Numerical Solution of the Two–Dimensional Time–Dependent Incompressible Euler Equations

By

David L. Whitfield

and

Lafayette K. Taylor

Computational Fluid Dynamics Laboratory

NSF Engineering Research Center for Computational Field Simulation

April 1994

Mississippi State University
P.O. Box 6176
Mississippi State, MS 39762

Table of Contents

Acknowledgements

This research was sponsored by the NASA Lewis Research Center under Research Grant NAG 3-1157 with Dr. John Adamczyk as the technical monitor. This support is gratefully acknowledged.

Abstract

A numerical method is presented for solving the artificial compressibility form of the two–dimensional time–dependent incompressible Euler equations. The approach is based on using an approximate Riemann solver for the cell face numerical flux of a finite volume discretization. Characteristic variable boundary conditions are developed and presented for all boundaries and in–flow out–flow situations. The system of algebraic equations is solved using the discretized Newton–relaxation (DNR) implicit method. Numerical results are presented for both steady and unsteady flow.

I. Introduction

As a predecessor to the three–dimensional time–dependent dynamic multiblock incompressible laminar and/or turbulent Navier–Stokes work [1 and 2], the two–dimensional incompressible Euler equations were investigated and a research code was written to solve the equations numerically. It turns out that this rather simple two–dimensional incompressible Euler code has been extremely useful in investigating various numerical flux formulations, boundary conditions, and numerical solution algorithms that have fed back into the three–dimensional codes. It has also proven to be a useful teaching tool for introducing computational techniques to various interested parties. Moreover, it has been used as a test code for parallelization. In view of a number of requests for information, it seemed appropriate to record the details of the particular numerical solution scheme used to solve these time–dependent incompressible Euler equations.

The computer code resulting from this effort is part of a family of Mississippi State codes that are now known as UNCLE (Unsteady Computation of Field Equations). Since this code is used as a research tool in solving the equations of Leonhard Euler, it is named UNCLE.LEONHARD.

The particular numerical methods used to solve the equations are the current ones used as of this writing. No attempt is made to review all of the techniques that have been tried and used, and/or tried and discarded. The emphasis is on simplicity, while simultaneously providing information on numerical methods that form much of the foundation of the three–dimensional time–dependent Navier–Stokes solvers.

The system of equations to be solved is based on the artificial compressibility idea of Chorin[3]. This nonlinear system and its mathematically equivalent quasilinear system is presented in Chapter II. The eigensystem of the flux Jacobian matrices of the quasilinear system is developed in Chapter III. A finite volume discretization is used and this formulation is given in Chapter IV. The cell face numerical flux is based on an approximate Riemann solver which uses the eigensystem of the quasilinear form of the equations and this flux formulation is developed in Chapter V. The treatment of boundary conditions is given in Chapter VI. The numerical method used to solve the resulting system of algebraic equations is developed and presented in Chapter VII. Numerical results are presented in Chapter VIII and concluding remarks are offered in Chapter IX.

The body of this report is concerned with time–dependent equations on a stationary grid. However, the class of problems that can be solved is expanded considerably if the grid is allowed to move. Therefore, since the basic methods presented can be applied to dynamic grids, an appendix is included that contains the two–dimensional time–dependent incompressible Euler equations on a dynamic grid, along with the corresponding eigensystem.

II. Equations

The two–dimensional time–dependent incompressible Euler equations in Cartesian coordinates are

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \tag{2.1}$$

where

$$q = \begin{bmatrix} 0 \\ u \\ v \end{bmatrix}$$

$$f = \begin{bmatrix} u \\ u^2 + p \\ uv \end{bmatrix}$$

$$g = \begin{bmatrix} v \\ uv \\ v^2 + p \end{bmatrix}$$

These equations were made dimensionless by the relations

$$\begin{aligned}
x &= \frac{\bar{x}}{\bar{L}}, & y &= \frac{\bar{y}}{\bar{L}} \\
u &= \frac{\bar{u}}{\bar{q}_r}, & v &= \frac{\bar{v}}{\bar{q}_r} \\
\varrho &= \frac{\bar{\varrho}}{\bar{\varrho}_r}, & p &= \frac{\bar{p} - \bar{p}_r}{\bar{\varrho}_r \, \bar{q}_r^2} \\
t &= \frac{\bar{t} \, \bar{q}_r}{\bar{L}}
\end{aligned} \tag{2.2}$$

where the barred quantities are dimensional variables, $\bar{L}$ is a characteristic length, $\bar{q}_r$ is the magni–

tude of a reference velocity, $\bar{\varrho}_r$ is a reference density, and $\bar{t}$ is time. The density is assumed to be constant and hence ϱ is unity. The velocity components u and v correspond to the x and y directions respectively. Notice that the pressure variable p is twice the pressure coefficient.

The three–dimensional time–dependent incompressible Euler equations in Cartesian coordinates were transformed to time–dependent curvilinear coordinates in Ref. 1. After the equations were transformed to curvilinear coordinates they were written in Chorin's [3] artificial compressibility form as explained in Ref. 1. Following this same approach, but for the more simple case of two dimensions rather than three and for a stationary curvilinear coordinate system as opposed to one that can move in time, the artificial compressibility form of the two–dimensional time–dependent incompressible Euler equations in curvilinear coordinates is

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0 \tag{2.3}$$

where

$$Q = J \begin{bmatrix} p \\ u \\ v \end{bmatrix} , \quad F = J \begin{bmatrix} \beta\, U \\ u\, U + \xi_x\, p \\ v\, U + \xi_y\, p \end{bmatrix} , \quad G = J \begin{bmatrix} \beta\, V \\ u\, V + \eta_x\, p \\ v\, V + \eta_y\, p \end{bmatrix}$$

$$U = \xi_x\, u + \xi_y\, v$$

$$V = \eta_x\, u + \eta_y\, v$$

$$\xi = \xi(x,y)$$

$$\eta = \eta(x,y)$$

$$\tau = t$$

$$J = \left| \frac{\partial(x,y)}{\partial(\xi,\eta)} \right| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} = x_\xi\, y_\eta - x_\eta\, y_\xi$$

$$\xi_x = J^{-1} y_\eta$$
$$\xi_y = -J^{-1} x_\eta$$
$$\eta_x = -J^{-1} y_\xi$$
$$\eta_y = J^{-1} x_\xi$$

and β = constant. Notice that flux vectors F and G can be written

$$K = J \begin{bmatrix} \beta\,\theta_k \\ u\theta_k + k_x\,p \\ v\theta_k + k_y\,p \end{bmatrix} \tag{2.4}$$

where

$$\theta_k = k_x\,u + k_y\,v \tag{2.5}$$

and

$$K = F \text{ and } \theta_k = U \text{ for } k = \xi$$
$$K = G \text{ and } \theta_k = V \text{ for } k = \eta$$

The quasilinear form of Eq. (2.3) is

$$\frac{\partial Q}{\partial \tau} + A\frac{\partial Q}{\partial \xi} + B\frac{\partial Q}{\partial \eta} = 0 \tag{2.6}$$

where

$$A = \frac{\partial F}{\partial Q}\,, \quad B = \frac{\partial G}{\partial Q} \tag{2.7}$$

The Jacobian matrices A and B are denoted by $\overline{K}$ where the elements of $\overline{K}$ are given by

$$\overline{K}_{ij} = \frac{\partial K_i}{\partial Q_j} \tag{2.8}$$

Therefore,

$$K = F \quad \text{and} \quad \overline{K} = A \quad \text{when} \quad k = \xi$$

and

$$K = G \quad \text{and} \quad \overline{K} = B \quad \text{when} \quad k = \eta$$

The flux vectors represented by Eq. (2.4) can be written in terms of the elements of the dependent variable vector $Q = [\ Jp,\ Ju,\ Jv\ ]^T = \left[\ Q_1,\ Q_2,\ Q_3\right]^T$ as

$$K = \begin{bmatrix} \beta\ (k_x\ Q_2 + k_y\ Q_3) \\ \frac{1}{J}\left[Q_2(k_x\ Q_2 + k_y\ Q_3)\right] + k_x\ Q_1 \\ \frac{1}{J}\left[Q_3(k_x\ Q_2 + k_y\ Q_3)\right] + k_y\ Q_1 \end{bmatrix} \tag{2.9}$$

The elements of the Jacobian matrix $\overline{K}$ are determined from Eq. (2.8) resulting in

$$\overline{K} = \begin{bmatrix} 0 & \beta\ k_x & \beta\ k_y \\ k_x & \theta_k + k_x\ u & k_y\ u \\ k_y & k_x\ v & \theta_k + k_y\ v \end{bmatrix} \tag{2.10}$$

III. Eigensystem

The development of the numerical method used to solve the Euler equations (Eq.(2.3)) depends on the eigenvalues and eigenvectors of the flux Jacobian matrix $\overline{K}$ given by Eq. (2.10). These eigenvalues and eigenvectors comprise the eigensystem of $\overline{K}$, which is developed in this Chapter.

The eigenvalues of $\overline{K}$ are given by the solution to the characteristic polynomial resulting from

$$| \overline{K} - \lambda I | = 0 \tag{3.1}$$

It is relatively easy to solve this polynomial and obtain the eigenvalues

$$\lambda_1 = \theta_k \tag{3.2a}$$

$$\lambda_2 = \theta_k + c \tag{3.2b}$$

$$\lambda_3 = \theta_k - c \tag{3.2c}$$

where

$$c = \left[\theta_k^2 + \beta \left(k_x^2 + k_y^2\right)\right]^{\frac{1}{2}} \tag{3.3}$$

and θ_k is given by Eq. (2.5).

Because the eigenvalues given by Eqs. (3.2) are real and distinct, the system is strictly hyperbolic and thus one is assured of a linearly independent set of eigenvectors [4]. The right eigenvectors, r, are obtained from the solution to

$$\left(\overline{K} - \lambda I\right) r = 0 \tag{3.4}$$

A set of linearly independent right eigenvectors is given by

$$r_1 = \begin{bmatrix} 0 \\ -k_y \\ k_x \end{bmatrix}, \quad r_2 = \begin{bmatrix} c \\ \dfrac{u\lambda_2}{\beta} + k_x \\ \dfrac{v\lambda_2}{\beta} + k_y \end{bmatrix}, \quad r_3 = \begin{bmatrix} -c \\ \dfrac{u\lambda_3}{\beta} + k_x \\ \dfrac{v\lambda_3}{\beta} + k_y \end{bmatrix} \tag{3.5}$$

A similarity matrix R_k can be formed by using these right eigenvectors as columns

$$R_k = \begin{bmatrix} 0 & c & -c \\ -k_y & \dfrac{u\lambda_2}{\beta} + k_x & \dfrac{u\lambda_3}{\beta} + k_x \\ k_x & \dfrac{v\lambda_2}{\beta} + k_y & \dfrac{v\lambda_3}{\beta} + k_y \end{bmatrix}. \tag{3.6}$$

A set of left eigenvectors, y^T, can be obtained from the solution to

$$y^T\,(\overline{K} - \lambda\ I) = 0 \tag{3.7}$$

In this case, since the R_k matrix is only 3x3, it is probably easier to invert R_k and obtain

$$R_k^{-1} = \frac{\beta}{2c^2}\begin{bmatrix} \dfrac{2}{\beta}\left(uk_y - vk_x\right) & -2\left(\dfrac{v\theta_k}{\beta} + k_y\right) & 2\left(\dfrac{u\theta_k}{\beta} + k_x\right) \\ -\dfrac{\lambda_3}{\beta} & k_x & k_y \\ -\dfrac{\lambda_2}{\beta} & k_x & k_y \end{bmatrix} \tag{3.8}$$

where the rows of R_k^{-1} are the left eigenvectors.

The Jacobian matrices are now diagonalized by

$$R_k^{-1}\,\overline{K}\,R_k = \Lambda_k \tag{3.9}$$

where Λ_k is the diagonal matrix with diagonal elements $\lambda_1, \lambda_2,$ and λ_3. That is

$$\Lambda_k = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} \theta_k & 0 & 0 \\ 0 & \theta_k + c & 0 \\ 0 & 0 & \theta_k - c \end{bmatrix} \tag{3.10}$$

where the subscript k corresponds to $k = \xi \; or \; k = \eta$, and θ_k is given by Eq.(2.5) and c is given by Eq.(3.3).

The eigensystem of $\overline{K}$ is therefore composed of the eigenvalues given by Eqs.(3.2), the right eigenvectors given as the columns of the matrix R_k (Eq.(3.6)), and the left eigenvectors given as the rows of the matrix R_k^{-1} (Eq.(3.8)). Note that each eigenvector corresponds to one particular eigenvalue.

IV. Discretization

MacCormack and Paullay [5] discretized the equations into what is now commonly referred to as the finite volume form. By integrating Eq. (2.3) in space with the index notation shown in Fig. 1, the finite volume form of the equations can be written

$$\overline{\frac{\partial Q}{\partial \tau}}\Bigg)_{i,j} \Delta\xi\Delta\eta + \left(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}\right)\Delta\eta + \left(G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}}\right)\Delta\xi = 0 \qquad (4.1)$$

where the bar over the time derivative indicates various numerical approximations to the time rate of change of the dependent variable vector, which is recognized in this approach to be located at the center of the cell. Using central difference notation, Eq. (4.1) can be written

$$\overline{\frac{\partial Q}{\partial \tau}}\Bigg)_{i,j} + \frac{\delta_i F}{\Delta\xi} + \frac{\delta_j G}{\Delta\eta} = 0 \qquad (4.2)$$

With $\Delta\xi = 1$ and $\Delta\eta = 1$, Eq. (4.2) becomes

$$\overline{\frac{\partial Q}{\partial \tau}}\Bigg)_{i,j} + \delta_i F + \delta_j G = 0 \qquad (4.3)$$

Various approximations of the time derivative have been used. The most commonly used approximation is first–order in time given by

$$\overline{\frac{\partial Q}{\partial \tau}}\Bigg)_{i,j} = \frac{Q^{n+1}_{i,j} - Q^{n}_{i,j}}{\Delta\tau} = \frac{\Delta Q^{n}_{i,j}}{\Delta\tau} \qquad (4.4)$$

where the superscripts indicate time level. For higher order in time the second–order backward–approximation

$$\left.\frac{\partial Q}{\partial \tau}\right)_{i,j} = \frac{3Q_{i,j}^{n+1} - 4Q_{i,j}^{n} + Q_{i,j}^{n-1}}{2\Delta\tau} \tag{4.5}$$

is commonly used.

