WITH EXTRA
     1TERMS) BY CYCLIC REDUCTION,    VERSION C.*/
     1* INCOMPRESSIBLE FLOW OVER THIN BICONVEX AIRFOIL. CHORD IS FROM X=
     2-.5 TO +.5.*//* NBC=*,I3,*. (IF NBC IS 1, EXACT BC' S ARE IMPOSED
     3FAR FROM AIRFOIL; IF NBC IS 2, U AND V ARE ZERO ON OUTER BOUNDARY.
     4*//* ALPHA1=*,E15.7,*,  ALPHA2=*,E15.7,*.*//
     5* NX=*,I3,*, NY=*,I3,*, XL=*,F10.6,*, XU=*,F10.6,*, YL=*,F10.6,
     6*, YU=*,F10.6,*, HX=*,E15.7,*, HY=*,E15.7,*.*//9X,1HJ,3X,1HK,6X,
     73HXJU,7X,3HYKU,8X,1HU,8X,3HUEX,8X,4HERRU,11X,3HXJV,7X,3HYKV,8X,1HV
     8,8X,3HVEX,8X,4HERRV)
      WRITE(6,19) NBC,ALPHA1,ALPHA2,NX,NY,XL,XU,YL,YU,HX,HY
 20   FORMAT(6X,2I4,F11.6,3F10.6,E15.7,F11.6,3F10.6,E15.7)
      KI = NY/16
      IF (KI.EQ.0) KI = 1
      IF (NX-2**NEX.EQ.0) GO TO 21
      JI = NX/20
      GO TO 22
 21   JI = NX/16
 22   IF (JI.EQ.0) JI = 1
 23   FORMAT(1X)
```

```
      DO 24 K = 1,KM,KI
        WRITE(6,23)
        YA = YKU(K)
        YB = YKV(K)
        DO 25 J = 1,JM,JI
          XA = XJU(J)
          XB = XJV(J)
          IF (J.EQ.JM.OR.K.EQ.1) GO TO 26
          UEXA = UEX(XA,YA)
          UA = U(J,K)
          GO TO 27
26        UEXA = 0.
          UA = 0.
27        IF (J.EQ.1) GO TO 28
          IF (K.NE.1) GO TO 35
          IF (ABS(XB*XB-.25).GE.1.E-5) GO TO 35
          VEXB = 0.
          GO TO 36
35        VEXB = VEX(XB,YB)
36        VB = V(J,K)
          GO TO 29
28        VEXB = 0.
          VB = 0.
29        ERRU = UA - UEXA
          ERRV = VB - VEXB
          WRITE(6,20) J,K,XA,YA,UA,UEXA,ERRU,XB,YB,VB,VEXB,ERRV
25      CONTINUE
24    CONTINUE
30    FORMAT(//* T1=*,F12.4,* SEC., T2=*,E12.4,* SEC., T3=*,E12.4,* SEC.
     1*)
      WRITE(6,30) T1,T2,T3
38    FORMAT(///* VELOCITIES AT Y = 0.*)
39    FORMAT(//26X,1HJ,6X,3HXJU,10X,1HU,13X,3HUEX,11X,3HXJV,10X,1HV,13X,
     13HVEX//)
40    FORMAT(23X,I4,F11.6,2E15.7,F11.6,2E15.7)
      WRITE(6,38)
      WRITE(6,39)
      DO 42 J = 2,NX
        XA = XJU(J)
        XB = XJV(J)
        UEXA = 0.
        VEXB = 0.
        IF (ABS(XA*XA-.25).GE.1.E-5) UEXA = UEX(XA,0.)
        IF (XB*XB.LE..25001) VEXB = -4.*XB
        VB = V(J,1)
        UA =        0.125*(9.*U(J,2)-U(J,3)+3.*BETA*(V(J,1)-V(J+1,1)))
        WRITE(6,40) J,XA,UA,UEXA,XB,VB,VEXB
42    CONTINUE
      WRITE(6,45)
      DELTX = .1*HX
      X = -.5 - 1.1*HX
      N = 0
46    N = N + 1
      IF (N.EQ.2) X = .5 - 1.1*HX
      WRITE(6,23)
      DO 47 I = 1,21
        X = X + DELTX
        UEX1 = 0.
```

```
        VEX1 = 0.
        IF (ABS(X*X-.25).GE.1.E-5) UEX1 = UEX(X,0.)
        IF (X*X.LE..25001) VEX1 = -4.*X
        WRITE(6,48) X,UEX1,VEX1
 47   CONTINUE
      IF (N.EQ.1) GO TO 46
 45   FORMAT(//* EXACT VELOCITIES AT EDGES.*//29X,1HX,11X,3HUEX,12X,3HVE
     1X)
 48   FORMAT(23X,F10.6,2E15.7)
      IF (LASCAS.EQ.0) GO TO 2
      STOP
      END
```

```
      SUBROUTINE CALCD(NEY)
C     DIMENSIONS OF D(L,M) ARE LMAX=NEY-1 AND MMAX= 2**LMAX.
      DIMENSION D(6,64)
      COMMON/DC/D
      LMAX = NEY - 1
      D(1,1) = SQRT(2.)
      D(1,2) = -D(1,1)
      DO 30 L = 2,LMAX
        MMAX = 2**L
        MMAXH = MMAX/2
        MMAXH1 = MMAXH + 1
        DO 10 M = 1,MMAXH
          D(L,M) = SQRT(2.+ D(L-1,M))
 10     CONTINUE
        DO 20 M = MMAXH1,MMAX
          MM = M - MMAXH
          D(L,M) = -D(L,MM)
 20     CONTINUE
 30   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CR2UVC
C     ROUTINE TO SOLVE BY CYCLIC REDUCTION A BLOCK-TRIDIAGONAL EQUATION
C     FOR USE IN UBIC1C.
C
      DIMENSION V(129,130),D(6,64 ),DUM(129)
      COMMON NEX,NX,NEY,NY,RETSQ,A,A1,ALFA,ALFA1,K,V/DC/D
C
      JM = 1 + NX
      KM = 1 + NY
      LMAX = NEY - 1
C+++  FORWARD RECURSION--FIRST LEVEL IS L=1.
      KH = NY - 1
      DO 1 K = 3,KH,2
```

```
      DO 2 J = 2,NX
        V(J,K) = 2.*V(J,K)
2     CONTINUE
      ALFA1 = A1
      ALFA = A
      CALL GE2N2
      DO 3 J = 2,NX
        V(J,K) = V(J,K) + V(J,K-1) +      V(J,K+1)
3     CONTINUE
1   CONTINUE
    DO 4 L = 2,LMAX
      NH1 = 2**(L-2)
      NH2 = 2*NH1
      NH3 = 3*NH1
      NH4 = 4*NH1
      KL = NH4 + 1
      KH = KM - NH4
      DO 5 K = KL,KH,NH4
        K11 = K - NH1
        K12 = K + NH1
        K21 = K - NH2
        K22 = K + NH2
        K31 = K - NH3
        K32 = K + NH3
        DO 6 J = 2,NX
          DUM(J) = V(J,K)
          V(J,K) = 2.*V(J,K)-V(J,K11)+V(J,K21)-V(J,K31)
     1                      -V(J,K12)+V(J,K22)-V(J,K32)
6       CONTINUE
        DO 7 M = 1,NH2
          ALFA1 = A1 + D(L-1,M)
          ALFA = A + D(L-1,M)
          CALL GE2N2
7       CONTINUE
        DO 8 J = 2,NX
          V(J,K) = V(J,K) + DUM(J) - V(J,K11) + V(J,K21)        -V(J,
     1               K12) + V(J,K22)
8       CONTINUE
5     CONTINUE
4   CONTINUE
C
C+++  BACKWARD RECURSION-- DEFINE NEW INDEX, LB = LMAX-L+1 = NEY-L.
    DO 12 LB = 1,LMAX
      L = NEY - LB
      NH2 = 2**(L-1)
      NH4 = 2*NH2
      KL = NH4 + 1
      KH = KM - NH4
      KI = 2*NH4
      DO 13 K = KL,KH,KI
        K21 = K - NH2
        K22 = K + NH2
        K41 = K - NH4
        K42 = K + NH4
        DO 14 J = 2,NX
          DUM(J) = V(J,K)
          V(J,K) = V(J,K) + V(J,K41) + V(J,K42)
14      CONTINUE
```

```
            DO 15 M = 1,NH4
              ALFA1 = A1 + D(L,M)
              ALFA = A   + D(L,M)
              CALL GE2N2
 15         CONTINUE
            DO 16 J = 2,NX
              V(J,K) = V(J,K) + .5*(DUM(J) - V(J,K21) - V(J,K22))
 16         CONTINUE
 13       CONTINUE
 12     CONTINUE
        DO 17 K = 2,NY,2
          DO 18 J = 2,NX
            V(J,K) = V(J,K) + V(J,K-1) + V(J,K+1)
 18       CONTINUE
          ALFA1 = A1
          ALFA = A
          CALL GE2N2
 17     CONTINUE
        RETURN
        END
```

```
        SUBROUTINE GE2N2
C+++    ROUTINE TO SOLVE BY GAUSSIAN ELIMINATION THE TRIDIAG. EQ.,
C       T(GAMMA,ALFA,DELTA)*V=F (BUT WITH THE FIRST DIAG.ELEMENT REPLACED
C       BY ALFA1), WHERE V AND F ARE 2-DIMENSIONAL.
C       IN THE PROGRAM, V AND F ARE EQUIVALENT.
C
        DIMENSION V(129,130),S(129)
        COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
        COMMON/COEF/GAMMA,DELTA,GD,DELTAI
C
        ALFA1I = 1./ALFA1
        ALFD = ALFA*DELTAI
        S(2) = ALFA1I*DELTA
        V(2,K) = V(2,K)*ALFA1I
        DO 1 J = 3,NX
          S(J) = 1./(ALFD - GD*S(J-1))
          V(J,K) = S(J)*(DELTAI*V(J,K)- GD*V(J-1,K))
 1      CONTINUE
C       FOR BACKWARD SWEEP, DEFINE NEW INDEX, JB = NX+2-J.
        NXP2 = NX + 2
        DO 2 JB = 2,NX
          J = NXP2 - JB
          V(J,K) = V(J,K) - S(J)*V(J+1,K)
 2      CONTINUE
        RETURN
        END
```

APPENDIX D

FORTRAN LISTING OF EXAMPLE PROGRAM AND SUBROUTINES FOR SECOND REVISED AND EXTENDED CAUCHY-RIEMANN SOLVER, VERSION D

