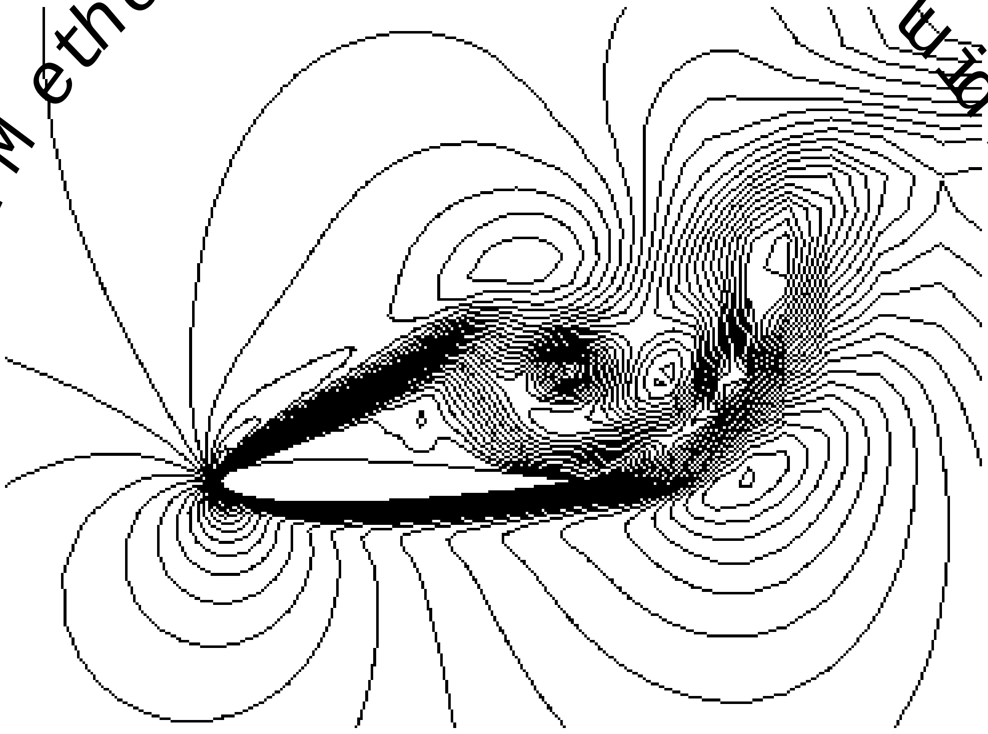
Notes on

Solution Methods In Computational Fluid Dynamics



by Thomas H. Pulliam

NASA Ames Research Center

Moffett Field, California

Based on notes from:
von K’arm’an Institute For Fluid Mechanics Lecture Serie
Numerical Techniques For Viscous Flow Computation In Turl
January 20-24, 1986, Rhode-St-Genese, Belgium.

Preliminary notes in revision, April 1994

Solution Methods In Computational Fluid Dynamics

Thomas H. Pulliam
Research Scientist CFD Branch
NASA Ames Research Center

Abstract

Implicit finite difference schemes for solving two dimensional and three dimensional Euler and Navier-Stokes equations will be addressed. The methods are demonstrated in fully vectorized codes for a CRAY type architecture. We shall concentrate on the Beam and Warming implicit approximate factorization algorithm in generalized coordinates. The methods are either time accurate or accelerated non-time accurate steady state schemes. Various acceleration and efficiency modifications such as matrix reduction, diagonalization and flux split schemes will be presented. Examples for 2-D inviscid and viscous calculations (e.g. airfoils with a deflected spoiler, circulation control airfoils and unsteady buffeting) and also 3-D viscous flow are included.

OUTLINE

Commentary, 1992

These notes were developed and put together in 1984-5 for a lecture series entitled "von Kármán Institute For Fluid Dynamics Lecture Series : Numerical Techniques for Viscous Flow Computation In Turbomachinery Bladings", von Kármán Institute, Rhode-St-Genese, Belgium , 1985. They may be a little dated today, but they still represent current algorithms and codes being used on a day to day basis for both research and development. I will update and correct some of the material, but I will not be attempting to completely modernize these lectures. Newer topics, e.g. TVD, ENO, are handled better in other forums and I will have to make my attempt at them some other time.

Finally, I would like to recognized the influence of one of my best friends and my mentor, Dr. Joseph L. Steger who passed away this year. I first met Joe in the early days of CFD at NASA Ames (circa 1974) when I started work on my thesis and Joe became my advisor. I consider myself Joe's first student and although Joe went on to teach at Stanford and U.C. Davis and produced many fine CFD researchers, I think my years with Joe will always be special since they were his and my first experiences as friends and teacher/student. Joseph Steger was a real pioneer for CFD, he did much of the ground breaking work in transonics and Euler/Navier-Stokes algorithms. I don't think he gets much credit for his transonic work, but if it wasn't for Joe many of the important advances made here at NASA Ames would never have happened. We always refer to the "Beam-Warming algorithm", but possibility it should be called the "Steger algorithm". Although, Beam and Warming can be credited with the initial development, Steger had much to do with the final developments and analysis. More importantly though, is the contribution Joe made in making the algorithm practical and popular. In 1976, Joe wrote what was then call AIR2D, based on the Beam-Warming algorithm and generalized coordinate transformations. That one effort has blossomed into the ARC2D and ARC3D codes and their subsequent impact on CFD today. Those kernels (gems of ideas) form the basis of most of the practical and useful codes in existence today. One only has to look anywhere in the literature to see that impact. Joe's other developments were equally as important to the whole picture. He developed elliptic grid generators (GRAPE) and hyperbolic grid generating algorithms. Joe can also be credited with producing the first practical Parabolized Navier-Stokes (PNS) codes and Incompressible Navier-Stokes (INS) based on psuedo-compressibility. His work on the Chimera approach for complicated geometry is currently the workhorse of computation here at NASA Ames and has been a significant part of the overall CFD effort. But besides all that, we really have suffered a great loss in losing Joe Steger, he always will be in my heart, his laugh and friendship are dear to me and I'm sure to all who knew him.

I. Introduction

Computational fluid dynamics is a growing technology. Even though there is still a substantial amount of theoretical development necessary before it becomes an consistent engineering tool, we can produce research codes which can be applied to relevant physical problems. At present full potential codes and panel methods have been the most widely used tools in the design cycle. Those methods are computational less expensive than the Euler and Navier - Stokes codes and in general more robust and accurate (mainly due to the ability to employ a large number of grid points or panels). Euler and Navier - Stokes codes require more storage and computational work per solution than the classical methods. Even with the present class of super computers we still have not reached a stage where the restrictions of computational speed and storage can be ignored. At this stage the Euler and Navier - Stokes codes available should be considered to be research codes. At first we strive to demonstrate the feasibility of the numerical technique used, then we should go on to establish the accuracy, efficiency and robustness of a developed code. These are the areas in code development and application which require careful consideration.

A wide variety of numerical techniques are in use today. Some have developed to a high enough level to be used in production codes (see Refs. [1-6]) while other techniques (for example TVD schemes, Ref. [7-9]) are just now entering the research code realm. In this presentation we shall concentrate on methods and techniques which have been applied to various computational fluid flow problems. These include, implicit finite differences, central space differencing, upwind differencing, approximate factorization, nonlinear dissipation models, characteristic boundary procedures, grid refinement - reclustering algorithms and various acceleration techniques for steady state and time accurate computations. Most of the applications are for external flows, but the methods have been and are easily extended to internal flows. A lot of the development will be in 2-D, with the extension to 3-D relatively straightforward.

A series of computer codes have been developed at NASA Ames Research Center based on the implicit approximate factorization algorithm of Beam and Warming [1] will be used for demonstration. A particular application in two dimensions was first presented by Steger [2] and for three dimensions by Pulliam and Steger [3]. Concurrent with this work has been the paralleled development and application of MacCormacks method [4]. I shall concentrate here on the theoretical development, application and assessment of the implicit algorithm which at this stage has produced two codes, ARC2D a two dimensional version and ARC3D the three dimensional code. The original development of these codes was more in the spirit of a demonstration effort, where we were more concerned with demonstrating the feasibility of the algorithm for general geometries and varying flow cases. A number of applications appeared over the years in the literature. More recently we have improved the accuracy, efficiency and robustness of the codes. I shall present below some details of the current versions of the implicit codes, ARC2D and ARC3D. Notable exceptions will be discussed. I shall not concentrate on the idiosyncrasies of the coding, I/O or other programming aspects except where they affect the algorithm application.

II. The Euler and Navier - Stokes Equations

The starting point is the strong conservation law form of the two-dimensional Navier-Stokes equations in Cartesian coordinates. The strong conservation law form is chosen because we wish to accurately capture shocks. The equations in nondimensional form are

$$\partial_t Q + \partial_x E + \partial_y F = Re^{-1}\left(\partial_x E_v + \partial_y F_v\right) \tag{2.1}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(e+p) \end{bmatrix}, \quad F = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(e+p) \end{bmatrix},$$
$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ f_4 \end{bmatrix}, \quad F_v = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ g_4 \end{bmatrix} \tag{2.2a}$$

with

$$\begin{aligned} \tau_{xx} &= \mu(4u_x - 2v_y)/3 \\ \tau_{xy} &= \mu(u_y + v_x) \\ \tau_{yy} &= \mu(-2u_x + 4v_y)/3 \\ f_4 &= u\tau_{xx} + v\tau_{xy} + \mu Pr^{-1}(\gamma-1)^{-1}\partial_x a^2 \\ g_4 &= u\tau_{xy} + v\tau_{yy} + \mu Pr^{-1}(\gamma-1)^{-1}\partial_y a^2 \end{aligned} \tag{2.2b}$$

Pressure is related to the conservative flow variables, Q, by the equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{2.3}$$

where γ is the ratio of specific heats, generally taken as 1.4. The speed of sound is a which for ideal fluids, $a^2 = \gamma p/\rho$. The dynamic viscosity is μ and is typically made up of a constant plus a computed turbulent eddy viscosity. The Reynolds number is Re and Prandtl number Pr.

The choice of nondimensional parameters is arbitrary. Here we have chosen to scale the variables ρ (density), u v (the Cartesian velocities), and e (the total energy) as

$$\widetilde{\rho} = \frac{\rho}{\rho_\infty}, \quad \widetilde{u} = \frac{u}{a_\infty}, \quad \widetilde{v} = \frac{v}{a_\infty}, \quad \widetilde{e} = \frac{e}{\rho_\infty a_\infty^2} \tag{2.4a}$$

where ∞ refers to free stream quantities. Assuming a reference length, l (usually taken as some characteristic physical length such as chord of an airfoil), time t scales as $\widetilde{t} = ta_\infty/l$. The viscous coefficients scale as

$$\widetilde{\mu} = \frac{\mu}{\mu_\infty}, \quad Re = \frac{\rho_\infty l a_\infty}{\mu_\infty} \tag{2.4b}$$

Note that Re uses a_∞ and therefore Re based on u_∞ (the usual case for experimentally given Reynolds number) must be scaled by $M_\infty = u_\infty/a_\infty$. For the remainder of this development the $\sim$ will be dropped for simplicity.

The Euler equations are recovered from Eqs. (2.1) and (2.2) by dropping the viscous terms, i.e. setting the right hand side of Eq. (2.1) equal to zero.

III. Generalized Curvilinear Coordinate Transformations

The Navier-Stokes equations can be transformed from Cartesian coordinates to general curvilinear coordinates where

$$\begin{aligned} \tau &= t \\ \xi &= \xi(x, y, t) \\ \eta &= \eta(x, y, t) \end{aligned} \tag{3.1}$$

The coordinate transformation introduced here follows the development of Viviand [10] and Vinokur [11]. Curvilinear coordinates are a representation of n-space in which arbitrary vectors are represented by two sets of basis vectors (not necessarily orthogonal). An arbitrary vector $\mathbf{V}$ (here demonstrated in 2 dimensions) is defined as

$$\mathbf{V} = v^1\overline{e}_1 + v^2\overline{e}_2$$

with $\overline{e}_i$ covariant basis vectors and v^i the contravariant components of $\mathbf{V}$. Since we don't require the basis vectors to be orthogonal, another set of component extracting basis vectors are required, the contravariant basis vectors $\overline{e}^i$. The basis vectors satisfy the relationship $\overline{e}^i \cdot \overline{e}_j = \delta_{i,j}$. So that, contravariant component $v^i = \overline{e}^i \cdot \mathbf{V}$. Another representation of $\mathbf{V}$ is in terms of covariant basis vectors $\overline{e}^i$ with

$$\mathbf{V} = v_1\overline{e}^1 + v_2\overline{e}^2$$

and v_i the covariant components of $\mathbf{V}$, i.e., $v_i = \overline{e}_i \cdot \mathbf{V}$. An excellent reference for curvilinear transforms is Korn and Korn [aa]. This representation and sets of basis vectors form the fundamental framework for the curvilinear transformations which will be applied to the Euler and Navier-Stokes equations.

The transformations are chosen so that the grid spacing in the curvilinear space is uniform and of unit length, see Fig. 1. This produces a computational space ξ and η which is a rectangular domain and which has a regular uniform mesh so that standard unweighted differencing schemes can be used in the numerical formulation. The original Cartesian space will be referred to as the physical domain. Typically there will be a one to one correspondence between a physical point in space and a computational point, except for regions where there are singularities or cuts due to the topology. In those cases it may be necessary to map one physical point to many computational points (this usually occurs at computational boundaries). With this construction we can produce one computational code for a wide variety of physical geometries and grid systems.

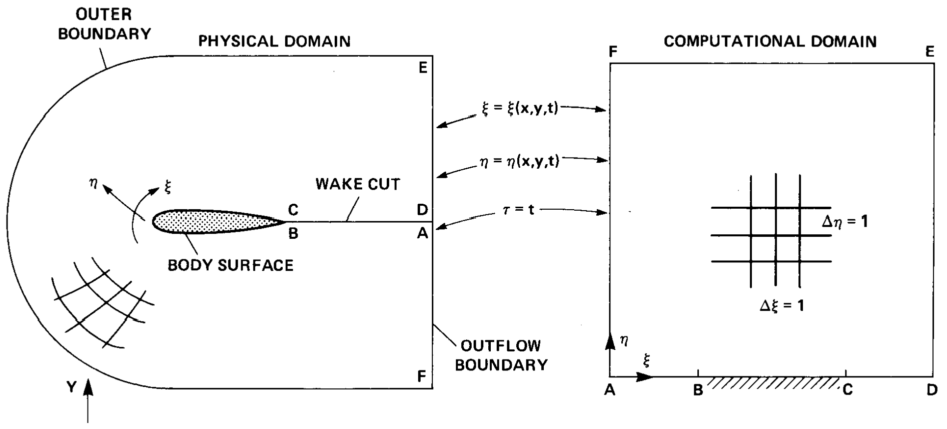


FIGURE 1. Generalized Curvilinear Coordinate Transformations.

Chain rule expansions are used to represent the Cartesian derivatives ∂_x and ∂_y of Eq. (2.1) in terms of the curvilinear derivatives where in matrix form

$$\begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} 1 & \xi_t & \eta_t \\ 0 & \xi_x & \eta_x \\ 0 & \xi_y & \eta_y \end{bmatrix} \begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} \tag{3.2}$$

Applying Eq. (3.2) to the Navier-Stokes equations, Eq. (2.1), we have

$$\begin{aligned} &\partial_\tau Q + \xi_t \partial_\xi Q + \eta_t \partial_\eta Q \\ &\qquad + \xi_x \partial_\xi E + \eta_x \partial_\eta E + \xi_y \partial_\xi F + \eta_y \partial_\eta F = \\ &Re^{-1} \left(\xi_x \partial_\xi E_v + \eta_x \partial_\eta E_v + \xi_y \partial_\xi F_v + \eta_y \partial_\eta F_v \right) \end{aligned} \tag{3.3}$$

3.1 Metric Relations

In most cases the transformation from physical space to computational space is not known analytically, rather it is generated numerically. That is, we usually are provided with just the x, y coordinates of grid points and we numerically generate the metrics $(\xi_t, \xi_x, \xi_y, \eta_t, \eta_x, \eta_y)$ using finite differences.

Reversing the role of the independent variables in the chain rule formulas, Eq. (3.2), we have,

$$\partial_\tau = \partial_t + x_\tau \partial_x + y_\tau \partial_y, \quad \partial_\xi = x_\xi \partial_x + y_\xi \partial_y, \quad \partial_\eta = x_\eta \partial_x + y_\eta \partial_y \tag{3.4}$$

which can be written in matrix form

$$\begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} = \begin{bmatrix} 1 & x_\tau & y_\tau \\ 0 & x_\xi & y_\xi \\ 0 & x_\eta & y_\eta \end{bmatrix} \begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} \tag{3.5}$$

Solving Eq. (3.5) for the curvilinear derivatives in terms of the Cartesian derivatives yields

$$\begin{bmatrix} \partial_t \\ \partial_x \\ \partial_y \end{bmatrix} = J \begin{bmatrix} (x_\xi y_\eta - y_\xi x_\eta) & (-x_\tau y_\eta + y_\tau x_\eta) & (x_\tau y_\xi - y_\tau x_\xi) \\ 0 & y_\eta & -y_\xi \\ 0 & -x_\eta & x_\xi \end{bmatrix} \begin{bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\eta \end{bmatrix} \tag{3.6}$$

where $J^{-1} = (x_\xi y_\eta - x_\eta y_\xi)$. Evaluating Eq. (3.6) for the metric terms by comparing to the matrix of Eq. (3.2) we find that

$$\begin{aligned} \xi_x &= J y_\eta, \quad \xi_y = -J x_\eta, \quad \xi_t = -x_\tau \xi_x - y_\tau \xi_y \\ \eta_x &= -J y_\xi, \quad \eta_y = J x_\xi, \quad \eta_t = -x_\tau \eta_x - y_\tau \eta_y \end{aligned} \tag{3.7}$$

where J is defined to be the metric Jacobian.

3.2 Invariants of the Transformation

At this point we notice that Eqs. (3.3) are in a weak conservation law form. That is, even though none of the flow variables (or more appropriately functions of the flow variables) occur as coefficients in the differential equations, the metrics do. There is some argument in the literature, see for instance Hindman [12], which advocates the use of the so called "chain rule form" since it should still have good shock capturing properties and in some ways it is a simpler form. Here, though, we shall restrict ourselves to the strong conservation law form which will be derived below.

To produce the strong conservation law form we first multiply Eqs.(3.3) by J^{-1} and use the chain rule on all terms such as

$$J^{-1}\xi_x\partial_\xi E = \partial_\xi\left(\frac{\xi_x}{J}E\right) - E\partial_\xi\left(\frac{\xi_x}{J}\right) \tag{3.8}$$

For simplicity, we examine only the inviscid terms, the derivation for the viscous terms is similar. Collecting all the terms into two groups,

$$Term_1 + Term_2 = 0$$

where

$$\begin{aligned}
Term_1 &= \partial_\tau(Q/J) + \partial_\xi[(\xi_t Q + \xi_x E + \xi_y F)/J] \\
&\qquad\qquad + \partial_\eta[(\eta_t Q + \eta_x E + \eta_y F)/J] \\
Term_2 &= -Q[\partial_\tau(J^{-1}) + \partial_\xi(\xi_t/J) + \partial_\eta(\eta_t/J)] \\
-E[\partial_\xi&(\xi_x/J) + \partial_\eta(\eta_x/J)] \ - F[\partial_\xi(\xi_y/J) + \partial_\eta(\eta_y/J)]
\end{aligned} \tag{3.9}$$

If $Term_2$ is eliminated then the strong conservation law form of the equations results, $Term_1 = 0$. Assuming solutions such that $Q \neq 0, E \neq 0$, and $F \neq 0$, the expressions

$$\begin{aligned}
\partial_\tau(J^{-1}) + \partial_\xi(\xi_t/J) + \partial_\eta(\eta_t/J) \\
\partial_\xi(\xi_x/J) + \partial_\eta(\eta_x/J) \\
\partial_\xi(\xi_y/J) + \partial_\eta(\eta_y/J)
\end{aligned} \tag{3.10}$$

are defined as invariants of the transformation and will be shown to be analytically zero. Substituting the metric definitions, Eq. (3.7), into the invariants, Eq. (3.10) we have

$$\begin{aligned}
\partial_\tau(x_\xi y_\eta - y_\xi x_\eta) + \partial_\xi(-x_\tau y_\eta + y_\tau x_\eta) + \partial_\eta(x_\tau y_\xi - y_\tau x_\xi) \\
\partial_\xi(y_\eta) + \partial_\eta(-y_\xi) = y_{\eta\xi} - y_{\xi\eta} \\
\partial_\xi(-x_\eta) + \partial_\eta(x_\xi) = -x_{\eta\xi} + x_{\xi\eta}
\end{aligned} \tag{3.11}$$

Now analytically differentiation is commutative and each of the above terms then sums to zero. This eliminates $Term_2$ of Eq. (3.9) and the resulting equations are in strong conservation law form.

There is an important problem associated with these invariants which can be discussed now. If $Term_1$ is evaluated for uniform flow,

$$\rho = 1, \quad u = M_\infty, \quad v = 0, \quad \text{and} \quad e = \frac{1}{\gamma(\gamma-1)} + \frac{1}{2}M_\infty^2$$

then the resulting equations which must sum to zero (if we require that our equations satisfy free stream) are exactly composed of the invariants, Eq. (3.10). Now when numerical differencing is applied to these equations (as developed in the Section V) then the numerical difference formulas used to evaluate the spatial differences of the fluxes and the finite difference forms used to calculate the metrics must satisfy the commutative law. It is not true in general that finite difference derivatives are commutative, (second order central differences are, but mixing second order and fourth order formulas is not). As we shall see, the central difference formulas used in two-dimensions can produce consistent invariants, but in three-dimensions it is not a straightforward procedure.

It should be at least a minimum requirement of any finite difference formulation that the finite difference equations satisfy free stream flow. Care must be taken to insure that the finite difference formulation is consistent in this area or at least we should recognize and correct as much as possible any errors of this type. Hindman [12], Pulliam and Steger [3] and Flores et. al. [13] have investigated this area for a variety of finite difference formulations.

The Navier-Stokes equations written in generalized curvilinear coordinates are

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} = Re^{-1}[\partial_\xi \widehat{E}_v + \partial_\eta \widehat{F}_v] \tag{3.12}$$

where

$$\widehat{Q} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \widehat{E} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) - \xi_t p \end{bmatrix}, \quad \widehat{F} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ V(e+p) - \eta_t p \end{bmatrix} \tag{3.13a}$$

with

$$U = \xi_t + \xi_x u + \xi_y v, \quad V = \eta_t + \eta_x u + \eta_y v \tag{3.13b}$$

the Contravariant velocities. The viscous flux terms are $\widehat{E}_v = J^{-1}(\xi_x E_v + \xi_y F_v)$ and $\widehat{F}_v = J^{-1}(\eta_x E_v + \eta_y F_v)$.

The stress terms, such as τ_{xx} are also transformed in terms of the ξ and η

derivatives where

$$\begin{aligned}
\tau_{xx} &= \mu(4(\xi_x u_\xi + \eta_x u_\eta) - 2(\xi_y v_\xi + \eta_y v_\eta))/3 \\
\tau_{xy} &= \mu(\xi_y u_\xi + \eta_y u_\eta + \xi_x v_\xi + \eta_x v_\eta) \\
\tau_{yy} &= \mu(-2(\xi_x u_\xi + \eta_x u_\eta) + 4(\xi_y v_\xi + \eta_y v_\eta))/3 \qquad (3.14) \\
f_4 &= u\tau_{xx} + v\tau_{xy} + \mu Pr^{-1}(\gamma-1)^{-1}(\xi_x \partial_\xi a^2 + \eta_x \partial_\eta a^2) \\
g_4 &= u\tau_{xy} + v\tau_{yy} + \mu Pr^{-1}(\gamma-1)^{-1}(\xi_y \partial_\xi a^2 + \eta_y \partial_\eta a^2)
\end{aligned}$$

with terms such as u_x expanded by chain rule.

IV Thin - Layer Approximation

In high Reynolds number viscous flows the effects of viscosity are concentrated near rigid boundaries and in wake regions. Typically in computations we only have enough grid points available to us (due to computer storage limits) to concentrate grid lines near the rigid surfaces. The resulting grid systems usually have fine grid spacing in directions nearly normal to the surfaces and coarse grid spacing along the surface, see Fig. 2.

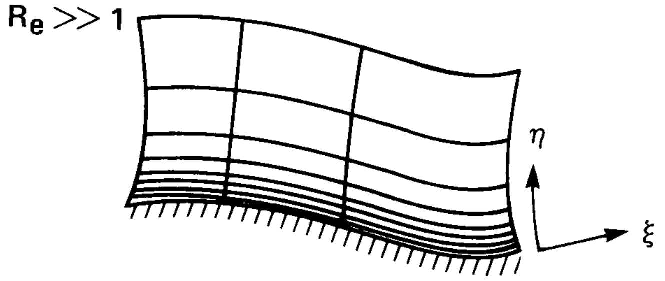


FIGURE 2. Thin Viscous Layer Near Body Surface.

Even though we may program the full Navier-Stokes equations, the viscous terms associated with derivatives along the body will not be resolved and in most cases for attached and mildly separated flows these terms are negligible. The terms in the near normal will be resolved for sufficiently fine grid spacing and these are substantial terms.

In boundary layer theory, appropriate scaling arguments show that streamwise components of the viscous terms can be neglected relative to the normal terms. We rely upon similar arguments as a justification for the thin layer approximation.

The thin layer approximation requires that

1. All body surfaces be mapped onto coordinate surfaces. Specifically, η = constant coordinate surfaces, see Fig. 1.
2. Grid spacing is clustered to the body surfaces such that sufficient resolution for a particular Reynolds number is obtained. (At least one or two grid points in the sublayer).
3. All the viscous derivatives in the ξ direction are neglected, while the terms in the η direction are retained. All of the inviscid terms are used.

The thin layer approximation is similar in philosophy but not the same as the boundary layer theory. The normal momentum equation is solved and pressure can vary through the boundary layer.

The thin layer approximation can break down for low Reynolds numbers and in regions of massive flow separation. It is not a necessary step in the development of the equations and numerical algorithm. The full Navier-Stokes equations are incorporated in cases where sufficient resolution was provided and the physical situation warranted it. The thin layer Navier-Stokes equations have been widely used for a variety of applications.

4.1 Thin - Layer Equations

Applying the thin layer approximation to Eqs.(3.12), (3.13) and Eq.(3.14), where all the viscous terms associated with ξ derivatives are neglected we obtain

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} = Re^{-1} \partial_\eta \widehat{S} \tag{4.1}$$

where

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x(u m_1 + v m_2 + m_4) + \eta_y(u m_2 + v m_3 + m_5) \end{bmatrix} \tag{4.2a}$$

with

$$\begin{aligned} m_1 &= \mu(4\eta_x u_\eta - 2\eta_y v_\eta)/3 \\ m_2 &= \mu(\eta_y u_\eta + \eta_x v_\eta) \\ m_3 &= \mu(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \\ m_4 &= \mu Pr^{-1}(\gamma - 1)^{-1} \eta_x \partial_\eta (a^2) \\ m_5 &= \mu Pr^{-1}(\gamma - 1)^{-1} \eta_y \partial_\eta (a^2) \end{aligned} \tag{4.2b}$$

4.2 Turbulence Model

The very polular and widely used Baldwin and Lomax [14] turbulence model has been the main workhorse for most computational efforts, at least until recently circa 1990. It is an algebraic mixing length two-layer model included to approximate the effect of turbulence. The inner layer is governed by the Prandtl mixing length with Van Driest damping, and the outer layer follows the Clauser approximation. Computed vorticity is used in defining the reference mixing length required for the outer layer. The turbulence model is detailed by Baldwin and Lomax [14] and was designed specifically for use with the thin layer approximation. The model is most appropriate to attached and mildly separated boundary layers. No attempt is made to model wake regions and massively separated flows. The model is used in both two and three dimensions with very little modification. It has been very successful for computing boundary layer growth, shock-boundary layer interaction, separation points or lines and other boundary layer properties.

More modern turbulence models include the Johnson-King model [xx], the one equations models of Baldwin and Barth [xx] and Spalart and Almaras [xx]. These models are more complicated than Baldwin and Lomax, but have been shown to be more accuate and applicable to separated and wakes flows. A wide variety of two equation turbulence models are available, (e.g. [xxxxxxxx]) and as one quickly finds out in this area, no single model seems universal or completely adequate. One aspect of using turbulence models which is often overlooked is that adequate resolution is always required to get reasonable results reguardless of the turbulence model employed. Typically, the inaccuracy or inadequacy of a solution is not the fault of the turbulence model, but rather a lack of proper resolution in the viscous and even inviscid regions of the flowfield.

V. Numerical Algorithm

There are a number of considerations to weigh when choosing a numerical algorithm to apply to a set of partial differential equations. If we restrict ourselves to finite difference schemes then the possibilities are narrowed somewhat to the two classical approaches for time integration, explicit and implicit techniques. The merits of either of these two have been extensively discussed in the literature. Explicit methods typically require less computational work and are simpler both in derivation and application. Implicit methods, while computationally expensive, have less stringent stability bounds (classical stability analysis shows unconditional stability but in practice on nonlinear problems bounds are encountered).