The finite volume formulation given by Eq. (4.3) states the relation between the summation of fluxes across the faces of a cell and the time rate of change of the dependent variables in the cell. The finite volume formulation offers considerable flexibility with regard to geometric complexity because the cells can be of any shape in physical space.

Use of central difference notation in this finite volume formulation (Eq. (4.3)) simply indicates the difference in flux vectors of opposite faces for the cell centered at the point *i, j* and should not be construed to mean the numerical solution scheme is central difference. The reason the scheme will not be central difference is that the flux vectors will be determined using upwind (one–sided) or upwind–biased techniques as explained in the following Chapter.

V. Numerical Flux

In this finite volume discretization the summation of fluxes passing across cell faces is balanced with the time rate of change of the dependent variables in the cell. There are numerous ways of obtaining the flux vector at cell faces; however, the quality of the numerical solution is critically dependent on how the flux vector is formulated. Having investigated numerous ways of forming the flux, it is the authors' observation that Godunov[6] type schemes have much to offer. In this approach a series of Riemann problems are solved where each Riemann problem corresponds to a cell face. Roe [7] followed Godunov's idea, but developed a much more computationally efficient method of solving the Riemann problems. This class of numerical approaches is referred to as approximate Riemann solvers and are frequently used in the computation of compressible flow where the convective portion of the equations form a hyperbolic system. By using Chorin's [3] idea of artificial compressibility, the incompressible equations are also hyperbolic and therefore approximate Riemann solvers can be used. The approximate Riemann solver of Roe is the foundation of the present numerical approach.

A tutorial type discussion of the Roe scheme is given in Ref. [8]. In the development of this method, waves are assumed to move normal to a cell face. This allows the numerical flux vector to be obtained independently in each of the computational coordinate directions ξ and η. It is popular in the literature to state three different ways of getting the flux vector at a cell face. The most frequently used method,usually written as the third expression for the flux vector, is actually an average of the first two expressions [7]. There is a larger operation count in using this averaged expression as compared to either one of the other two, and so the expression used here for the first–order Roe numerical flux vector at cell face $i+\frac{1}{2},j$ in the ξ computational direction is

$$\overline{F}_{i+\frac{1}{2},j} = \left[\, F\left(Q_{i,j}\right) \right]_{i+\frac{1}{2},j} + \sum_{j=1}^{3} a^{(j)}_{i+\frac{1}{2},j} \; \lambda^{-(j)}_{i+\frac{1}{2},j} \; r^{(j)}_{i+\frac{1}{2},j} \qquad (5.1)$$

where

$$a^{(j)}_{i+\frac{1}{2},j} = l^{(j)}_{i+\frac{1}{2},j} \cdot \left(Q_{i+1,j} - Q_{i,j}\right) \qquad (5.2)$$

and $\lambda^{-(j)}$ correspond to the negative eigenvalues of the Roe matrix, $r^{(j)}$ are the right eigenvectors of the Roe matrix, $l^{(j)}$ are left eigenvectors of the Roe matrix, and the scalars $\alpha^{(j)}$ are jumps in the characteristic variables. The subscript $i+\frac{1}{2},j$ in the equations above indicates that the metrics used correspond to the cell face located at $i+\frac{1}{2},j$. The dependent variables in the eigenvalues and eigenvectors are the Roe averaged variables [7] at cell face $i+\frac{1}{2},j$ since they comprise the eigensystem of the Roe matrix. The flux vector $F(Q_{i,j})$ in Eq. (5.1) is evaluated using the dependent variable vector as indicated (not the Roe averaged variables), but the subscript $i+\frac{1}{2},j$ on the bracket indicates the metrics at $i+\frac{1}{2},j$ are to be used. The eigensystem used in Eqs. (5.1) and (5.2) are, therefore, the columns of Eq. (3.6) for the right eigenvectors, the rows of Eq. (3.8) for the left eigenvectors, and the diagonal elements of Eq. (3.10) (with the notation that λ_j in Eq. (3.10) is $\lambda^{(j)}$ in Eq. (5.1)), all with the Roe averaged velocities

$$\overline{u}_{i+\frac{1}{2},j} = \frac{1}{2}\left(u_{i+1,j} + u_{i,j}\right) \tag{5.3a}$$

and

$$\overline{v}_{i+\frac{1}{2},j} = \frac{1}{2}\left(v_{i+1,j} + v_{i,j}\right) \tag{5.3b}$$

and with $k = \xi$ in Eqs. (3.6), (3.8), and (3.10). The flux vector $F(Q_{i,j})$ in Eq. (5.1) comes from Eq. (2.4) with $k = \xi$ and is

$$F\left(Q_{i,j}\right) = J\begin{bmatrix} \beta\left(\theta_\xi\right)_{i,j} \\ u_{i,j}\left(\theta_\xi\right)_{i,j} + \xi_x\, p_{i,j} \\ v_{i,j}\left(\theta_\xi\right)_{i,j} + \xi_y\, p_{i,j} \end{bmatrix} \tag{5.4}$$

where

$$\left(\theta_{\xi}\right)_{i,j} = U_{i,j} = \xi_x \, u_{i,j} + \xi_y \, v_{i,j} \tag{5.5}$$

Another way of writing Eq. (5.1) is [8]

$$\overline{F}_{i+\frac{1}{2},j} = \left[\, F\left(Q_{i,j}\right) \right]_{i+\frac{1}{2},j} + \overline{A}^{-} \left(Q_{i,j} \; , \; Q_{i+1,j}\right)\left(Q_{i+1,j} - Q_{i,j}\right) \tag{5.6}$$

where $\overline{A}^{-}$ is the Roe matrix given by

$$\overline{A}^{-} = R_{\xi} \, \Lambda_{\xi}^{-} \; R_{\xi}^{-1} \tag{5.7}$$

where Λ_{ξ}^{-} has only nonpositive elements. All the dependent variables in Eq. (5.7) are the Roe averaged variables given by Eqs. (5.3). All the metrics in the $\overline{A}^{-}$ matrix correspond to cell face $i + \frac{1}{2},j$.

The numerical flux vector, $\overline{G}$, for the η direction can be written in a fashion analogous to the vector $\overline{F}$ for the ξ direction. For example, the vector $\overline{G}$ in the form similar to Eq. (5.6) would be

$$\overline{G}_{i,j+\frac{1}{2}} = \left[\, G\left(Q_{i,j}\right) \right]_{i,j+\frac{1}{2}} + \overline{B}^{-} \left(Q_{i,j} \, , \, Q_{i,j+1}\right)\left(Q_{i,j+1} - Q_{i,j}\right) \tag{5.8}$$

where $\overline{B}^{-}$ is the Roe matrix given by

$$\overline{B}^{-} = R_{\eta} \, \Lambda_{\eta}^{-} \; R_{\eta}^{-1} \tag{5.9}$$

where A_η^- has only nonpositive elements. The dependent variables in Eq. (5.9) are the Roe averaged variables given in the η direction by

$$\bar{u}_{i,j+\frac{1}{2}} = \frac{1}{2}\left(u_{i,j+1} + u_{i,j}\right) \tag{5.10a}$$

$$\bar{v}_{i,j+\frac{1}{2}} = \frac{1}{2}\left(v_{i,j+1} + v_{i,j}\right) \tag{5.10b}$$

All the metrics in the $\bar{B}^-$ matrix correspond to cell face $i\,,j + \frac{1}{2}$.

Consider the numerical flux vector in the ξ direction given by Eq. (5.6). Notice that since Eq. (5.6) is the solution to a Riemann problem, the conditions to the right of the cell face at $i + \frac{1}{2},j$ are given by $Q_{i+1\,,j}$ and the conditions to the left of this cell face are given by $Q_{i\,,j}$. Equation (5.6), therefore, could be written as

$$\bar{F}_{i+\frac{1}{2},j} = \left[F\left(Q^L_{i+\frac{1}{2},j}\right)\right]_{i+\frac{1}{2},j} + \bar{A}^-\left(Q^L_{i+\frac{1}{2},j}\,,\; Q^R_{i+\frac{1}{2},j}\right)\left(Q^R_{i+\frac{1}{2},j} - Q^L_{i+\frac{1}{2},j}\right) \tag{5.11}$$

where

$$Q^R_{i+\frac{1}{2},j} = Q_{i+1\,,j} \tag{5.12a}$$

and

$$Q^L_{i+\frac{1}{2},j} = Q_{i\,,j} \tag{5.12b}$$

The corresponding Roe averaged variables used in $\bar{A}^-$ would be

$$\bar{u}_{i+\frac{1}{2},j} = \frac{1}{2}\left(u^R_{i+\frac{1}{2},j} + u^L_{i+\frac{1}{2},j}\right) \tag{5.13a}$$

and

$$\bar{v}_{i+\frac{1}{2},j} = \frac{1}{2}\left(v^R_{i+\frac{1}{2},j} + v^L_{i+\frac{1}{2},j}\right) \tag{5.13b}$$

The numerical flux vectors introduced thus far lead to first–order accurate schemes in space. These numerical schemes can be made higher order by using the MUSCL approach of van Leer [9]. Anderson, Thomas, and van Leer [10] used a MUSCL–type approach in flux vector split schemes. Following [10], the dependent variables just to the right of the cell face, Q^R, located at $i + \frac{1}{2},j$ and just to the left of the same cell face, Q^L, are written as

$$Q^R_{i+\frac{1}{2},j} = Q_{i+1,j} - \frac{\phi}{4}\left[(1-\varkappa)\left(Q_{i+2,j} - Q_{i+1,j}\right) + (1+\varkappa)\left(Q_{i+1,j} - Q_{i,j}\right)\right] \tag{5.14a}$$

$$Q^L_{i+\frac{1}{2},j} = Q_{i,j} + \frac{\phi}{4}\left[(1-\varkappa)\left(Q_{i,j} - Q_{i-1,j}\right) + (1+\varkappa)\left(Q_{i+1,j} - Q_{i,j}\right)\right] \tag{5.14b}$$

For $\phi = 0$, Eqs. (5.14) recover Eqs. (5.12) and hence the numerical scheme would be first–order in space. For higher order schemes set $\phi = 1$. With $\varkappa = -1$ only points to the right of the cell face are used for Q^R and only points to the left of the cell face are used for Q^L, and a second–order scheme results. With $\varkappa = 1/3$ two points to the right and one point to the left of the cell face are used for Q^R, and two points to the left and one point to the right of the cell face are used for Q^L. The resulting scheme is referred to as third–order upwind–biased [10]. Note that the third–order upwind–biased scheme depends on information from the same number of points as the second–order scheme.

The same approach as used here for the $\bar{F}$ numerical flux vector corresponding to the ξ, or i, computational direction is applied to the $\bar{G}$ numerical flux vector corresponding to the η, or j, computational direction. For example, $\bar{G}$ would be

$$\bar{G}_{i,j+\frac{1}{2}} = \left[G\left(Q^L_{i,j+\frac{1}{2}}\right)\right]_{i,j+\frac{1}{2}} + \bar{B}^-\left(Q^L_{i,j+\frac{1}{2}},\ Q^R_{i,j+\frac{1}{2}}\right)\left(Q^R_{i,j+\frac{1}{2}} - Q^L_{i,j+\frac{1}{2}}\right) \tag{5.15}$$

Note from Eqs. (3.2) and (3.3) that for incompressible flow, unlike compressible flow, the second eigenvalue is never less than zero. Note, also, that the numerical flux vector formulation used for $\overline{F}$ and $\overline{G}$ in Eqs. (5.6) and (5.8) do not use the positive eigenvalues. Advantage is taken of this fact during coding in order to reduce memory and floating point operations. To illustrate this, consider, for example, the second term on the right–hand–side of Eq. (5.6) (or Eq. (5.1) since they are the same) and use the following generic definitions

$$A = R\,\Lambda\,R^{-} \tag{5.16}$$

where

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$R^{-1} = \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

Then

$$A \cdot \delta Q = \lambda_1(l_{11}\,\delta Q_1 + l_{12}\,\delta Q_2 + l_{13}\,\delta Q_3)\begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix}$$

$$+ \lambda_2(l_{21}\,\delta Q_1 + l_{22}\,\delta Q_2 + l_{23}\,\delta Q_3)\begin{bmatrix} r_{12} \\ r_{22} \\ r_{32} \end{bmatrix}$$

$$+ \lambda_3(l_{31}\,\delta Q_1 + l_{32}\,\delta Q_2 + l_{33}\,\delta Q_3)\begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix} \qquad (5.17)$$

Because λ_2 will never be less than zero the second term in Eq. (5.17) will never be needed, which means that the second left and right eigenvectors will never be needed. Moreover, the first term will be needed only part of the time, i.e. when $\lambda_1 < 0$.

It should be pointed out that the alternative flux vector formulation of Roe [7] given by

$$\overline{F}_{i+\frac{1}{2},j} = \left[\, F\left(Q_{i+1,j}\right) \right]_{i+\frac{1}{2},j} - \overline{A}^{+}\left(Q_{i,j}\,,\; Q_{i+1,j}\right)\left(Q_{i+1,j} - Q_{i,j}\right) \qquad (5.18)$$

where

$$\overline{A}^{+} = R_{\xi}\,\Lambda_{\xi}^{+}\,R_{\xi}^{-1}$$

and Λ_{ξ}^{+} has only non–negative elements, could have been used. Because λ_3 will never be greater than zero (see Eq. (3.2c)), either formulation would lead to the same savings and, more importantly, to the same numerical results.

It might be remarked that the approach just outlined for obtaining the numerical flux vector is a significant departure from the approach used in Refs. [1] and [2]. The reason is that the method previously used came from experiences gained in compressible flows [8] and [11]. Limiters were used in compressible flow and the method used to formulate the numerical flux vector in [8] was found to have certain advantages compared to a MUSCL–type approach. However, limiters have never been required in incompressible flow and the approach just outlined, and the approach of [8], tend to give the same numerical result for incompressible flow as will be demonstrated in Chapter VIII. Since the convergence rate of the two methods has been found to be essentially the same, and because the present approach requires less memory and less floating point operations, the present approach is currently the preferred approach.