```
C      MAIN PROGRAM, UBICID  FOR CDC 7600
C+++   SOLUTION OF CAUCHY-RIEMANN EQS. FOR INCOMPR. FLOW OVER THIN BICONV
C      AIRFOIL USING VERSION D OF CAUCHY-RIEMANN SOLVER (WITH EXTRA TERMS
C      CONTAINING CONSTANT COEFFS.).
C      AIRFOIL CHORD IS FROM X=-.5 TO +.5.
C
C      THE METHOD AND THE PROGRAM ARE DESCRIBED IN NASA TN D-7934
C      BY E. D. MARTIN AND H. LOMAX.
C
       DIMENSION U(129,129),V(129,130),VT(129),VB(129),UL(129),VR(129)
       DIMENSION FKM(129),UT(129)
       COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
       COMMON/COEF/GAMMA,DELTA,GD,DELTAI
C
       XJU(JU) = XL + HX*FLOAT(JU-1)
       YKU(KU) = YL + HY*(FLOAT(KU)-1.5)
       XJV(JV) = XL + HX*(FLOAT(JV)-1.5)
       YKV(KV) = YL + HY*FLOAT(KV-1)
       RHO(X,Y) =  SQRT(((X+.5)**2 + Y*Y)/((X-.5)**2 + Y*Y))
       ANPHI(X,Y) = ATAN2(Y,X-.5) - ATAN2(Y,X+.5)
       UEX(X,Y) = FCTR*(1. - X*ALOG(RHO(X,Y)) - Y*ANPHI(X,Y))
       VEX(X,Y) = FCTR*(Y*ALOG(RHO(X,Y))- X*ANPHI(X,Y))
C
       FCTR = 4./3.14159265
 1     FORMAT(4I5,4E10.0,2I5)
 2     READ(5,1) NEX,NX,NEY,NY,XL,XU,YL,YU,NBC,NUBDRY
 103   FORMAT(2E10.0,I5)
 104   READ(5,103) ALPHA1,ALPHA2,LASCAS
       CALL SECOND(TT1)
C      THE INPUT VALUES OF XL AND XU ARE NOMINAL VALUES,USED TO
C      COMPUTE THE EXACT VALUES.
       HX = (XU - XL)/FLOAT(NX)
       JLE = 1.005 + (-.5 - XL)/HX
       XL = -.5 - HX*(JLE - 1)
       XU = XL + HX*(NX - .5)
       HY = (YU - YL)/FLOAT(NY)
       JM = 1 + NX
       KM = 1 + NY
       NYP2 = NY + 2
       BETA = HY/HX
       BETSQ = BETA**2
       BETA1 = BETA*(1. - ALPHA1)
       BETA2 = BETA*(1. + ALPHA2)
       BETA3 =-BETA*ALPHA1
       BETA4 =-BETA*ALPHA2
       GAMMA = -BETA*BETA2
       DELTA = -BETA*BETA1
       GD = GAMMA/DELTA
       DELTAI = 1./DELTA
       BETA1I = 1./BETA1
C+++   INITIALIZE INTERIOR VALUES OF MASS SOURCES AND VORTICITY (F IS
C      EQUIV. TO V EXCEPT AT K = KM AND G IS EQUIV. TO U)--
       DO 3 K = 2,NY
         Y = YKU(K)
         X1 = XL
         UEX1 = UEX(X1,Y)
         DO 4 J = 2,NX
           X = XJU(J)
           UEXX = UEX(X,Y)
```

```
          U(J,K) = 0.
          V(J,K) =    BETA3*UEXX      + BETA4*UEX1
          X1 = X
          UEX1 = UEXX
 4      CONTINUE
 3    CONTINUE
      Y = YKU(KM)
      X1 = XL
      UEX1 =UEX(X1,Y)
      DO 105 J = 2,NX
        X = XJU(J)
        UEXX = UEX(X,Y)
        FKM(J) = BETA3*UEXX      + BETA4*UEX1
        U(J,KM) = 0.
        X1 = X
        UEX1 = UEXX
 105  CONTINUE
C++++ CALC BOUNDARY VALUES OF U AND V--
C     IF NBC IS 1, CALC EXACT U AND V; IF NBC IS 2, CALC U=0 AND V=0 FAR
C     FROM AIRFOIL.
      DO 5 J = 2,JM
        VB(J) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) VB(J)= -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) VB(J) = .5*VB(J)
        IF (NBC.EQ.1) VT(J) = VEX(X,YU)
        IF (NBC.EQ.2) VT(J) = 0.
 5    CONTINUE
      IF (NUBDRY.EQ.0) GO TO 116
      Y = YKU(KM)
      IF (NUBDRY.GT.1) Y = YU
      DO 117 J = 2,NX
        IF (NBC.EQ.2) GO TO 118
        X = XJU(J)
        UT(J) = UEX(X,Y)
        GO TO 117
 118    UT(J) = 0.
 117  CONTINUE
 116  CONTINUE
      DO 6 K = 2,KM
        IF (NBC.EQ.2) GO TO 33
        UL(K) = UEX(XL,YKU(K))
        VR(K) = VEX(XU,YKV(K))
        GO TO 6
 33     UL(K) = 0.
        VR(K) = 0.
 6    CONTINUE
C+++  MODIFY FRINGES OF INTERIOR TO INCLUDE BOUNDARY VALUES--
      DO 7 J = 2,NX
        V(J,2) = V(J,2) + VB(J)
        FKM(J) = FKM(J) - VT(J)
        V(J,KM) = FKM(J)
 7    CONTINUE
      DO 8 K = 2,NY
        V(2,K) = V(2,K) + BETA2*UL(K)
        U(NX,K) = U(NX,K) + BETA*VR(K)
 8    CONTINUE
      FKM(2) = FKM(2) + BETA2*UL(KM)
      V(2,KM) = FKM(2)
```

```
C+++  ZERO APPROPRIATE BOUNDARY VALUES ON LEFT AND RIGHT--
      DO 10 K = 1,KM
        V(JM,K) = 0.
        U(1,K) = 0.
        V(1,K) = 0.
 10   CONTINUE
C+++  NEXT DETERMINE VECTOR  FO (EQUIV. TO V).
      DO 11 K = 2,NY
        DO 12 J = 2,NX
          V(J,K) = V(J,K) - V(J,K+1) + BETA1*U(J,K) - BETA2*U(J-1,K)
 12     CONTINUE
 11   CONTINUE
C+++  ZERO APPROPRIATE BOUNDARY VALUES ON TOP AND BOTTOM.
      DO 9 J = 1,JM
        V(J,KM) = 0.
        V(J,1) = 0.
 9    CONTINUE
      CALL CALCD(NEY)
      A1 = 2. + BETA*BETA1
      A = A1 + BETA*BETA2
      TT2 = TT1
      T1 = SECOND(TT1) - TT2
      CALL CR2UVC
      TT2 = TT1
      T2 = SECOND(TT1) - TT2
C+++  NOW V IS DETERMINED FOR J = 2 TO NX AND K = 2 TO NY.
C     NEXT DETERMINE U FOR ALL J AND K.
      IF (NUBDRY.NE.0) GO TO 119
      DO 106 J = 2,NX
        U(J,KM) = BETA1I*(BETA2*U(J-1,KM) + V(J,NY) + FKM(J))
 106  CONTINUE
      GO TO 120
 119  GO TO (121,122,123) NUBDRY
 121  DO 124 J = 2,NX
        U(J,KM) = UT(J)
 124  CONTINUE
      GO TO 120
 122  BETA5 = .5*BETA
      DO 125 J = 2,NX
        U(J,KM) = UT(J) + BETA5*(VT(J) - VT(J+1))
 125  CONTINUE
      GO TO 120
 123  BETA6 = .125*BETA
      BETA7 = 3.*BETA6
      V(JM,NY) = VR(NY)
      DO 126 J = 2,NX
        U(J,KM) = UT(J)+BETA7*(VT(J)-VT(J+1))+BETA6*(V(J,NY)-V(J+1,NY))
 126  CONTINUE
      V(JM,NY) = 0.
 120  CONTINUE
      DO 15 KB = 2,NY
        K = NYP2 - KB
        DO 16 J = 2,NX
          U(J,K) = U(J,K+1) - U(J,K) + BETA*(V(J,K) - V(J+1,K))
 16     CONTINUE
 15   CONTINUE
      TT2 = TT1
      T3 = SECOND(TT1) - TT2
```