Implicit numerical schemes are usually chosen because we wish to obtain solutions which require fine grid spacing for numerical resolution, and we do not want to limit the time steps by employing a conditionally stable explicit scheme. Explicit schemes are very useful and schemes such as MacCormack's explicit algorithm [4] have senn a lot of use and are even popular in wide use today. The extra work required for an implicit scheme is usually offset by the advantages obtained by the increased stability limits, and in general implicit schemes have been very useful and successful for a variety of inviscid and viscous flowfield calculations.

With the advent of high speed vector and parallel processors one must also consider the degree to which a certain algorithm can be vectorized/parallelized when choosing a scheme. As a rule explicit schemes are more easily vectorized/parallelized than implicit schemes. But implicit schemes can be fully vectorized and have been sucessfully employed on parallel machines. This requires though a substantial amount of temporary storage and a commitment to the details of data management, see for instance, Lomax and Pulliam [15].

Another consideration is the question of time accuracy verses non-time-accurate steady state iteration. For unsteady, transient problems we wish to employ time accurate methods, initialize the flow with some realizable state and integrate forward in time with time steps commensurate with the unsteady phenomena which is being calculated. Both implicit and explicit methods are capable of computing time accurately. In steady state calculation we wish to integrate from some arbitrary state to the asymptotic solution in any manner which will get us there in the least amount of computational work. Non-time-accurate techniques (for instance relaxation methods, variable time steps, matrix preconditioning, large time steps) can be employed as long as they are convergent and do not distort the steady state equations so as to produce inaccurate results. The methods presented below can be employed either for time accurate calculations or for steady state rapidly convergent solutions.

The algorithm to be presented is an implicit approximate factorization finite difference scheme which can be either first or second order accurate in time. Local time linearizations are applied to the nonlinear terms and an approximate factorization of the two-dimensional implicit operator is used to produce locally one-dimensional operators. This results in block tridiagonal matrices, which are easy to solve. The spatial derivative terms are approximated with second order central differences. Explicit and implicit artificial dissipation terms are added to achieve nonlinear stability. A spatially variable time step is used to accelerate convergence for steady-state calculations. A diagonal form of the algorithm is also discussed, which produces a computationally efficient modification of the standard algorithm where the diagonalization results in scalar tridiagonal or pentadiagonal operators in place of the block operators. This diagonal form of the algorithm produces a robust, rapid and versatile scheme for steady state calculations. We also discuss the details of a matrix reduction scheme, due to Barth and Steger [16] where the block matrices of the standard implicit scheme are reduced to sets of lower rank matrices (e.g. two scalars and a 2×2 in 2-D).

5.1 Implicit Time Differencing

Consider Eq. (4.1) (the derivation will be done for the thin layer equations but is easily extended to the full Navier-Stokes) and apply an implicit three point time differencing scheme of the form, Warming and Beam [17]

$$\begin{aligned}\Delta\widehat{Q}^n = \frac{\vartheta\Delta t}{1+\varphi}\frac{\partial}{\partial t}\left(\Delta\widehat{Q}^n\right) + \frac{\Delta t}{1+\varphi}\frac{\partial}{\partial t}\widehat{Q}^n + \frac{\varphi}{1+\varphi}\Delta\widehat{Q}^{n-1} \\ +O\left[(\vartheta - \frac{1}{2} - \varphi)\Delta t^2 + \Delta t^3\right]\end{aligned} \tag{5.1}$$

where $\Delta\widehat{Q}^n = \widehat{Q}^{n+1} - \widehat{Q}^n$ and $\widehat{Q}^n = \widehat{Q}(n\Delta t)$. The parameters ϑ and φ can be chosen to produce different schemes of either first or second order accuracy in time.

For $\vartheta = 1$ and $\varphi = 0$, we have the first order Euler implicit scheme, and for $\vartheta = 1$ and $\varphi = 1/2$, the three point implicit scheme.

Let us restrict ourselves to the first order in time scheme (although all of the subsequent development can easily be extended to any second order scheme formed from Eq. (5.1)). Applying Eq. (5.1) to Eq. (4.1), results in

$$\widehat{Q}^{n+1} - \widehat{Q}^n + h\left(\widehat{E}_\xi^{n+1} + \widehat{F}_\eta^{n+1} - Re^{-1}\widehat{S}_\eta^{n+1}\right) = 0 \tag{5.2}$$

with $h = \Delta t$.

5.2 Local Time Linearizations

We wish to solve Eq. (5.2) for $\widehat{Q}^{n+1}$ given $\widehat{Q}^n$. The flux vectors $\widehat{E}$,$\widehat{F}$ and $\widehat{S}$ are nonlinear functions of $\widehat{Q}$ and therefore Eq. (5.2) is nonlinear in $\widehat{Q}^{n+1}$. The nonlinear terms are linearized in time about $\widehat{Q}^n$ by a Taylor series such that

$$\begin{aligned} \widehat{E}^{n+1} &= \widehat{E}^n + \widehat{A}^n \Delta \widehat{Q}^n + O(h^2) \\ \widehat{F}^{n+1} &= \widehat{F}^n + \widehat{B}^n \Delta \widehat{Q}^n + O(h^2) \\ Re^{-1}\widehat{S}^{n+1} &= Re^{-1}\left[\widehat{S}^n + \widehat{M}^n \Delta \widehat{Q}^n)\right] + O(h^2) \end{aligned} \tag{5.3}$$

where $\widehat{A} = \partial\widehat{E}/\partial\widehat{Q}$, $\widehat{B} = \partial\widehat{F}/\partial\widehat{Q}$ and $\widehat{M} = \partial\widehat{S}/\partial\widehat{Q}$ are the flux Jacobians and $\Delta\widehat{Q}^n$ is $O(h)$.

Note that the linearizations are second order accurate and so if a second order time scheme had been chosen the linearizations would not degrade the time accuracy.

The Jacobian matrices are $\widehat{A}$ or $\widehat{B}$ =

$$\begin{bmatrix} \kappa_t & \kappa_x & \kappa_y & 0 \\ -u\theta + \kappa_x\phi^2 & \kappa_t + \theta - (\gamma-2)\kappa_x u & \kappa_y u - (\gamma-1)\kappa_x v & (\gamma-1)\kappa_x \\ -v\theta + \kappa_y\phi^2 & \kappa_x v - (\gamma-1)\kappa_y u & \kappa_t + \theta - (\gamma-2)\kappa_y v & (\gamma-1)\kappa_y \\ \theta[\phi^2 - a_1] & \kappa_x a_1 - (\gamma-1)u\theta & \kappa_y a_1 - (\gamma-1)v\theta & \gamma\theta + \kappa_t \end{bmatrix} \tag{5.4}$$

with $a_1 = \gamma(e/\rho) - \phi^2$, $\theta = \kappa_x u + \kappa_y v$, $\phi^2 = \frac{1}{2}(\gamma-1)(u^2+v^2)$, and $\kappa = \xi$ or η for $\widehat{A}$ or $\widehat{B}$, respectively.

The viscous flux Jacobian is

$$\widehat{M} = J^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1\partial_\eta(\rho^{-1}) & \alpha_2\partial_\eta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2\partial_\eta(\rho^{-1}) & \alpha_3\partial_\eta(\rho^{-1}) & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} J \tag{5.5a}$$

where

$$\begin{aligned} m_{21} =& -\alpha_1\partial_\eta(u/\rho) - \alpha_2\partial_\eta(v/\rho) \\ m_{31} =& -\alpha_2\partial_\eta(u/\rho) - \alpha_3\partial_\eta(v/\rho) \\ m_{41} =& \alpha_4\partial_\eta\left[-(e/\rho^2) + (u^2+v^2)/\rho\right] \\ & - \alpha_1\partial_\eta(u^2/\rho) - 2\alpha_2\partial_\eta(uv/\rho) \\ & - \alpha_3\partial_\eta(v^2/\rho) \\ m_{42} =& -\alpha_4\partial_\eta(u/\rho) - m_{21} \\ m_{43} =& -\alpha_4\partial_\eta(v/\rho) - m_{31} \\ m_{44} =& \alpha_4\partial_\eta(\rho^{-1}) \\ \alpha_1 &= \mu[(4/3){\eta_x}^2 + {\eta_y}^2], \quad \alpha_2 = (\mu/3)\eta_x\eta_y \\ \alpha_3 &= \mu[{\eta_x}^2 + (4/3){\eta_y}^2], \quad \alpha_4 = \gamma\mu Pr^{-1}({\eta_x}^2 + {\eta_y}^2) \end{aligned} \tag{5.5b}$$

Applying Eqs. (5.3) to Eq. (5.2) and combining the $\Delta\widehat{Q}^n$ terms produces the "delta form" of the algorithm

$$\begin{aligned}\left[I + h\partial_\xi \widehat{A}^n + h\partial_\eta \widehat{B}^n - Re^{-1} h\partial_\eta \widehat{M}\right]\Delta\widehat{Q}^n = \\ -h\left(\partial_\xi \widehat{E}^n + \partial_\eta \widehat{F}^n - Re^{-1}\partial_\eta \widehat{S}^n\right)\end{aligned} \tag{5.6}$$

This is the unfactored form of the block algorithm. We shall call the right hand side of Eq. (5.6) the "explicit" part and the left hand side the "implicit" part of the algorithm.

5.3 Space Differencing

The next step is to take the continuous differential operators ∂_ξ and ∂_η and approximate them with finite difference operators on a discrete mesh.

Introducing a grid of mesh points (j, k), variables are defined at mesh points as

$$u_{j,k} := u(j\Delta\xi, k\Delta\eta) \tag{5.7}$$

The grid spacing in the computational domain is chosen to be unity so that

$$\Delta\xi = 1 \quad \text{and} \quad \Delta\eta = 1$$

Second order central difference operators can be used where for example,

$$\delta_\xi u_{j,k} = \left(u_{j+1,k} - u_{j-1,k}\right)/2 \quad \text{and} \quad \delta_\eta u_{j,k} = \left(u_{j,k+1} - u_{j,k-1}\right)/2 \tag{5.8a}$$

For the viscous derivatives the terms take the form

$$\partial_\eta\left(\alpha_{j,k}\partial_\eta \beta_{j,k}\right) \tag{5.8b}$$

which is differenced in the compact three point form as

$$\left[\left(\alpha_{j,k+1} + \alpha_{j,k}\right)\left(\beta_{j,k+1} - \beta_{j,k}\right) - \left(\alpha_{j,k} + \alpha_{j,k-1}\right)\left(\beta_{j,k} - \beta_{j,k-1}\right)\right]/2 \tag{5.8c}$$

The choice of the type and order of the spatial differencing is important both in terms of accuracy and stability. In most applications second order accuracy has proven to be sufficient provided the grid resolution is reasonable. The choices for differencing type include central and upwind operators. These choices are dictated by stability, and in the next section we discuss what motivates certain choices.

5.4 Stability Analysis of Difference Forms

The choice of the type of difference forms to use for the Euler equations can be justified by a linear stability analysis. For simplicity, let us examine a one dimensional coupled system of linear equations of the form

$$Q_t + AQ_x = 0 \tag{5.9}$$

where A is analogous to the flux Jacobian matrix. Assume that A has a complete set of real eigenvalues and eigenvectors (a property that the Euler flux Jacobians have) then

$$\Lambda = X^{-1}AX \tag{5.10}$$

Multiplying Eq. (5.9) by X^{-1} and combining terms using Eq. (5.10) we have

$$X^{-1}Q_t + X^{-1}AXX^{-1}Q_x = W_t + \Lambda W_x = 0 \tag{5.11}$$

with $W = X^{-1}Q$. Since A is linear and constant the eigenvector matrix X^{-1} can be brought through the derivatives.

The resulting system is now uncoupled and we can examine the representative model equation

$$w_t + \lambda w_x = 0 \tag{5.12}$$

where λ represents an eigenvalue of A.

We shall apply different finite difference approximations for the spatial derivative and use Fourier analysis to determine conditions on λ for stability.

If the second order central difference operator is applied to the model equation one gets

$$(w_j)_t + \lambda\left(w_{j+1} - w_{j-1}\right)/(2\Delta x) = 0 \tag{5.13}$$

where j is the spatial index. This is the ODE (ordinary differential equation) approach to the analysis, since now we are dealing with a system of ODE's.

Classical Fourier analysis can be performed by assuming periodic boundary conditions and a solution of the form

$$w(x_j, t) = e^{\alpha t}e^{i\beta j\Delta x} \tag{5.14}$$

with $i = \sqrt{-1}$ and $x = j\Delta x$.

Substituting this into Eq. (5.12) yields

$$\alpha e^{\alpha t}e^{i\beta j\Delta x} + \lambda\left(e^{\alpha t}e^{i\beta(j+1)\Delta x} - e^{\alpha t}e^{i\beta(j-1)\Delta x}\right)/(2\Delta x) = 0 \tag{5.15}$$

The stability of the ODE is dependent on the sign of $\Re(\alpha)$ (the real part). Obviously, if $\Re(\alpha) > 0$ then $w(x,t)$ will grow unboundedly with time.

For Eq. (5.15)

$$\alpha = -\lambda \left(e^{i\beta\Delta x} - e^{-i\beta\Delta x}\right)/(2\Delta x) = -\lambda i \ \sin(\beta\Delta x)/\Delta x \tag{5.16}$$

Since α is pure imaginary ($\Re(\alpha) = 0$) the scheme is stable in the ODE sense independent of the sign of λ.

If one-sided difference formulas are employed, conditions on λ arise. For simplicity, let us consider first order one-sided differences.

Applying forward differencing to the model Eq. (5.12) gives

$$(w_j)_t + \lambda\left(w_{j+1} - w_j\right)/\Delta x = 0 \tag{5.17}$$

Fourier analysis produces

$$\alpha + \lambda\left(e^{i\beta\Delta x} - 1\right)/\Delta x = 0 \tag{5.18}$$

so that,

$$\alpha = \lambda\left(1 - e^{i\beta\Delta x}\right)/\Delta x = \lambda\left[1 - \cos\left(\beta\Delta x\right) + i\sin\left(\beta\Delta x\right)\right]/\Delta x \tag{5.19}$$

Since $\cos(\beta\Delta x)$ is bounded by 1, $\Re(\alpha)$ will be less than zero if $\lambda < 0$. So for forward spatial differencing λ must be less than zero for stability. A similar argument for first order backward differencing shows that $\lambda > 0$ for stability. It can be shown that for higher order central and one sided differences the stability requirements on λ remain the same.

These results have a direct application to the choice of differencing for the Euler equations. As we shall see below the inviscid flux Jacobians have eigenvalues (equivalent to λ) with both positive and negative sign. In their basic form the only stable spatial differencing is central differencing, but as we shall see when flux splitting is used or when the eigenvalues can be restricted to one sign then upwind differencing can be employed. A class of upwind schemes shall be discussed in Section 6.2.

5.5 Matrix Form of Unfactored Algorithm

We now turn to examining the matrices we get when difference formulas are applied to the implicit algorithm. It is always instructive to examine the matrix structure of any finite difference equation. With the application of central differences to Eqs.(5.6) it is easy to show that the implicit algorithm produces a large banded system of algebraic equations. Let the mesh size in ξ be $Jmax$ and in η $Kmax$.

Then the banded matrix is a $(Jmax \cdot Kmax \cdot 4) \times (Jmax \cdot Kmax \cdot 4)$ square matrix of the form

$$\left[\begin{array}{cccccccccccc}
[\ ] & [\ ] & & & [\ ] & & & & & & & \\
[\ ] & [\ ] & [\ ] & & & [\ ] & & & & & & \\
 & [\ ] & [\ ] & [\ ] & & & [\ ] & & & & & \\
 & & [\ ] & [\ ] & & & & [\ ] & & & & \\
[\ ] & & & & [\ ] & [\ ] & & & [\ ] & & & \\
 & \ddots & & & \ddots & \ddots & \ddots & & & \ddots & & \\
 & & -B & & & -A & I & A & & & B & \\
 & & & [\ ] & & & [\ ] & [\ ] & & & & [\ ] \\
 & & & & [\ ] & & & & [\ ] & [\ ] & & \\
 & & & & & [\ ] & & & [\ ] & [\ ] & [\ ] & \\
 & & & & & & [\ ] & & & [\ ] & [\ ] & [\ ] \\
 & & & & & & & [\ ] & & & [\ ] & [\ ]
\end{array}\right] \tag{5.20}$$

where the variables have been ordered with j running first and then k.

The matrix is sparse but it would be very expensive (computationally) to solve the algebraic system. For instance, for a reasonable two-dimensional calculation of transonic flow past an airfoil we could use approximately 80 points in the ξ direction and 40 points in the η direction. The resulting algebraic system is a 12,800 $\times$ 12,800 matrix problem to be solved and although we could take advantage of its banded sparse structure it would still be very costly in terms of both CPU time and storage.

5.6 Approximate Factorization

As we have seen, the integration of the full two-dimensional operator can be very expensive. One way to simplify the solution process is to introduce an approximate factorization of the two-dimensional operator into two one-dimensional operators. The implicit side of Eq. (5.6) can be written as

$$\begin{gathered}
\left[I + h\delta_\xi \widehat{A}^n + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n = \\
\left[I + h\delta_\xi \widehat{A}^n\right] \left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right] \Delta\widehat{Q}^n \\
-h^2\delta_\xi \widehat{A}^n \delta_\eta \widehat{B}^n \, \Delta\widehat{Q}^n + h^2 Re^{-1}\delta_\xi \widehat{A}^n \delta_\eta \widehat{M}^n \, \Delta\widehat{Q}^n
\end{gathered} \tag{5.21}$$

Noting that $\Delta\widehat{Q}^n$ is $O(h)$, one sees that the cross terms (h^2 terms) are second order in time and can be neglected without lowereing the time accuracy below second order.

The resulting factored form of the algorithm is

$$\begin{aligned}\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right]\Delta\widehat{Q}^n = \\ -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right]\end{aligned} \tag{5.22}$$

We now have two implicit operators each of which is block tridiagonal. The structure of the block tridiagonal matrix is

$$\begin{bmatrix} [\ ] & [\ ] & & & & & & \\ [\ ] & [\ ] & [\ ] & & & & & \\ & [\ ] & [\ ] & [\ ] & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & -A & I & A & & \\ & & & & [\ ] & [\ ] & [\ ] & \\ & & & & & [\ ] & [\ ] & [\ ] \\ & & & & & & [\ ] & [\ ] \end{bmatrix} \tag{5.23}$$

Note that the thin layer implicit viscous term $\widehat{M}$ is kept with the η factor. Since it is a three point stencil, it will not affect the tridiagonal structure. Also when vectorization and parallization issues are considered the one dimensional form of the factored algorithm will be advantageous.

The solution algorithm now consists of two one-dimensional sweeps, one in the ξ and one in the η direction. The block matrix size is now at most $(\max[Jmax, Kmax] \cdot 4) \times (\max[Jmax, Kmax] \cdot 4)$. Each step requires the solution of a linear system involving a block tridiagonal which is solved by block LUD (lower-upper decomposition). The resulting solution process is much more economical than the unfactored algorithm in terms of computer storage and CPU time.

5.7 Reduced Forms of The Implicit Algorithm

Even though the factorization has improved the efficiency of the block implicit algorithm the major expense of the implicit scheme still resides in the block tridiagonal inversions. Compared to standard explicit algorithms the implicit scheme is still computationally expensive. The increased stability bounds of the implicit scheme offsets some of this disadvantage especially for problems where refined grids are used. In general, this holds true for time accurate applications where mesh refinement would unduly restrict the time steps for explicit schemes, but developments in multigrid techniques, see Jespersen [18] for a review, applied to steady state problems requires us to reexamine the implicit schemes. One way to capture back the advantage is to make the implicit scheme less computationally expensive, we will discuss other ways, such as accelerated convergence and improved accuracy, in later sections.

To improve the efficiency of a numerical scheme we can modify or simplify the algorithm so that the computational work is decreased. Most of the computational work for the implicit algorithm is tied to the block tridiagonal solution process. One way to reduce that work would be to reduce the block size for the tridiagonals. This can be accomplished by reducing the equation set from four variables (density, x-momentum, y-momentum, and energy) to three variables (density and the two momentum) by assuming constant total enthalpy, $H = (e + p)/\rho = H_0$ or similar thermodynamic approximations. The energy equation is then replace by the thermodynamic relation and the simplified set of equations can be solved. Such approximations can be restrictive in terms of the physical situations where they can be applied.

5.7a Diagonal Form

The computational work can also be decreased by introducing a diagonalization of the blocks in the implicit operators as developed by Pulliam and Chaussee [19]. The eigensystem of the flux Jacobians $\widehat{A}$ and $\widehat{B}$ are used in this construction. For now lets us again restrict ourselves just to the Euler equations, the application to the Navier-Stokes is discussed later.

The flux Jacobians $\widehat{A}$ and $\widehat{B}$ each have real eigenvalues and a complete set of eigenvectors. Therefore, the Jacobian matrices can be diagonalized, see Warming, Beam and Hyett [20],

$$\Lambda_\xi = T_\xi^{-1} \widehat{A} T_\xi \quad \text{and} \quad \Lambda_\eta = T_\eta^{-1} \widehat{B} T_\eta \tag{5.24}$$

with T_ξ the matrix whose columns are the eigenvectors of $\widehat{A}$ and T_η the corresponding eigenvector matrix for $\widehat{B}$. They are written out in the Appendix.

Here we take the factored algorithm in delta form, Eq. (5.22) and replace $\widehat{A}$ and $\widehat{B}$ with their eigensystem decompositions.

$$\begin{aligned} \left[T_\xi\, T_\xi^{-1} + h\ \delta_\xi \left(T_\xi\, \Lambda_\xi\, T_\xi^{-1}\right)\right] \ \left[T_\eta\, T_\eta^{-1} + h\ \delta_\eta \left(T_\eta\, \Lambda_\eta\, T_\eta^{-1}\right)\right] \ \Delta Q^n & \\ = \quad \text{the explicit right hand side of Eq. (5.22)} = \widehat{R}^n. & \end{aligned} \tag{5.25}$$

At this point Eq. (5.22) and (5.25) are equivalent. A modified form of Eq. (5.25) can be obtained by factoring the T_ξ and T_η eigenvector matrices outside the spatial derivative terms δ_ξ and δ_η. The eigenvector matrices are functions of ξ and η and therefore this modification reduces the time accuracy to at most first order in time, as shown in [19]. The resulting equations are

$$T_\xi \left[I + h\, \delta_\xi\, \Lambda_\xi\right]\ \widehat{N}\ \left[I + h\ \delta_\eta\, \Lambda_\eta\right] T_\eta^{-1} \Delta \widehat{Q}^n = \widehat{R}^n \tag{5.26}$$

where $\widehat{N} = T_{\xi}^{-1}T_{\eta}$, see Appendix.

The explicit side of the diagonal algorithm (the steady-state finite difference equations) is exactly the same as in the original algorithm, Eq. (5.22). The modifications are restricted to the implicit side and so if the diagonal algorithm converges, the steady-state solution will be identical to one obtained with the unmodified algorithm. In fact, linear stability analysis would show that the diagonal algorithm has exactly the same unconditional stability as the original algorithm. (This is because the linear stability analysis assumes constant coefficients and diagonalizes the blocks to scalars, the diagonal algorithm then reduces to the unmodified algorithm.) The modification (pulling the eigenvector matrices outside the spatial derivatives) of the implicit operator does affect the time accuracy of the algorithm. It reduces the scheme to at most first order in time and also gives time accurate shock calculations a nonconservative feature, i.e., errors in shock speeds and shock jumps, see [19]. The steady-state is in full conservation law form since the steady-state equations are unmodified. Also, computational experiments by Pulliam and Chaussee [19] have shown that the convergence and stability limits of the diagonal algorithm are similar to that of the unmodified algorithm.

The diagonal algorithm reduces the block tridiagonal inversion to 4×4 matrix multiplies and scalar tridiagonal inversions. The operation count associated with the implicit side of the full block algorithm is 410 multiplies, 356 adds, and 10 divides, a total of 776 operations, while the diagonal algorithm requires 233 multiplies, 125 adds, and 26 divides or 384 operations. Adding in the explicit side and other overhead such as I/O (input/output) and initialization, the overall savings in computational work can be as high as 40%. In fact the computational work can be further decreased by noting that the first two eigenvalues of the system are identical (see Appendix). This allows us to combine the coefficient calculations and part of the inversion work for the first two scalar operators.

The diagonal algorithm as presented above is really only rigorously valid for the Euler equations. This is because we have neglected the implicit linearization of the viscous flux $\widehat{S}^n$ in the implicit operator for the η direction. The viscous flux Jacobian $\widehat{M}^n$ is not simultaneously diagonalizable with the flux Jacobian $\widehat{B}^n$ and therefore to retain the full diagonalization we neglect it. For viscous flows we have investigated four options. One possibility is the use the block tridiagonal algorithm in the η direction and include the viscous Jacobian $\widehat{M}^n$. This increases the computational work and restricts us from using some of the convergence acceleration techniques which will be discussed below. Another option is to introduce a third factor to the implicit side of Eq. (8.2) where we use

$$\left[I - hRe^{-1}\delta_{\eta}\widehat{M}^n\right] \tag{5.27}$$

This again increases the computational work since we now have an added block tridiagonal inversion. We take these measures though because of the questionable stability of just completely neglecting the implicit viscous terms. The third option is to throw caution to the wind and actually neglect the viscous Jacobian, thereby gaining the increase efficiency of the diagonal algorithm. As long as the algorithm remains stable and convergent, the steady state obtained is identical for all three options since the explicit side is unchanged. The fourth option is to include a diagonal term on the implicit side which is an approximation to the viscous Jacobian eigenvalues. The estimate (taken from an examination of the terms of the Jacobian $\widehat{M}$) currently used is

$$\begin{aligned}\lambda_v(\xi) &= \overline{\mu Re^{-1} J^{-1}\left(\xi_x^2 + \xi_y^2\right)}\, J\rho^{-1} \\ \lambda_v(\eta) &= \overline{\mu Re^{-1} J^{-1}\left(\eta_x^2 + \eta_y^2\right)}\, J\rho^{-1}\end{aligned} \tag{5.28a}$$

which are added to the appropriate implicit operators in Eq. (5.26) with a differencing stencil taken from Eq. (5.8b). The new form of the diagonal allgorithm is given as

$$T_\xi \left[I + h\,\delta_\xi\,\Lambda_\xi - h\,I\,\delta_{\xi\xi}\lambda_v(\xi)\right]\,\widehat{N}\,\left[I + h\,\delta_\eta\,\Lambda_\eta - h\,I\,\delta_{\eta\eta}\lambda_v(\eta)\right]\,T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \tag{5.28b}$$

The terms in Eq. (5.28a) which are contained under the overbar are distinguished from the $J\rho^{-1}$ because in the application to the difference forms require those terms to be averaged fashion as in Eq. (5.8c). The ξ term is not added if the thin layer approximation is used. In all the above cases the explicit viscous operator is unchanged from the standard algorithm.

We have compared the four options for a number of test cases. For the first option, block tridiagonal with second order implicit dissipation the convergence rate was the slowest. For the second option, the third factor, fast convergence rates and stability were obtained at the expense of more computation. The third option, neglecting the viscous flux Jacobian, produced identical stability and convergence as the second option in most cases but required less computational work. In the fourth option (which is the recommended form) the convergence rates are typically the best and the overall robustness of a numerical code is improved. In all cases the converged solutions are identical.

5.7b Pressure–Velocity Splitting

Another way is to reduce the block size by similarity transformations as proposed by Steger [21]. This was originally restricted to Cartesian variables. Barth and Steger [16] have removed some of this restriction and developed an algorithm where two scalar tridiagonals and one block two by two tridiagonal inversion is required. The basic concept can be demonstrated in two-dimensional Cartesian coordinates, see Barth and Steger [16] for the extension to generalized coordinates.