VI. Boundary Conditions

Boundary conditions on all surfaces are developed using characteristic variable boundary conditions. The approach follows that presented in Ref.[12] for the numerical solution of the three–dimensional unsteady Euler equations for compressible flow. Loper [13] refers to this approach as "reference plane" boundary conditions, where the reference plane corresponds to the plane resulting from keeping constant the computational coordinate along which the boundary conditions are developed. In what follows, this boundary plane will be located on the plane resulting from keeping k = *constant*, where k can be either the ξ or η computational coordinate. Then from Ref. [12], the characteristic variable boundary conditions will depend on the equation

$$R_k^{-1}\left(\frac{\partial Q}{\partial \tau} + R_k \Lambda_k R_k^{-1} \frac{\partial Q}{\partial k}\right) = 0 \tag{6.1}$$

or

$$R_k^{-1}\frac{\partial Q}{\partial \tau} + \Lambda_k R_k^{-1} \frac{\partial Q}{\partial k} = 0 \tag{6.2}$$

Assuming R_k^{-1} to be a constant matrix with elements corresponding to some reference conditions denoted by the subscript "o", Eq. (6.2) can be written

$$\frac{\partial W_k}{\partial \tau} + \Lambda_k \frac{\partial W_k}{\partial k} = 0 \tag{6.3}$$

where W_k is the characteristic variable vector and is defined as

$$W_k = R_{k,o}^{-1} Q \tag{6.4}$$

Because Λ_k is a diagonal matrix, Eq. (6.3) represents three uncoupled equations. For example, if the elements of W_k are given by

$$W_k = \begin{bmatrix} w_{k,1} \\ w_{k,2} \\ w_{k,3} \end{bmatrix} \tag{6.5}$$

then Eq. (6.3) can be written as

$$\frac{\partial w_{k,m}}{\partial \tau} + \lambda_m \frac{\partial w_{k,m}}{\partial k} = 0 \tag{6.6}$$

for $m=1,2,$ *and* 3. Note that Eq. (6.6) can also be written as

$$\frac{dw_{k,m}}{d\tau} = \frac{\partial w_{k,m}}{\partial \tau} + \lambda_m \frac{\partial w_{k,m}}{\partial k} = 0 \tag{6.7}$$

where

$$\lambda_m = \frac{dk}{d\tau} \tag{6.8}$$

Therefore, each characteristic variable is associated with one particular eigenvalue, and this characteristic variable is constant along the characteristic line described by the direction given by Eq. (6.8).

Since the left eigenvectors are rows of the matrix R_k^{-1}, each characteristic variable $w_{k,m}$ is the dot product of the m^{th} left eigenvector with the dependent variable vector Q. Therefore, the characteristic variables are obtained using R_k^{-1} given by Eq. (3.8) evaluated at some reference conditions times Q. However, note from Eq. (6.7) that any constant times a characteristic variable also satisfies Eq. (6.7). This fact can be used to simplify the characteristic variables somewhat. Taking advantage of this fact is helpful in order to simplify the algebra involved in developing explicit expressions for the boundary conditions. The characteristic variables used here are

$$w_{k,1} = (u\ k_y - v\ k_x)_o\ p - (v\ \theta_k + \beta\ k_y)_o\ u + (u\ \theta_k + \beta\ k_x)_o\ v \tag{6.9a}$$

$$w_{k,2} = (\theta_k - c)_o\ p - \beta(k_x\ u + k_y\ v) \tag{6.9b}$$

$$w_{k,3} = (\theta_k + c)_o\ p - \beta(k_x\ u + k_y\ v) \tag{6.9c}$$

where the characteristic variables correspond in order to the eigenvalues

$$\lambda_1 = \theta_k \tag{6.10a}$$

$$\lambda_2 = \theta_k + c \tag{6.10b}$$

$$\lambda_3 = \theta_k - c \tag{6.10c}$$

As a reminder

$$\theta_k = k_x\, u + k_y\, v \tag{6.11}$$

and

$$c = \left[\theta_k^2 + \beta(k_x^2 + k_y^2)\right]^{\frac{1}{2}} \tag{6.12}$$

These equations are now used to develop far field and impermeable wall boundary conditions for the two–dimensional incompressible Euler equations.

Codirectional Inflow:

In the development of boundary conditions for the compressible Euler equations on dynamic grids, Janus [14] coined the terminology codirectional and contradirectional. This can be a convenient terminology, and it is used here for the incompressible Euler equations on stationary grids. From Fig. 2a, codirectional flow is shown to be flow traveling across the boundary of interest in the direction of increasing *k* computational coordinate. Therefore, flow coming into the computational domain traveling in the direction of increasing *k* is codirectional inflow. In Fig. 2 the point labeled *"a"* corresponds to flow approaching the boundary, the point *"b"* corresponds to the boundary, and the point labeled *"l"* corresponds to flow leaving the boundary.

For codirectional inflow the first two eigenvalues of Eqs. (6.10) are positive and run from outside the computational domain toward the boundary. The third eigenvalue is negative and runs from inside the computational domain toward the boundary. Since the respective characteristic variables along these characteristics are constant, reference to Fig. 2a indicates that

$$\left(w_{k,1}\right)_a = \left(w_{k,1}\right)_b \tag{6.13a}$$

$$\left(w_{k,2}\right)_a = \left(w_{k,2}\right)_b \tag{6.13b}$$

$$\left(w_{k,3}\right)_l = \left(w_{k,3}\right)_b \tag{6.13c}$$

That is, the first two characteristic variables are constant from point *"a"* to point *"b"* and the third characteristic variable is constant from point *"l"* to point *"b"*. Point *"a"* is taken as some reference point. For example, for flow about an airfoil in free air the characteristic variable is evaluated using freestream conditions. The point *"l"* is taken as the center of the first cell inside the computational domain. This leads to the equations

$$\begin{aligned}&\left(u\ k_y - v\ k_x\right)_o p_b - \left(v\ \theta_k + \beta\ k_y\right)_o u_b + \left(u\ \theta_k + \beta\ k_x\right)_o v_b = \\ &\left(u\ k_y - v\ k_x\right)_o p_\infty - \left(v\ \theta_k + \beta\ k_y\right)_o u_\infty + \left(u\ \theta_k + \beta\ k_x\right)_o v_\infty\end{aligned} \tag{6.14a}$$

$$\left(\theta_k - c\right)_o p_b - \beta k_x\ u_b - \beta k_y\ v_b = \left(\theta_k - c\right)_o p_\infty - \beta\ \theta_{k,\infty} \tag{6.14b}$$

$$\left(\theta_k + c\right)_o p_b - \beta k_x\ u_b - \beta k_y\ v_b = \left(\theta_k + c\right)_o p_l - \beta\ \theta_{k,l} \tag{6.14c}$$

Equations (6.14) can be solved for the three unknowns $p_b, u_b,$ and v_b to obtain

$$p_b = \frac{p_\infty + p_l}{2} + \frac{1}{2c_o}\left[\ \beta\left(\theta_{k,\infty} - \theta_{k,l}\right) - \theta_{k,o}\left(p_\infty - p_l\right)\right] \tag{6.15a}$$

$$u_b = u_\infty + \frac{p_\infty - p_b}{\beta c_o}\left[u_o\left(\theta_{k,o} - c_o\right) + \beta\ k_x\right] \tag{6.15b}$$

$$v_b = v_\infty + \frac{p_\infty - p_b}{\beta c_o}\left[v_o\left(\theta_{k,o} - c_o\right) + \beta\ k_y\right] \tag{6.15c}$$

The reference conditions indicated by the subscript *"o"* are constructed from an average of dependent variables located at the boundary and at the center of the first cell inside the computational domain at the previous time step.

Contradirectional Inflow:

Contradirectional inflow is illustrated in Fig. 2b. In this case the first and third eigenvalues given by Eqs. (6.10) are negative and run from outside the computational domain to the boundary. The second eigenvalue is positive and runs from inside the computational domain to the boundary. The characteristic variables are, therefore, grouped as

$$\left(w_{k,1}\right)_a = \left(w_{k,1}\right)_b \tag{6.16a}$$

$$\left(w_{k,2}\right)_l = \left(w_{k,2}\right)_b \tag{6.16b}$$

$$\left(w_{k,3}\right)_a = \left(w_{k,3}\right)_b \tag{6.16c}$$

These relations lead to the equations

$$\begin{aligned}&\left(u\ k_y - v\ k_x\right)_o p_b - \left(v\ \theta_k + \beta\ k_y\right)_o u_b + \left(u\ \theta_k + \beta\ k_x\right)_o v_b = \\ &\left(u\ k_y - v\ k_x\right)_o p_\infty - \left(v\ \theta_k + \beta\ k_y\right)_o u_\infty + \left(u\ \theta_k + \beta\ k_x\right)_o v_\infty\end{aligned} \tag{6.17a}$$

$$\left(\theta_k - c\right)_o p_b - \beta k_x\ u_b - \beta k_y\ v_b = \left(\theta_k - c\right)_o p_l - \beta k_x\ u_l - \beta k_y\ v_l \tag{6.17b}$$

$$\left(\theta_k + c\right)_o p_b - \beta k_x\ u_b - \beta k_y\ v_b = \left(\theta_k + c\right)_o p_\infty - \beta k_x\ u_\infty - \beta k_y\ v_\infty \tag{6.17c}$$

Equations (6.17) can be solved for $p_b, u_b,$ and v_b to obtain

$$p_b = \frac{p_\infty + p_l}{2} - \frac{1}{2c_o}\left[\ \beta\left(\theta_{k,\infty} - \theta_{k,l}\right) - \theta_{k,o}\left(p_\infty - p_l\right)\right] \tag{6.18a}$$

$$u_b = u_\infty - \frac{p_\infty - p_b}{\beta c_o}\left[\ u_o\left(\theta_{k,o} + c_o\right) + \beta\ k_x\right] \tag{6.18b}$$

$$v_b = v_\infty - \frac{p_\infty - p_b}{\beta c_o}\left[\ v_o\left(\theta_{k,o} + c_o\right) + \beta\ k_y\right] \tag{6.18c}$$

Codirectional Outflow:

In this case the first two eigenvalues are positive and run from inside the computational domain to the boundary. The third eigenvalue is negative and runs from outside the computational domain to the boundary. However, unlike inflow, the characteristic variable associated with this third eigenvalue is not easy (or perhaps not enough is known about it) to evaluate. Therefore, for outflow, the pressure is specified on the boundary, and this specification of pressure eliminates, or replaces, the relation involving the incoming characteristic variable. The equations to solve for u_b and v_b, therefore, are

$$\begin{aligned} -\left(v\,\theta_k + \beta\,k_y\right)_o u_b + \left(u\,\theta_k + \beta\,k_x\right)_o v_b &= \left(u\,k_y - v\,k_x\right)_o \left(p_a - p_b\right) \\ &\quad - \left(v\,\theta_k + \beta\,k_y\right)_o u_a + \left(u\,\theta_k + \beta\,k_x\right)_o v_a \end{aligned} \tag{6.19a}$$

$$-\beta k_x\,u_b - \beta k_y\,v_b = \left(\theta_k - c\right)_o \left(p_a - p_b\right) - \beta k_x\,u_a - \beta k_y\,v_a \tag{6.19b}$$

$$p_b = p_{exit} \tag{6.19c}$$

Recall that the point *"a"* (see Fig. 2a) is associated with approaching flow and in this case it is located inside the computational domain at the center of the first cell inside the domain. With p_b specified by Eq. (6.19c), Eqs. (6.19a) and (6.19b) can be solved for u_b and v_b yielding

$$p_b = p_{exit} \tag{6.20a}$$

$$u_b = u_a + \frac{p_a - p_b}{\beta c_o}\left[\,u_o\left(\theta_{k,o} - c_o\right) + \beta k_x\right] \tag{6.20b}$$

$$v_b = v_a + \frac{p_a - p_b}{\beta c_o}\left[\,v_o\left(\theta_{k,o} - c_o\right) + \beta k_y\right] \tag{6.20c}$$

Contradirectional Outflow:

In this case the first and third eigenvalues are negative and run from inside the computational domain to the boundary. The second eigenvalue is positive and runs from outside the computational domain to the boundary. The characteristic variable relation corresponding to this incoming characteristic is replaced by the specification of pressure on the boundary just as was done for the codirectional outflow case. The equations for the boundary conditions, therefore, are

$$- \left(v\ \theta_k + \beta\ k_y\right)_o u_b + \left(u\ \theta_k + \beta\ k_x\right)_o v_b = \left(u\ k_y - v\ k_x\right)_o \left(p_a - p_b\right) - \left(v\ \theta_k + \beta\ k_y\right)_o u_a + \left(u\ \theta_k + \beta\ k_x\right)_o v_a \tag{6.21a}$$

$$p_b = p_{exit} \tag{6.21b}$$

$$- \beta k_x\ u_b - \beta k_y\ v_b = \left(\theta_k + c\right)_o \left(p_a - p_b\right) - \beta k_x\ u_a - \beta k_y\ v_a \tag{6.21c}$$

These equations can be solved to give

$$p_b = p_{exit} \tag{6.22a}$$

$$u_b = u_a - \frac{p_a - p_b}{\beta c_o} \left[\ u_o\left(\theta_{k,o} + c_o\right) + \beta\ k_x \right] \tag{6.22b}$$

$$v_b = v_a - \frac{p_a - p_b}{\beta c_o} \left[\ v_o\left(\theta_{k,o} + c_o\ \right) + \beta\ k_y \right] \tag{6.22c}$$

Impermeable Surface:

Consider an impermeable surface corresponding to a constant k line where the computational domain is located in the direction of increasing k. Then the center of the first cell inside the computational domain can be connected to the impermeable surface by a negative running characteristic corresponding to the negative eigenvalue. The characteristic variable relation corresponding to the positive eigenvalue is then replaced by the physical fact that there is no flow through the impermeable surface, or

$$\theta_{k,b} = 0 \tag{6.23}$$

If the computational domain is in the direction of decreasing k from the impermeable surface, the center of the first cell inside the computational domain can be connected to the impermeable surface by a positive running characteristic and Eq. (6.23) replaces the characteristic variable relation corresponding the negative eigenvalue. This leads to the following characteristic variable relationship

$$c_o\, p_b \mp \beta\, \theta_{k,b} = c_o\, p_{in} \mp \beta\, \theta_{k,in} \tag{6.24}$$

The $\theta_{k,o}$ term in Eqs. (6.9) has been set to zero in arriving at Eq. (6.24), and the subscript "*in*" in Eq. (6.24) identifies the center of the first cell inside the computational domain. By using Eq. (6.23) in Eq. (6.24) an expression for the pressure at an impermeable surface can be obtained as

$$p_b = p_{in} \mp \frac{\beta\, \theta_{k,in}}{c_o} \tag{6.25}$$

where the minus sign corresponds to the situation where the computational domain is in the positive k direction relative to the boundary, and the plus sign corresponds to the computational domain being in the negative k direction relative to the boundary.