```
C+++  RELOAD BOUNDARY VALUES OF U AND V.
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      DO 17 J = 2,NX
        V(J,1) = VB(J)
        V(J,KM) = VT(J)
 17   CONTINUE
      DO 18 K = 2,KM
        U(1,K) = UL(K)
        V(JM,K) = VR(K)
 18   CONTINUE
 19   FORMAT(1H1*UBIC1D*//* SOLUTION OF CAUCHY-RIEMANN EQS. (WITH EXTRA
     1TERMS) BY CYCLIC REDUCTION,    VERSION D.*/
     1* INCOMPRESSIBLE FLOW OVER THIN BICONVEX AIRFOIL. CHORD IS FROM X=
     2-.5 TO +.5.*//* NBC=*,I3,*.  IF NBC IS 1, EXACT BC' S ARE IMPOSED
     3FAR FROM AIRFOIL; IF NBC IS 2, U AND V ARE ZERO ON OUTER BOUNDARY.
     4*//* NUBDRY=*,I3,*.  ROW OF U(J,K) JUST INSIDE UPPER BOUNDARY,I.E.
     4, U(J,KM), IS COMPUTED DIFFERENTLY ACCORDING TO SPECIFIED NUBDRY.*
     4/* IF NUBDRY= 0, U(J,KM) IS COMPUTED FROM CONTINUITY EQ.,(UNSTABLE
     4 CALCULATION IF BETA2/BETA1 IS GT 1.0)*
     4/*           = 1, U(J,KM) IS COMPUTED FROM U SPECIFIED AT Y=YKU(KM)
     4, DETERMINED FROM SAME RELATION THAT DETERMINES OUTER CONDS.*
     4/*           = 2, U(J,KM) IS COMPUTED USING 1ST-ORDER-ACC. ONE-SIDE
     4D Y-DERIV. IN ROTAT. EQ. AT Y = YU,    WITH U SPECIFIED THERE.*
     4/*           = 3, U(J,KM) IS COMPUTED USING 2ND-ORDER-ACC. ONE-SIDE
     4D Y-DERIV. IN ROTAT. EQ. AT Y = YU,  WITH U SPECIFIED THERE.
     4*//* ALPHA1=*,E15.7,*,  ALPHA2=*,E15.7,*.*//
     5* NX=*,I3,*, NY=*,I3,*, XL=*,F10.6,*, XU=*,F10.6,*, YL=*,F10.6,
     6*, YU=*,F10.6,*, HX=*,E15.7,*, HY=*,E15.7,*.*//9X,1HJ,3X,1HK,6X,
     73HXJU,7X,3HYKU,8X,1HU,8X,3HUEX,8X,4HERRU,11X,3HXJV,7X,3HYKV,8X,1HV
     8,8X,3HVEX,8X,4HERRV)
      WRITE(6,19) NBC,NUBDRY,ALPHA1,ALPHA2,NX,NY,XL,XU,YL,YU,HX,HY
 20   FORMAT(6X,2I4,F11.6,3F10.6,F15.7,F11.6,3F10.6,E15.7)
      KI = NY/16
      IF (KI.EQ.0) KI = 1
      IF (NX-2**NEX.EQ.0) GO TO 21
      JI = NX/20
      GO TO 22
 21   JI = NX/16
 22   IF (JI.EQ.0) JI = 1
 23   FORMAT(1X)
      DO 24 K = 1,KM,KI
        WRITE(6,23)
        YA = YKU(K)
        YB = YKV(K)
        DO 25 J = 1,JM,JI
          XA = XJU(J)
          XB = XJV(J)
          IF (J.EQ.JM.OR.K.EQ.1) GO TO 26
          UEXA = UEX(XA,YA)
          UA = U(J,K)
          GO TO 27
 26       UEXA = 0.
          UA = 0.
 27       IF (J.EQ.1) GO TO 28
          IF (K.NE.1) GO TO 35
          IF (ABS(X*X-.25).GE.1.E-5) GO TO 35
          VEXB = 0.
          GO TO 36
```

```
35        VEXB = VEX(XB,YB)
36        VB1 = V(J,K)
          GO TO 29
28        VEXB = 0.
          VB1= 0.
29        ERRU = UA - UEXA
          ERRV = VB1 - VEXB
          WRITE(6,20) J,K,XA,YA,UA,UEXA,ERRU,XB,YB,VB1,VEXB,ERRV
25      CONTINUE
24    CONTINUE
30    FORMAT(//* T1=*,F12.4,* SEC., T2=*,E12.4,* SEC., T3=*,E12.4,* SEC.
     1*)
      WRITE(6,30) T1,T2,T3
38    FORMAT(///* VELOCITIES AT Y = 0.*)
39    FORMAT(//26X,1HJ,6X,3HXJU,10X,1HU,13X,3HUEX,11X,3HXJV,10X,1HV,13X,
     13HVEX//)
40    FORMAT(23X,I4,F11.6,2F15.7,F11.6,2F15.7)
      WRITE(6,38)
      WRITE(6,39)
      DO 42 J = 2,NX
        XA = XJU(J)
        XB = XJV(J)
        UEXA = 0.
        VEXB = 0.
        IF (ABS(XA*XA-.25).GE.1.E-5) UEXA = UEX(XA,0.)
        IF (XB*XB.LE..25001) VEXB = -4.*XB
        VB1 = V(J,1)
        UA =        0.125*(9.*U(J,2)-U(J,3)+3.*BETA*(V(J,1)-V(J+1,1)))
        WRITE(6,40) J,XA,UA,UEXA,XB,VB1,VEXB
42    CONTINUE
      WRITE(6,45)
      DELTX = .1*HX
      X = -.5 - 1.1*HX
      N = 0
46    N = N + 1
      IF (N.EQ.2) X = .5 - 1.1*HX
      WRITE(6,23)
      DO 47 I = 1,21
        X = X + DELTX
        UEX1 = 0.
        VEX1 = 0.
        IF (ABS(X*X-.25).GE.1.E-5) UEX1 = UEX(X,0.)
        IF (X*X.LE..25001) VEX1 = -4.*X
        WRITE(6,48) X,UEX1,VEX1
47    CONTINUE
      IF (N.EQ.1) GO TO 46
45    FORMAT(//* EXACT VELOCITIES AT EDGES.*//29X,1HX,11X,3HUEX,12X,3HVE
     1X)
48    FORMAT(23X,F10.6,2E15.7)
      IF (LASCAS.EQ.0) GO TO 2
      STOP
      END
```

```
      SUBROUTINE CALCD(NEY)
C     DIMENSIONS OF D(L,M) ARE LMAX=NEY-1 AND MMAX= 2**LMAX.
      DIMENSION D(6,64)
      COMMON/DC/D
      LMAX = NEY - 1
      D(1,1) = SQRT(2.)
      D(1,2) = -D(1,1)
      DO 30 L = 2,LMAX
        MMAX = 2**L
        MMAXH = MMAX/2
        MMAXH1 = MMAXH + 1
        DO 10 M = 1,MMAXH
          D(L,M) = SQRT(2.+ D(L-1,M))
 10     CONTINUE
        DO 20 M = MMAXH1,MMAX
          MM = M - MMAXH
          D(L,M) = -D(L,MM)
 20     CONTINUE
 30   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CR2UVC
C     ROUTINE TO SOLVE BY CYCLIC REDUCTION A BLOCK-TRIDIAGONAL EQUATION
C     FOR USE IN UBIC1C.
C
      DIMENSION V(129,130),D(6,64 ),DUM(129)
      COMMON NEX,NX,NEY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V/DC/D
C
      JM = 1 + NX
      KM = 1 + NY
      LMAX = NEY - 1
      IF (NY.EQ.2) GO TO 20
C+++  FORWARD RECURSION--FIRST LEVEL IS L=1.
      KH = NY - 1
      DO 1 K = 3,KH,2
        DO 2 J = 2,NX
          V(J,K) = 2.*V(J,K)
 2      CONTINUE
        ALFA1 = A1
        ALFA = A
        CALL GE2N2
        DO 3 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) +      V(J,K+1)
 3      CONTINUE
 1    CONTINUE
      IF (NY.EQ.4) GO TO 21
      DO 4 L = 2,LMAX
        NH1 = 2**(L-2)
        NH2 = 2*NH1
        NH3 = 3*NH1
        NH4 = 4*NH1
```

```
      KL = NH4 + 1
      KH = KM - NH4
      DO 5 K = KL,KH,NH4
        K11 = K - NH1
        K12 = K + NH1
        K21 = K - NH2
        K22 = K + NH2
        K31 = K - NH3
        K32 = K + NH3
        DO 6 J = 2,NX
          DUM(J) = V(J,K)
          V(J,K) = 2.*V(J,K)-V(J,K11)+V(J,K21)-V(J,K31)
     1                      -V(J,K12)+V(J,K22)-V(J,K32)
    6   CONTINUE
        DO 7 M = 1,NH2
          ALFA1 = A1 + D(L-1,M)
          ALFA = A + D(L-1,M)
          CALL GF2N2
    7   CONTINUE
        DO 8 J = 2,NX
          V(J,K) = V(J,K) + DUM(J) - V(J,K11) + V(J,K21)        -V(J,
     1               K12) + V(J,K22)
    8   CONTINUE
    5 CONTINUE
    4 CONTINUE
   21 CONTINUE
C
C  *  BACKWARD RECURSION-- DEFINE NEW INDEX, LB = LMAX-L+1 = NEY-L.
      DO 12 LB = 1,LMAX
        L = NEY - LB
        NH2 = 2**(L-1)
        NH4 = 2*NH2
        KL = NH4 + 1
        KH = KM - NH4
        KI = 2*NH4
        DO 13 K = KL,KH,KI
          K21 = K - NH2
          K22 = K + NH2
          K41 = K - NH4
          K42 = K + NH4
          DO 14 J = 2,NX
            DUM(J) = V(J,K)
            V(J,K) = V(J,K) + V(J,K41) + V(J,K42)
   14     CONTINUE
          DO 15 M = 1,NH4
            ALFA1 = A1 + D(L,M)
            ALFA = A   + D(L,M)
            CALL GF2N2
   15     CONTINUE
          DO 16 J = 2,NX
            V(J,K) = V(J,K) + .5*(DUM(J) - V(J,K21) - V(J,K22))
   16     CONTINUE
   13   CONTINUE
   12 CONTINUE
   20 CONTINUE
      DO 17 K = 2,NY,2
        DO 18 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) + V(J,K+1)
   18   CONTINUE
```

```
        ALFA1 = A1
        ALFA = A
        CALL GE2N2
 17   CONTINUE
      RETURN
      END



      SUBROUTINE GE2N2
C+++  ROUTINE TO SOLVE BY GAUSSIAN ELIMINATION THE TRIDIAG. EQ.,
C     T(GAMMA,ALFA,DELTA)*V=F (BUT WITH THE FIRST DIAG.ELEMENT REPLACED
C     BY ALFA1), WHERE V AND F ARE 2-DIMENSIONAL.
C     IN THE PROGRAM, V AND F ARE EQUIVALENT.
C
      DIMENSION V(129,130),S(129)
      COMMON NFX,NX,NFY,NY,BETSQ,A,A1,ALFA,ALFA1,K,V
      COMMON/COEF/GAMMA,DELTA,GD,DELTAI
C
      ALFA1I = 1./ALFA1
      ALFD = ALFA*DELTAI
      S(2) = ALFA1I*DELTA
      V(2,K) = V(2,K)*ALFA1I
      DO 1 J = 3,NX
        S(J) = 1./(ALFD - GD*S(J-1))
        V(J,K) = S(J)*(DELTAI*V(J,K)- GD*V(J-1,K))
 1    CONTINUE
C     FOR BACKWARD SWEEP, DEFINE NEW INDEX, JB = NX+2-J.
      NXP2 = NX + 2
      DO 2 JB = 2,NX
        J = NXP2 - JB
        V(J,K) = V(J,K) - S(J)*V(J+1,K)
 2    CONTINUE
      RETURN
      END
```

APPENDIX E

FORTRAN LISTING OF EXAMPLE PROGRAM AND SUBROUTINES FOR SECOND REVISED AND EXTENDED CAUCHY-RIEMANN SOLVER (VARIABLE COEFFICIENTS), VERSION E