The development of the sound speed - velocity splitting begins by considering the nonconservative form of the Euler equations

$$\partial_t R + M\partial_x R + N\partial_y R = 0 \tag{5.29a}$$

where

$$R = \begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix}, \quad M = \begin{bmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & \rho^{-1} \\ 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & u \end{bmatrix}, \quad N = \begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & \rho^{-1} \\ 0 & 0 & \gamma p & v \end{bmatrix} \tag{5.29b}$$

The eigenvalues of coefficient matrices, M and N, are the usual characteristic speeds

$$\Lambda_M = [u, u, u+c, u-c], \quad \Lambda_N = [v, v, v-c, v+c] \tag{5.30}$$

These coefficient matrices can each be split into two submatrices: one derived from the velocity part of the eigenvalues and the other from the sound speed part of the eigenvalues. A particular matrix splitting (there are many possibilities) was chosen to satisfy the following conditions

$$\Lambda(M) = \Lambda(M_u) + \Lambda(M_c), \quad \Lambda(M_u) = (u, u, u, u), \quad \Lambda(M_c) = (0, 0, c, -c)$$
$$\Lambda(N) = \Lambda(N_v) + \Lambda(N_c), \quad \Lambda(N_v) = (v, v, v, v), \quad \Lambda(N_c) = (0, 0, c, -c)$$

Specifically, M and N are split as

$$M = M_u + M_c = \begin{bmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho^{-1} \\ 0 & 0 & 0 & 0 \\ 0 & \gamma p & 0 & 0 \end{bmatrix}$$
$$N = N_v + N_c = \begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & v \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho^{-1} \\ 0 & 0 & \gamma p & 0 \end{bmatrix} \tag{5.31}$$

Given the coefficient matrices M and N, a similarity transformation exists that transforms these matrices into their conservative counterpart, the flux Jacobians A and B. $A = SMS^{-1}$, $B = SNS^{-1}$ where $S = \frac{\partial Q}{\partial R}$. Using this similarity transformation, M_c and N_c transform to $A_c = SM_cS^{-1}$ and $B_c = SN_cS^{-1}$ written out as

$$\begin{aligned} A_c =& (\gamma - 1)\begin{bmatrix} 0 & 0 & 0 & 0 \\ (u^2+v^2)/2 & -u & -v & 1 \\ 0 & 0 & 0 & 0 \\ a_{41}^c & a_{42}^c & -uv & u \end{bmatrix}, \\ B_c =& (\gamma - 1)\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ (u^2+v^2)/2 & -u & -v & 1 \\ b_{41}^c & -uv & b_{43}^c & v \end{bmatrix} \end{aligned} \tag{5.32}$$

where

$$a_{41}^c = [u(u^2+v^2)/2] - \gamma up/[\rho(\gamma-1)^2], \quad a_{42}^c = \gamma p/[\rho(\gamma-1)^2] - u^2$$

$$b_{41}^c = [v(u^2+v^2)/2] - \gamma vp/[\rho(\gamma-1)^2], \quad b_{43}^c = \gamma p/[\rho(\gamma-1)^2] - v^2$$

while A_u and B_v are

$$A_u = A - A_c, \quad B_v = B - B_c \tag{5.33}$$

This splitting produces matrices A_u and B_v that are more complex than A and B. But it is found that Q is an eigenvector of A_u and B_v, i.e.

$$A_u Q = uQ, \quad B_v Q = vQ \tag{5.34a}$$

which motivates the following substitution

$$AQ = (uI + A_c)Q, \quad BQ = (vI + B_c)Q \tag{5.34b}$$

Insertion of Eq. (5.34b) into the equations for local linearization of the Jacobians, the Cartesian equivalent of Eqs. (5.3), produces

$$E^{n+1} = E^n + (uI + A_c)^n (Q^{n+1} - Q^n) \tag{5.35a}$$

$$F^{n+1} = F^n + (vI + B_c)^n (Q^{n+1} - Q^n) \tag{5.35b}$$

Utilizing these linearizations in the basic algorithm equation (5.22) yields

$$L_x L_y \Delta Q = -\Delta t\, [\delta_x E^n + \delta_y F^n] \tag{5.36a}$$

with

$$L_x = [I + \theta \Delta t \delta_x (uI + A_c)^n] \tag{5.36b}$$

$$L_y = [I + \theta \Delta t \delta_y (vI + B_c)^n] \tag{5.36c}$$

The end result of this splitting is that the new operators L_x and L_y form matrices that no longer require 4×4 block tridiagonal inversions. In matrix operator form, we have

$$L_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \theta \Delta t \delta_x \begin{bmatrix} u & 0 & 0 & 0 \\ a_{21}^c & u + a_{22}^c & a_{23}^c & a_{24}^c \\ 0 & 0 & u & 0 \\ a_{41}^c & a_{42}^c & a_{43}^c & u + a_{44}^c \end{bmatrix} \tag{5.37a}$$

$$L_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \theta \Delta t \delta_y \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ b_{31}^c & b_{32}^c & v + b_{33}^c & b_{34}^c \\ b_{41}^c & b_{42}^c & b_{43}^c & v + b_{44}^c \end{bmatrix} \tag{5.37b}$$

where a^c and b^c are the respective elements of A_c and B_c given by Eq. (5.32).

In the L_x operator, for example, the first and third rows decouple from the system and can be solved as scalar tridiagonal matrices with their respective right-hand-sides. Once these rows are solved, the elements of the first and third columns can be moved to the right-hand-side. The second and fourth equations remain coupled and are solved as a 2×2 block tridiagonal matrix. The block decoupling of the L_y operator is even more conspicuous and is inverted (i.e., solved for) in a similar manner.

The use of the pressure–velocity splitting has substantially reduced the computational work over the basic block implicit scheme. A typical 2×2 block tridiagonal requires 55 operations per point, so the overall inversion, including the two scalar tridiagonals, requires 73 operations per entry. Because the two scalar tridiagonals have identical coefficients, this work can be even further cut by solving them together.

The matrix splitting produces the flux vectors

$$E = AQ = uIQ + A_cQ = E_u + E_c, \quad F = BQ = vIQ + B_cQ = F_v + F_c \tag{5.38}$$

where

$$E_u = \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho v u \\ ue \end{pmatrix}, \quad E_c = \begin{pmatrix} 0 \\ p \\ 0 \\ up \end{pmatrix}, \quad F_v = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 \\ ve \end{pmatrix}, \quad F_c = \begin{pmatrix} 0 \\ 0 \\ p \\ vp \end{pmatrix}$$

Note that the Jacobians of E_c and F_c are not A_c and B_c as defined above. Usually, the use of implicit linearizations which are not the Jacobians of the explicit flux vectors leads to restricted stability bounds or unconditionally instability. Linear stability analysis presented by Barth and Steger, as well as numerical experiment have shown though that the use of A_c and B_c leads to unconditional stability.

The generalized coordinate form of pressure–velocity splitting is developed in Barth and Steger [16]. A rotation transformation is used to align the momentum equations with generalized coordinate directions, e.g. in the ξ direction they use

$$C_\xi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\xi_x}{L_1} & \frac{\xi_y}{L_1} & 0 \\ 0 & -\frac{\xi_y}{L_1} & \frac{\xi_x}{L_1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.39}$$

with $L_1 = \sqrt{\xi_x^2 + \xi_y^2}$. This produces the transformed splitting matrix

$$\widetilde{A}_c = (\gamma - 1)\begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{L_1(u^2+v^2)}{2} & -U & -\widehat{V} & L_1 \\ 0 & 0 & 0 & 0 \\ a_{41}^c & a_{42}^c & -\frac{U\widehat{V}}{L_1} & U \end{bmatrix} \tag{5.40}$$

where

$$a_{41}^c = U[(u^2+v^2)/2 - \frac{c^2}{(\gamma-1)^2}], \quad a_{42}^c = L_1\frac{c^2}{(\gamma-1)^2} - \frac{U^2}{L_1}$$

with $\widehat{U} = \eta_y u - \eta_x v$.

The structure of Eq. (5.40) is identical to Eq. (5.32) so that the implicit operators in generalized coordinates are again reducible to 2 scalars and one 2×2 block operator for each direction. Barth and Steger also discuss the application of pressure–velocity splitting to the Navier-Stokes equations.

5.8 Metric Differencing and Invariants

Once a fluid dynamics code has been written, there are a number of first order checks which must be passed to assess accuracy and efficiency. A first test is that the code recovers free stream or uniform flow. In the case of arbitrary curvilinear coordinates and general finite differences this is not a trivial exercise. By construction finite volume schemes automatically balance fluxes and therefore they are not as susceptible to this type of error. There are a number of examples in the literature where finite volume schemes have been employed, see Jameson et.al. [6] or Rizzi and Erikson [5]. We prefer to employ finite difference formulations since they are usually more flexible, especially in the implementation of boundary conditions and in obtaining higher order accuracy. In this case the differencing used to form the flux derivatives and the differencing used to form the metrics must obey certain relations if free stream is to be captured.

As discussed in Section III, applying free stream or constant flow reduces the flow equations to the invariants of Eq. (3.11). Examining one of these terms $\partial_\xi(y_\eta)+\partial_\eta(-y_\xi)$ where central differences are used to form the metric terms, and using central differencing for the flux derivatives, we have,

$$\begin{aligned} &\delta_\xi^c\delta_\eta^c y - \delta_\eta^c\delta_\xi^c y = \\ &[y_{j+1,k+1} - y_{j-1,k+1} - y_{j+1,k-1} + y_{j-1,k-1}]/4 \\ &+[-y_{j+1,k+1} + y_{j+1,k-1} + y_{j-1,k+1} - y_{j-1,k-1}]/4 = 0 \end{aligned} \tag{5.41}$$

We see that central differencing in two dimension does satisfy the invariant relations. This becomes obvious when one realizes that second order central differencing operators commute, i.e. $\delta_\xi^c\delta_\eta^c = \delta_\eta^c\delta_\xi^c$. This is not true for general differences, e.g. $\delta_\xi^b \delta_\eta^c \neq \delta_\xi^c \delta_\eta^b$. Take the case of central differencing to form metrics and one sided backward differencing for the fluxes. Then we have,

$$\begin{aligned}
&\nabla_\xi \delta_\eta y - \nabla_\eta \delta_\xi y = \\
&[y_{j,k+1} - y_{j-1,k+1} - y_{j,k-1} + y_{j-1,k-1}]/2 \\
+&[-y_{j+1,k} + y_{j+1,k-1} + y_{j-1,k} - y_{j-1,k-1}]/2 \neq 0
\end{aligned} \tag{5.42}$$

The error associated with not satisfying the invariant relations is truncation error equal to or less than the lowest order accurate operator used. The error can be eliminated by modifying the difference formulas, for example introducing simple averages. The equivalent relationships for three dimensions can be very complicated. In most cases the error introduced is small except in regions of large mesh spacing or large distorted cells (high aspect ratios). It should be stressed though, that the satisfaction of the invariant relations is at least a high priority for any flow code. Hindman [12] has investigated this area for the Euler equations and Flores et.al. [13] give an interesting account of similar problems and solutions for the conservative full potential equations.

VI. Artificial Dissipation Added to Implicit Schemes

Even though linear stability analysis shows unconditional stability for the implicit algorithm, in practice stability bounds are encountered. This is especially true in strongly nonlinear cases, such as flows with shocks.

Whenever discrete methods are used to "capture" shocks (as opposed to fitting them), or to compute high Reynolds number viscous behavior, scales of motion appear which cannot be resolved by the numerics. These can be brought about by the nonlinear interactions in the convection terms of the momentum equations. If scale is represented by wave length or frequency, it can be easily shown that two waves interact as products to form a wave of higher frequency (the sum of the original two) and one of lower frequency (the difference). The lower frequencies do not cause a problem, but the continual cascading into higher and higher frequencies does. It is accounted for physically by shock formation (the harmonic analysis of a discontinuity contains all frequencies) or by viscous dissipation of the very high wave numbers. In numerical computations it can not be ignored and must be accounted for in the algorithm constructed. In any finite discrete mesh the cascading frequencies can eventually exceed the capacity of the mesh resolution at which point they can either, a) alias back into the lower frequencies or b) pile up at the higher frequency side. In either case, if uncontrolled, these terms can lead to serious inaccuracies and possible numerical instability.

The most common way of coping with the high-frequency cascade is to add to the complete algorithm some form of numerical dissipation with an error level that does not interfere with the accuracy of any physical viscous effects. This can be done in a variety of ways.

6.1 Constant Coefficient Implicit and Explicit Dissipation

Historically, in the class of implicit finite difference codes developed in the mid 1970's, a common procedure was to add explicit fourth order artificial dissipation to the central difference algorithm of the form

$$-\epsilon_e \Delta t J^{-1}[(\nabla_\xi \Delta_\xi)^2 + (\nabla_\eta \Delta_\eta)^2] J\widehat{Q}^n \tag{6.1a}$$

which is added to the right-hand side of Eq. (5.22) and implicit second-order smoothing

$$-\epsilon_i \Delta t J^{-1} \nabla_\xi \Delta_\xi J, \quad -\epsilon_i \Delta t J^{-1} \nabla_\eta \Delta_\eta J \tag{6.2b}$$

which was inserted into the respective implicit block operators. Second order implicit dissipation was used to keep the block implicit operators tridiagonal. The difference operators are defined as

$$\begin{aligned} \nabla_\xi q_{j,k} &= q_{j,k} - q_{j-1,k}, \quad \Delta_\xi q_{j,k} = q_{j+1,k} - q_{j,k} \\ \nabla_\eta q_{j,k} &= q_{j,k} - q_{j,k-1}, \quad \Delta_\eta q_{j,k} = q_{j,k+1} - q_{j,k} \end{aligned} \tag{6.3}$$

and are applied at all interior points. The parameter ϵ_e is chosen to be $O(1)$ and $\epsilon_i = 2\epsilon_e$. The smoothing terms are scaled with Δt which makes the steady state independent of the time step.

It is important to assess the effect on stability when these terms are added. In Section 7.2, we provide a linear analysis of the effect of added dissipation on stability. We summarize the results here. In the original development of the implicit algorithm we only added in the explicit dissipation, but this lead to a linear stability bound which was dependent on the magnitude of $\epsilon_e \Delta t$. The implicit second order term was added to eliminate this stability bound. The proper approach would be to make the fourth order dissipation implicit. This would then necessitate the use of block pentadiagonal solvers which is too computationally expensive. The second order implicit dissipation stabilizes the algorithm and allows us to retain block tridiagonal inversions. Linear analysis shows that if $\epsilon_i \gg \epsilon_e$ then unconditional stability is obtained. It should be noted that in practice for nonlinear problems the total algorithm has large but conditional stability bounds.

Beam and Bailey [22] suggest that while the implicit second order dissipation improves the practical stability bound, the use of fourth order implicit dissipation matching the explicit terms produces larger stability bounds and enhanced convergence. This is consistent with a concept which I will discuss in more detail below. That is, maximum stability bounds and optimal convergence rates are only achieved if we properly linearize the explicit side of the algorithm. In this case a proper linearization of the explicit dissipation produces improved stability and convergence. Beam and Bailey employed a block pentasolver which greatly increased the computational work and storage. We take advantage of the diagonal algorithm to produce a much more efficient scheme. Within the framework of the diagonal scheme we can replace the four scalar tridiagonals with scalar pentadiagonals which is just a minor increase in computational work. The resulting scheme has the advantage of increased stability bounds and convergence rates with the total computation work still less than the standard block tridiagonal scheme. Computational experiments demonstrate the increased efficiency and stability.

The approach of adding a constant coefficient fourth order explicit dissipation can produce some problems which are only evident in the case of refined meshes. Initially, because of computer limitations we only employed coarse grids and this type of dissipation was sufficient to produce stability and limited accuracy. With the advent of more powerful computers we have gone to grid refinement especially to resolve shocks. The use of the above type of fourth order dissipation with refine meshes produces wild oscillations near shocks even in cases where the computation is completely stable and converged. In Fig. 3, we show a converged solution for a NACA 0012 airfoil at a transonic Mach number, $M_\infty = 0.8$ and angle of attack, $\alpha = 0^\circ$ isolating the region near the shock.

As can be seen, the solution seems perfectly fine except in the region of the

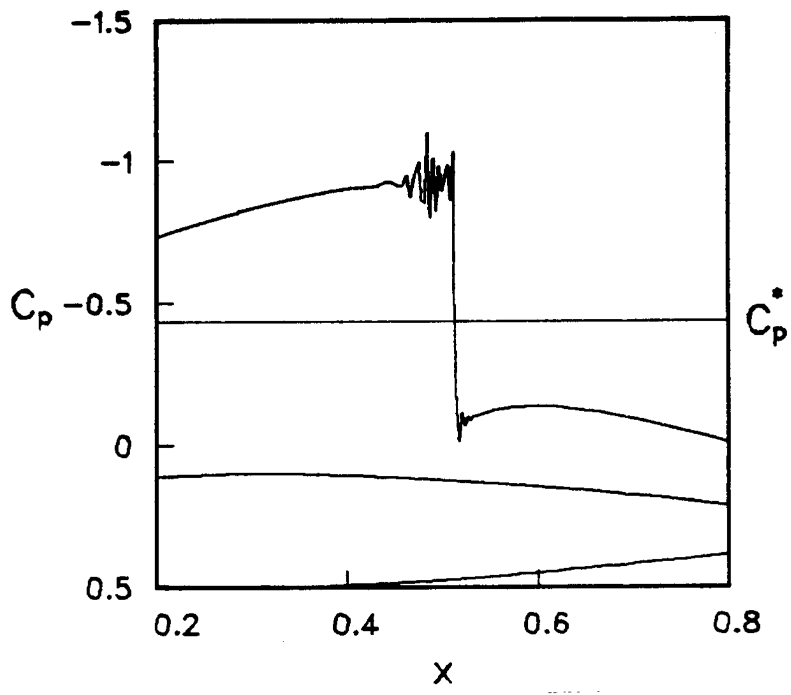


FIGURE 3. Coefficient of Pressure Showing Oscillations at Shock.

shock where a large every other point oscillation is evident. Varying the coefficient of artificial dissipation over a fairly wide range did not alter the nature of this oscillation. This is obviously an undesirable result which can be eliminated as shown below.

6.2 The Upwind Connection to Artificial Dissipation

In the early 1980's a number of schemes were developed based on upwind differencing. The flux split schemes of Steger and Warming [23], Roe [24], and Van Leer [25] employ a decomposition of the flux vectors in such a way that each element can be stably differenced in an upwind fashion. Other schemes of a similar nature but based on complicated theories are the flux difference scheme of Osher and Chakravarthy [26] and Harten's TVD methods [27]. These schemes all claim (with good justification) to be physically consistent since they follow in some sense the characteristics of the flow. They in general can be shown to produce sharp oscillation free shocks without added artificial dissipation. They are, though, complicated schemes which are just now being applied to complicated flowfield situations. Also these scheme have an inherent amount of internal dissipation, due to the one sided differences, which cannot be modified or decreased. It may be advantageous to have the flexibility of a simple central difference scheme with a controllable amount of artificial dissipation.

It can be shown (as done below) that the upwind schemes have an equivalence to central difference schemes with added dissipation. The central schemes are much simpler and more flexible and are therefore desirable if the dissipation can be added in an analogous fashion to the upwind schemes.

The plus - minus flux split method of Steger and Warming [23] will be used here to demonstrate the dissipative nature of upwind schemes. The approach taken is to split the eigenvalue matrix Λ of the flux Jacobians into two matrices, one with all positive eigenvalues and the other with all negative eigenvalues. Then the similarity transformations X or Y are used to form new matrices A^+, A^- and B^+, B^-. Formally,

$$A = X\Lambda_A X^{-1} = X(\Lambda_A^+ + \Lambda_A^-)X^{-1} = A^+ + A^- \tag{6.4a}$$

with

$$\Lambda_A^{\pm} = \frac{\Lambda_A \pm |\Lambda_A|}{2} \tag{6.4b}$$

Here, $|\Lambda|$ implies that we take the absolute values of the elements of Λ. The two matrices, A^+ and A^- have by construction all positive and all negative eigenvalues, respectively.

New flux vectors can be constructed as

$$\begin{aligned} E =& AQ = (A^+ + A^-)Q = E^+ + E^- \\ F =& BQ = (B^+ + B^-)Q = F^+ + F^- \end{aligned} \tag{6.5}$$

Different type of spatial differencing can now be used for each of the new flux vectors. One stable form is to use one sided backward differencing for the positive terms and one sided forward differencing for the negative terms. The one-sided difference operators are usually either first order accurate

$$\nabla_\xi^b\, u_{j,k} = \frac{u_{j,k} - u_{j-1,k}}{\Delta\xi} \quad \text{and} \quad \Delta_\xi^f\, u_{j,k} = \frac{u_{j+1,k} - u_{j,k}}{\Delta\xi} \tag{6.6a}$$

or second order accurate

$$\begin{aligned} \delta_\xi^b\, u_{j,k} &= \frac{\frac{3}{2}\, u_{j,k} - 2\, u_{j-1,k} + \frac{1}{2}\, u_{j-2,k}}{\Delta\xi} \\ \delta_\xi^f\, u_{j,k} &= \frac{-\frac{3}{2}\, u_{j,k} + 2\, u_{j+1,k} - \frac{1}{2}\, u_{j+2,k}}{\Delta\xi} \end{aligned} \tag{6.6b}$$

Similar expressions are used for the η derivatives. Note that $\Delta\xi = 1$, bull will appear in formulas where its presence conveys meaning.

The plus-minus matrices, A^+ and A^- can be written as

$$A^\pm = X\left(\frac{\Lambda \pm |\Lambda|}{2}\right)X^{-1} = \frac{A \pm |A|}{2} \tag{6.7a}$$

which gives

$$E^\pm = A^\pm Q = \frac{A}{2}Q \pm \frac{|A|}{2}Q = \frac{E}{2} \pm \frac{|A|}{2}Q \tag{6.7b}$$

Similar expressions are obtainable for the B matrices and flux vector F.

Examining the flux derivative

$$\delta^b_\xi E^+ + \delta^f_\xi E^- \tag{6.8a}$$

where second order one sided difference approximations are chosen

$$\begin{aligned} \delta^b_\xi &= (3I - 4\mathcal{E}^{-1} + \mathcal{E}^{-2})/(2\Delta\xi) \\ \delta^f_\xi &= (-3I + 4\mathcal{E}^{+1} - \mathcal{E}^{+2})/(2\Delta\xi) \end{aligned} \tag{6.8b}$$

with $\mathcal{E}^i$ the shift operator, i.e., $\mathcal{E}^{\pm i}u_j = u_{j\pm i}$.

Combining Eqs. (6.7b) and (6.8) we have

$$\frac{1}{2}\left[(\delta^b_\xi + \delta^f_\xi)E + (\delta^b_\xi - \delta^f_\xi)|A|Q\right] \tag{6.9}$$

for the difference equation.

It is easily shown that

$$(\delta^b_\xi + \delta^f_\xi)/2 = (-\mathcal{E}^{+2} + 4\mathcal{E}^{+1} - 4\mathcal{E}^{-1} + \mathcal{E}^{-2})/(4\Delta\xi) = \overline{\delta}_\xi \tag{6.10a}$$

which is a second order central difference operator, but not δ_ξ. The other term of Eq. (6.9) is of more interest, where

$$(\delta^b_\xi - \delta^f_\xi)/2 = (\mathcal{E}^{+2} - 4\mathcal{E}^{+1} + 6I - 4\mathcal{E}^{-1} + \mathcal{E}^{-2})/(4\Delta\xi) = \frac{1}{4\Delta\xi}(\Delta_\xi\nabla_\xi)^2 \tag{6.10b}$$

which is a fourth order difference stencil. The difference operators are define in Eq. (6.3).

Now Eq. (6.9) can be written as

$$\left(\overline{\delta}_\xi E + \frac{1}{4\Delta\xi}(\Delta_\xi\nabla_\xi)^2|A|Q\right) \tag{6.11a}$$

The form now is a second order central difference term plus fourth order dissipation. The dissipative term is a natural consequence of the upwind differencing. It is interesting to note that the central difference term Eq. (6.10a) is not the standard three point difference. If first order formulas are employed for the upwind differences then a similar analysis would produce the standard second order three point central differencing plus a second order dissipative term. For instance, Eq. (6.11a) is replace by

$$\left(\delta_\xi E - \frac{1}{2\Delta\xi}(\Delta_\xi \nabla_\xi)|A|Q\right) \tag{6.11b}$$

We note a number of things from the form of Eqs. (6.11) which can guide us in developing artificial dissipation models for a central difference scheme. Adding fourth order dissipation to a central difference produces the equivalent of some second order upwind scheme. The use of second order dissipation can produce a first order upwind equivalent. Research has shown that applying flux limiters to upwind schemes and some of the TVD concepts suggest that the best approach for an upwind algorithm is to use a locally first order upwind difference at a shock and second order elsewhere. This can be accomplished by some switching and transitioning of second order and fourth order dissipation added to a central scheme. The coefficients for the dissipation parts of Eq. (6.11) suggest some sort of flux Jacobian scaling where for instance a spectral radius of the Jacobians could be used.

6.3 Nonlinear Artificial Dissipation Model

As seen from the previous analysis a mixed second and fourth order dissipation model with appropriate coefficients should produce a scheme with good shock capturing capabilities. Jameson et.al. [6] has employed a dissipative model of such a form where second and fourth order dissipation are combined. The model rewritten in our notation is

$$\nabla_\xi \left(\sigma_{j+1,k} J^{-1}_{j+1,k} + \sigma_{j,k} J^{-1}_{j,k}\right)\left(\epsilon^{(2)}_{j,k}\Delta_\xi Q_{j,k} - \epsilon^{(4)}_{j,k}\Delta_\xi\nabla_\xi\Delta_\xi Q_{j,k}\right) \tag{6.12}$$

with

$$\begin{aligned}\epsilon^{(2)}_{j,k} &= \kappa_2 \Delta t \max(\Upsilon_{j+1,k}, \Upsilon_{j,k}, \Upsilon_{j-1,k})\\ \Upsilon_{j,k} &= \frac{|p_{j+1,k} - 2p_{j,k} + p_{j-1,k}|}{|p_{j+1,k} + 2p_{j,k} + p_{j-1,k}|}\\ \epsilon^{(4)}_{j,k} &= \max(0, \kappa_4\Delta t - \epsilon^{(2)}_{j,k})\end{aligned} \tag{6.13}$$

where typical values of the constants are $\kappa_2 = 1/4$ and $\kappa_4 = 1/100$. Similar terms are used in the η direction. The term $\sigma_{j,k}$ is a spectral radius scaling and is defined as

$$\sigma_{j,k} = |U| + a\sqrt{\xi_x^2 + \xi_y^2} \tag{6.14}$$

which is the spectral radii of $\widehat{A}$, the spectral radius of $\widehat{B}$ is used for the η dissipation.

The first term of Eq. (6.12) is a second order dissipation with an extra pressure gradient coefficient to increase its value near shocks. The second term is a fourth order term where the logic to compute $\epsilon_{j,k}^{(4)}$ switches it off when the second order nonlinear coefficient is larger then the constant fourth order coefficient. This occurs right near a shock. In Figs. 4 and 5 , we show solutions for the flow problem of Fig. 3, using this nonlinear artificial dissipation. For Fig. 4 we employ just the fourth order term, i. e. $\kappa_2 = 0$.

The oscillations at the shock are eliminated and a sharp shock is obtained. In this case though there is an overshoot and undershoot at the top and bottom of the shock which is eliminated in Fig. 5 by adding the second order term, $\kappa_2 = 1/4$.

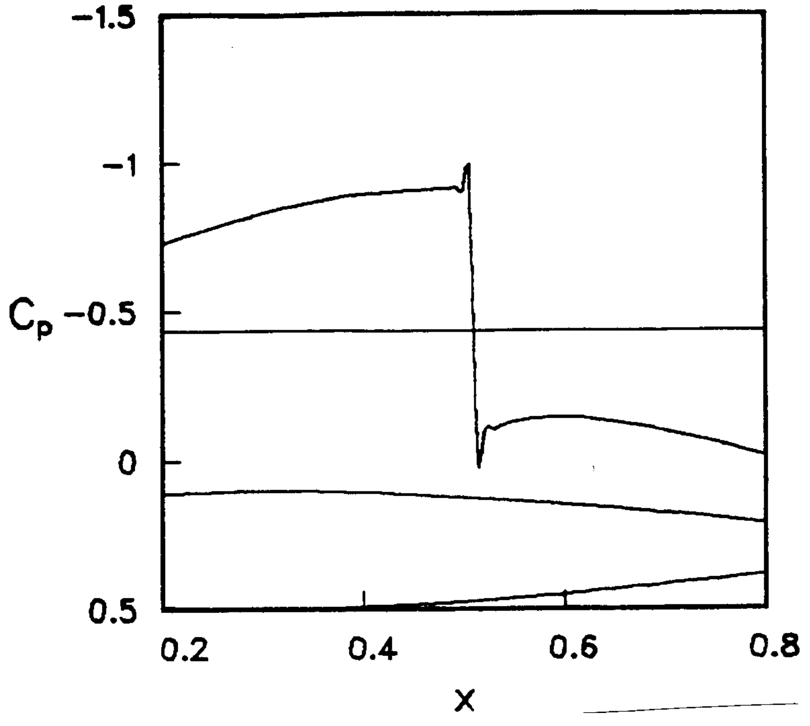


FIGURE 4. Coefficient of Pressure Obtained Using Fourth Order Nonlinear Dissipation.