Pressure is formally the only thing that is needed at an impermeable boundary for an Euler equation solver. However, to use a flux formulation higher than first order at the first cell face from a boundary, the remaining dependent variables are needed at the surface. An approximate expression for these remaining variables can be obtained by using the first characteristic variable (Eq. (6.9a)). The first eigenvalue is zero at an impermeable surface and consequently the first eigenvalue does not point from inside the computational domain to the surface. However, if the approximation is made that the first characteristic variable relation can be written as

$$\left(u\, k_y - v\, k_x\right)_o p_b - \beta k_y\, u_b + \beta k_x\, v_b = \left(u\, k_y - v\, k_x\right)_o p_{in} - \beta k_y\, u_{in} + \beta k_x\, v_{in} \tag{6.26}$$

then both additional velocity components can be obtained by solving Eqs. (6.26) and (6.23) for u_b and v_b. The complete impermeable surface boundary conditions can be written as

$$p_b = p_{in} \mp \frac{\beta\,\theta_{k,in}}{c_o} \tag{6.27a}$$

$$u_b = u_{in} + \left(\mp \frac{u_o}{c_o} - \frac{k_x}{k_x^2 + k_y^2} \right) \theta_{k,in} \tag{6.27b}$$

$$v_b = v_{in} + \left(\mp \frac{v_o}{c_o} - \frac{k_y}{k_x^2 + k_y^2} \right) \theta_{k,in} \tag{6.27c}$$

where, again, it is noted that the minus sign corresponds to the computational domain being located in the positive k direction relative to the boundary, and the plus sign corresponds to the computational domain being located in the negative k direction relative to the boundary.

VII. Numerical Solution

7.1 Formulation

An implicit solution scheme is developed by using the numerical flux vectors $\overline{F}$ and $\overline{G}$ from Eqs. (5.11) and (5.15) in the finite volume descretization given by Eq. (4.3) to obtain

$$\frac{\Delta Q^n_{i,j}}{\Delta \tau} + \delta_i\, \overline{F}\left(Q^{n+1}\right) + \delta_j\, \overline{G}\left(Q^{n+1}\right) = 0 \tag{7.1}$$

where

$$\Delta Q^n_{i,j} = Q^{n+1}_{i,j} - Q^n_{i,j} \tag{7.2a}$$

$$\delta_i\, \overline{F}\left(Q^{n+1}\right) = \overline{F}_{i+\frac{1}{2},j}\left(Q^{n+1}\right) - \overline{F}_{i-\frac{1}{2},j}\left(Q^{n+1}\right) \tag{7.2b}$$

$$\delta_j\, \overline{G}\left(Q^{n+1}\right) = \overline{G}_{i,j+\frac{1}{2}}\left(Q^{n+1}\right) - \overline{G}_{i,j-\frac{1}{2}}\left(Q^{n+1}\right) \tag{7.2c}$$

The first–order time derivative given by Eq. (4.4) is used in Eq. (7.1) for illustration. The approach used to develop methods of higher–order accuracy in time is the same as that presented below for first–order time accuracy.

The normal procedure for the solution of Eq. (7.1) would be to linearize the spatial difference terms, move the terms not containing ΔQ^n to the right–hand–side of the equations, and solve for ΔQ^n. This is particularly true for problems expected to have steady state solutions, because the sum of the spatial difference operator terms as well as ΔQ^n would both go to zero. However, for unsteady flow, the ideal situation would be to find Q^{n+1} such that Eq. (7.1) is satisfied. That is, find Q^{n+1} such that [15]

$$\mathcal{F}\left(Q^{n+1}\right) = 0 \tag{7.3}$$

where

$$\mathfrak{F}\left(Q^{n+1}\right) = \frac{\Delta Q^{n}_{i,j}}{\Delta\tau} + \delta_i\,\overline{F}\left(Q^{n+1}\right) + \delta_j\,\overline{G}\left(Q^{n+1}\right) \tag{7.4}$$

One way of attempting to solve this problem is to use Newton's method. Newton's method [16] for the function $\mathfrak{F}(x)$ would be

$$\mathfrak{F}'\left(x^m\right)\left(x^{m+1} - x^m\right) = -\,\mathfrak{F}\left(x^m\right) \tag{7.5}$$

where $m = 1, 2, 3, \ldots$ and $\mathfrak{F}'(x)$ is the Jacobian matrix of the vector $\mathfrak{F}(x)$. With $Q^{n+1} = x$, Eq. (7.5) becomes

$$\mathfrak{F}'\left(Q^{n+1,m}\right)\left(Q^{n+1,m+1} - Q^{n+1,m}\right) = -\,\mathfrak{F}\left(Q^{n+1,m}\right) \tag{7.6}$$

In principle the generated sequence $Q^{n+1,m+1}$ converges to Q^{n+1} and, hence, Eq. (7.3) is satisfied.

Newton's method has the attractive feature that the convergence rate can be quadratic [16]. Whereas, quadratic convergence is possible for computational fluid dynamic problems it is difficult to achieve; and even if achieved, it may not be the most computationally efficient way to solve the problem [17]. It usually occurs in practice that in the interest of computer time and memory one does not strive for quadratic convergence, rather one seeks variations [16] of Newton's method. Because the numerical problem under investigation is well formulated in the context of Newton's method, particularly for unsteady flows since the residual (which is driven to zero) now contains the time derivative, this approach of using variations or approximations of Newton's method is followed here.

7.2 Jacobian

The solution to Eq. (7.6) formally requires the Jacobian matrix $\mathfrak{F}'(x)$, where the elements of $\mathfrak{F}'(x)$ are defined as

$$a_{i,j}(x) = \frac{\partial \mathfrak{F}_i(x)}{\partial x_j} \tag{7.7}$$

Analytical development of the Jacobian of the Roe numerical flux vector is not simple, and evidently impractical to obtain in three dimensions. Since it is the intent of this work to produce techniques that can be used in three dimensions, an analytical derivation of the Jacobian doesn't appear feasible. In some cases flux vector splitting [18] has been used to obtain approximate analytical Jacobians for the solution matrix while flux differencing splitting, such as Roe's method, has been used on the right–hand–side of the equations [19]. Whereas this approach is inconsistent in that flux vector splitting is used on the left, and flux difference splitting is used on the right, it has been shown to work quite well [19 and 20]. This approach, however, was used in compressible flow where the flux vector was homogeneous of degree one and splitting of the flux vector was straightforward. For this incompressible formulation the flux vector does not have this homogeneity property and it is not obvious how to follow this approach. Therefore, the approach used here is to obtain the Jacobian numerically by using difference quotients [11]. The Jacobian elements are obtained from

$$a_{i,j}(x) = \frac{\mathcal{F}_i(x + h\, e_j) - \mathcal{F}_i(x)}{h} \tag{7.8}$$

where e_j is the j^{th} unit vector and

$$h \approx \sqrt{machine\ \ \varepsilon} \tag{7.9}$$

When a numerical Jacobian is used, Eq. (7.6) is known as discretized Newton iteration [16]. Define the following

$$D\overline{F}_{i+\frac{1}{2},j} = \left.\frac{\partial \overline{F}_{i+\frac{1}{2},j}}{\partial Q_{i,j}}\right)_m \tag{7.10a}$$

$$D\overline{G}_{j+\frac{1}{2},j} = \left.\frac{\partial \overline{G}_{i,j+\frac{1}{2}}}{\partial Q_{i,j}}\right)_m \tag{7.10b}$$

where the first subscript of $D\overline{F}$ and $D\overline{G}$ refers to the location of the cell face and the second subscript refers to the location of the dependent variable vector that the numerical flux vector is differentiated with respect to. Using these definitions, Eq. (7.6) can be written

$$- D\overline{G}_{j-\frac{1}{2},j-1}\, \Delta Q^{n+1,m}_{i,j-1} - D\overline{F}_{i-\frac{1}{2},i-1}\, \Delta Q^{n+1,m}_{i-1,j}$$

$$+ \left(\frac{I}{\Delta\tau} + D\overline{F}_{i+\frac{1}{2},i} + D\overline{G}_{j+\frac{1}{2},j} - D\overline{F}_{i-\frac{1}{2},i} - D\overline{G}_{j-\frac{1}{2},j}\right) \Delta Q^{n+1,m}_{i,j}$$

$$+ D\overline{F}_{i+\frac{1}{2},i+1}\, \Delta Q^{n+1,m}_{i+1,j} + D\overline{G}_{j+\frac{1}{2},j+1}\, \Delta Q^{n+1,m}_{i,j+1}$$

$$= - \left[I_a \frac{Q^{n+1,m}_{i,j} - Q^{n}_{i,j}}{\Delta\tau} + \delta_i\, \overline{F}\left(Q^{n+1,m}\right) + \delta_j\, \overline{G}\left(Q^{n+1,m}\right) \right] \tag{7.11}$$

where

$$\Delta Q^{n+1,m} = Q^{n+1,m+1} - Q^{n+1,m} \tag{7.12}$$

The term I_a is an identity matrix except the first diagonal element is zero in order to satisfy the true continuity equation.

Note that although the higher order flux vectors given by Eqs. (5.11) and (5.15) involve the dependent variable vector at more than one point on either side of a cell face only one point on either side was used in Eq. (7.11). This means that the Jacobian of the first–order flux vectors is used on the left, whereas, higher–order flux vectors are used on the right. There are two reasons for this. Firstly, by using more than one point on either side the band width of the solution matrix (the Jacobian) would increase, requiring more memory and more CPU time to solve the problem. Secondly, experiences gained in using more than one point have indicated that convergence is not improved a great deal when more points are used. For quadratic convergence, all points involved in the flux vectors should be included in the Jacobian. However, quadratic convergence is not obtained here and the approximation made by using only one point on either side of a cell face works rather well.

During the course of this work, another formulation for the Jacobian in Eq. (7.11) was found to work rather well. In fact, it will be demonstrated that at least for the test case considered, it works better than Eq. (7.11). Note from the expressions for the higher–order numerical fluxes given by Eqs. (5.11) and (5.15), that they depend on Q^R and Q^L given, for example, by Eq. (5.14). By forming Q^R and Q^L using Eq. (5.14) and incrementing Q^R and Q^L by $(h\ e_j)$ in the numerical derivative (Eq. (7.8)) one obtains an approximate numerical Jacobian that is computationally more efficient than using a strictly first–order numerical Jacobian. (That is, it is computationally more efficient when a higher–order flux is used on the right–hand–side of the equation.) The reason it is computationally more efficient is that if a higher–order flux is used on the right in the residual term then this flux which has already been computed and is available, can be used in Eq. (7.8) and another first–order flux does not have to be computed as is necessary if a strictly first–order Jacobian is used on the left–hand–side. By using this slightly different numerical Jacobian, the system of equations to be solved that is analogous to Eq. (7.11) is given by

$$- D\overline{G}_{j-\frac{1}{2},L}\, \Delta Q^{n+1,m}_{i,j-1} - D\overline{F}_{i-\frac{1}{2},L}\, \Delta Q^{n+1,m}_{i-1,j}$$

$$+ \left(\frac{I}{\Delta\tau} + D\overline{F}_{i+\frac{1}{2},L} + D\overline{G}_{j+\frac{1}{2},L} - D\overline{F}_{i-\frac{1}{2},R} - D\overline{G}_{j-\frac{1}{2},R}\right) \Delta Q^{n+1,m}_{i,j}$$

$$+ D\overline{F}_{i+\frac{1}{2},R}\, \Delta Q^{n+1,m}_{i+1,j} + D\overline{G}_{j+\frac{1}{2},R}\, \Delta Q^{n+1,m}_{i,j+1}$$

$$= -\left[I_a \frac{Q^{n+1,m}_{i,j} - Q^n_{i,j}}{\Delta\tau} + \delta_i\, \overline{F}\left(Q^{n+1,m}\right) + \delta_j\, \overline{G}\left(Q^{n+1,m}\right)\right] \qquad (7.13)$$

where the first subscript on $D\overline{F}$ and $D\overline{G}$ again refers to the location of the cell face and the second subscript refers to whether the numerical partial derivative is with respect to Q constructed from the right (R) or left (L).

To begin the calculation at each time step, the following is used for the initial $(m = 1)$ $Q^{n+1,m}$

$$Q^{n+1,1} = Q^n \qquad (7.14)$$

Note that when Eq. (7.13) is solved for $m = 1$ only, the time derivative in the residual term is zero. This would correspond to the situation of linearizing only the spatial difference terms in Eq. (7.1).

This approach is used for steady state calculations and for some unsteady calculations when the variations in time are not sever. This results in an approximate first–order time derivative solution.

It was stated in Section 7.1 that the development of the solution process for higher–order in time is the same as for first–order in time. If the second–order backward time derivative given by Eq. (4.5) was used in place of the first–order time derivative given by Eq. (4.4), then the resulting system of equations analogous to Eq. (7.13) would be

$$- D\overline{G}_{j-\frac{1}{2},L}\, \Delta Q_{i,j-1}^{n+1,m} - D\overline{F}_{i-\frac{1}{2},L}\, \Delta Q_{i-1,j}^{n+1,m}$$

$$+ \left(\frac{3I}{2\Delta\tau} + D\overline{F}_{i+\frac{1}{2},L} + D\overline{G}_{j+\frac{1}{2},L} - D\overline{F}_{i-\frac{1}{2},R} - D\overline{G}_{j-\frac{1}{2},R} \right) \Delta Q_{i,j}^{n+1,m}$$

$$+ D\overline{F}_{i+\frac{1}{2},R}\, \Delta Q_{i+1,j}^{n+1,m} + D\overline{G}_{j+\frac{1}{2},R}\, \Delta Q_{i,j+1}^{n+1,m}$$

$$= - \left[I_a \frac{3Q_{i,j}^{n+1,m} - 4Q_{i,j}^{n} + Q_{i,j}^{n-1}}{2\Delta\tau} + \delta_i\, \overline{F}\left(Q^{n+1,m}\right) + \delta_j\, \overline{G}\left(Q^{n+1,m}\right) \right] \quad (7.15)$$

where, again, I_a is an identity matrix except the first diagonal element is zero in order to satisfy the true continuity equation. Equation (7.14) is used in Eq. (7.15) for the initial value of $Q^{n+1,1}$, just as was done for Eq. (7.13). Although it is not quite so easily seen, a solution to Eq. (7.15) for $m = 1$ only, results in an approximate second–order time derivative solution analogous to the first–order time derivative solution resulting from Eq. (7.13) when Eq. (7.14) is used for an $m = 1$ only solution.

Equation (7.13) for first–order in time or Eq. (7.15) for second–order in time constitute the system of equations that must be solved for the dependent variable vector Q^{n+1}. The solution of this system is the subject of the next Section.