```
C       MAIN PROGRAM, UBICIE  FOR CDC 7600
C+++    SOLUTION OF CAUCHY-RIEMANN EQS. FOR INCOMPR. FLOW OVER THIN BICONV
C       AIRFOIL USING VERSION E OF CAUCHY-RIEMANN SOLVER (WITH EXTRA TERMS
C       CONTAINING VARIABLE COEFFS.).
C       AIRFOIL CHORD IS FROM X=-.5 TO +.5.
C
C       THE METHOD AND THE PROGRAM ARE DESCRIBED IN NASA TN D-7934
C       BY E. O. MARTIN AND H. LOMAX.
C
        DIMENSION U(129,129),V(129,130),VT(129),VB(129),UL(129),VR(129)
        DIMENSION UT(129),ALF(129),BET(129),GAM(129)
        DIMENSION BETA1(129),BETA2(129),BETA3(129),BETA4(129)
        COMMON NEX,NX,NEY,NY,NXP2,K,V
        COMMON/COEF/N,M,ALF,BET,GAM
C
        XJU(JU) = XL + HX*FLOAT(JU-1)
        YKU(KU) = YL + HY*(FLOAT(KU)-1.5)
        XJV(JV) = XL + HX*(FLOAT(JV)-1.5)
        YKV(KV) = YL + HY*FLOAT(KV-1)
        RHO(X,Y) =  SQRT(((X+.5)**2 + Y*Y)/((X-.5)**2 + Y*Y))
        ANPHI(X,Y) = ATAN2(Y,X-.5) - ATAN2(Y,X+.5)
        UEX(X,Y) = FCTR*(1. - X*ALOG(RHO(X,Y)) - Y*ANPHI(X,Y))
        VEX(X,Y) = FCTR*(Y*ALOG(RHO(X,Y))- X*ANPHI(X,Y))
C
        F1(JU) = XJU(JU) - .5
        F2(JU) = XJU(JU) + .5
C
        FCTR = 4./3.14159265
  1     FORMAT(4I5,4E10.0,3I5)
  2     READ(5,1) NEX,NX,NEY,NY,XL,XU,YL,YU,NBC,NUBDRY,LASCAS
        CALL SECOND(TT1)
C       THE INPUT VALUES OF XL AND XU ARE NOMINAL VALUES,USED TO
C       COMPUTE THE EXACT VALUES.
        HX = (XU - XL)/FLOAT(NX)
        JLE = 1.005 + (-.5 - XL)/HX
        XL = -.5 - HX*(JLE - 1)
        XU = XL + HX*(NX - .5)
        HY = (YU - YL)/FLOAT(NY)
        JM = 1 + NX
        KM = 1 + NY
        NXP2 = 2 + NX
        NYP2 = 2 + NY
        BETA = HY/HX
        BETSQ = BETA**2
        ALPHA2 = -1.0
        DO 127 J = 2,NX
          ALPHA1 = F1(J)
          BETA4(J) = -BETA*ALPHA2
          BETA3(J) = -BETA*ALPHA1
          BETA2(J) = BETA - BETA4(J)
          BETA1(J) = BETA + BETA3(J)
          ALF(J) = BETA*BETA2(J)
          GAM(J) = BETA*BETA1(J)
          BET(J) = 2. + ALF(J) + GAM(J)
          ALPHA2 = F2(J)
 127    CONTINUE
C+++    INITIALIZE INTERIOR VALUES OF MASS SOURCES AND VORTICITY (F IS
C       EQUIV. TO V EXCEPT AT K = KM AND G IS EQUIV. TO U)--
```

```
      DO 3 K = 2,KM
        Y = YKU(K)
        X1 = XL
        UEX1 = UEX(X1,Y)
        DO 4 J = 2,NX
          X = XJU(J)
          UEXX = UEX(X,Y)
          U(J,K) = 0.
          V(J,K) =   BETA3(J)*UEXX + BETA4(J)*UEX1
          X1 = X
          UEX1 = UEXX
 4      CONTINUE
 3    CONTINUE
C++++ CALC BOUNDARY VALUES OF U AND V--
C     IF NBC IS 1, CALC EXACT U AND V; IF NBC IS 2, CALC U=0 AND V=0 FAR
C     FROM AIRFOIL.
      DO 5 J = 2,JM
        VB(J) = 0.
        X = XJV(J)
        IF (X*X.LE..25001) VB(J)= -4.*X
        IF (ABS(X*X-.25).LT.1.E-5) VB(J) = .5*VB(J)
        IF (NBC.EQ.1) VT(J) = VEX(X,YU)
        IF (NBC.EQ.2) VT(J) = 0.
 5    CONTINUE
      Y = YKU(KM)
      IF (NUBDRY.GT.1) Y = YU
      DO 117 J = 2,NX
        IF (NBC.EQ.2) GO TO 118
        X = XJU(J)
        UT(J) = UEX(X,Y)
        GO TO 117
 118    UT(J) = 0.
 117  CONTINUE
 116  CONTINUE
      DO 6 K = 2,KM
        IF (NBC.EQ.2) GO TO 33
        UL(K) = UEX(XL,YKU(K))
        VR(K) = VEX(XU,YKV(K))
        GO TO 6
 33     UL(K) = 0.
        VR(K) = 0.
 6    CONTINUE
C+++  MODIFY FRINGES OF INTERIOR TO INCLUDE BOUNDARY VALUES--
      DO 7 J = 2,NX
        V(J,2) = V(J,2) + VB(J)
        V(J,KM) = V(J,KM) - VT(J)
 7    CONTINUE
      DO 8 K = 2,KM
        V(2,K) = V(2,K) + BETA2(2)*UL(K)
        U(NX,K) = U(NX,K) + BETA*VR(K)
 8    CONTINUE
C+++  ZERO APPROPRIATE BOUNDARY VALUES ON LEFT AND RIGHT--
      DO 10 K = 1,KM
        V(JM,K) = 0.
        U(1,K) = 0.
        V(1,K) = 0.
 10   CONTINUE
C+++  NEXT DETERMINE VECTOR  FO (EQUIV. TO V).
```

```
      DO 11 K = 2,NY
        DO 12 J = 2,NX
          V(J,K) = V(J,K) - V(J,K+1) + BETA1(J)*U(J,K)-BETA2(J)*U(J-1,K)
 12     CONTINUE
 11   CONTINUE
C+++  ZERO APPROPRIATE BOUNDARY VALUES ON TOP AND BOTTOM.
      DO 9 J = 1,JM
        V(J,KM) = 0.
        V(J,1) = 0.
 9    CONTINUE
      CALL CALCD(NEY)
      TT2 = TT1
      T1 = SECOND(TT1) - TT2
      CALL CR2UVE
      TT2 = TT1
      T2 = SECOND(TT1) - TT2
C+++  NOW V IS DETERMINED FOR J = 2 TO NX AND K = 2 TO NY.
C     NEXT DETERMINE U FOR ALL J AND K.
 119  GO TO (121,122,123) NUBDRY
 121  DO 124 J = 2,NX
        U(J,KM) = UT(J)
 124  CONTINUE
      GO TO 120
 122  BETA5 = .5*BETA
      DO 125 J = 2,NX
        U(J,KM) = UT(J) + BETA5*(VT(J) - VT(J+1))
 125  CONTINUE
      GO TO 120
 123  BETA6 = .125*BETA
      BETA7 = 3.*BETA6
      V(JM,NY) = VR(NY)
      DO 126 J = 2,NX
        U(J,KM) = UT(J)+BETA7*(VT(J)-VT(J+1))+BETA6*(V(J,NY)-V(J+1,NY))
 126  CONTINUE
      V(JM,NY) = 0.
 120  CONTINUE
      DO 15 KB = 2,NY
        K = NYP2 - KB
        DO 16 J = 2,NX
          U(J,K) = U(J,K+1) - U(J,K) + BETA*(V(J,K) - V(J+1,K))
 16     CONTINUE
 15   CONTINUE
      TT2 = TT1
      T3 = SECOND(TT1) - TT2
C+++  RELOAD BOUNDARY VALUES OF U AND V.
C     IF NBC IS 1, LOAD EXACT U AND V; IF NBC IS 2, LOAD U=0 AND V=0 FAR
C     FROM AIRFOIL.
      DO 17 J = 2,NX
        V(J,1) = VB(J)
        V(J,KM) = VT(J)
 17   CONTINUE
      DO 18 K = 2,KM
        U(1,K) = UL(K)
        V(JM,K) = VR(K)
 18   CONTINUE
 19   FORMAT(1H1*UBIC1F*//* SOLUTION OF CAUCHY-RIEMANN EQS. (WITH EXTRA
     1TERMS AND VARIABLE COEFFS.) BY CYCLIC REDUCTION,    VERSION F.*/
     1* INCOMPRESSIBLE FLOW OVER THIN BICONVEX AIRFOIL. CHORD IS FROM X=
```