The results shown are fully converged to machine zero and in the case of Fig. 5 represent the current quality of our shock capturing capabilities. The chosen values

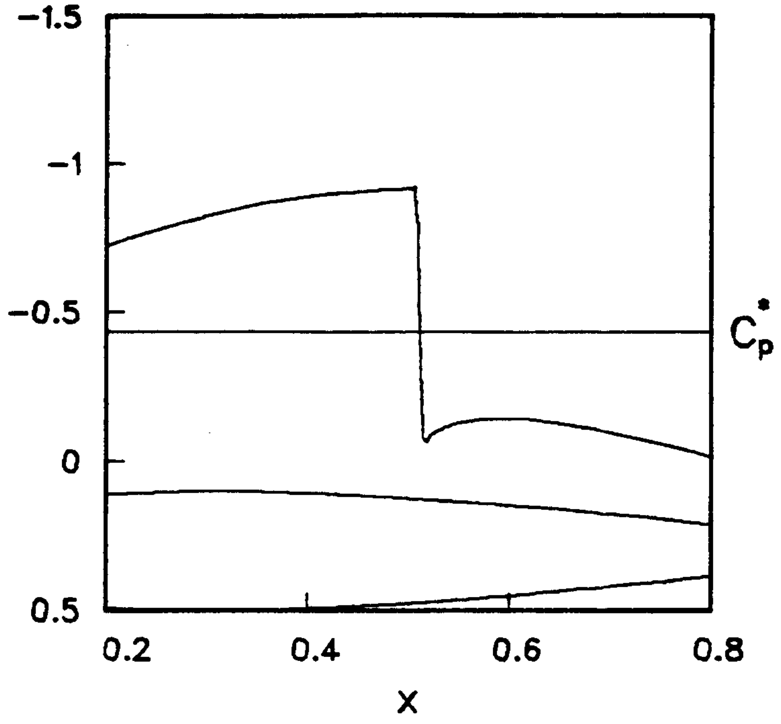


FIGURE 5. Coefficient of Pressure Obtained Using Second + Fourth Order Nonlinear Dissipation.

of the coefficients have, at least to date, been static and are not changed from case to case.

The implicit dissipation used with Eq. (6.12) is the linearization of the equation treating the pressure coefficient Υ and the spectral radius σ as space varying functions but ignoring their functional dependency on Q. Then the dissipation is linear in $Q_{j,k}$ and is added to the diagonal algorithm again necessitating scalar pentadiagonal solvers. This produces a very efficient, stable and convergent form of the implicit algorithm.

Near computational boundaries we modify the fourth order dissipation so as to maintain a dissipative term. A derivation and analysis of various boundary treatments in given in Ref. [28]. The modification is needed at the first interior point (e.g. $Q_{j+1,k}$) where the five point fourth order term $Q_{j+2,k} - 4Q_{j+1,k} + 6Q_{j,k} - 4Q_{j-1,k} + Q_{j-2,k}$ is to be applied. There the point $Q_{j+2,k}$ doesn't exist, the formula is modified to a one sided second order term with the differencing stencil $-2Q_{j+1,k} + 5Q_{j,k} - 4Q_{j-1,k} + Q_{j-2,k}$. Similar formulas are used at other boundaries.

6.4 Total Variation Diminishing Schemes, TVD

The development of monotone, flux vector/difference splitting, TVD and other nonoscillatory schemes can be found in numerous publications, see for example, Refs. [7,8,9,23,24,25,26,27]. Here we shall just briefly define the conditions for a class of TVD schemes introduced by Harten [27].

The conditions for a scheme to be TVD in Harten's sense can be developed for the scalar hyperbolic conservation law

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0 \tag{6.15}$$

where f (the flux) is a nonlinear function of u . We can define a characteristic speed $a(u)=\partial f/\partial u$.

A one parameter family of schemes can be defined

$$u_j^{n+1}+\lambda\theta\left(h_{j+\frac{1}{2}}^{n+1}-h_{j-\frac{1}{2}}^{n+1}\right)=u_j^n-\lambda(1-\theta)\left(h_{j+\frac{1}{2}}^{n}-h_{j-\frac{1}{2}}^{n}\right) \tag{6.16a}$$

rewritten as

$$Lu^{n+1}=Ru^n \tag{6.16b}$$

4where $u_j^n=u(j\Delta x,n\Delta t)$, $\lambda=\Delta t/\Delta x$, θ parameterizes the equations from the fully explicit to fully implicit forms, and h is the numerical flux function with $h_{j\pm\frac{1}{2}}=h\left(u_{j\mp1},u_j,u_{j\pm1},u_{j\pm2}\right)$.

The total variation of a mesh function u^n is defined as

$$TV\left(u^n\right)=\sum_{j=-\infty}^{\infty}\left|u_{j+1}^n-u_j^n\right|=\sum_{j=-\infty}^{\infty}\left|\Delta_{j+\frac{1}{2}}u^n\right| \tag{6.17}$$

where $\Delta_{j+\frac{1}{2}}=u_{j+1}-u_j$.

A numerical scheme is TVD is

$$TV\left(u^{n+1}\right)\leq TV\left(u^n\right) \tag{6.18}$$

For Equation (6.16) the conditions due to Harten [27] are

$$TV\left(Ru^n\right)\leq TV\left(u^n\right) \tag{6.19a}$$

and

$$TV\left(Lu^{n+1}\right)\geq TV\left(u^{n+1}\right) \tag{6.19b}$$

Rewritting Eq. (6.16), assuming h is Lipschitz continuous,

$$\begin{aligned} u_j^{n+1} - \lambda\theta \left(C^-_{j+\frac{1}{2}} \Delta_{j+\frac{1}{2}} u - C^+_{j-\frac{1}{2}} \Delta_{j-\frac{1}{2}} u \right)^{n+1} &= \\ u_j^n + \lambda(1-\theta) \left(C^-_{j+\frac{1}{2}} \Delta_{j+\frac{1}{2}} u - C^+_{j-\frac{1}{2}} \Delta_{j-\frac{1}{2}} u \right)^n & \end{aligned} \tag{6.20}$$

with $C^{\mp}$ bounded functions. Sufficient conditions for Eq. (6.19) are for all j :

$$\begin{aligned} \lambda(1-\theta) C^{\mp}_{j+\frac{1}{2}} &\geq 0 \\ \lambda(1-\theta) \left(C^+_{j+\frac{1}{2}} + C^-_{j+\frac{1}{2}} \right) &\leq 1 \\ -\infty < C \leq -\lambda\theta C^{\pm}_{j+\frac{1}{2}} &\leq 0 \end{aligned} \tag{6.21}$$

for finite C.

These conditions can be used to analyze and construct various TVD schemes. Refer to References [27], [7] and [8] for two forms of high resolution (at least second order accurate) TVD schemes applied to hyperbolic conservation law equations.

VII. Time Accuracy, Steady States, Convergence and Stability

7.1 Time Accuracy vrs Steady-State Computation

The implicit algorithm is designed to be time accurate where second order accuracy can be maintained and the equations are integrated through time from some meaningful initial condition to the the solution at time T. In this case the time step is chosen to be commensurate with some time scale of the problem. The evolution of the solution through time is physically realistic and good solution accuracy is dependent on the mesh spacing and boundary conditions.

The equations can also be applied to steady-state problems. Typically we employ the first order scheme in time and attempt to accelerate the algorithm by various non-time-like maneuvers. The equations are then integrated from an arbitrary initial condition to a time asymptotic state. Any procedure which drives us to the steady-state faster must also be stable and accurate at convergence. It might be expected that large time steps could be used to drive the solution to the steady-state faster. As we shall see, based on linear analysis large time steps can increase the convergence rate, but for factored forms the limit of the amplification factor (a measure of the maximum convergence rate) as $h = \Delta t \to \infty$ is 1.

A. Effect of Factorization Errors On Convergence

Let us divide the total solution into the transient (time-like) and particular (steady-state) parts. The goal of any fast steady-state algorithm is to eliminate the transient as quickly as possible. We can examine the ability of the implicit scheme to eliminate transients by investigating the model problem, Eq. (5.12). In this case, instead of Eq. (5.14) we take

$$w = \widetilde{w}(t)\, e^{i\beta\, x} \tag{7.1}$$

and treat the spatial derivative ∂_x analytically, then examine the temporal differencing schemes in one and two dimensions. This gives us the purely transient one dimensional model problem

$$\widetilde{w}_t + \lambda_x\, \widetilde{w} = 0 \tag{7.2}$$

with $\lambda_x = i\beta\lambda$.

The delta form of the first order implicit algorithm is

$$(1 + h\, \lambda_x)\, \Delta\widetilde{w}^n = -h\lambda_x\, \widetilde{w}^n \tag{7.3}$$

which can be rewritten as

$$\widetilde{w}^{n+1} = \left[\frac{1}{(1 + h\lambda_x)}\right] \widetilde{w}^n$$

$$\text{or} \tag{7.4}$$

$$\widetilde{w}^n = \left[\frac{1}{(1 + h\lambda_x)}\right]^n \widetilde{w}^0$$

where $\widetilde{w}^0$ is some initial value. The term in the brackets is the amplification factor, σ.

For $h \to \infty$, $\widetilde{w}^n \to 0$ and the transient can actually be eliminated directly for large h.

In contrast, let us examine a two-dimensional factored implicit scheme for the two-dimensional transient problem

$$\widetilde{w}_t + \lambda_x \widetilde{w} + \lambda_y \widetilde{w} = 0 \; . \tag{7.5}$$

This is the two-dimensional counterpart to Eq. (7.2). Applying the first order implicit approximate factorization delta algorithm to Eq. (7.5) we have

$$[1 + h\,\lambda_x]\;[1 + h\,\lambda_y]\;\Delta\widetilde{w}^n = -h\;(\lambda_x + \lambda_y)\;\widetilde{w}^n \tag{7.6}$$

Expanding $\Delta\widetilde{w}^n = \widetilde{w}^{n+1} - \widetilde{w}^n$ and combining terms we have

$$\widetilde{w}^n = \left[\frac{\left(1 + h^2\;\lambda_x\lambda_y\right)}{\left(1 + h\;\lambda_x + h\;\lambda_y + h^2\;\lambda_x\lambda_y\right)}\right]^n \widetilde{w}^0 \tag{7.7}$$

and so $|\sigma| \to 1$ as $h \to \infty$.

A close examination of this result shows that the factorization has destroyed the good convergence characteristics at large time steps. The factoring error term has introduced a h^2 term in the numerator of the amplification factor. Therefore the factored schemes do not have good convergence characteristics for large time steps. Actually, there is a range of moderately large time steps where the amplification factor is a minimum, see for instance Abarbanel, Dwoyer, and Gottlieb [29]. Convergence can therefore be accelerated by using a time step which minimizes the amplification factor.

Note that for the delta form of the algorithm (either factored or unfactored) the steady-state solution is independent of the time step, h. (There are numerical schemes where this is not the case, such as Lax-Wendroff.) Therefore, the time step path to the steady-state does not affect the final solution and we can envision using time step sequences or spatially variable time steps to accelerate convergence.

B. Space Varying Δt

Manipulation of the time step can have a substantial influence on convergence even within the framework of the factored algorithms. If only a steady state solution is required, one can let h (or Δt) change in space. This approach can be view as a way to condition the iteration matrix of the relaxation scheme defined via Eq. (5.22) or Eq. (5.26). Use of a space varying Δt can also be interpreted as an attempt to use a more uniform Courant number throughout the field. In any event, changing Δt can be effective for grid spacings that vary from very fine to very coarse

- a situation usually encountered in aerodynamic simulations where grids contain a wide variety of length scales.

A space varying Δt has been used in both explicit and implicit schemes (e.g. Shang and Hankey [30], McDonald and Briley [31], Shirnivasan et al [32], Coakley [33], Jameson [6], etc). As a rule one wishes to adjust Δt at each point proportional to the grid spacing and the characteristic speed of the flow. Something like the Courant number restriction (which for the Euler equations in multi-dimensions is a bit of an approximation).

For highly stretched grids the space variation of the grid is the most important parameter to scale with. In subsonic and transonic flow the characteristic speeds have moderate variation and we have found that a purely geometric variation of Δt is adequate, specifically

$$\Delta t = \frac{\Delta t|_{ref}}{1. + \sqrt{J}} \tag{7.8a}$$

To illustrate the advantage of using a variable time step, Fig. 6 shows the degradation in convergence rate when a constant step size is substituted for the variable time step in a NACA 0012 test case. For this comparison all other possible parameters were held constant and no other changes were employed. We should note at this time that the above variation of time step has often worked poorly in viscous flow until the numerical dissipation terms were also put in implicitly as described later. Also other forms of the variable step size sometimes perform better than Eq. (7.8a1), for example

$$\Delta t = \frac{\Delta t|_{ref}}{|U| + |V| + a\sqrt{\xi_x^2 + \xi_y^2 + \eta_x^2 + \eta_y^2}} \tag{7.8b}$$

which is approximately a constant CFL condition. However, Eq. (7.8b) is more costly to compute then Eq. (7.8a).

C. Mesh Sequences

For inviscid airfoil calculations on a grid of O(250 x 50) practical convergence is usually obtained in 500-600 fine grid iterations when the flow field has been started from an initial condition of uniform free stream flow. Typically the first 100 to 200 iterations on the fine mesh are needed to get past the initial transients which can be a substantial portion of the total solution time. For instance, in the above test case it takes on the order of 600 fine grid iterations for a tight convergence criteria (e.g. lift to 5 decimal places) , 200 of which are spent on clearing out the impulsive start. One way to accelerate convergence to a steady state is to obtain a good initial guess for a fine mesh by first iterating on a sequence of coarse grids and then interpolating the solution up to the next refined grid. Such a mesh sequence procedure can often reduce the amount of time required to obtain a solution to plotable accuracy by

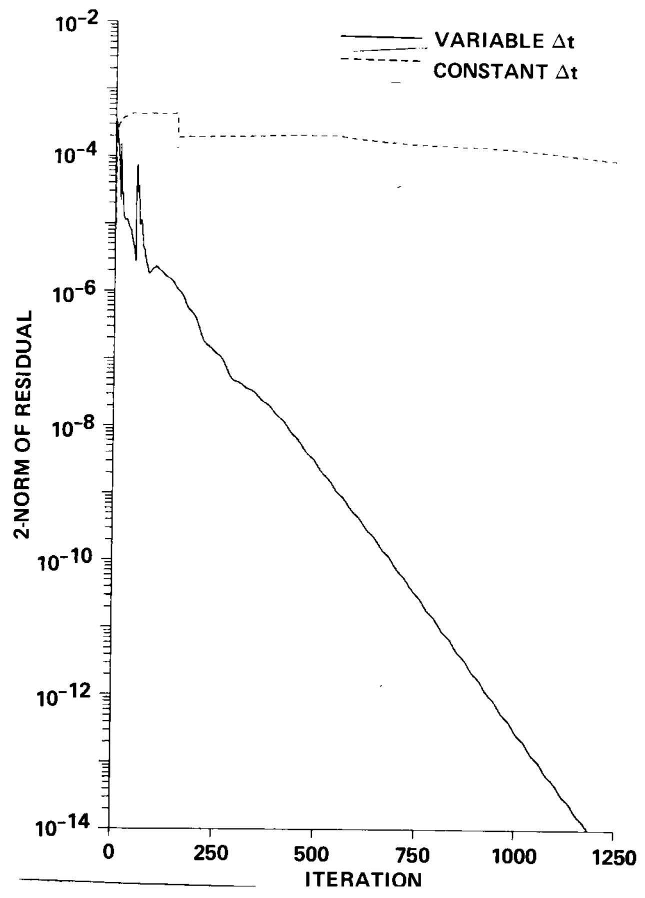


FIGURE 6. Convergence Improvement Due to Variable Time Step.

a factor of two. Also, because a coarse grid tends to damp high frequency waves, using a mesh sequence procedure can improve the overall robustness of the code.

A mesh sequencing procedure can be implemented in an optionally called stand alone routine. If a sequence of m grids are used, a coarsened grid is cut from each previous grid by halving the number of points in the ξ-direction and by regenerating a new η-distribution of points in the η-direction using a fewer number of points. The η-distribution is regenerated because in viscous flows the spacing near the wall would be too coarse if the halving procedure is used. A finite number of iterations (perhaps 50) are carried out on each coarsened grid at which point the approximate solution is interpolated onto a more refined grid. The finest grid is then iterated to convergence. The result is faster convergence to practical levels and a more robust starting procedure.

For a NACA 0012 test case a sequence of 3 grids has been used; 48 by 18 and

96 by 25 and the final grid of 192 by 33 points. The convergence of C_l is shown in Fig. 7 to indicate the overall improvement in convergence due to the use of mesh sequencing in comparison to the use of a fine grid only. Both cases were started with a free stream initial condition.

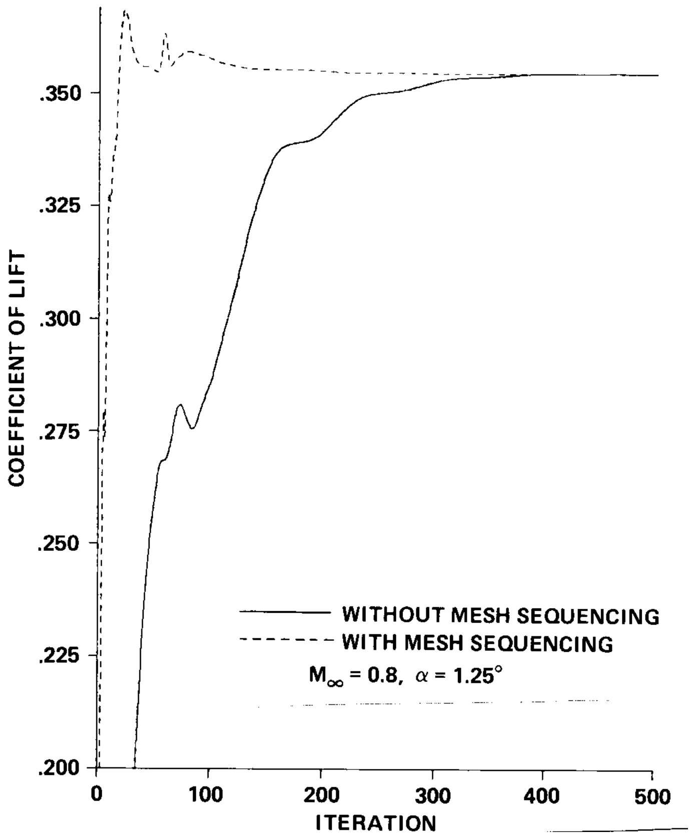


FIGURE 7. Improvement In Total Convergence of Lift Due to Mesh Sequencing.

7.2 Effect of Dissipation Model on Convergence and Stability

As discussed in Section VI., based on linear theory the use of explicit dissipation produces an explicit stability bounds unless implicit dissipation is added. The second-difference dissipation, Eq. (6.2b), will stabilize the fourth-difference dissipation if the coefficients are chosen properly. Ideally though, it would be better to treat the explicit dissipation in a fully implicit manner. That is, use implicit fourth-difference dissipation which is an exact linearization of the explicit fourth-difference dissipation. In fact, although the implicit second-difference dissipation stabilizes the fourth-difference dissipation it can have a detrimental effect on the convergence rates of an implicit algorithm for steady-state computations.

Consider the model problem in one-dimension (equivalent to Eq.(5.12) with a convenient change in notation),

$$q_t + aq_x = 0 \tag{7.9}$$

Applying the first-order time accurate Euler implicit scheme in delta form to Eq. (7.9) and adding explicit fourth-difference dissipation ($\beta_4 > 0$), implicit second-difference dissipation ($\alpha_2 > 0$), and implicit fourth-difference dissipation ($\alpha_4 > 0$) gives the algorithm

$$\left[1 + ha\delta_x - h\alpha_2\nabla_x\Delta_x + h\alpha_4(\nabla_x\Delta_x)^2\right](q^{n+1} - q^n) = -h\left(a\delta_x + \beta_4(\nabla_x\Delta_x)^2\right)q^n \tag{7.10}$$

Fourier analysis using $q^n = w^n e^{ik_j j\Delta x}$ (with k_j the wave number in x) produces

$$\left[1 + ha\lambda_x - h\alpha_2\mu_x + h\alpha_4\mu_x^2\right](w^{n+1} - w^n) = -h\left(a\lambda_x + \beta_4\mu_x^2\right)w^n \tag{7.11}$$

where $\lambda_x = 2i\sin(k_j\Delta x)/\Delta x$ represents the Fourier signature for the central difference δ_x, $\mu_x = -2 + 2\cos(k_j\Delta x)$ the signature of the second-difference dissipation operator $\nabla_x\Delta_x$, and μ_x^2 for the fourth-difference dissipation.

The amplification factor for $w^{n+1} = \sigma w^n$ is then

$$\sigma = \frac{1 + h\left((\alpha_4 - \beta_4)\mu_x^2 - \alpha_2\mu_x\right)}{1 + h\left(a\lambda_x - \alpha_2\mu_x + \alpha_4\mu_x^2\right)} \tag{7.12}$$

The choices which will be investigated are

1. $\beta_4 \neq 0$ and $\alpha_4 = \alpha_2 = 0$, explicit dissipation only.
2. $\beta_4 \neq 0$, $\alpha_2 \neq 0$, and $\alpha_4 = 0$, explicit fourth-difference dissipation and implicit second-difference dissipation, no implicit fourth-difference dissipation.
3. $\beta_4 \neq 0$, $\alpha_4 \neq 0$ and $\alpha_2 = 0$, explicit and implicit fourth-difference dissipation with no implicit second-difference dissipation.

For case 1, explicit dissipation only, Eq. (7.12) becomes

$$\sigma = \frac{1 - h\beta_4\mu_x^2}{1 + ha\lambda_x} \tag{7.13a}$$

Now, since λ_x is pure imaginary and has a minimum of 0, and $-4 \leq \mu_x \leq 0$ the explicit stability bound is $h\beta_4 < \frac{1}{8}$. This is a limit on the product of h and β_4 and therefore one can always find a combination which will be stable. But, for arbitrary h, especially in the case where large h are used to accelerate convergence, this bound is too restrictive.

In the second case, implicit second-difference dissipation can eliminate the above stability bound. The amplification factor σ is now

$$\sigma = \frac{1 - h\beta_4\mu_x^2 - h\alpha_2\mu_x}{1 + ha\lambda_x - h\alpha_2\mu_x} \tag{7.13b}$$

The numerator term λ_x can only improve the stability bounds since it is pure imaginary, so it is taken at its minimum, 0. Let $\alpha_2 = 2\beta_4$ and apply the stability condition $|\sigma| \leq 1$ which results in the condition $-2 \leq -h\beta_4\mu_x(4 + \mu_x)$. Since $\mu_x \leq 0$, the condition can be rewritten as $-2 \leq h\beta_4|\mu_x|(4 + \mu_x)$ which is satisfied because $-4 \leq \mu_x$. Therefore, using $\alpha_2 = 2\beta_4$ leads to unconditional stability. The disadvantage of this is form is evident from the amplification factor, Eq. (7.13b). Even though the scheme is unconditionally stable, $\sigma \to 1$ as $h \to \infty$. In fact, the amplification factor has a minimum at a finite h and then asymptotes rapidly to 1 as h increases. For this reason, large h cannot be used to accelerate convergence even in this simple one-dimensional example.

In contrast, the third case of implicit and explicit dissipation is unconditionally stable and has good convergence characteristics for large h. The amplification factor σ for $\alpha_4 = \beta_4$ and $\alpha_2 = 0$ is

$$\sigma = \frac{1}{1 + h\left(a\lambda_x + \alpha_4\mu_x^2\right)} \tag{7.13c}$$

which is unconditionally stable and $\sigma \to 0$ as $h \to \infty$.

The analysis for two and three dimensions is straightforward and gives similar results for the unfactored forms. The optimal algorithm is a fully implicit one. In general, optimal stability and convergence only occurs for the fully implicit form of the algorithm.

We demonstrate the improved convergence and stability bounds in Fig. 8.

The curves in Fig. 8 are convergence histories for a transonic airfoil computation showing the effect of a fully implicit treatment of the artificial dissipation. The upper curve is the result of second order constant coefficient implicit dissipation, Eq. (6.2b), with nonlinear explicit dissipation, Eq. (6.12). A much faster convergence rate is obtained in this problem when the second order implicit dissipation is replaced by an implicit linearization of the nonlinear dissipation of Eq. (6.12), see Ref. [28] for more details. Also the maximum allowable time step is at least 10 times larger for the fully implicit scheme.

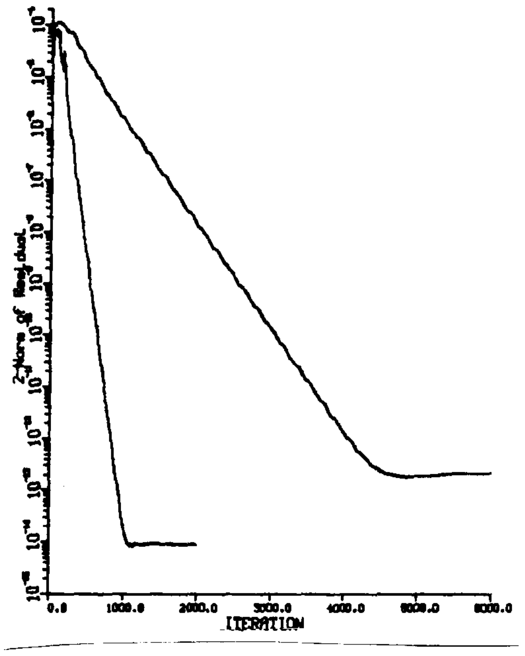


FIGURE 8. Improvement in Convergence Rate Due to Implicit Treatment of Artificial Dissipation.

VIII. ARC2D - ARC3D Algorithms

General purpose centrally space differenced implicit finite difference codes in two [2] and three [3] dimensions have been developed at NASA Ames and have been widely distributed since their introduction in 1977 and 1978. These codes, now referred to ARC2D and ARC3D, can run either in inviscid or viscous mode for steady or unsteady flow. They use general coordinate systems and can be run on any smoothly varying curvilinear mesh, even a mesh that is quite skew. Because they use well ordered finite difference grids, the codes can take advantage of vectorized computer processors and have been implemented for the Control Data 205 and the CRAY 1-S and X-MP. On a single processor of the X-MP a vectorized version of the code runs approximately 20 times faster than the original code which was written for the Control Data 7600.

Traditionally gains in computational efficiency due to improved numerical algorithms have kept pace with gains due to increased computer power. Since the ARC2D and ARC3D codes were introduced, a variety of algorithmic changes have been individually tested and have been shown to improve overall computational efficiency. These include use of a spatially varying time step (Δt), use of a sequence of mesh refinements to establish approximate solutions, implementation of various ways to reduce inversion work, improved numerical dissipation terms, and more implicit treatment of terms. Although the various individual algorithm improvements can interact with each other, sometimes adversely making optimization difficult, their combined effect has lead to an order of magnitude gain in computational efficiency for steady state applications. This is a gain equivalent to that achieved with computer hardware. Unsteady flow calculations have also benefited from some of the above improvements.

We now summarize the two basic algorithms used in the code ARC2D, see Section VIII for details of ARC3D. The standard algorithm is used mainly for time accurate calculations. The equations are reproduce from Eq. (5.22)

$$\begin{aligned}\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right]\Delta\widehat{Q}^n &= \widehat{R}^n \\ \widehat{R}^n = -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right]\end{aligned} \tag{8.1}$$

This scheme consists of first forming the right hand side, $\widehat{R}^n$ then performing two block tridiagonal inversions. Central differences are used for the flux and Jacobian differences. The dissipation models used are the implicit second order, Eqs. (6.1b), added to the appropriate implicit operator to keep the band width tridiagonal and the explicit nonlinear term Eq. (6.12). Since this scheme is used for time accurate calculation the typical time step will be small enough to assure stability even though the explicit dissipation operator is not properly linearized. The advantage of this scheme is time accuracy while the disadvantage is substantial computational work.

In most instances we are interested in steady state computations. In that case we can take advantage of simplifications such as the diagonal algorithm as long as the scheme converges and we do not distort the steady state equations. The diagonal algorithm used in ARC2D is

$$T_\xi \left[I + h\,\delta_\xi\,\Lambda_\xi\right]\,\widehat{N}\,\left[I + h\;\delta_\eta\,\Lambda_\eta\right]\,T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \tag{8.2}$$

In this case we always employ the nonlinear dissipation models, Eq. (6.12) with a linearization of the terms which necessitates the use of scalar pentadiagonal solvers. (Note that the implicit artificial dissipation terms are placed inside the bracketed terms and are operated on by the similarity transformations). This form of the diagonal scheme gives us a very efficient code in terms of computational work and enhanced stability and convergence due to the proper linearization of the explicit steady state equations. The time accuracy though is at most first order. We also employ the variable time step Eq. (7.8) and mesh sequencing to accelerate convergence.

We shall employ ARC2D and ARC3D to demonstrate various aspects of algorithm improvements, accuracy, and application in the remainder of these notes.

IX. Boundary Conditions

The implementation of a sophisticated numerical algorithm and the development of a flow code usually are trivial tasks when compared with the work of actually solving a particular fluid dynamics problem. We can always assess the applicability of a numerical algorithm based on stability and accuracy considerations. The writing of specialized input and output routines, while not unimportant, is usually mechanical. The real stumbling block comes with the selection, implementation and assessment of boundary conditions (BC).

There is a hierarchy of decisions which are made when the boundary condition problem is attacked. The important aspects of boundary condition development are:

1. The physical definition of the flow problem must be satisfied. For example, inviscid flow requires tangency at solid surfaces, or we may want to specify pressure at some boundary.
2. The physical conditions must be posed in terms of the mathematics of the problem. Characteristic theory suggests the number of conditions required at a boundary. The condition of no slip for viscous flow is imposed by setting the flow velocities to zero at solid surfaces.
3. The mathematical conditions are numerically approximated.
4. The numerical interior scheme may require more boundary information than the physics provides. For example, standard central differencing as an interior scheme requires all flow quantities at boundaries, but this may not be consistent with mathematical theory. Additional numerical boundary conditions may be adjoined.
5. The combination of interior numerical scheme and boundary scheme should be checked for stability and accuracy.
6. Finally, we must assess the efficiency and generality of a flow code in terms of its ability to handle a wide variety of problems and flow conditions.