7.3 Solution of the System of Equations

The numerical solution of Eq. (7.13) or (7.15) from time level n to time level $n+1$ calls for m iterations. The solution of these equations for each value of m is referred to as a Newton iteration. For unsteady flows, values of m from *1 to 5* are typically used. The case of $m = 1$ is sometimes referred to as no Newton iterations since the equations solved are the same as those that would be solved if only the spatial difference terms of Eq. (7.1) were linearized.

Note that the numerical solution of Eq. (7.13) or (7.15) for each Newton iteration amounts to solving a linear system of equations that is represented here as the linear algebra problem

$$A\,x = b \tag{7.16}$$

where x in this case is the change in Q^{n+1} from iteration *m to m+1*, A is the Jacobian at iteration *m*, and b is the residual vector at iteration *m*. Matrix A is a blocked banded matrix. The blocks are square matrices that are *3x3* in this case since there are three dependent variables at each point. Note from Eq. (7.13) or (7.15) that the Jacobian is evaluated at each *m* iteration. However, because x goes to zero (in principle) as *m* increases, the Jacobian has no influence on the converged solution so long as the Jacobian that is used will work. Therefore, in practice, the Jacobian is not updated at each Newton iteration in order to save CPU time. The solution process discussed below is appropriate whether or not the Jacobian A is updated after each Newton iteration.

Write the matrix A in the form

$$A = L + B + U \tag{7.17}$$

where L is a lower block triangular matrix with zeros on the diagonal, B is a block diagonal matrix, and U is an upper block triangular matrix with zeros on the diagonal. The first two terms of Eq. (7.13) or (7.15) contribute the elements of *L,* the third term contributes the elements of *B,* and the fourth and fifth terms contribute the elements of *U.* The equation to solve is

$$(L + B + U)\,x = b \tag{7.18}$$

Equation (7.18) is frequently solved in the CFD community by various methods of factoring, where the most commonly used method is probably that of Briley and McDonald [21]. The approach used here will be relaxation [22]. In many respects various factored schemes are similar to relaxation schemes, and some of these similarities are discussed in [11, 15, and 23].

Equation (7.18) is solved using the symmetric block Gauss–Seidel iterative method [24]. This method performs a forward sweep through the computational domain by

$$(L + B)\, x^{(1)} + U\, x^{(0)} = b \tag{7.19}$$

followed by a backward sweep

$$L\, x^{(1)} + (B + U)\, x^{(2)} = b \tag{7.20}$$

This forward and backward iterative process can be written as

$$(L + B)\, x^{(2p-1)} + U\, x^{(2p-2)} = b \tag{7.21a}$$

$$L\, x^{(2p-1)} + (B + U)\, x^{(2p)} = b \tag{7.21b}$$

where $p = 1, 2, 3, \ldots$ Therefore, there are $2p$ sweeps through the computational domain, half of them forward and half of them backward, for p complete symmetric block Gauss–Seidel iterations.

To illustrate rather specifically the solution process, consider Eq. (7.21a) in the expanded form

$$\begin{aligned} & L_{i,j-1}\, x_{i,j-1}^{(2p-1)} + L_{i-1,j}\, x_{i-1,j}^{(2p-1)} + B_{i,j}\, x_{i,j}^{(2p-1)} \\ & + U_{i+1,j}\, x_{i+1,j}^{(2p-2)} + U_{i,j+1}\, x_{i,j+1}^{(2p-2)} = b_{i,j} \end{aligned} \tag{7.22}$$

In Eq. (7.22) each subscripted matrix L and U is a block (*3x3* matrix in this case) and each subscripted vector x and b is a subvector (*3* element vector in this case). Note that when one moves forward through the computational domain, then at an *i,j* location every subvector x behind this point has already been solved for and Eq. (7.22) can be written as

$$B_{i,j}\, x_{i,j}^{(2p-1)} = b_{i,j} - L_{i,j-1}\, x_{i,j-1}^{(2p-1)} - L_{i-1,j}\, x_{i-1,j}^{(2p-1)} - U_{i+1,j}\, x_{i+1,j}^{(2p-2)} - U_{i,j+1}\, x_{i,j+1}^{(2p-2)} \qquad (7.23a)$$

In an analogous fashion, for the backward sweep given by Eq. (7.21b) one can solve the following equation for $x^{(2p)}$

$$B_{i,j}\, x_{i,j}^{(2p)} = b_{i,j} - L_{i,j-1}\, x_{i,j-1}^{(2p-1)} - L_{i-1,j}\, x_{i-1,j}^{(2p-1)} - U_{i+1,j}\, x_{i+1,j}^{(2p)} - U_{i,j+1}\, x_{i,j+1}^{(2p)} \qquad (7.23b)$$

where $p = 1, 2, 3, \ldots$

Although the overall solution process is an iterative method, the numerical solution of x at point i,j in Eqs. (7.23a) and (7.23b) is carried out using the direct method of Doolittle, which is a compact scheme for Gaussian elimination [25]. Therefore, the overall process is iterative but the solution of the subsystems at each point i,j is direct.

In the numerical solution of the linear algebra problem given by Eq. (7.16) it usually is impractical to consider the solution in the form

$$x = A^{-1}\, b \qquad (7.24)$$

where the coefficient matrix A is inverted. The same is true here with regard to solving Eq. (7.16) because of the size of A involved in this problem. However, when a matrix is much smaller, like B is in the direct solution of Eq. (7.23a) and (7.23b), it may be of benefit to consider inverting the B block coefficient matrices. One is motivated to investigate this approach even more when it is noted that the Jacobian matrix, and hence the B block matrices, do not have to be updated for each Newton iteration, and certainly the B's are not updated during the symmetric block Gauss–Seidel iterations. It turns out for the solution of these two–dimensional equations, it pays to invert the block matrices *B*, which are only *3x3* in this case, and solve the equations for *x* using the equations

$$x_{i,j}^{(2p-1)} = \bar{b}_{i,j} - \bar{L}_{i,j-1}\, x_{i,j-1}^{(2p-1)} - \bar{L}_{i-1,j}\, x_{i-1,j}^{(2p-1)} - \bar{U}_{i+1,j}\, x_{i+1,j}^{(2p-2)} - \bar{U}_{i,j+1}\, x_{i,j+1}^{(2p-2)} \tag{7.25a}$$

$$x_{i,j}^{(2p)} = \bar{b}_{i,j} - \bar{L}_{i,j-1}\, x_{i,j-1}^{(2p-1)} - \bar{L}_{i-1,j}\, x_{i-1,j}^{(2p-1)} - \bar{U}_{i+1,j}\, x_{i+1,j}^{(2p)} - \bar{U}_{i,j+1}\, x_{i,j+1}^{(2p)} \tag{7.25b}$$

where $p = 1, 2, 3, \ldots$, and

$$\bar{b}_{i,j} = B_{i,j}^{-1}\, b_{i,j} \tag{7.26a}$$

$$\bar{L}_{i,j-1} = B_{i,j}^{-1}\, L_{i,j-1} \tag{7.26b}$$

$$\bar{L}_{i-1,j} = B_{i,j}^{-1}\, L_{i-1,j} \tag{7.26c}$$

$$\bar{U}_{i+1,j} = B_{i,j}^{-1}\, U_{i+1,j} \tag{7.26d}$$

$$\bar{U}_{i,j+1} = B_{i,j}^{-1}\, U_{i,j+1} \tag{7.26e}$$

Even though the Jacobian is not updated, in general, for each Newton iteration, the residual vector is updated after each Newton iteration. This means that for each Newton iteration a new $\bar{b}$ must be obtained from Eq. (7.26a). This, however, involves only a matrix–vector multiply and not a matrix–matrix multiply as is required for $\bar{L}$ and $\bar{U}$, which do not have to be recomputed. By solving Eqs. (7.25a) and (7.25b) in place of Eq. (7.23a) and (7.23b) a saving of from zero to a factor of two is obtained depending on the number of Newton iterations and symmetric block Gauss–Seidel iterations that are used. Part of this saving is due to the way the *B's* are inverted. The approach presently used is to first perform a decomposition of *B* using Doolittle's method. Then invert the two triangular matrices resulting from the decomposition, and finally obtain the inverse of *B* by multiplying the two inverted triangular matrices in reverse order [25]. There is no penalty in storage in using the inverse B method given by Eqs. (7.25a) and (7.25b) compared to using Eqs. (7.23a) and (7.23b). Experiences gained thus far, indicate that solving Eqs. (7.25a) and (7.25b) is the preferred approach for this two–dimensional problem.

VIII. Results

The numerical approach presented here differs somewhat from that presented in [1, 2, and 11]. The differences include: (1) the numerical flux formulation at a cell face, (2) the boundary conditions, (3) as a consequence of (1) the numerical Jacobian is different, and moreover, it is different from what one would probably consider to be the customary approach, and (4) the use of the inverse of the diagonal block matrices in the solution of the linear system. Numerical results addressing each of these changes are given that illustrate the present approach yields results as good or better as before, at a reduced computational cost. In addition, a time–dependent solution will be presented and compared to a known theoretical solution to demonstrate the accuracy of the method for unsteady computations.

All numerical solutions presented are third–order accurate in space which corresponds to $\varkappa = 1/3$ in the construction of the numerical flux vector as discussed in Chapter V. With the exception of the time–dependent solution, all numerical results were carried out for a NACA 0012 airfoil with a 161x35 O–type grid. Local time stepping was used for all the steady state solutions. Also, all numerical computations were carried out on an IBM RS/6000 Model 560 in 64–bit arithmetic using the compilation command "xlf –qautodbl=dblpad –O –bmaxdata:0x3e800000".

8.1 Numerical Flux

The numerical flux formulation used in the past [2] was never given a name. In formulation, it falls somewhere between flux extrapolation [26] and MUSCL [9] as explained in [8 and 11]. A comparison of the MUSCL approach used here and the no–name flux formulation [8] are compared in Fig. 3. For this incompressible flow computation, one will notice that the two flux formulations lead to indistinguishable results (except for the region extremely close to the trailing edge where MUSCL produce a slightly larger (more positive) pressure). In compressible flow where limiters are used, this close agreement between the two numerical flux formulations is not the case [8]. The incompressible results given in Fig. 3 are typical of the results obtained to date.

8.2 Boundary Conditions

The modification presented in Chapter VI is actually only slightly different from the way boundary conditions have been treated in the past [1 and 2]. The difference in the two methods is that previously phantom points were used and the boundary conditions were determined on the boundaries using characteristic variable boundary conditions and then extrapolated to the phantom points [12]. In the new method, the boundary conditions are still determined on the boundaries using the same characteristic variable approach, but they are not extrapolated to phantom points, rather they are retained on the boundaries. This slight modification had an influence on the results in the following sense. The converged solutions of the two approaches are essentially indistinguishable as shown in Fig. 4, but the convergence rate was changed. Consider the top two curves in Fig. 5 represented by the dashed curve and long–dashed curve. The dashed curve is the result from using the old boundary conditions of [1], where ISGS is the number of symmetric block Gauss–Seidel iterations, IFREQ represents how often the flux Jacobians are updated, and CFL is, of course, the CFL number. Note there is some improvement in the convergence rate brought about by using this slight modification in the boundary condition treatment. This new boundary condition treatment however, allows the CFL number to be increased considerably as illustrated by the solid curve in Fig. 5. This increase comes at the expense of having to increase ISGS, in this case from 5 to 15. This, unfortunately, increases the CPU time required per iteration. Notice also that the flux Jacobians were updated after every iteration (IFREQ=1) with the new boundary conditions for a CFL number of 1000. This also increased the CPU time relative to updating them infrequently. However, even with this increase in computational work per cycle the end result is a savings in CPU time. For example, the solid line in Fig. 5 takes 79 CPU seconds to reach machine zero, whereas the top dashed line in Fig. 5 takes 109 CPU seconds to reach machine zero. The updating of the flux Jacobians every cycle is, in general, not required. By updating the flux Jacobians every cycle for the first 10 cycles and then updating every twenty cycles afterwards, the convergence rate of the solid curve with CFL=1000 is essentially the same and it takes only 38 CPU seconds to reach machine zero. The dash–dot curve in Fig. 5 for the old boundary condition treatment with ISGS=15, IFREQ=1, and CFL=40 is included to show that the convergence rate is improved relative to using ISGS=5 and IFREQ=20. However, even with this extra computational work due to updating Jacobians every cycle, the CFL number could not be increased with the old boundary condition treatment as it could with the new boundary condition treatment. The CFL numbers used in Fig. 5 are near optimum for each boundary condition treatment.

Obviously the new method of handling the boundary conditions is an improvement over the previous method, and the improved convergence brought about by the capability to handle a larger CFL number is attractive. However, the real pay–off for the capability to handle large CFL numbers is, oddly enough, probably greater for unsteady flow than it is for steady state flow. The reason for this is that it is always desirable to have the time step restricted by the physics of the problem and not the numerics of the solver. This point is discussed in some detail in [27].

Besides increasing the CFL number and improving the convergence rate of the numerical solutions, this new boundary condition treatment has recently been beneficial in another regard. In using a Navier–Stokes version of this numerical approach [1] to compute the flow in the respiratory system where there is the problem of handling branching, Gatlin has encountered the problem of the flow oscillating considerably among the branches during the transient process of obtaining a steady state solution. By using this new boundary condition treatment at the outflow boundary only, the solution process was accelerated considerably [28].

8.3 Numerical Jacobian

The solution matrix is usually the Jacobian of a first–order numerical flux vector, even when a higher–order numerical flux vector is used in the residual vector. The reason for this is to reduce the size of the solution matrix and consequently save storage and usually CPU time. The Jacobian used here is developed in Section 7.2. It is not constructed from a first–order numerical flux (unless a first–order numerical flux is used in the residual), rather it takes into account some flavor of the higher–order numerical flux vector. The reason for trying this approach was due to simplicity and reduced operation count, whereas, the reason for keeping this approach is due to the fact it has been successful. In fact, the convergence rates experienced in all cases considered thus far have been superior to the convergence rates obtained with a Jacobian corresponding to a first–order numerical flux vector. To illustrate this, consider Fig. 6 which is a comparison of results from the method used to obtain the discretized Jacobian presented in Section 7.2 with the results from using the discretized Jacobian obtained from a first–order numerical flux vector. Obviously the convergence rate of the present method is superior. In this case improved results were obtained at a reduced operation count. The reason for this is that a first–order numerical flux does not have to be computed in addition to the higher–order numerical flux in order to obtain the discretized Jacobian.