```
     2-.5 TO +.5.*//* NBC=*,I3,*.  IF NBC IS 1, EXACT BC' S ARE IMPOSED
     3FAR FROM AIRFOIL; IF NBC IS 2, U AND V ARE ZERO ON OUTER BOUNDARY.
     4*//* NUBDRY=*,I3,*.  ROW OF U(J,K) JUST INSIDE UPPER BOUNDARY, I.E.
     4, U(J,KM), IS COMPUTED DIFFERENTLY ACCORDING TO SPECIFIED NUBDRY.*
     4/* IF NUBDRY= 1, U(J,KM) IS COMPUTED FROM U SPECIFIED AT Y=YKU(KM)
     4, DETERMINED FROM SAME RELATION THAT DETERMINES OUTER CONDS.*
     4/*          = 2, U(J,KM) IS COMPUTED USING 1ST-ORDER-ACC. ONE-SIDE
     4D Y-DERIV. IN ROTAT. EQ. AT Y = YU,    WITH U SPECIFIED THERE.*
     4/*          = 3, U(J,KM) IS COMPUTED USING 2ND-ORDER-ACC. ONE-SIDE
     4D Y-DERIV. IN ROTAT. EQ. AT Y = YU,  WITH U SPECIFIED THERE.*//
     5* NX=*,I3,*, NY=*,I3,*, XL=*,F10.6,*, XU=*,F10.6,*, YL=*,F10.6,
     6*, YU=*,F10.6,*, HX=*,E15.7,*, HY=*,E15.7,*.*//9X,1HJ,3X,1HK,6X,
     73HXJU,7X,3HYKU,8X,1HU,9X,3HUEX,8X,4HERRU,11X,3HXJV,7X,3HYKV,8X,1HV
     8,8X,3HVEX,8X,4HERRV)
      WRITE(6,19) NBC,NUBDRY,NX,NY,XL,XU,YL,YU,HX,HY
20    FORMAT(6X,2I4,F11.6,3F10.6,E15.7,F11.6,3F10.6,E15.7)
      KI = NY/16
      IF (KI.EQ.0) KI = 1
      IF (NX-2**NEX.EQ.0) GO TO 21
      JI = NX/20
      GO TO 22
21    JI = NX/16
22    IF (JI.EQ.0) JI = 1
23    FORMAT(1X)
      DO 24 K = 1,KM,KI
        WRITE(6,23)
        YA = YKU(K)
        YB = YKV(K)
        DO 25 J = 1,JM,JI
          XA = XJU(J)
          XB = XJV(J)
          IF (J.EQ.JM.OR.K.EQ.1) GO TO 26
          UEXA = UEX(XA,YA)
          UA = U(J,K)
          GO TO 27
26        UEXA = 0.
          UA = 0.
27        IF (J.EQ.1) GO TO 28
          IF (K.NE.1) GO TO 35
          IF (ABS(X*X-.25).GE.1.E-5) GO TO 35
          VEXB = 0.
          GO TO 36
35        VEXB = VEX(XB,YB)
36        VB1 = V(J,K)
          GO TO 29
28        VEXB = 0.
          VB1 = 0.
29        ERRU = UA - UEXA
          ERRV = VB1 - VEXB
          WRITE(6,20) J,K,XA,YA,UA,UEXA,ERRU,XB,YB,VB1,VEXB,ERRV
25      CONTINUE
24    CONTINUE
30    FORMAT(//* T1=*,F12.4,* SEC., T2=*,E12.4,* SEC., T3=*,E12.4,* SEC.
     1*)
      WRITE(6,30) T1,T2,T3
38    FORMAT(///* VELOCITIES AT Y = 0.*)
39    FORMAT(//26X,1HJ,6X,3HXJU,10X,1HU,13X,3HUEX,11X,3HXJV,10X,1HV,13X,
     13HVEX//)
40    FORMAT(23X,I4,F11.6,2E15.7,F11.6,2E15.7)
```

```
      WRITE(6,38)
      WRITE(6,39)
      DO 42 J = 2,NX
        XA = XJU(J)
        XB = XJV(J)
        UEXA = 0.
        VEXB = 0.
        IF (ABS(XA*XA-.25).GE.1.E-5) UEXA = UEX(XA,0.)
        IF (XB*XB.LE..25001) VEXB = -4.*XB
        VB1 = V(J,1)
        UA =        0.125*(9.*U(J,2)-U(J,3)+3.*BETA*(V(J,1)-V(J+1,1)))
        WRITE(6,40) J,XA,UA,UEXA,XB,VB1,VEXB
 42   CONTINUE
      WRITE(6,45)
      DELTX = .1*HX
      X = -.5 - 1.1*HX
      N = 0
 46   N = N + 1
      IF (N.EQ.2) X = .5 - 1.1*HX
      WRITE(6,23)
      DO 47 I = 1,21
        X = X + DELTX
        UEX1 = 0.
        VEX1 = 0.
        IF (ABS(X*X-.25).GE.1.E-5) UEX1 = UEX(X,0.)
        IF (X*X.LE..25001) VEX1 = -4.*X
        WRITE(6,48) X,UEX1,VEX1
 47   CONTINUE
      IF (N.EQ.1) GO TO 46
 45   FORMAT(//* EXACT VELOCITIES AT EDGES.*//29X,1HX,11X,3HUEX,12X,3HVE
     1X)
 48   FORMAT(23X,F10.6,2E15.7)
      IF (LASCAS.EQ.0) GO TO 2
      STOP
      END
```

```
      SUBROUTINE CALCD(NEY)
C     DIMENSIONS OF D(L,M) ARE LMAX=NEY-1 AND MMAX= 2**LMAX.
      DIMENSION D(6,64)
      COMMON/DC/D
      LMAX = NEY - 1
      D(1,1) = SQRT(2.)
      D(1,2) = -D(1,1)
      DO 30 L = 2,LMAX
        MMAX = 2**L
        MMAXH = MMAX/2
        MMAXH1 = MMAXH + 1
        DO 10 M = 1,MMAXH
          D(L,M) = SQRT(2.+ D(L-1,M))
 10     CONTINUE
        DO 20 M = MMAXH1,MMAX
          MM = M - MMAXH
          D(L,M) = -D(L,MM)
 20     CONTINUE
```

```
 30   CONTINUE
      RETURN
      END



      SUBROUTINE CR2UVF
C     ROUTINE TO SOLVE BY CYCLIC REDUCTION A BLOCK-TRIDIAGONAL
C     MATRIX EQUATION  WITH VARIABLE COEFFS. FOR USE IN UBICLE.
C
      DIMENSION V(129,130),ALF(129),BET(129),GAM(129),DUM(129)
      COMMON NEX,NX,NEY,NY,NXP2,K,V
      COMMON/COEF/N,M,ALF,BET,GAM
C
      JM = 1+NX
      KM = 1+NY
      L2MAX = NEY-1
C
C     FORWARD RECURSION--FIRST LEVEL IS L2=1
C
      KH = NY-1
      DO 5 K = 3,KH,2
        DO 6 J = 2,NX
          V(J,K) = 2.*V(J,K)
 6      CONTINUE
        N = 0
        M = 1
        CALL GEUVF
        DO 7 J = 2,NX
          V(J,K) = V(J,K) + V(J,K-1) + V(J,K+1)
 7      CONTINUE
 5    CONTINUE
      DO 8 L2 = 2,L2MAX
        N = L2 - 1
        NH1 = 2**(L2-2)
        NH2 = 2*NH1
        NH3 = 3*NH1
        NH4 = 4*NH1
        KL = NH4 + 1
        KH = NY - NH4 +1
        DO 9 K = KL,KH,NH4
          K11 = K-NH1
          K12 = K+NH1
          K21 = K-NH2
          K22 = K+NH2
          K31 = K-NH3
          K32 = K+NH3
          DO 10 J = 2,NX
            DUM(J) = V(J,K)
            V(J,K) = 2.*V(J,K)-V(J,K11)-V(J,K12)+V(J,K21)+V(J,K22)
     1               -V(J,K31)-V(J,K32)
 10       CONTINUE
          DO 11 M = 1,NH2
            CALL GEUVF
 11       CONTINUE
```

```
            DO 12 J = 2,NX
              V(J,K) = V(J,K)+DUM(J)-V(J,K11)-V(J,K12)+V(J,K21)+V(J,K22)
 12         CONTINUE
 9        CONTINUE
 8      CONTINUE
C
C      BACKWARD RECURSION-- DEFINE NEW INDEX, L2B=L2MAX-L2+1=NEY-L2.
C
       DO 21 L2B = 1,L2MAX
         L2 = NEY-L2B
         N = L2
         NH2 = 2**(L2-1)
         NH4 = 2*NH2
         KL = NH4+1
         KH = KM-NH4
         KI = 2*NH4
         DO 22 K = KL,KH,KI
           K21 = K-NH2
           K22 = K+NH2
           K41 = K-NH4
           K42 = K+NH4
           DO 23 J = 2,NX
             DUM(J) = V(J,K)
             V(J,K) = V(J,K41) + V(J,K42) + V(J,K)
 23        CONTINUE
           DO 24 M = 1,NH4
             CALL GEUVE
 24        CONTINUE
           DO 25 J = 2,NX
             V(J,K) = V(J,K) + .5*(DUM(J)-V(J,K21)-V(J,K22))
 25        CONTINUE
 22      CONTINUE
 21    CONTINUE
       DO 26 K = 2,NY,2
         DO 27 J = 2,NX
           V(J,K) = V(J,K-1) + V(J,K+1) + V(J,K)
 27      CONTINUE
         N = 0
         M = 1
         CALL GEUVE
 26    CONTINUE
       RETURN
       END




       SUBROUTINE GEUVE
C      ROUTINE TO SOLVE BY GAUSSIAN ELIMINATION THE TRIDIAG. EQ.,
C      T(-ALF(J),BETP(J),-GAM(J))*V = F, WHERE V AND F ARE 2-D.
C
       DIMENSION V(129,130),ALF(129),BET(129),GAM(129),
      1          S(129),D(6,64)
       COMMON NEX,NX,NEY,NY,NXP2,K,V
       COMMON/DC/D /COEF/N,M,ALF,BET,GAM
C
       S(1) = 0.
       IF (N.EQ.0) GO TO 3
```