The physical definition of the flow problem is the first and foremost consideration. Once a geometry and topology have been chosen, then physics dictates the constraints on the boundaries.

9.1 Characteristic Approach

The concept of characteristic theory is best demonstrated with the one-dimensional Euler equations, where

$$\partial_t Q + \partial_x (AQ) = 0 \tag{9.1}$$

represents the model equation. Assuming that A is a constant coefficient matrix we can diagonalize Eq. (9.1) using the eigenvector matrix X, so that

$$\partial_t \left(X^{-1}Q\right) + \Lambda_A \partial_x \left(X^{-1}Q\right) = 0. \tag{9.2}$$

Defining $X^{-1}Q = W$, we now have a diagonal system, with

$$\Lambda_A = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a & 0 \\ 0 & 0 & u-a \end{bmatrix} \tag{9.3}$$

At the left boundary of a closed physical domain, see Fig. 9, where say $0 < u < a$, (for example, subsonic inflow for a channel flow) the two characteristic speeds $u, u+a$ are positive, while $u-a$ is negative. At inflow then, only two pieces of information enter the domain along the two incoming characteristics and one piece leaves along the outgoing characteristic. At the outflow boundary one piece of information enters and two leave. We would like to specify the first two components of W, which are the two incoming characteristic variables and then handle the third characteristic variable such that its value is not constrained, i.e., is determined by the interior flow.

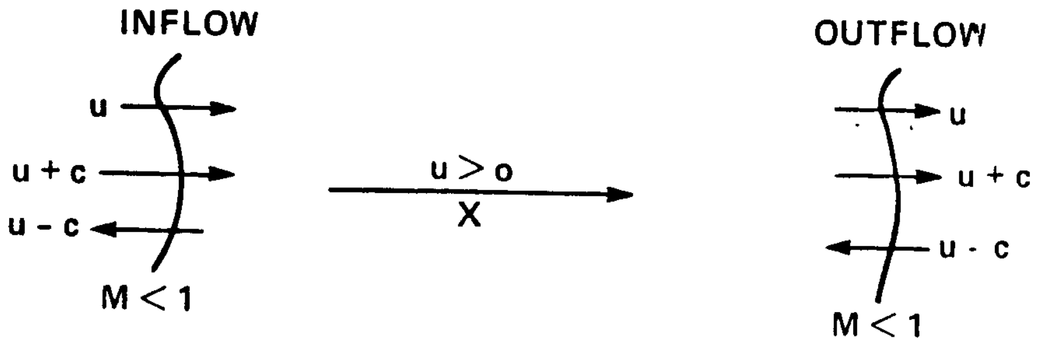


FIGURE 9. Characteristics at Subsonic Inflow and Outflow Boundaries of a Closed Domain.

It is not necessary to fix values only in terms of the characteristic variables, other flow quantities could be employed, as long as they lead to well posed conditions (that is, conditions which guarantee the stability of the mathematical problem). Yee [34] provides an excellent survey and development of boundary conditions within the framework of an implicit algorithm. The major constraints which occur are that the correct number of boundary values corresponding to incoming characteristics are specified and that the actual implementation is stable and well posed. Chakravarthy [35] presents an implicit characteristic boundary procedure. In this the eigenvectors of the system are coupled with the chosen fixed boundary values and one sided finite differences to develop an equation which is solved for the flow variables at the boundaries.

9.2 Well Posedness

A simple check on the well posedness of boundary conditions is obtained as part of Chakravarthy's development. As an example, let us consider one-dimensional flow with subsonic inflow and subsonic outflow. Then two variables can be specified at inflow, associated with the first two eigenvalues, and one variable can be specified at outflow, associated with the third eigenvalue. As specified values take, $\rho = \rho_{in}$, $\rho u = (\rho u)_{in}$ and $p = p_{out}$. These can be written as

$$B_{in}(Q) = \begin{bmatrix} q_1 \\ q_2 \\ 0 \end{bmatrix} = B_{in}(Q_{in}) \tag{9.4a}$$

$$B_{out}(Q) = \begin{bmatrix} 0 \\ 0 \\ (\gamma-1)(q_3 - \frac{1}{2}q_2^2/q_1) \end{bmatrix} = B_{out}(Q_{out}) \tag{9.4b}$$

with $q_1 = \rho$, $q_2 = \rho u$, $q_3 = e$.

Forming the Jacobians $C_{in} = \partial B_{in}/\partial Q$, and $C_{out} = \partial B_{out}/\partial Q$ we have

$$C_{in} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad C_{out} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ [(\gamma-1)/2]\, u^2 & -(\gamma-1)u & \gamma-1 \end{bmatrix} \tag{9.5}$$

The eigenvector matrix X^{-1} is

$$\begin{bmatrix} 1 - \frac{u^2}{2}(\gamma-1)a^{-2} & (\gamma-1)ua^{-2} & -(\gamma-1)a^{-2} \\ \beta[(\gamma-1)\frac{u^2}{2} - ua] & \beta[a - (\gamma-1)u] & \beta(\gamma-1) \\ \beta[(\gamma-1)\frac{u^2}{2} + ua] & -\beta[a + (\gamma-1)u] & \beta(\gamma-1) \end{bmatrix} \tag{9.6}$$

with $\beta = 1/(\sqrt{2}\rho a)$.

The condition for well posedness of these example boundary conditions is that $\overline{C}_{in}^{-1}$ and $\overline{C}_{out}^{-1}$ exist where

$$\overline{C}_{in} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta[(\gamma-1)\frac{u^2}{2} + ua] & -\beta[a + (\gamma-1)u] & \beta(\gamma-1) \end{bmatrix} \tag{9.7a}$$

and

$$\overline{C}_{out} = \begin{bmatrix} 1 - \frac{u^2}{2}(\gamma-1)a^{-2} & (\gamma-1)ua^{-2} & -(\gamma-1)a^{-2} \\ \beta[(\gamma-1)\frac{u^2}{2} - ua] & \beta[a - (\gamma-1)u] & \beta(\gamma-1) \\ (\gamma-1)\frac{u^2}{2} & -(\gamma-1)u & \gamma-1 \end{bmatrix} \tag{9.7b}$$

These matrices are formed by adjoining the eigenvector associated with the outgoing characteristic to the Jacobian matrices of the boundary conditions. The inverses of the above matrices will exist if $det(\overline{C})$ is nonzero. For the two boundaries we have $det(\overline{C}_{in}) = \beta(\gamma - 1) \neq 0$ and $det(\overline{C}_{out}) = \beta(\gamma - 1)a \neq 0$ and so the boundary conditions are well posed. Other choices for specified boundary values can be similarly checked.

Rather than go into any more detail on boundary condition theory we refer the reader to Refs. [34] and [35]. For the remainder of this section, we shall discuss some of the current types of physical conditions which are used in ARC2D.

9.3 Computational Mapping of Boundaries

Usually in a flow field computation, we are faced with a variety of boundary surfaces and conditions. Our experience has been mostly in external flows and so we shall outline below some of the more commonly used boundary conditions. The curvilinear coordinate transformations are made such that physical boundaries are mapped to boundaries in the computational domain. This makes the formulation and implementation of boundary conditions easier.

At this stage in the development of ARC2D we have kept the boundary conditions explicit. This gives us a more flexible clean code since all BC are handled in just one subroutine. We realize that implicit boundary treatment will enhance the stability and convergence rates of the codes. Our experience has been that the basic code in its present form is fairly robust and can be implemented for a wide variety of cases. The user is then responsible for the implementation of boundary conditions.

A particular set of BC for an airfoil calculation is used below for demonstration purposes. The geometry is mapped onto the computational rectangle such that all the boundary surfaces are edges of the rectangle, for example see Fig. 10 .

In “O” mesh topologies the wake cut boundary is periodic and can be handled as such where periodic solvers are used in the implicit inversions.

A. Body Surfaces

At a rigid body surface, tangency must be satisfied for inviscid flow and the no slip condition for viscous flow. In two-dimensions body surfaces are usually mapped to η = constant coordinates. The normal component of velocity in terms of the curvilinear metrics is given by

$$V_n = \frac{\eta_x u + \eta_y v}{\sqrt{(\eta_x^2 + \eta_y^2)}} \tag{9.8a}$$

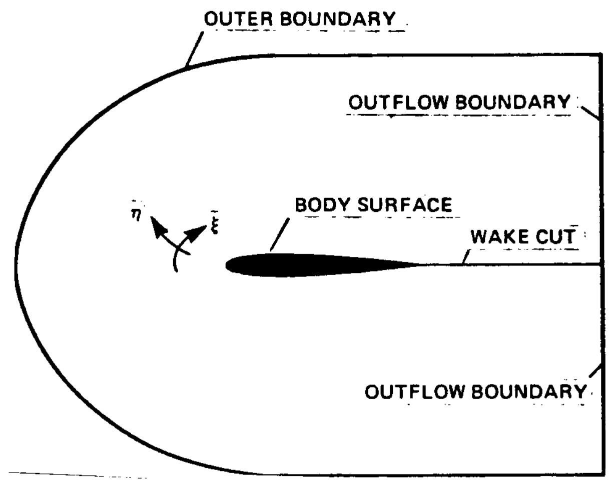


FIGURE 10. Topological Mapping of an Airfoil onto a "C" Mesh.

and the tangential component is

$$V_t = \frac{\eta_y u - \eta_x v}{\sqrt{(\eta_x^2 + \eta_y^2)}} \tag{9.8b}$$

Therefore, tangency is satisfied by $V_n = 0$ (no flow through the body). The tangential velocity V_t is obtained at the body surface through linear extrapolation for inviscid cases and is set to zero for viscous cases. The Cartesian velocities are then formed from the inverse relation between them and Eq. (9.8) where

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1.}{\sqrt{(\eta_x^2 + \eta_y^2)}} \begin{bmatrix} \eta_y & \eta_x \\ -\eta_x & \eta_y \end{bmatrix} \begin{pmatrix} V_t \\ V_n \end{pmatrix} \tag{9.9}$$

The extrapolation of V_t produces less error if the mesh lines are clustered to the body surface. The velocities of Eqs. (9.8a), and (9.8b) are scaled such that the metric variations are removed which decreases the errors in the extrapolations, especially for nonorthogonal meshes.

The pressure on the body surface is obtained from the normal momentum equation

$$\begin{aligned} \rho[\partial_\tau \eta_t + u\partial_\tau \eta_x + v\partial_\tau \eta_y] - \rho U(\eta_x u_\xi + \eta_y v_\xi) = \\ (\eta_x \xi_x + \xi_y \eta_y) p_\xi + ({\eta_x}^2 + {\eta_y}^2) p_\eta = p_n \sqrt{{\eta_x}^2 + {\eta_y}^2} \end{aligned} \tag{9.10}$$

where n is the local normal to the body surface. Equation (9.10) is solved at the surface using central second-order accurate differences in ξ and one-sided first- or second-order accurate differences in η. For steady uniform incoming flow free-stream stagnation enthalpy H_0 is held constant along the body in inviscid flow. Using the equation for enthalpy $H_0 = (e+p)/\rho$ and the computed velocities and pressure, a value of density is obtained at the body. Adiabatic or constant temperature walls are used for viscous and unsteady flows to obtain density at the surface. In either case, total energy e is decoded from the equation of state.

B. Free Surfaces Stretched grids are usually used to place far field boundaries far away from body surfaces. When bow shocks and attached shocks are generated at a body surface care is taken to ensure that the shocks are sufficiently weak when they reach far field boundaries so that they are not reflected or at least they reflect outside the flow domain. A nonreflective characteristic like boundary procedure is used at far field boundaries.

For subsonic free stream locally one-dimensional Riemann invariants are used at the outer far field boundaries. The locally one-dimensional Riemann invariants are given in terms of the normal velocity component as

$$R_1 = V_n - 2a/(\gamma-1) \quad \text{and} \quad R_2 = V_n + 2a/(\gamma-1) \tag{9.11}$$

The Riemann invariants R_1, R_2 are associated with the two characteristic velocities (locally one-dimensional) $\lambda_1 = V_n - a$ and $\lambda_2 = V_n + a$ respectively. Two other equations are needed so that four unknowns (the four flow variables) can be calculated. We choose V_t and $S = \ln(p/\rho^\gamma)$ where S is entropy. At the far field boundaries shown in Fig. 10, the normal n is directed away from the boundary. For subsonic inflow $V_n < 0$ and the characteristic velocity $\lambda_2 > 0$, therefore the characteristic variable R_2 can be specified along with two other conditions. The Riemann invariant R_2, V_t and S are all set to free stream values. The other characteristic velocity $\lambda_1 < 0$ and R_1 is extrapolated from the interior flow variables. On subsonic outflow $V_n > 0$ and $\lambda_2 > 0$ while $\lambda_1 < 0$ so only R_1 is fixed to free stream and R_2, V_t and $\ln(S)$ are extrapolated. Once these four variables are available at the boundary the four flow variables Q can be obtained. For supersonic inflow boundaries all flow variables are specified and for supersonic outflow all variables are extrapolated.

Along singularities or cuts in the geometry (such as the wake cut in a "C" mesh), averaging is used to provide continuous flow variables. As mentioned above periodic conditions are used for "O" meshes.

C. Far Field Circulation Correction

For lifting airfoils in subsonic free stream, circulation at the far field boundary is accounted for to first-order (following Salas, et. al. [36]) by imposing a compressible potential vortex solution which is added as a perturbation to the free stream quantities ($u_\infty = M_\infty \cos(\alpha)$ and $v_\infty = M_\infty \sin(\alpha)$). The perturbed far field boundary velocities are defined as

$$u_f = u_\infty + \frac{\beta ,\ \sin(\theta)}{2\pi r \left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \tag{9.12a}$$

and

$$v_f = v_\infty - \frac{\beta ,\ \cos(\theta)}{2\pi r \left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \tag{9.12b}$$

where the circulation , $= \frac{1}{2} M_\infty l C_l$, l is the chord length, C_l the coefficient of lift at the surface, M_∞ the free stream Mach number, α the angle of attack, $\beta = \sqrt{1 - M_\infty^2}$ and r, θ are polar coordinates to the point of application on the outer boundary relative to an origin at the quarter chord point on the airfoil center line. A corrected speed of sound is also used which enforces constant free stream enthalpy at the boundary where

$$a_f^2 = (\gamma - 1)\left(H_\infty - \frac{1}{2}(u_f^2 + v_f^2)\right) \tag{9.12c}$$

Equations (9.12) are used instead of free stream values in defining the fixed quantities for the far field characteristic boundary conditions to be consistent with the surface lift.

Figure 11 shows the coefficient of lift C_l plotted against the inverse of the distance to the outer boundary for a NACA 0012 airfoil at the transonic condition $M_\infty = 0.8, \quad \alpha = 1.25^\circ$ and at subcritical conditions $M_\infty = 0.63, \quad \alpha = 2.0^\circ$. In these cases the outer boundary varies for 4.5 chords to 96 chords where outer mesh rings were eliminated from the 96 chord grid to produce the cut down meshes. This insures that the grid spacing between the body and outer boundary is identical for all the cases. Without the far field vortex correction the lift of the subcritical case can vary by as much as 12 % as seen in Fig. 11. With the far field vortex logic the subcritical case now has virtually no variation with outer boundary distance. For the transonic case we see roughly a 1 - 2 % change which is quite good considering the strength of the shocks. The typical distance chosen for most cases presented here is 25 chords.

The vortex correction logic can be modified to produce boundary conditions which allow one to compute the angle of attack for a given lift. This is done by fixing the circulation , in Eq. (9.12) at its value for the given lift. An iterative procedure is used where the lift computed at the surface is compared to the desired lift and then the initial angle of attack is modified by the formula

$$\Delta\alpha = -\beta_\alpha \left(C_l(input) - C_l(calculated)\right)$$

with β_α a relaxation parameter on the order of 2 . Computations in which a specified lift resulted in an angle of attack were compared with fixed α solutions at the same

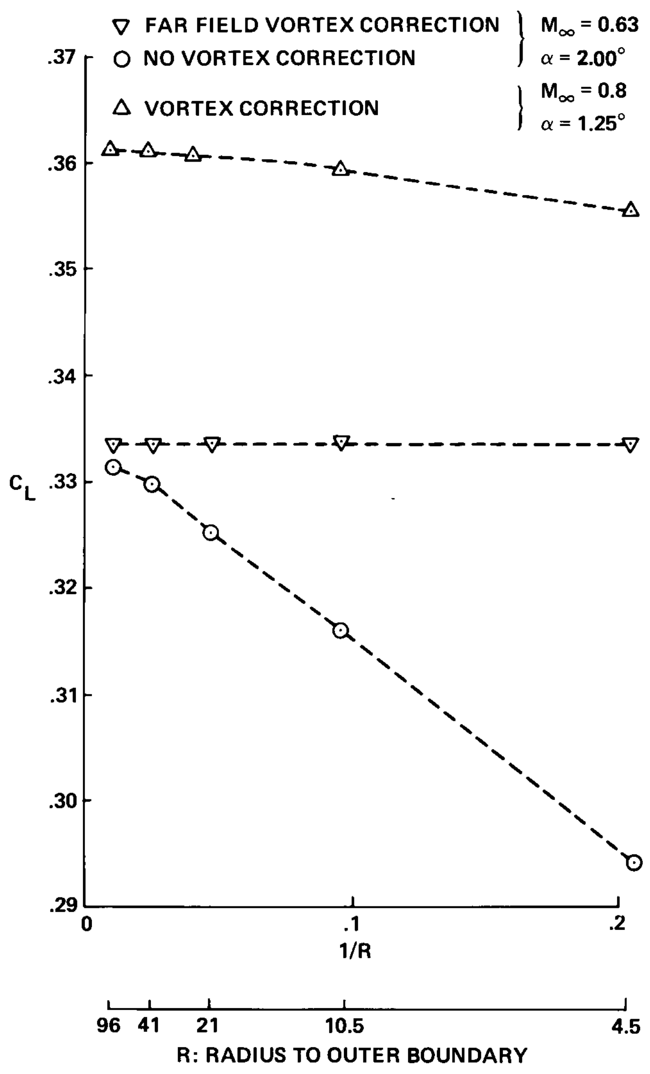


FIGURE 11. Effect on Lift of Varying Outer Boundary Distances With and Without Vortex Correction.

Mach number and showed excellent agreement. This procedure has been verified in numerious numerical examples.

X. Geometry and Grid Generation

The generalized coordinate transformation produces a system of equations which can be applied to any regular and nonsingular geometry or grid system. The advantages of this form are : Since uniform unit spacing is used, the computational domain has a one to one correspondence with the positive integers and therefore regular unweighted difference formulas can be used in the numerical scheme. This produces a computer code which can be applied to a wide variety of problems without modification of the equations and numerical scheme. Physical boundary surfaces can be mapped onto coordinate surfaces, which makes application of boundary conditions easier. The transformation allows for unsteady motion of the coordinates, so that moving meshes and distorting surfaces can be computed. Grid lines can be concentrated in regions of high gradients, for instance clustered to body surfaces to calculate boundary layers, or clustered near shocks.

There are a wide variety of methods for generating grid systems. Algebraic methods such as conformal mappings, quadratic functions, or the control function approach of Eiseman [37] have been widely employed. The numerical approach of using elliptic solvers, Thompson, Thames and Mastin [38], is also widely used. Thompson [39] provides a good review of the current state of the art in grid generation. Figures 12 and 13 show a "C" mesh and an "O" mesh topology for an airfoil where the mesh has been generated using a variant of Eiseman's method [37].The terminology "C" comes from the wrap around nature of the grid. In the case shown grid lines are clustered at the leading and trailing edge, near the body in the near normal direction and on the upper surface to capture an expected shock.

One of the major deficiencies in computational fluid dynamics today lies in the area of surface definition and grid generation. While there are a wide variety of generation methods, there is no efficient and accurate means to assess the usefulness of a particular grid. The obvious first checks such as not having grid lines cross, no discontinuous changes in grid spacing and other cosmetic qualities can be checked. But, aspects such as high skewness, curvature smoothness, and other intrinsic properties are hard to assess. What is needed at this point is a set of well defined qualitative and quantitative checks for grid systems which will allow us to distinguish between a "bad" grid and a "good" grid. A systematic study of this form is lacking and hopefully will be pursued in the near future.

XI. Examples and Application in 2-D

11.1 Code Validation

Once a computational code has been written and debugged, the author is faced with the difficult challenge of assessing the accuracy, efficiency and robustness of the piece of work. Each code is obviously tailored toward a class of problems and at least initially one should restrict attention to that class. A series of test cases should be evaluated and then detailed goals should be attacked.

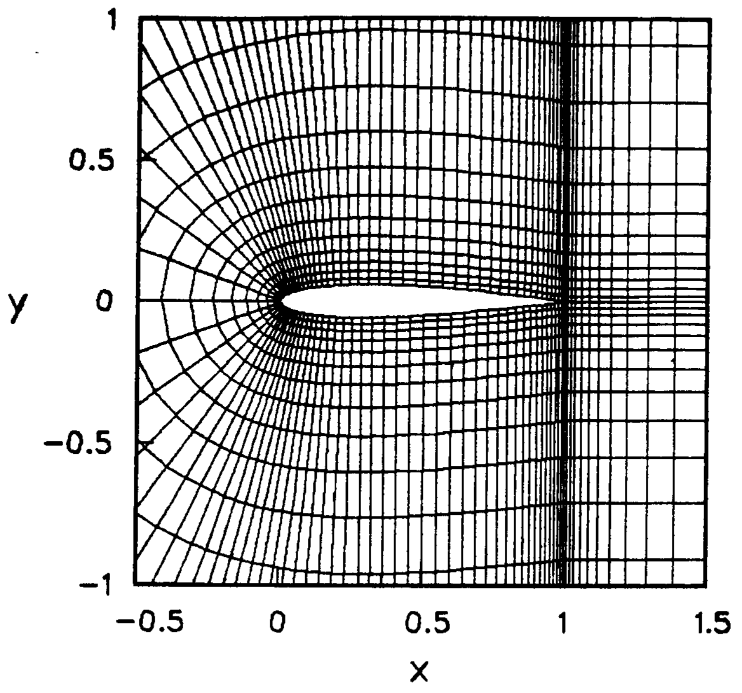


FIGURE 12. "C" Mesh for a NACA0012 Airfoil.

In the case of ARC2D and ARC3D a number of sample cases have been computed and details of these can be found in the literature, see for instance [2], [3], [40-47], and numerious others. Here I shall briefly discuss some of the more exotic cases with the goal of demonstrating the versatility and breath of applications. I refer the reader to the original papers (where appropriate) for extensive details.

11.2 Inviscid Airfoils

The code ARC2D has the option of computing the inviscid equations (the Euler equations). The basic version of the code is written specifically for airfoil computations. The particular set of boundary conditions used now though are directed toward the solution of flow past general airfoil shapes. The code has been applied to a wide variety of airfoil shapes, flow conditions, and other geometries. We have validated the code against other computational methods, [2], [47]. To demonstrate the accuracy and efficiency we have chosen two test cases, a NACA0012 airfoil at $M_\infty = 0.8$, $\alpha = 1.25°$ on a coarse grid (192 by 33 points) and a fine grid (248 by 49 points). For comparison purposes we use results from Jameson's multigrid Euler code FLO52R [48]. FLO52R is an Euler code using a multistage Runge-Kutta like algorithm with a multigrid scheme to accelerate convergence.

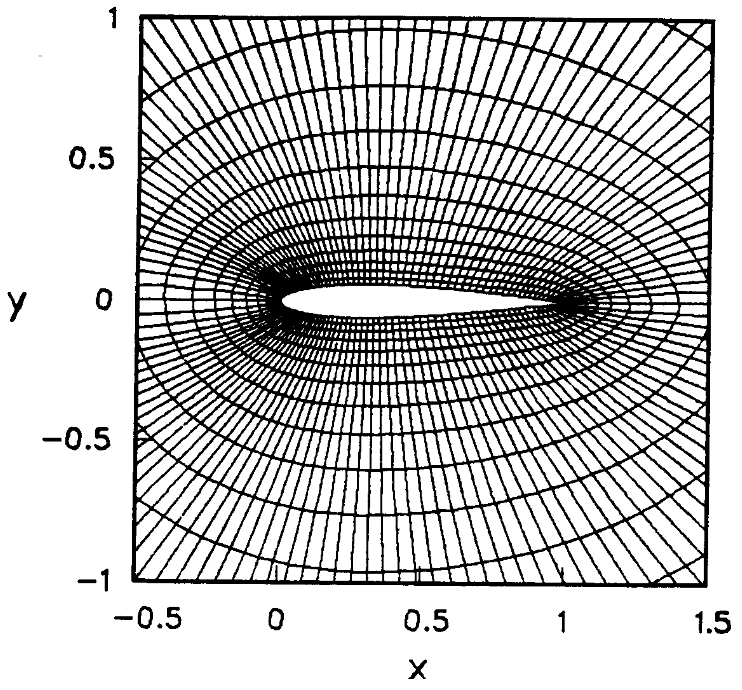


FIGURE 13. "O" Mesh for a NACA0012 Airfoil.

The code employs enthalpy damping, residual averaging and an artificial dissipation model of the same form as presented in Section 6.3. In fact the boundary conditions and artificial dissipation model used in ARC2D were modified to be the same as in FLO52R so that quantitative as well as qualitative comparisons could be made. The two codes were run on the same machine, the CRAY XMP at NASA Ames, on the same meshes and at the same flow conditions.

The first case is the NACA0012 airfoil at $M_\infty = 0.8$ and $\alpha = 1.25°$. The grid used is an "O" mesh topology with 192 points on the airfoil surface (running from the lower trailing edge around the nose to the upper trailing edge) and 33 point in the normal direction. The grid which is clustered at the leading and trailing edges, near the expected shock locations on the upper and lower surfaces and in the normal direction is shown in Fig. 14

Results from this case using ARC2D are shown in Fig. 15. We show here coefficient of pressure, Mach contours, pressure contours and contours of entropy. In Fig. 16 we show similar results for FLO52R. Computed lift for ARC2D is $C_L = 0.33957$ and for FLO52R $C_L = 0.32408$. The comparison between the two codes is quite good, despite the differences in spatial discretization.