8.4 Inverse of Diagonal Blocks

This method of solving the linear system of equations was developed in Section 7.3. In principle it should produce the same numerical results as solving the equations without taking the inverse of the diagonal blocks as has been the practice in the past [11]. All numerical solutions have been the same whether or not the inverse is taken. However, there has been some differences noticed in the residual histories when the numbers become small, but this might be due to the different arithmetic operations involved in obtaining a solution and the 64–bit accuracy of the computation.

Timing results of numerical experiments carried out for different values of ISGS (number of symmetric block Gauss–Seidel iterations) and IFREQ (number of cycles taken before the numerical

Jacobians are updated) are presented in Fig. 7. The times in Fig. 7 are for 100 cycles and all of the solutions from which the results of Fig. 7 were extracted were not necessarily converged. These results were obtained simply to illustrate the difference in CPU time brought about by using or not using the inverse of the diagonal blocks. Three values of IFREQ were considered in Fig. 7. Updating the Jacobians every time step corresponds to IFREQ=1 and, of course, this is the most computationally intense case. For IFREQ=1 it does not pay to use the inverse of the diagonal blocks for an ISGS less than 3 as illustrated by the fact that the solid and dashed curves, with circles, cross at this point in Fig. 7. However, for an IFREQ of 9999 (since these solutions were run for 100 cycles an IFREQ of 9999 simply means that the Jacobians were computed at the first cycle and then never recomputed (frozen) thereafter) Fig. 7 indicates it pays to use the inverse of the diagonal blocks for all values of ISGS. Also included in Fig. 7 is a set of curves for IFREQ=5 where, like for IFREQ=9999, it pays to use the inverse for all values of ISGS. Figure 7 illustrates that by freezing the Jacobians for as few as 5 steps (IFREQ=5) there is a large savings in CPU time compared to updating the Jacobians every time (IFREQ=1). Typically steady state solutions are run with an IFREQ of 20. A curve is not shown for this, but it is only a small increase above the IFREQ=9999 curves. An IFREQ of 10 or 20 seems to be sufficient, even for Navier–Stokes computations of complex flows including separation, with extremely tight grids (like maximum y–plus values of 1 and less [29]). For steady state solutions of complex problems it is usually not required to update the Jacobians very often after the first few cycles as mentioned in Section 8.2 above, so the times corresponding to the IFREQ=9999 (Jacobian frozen) curves can be achieved in practice.

Obviously an advantage of taking the inverse of the diagonal blocks is that the number of symmetric block Gauss–Seidel iterations can be increased without as much increase in CPU time as that required by not taking the inverse. This, of course, is illustrated in Fig. 7 by the difference in slopes of the solid and dashed curves.

The reader is cautioned that the timings above are for two dimensions, and using the inverse of the block diagonals may or may not prove to be as advantageous for three dimensions as it has proven to be here for two dimensions.

8.5 Numerical Solution

Thus far attention has been focused on speed, storage, and convergence rate. The test case considered is rather simple, but the quality of the solution is yet to be demonstrated. To investigate this, the numerical solution was compared with a potential flow solution of Bernard [30] in Fig. 8. As shown in Fig. 8, the present Euler solution and the potential flow solution [30] are in extremely good agreement, except just at the trailing edge of the airfoil where the potential flow solution goes to a pressure coefficient of unity whereas the finite volume Euler solution does not. There is no cell volume centered at the trailing edge of the airfoil in this finite volume formulation and, therefore, the finite volume Euler solution does not recompress to a pressure coefficient of exactly unity as the node based potential flow solution does. The Euler solution shown in Fig. 8 is considered good.

8.6 Unsteady Cascade Solution

To verify an unsteady solution that uses the eigensystem for the time varying curvilinear coordinate system given in Appendix A, an unsteady solution was obtained for a cascade of infinitely thin, unstaggered blades oscillating in plunge and in–phase at a reduced frequency based on semi–chord of unity. This test case was selected because the theory of Smith [31] can be compared to the present numerical solution. The first harmonic is compared with Smith's theory in Fig. 9, and the comparison is considered good. The grid used was an H–type 121x51 grid with 100 evenly spaced points on the blades and the 51 points from blade–to–blade were evenly spaced. A complete period of motion used 500 time steps and the numerical solution presented was from the fifth period of motion. This number of grid points, time steps per period, and number of periods run was over–kill, but the objective was to verify the unsteady computational capability of the method and not to try and minimize any of these parameters. The reason this computation is considered over–kill is based on experience in performing computations for this configuration in compressible flow with a similar numerical approach [32].

IX. Concluding Remarks

This report describes the most recent computational technology used to solve the two–dimensional time–dependent incompressible Euler equations. Much of this same technology is used in the three–dimensional time–dependent laminar or turbulent, single or multiblock, stationary or dynamic grid, incompressible Navier–Stokes equation codes currently being developed in the CFD Lab. Consequently, the material learned will be of use for solving more geometrically and physically challenging problems. The report includes considerable detail in certain areas and it is anticipated that it might be a helpful aid for teaching some aspects of computational fluid dynamics.

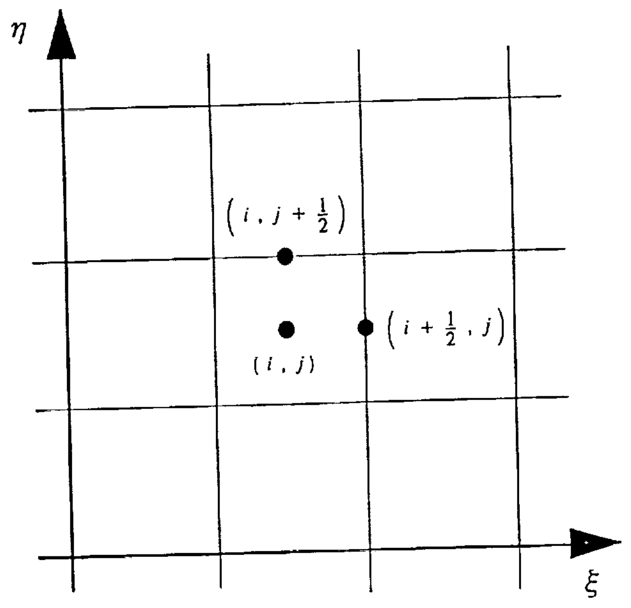
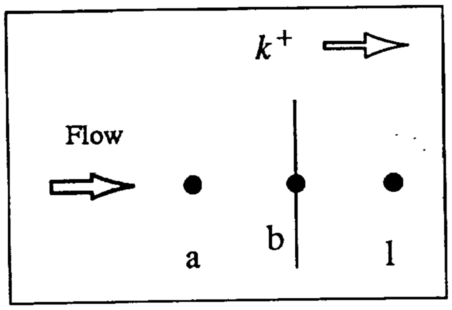
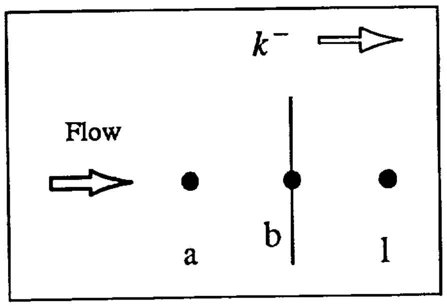

η
$\left(i, j+\frac{1}{2}\right)$
$\left(i+\frac{1}{2}, j\right)$
(i, j)
ξ

Fig. 1 Index Notation





$\lambda_{k,1} > 0$;	$(w_{k,1})_a = (w_{k,1})_b$
$\lambda_{k,2} > 0$;	$(w_{k,2})_a = (w_{k,2})_b$
$\lambda_{k,3} < 0$;	$(w_{k,3})_l = (w_{k,3})_b$

2a. Codirectional

$\lambda_{k,1} < 0$;	$(w_{k,1})_a = (w_{k,1})_b$
$\lambda_{k,2} > 0$;	$(w_{k,2})_l = (w_{k,2})_b$
$\lambda_{k,3} < 0$;	$(w_{k,3})_a = (w_{k,3})_b$

2b. Contradirectional

Fig. 2 Boundary Condition Terminology

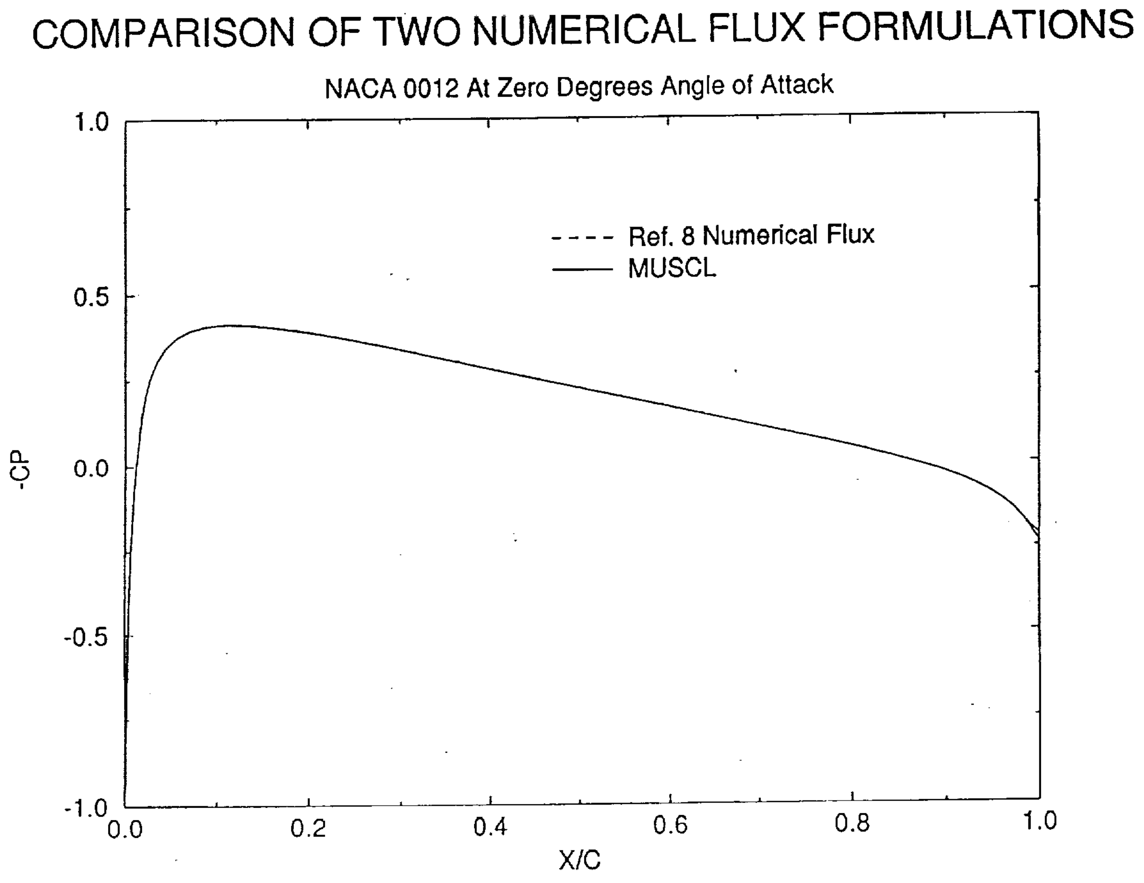


Fig. 3 Surface Pressure Distributions for the NACA 0012 at Zero Degrees Angle of Attack from Two Solutions with Different Numerical Flux Formulations

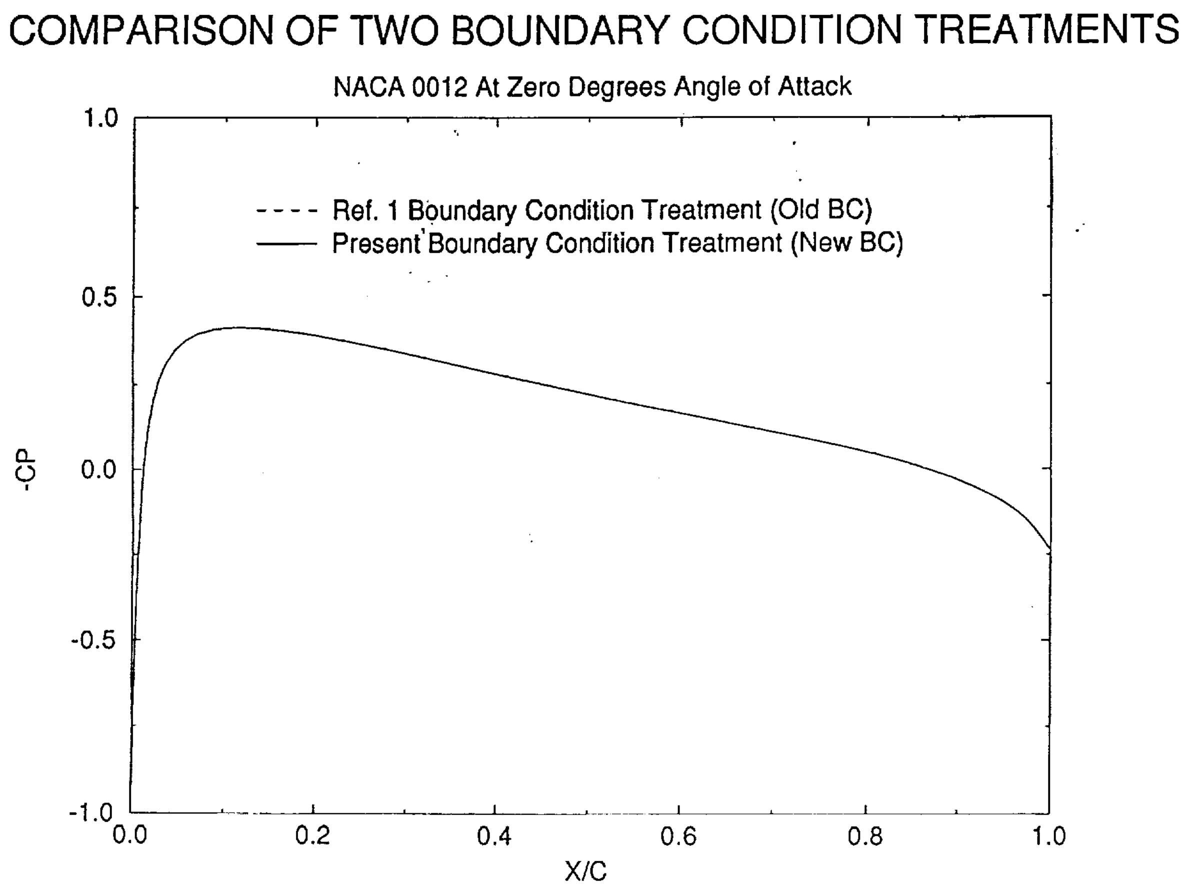


Fig. 4 Surface Pressure Distributions for the NACA 0012 at Zero Degrees Angle of Attack from Two Solutions with Different Boundary Condition Treatments