```
      DO 1 J = 2,NX
        BETP = BET(J) + D(N,M)
        T = 1./(BETP + ALF(J)*S(J-1))
        S(J) = -GAM(J)*T
        V(J,K) = T*(V(J,K) + ALF(J)*V(J-1,K))
 1    CONTINUE
      GO TO 4
 3    DO 5 J = 2,NX
        BETP = BET(J)
        T = 1./(BETP + ALF(J)*S(J-1))
        S(J) = -GAM(J)*T
        V(J,K) = T*(V(J,K) + ALF(J)*V(J-1,K))
 5    CONTINUE
C     FOR BACKWARD SWEEP, DEFINE NEW INDEX, JB = NX+2-J.
 4    DO 2 JB = 2,NX
        J = NXP2 - JB
        V(J,K) = V(J,K) - S(J)*V(J+1,K)
 2    CONTINUE
      RETURN
      END
```

APPENDIX F

CYCLIC REDUCTION FOR VERSIONS B TO E OF CAUCHY-RIEMANN SOLVER

A detailed algorithm is outlined here for the recursive cyclic reduction of the block-tridiagonal matrix equation in the form of equations (50) and (59) for use in the FORTRAN programs for versions B to E of the extended Cauchy-Riemann solvers. The algorithm is a variant of Buneman's (ref. 2) double cyclic reduction in which scalar-tridiagonal-equation solutions are obtained by the Thomas algorithm (Gaussian elimination). Recursive cyclic reduction was devised by Prof. G. Golub with collaboration of Dr. R. Hockney (see refs. 1 to 3). The method is based on odd/even reduction, which was used extensively by Hockney (ref. 1) in direct two-dimensional Poisson solvers and was the basis for the extension by Buneman (ref. 2) to his double cyclic reduction algorithm for solving Poisson's equation. Refer to reference 5 for a treatise on the development of the method.

The block-tridiagonal matrix equation to be solved represents the system of n matrix equations:

$$-\mathbf{V}_{k-1} + \mathbf{C}_0\mathbf{V}_k - \mathbf{V}_{k+1} = \mathbf{p}_k^{(0)} \qquad (k = 1, 2, \ldots, n) \tag{F1}$$

where $n = n_1 - 1$, $n_1 = 2^L$ with L an integer, and in which each $\mathbf{p}_k^{(0)}$ is a known m-dimensional vector,

$$\mathbf{p}_k^{(0)} \equiv \mathbf{F}_k = \mathrm{col}[\mathbf{F}_{1,k}, \mathbf{F}_{2,k}, \ldots, \mathbf{F}_{m,k}] \tag{F2}$$

Each $\mathbf{V}_k$ is an m-dimensional vector to be determined:

$$\mathbf{V}_k = \mathrm{col}[v_{1,k}, v_{2,k}, \ldots, v_{m,k}] \tag{F3}$$

where, for simplicity, we have set, in equation (F1),

$$\mathbf{V}_0 = \mathbf{V}_{n_1} = 0 \tag{F4}$$

and $\mathbf{C}_0$ is the m-dimensional tridiagonal matrix $\mathbf{C}$ defined by equations (21c) and (21d):

$$\mathbf{C}_0 = \mathbf{T}_m(-\alpha_j, \beta_j, -\gamma_j) \tag{F5}$$

with α_j, β_j, and γ_j defined in terms of the parameter β and in terms of $\beta_{1,j}$ and $\beta_{2,j}$, which are specified for $j = 1$ to m in each version of the solver.

The cyclic-reduction algorithm depends on the matrix factorization defined as follows: each level of the recursion process is denoted by integer ℓ, which ranges from 1 to L-1. For each ℓ, define an integer N and quantities $d_{\ell,M}$ as follows:

$$N = 2^{\ell} \qquad (\ell = 1, 2, \ldots, L - 1) \tag{F6}$$

$$d_{\ell,1} = \sqrt{2}, \qquad d_{\ell,2} = -d_{\ell,1} \qquad (\ell = 1) \tag{F7a}$$

$$d_{\ell,M} = \sqrt{2 + d_{\ell-1,M}} \qquad \begin{pmatrix} \ell = 2, 3, \ldots, L - 1, \\ M = 1, 2, \ldots, \frac{1}{2} N \end{pmatrix} \tag{F7b}$$

$$d_{\ell,M} = -d_{\ell,M-\frac{1}{2}N} \qquad \begin{pmatrix} \ell = 2, 3, \ldots, L - 1, \\ M = \frac{1}{2} N + 1, \ldots, N - 1, N \end{pmatrix} \tag{F7c}$$

Then, given $\mathbf{C}_0$, for each ℓ there is a matrix $\mathbf{C}_\ell$ defined by

$$\mathbf{C}_\ell = \mathbf{C}_{\ell-1}^2 - 2\mathbf{I} \tag{F8}$$

which has N tridiagonal matrices as factors:

$$\mathbf{C}_\ell = \mathbf{C}_\ell^{(1)} \mathbf{C}_\ell^{(2)} \ldots \mathbf{C}_\ell^{(N)} \tag{F9}$$

in which the Mth factor is

$$\mathbf{C}_\ell^{(M)} = \mathbf{C} + d_{\ell,M}\mathbf{I} \qquad \begin{pmatrix} \ell = 1 \text{ to } L - 1, \\ M = 1 \text{ to } N \end{pmatrix} \tag{F10a}$$

$$= \mathbf{T}_m(-\alpha_j, \beta_j', -\gamma_j) \tag{F10b}$$

and

$$\beta_j' = \beta_j + d_{\ell,M} \tag{F10c}$$

where $-\alpha_j$, β_j, and $-\gamma_j$ are the elements of $\mathbf{C}_0$ in equations (F5). (Note that the values of $d_{\ell,M}$ can also be obtained from a cosine function (e.g., eq. (21b) in ref. 25), but are calculated in the sample FORTRAN programs listed in appendices A to E by use of equations (F7). Note also the difference in notation, where $\mathbf{C}_\ell$ here has the same role as $\mathbf{C}_N$ in ref. 25.)

For the first level of reduction ($\ell = 1$), multiply each of the even equations ($k = 2, 4, \ldots, n - 1$) in equations (F1) by $\mathbf{C}_0$ and add to it the adjacent equations above and below to obtain the reduced system

$$-\mathbf{V}_{k-2} + \mathbf{C}_1\mathbf{V}_k - \mathbf{V}_{k+2} = \mathbf{r}_k^{(1)} \qquad (k = 2, 4, \ldots, n-1) \tag{F11}$$

where

$$\mathbf{r}_k^{(0)} = \mathbf{p}_k^{(0)} \qquad (k = 1, 2, \ldots, n) \tag{F12a}$$

$$\mathbf{r}_k^{(1)} = \mathbf{C}_0\mathbf{r}_k^{(0)} + \mathbf{r}_{k-1}^{(0)} + \mathbf{r}_{k+1}^{(0)} \qquad (k = 2, 4, \ldots, n-1) \tag{F12b}$$

A special device was introduced by Buneman (ref. 2) for avoiding all matrix multiplications (and a consequent stability problem) in the final solution process. The device removes the matrix $\mathbf{C}_0$ from the right side of equation (F12b) and replaces it by $\mathbf{C}_1$, so that the implied matrix multiplication need never be performed. To use this device, since we wish to remove $\mathbf{C}_0$ and insert $\mathbf{C}_1$, and since $\mathbf{C}_1$ contains $\mathbf{C}_0^2$, we write equation (F12b) as

$$\begin{aligned} r_k^{(1)} &= \mathbf{C}_0^2\left[\mathbf{C}_0^{-1}\mathbf{r}_k^{(0)}\right] + \mathbf{r}_{k-1}^{(0)} + \mathbf{r}_{k+1}^{(0)} \\ &= (\mathbf{C}_0^2 - 2\mathbf{I})\left[\mathbf{C}_0^{-1}\mathbf{r}_k^{(0)}\right] + 2\mathbf{I}\left[\mathbf{C}_0^{-1}\mathbf{r}_k^{(0)}\right] + \mathbf{r}_{k-1}^{(0)} + \mathbf{r}_{k+1}^{(0)} \end{aligned}$$

Now *define* $\mathbf{q}_k^{(1)}$ and $\mathbf{p}_k^{(1)}$ by denoting the first term by $\mathbf{C}_1\mathbf{q}_k^{(1)}$ and the remainder by $\mathbf{p}_k^{(1)}$. Thus

$$\mathbf{r}_k^{(1)} \equiv \mathbf{C}_1\mathbf{q}_k^{(1)} + \mathbf{p}_k^{(1)} \tag{F13a}$$

where

$$\mathbf{q}_k^{(1)} = \mathbf{C}_0^{-1}\mathbf{p}_k^{(0)} \tag{F13b}$$

$$\mathbf{p}_k^{(1)} = 2\mathbf{q}_k^{(1)} + \mathbf{p}_{k-1}^{(0)} + \mathbf{p}_{k+1}^{(0)} \tag{F13c}$$

Note that equation (F13b) is just a tridiagonal matrix equation that can be solved for $\mathbf{q}_k^{(1)}$, and then $\mathbf{p}_k^{(1)}$ is obtained from equation (F13c). Also, substitution of equation (F13b) into (F13c) gives an equation that can be solved for $\mathbf{p}_k^{(1)}$ without knowing $\mathbf{q}_k^{(1)}$:

$$\mathbf{p}_k^{(1)} = \mathbf{C}_0^{-1}\left[2\mathbf{p}_k^{(0)}\right] + \mathbf{p}_{k-1}^{(0)} + \mathbf{p}_{k+1}^{(0)} \tag{F13d}$$

For the second and higher levels of reduction ($\ell = 2, 3, \ldots, L-1$), we let

$$h = (2)^{\ell-2} = N/4 \tag{F14}$$

and in the remaining reduced set of equations we multiply each of the even equations ($k = N, 2N, 3N, \ldots, n_1 - N$) by $\mathbf{C}_{\ell-1}$ and add to it the adjacent equations above and below to obtain, for $\ell = 2$ to $L - 1$:

$$-\mathbf{V}_{k-N} + \mathbf{C}_{\ell}\mathbf{V}_{k} - \mathbf{V}_{k+N} = \mathbf{r}_{k}^{(\ell)} \qquad (k = N, 2N, 3N, \ldots, n_1 - N) \tag{F15}$$

where

$$\mathbf{r}_{k}^{(\ell)} = \mathbf{C}_{\ell-1}\mathbf{r}_{k}^{(\ell-1)} + \mathbf{r}_{k-2h}^{(\ell-1)} + \mathbf{r}_{k+2h}^{(\ell-1)} \tag{F16}$$

As in equations (F13) above, we wish to define $\mathbf{q}_{k}^{(\ell)}$ and $\mathbf{p}_{k}^{(\ell)}$ so that

$$\mathbf{r}_{k}^{(\ell)} \equiv \mathbf{C}_{\ell}\mathbf{q}_{k}^{(\ell)} + \mathbf{p}_{k}^{(\ell)} \tag{F17a}$$