We have established a number of accuracy checks and convergence criteria for comparison purposes. In terms of accuracy we recommend comparison of pressure

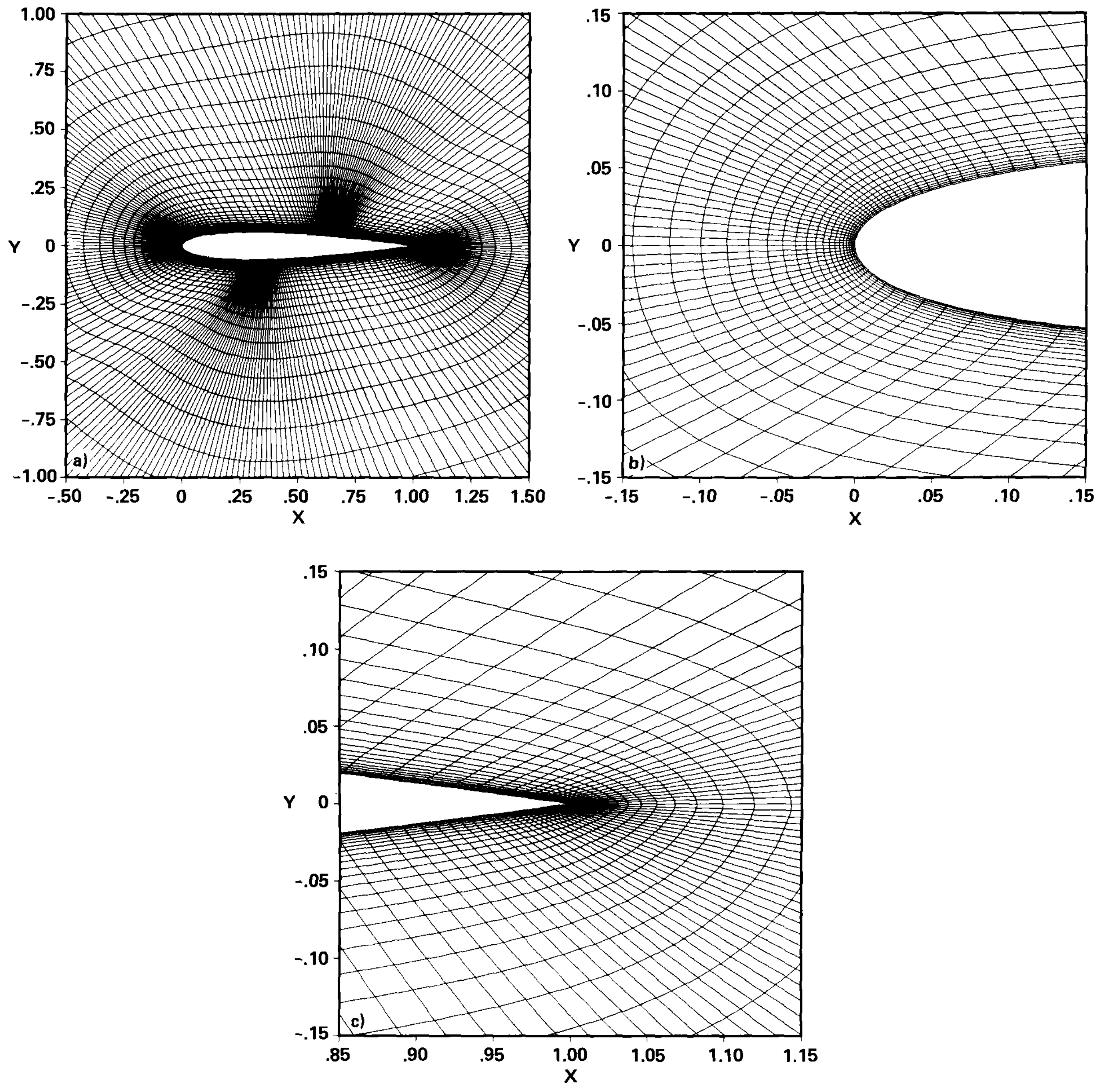


FIGURE 14. NACA0012 Grid Using 192 by 33 Grid Points.

coefficients, lift and other flow quantities. It is also important to establish the accuracy of certain flow regions. The stagnation region near the nose of the airfoil is particularly susceptible to errors due to poor boundary conditions, resolution, or physical assumptions. The best measure of this error is the entropy field. For

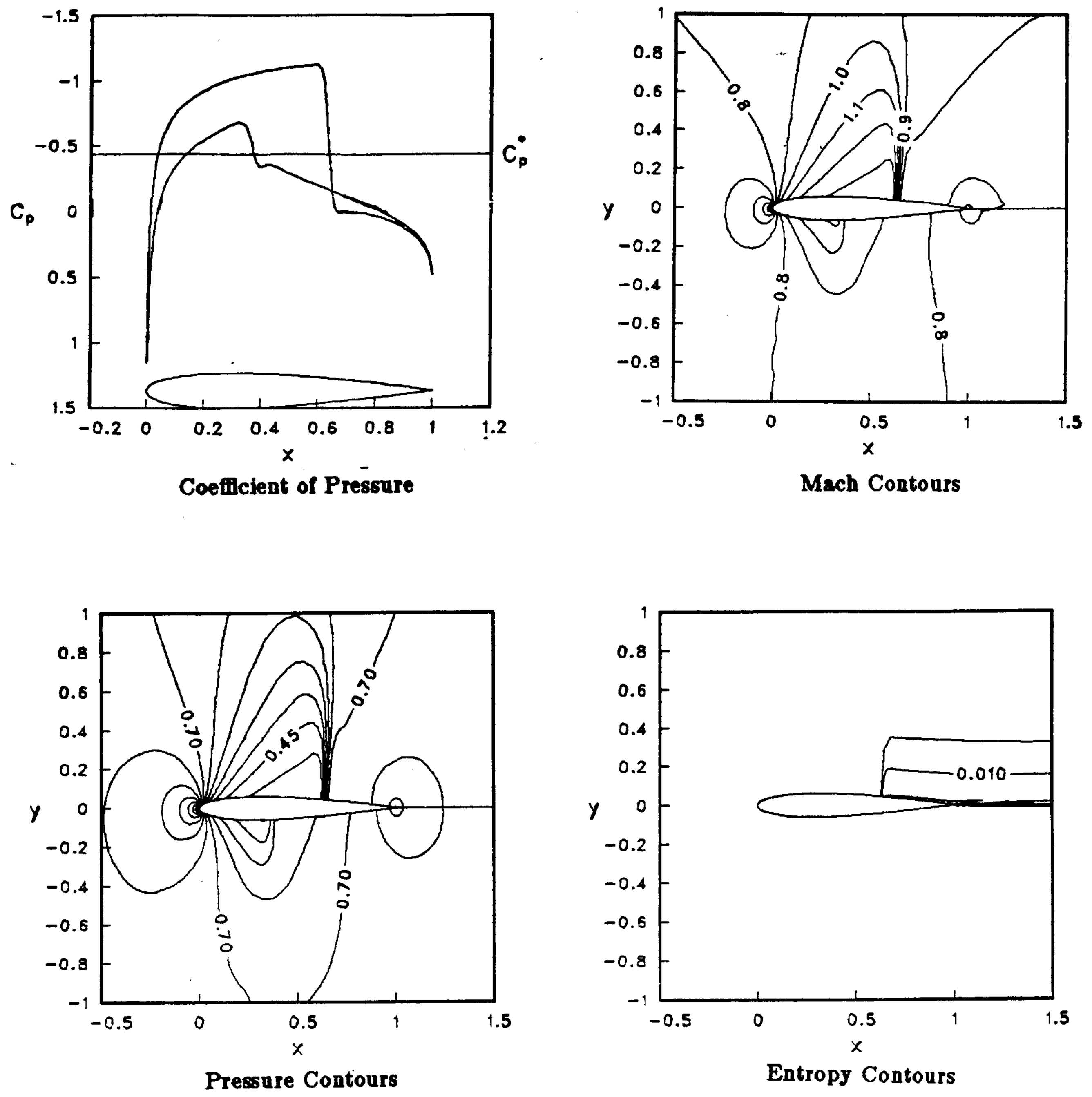


FIGURE 15. ARC2D Results for 192 by 33 Grid.

inviscid flow there should be no generation of entropy at the leading edge of an airfoil in the absence of a leading edge shock. Examination of the entropy at the leading edge for the above case shows, see Fig. 17, that both codes give rise to some error at the leading edge, although the magnitude is rather small.

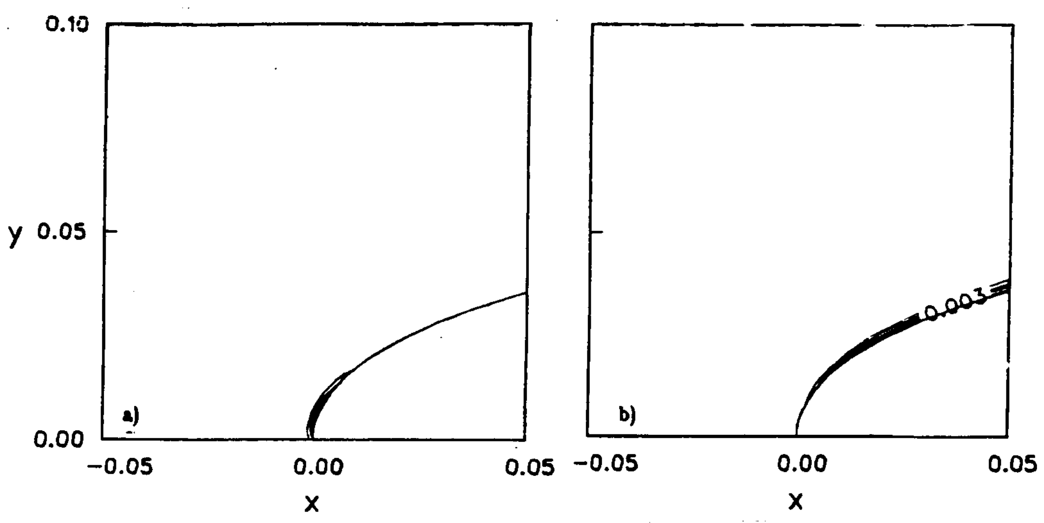


FIGURE 17. Entropy Errors at Leading Edge. a) ARC2D, b) FLO52R.

A number of convergence criteria have been chosen to assess the efficiency and convergence rates of the codes. We have chosen to use computer times as our measure of relative speed. Since the two codes are run on the same machines and with the same meshes this is an adequate measure. Other measures such as operation count, work or iteration are usually programming dependent or susceptible to misinterpretation. The convergence criteria used here are:

1. Coefficient of lift (C_L) to 1% of converged value.
2. Coefficient of lift (C_L) to 1/2% of converged value.
3. Coefficient of lift (C_L) to 5 decimal places.
4. Number of supersonic points to converged value.
5. Residual to machine zero. (10^{-13} on the Cray XMP.)

The residual is the l_2 norm of the explicit or right hand side of Eq.(8.1). We use just the component from the continuity equation, the other components behave

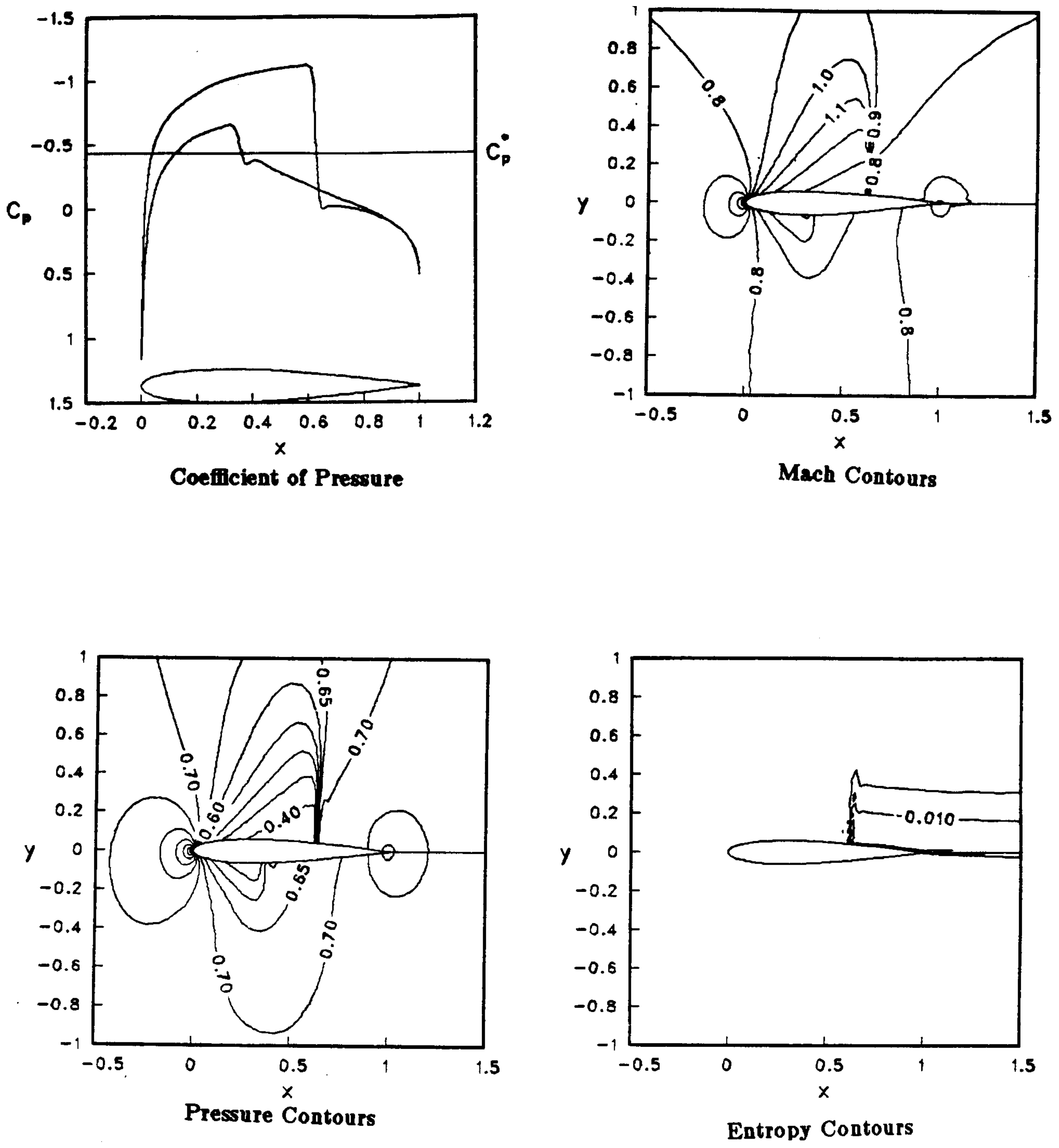


FIGURE 16. FLO52R Results for 192 by 33 Grid.

Convergence Comparison (seconds)		
Criteria	ARC2D	FLO52R
1% of C_L	6	8
1/2% of C_L	17	10.5
C_L to 5 places	57	31
No. S.S. pts	36	17
Machine zero	120	97

Table 1. Convergence Data for 192 by 33 grid.

similarly. For the above case on the 192 by 33 mesh the computer times for the convergence criteria are given in Table 1.

As can be seen for this case FLO52R is up to twice as fast as ARC2D for some criteria. In either event these are fairly good convergence times. In general, these numbers carry over fairly consistently for a wide variety of airfoils and flow conditions for similar meshes.

A more stringent test is obtained with a finer grid and more grid points. A grid of 248 by 49 points is employed as the second study. The mesh is refined more at the nose, tail and near the shocks. Also to reduce the entropy errors at the nose the grid is clustered more tightly in the normal direction by reducing the minimum normal spacing by a factor of 2. The mesh is shown in Fig. 18.

Computational results for ARC2D and FLO52R are shown in Figs. 19 and 20. In this case the shocks are sharper and entropy errors at the leading edge are eliminated.

Convergence data for this case is contained in Table 2. In Figure 21, we show convergence history vrs iteration for the two ARC2D results. All the results obtained with ARC2D were done using the fully implicit pentadiagonal algorithm. As mentioned above, numerious other cases and airfoils have been computed and perform similarly.

Convergence Comparison (seconds)		
Criteria	ARC2D	FLO52R
1% of C_L	38	23
1/2% of C_L	52.5	25.5
C_L to 5 places	174	168.5
No. S.S. pts	118	160
Machine zero	376	800+

Table 2. Convergence Data for 248 by 49 grid.

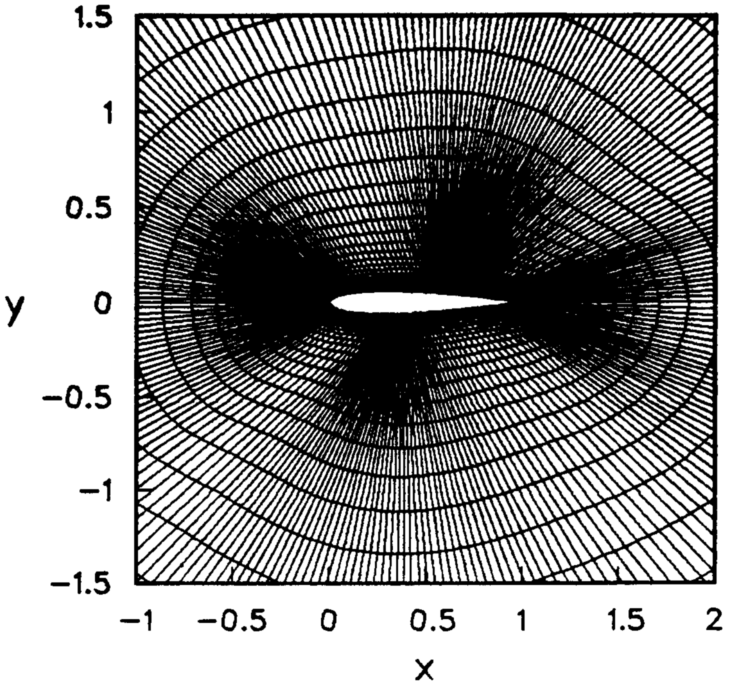


FIGURE 18. NACA0012 Mesh, 248 by 49.

11.3 Viscous Airfoils

The code ARC2D has been applied to a wide variety of viscous computations for airfoils [2], cascades [41], inlets [42], airfoils with a spoiler [43], circulation controlled airfoils [44], and others. It has been used in an unsteady mode (see the next section) and for steady viscous computations. The algorithm as presented above performs very well for viscous cases. It is convergent, fast and accurate. Two example cases are presented below. The cases are taken from the suggested problems of the 1981 Stanford Olympics [49], an RAE2822 airfoil at $M_\infty = 0.676$, $\alpha = 1.93°$, $Re = 5.7 \cdot 10^6$ and $M_\infty = 0.73$, $\alpha = 2.79$ and $Re = 6.5 \cdot 10^6$.

Results obtained from ARC2D for the first case are shown in Fig. 22. The grid used is a 248 by 51 point "O" mesh. The turbulence model was used and transition was fixed at 11% chord. Experimental data due to Cook et. al. [50] is used for comparison. We see a good comparison with experiment for pressure coefficient, and boundary layer properties.

The computed lift, drag and moment are compared with other computations and the experiment in Table 3. Due to the uncertainty of the angle of attack correction all computors matched lift. We show here our computation for both the experimentally corrected angle of attack and the values when lift is matched. Also shown are results from computors at the Stanford Olympics and some results of Mehta [51]. For the present computations at the two angles of attack the pressure

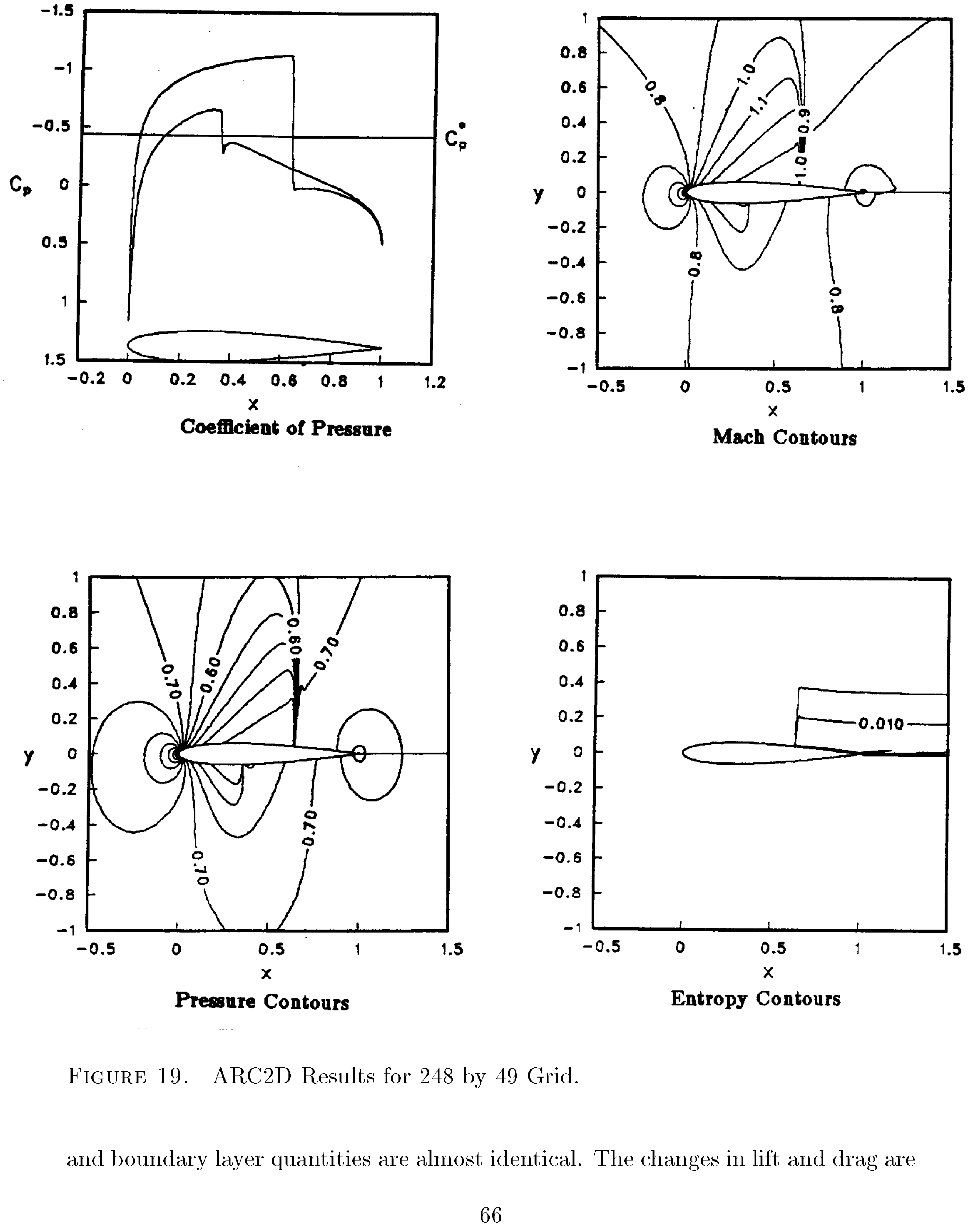


FIGURE 19. ARC2D Results for 248 by 49 Grid.

and boundary layer quantities are almost identical. The changes in lift and drag are

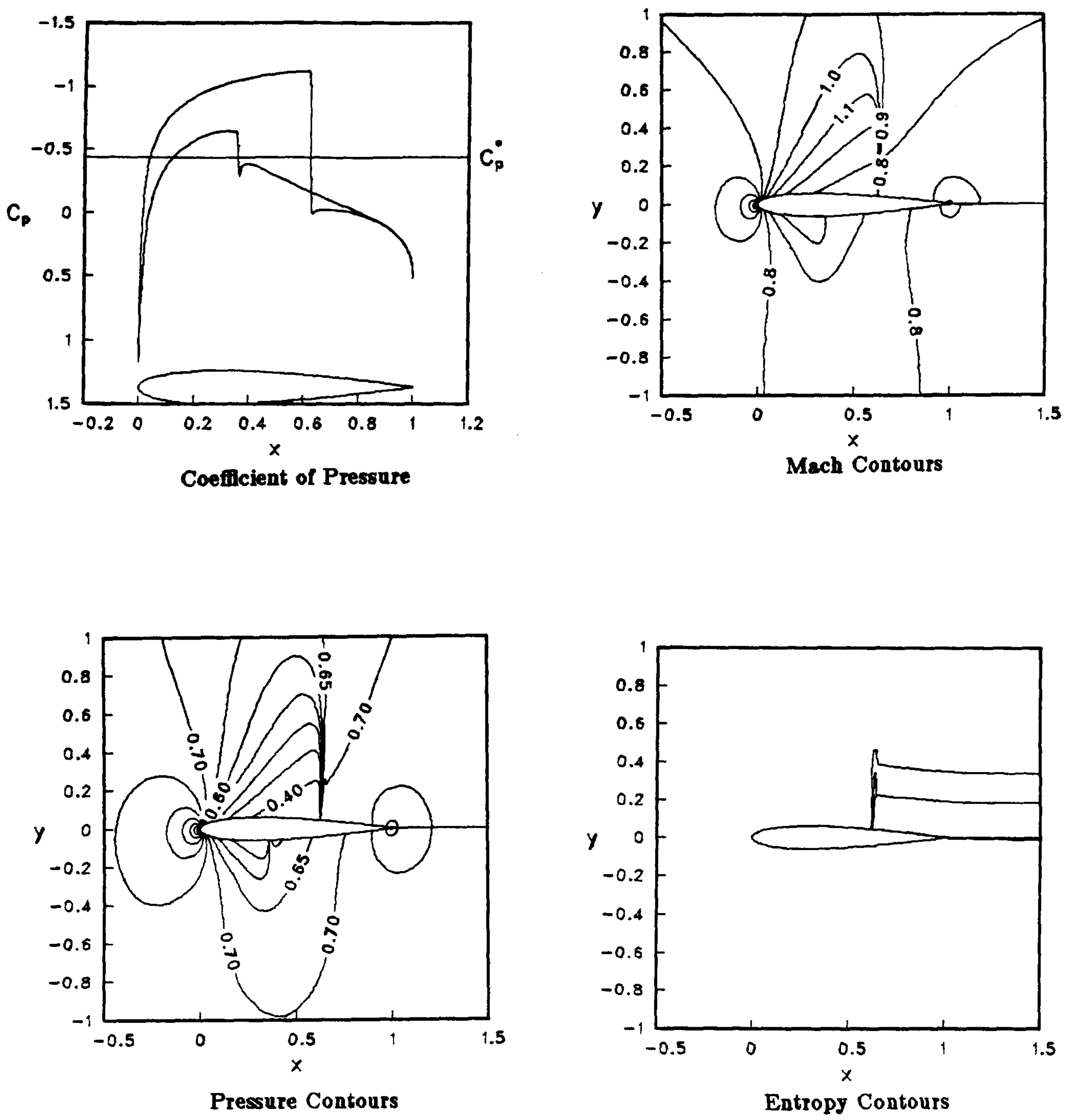


FIGURE 20. FLO52R Results for 248 by 49 Grid.

noted. The overall comparison with experiment and other computations is quite

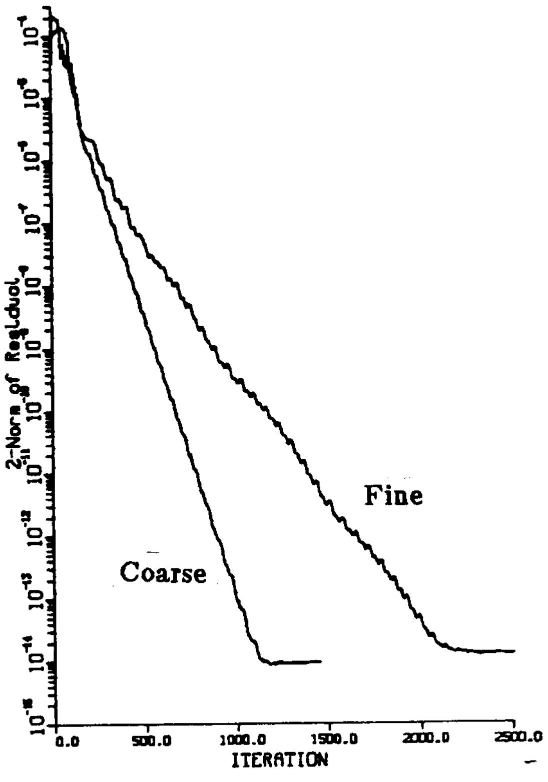


FIGURE 21. Convergence History vrs Iteration for ARC2D Results.

good.

Results obtained for the second case are shown in Fig. 23. The grid used is a 248 by 51 point "O" mesh. The turbulence model was used and transition was fixed at 3% chord. We again see a good comparison with experiment for pressure coefficient, and boundary layer properties.

The computed lift, drag and moment are compared with other computations and the experiment in Table 4. Results from computors at the Stanford Olympics and some results of Mehta [51] are shown. The overall comparison with experiment and other computations is again quite good. The shock location on the upper surface compares well. In the present computations a small region of separated flow occurs at the base of the shock and near the trailing edge on the upper surface.

Convergence history vrs iteration for these cases are shown in Fig. 24. Table 5 shows the computed convergence criteria for these cases. The convergence for these cases is quite good.

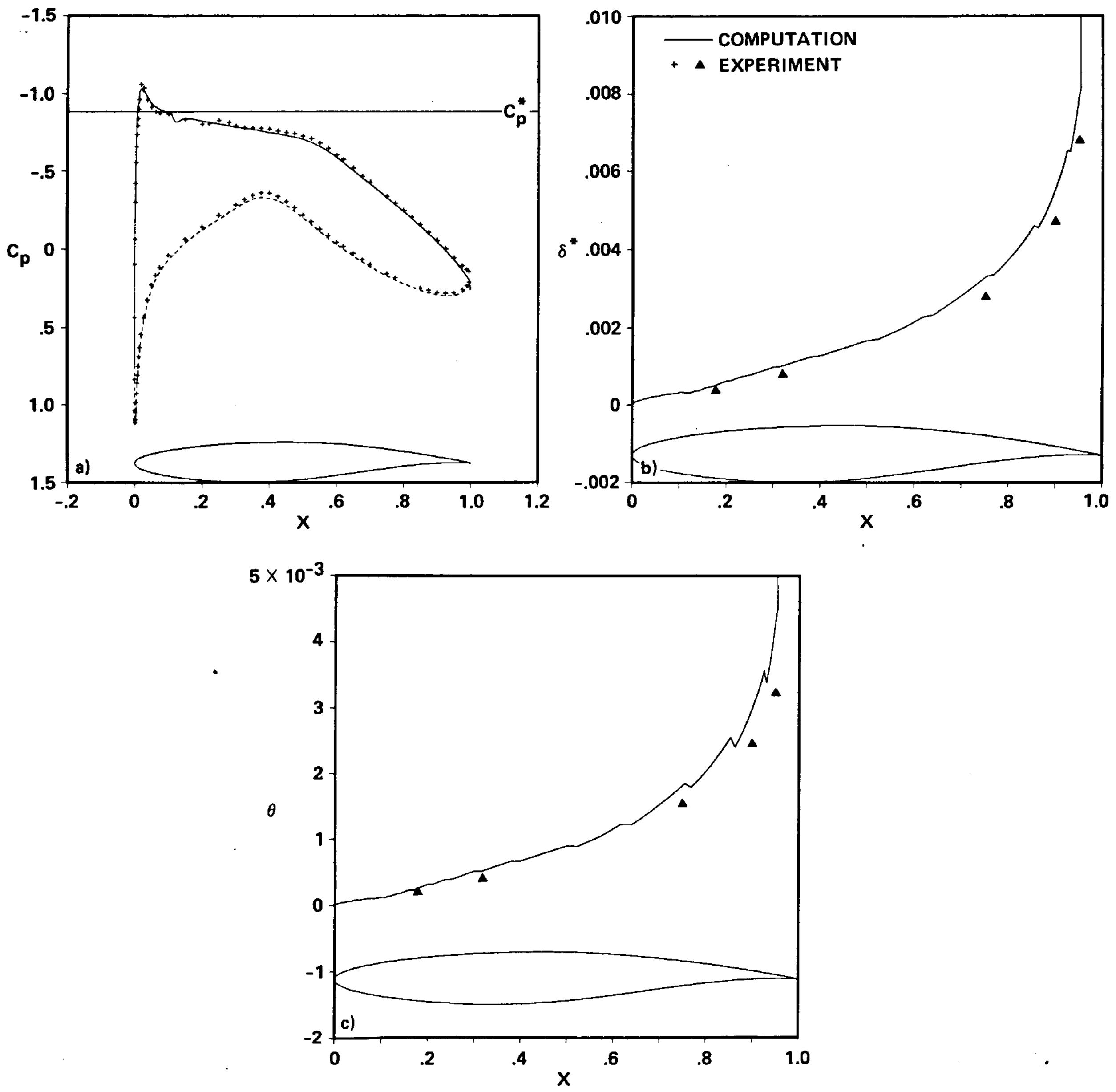


FIGURE 22. Viscous Results for RAE2822 Airfoil at $Re = 5.7 \times 10^6$, $M_\infty = 0.676$, $\alpha = 1.93°$.