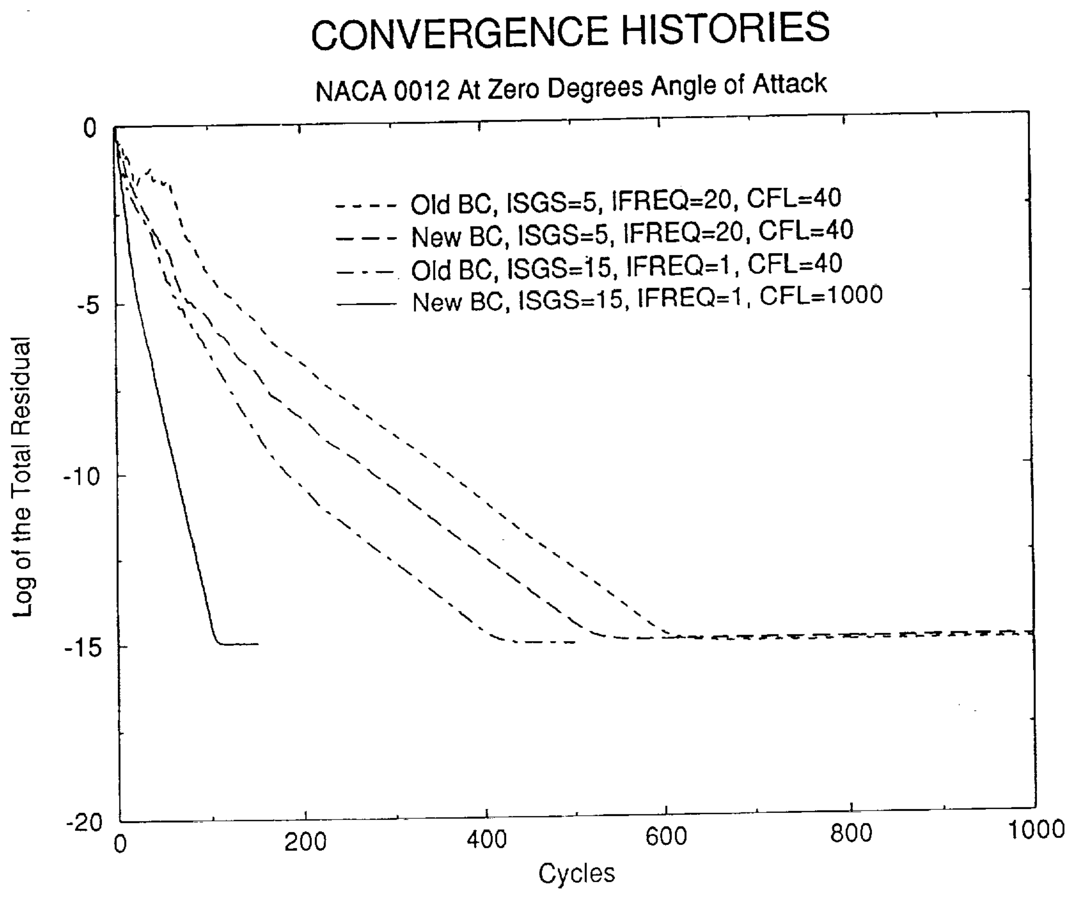


Fig. 5 Convergence Histories for Different Boundary Condition Treatments

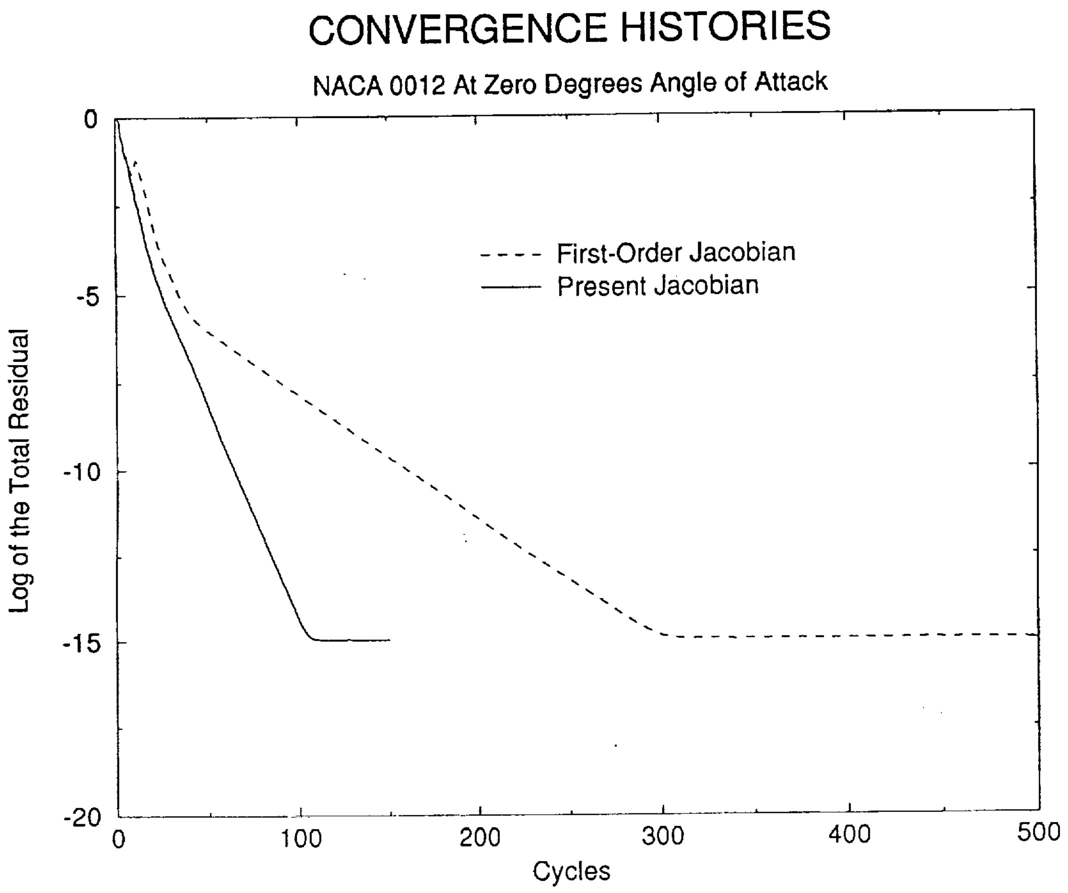


Fig. 6 Convergence Histories for Different Discretized Jacobians

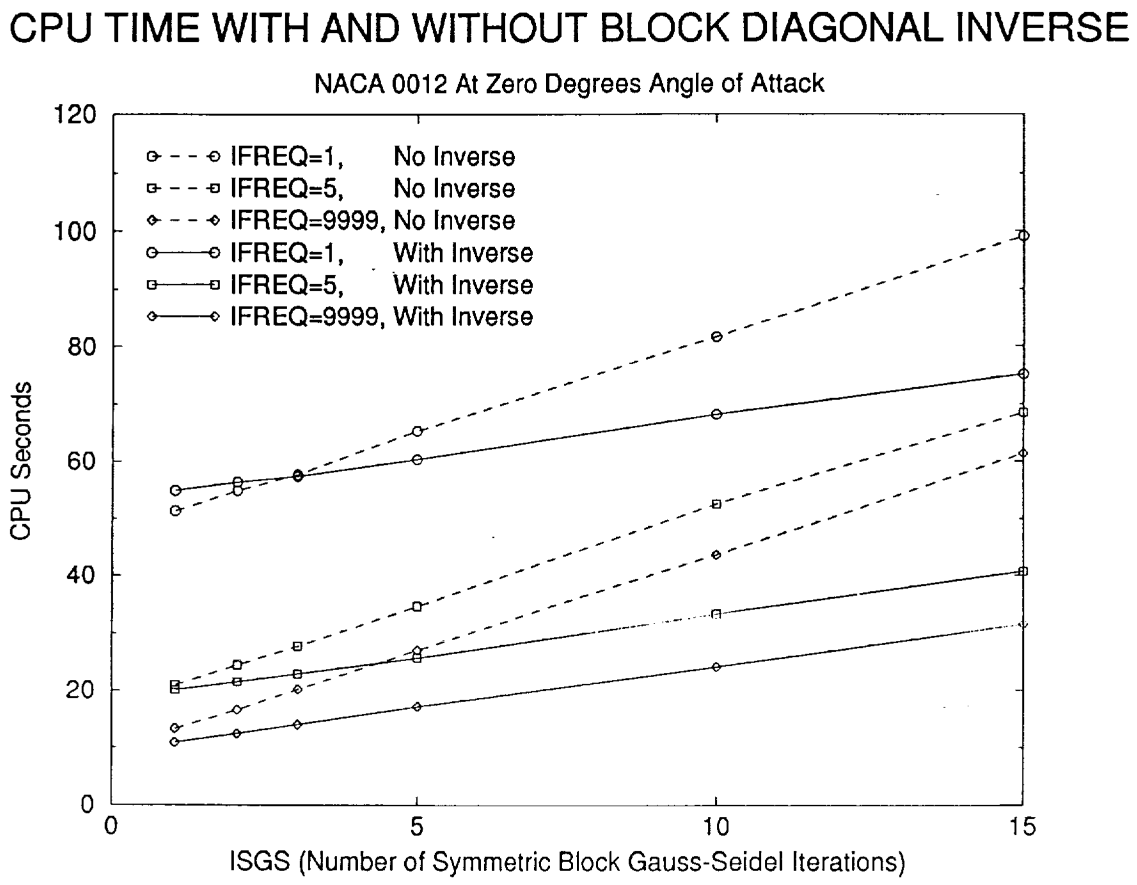


Fig. 7 Computational Times With and Without the Use of the Block Diagonal Inverse

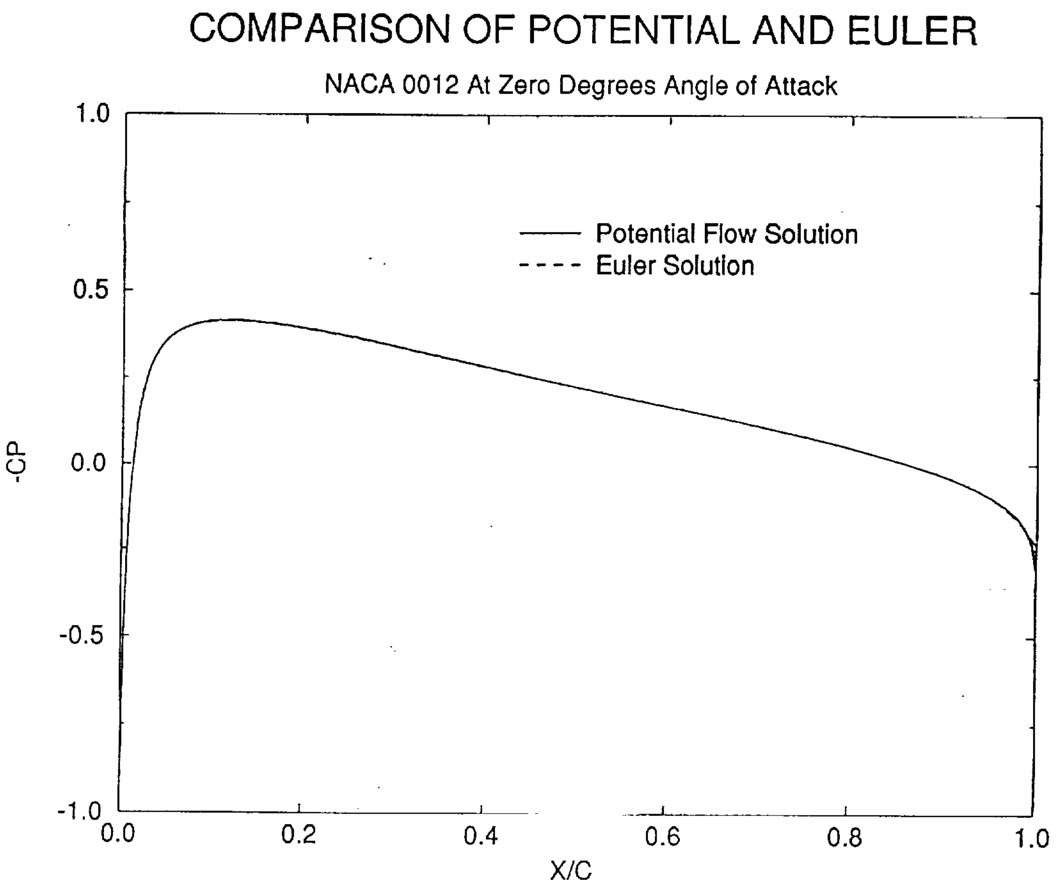


Fig. 8 Comparison of Euler and Potential Flow Solutions
for an NACA 0012 Airfoil at Zero Degrees of Attack