and this can be done by first substituting equation (F17a) (treating $\mathbf{r}_{k}^{(\ell)}$ as a function of its superscript ℓ and its subscript k) into *each* term on the right side of equation (F16) and then using a procedure directly analogous to that used to obtain equation (F13) from (F12) to eliminate $\mathbf{C}_{\ell-1}$ in favor of $\mathbf{C}_{\ell}$. We then have equation (F17a) in place of (F16), with the results

$$\mathbf{q}_{k}^{(\ell)} = \mathbf{q}_{k}^{(\ell-1)} + \left(\mathbf{C}_{\ell-1}\right)^{-1}\left[\mathbf{p}_{k}^{(\ell-1)} + \mathbf{q}_{k-2h}^{(\ell-1)} + \mathbf{q}_{k+2h}^{(\ell-1)}\right] \tag{F17b}$$

$$\mathbf{p}_{k}^{(\ell)} = 2\mathbf{q}_{k}^{(\ell)} + \mathbf{p}_{k-2h}^{(\ell-1)} + \mathbf{p}_{k+2h}^{(\ell-1)} \tag{F17c}$$

Furthermore, one can obtain (analogous to eq. (F13d)) an equation to solve for $\mathbf{p}_{k}^{(\ell)}$ without knowing any of the $\mathbf{q}_{k}^{(\ell)}$. If equation (F17c) is solved for $\mathbf{q}_{k}^{(\ell)}$ in terms of functions $\mathbf{p}_{k}^{(\ell)}$, and the resulting $\mathbf{q}_{k}^{(\ell)}$ is then treated as a function of its subscript k and superscript ℓ to obtain expressions for $\mathbf{q}_{k}^{(\ell-1)}$, $\mathbf{q}_{k-2h}^{(\ell-1)}$, and $\mathbf{q}_{k+2h}^{(\ell-1)}$ to substitute into (F17b), one obtains

$$\begin{aligned}\mathbf{p}_{k}^{(\ell)} = {} & \mathbf{p}_{k-2h}^{(\ell-1)} + \mathbf{p}_{k+2h}^{(\ell-1)} + \mathbf{p}_{k}^{(\ell-1)} - \mathbf{p}_{k-h}^{(\ell-2)} - \mathbf{p}_{k+h}^{(\ell-2)} \\ & + \left(\mathbf{C}_{\ell-1}\right)^{-1}\left[2\mathbf{p}_{k}^{(\ell-1)} - \mathbf{p}_{k-h}^{(\ell-2)} - \mathbf{p}_{k+h}^{(\ell-2)} + \mathbf{p}_{k-2h}^{(\ell-1)} + \mathbf{p}_{k+2h}^{(\ell-1)}\right. \\ & \left. - \mathbf{p}_{k-3h}^{(\ell-2)} - \mathbf{p}_{k+3h}^{(\ell-2)}\right]\end{aligned} \tag{F17d}$$

The procedure for the algorithm to solve equations (F1), to determine all $v_{j,k}$, uses the equations derived above as follows:

1. First, for integer $L = \log_2 n_1$, compute the array $d_{\ell,M}$ for $\ell = 1$ to L-1 and $M = 1$ to N using equations (F6) and (F7).

2. For $\ell = 1$, start the forward recursion with $\mathbf{p}_k^{(0)}$ given by equation (F2) for $k = 1$ to n. Use the Thomas algorithm (Gaussian elimination) to solve equation (F13d) in the form

$$\mathbf{C}_0 \Theta_k^{(1)} = 2\mathbf{p}_k^{(0)} \tag{F18a}$$

successively for $k = 2, 4, \ldots, n-1$, where $\mathbf{C}_0$ is the tridiagonal matrix defined in equation (F5) and where $\Theta_k^{(1)}$ is defined by

$$\mathbf{p}_k^{(1)} \equiv \Theta_k^{(1)} + \mathbf{p}_{k-1}^{(0)} + \mathbf{p}_{k+1}^{(0)} \tag{F18b}$$

After solving equation (F18a) for each unknown vector $\Theta_k^{(1)}$, then determine $\mathbf{p}_k^{(1)}$ from (F18b).

3. For each $\ell = 2, 3, \ldots, L-1$, first write equation (F17d) in the form

$$\begin{aligned}\mathbf{C}_{\ell-1}\Theta_k^{(\ell)} = {} & 2\mathbf{p}_k^{(\ell-1)} - \mathbf{p}_{k-h}^{(\ell-2)} - \mathbf{p}_{k+h}^{(\ell-2)} + \mathbf{p}_{k-2h}^{(\ell-1)} + \mathbf{p}_{k+2h}^{(\ell-1)} \\ & - \mathbf{p}_{k-3h}^{(\ell-2)} - \mathbf{p}_{k+3h}^{(\ell-2)}\end{aligned} \tag{F19a}$$

for each $k = N, 2N, \ldots, n_1-N$, where $\Theta_k^{(\ell)}$ is defined by

$$\mathbf{p}_k^{(\ell)} \equiv \Theta_k^{(\ell)} + \mathbf{p}_k^{(\ell-1)} - \mathbf{p}_{k-h}^{(\ell-2)} - \mathbf{p}_{k+h}^{(\ell-2)} + \mathbf{p}_{k-2h}^{(\ell-1)} + \mathbf{p}_{k+2h}^{(\ell-1)} \tag{F19b}$$

Equation (F19a) is of the form (cf. eq. (F9)):

$$\mathbf{C}_{\ell-1}^{(1)}\left[\mathbf{C}_{\ell-1}^{(2)} \cdots \mathbf{C}_{\ell-1}^{(N)}\Theta_k^{(\ell)}\right] = \mathbf{z}_k^{(1)} \tag{F20a}$$

where $\mathbf{z}_k^{(1)}$ represents the total vector on the right side of equation (F19a) and each $\mathbf{C}_{\ell-1}^{(M)}$ is a tridiagonal matrix defined by (F10). Similarly, let the bracketed quantity on the left side of equation (F20a) be denoted by $\mathbf{z}_k^{(2)}$ so that

$$\mathbf{C}_{\ell-1}^{(2)}\left[\mathbf{C}_{\ell-1}^{(3)} \cdots \mathbf{C}_{\ell-1}^{(N)}\Theta_k^{(\ell)}\right] = \mathbf{z}_k^{(2)} \tag{F20b}$$

etc. Thus the procedure at this step is to first consider $\mathbf{z}_k^{(2)}$ as the unknown vector in equation (F20a), and use the Thomas algorithm to solve (F20a) for $\mathbf{z}_k^{(2)}$, then with that vector known, solve equation (F20b) for the bracketed quantity, and so forth, until $\Theta_k^{(\ell)}$ is known. Thus $\Theta_k^{(\ell)}$ has been obtained by a sequence of tridiagonal solutions, and then $\mathbf{p}_k^{(\ell)}$ is obtained

from equation (F19b). When this is done for all $k = N, 2N, \ldots, n_1-N$, increase ℓ and repeat this step 3.

4. After step 3 has been done for $\ell = L-1$, all the $\mathbf{p}_k^{(\ell)}$ are known that are required for the backward recursion (back substitution). The back substition proceeds successively, for each ℓ from $L-1$ to 0.
For $\ell = L-1, L-2, \ldots, 1$, use equations (F15) and (F17a) (with (F11) and (F13a) as well) in the form

$$\mathbf{C}_\ell \phi_k^{(\ell)} = \mathbf{V}_{k-N} + \mathbf{V}_{k+N} + \mathbf{p}_k^{(\ell)} \tag{F21a}$$

for $k = N, 3N, 5N, \ldots, n_1-N$, where $\phi_k^{(\ell)}$ is defined by

$$\mathbf{V}_k \equiv \phi_k^{(\ell)} + \mathbf{q}_k^{(\ell)} \tag{F21b}$$

or with equations (F17c) and (F13c),

$$\mathbf{V}_k = \phi_k^{(\ell)} + \frac{1}{2}\left[\mathbf{p}_k^{(\ell)} - \mathbf{p}_{k-2h}^{(\ell-1)} - \mathbf{p}_{k+2h}^{(\ell-1)}\right] \tag{F21c}$$

for $k = N, 3N, 5N, \ldots, n_1-N$. Thus, one can now solve equation (F21a), by use of the Thomas algorithm for a sequence of tridiagonal solutions in a manner analogous to that used in equations (F20), to obtain $\phi_k^{(\ell)}$, with which $\mathbf{V}_k$ is then obtained from (F21c) for $k = N, 3N, 5N, \ldots, n_1-N$.

5. For $\ell = 0$, equation (F1) can be written as

$$\mathbf{C}_0 \mathbf{V}_k = \mathbf{V}_{k-1} + \mathbf{V}_{k+1} + \mathbf{p}_k^{(0)} \tag{F22}$$

and solved using the Thomas algorithm to obtain $\mathbf{V}_k$ for $k = 1, 3, 5, \ldots, n$.

In a machine computation using this algorithm, the components (as needed) of each of the vectors $\mathbf{F}_k$, $\mathbf{p}_k^{(\ell)}$, and $\mathbf{V}_k$ can all occupy the same array in computer memory for one value of k. Therefore, in addition to a relatively small array for $d_{\ell,M}$ given by equations (F7), only one large array, $(m_1+1)\times(n_1+1)$, is needed $(v_{j,k})$ along with an m_1-dimensional dummy vector for the intermediate storage either of $\Theta_k^{(\ell)}$ in equations (F18) or of $\phi_k^{(\ell)}$ in equations (F21). In addition, the Thomas algorithm itself requires one dummy m_1 vector.

REFERENCES

1. Hockney, R. W.: A Fast Direct Solution of Poisson's Equation Using Fourier Analysis. J. Assoc. for Computing Machinery, vol. 12, Jan. 1965, pp. 95-113.

2. Buneman, O.: A Compact Non-Iterative Poisson Solver, SUIPR Rept. 294, Institute for Plasma Research, Stanford Univ., May 1969.

3. Hockney, R. W.: The Potential Calculation and Some Applications. Methods in Computational Physics, vol. 9, B. Alder, S. Fernbach, and M. Rotenberg, eds., Academic Press, 1970, pp. 135-211.

4. Dorr, F. W.: The Direct Solution of the Discrete Poisson Equation on a Rectangle. SIAM Rev., vol. 12, April 1970, pp. 248-263.

5. Buzbee, B. L.; Golub, G. H.; and Nielson, C. W.: On Direct Methods for Solving Poisson's Equations. SIAM J. Numer. Anal., vol. 7, Dec. 1970, pp. 627-656.

6. Martin, E. D.: Direct Numerical Solution of Three-Dimensional Equations Containing Elliptic Operators. Intern. J. Numer. Meth. in Engineering, vol. 6, 1973, pp. 201-212.