11.4 Unsteady Aileron Buzz

A calculation of unsteady aileron buzz was performed by Steger and Bailey [40]. A composite of the results from their paper is shown in Fig. 25.

Loads – RAE2822 Airfoil – $M_\infty = 0.676, Re = 5.7 \times 10^6$						
	α	C_L	C_{DP}	C_{Df}	C_D	C_M
Experiment	2.40	0.566			0.0085	–0.082
Corrected Exp.	1.93	0.566			0.0085	–0.082
Mehta (1983)	1.80	0.566	0.0033	0.0061	0.0094	–0.087
Melnik (1981)	1.84	0.566	0.0027	0.0060	0.0087	–0.082
Le Balleur (1981)	1.93	0.566	0.0036	0.0056	0.0092	–0.080
Present	1.93	0.576	0.0034	0.0055	0.0089	–0.081
Present Cor. α	1.87	0.566	0.0034	0.0055	0.0089	–0.081

Table 3. Forces for RAE2822 Viscous Calculation.

In this case a two dimensional simulation of the interaction of a shock and the movable aileron (flap) on a wing was performed. Steger and Bailey used a predecessor of ARC2D coupled with a one degree of freedom equation describing the motion of the aileron on an airfoil. A airfoil with a hinge point at 75% chord was used. The aileron was free moving without damping and responded to the aerodynamic forces in balance with the inertia forces.

The flow conditions were in the transonic range of Mach number from $M_\infty =$ 0.76 to 0.83 and angles of attack of $\alpha = -1.0°$ to $1.0°$. Experiments performed by Erikson and Stephenson [52] on a P-80 wing-aileron arrangement were used for comparison. A buzz boundary was established in the experiments (in terms of Mach number and angle of attack) where below the boundary the aileron remained stationary. Above the buzz boundary the shock system on the airfoil moves onto the aileron and excites the buzz motion of the aileron. An unsteady harmonic motion occurs with the upper and lower shocks running across the hinge onto and off of the aileron.

Steger and Bailey simulated this flow using the thin layer Navier Stokes equations for the conditions shown by the symbols in Fig. 25b. Figure 25c shows a case below the buzz boundary. In this case they gave the aileron an initial deflection of $4°$ and integrated forward in time. As seen the aileron motion damps to the neutral position of $0°$ deflection. Above the buzz boundary even an aileron deflection of $0°$ is excited to the unsteady motion. In Fig. 25d the results are compared with the measure deflection angles. In Fig. 25c the computed buzz boundary compares quite well with the measured boundary.

11.5 High Angle of Attack Airfoils

Application of the code ARC2D to the study of airfoils at high angles of attack was carried out by Barton and Pulliam [53]. In this study NACA0012 airfoil flows at

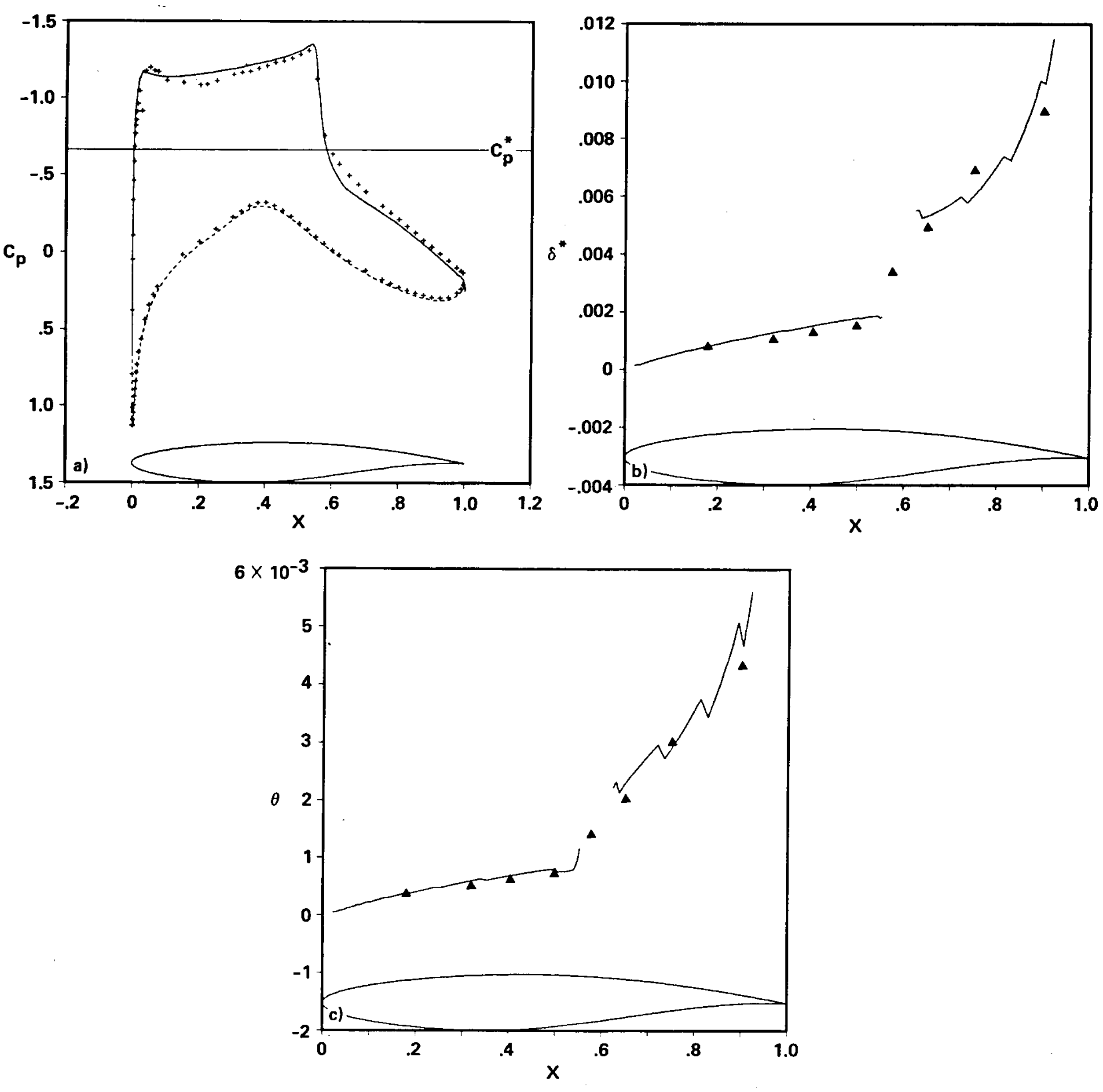


FIGURE 23. Viscous Results for RAE2822 Airfoil at $Re = 6.5 \times 10^6$, $M_\infty = 0.73$, $\alpha = 2.79°$.

low Mach number, $M_\infty = 0.25$ to 0.40 and angles of attack up to 15° were examined. Computations were performed for the Euler equations and thin layer Navier-Stokes. The calculations presented were run time accurately because unsteady effects were

Loads – RAE2822 Airfoil – $M_\infty = 0.73, Re = 6.5 \times 10^6$						
	α	C_L	C_{DP}	C_{Df}	C_D	C_M
Experiment	3.19	0.803			0.0168	–0.099
Corrected Exp.	2.79	0.803			0.0168	–0.099
Mehta (1983)	2.79	0.793	0.0118	0.0059	0.0177	–0.094
Melnik (1981)	2.54	0.803	0.0100	0.0057	0.0157	–0.094
Le Balleur (1981)	2.79	0.787	0.0111	0.0055	0.0166	–0.086
Present	2.79	0.824	0.0128	0.0050	0.0178	–0.092
Present Cor. α	2.67	0.803	0.0113	0.0051	0.0164	–0.092

Table 4. Forces for RAE2822 Viscous Calculation.

anticipated. In that paper unsteady separated but inviscid results were obtained. Comparisons were made with viscous computation and experiment to demonstrate that inviscid flow separation can be qualitatively different than viscous flow. A similar study which concentrated on comparisons with viscous experimental data was carried out by Anderson, Thomas and Rumsey [54] in which good quantitative comparison were obtained. We shall briefly discuss here the computations of Barton and Pulliam.

Barton and Pulliam presented two types of inviscid flow separation. In the first case at flow conditions, $M_\infty = 0.25 \quad \alpha = 15°$ a shock free solution with inviscid flow separation was obtained and the cause of the separation was traced to numerical error. At the high angle of attack, inaccurate boundary conditions and resolution at the leading edge produced vorticity (entropy gradients) which was convected downstream, resulting in an unsteady separation on the upper surface near the trailing edge. By refining the grid and improving the boundary conditions a steady error free solution was obtained. Figure 26 shows the entropy contours at the leading edge before and after the improvement. Figure 27 shows a comparison with full potential results using TAIR [55] and shows that good inviscid results are obtained.

At a higher Mach number $M_\infty = 0.4$ and the same angle of attack $\alpha = 15°$ a shock forms at the leading edge, see Fig. 28. In this case the shock is the source of vorticity which is then convected downstream and forms an unsteady separation. Grid refinement and the improved boundary conditions were used producing an error free leading edge, but the unsteady motion was unaffected. The vorticity (entropy) generation is a result of the strong normal shock strength gradient and high curvature of the leading edge.

The unsteady motion of the solution is depicted in Fig. 29, which shows the time history of the pressure coefficient, stream function contours and entropy fields over a complete cycle. A description of the evolution of this case is as follows. As the flow develops, a strong shock is generated at the leading edge. Entropy,

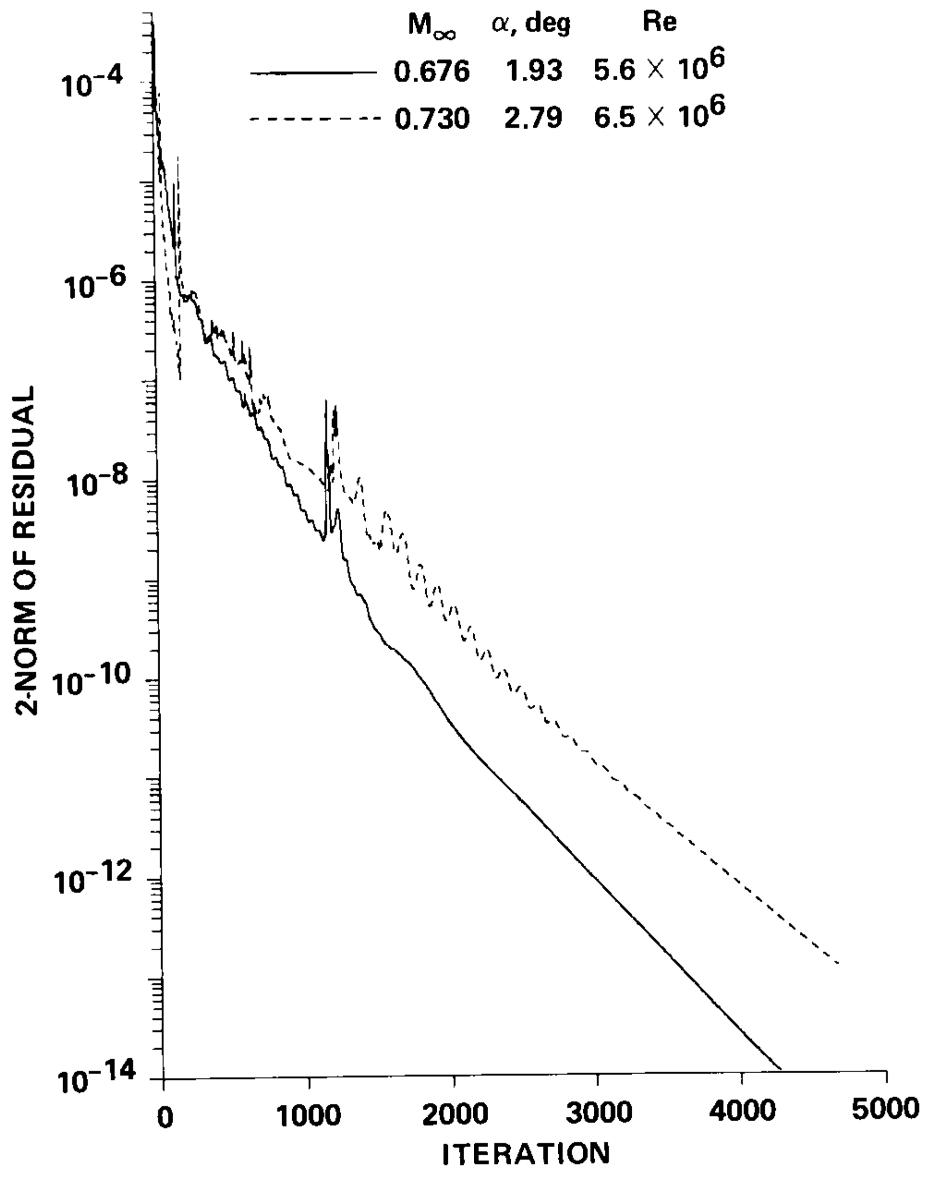


FIGURE 24. Convergence History for RAE2822 Viscous Cases.

vorticity, and pressure loss are created at the shock near the leading edge, and convected downstream along the body. A small separation region appears at the trailing edge, which grows along the body towards the leading edge. At some point the recirculation region is captured by the oncoming flow and is swept off the

Convergence Comparison (seconds)		
Criteria	$M_\infty = 0.676$	$M_\infty = 0.73$
1% of C_L	212	191
1/2% of C_L	264	295
C_L to 5 places	712	738
No. S.S. pts	608	513
Machine zero	1147	1203

Table 5. Convergence Data RAE2822 Viscous Cases.

airfoil by convection. As the recirculation region passes the trailing edge, another pocket of recirculation forms at the trailing edge, rotating in the opposite direction. This counter-rotation is caused by the flow off the lower surface, whose direction is opposite to that of the original region of recirculation. The shock then collapses, and begins to slowly grow in strength as the pattern repeats itself. This flow pattern has a well defined period and amplitude and has been reproduced in other computations with similar grids and different values of time step and artificial viscosity.

As a final case Barton and Pulliam computed a viscous calculation at similar conditions, a Mach number $M_\infty = 0.301$ and $\alpha = 13.5°$. The Reynolds number used was $Re = 3.91 \times 10^6$ and the calculation was performed using the algebraic eddy-viscosity turbulence model. In this case experimental data due to McCroskey [56] was available. For an inviscid simulation unsteady results similar to the above $M_\infty = 0.4$ case were obtained, but for the viscous computation a steady results occurred which compared well with the experimental data. The steady viscous comparison is shown in Fig. 30.

Assuming the validity of the inviscid oscillation for this case, it was concluded conclude that the Euler solution is not a good approximation to the Navier-Stokes solution, under these conditions. I refer the reader to the full paper for more details.

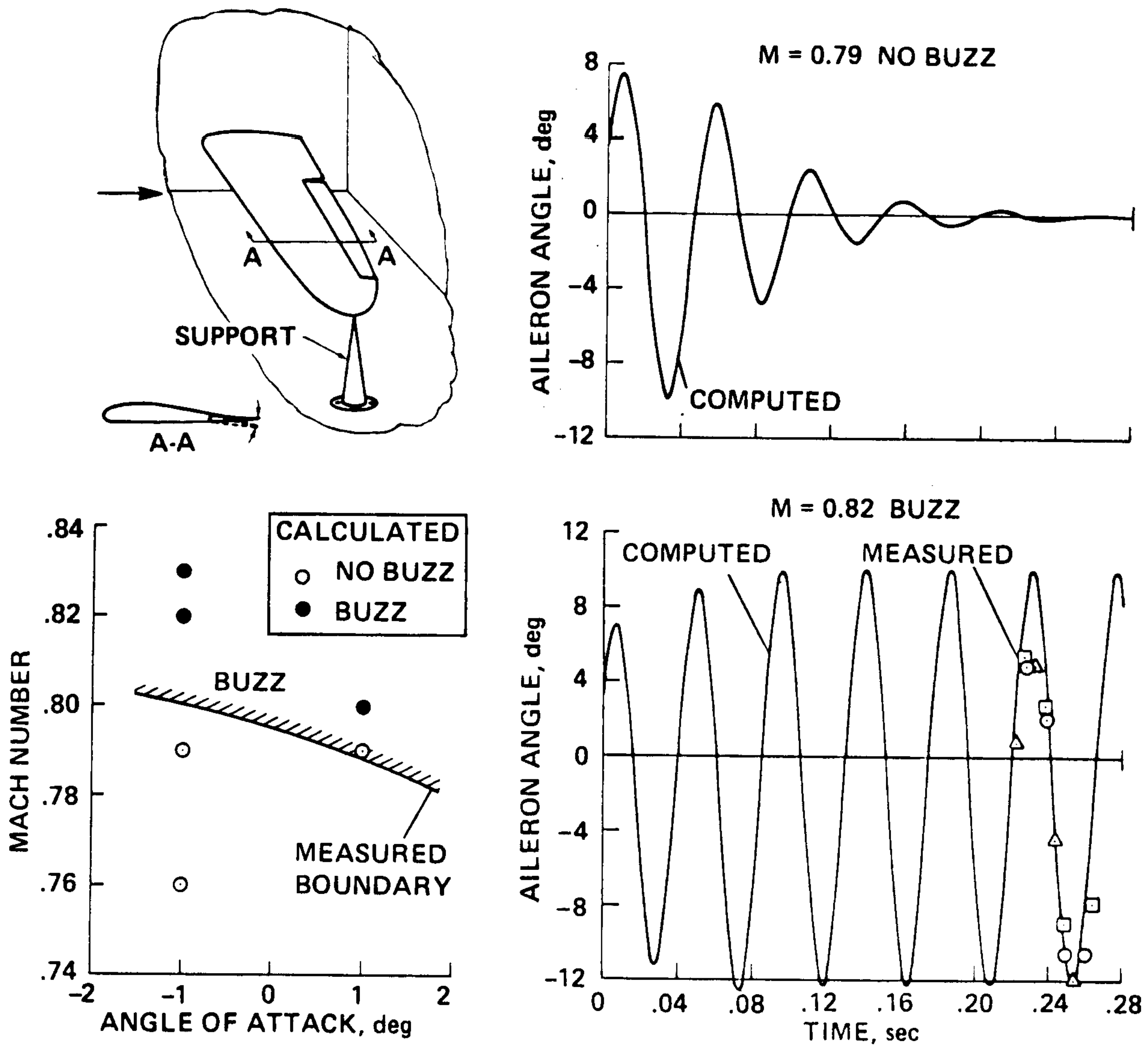


FIGURE 25. Unsteady Aileron Buzz, Steger and Bailey 1980.

XII. Three - Dimensional Algorithm

The 3 - D form of the implicit algorithm follows the same development as the 2 - D algorithm. The curvilinear transformations are carried out in the same fashion. The standard and diagonal algorithm take the same format. We also employ the thin layer approximation. Boundary conditions are similar. The equations, algorithm, and other details can be found in Pulliam and Steger [3]. We shall briefly outline the important aspects and point out the pertinent differences from the 2 - D development.

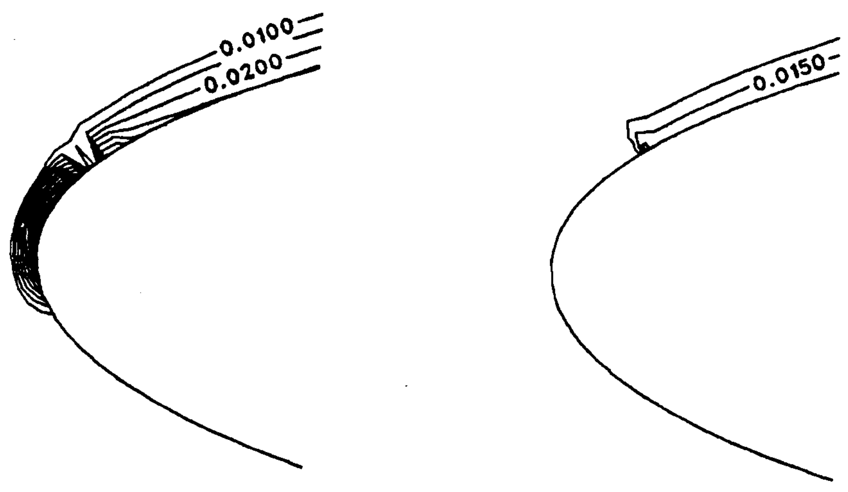


FIGURE 26. Entropy Contours at Leading Edge Before and After Improved Accuracy.

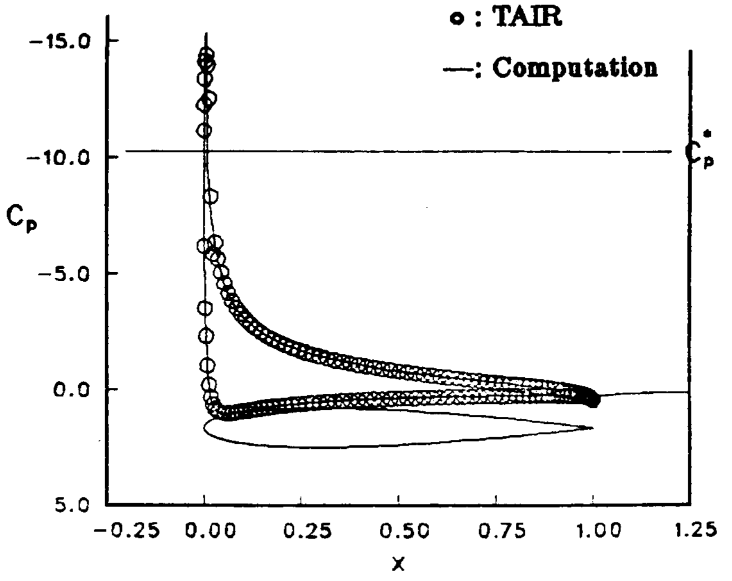


FIGURE 27. Inviscid Solution Compared with TAIR Result.

12.1 Flow Equations

The full three dimensional Navier-Stokes equations in strong conservation law form are reduced to the thin layer form under the same restrictions and assumptions

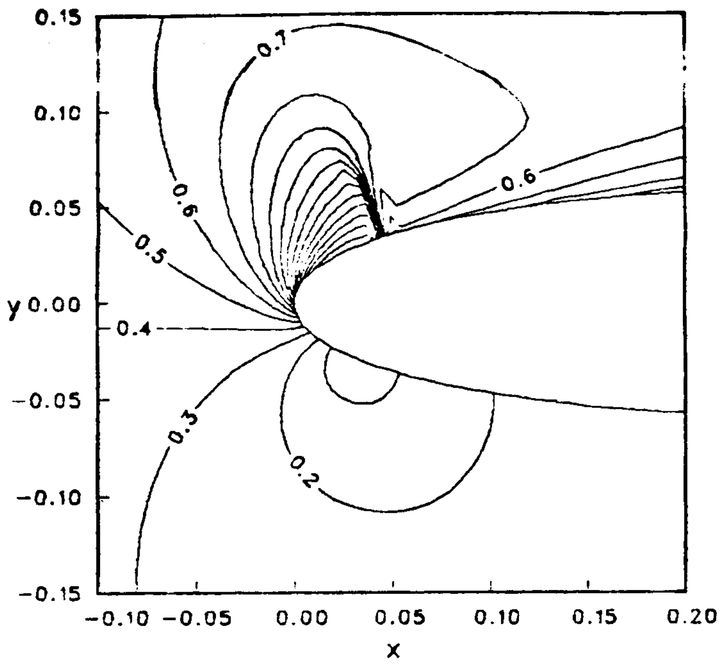


FIGURE 28 MACH CONTOURS AT LEADING EDGE SHOWING SHOCK.

as in two dimensions. The equations in generalized curvilinear coordinates are

$$\partial_\tau \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} + \partial_\zeta \widehat{G} = Re^{-1} \partial_\zeta \widehat{S} \tag{12.1}$$

where now

$$\widehat{Q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \widehat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U(e+p) - \xi_t p \end{bmatrix},$$
$$\widehat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V(e+p) - \eta_t p \end{bmatrix}, \quad \widehat{G} = J^{-1} \begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ W(e+p) - \zeta_t p \end{bmatrix} \tag{12.2a}$$

with

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w, \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \tag{12.2b}$$

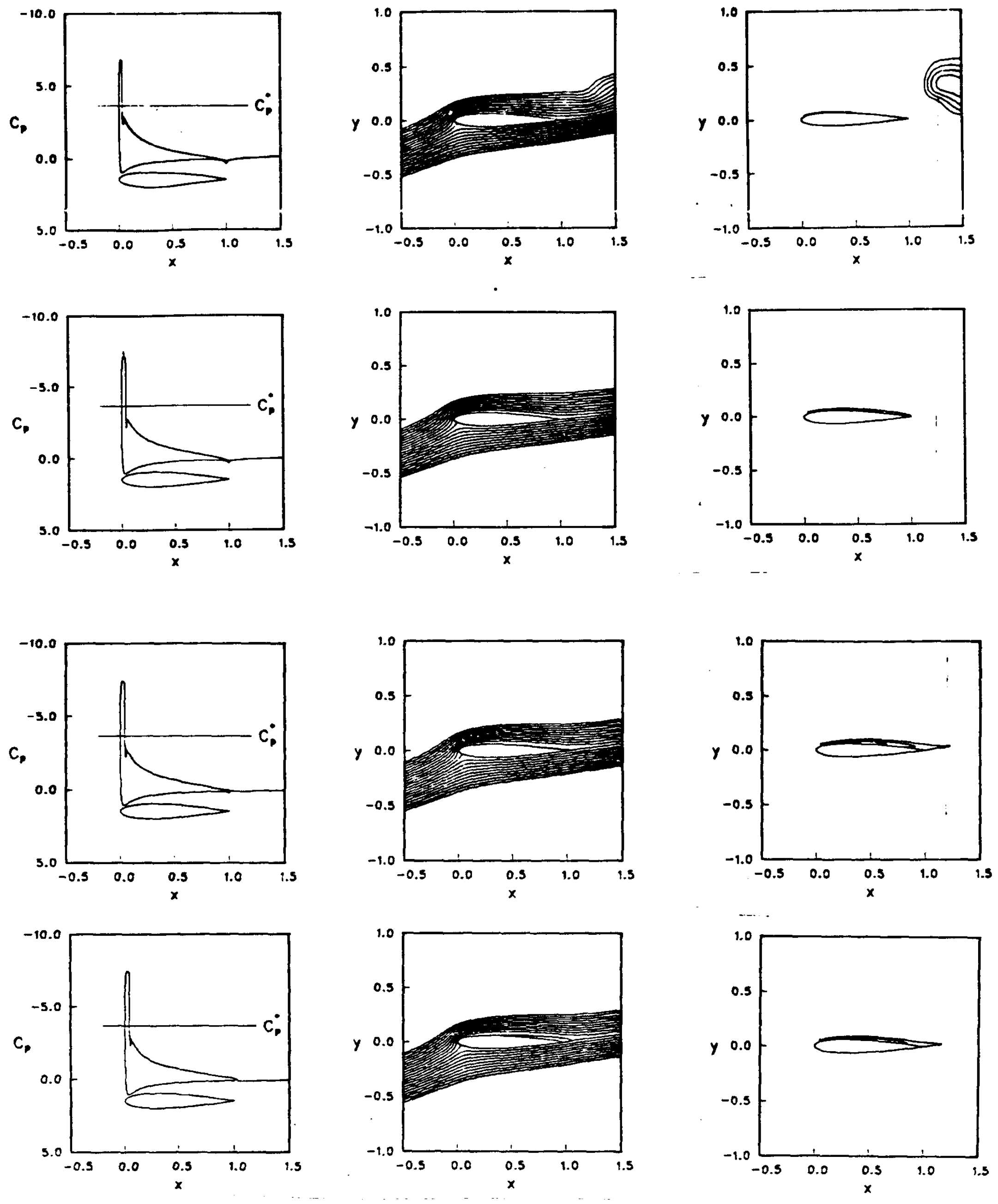


FIGURE 29. Unsteady Solution at $M_\infty = 0.4,\ \alpha = 15°$.