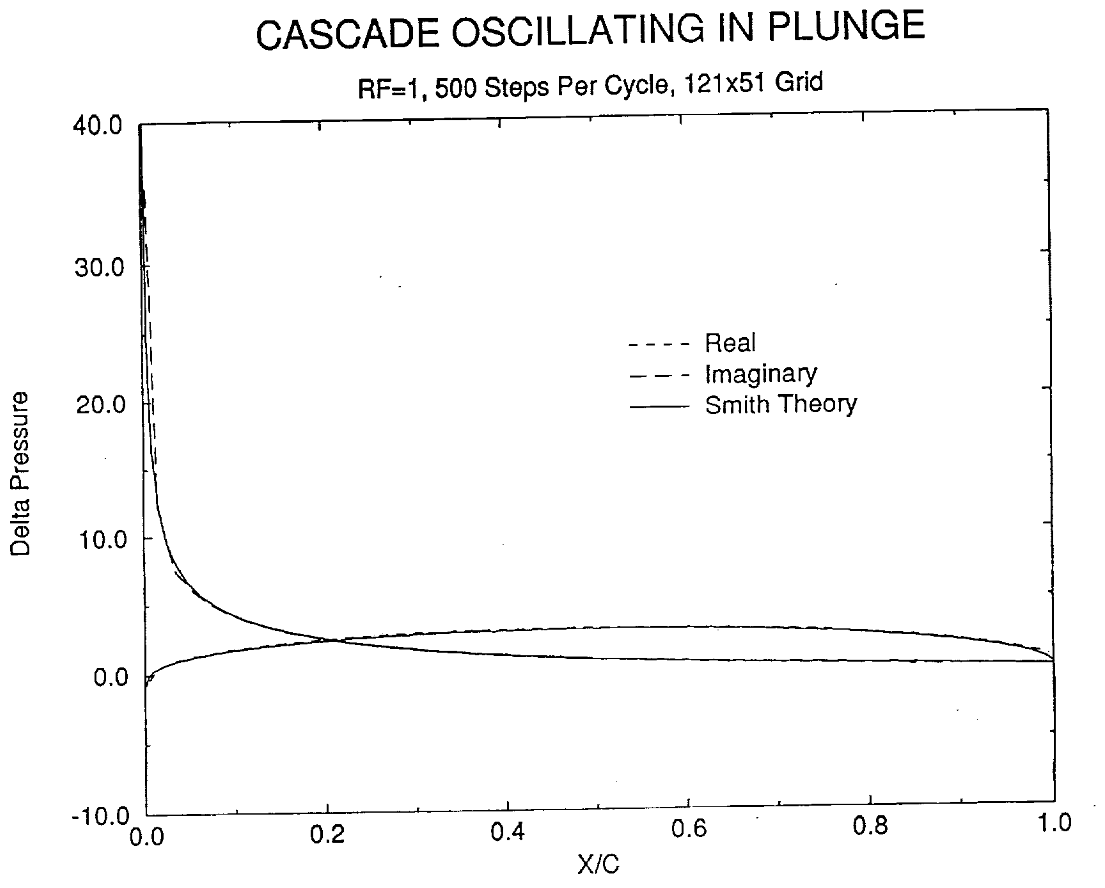


Fig. 9 Unsteady Results for a Cascade of Unstaggered Infinitely Thin Blades Oscillating in Plunge and In–Phase at a Reduced Frequency of Unity

Appendix A

Eigensystem of the Two–Dimensional Unsteady Incompressible Euler Equations in Time–Dependent Curvilinear Coordinates

The three–dimensional time–dependent incompressible Euler equations in Cartesian coordinates were transformed to time–dependent curvilinear coordinates in Ref. 1. After the equations were transformed to curvilinear coordinates they were written in Chorin's [3] artificial compressibility form as explained in Ref. 1. Following this same approach, but for the more simple case of two dimensions rather than three dimensions, the artificial compressibility form of the two–dimensional time–dependent incompressible Euler equations in time–dependent curvilinear coordinates is

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0 \tag{A.1}$$

where

$$Q = J \begin{bmatrix} p \\ u \\ v \end{bmatrix} , \quad F = J \begin{bmatrix} \beta\,(U - \xi_t) \\ u\,U + \xi_x\,p \\ v\,U + \xi_y\,p \end{bmatrix} , \quad G = J \begin{bmatrix} \beta\,(V - \eta_t) \\ u\,V + \eta_x\,p \\ v\,V + \eta_y\,p \end{bmatrix}$$

$$U = \xi_x\,u + \xi_y\,v + \xi_t$$

$$V = \eta_x\,u + \eta_y\,v + \eta_t$$

$$\xi = \xi(x,y,t)$$

$$\eta = \eta(x,y,t)$$

$$\tau = t$$

$$J = \left| \frac{\partial(x,y)}{\partial(\xi,\eta)} \right| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} = x_\xi\,y_\eta - x_\eta\,y_\xi$$

$$
\begin{aligned}
\xi_x &= J^{-1}\, y_\eta \\
\xi_y &= -\, J^{-1}\, x_\eta \\
\eta_x &= -\, J^{-1}\, y_\xi \\
\eta_y &= J^{-1}\, x_\xi
\end{aligned}
$$

and β = constant. Notice that flux vectors F and G can be written

$$K = J \begin{bmatrix} \beta\,(\theta_k - k_t) \\ u\theta_k + k_x\, p \\ v\theta_k + k_y\, p \end{bmatrix} \tag{A.2}$$

where

$$\theta_k = k_x\, u + k_y\, v + k_t \tag{A.3}$$

and

$$
\begin{aligned}
&K = F \quad \text{and} \quad \theta_k = U \quad \text{for} \quad k = \xi \\
&K = G \quad \text{and} \quad \theta_k = V \quad \text{for} \quad k = \eta
\end{aligned}
$$

The quasilinear form of Eq. (A.1) is

$$\frac{\partial Q}{\partial \tau} + A\frac{\partial Q}{\partial \xi} + B\frac{\partial Q}{\partial \eta} = 0 \tag{A.4}$$

where

$$A = \frac{\partial F}{\partial Q}\,, \quad B = \frac{\partial G}{\partial Q} \tag{A.5}$$

The Jacobian matrices A and B are denoted by $\overline{K}$ where the elements of $\overline{K}$ are given by

$$\overline{K}_{ij} = \frac{\partial K_i}{\partial Q_j} \tag{A.6}$$

Therefore,

$$K = F \text{ and } \overline{K} = A \quad \text{when} \quad k = \xi$$

and

$$K = G \text{ and } \overline{K} = B \quad \text{when} \quad k = \eta$$

The flux vectors represented by Eq. (A.2) can be written in terms of the elements of the dependent variable vector $Q = [\, Jp,\ Ju,\ Jv\,]^T = \left[\, Q_1,\ Q_2,\ Q_3 \right]^T$ as

$$K = \begin{bmatrix} \beta\,(k_x\, Q_2 + k_y\, Q_3) \\ \frac{1}{J}\left[Q_2(k_x\, Q_2 + k_y\, Q_3 + k_t)\right] + k_x\, Q_1 \\ \frac{1}{J}\left[Q_3(k_x\, Q_2 + k_y\, Q_3 + k_t)\right] + k_y\, Q_1 \end{bmatrix} \tag{A.7}$$

The elements of the Jacobian matrix $\overline{K}$ are determined from Eq. (A.6) resulting in

$$\overline{K} = \begin{bmatrix} 0 & \beta\, k_x & \beta\, k_y \\ k_x & \theta_k + k_x\, u & k_y\, u \\ k_y & k_x\, v & \theta_k + k_y\, v \end{bmatrix} \tag{A.8}$$

The eigenvalues of $\bar{K}$ are given by

$$\lambda_1 = \theta_k \tag{A.9a}$$

$$\lambda_2 = \theta_k - c^- \tag{A.9b}$$

$$\lambda_3 = \theta_k - c^+ \tag{A.9c}$$

where

$$c^+ = \frac{k_t}{2} + c \tag{A.10a}$$

$$c^- = \frac{k_t}{2} - c \tag{A.10b}$$

$$c = \left[\left(\theta_k - \frac{k_t}{2}\right)^2 + \beta\left(k_x^2 + k_y^2\right)\right]^{\frac{1}{2}} \tag{A.10c}$$

and θ_k is given by Eq. (A.3). Therefore,

$$\Lambda_k = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} \theta_k & 0 & 0 \\ 0 & \theta_k - c^- & 0 \\ 0 & 0 & \theta_k - c^+ \end{bmatrix} \tag{A.11}$$

where the subscript k corresponds to $k = \xi \; or \; k = \eta$. The corresponding right eigenvectors of $\bar{K}$ are given by the columns of

$$R_k = \begin{bmatrix} 0 & -c^- & -c^+ \\ -k_y & \frac{u\lambda_2}{\beta} + k_x & \frac{u\lambda_3}{\beta} + k_x \\ k_x & \frac{v\lambda_2}{\beta} + k_y & \frac{v\lambda_3}{\beta} + k_y \end{bmatrix} \tag{A.12}$$

The corresponding left eigenvectors of $\bar{K}$ are given by the rows of

$$R_k^{-1} = -\frac{\beta}{2cc^+c^-}\begin{bmatrix} \frac{2c}{\beta}\left(uk_y - vk_x\right) & -2c\left(\frac{v\theta_k}{\beta} + k_y\right) & 2c\left(\frac{u\theta_k}{\beta} + k_x\right) \\ -\frac{c^+\lambda_3}{\beta} & c^+k_x & c^+k_y \\ \frac{c^-\lambda_2}{\beta} & -c^-k_x & -c^-k_y \end{bmatrix} \tag{A.13}$$

The eigensystem of $\overline{K}$ is therefore composed of the eigenvalues given by Eqs.(A.9), the right eigenvectors given as the columns of the matrix R_k (Eq.(A.12)), and the left eigenvectors given as the rows of the matrix R_k^{-1} (Eq.(A.13)). Note that each eigenvector corresponds to one particular eigenvalue. The Jacobian matrices are now diagonalized by

$$R_k^{-1}\,\overline{K}\,R_k = \Lambda_k \tag{A.14}$$

References

1. Taylor, L.K., "Unsteady Three–Dimensional Incompressible Algorithm Based on Artificial Compressibility," Ph.D. Dissertation, Mississippi State University, May 1991.

2. Taylor, L.K. and Whitfield, D.L., "Unsteady Three–Dimensional Incompressible Euler and Navier–Stokes Solver for Stationary and Dynamic Grids," AIAA Paper No. 91–1650, June 1991.

3. Chorin, A.J., "A Numerical Method for Solving Incompressible Viscous Flow Problems," Journal of Computational Physics, Vol.2, 1967, pp. 12–26.

4. Horn, R.A. and Johnson, C.R. Matrix Analysis. Cambridge University Press, New York, 1991.

5. MacCormack, R.W. and Paullay, A.J., "Computational Efficiency Achieved by Time Splitting of Finite Difference Operators," AIAA Paper No. 72–154, 1972.

6. Godunov, S.K., "Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics," Mat. Sbornik, Vol. 47, No. 3, 1959, pp. 271–306. Translated as JPRS 7225 by US Department of Commerce, November 1960.

7. Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," Journal of Computational Physics, Vol. 43, 1981, pp. 357–372.

8. Whitfield, D.L., Janus, J.M., and Simpson, L.B., "Implicit Finite Volume High Resolution Wave–Split Scheme for Solving the Unsteady Three–Dimensional Euler and Navier–Stokes Equations on Stationary or Dynamic Grids," Engineering and Industrial Research Station Report MSSU–EIRS–ASE–88–2, Mississippi State University, Mississippi State, MS, February 1988.

9. van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov's Method," Journal of Computational Physics, Vol. 32, 1979, pp. 101–136.

10. Anderson, W.K., Thomas, J.L., and van Leer, B., "Comparison of Finite Volume Flux Vector Splittings for the Euler Equations," AIAA Journal, Vol. 24, No. 9, September 1986, pp. 1453–1460.

11. Whitfield, D.L. and Taylor, L.K., "Discretized Newton–Relaxation Solution of High Resolution Flux–Difference Split Schemes," AIAA Paper No. 91–1539, June 1991.

12. Whitfield, D.L. and Janus, J.M., "Three–Dimensional Unsteady Euler Equations Solutions Using Flux Vector Splitting," AIAA Paper No. 84–1552, June 1984.

13. Loper, Fran, Private Communication, Arnold Air Force Station, TN, 1978.

14. Janus, J.M., "The Development of a Three–Dimensional Split Flux Vector Euler Solver with Dynamic Grid Applications," M.S. Thesis, Mississippi State University, August 1984.

15. Whitfield, D.L., "Newton–Relaxation Schemes for Nonlinear Hyperbolic Systems," Engineering and Industrial Research Station Report MSSU–EIRS–ASE–90–3, Mississippi State University, Mississippi State, MS, October 1990.

16. Ortega, J.M. and Rheinboldt, W.C. Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, Inc., New York, 1970.

17. Vanden, K.J. and Whitfield, D.L., "Direct and Iterative Algorithms for the Three–Dimensional Euler Equations," AIAA Paper No. 93–3378, 11th Computational Fluid Dynamics Conference, July 1993.

18. Belk, D.M. and Whitfield, D.L., "Three–Dimensional Euler Solutions on Blocked Grids Using an Implicit Two–Pass Algorithm," AIAA Paper No. 87–0450, Januarty 1987.

19. Janus, J.M., "Advanced 3–D CFD Algorithm for Turbomachinery," Ph.D. Dissertation, Mississippi State University, May 1989.

20. Chen, J.P. and Whitfield, D.L., "Navier–Stokes Calculations for the Unsteady Flowfield of Turbomachinery," AIAA Paper No. 93–0676, January 1993.

21. Briley, T.R. and McDonald, H., "Solution of the Multidimensional Compressible Navier–Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, 1977, pp. 372–397.

22. Chakravarthy, S., "Relaxation Methods for Unfactored Implicit Schemes," AIAA Paper No. 84–0165, 1984.

23. Janus, J.M., Arabshahi, A., and Whitfield, D.L., "Numerical Solution and Algorithm Analysis for the Unsteady Navier–Stokes Equations on Dynamic Multiblock Grids," Volume I, Wright Laboratory Report, WL–TR–92–7044, October 1992.

24. Hageman, L.A. and Young, D.M. Applied Iterative Methods. Academic Press, Inc., New York, 1981.

25. Dahlquist, G. and Björck, A. Numerical Methods. Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1974.

26. Osher, S. and Chakravarthy, S.R., "Very High Order Accurate TVD Schemes," ICASE Report No. 84–44, September 1984.

27. Simpson, L.B. and Whitfield, D.L., "Flux–Difference Split Algorithm for Unsteady Thin–Layer Navier–Stokes Solutions," AIAA Journal, Vol. 30, No. 4, April 1992, pp. 914–922.

28. Gatlin, B., Private Communication, Engineering Research Center for Computational Field Simulation, Mississippi State University, August 1993.

29. Taylor, L.K., Busby, J.A., Jiang, M.Y., Arabshahi, A., Sreenivas, K., and Whitfield, D.L., "Time Accurate Incompressible Navier–Stokes Simulation of the Flapping Foil Experiment," Sixth International Conference on Numerical Ship Hydrodynamics, Iowa City, Iowa, August 2–5, 1993.

30. Bernard, R.S. and Schneider, M.L., "Three–Dimensional Incompressible Flow Calculations with MacCormack's Method," Hydraulic Engineering, Saving a Threaten Resource – In Search of Solution, ASCE, 1992, pp. 219–224.

31. Smith, S.N., "Discrete Frequency Sound Generation in Axial Flow Turbomachines," ARC–R/M–3709, 1971.

32. Sreenivas, K. and Whitfield, D.L., "High Resolution Numerical Simulation of the Linearized Euler Equations in Conservation Law Form," AIAA Paper No. 93–2834, July 1993.