7. Roache, P. J.: A New Direct Method for the Discretized Poisson Equation. Proc. Second Intern. Conf. on Numerical Methods in Fluid Dynamics, Sept. 15-19, 1970, Univ. of Calif., Berkeley, Maurice Holt, ed., Lecture Notes in Physics, vol. 8, Springer-Verlag, Berlin, 1971, pp. 48-53.

8. Le Bail, R. C.: Use of Fast Fourier Transforms for Solving Partial Differential Equations in Physics. J. Comp. Phys., vol. 9, June 1972, pp. 440-465.

9. Sweet, R. A.: Direct Methods for the Solution of Poisson's Equation on a Staggered Grid. J. Comp. Phys., vol. 12, July 1973, pp. 422-428.

10. Schwarztrauber, P. N.; and Sweet, R. A.: The Direct Solution of the Discrete Poisson Equation on a Disk. SIAM J. Numer. Anal., vol. 10, Oct. 1973, pp. 900-907.

11. Schwarztrauber, P. N.; and Sweet, R. A.: Efficient Subroutines for the Solution of General Elliptic and Parabolic Partial Differential Equations. Atmospheric Technology, National Center for Atmospheric Research, Sept. 1973, pp. 79-81.

12. Fischer, D.; Golub, G.; Hald, O.; Leiva, C.; and Widlund, O.: On Fourier-Toeplitz Methods for Separable Elliptic Problems. Math. Comp., vol. 28, Apr. 1974, pp. 349-368.

13. Schwarztrauber, P. N.: The Direct Solution of the Discrete Poisson Equation on the Surface of a Sphere. J. Comp. Phys., vol. 15, May 1974, pp. 46-54.

14. Sweet, R. A.: A Generalized Cyclic Reduction Algorithm. SIAM J. Numer. Anal., vol. 11, June 1974, pp. 506-520.

15. Schwarztrauber, P. N.: A Direct Method for the Discrete Solution of Separable Elliptic Equations. SIAM J. Numer. Anal., vol. 11, Dec. 1974, pp. 1136-1150.

16. Schwarztrauber, P.; and Sweet, R.: Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations. NCAR-TN/IA-109, National Center for Atmospheric Research, Boulder, Colorado, July 1975.

17. Schumann, U.; and Sweet, R. A.: A Direct Method for the Solution of Poisson's Equation with Neumann Boundary Conditions on a Staggered Grid of Arbitrary Size. J. Comp. Phys., vol. 20, Feb. 1976, pp. 171-182.

18. Hockney, R. W.: Formation and Stability of Virtual Electrodes in a Cylinder. J. App. Phys., vol. 39, Aug. 1968, pp. 4166-4170.

19. George, J. A.: The Use of Direct Methods for the Solution of the Discrete Poisson Equation on Non-Rectangular Regions. Computer Science Dept., Rept. 159, Stanford Univ., 1970.

20. Buzbee, B. L.; Dorr, F. W.; George, J. A.; and Golub, G. H.: The Direct Solution of the Discrete Poisson Equation on Irregular Regions. SIAM J. Numer. Anal., vol. 8, Dec. 1971, pp. 722-736.

21. Martin, E. D.: A Generalized-Capacity-Matrix Technique for Computing Aerodynamic Flows. Presented at the Symposium on Application of Computers to Fluid Dynamics Analysis and Design, Polytechnic Institute of Brooklyn Graduate Center, Farmingdale, N.Y., Jan. 1973; also Computers and Fluids, vol. 2, March 1974, pp. 79-97.

22. Buzbee, B. L.; and Dorr, F. W.: The Direct Solution of the Biharmonic Equation on Rectangular Regions and the Poisson Equation on Irregular Regions. SIAM J. Numer. Anal., vol. 11, Sept. 1974, pp. 753-763.

23. Proskurowski, W.; and Widlund, O.: On the Numerical Solution of Helmholtz's Equation by the Capacitance Matrix Method. Math. Comp., vol. 30, July 1976, pp. 433-468.

24. Bauer, L; and Reiss, E. L.: Block Five Diagonal Matrices and the Fast Numerical Solution of the Biharmonic Equation. Math. Comp., vol. 26, April 1972, pp. 311-326.

25. Lomax, H.; and Martin, E. D.: Fast Direct Numerical Solution of the Nonhomogeneous Cauchy-Riemann Equations. J. Comp. Phys., vol. 15, May 1974, pp. 55-80.

26. von Rosenberg, D. U.: Methods for the Numerical Solution of Partial Differential Equations. American Elsevier Publ. Co., 1969.

27. Widlund, O. B.: On the Use of Fast Methods for Separable Finite-Difference Equations for the Solution of General Elliptic Problems. Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, 1972, pp. 121-134.

28. Concus, P.; and Golub, G. H.: Use of Fast Direct Methods for the Efficient Numerical Solution of Nonseparable Elliptic Equations. SIAM J. Numer. Anal., vol. 10, Dec. 1973, pp. 1103-1120.

29. Fromm, J. E.: A Numerical Study of Buoyancy Driven Flows in Room Enclosures. Proc. Second. Intern. Conf. on Numerical Methods in Fluid Dynamics, Sept. 15-19, 1970, Univ. of Calif., Berkeley, Maurice Holt, ed., Lecture Notes in Physics, vol. 8, Springer-Verlag (Berlin), 1971, pp. 120-126.

30. Lomax, H.; and Bailey, F. R.: Computational Method for Calculating Convection in a Rotating Tank. NASA TR R-386, 1972, Chap. 3, pp. 39-59.

31. Roache, P. J.: Computational Fluid Dynamics. (Revised printing), Hermosa Publ., Albuquerque, N. M., 1976.

32. Roache, P. J.: Finite-Difference Methods for the Steady-State Navier-Stokes Equations. Rept. SC-RR-73-0419, Sandia Labs., Albuquerque, N.M., Dec. 1972.

33. Roache, P. J.: The LAD, NOS, and Split NOS Methods for the Steady-State Navier-Stokes Equations. Computers and Fluids, vol. 3, June 1975, pp. 179-195.

34. Roache, P. J.; and Ellis, M. A.: The BID Method for the Steady-State Navier-Stokes Equations. Computers and Fluids, vol. 3, Dec. 1975, pp. 305-320.

35. Martin, E. D.; and Lomax, H.: Rapid Finite-Difference Computation of Subsonic and Slightly Supercritical Aerodynamic Flows. AIAA Paper 74-11, Jan. 1974; also AIAA J., vol. 13, May 1975, pp. 579-586.

36. Martin, E. D.: Progress in Application of Direct Elliptic Solvers to Transonic Flow Computations. Aerodynamic Analyses Requiring Advanced Computers, Part II. NASA SP-347, 1975, pp. 839-870.

37. Martin, E. D.: A Fast Semidirect Method for Computing Transonic Aerodynamic Flows. AIAA 2nd Computational Fluid Dynamics Conf. Proc., Hartford, Conn., June 1975, pp. 162-174; also AIAA J., vol. 14, July 1976, pp. 914-922; and Errata, AIAA J., vol. 14, Nov. 1976, p. 1664.

38. Martin, E. D.: Advances in the Application of Fast Semidirect Computational Methods in Transonic Flow. Presented at the IUTAM Symposium Transsonicum II, Göttingen, Germany, Sept. 8-13, 1975. In Symposium Transsonicum II, K. Oswatitsch and D. Rues, eds., Springer-Verlag (Berlin), 1976, pp. 431-438.

39. Jameson, A.: Numerical Solution of Nonlinear Partial Differential Equations of Mixed Type, SYNSPADE 1975, Univ. of Maryland, May 1975.

40. Jameson, A.: Numerical Computation of Transonic Flows with Shock Waves. Presented at the IUTAM Symposium Transsonicum II, Göttingen, Germany, Sept. 8-13, 1975. In Symposium Transsonicum II, K. Oswatitsch and D. Rues, eds., Springer-Verlag (Berlin), 1976, pp. 384-414.

41. Caughey, D. A.; and Jameson, A.: Accelerated Iterative Calculation of Transonic Nacelle Flowfields. AIAA Paper 76-100, Jan. 1976.

42. Stewart, G. W.: Introduction to Matrix Computations. Academic Press, 1973.

43. Lighthill, M. J.: Higher Approximations, Section E in General Theory of High Speed Aerodynamics. High Speed Aerodynamics and Jet Propulsion, Vol. VI, W. R. Sears, ed., Princeton Univ. Press, Princeton, N. J., 1954; also available as Princeton Aeronautical Paperback 5, entitled Higher Approximations in Aerodynamic Theory, 1960.

44. Jones, R. T.; and Cohen, D.: Aerodynamics of Wings at High Speeds, Section A in Aerodynamic Components of Aircraft at High Speeds. High Speed Aerodynamics and Jet Propulsion, Vol. VII, A. F. Donovan and H. R. Lawrence, eds., Princeton Univ. Press, Princeton, N. J., 1957; also available as Princeton Aeronautical Paperback 6, entitled High Speed Wing Theory, 1960.

45. Ames, W. F.: Numerical Methods for Partial Differential Equations. Barnes and Noble, 1969.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546