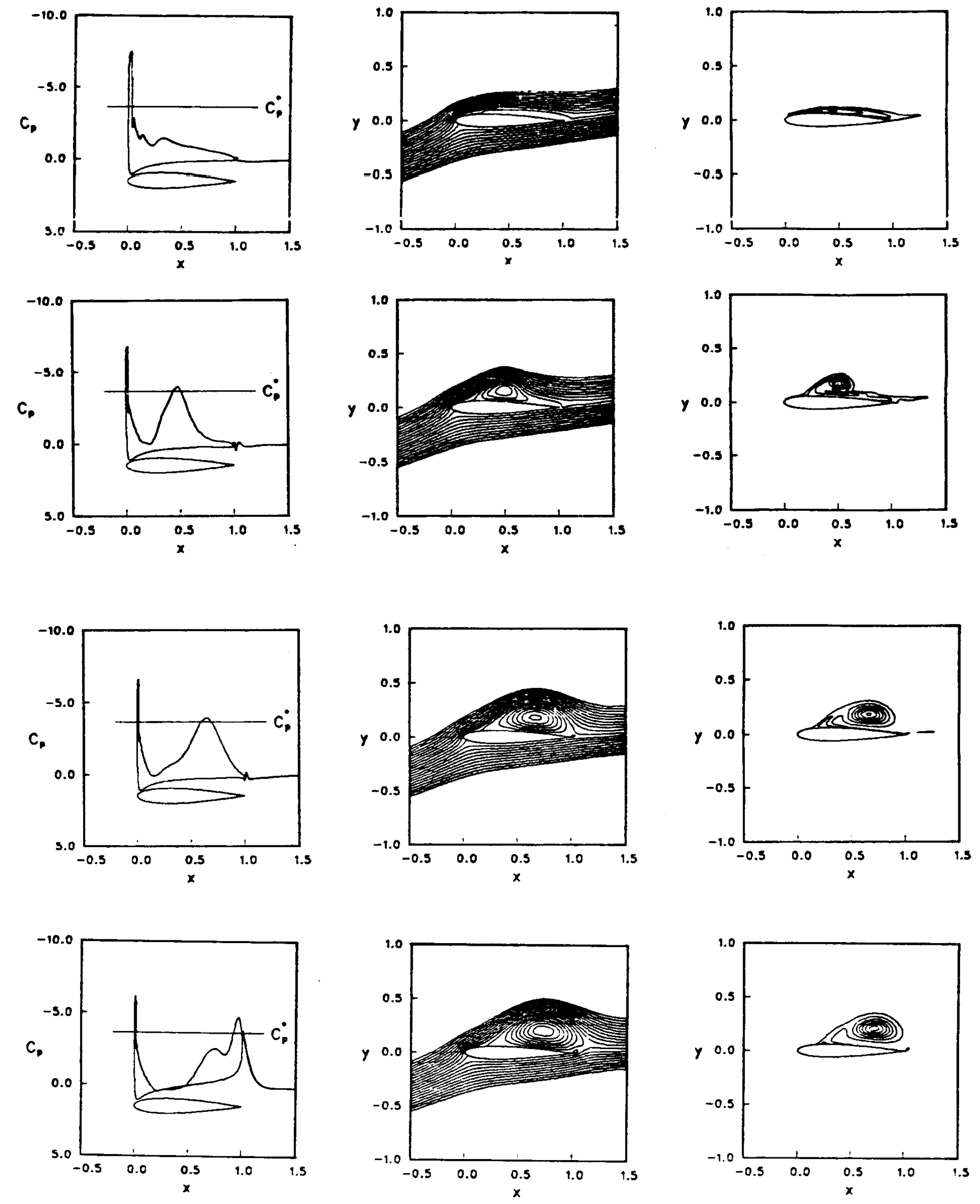


FIGURE 29. Continued.

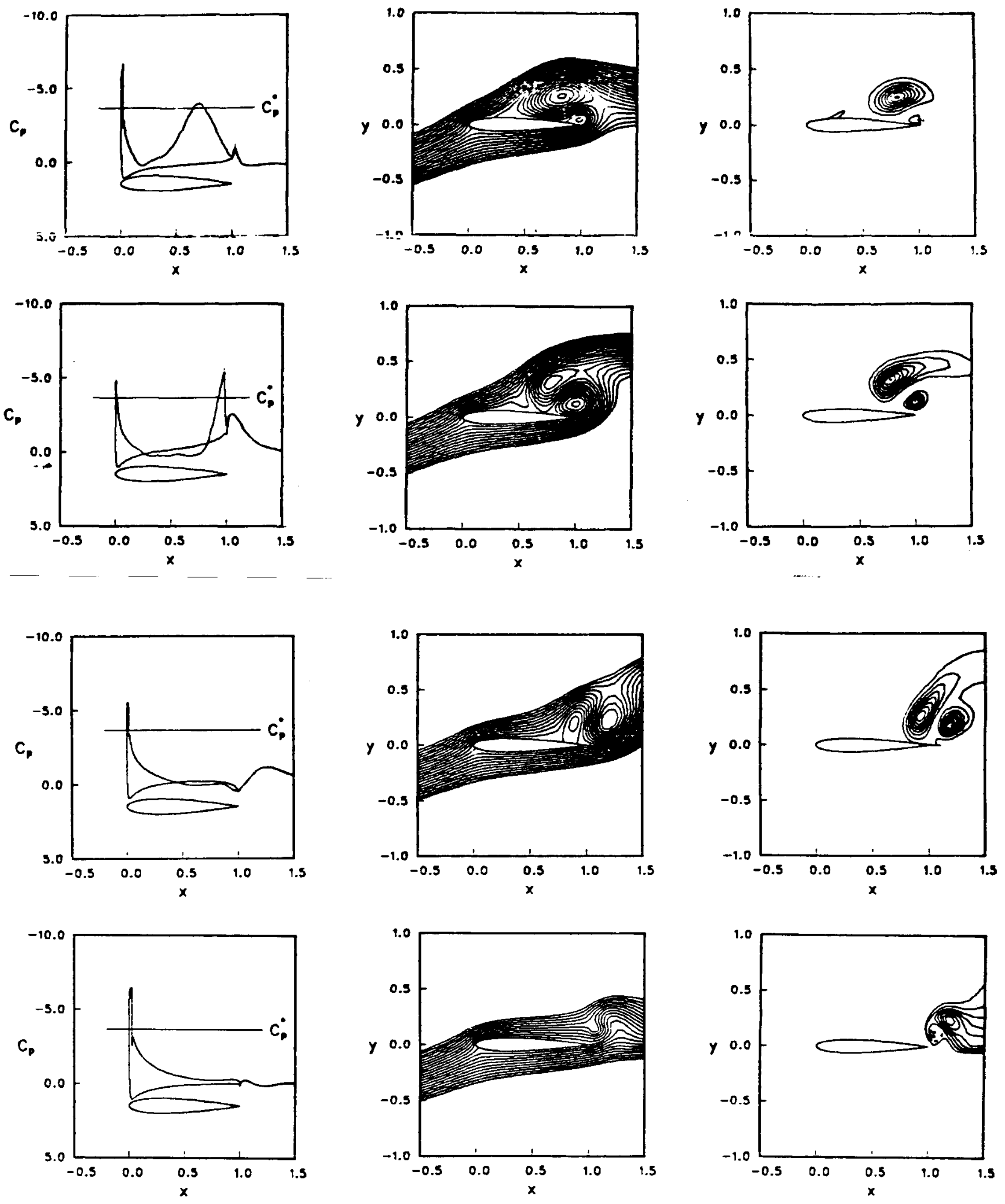


FIGURE 29. Continued.

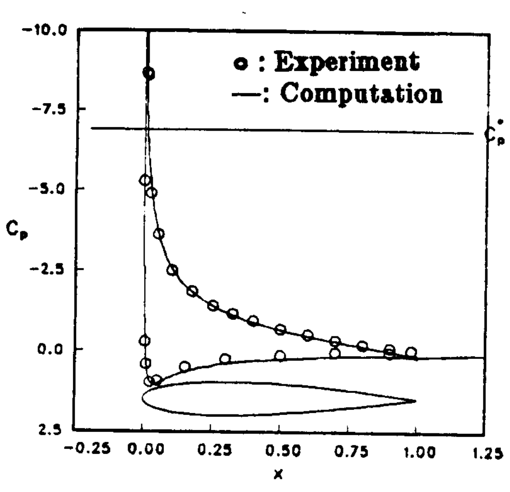


FIGURE 30. Viscous Solution Compared with Experimental Data of McCroskey, et. al.

with

$$\widehat{S} = J^{-1}\begin{bmatrix} 0 \\ \mu m_1 u_\zeta + (\mu/3)m_2\zeta_x \\ \mu m_1 v_\zeta + (\mu/3)m_2\zeta_y \\ \mu m_1 w_\zeta + (\mu/3)m_2\zeta_z \\ \mu m_1 m_3 + (\mu/3)m_2(\zeta_x u + \zeta_y v + \zeta_z w) \end{bmatrix} \tag{12.2c}$$

here $m_1 = \zeta_x^2 + \zeta_y^2 + \zeta_z^2$, $m_2 = \zeta_x u_\zeta + \zeta_y v_\zeta + \zeta_z w_\zeta$, and $m_3 = (u^2+v^2+w^2)_\zeta/2 + Pr^{-1}(\gamma-1)^{-1}(a^2)_\zeta$.

Pressure is again related to the conservative flow variables, Q, by the equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2+v^2+w^2)\right) \tag{12.3}$$

The metric terms are defined as

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta), & \eta_x &= J(z_\xi y_\zeta - y_\xi z_\zeta) \\ \xi_y &= J(z_\eta x_\zeta - z_\zeta x_\eta), & \eta_y &= J(x_\xi z_\zeta - z_\xi x_\zeta) \\ \xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta), & \eta_z &= J(y_\xi x_\zeta - x_\xi y_\zeta) \\ \zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta), & \xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\ \zeta_y &= J(z_\xi x_\eta - x_\xi z_\eta), & \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\ \zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta), & \zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z \end{aligned} \tag{12.4a}$$

with

$$J^{-1} = x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi \tag{12.4b}$$

12.2 Numerical Methods

The implicit approximate factorization algorithm applied to the three dimensional equations is

$$\begin{aligned}\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n\right]\left[I + h\delta_\zeta \widehat{C}^n - hRe^{-1}\delta_\zeta \widehat{M}^n\right]\,\Delta\widehat{Q}^n = \\ -h\left(\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n + \delta_\zeta \widehat{G}^n - Re^{-1}\delta_\zeta \widehat{S}^n\right)\end{aligned} \tag{12.5}$$

The inviscid three dimensional flux Jacobians, $\widehat{A}, \widehat{B}, \widehat{C}$ are defined in the Appendix along with the viscous flux Jacobian $\widehat{M}$. Artificial dissipation terms can be added in the constant coefficient form, straightforward extension of Eqs. (6.1), or in the nonlinear form Eq. (6.12).

The scalar pentadiagonal algorithm in three dimensions has the form

$$T_\xi \left[I + h\,\delta_\xi\,\Lambda_\xi\right]\,\widehat{N}\,\left[I + h\,\delta_\eta\,\Lambda_\eta\right]\widehat{P}\left[I + h\,\delta_\zeta\,\Lambda_\zeta\right]T_\zeta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \tag{12.6}$$

with $\widehat{N} = T_\xi^{-1}T_\eta$ and $\widehat{P} = T_\eta^{-1}T_\zeta$. Just as in two dimensions we use the explicit and implicit nonlinear artificial dissipation terms.

It is interesting (and somewhat disturbing) to note that linear constant coefficient Fourier analysis (periodic BC) for the three dimensional model wave equation shows unconditional instability for the three dimensional factored algorithm. This is due to the cross term errors. In contrast to the case of two dimensions where the cross terms error just affect the rapid convergence capability (at large time steps) of the algorithm. In three dimensions we also have a weak instability due to the cross terms. Linear analysis shows that the instability is a weak one where the amplification factor is very close but greater than one. It can be shown though that a small amount of added artificial dissipation moves the amplification factor below one and therefore we have conditional stability. Also practical model equation analysis using nonperiodic boundary conditions also shows a stability range. In actual practice on three dimensional nonlinear problems we have never encountered a case where we could attribute an instability to this problem area. In fact, numerical experiments show that if anything the three dimensional algorithm seems to be more stable and convergent for a given problem than in two dimensions. Numerous cases have been calculated where the three dimensional algorithm converges and residuals go to machine zero.

12.3 Boundary Conditions and Geometry

Physical boundaries are again mapped to computational boundaries. Explicit numerical and physical conditions can be applied are necessary. The actual conditions used are the straightforward extensions of the methods outlined in Section IX.

One aspect of the three dimensional application which is more complicated then in two dimensions is the development of grid topology and mesh systems. Surface definition and the computational map are complicated by the many surfaces involved, coordinate singularities (unavoidable when mapping a closed 3-D body), the lack of an adequate number of grid points since 3-D is a bigger strain on computer storage limitations. This will definitely drive the computational community towards zonal or patch methods, i.e. the large 3-D problem is broken into multiple sections and each section is handled separately. The interaction between zones can be explicit, see Rai [57] or implicit, Hessenius and Pulliam [58]. Also grid refinement techniques need more development, see Berger [59] and Nakahashi and Deiwert [60] for examples of these concepts.

Geometry seems to be one of the biggest stumbling blocks in three dimensions. Another aspect of 3-D which causes problems is the question of the metric invariants. If central differencing is used to compute the metrics and for the flux derivatives then for 3-D the metric invariants are not satisfied. This was pointed out in the original paper by Pulliam and Steger [3]. By modifying the computation of the metrics we can satisfy the invariants. This is done by averaging the central differences of the grid values, (x_ξ, x_η, etc.) to produce metrics which are similar to terms which would be computed by a finite volume method. For instance, ξ_x would be computed as

$$\xi_x = J\left[(\mu_\zeta \delta_\eta y)(\mu_\eta \delta_\zeta z) - (\mu_\eta \delta_\zeta y)(\mu_\zeta \delta_\eta z)\right] \tag{12.7}$$

where δ is the standard central difference operator and μ is an average operator, for instance $\mu_\eta x_k = (x_{k+1} + x_{k-1})/2$. If all the metric terms are calculated is this manner then the metric invariants are satisfied.

12.4 Code Structure and Vectorization

With the advent of vectorized computer architecture one cannot just program in a linear fashion without regard to code structure and expect to produce efficient code. Two major decisions face the programmer when writing code for a vector machine, for instance CRAY type architecture. First you must identify the vector length or construct. Which indices can be considered vectors, what are the vector lengths (some machines require long vectors) and how is a vector loop implemented. Secondly, how do we manage the large data bases which now can be processed because of the efficiency and speed of vector machines. Many of the near future computers will have either large in-core memory or high speed out-of-core storage. These large amounts of data will have to be managed efficiently. In three dimensions for instance a data base of 1 million grid points and 32 variables will be common place requiring us to manage 32 millions words. To efficiently handle this data base, management systems such as plane slices or pencil concepts, see Lomax and

Pulliam [15], need to be developed and refined. The proper way to handle this is to divide memory into two parts, the operating part of memory and the storage part. Identifiable blocks of data are brought into the working area, processed and then moved back to storage. There are a number of advantages to such a system. One is that large blocks of data can be moved more efficiently especially for out-of-core storage devices but even for in-core storage. The dimensions of the blocks define the vector lengths. For SIMD (single instruction - multiple data) or MIMD (multiple instruction - multiple data) architecture the blocks define multiple data strings to be operated on. In general, as machines become larger and more powerful we will have to take more care in the development of a well structure efficient code.

12.5 Application in Three Dimensions

Applications in three dimensions require substantial computational resources. Most of the problems attempted so far have been for simple geometries and limited flow conditions. The advent of the high speed – large memory machines, such as the CRAY 1S, XMP or CYBER 205, will enable us to attempt realistic problems but even they fall short of providing enough computer power for general purposes. I shall present below applications of the code ARC3D for simple body shapes. The flow conditions used, though, produce some interesting and complicated flowfields. These computations demonstrate the capabilities of the code and demonstrate the accuracy and efficiency.

A) Hemisphere-Cylinder At High Angle Of Attack

The first application is flow past a semi–infinite hemisphere–cylinder at a supersonic Mach number $M_\infty = 1.2$ and high angle of attack $\alpha = 19°$. The calculation is for a laminar Reynolds number $RE = 222500.0$ This calculation was originally presented by Pulliam and Steger [3] where it was computed on a grid with 30 points in the axial direction , 19 points circumferentially and 30 points in the normal direction, a total of 17100 grid points. The grid is a warped spherical topology, see Fig. 31 and is clustered in the normal direction for boundary layer resolution. In the original code [3] the convergence rate (reduction in residual per iteration) for typical cases was on the order of 0.996, for the current algorithm it has been reduced to approximately 0.986. Employing the algorithm as described above and using the fully implicit pentadiagonal algorithm the computation time for three orders of magnitude drop in residual (plotting accuracy) was reduced from about 300 minutes for a converged case on the CDC7600 to about 5 minutes on a CRAY-XMP. In fact cases which could not be converged before on the CDC 7600 are now convergent. This is a substantial reduction in the compute time and can be improved further.

An example of the computation is shown in Fig. 32 and 33. A crossflow separation occurs at this angle of attack which is indicated by the pressure contours and velocity vectors at the cross sectional plane shown. In Fig. 33 pressure along

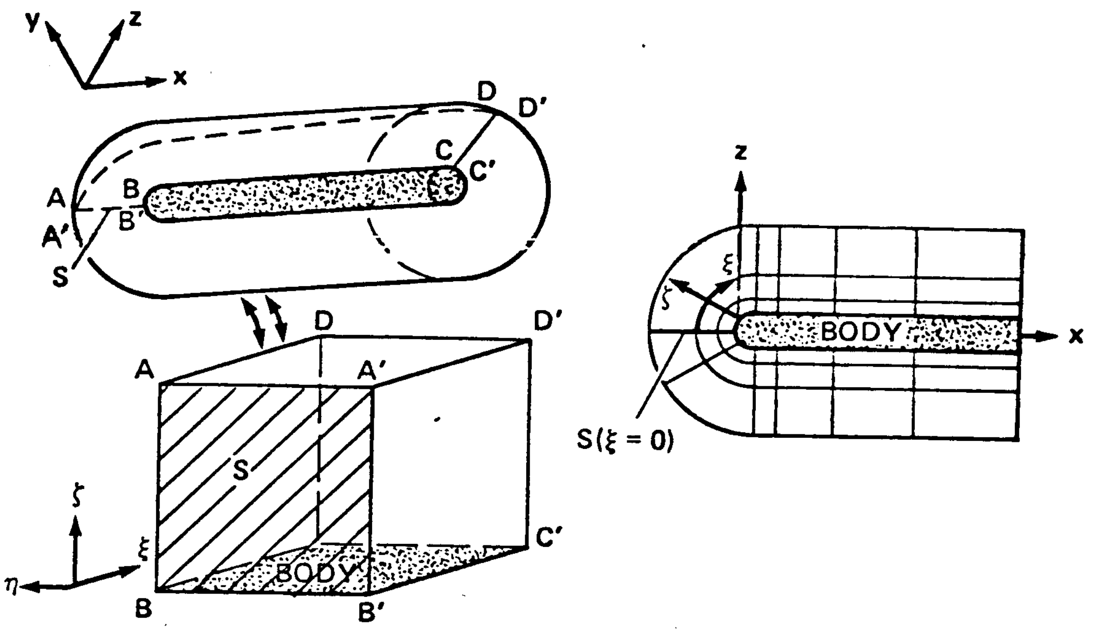


FIGURE 31. Warped Spherical Topology for Hemisphere–Cylinder.

the body at three circumferential locations is compared with experimental data due to Hsieh [61] and compares quite well. Also shown is the computed crossflow separation angle against experiment.

Details of the interaction of the crossflow and symmetry plane as well as other features of this flow require further study. Results by Pan and Pulliam [46] have expanded ARC3D to use the SSD (solid state disk) on the XMP. This give us the capacity for up to 1 million grid points. With that resolution and the increase efficiency of the code we are carring out a detailed study of high angle of attack flowfields.

Other computations presented for this configuration at lower Mach numbers and angles of attack were reported by Pulliam and Steger and compared well with data. The code is also used to obtain starting solutions for a PNS (parabolized Navier-Stokes) code, see Schiff and Steger [62], Chaussee, et. al. [63].

B) Boattail

As a second application, Deiwert employed a version of the code to study boattails at angles of attack [64] and boattail exhaust plumes [65]. A composite of Deiwerts boattail computation is given in Fig. 34. The computation was performed using a boattail configuration with a infinite sting. The region of interest is the converging area of the boattail. The flow conditions are $M_\infty = 2.0$, $Re = 6.5 \cdot 10^7$ and angles of attack from 0° to 12°. Computed pressures at various angles of attack

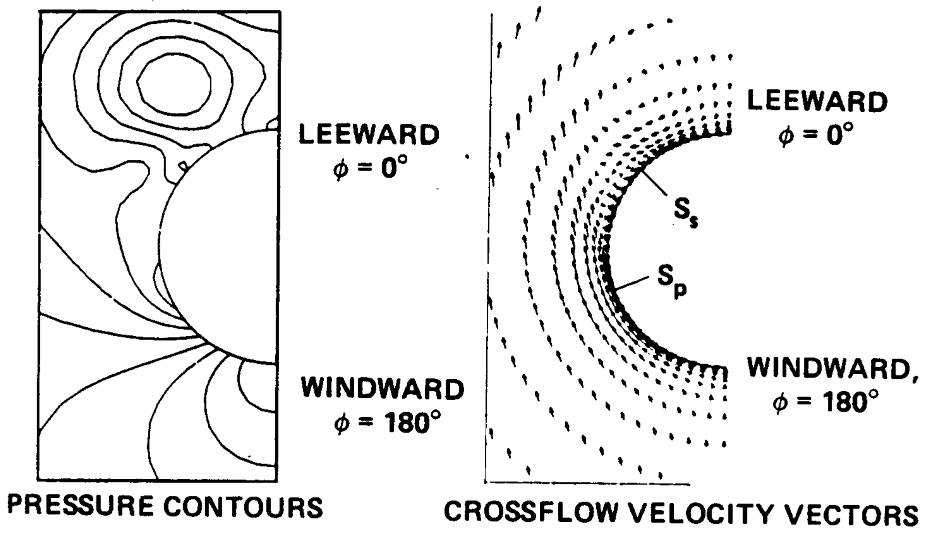


FIGURE 32. Hemisphere–Cylinder at $M_\infty = 1.2, \alpha = 19°$.

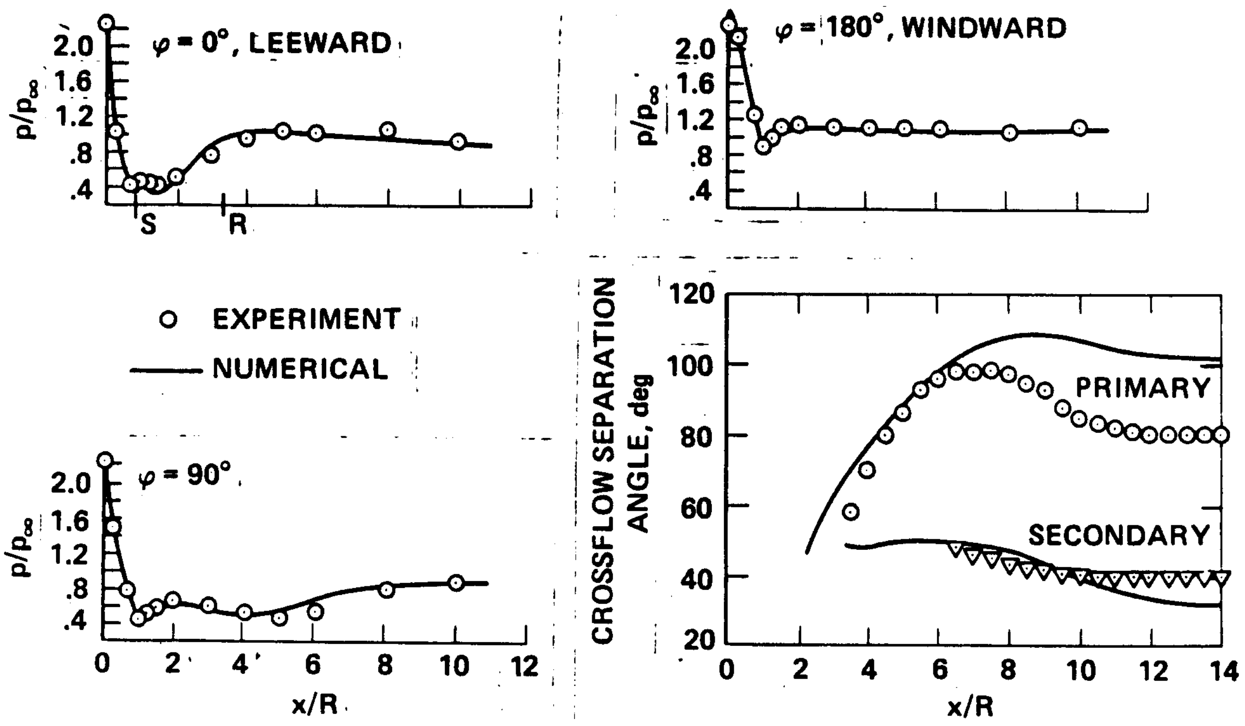


FIGURE 33. Comparison with Experiment.

are compared with experiment in the paper. The case shown here is for $\alpha = 6°$ and shows comparison at three circumferential stations. Computed surface oil flow (particle paths restricted to the body) and surface pressure for $\alpha = 6°$ show the type of results presented. The results reported by Deiwert [64] compared very well with the experimental data of Shrewsbury [66].

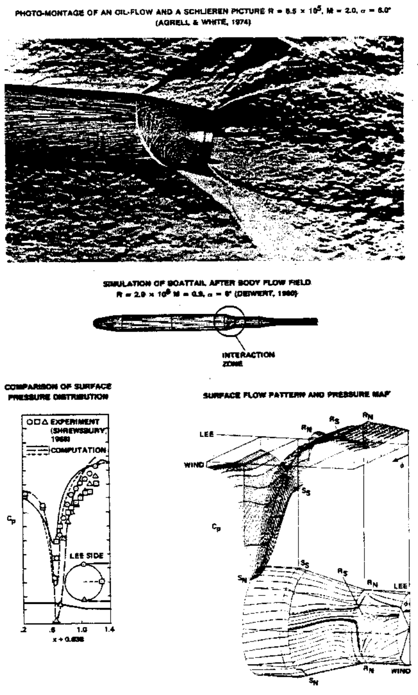
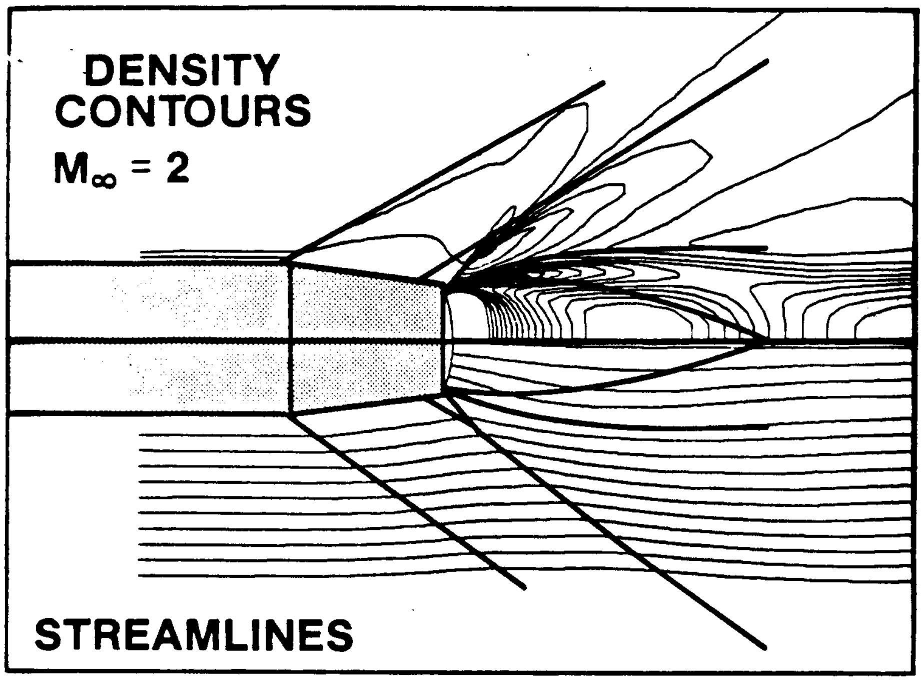


PHOTO-MONTAGE OF AN OIL-FLOW AND A SCHLIEREN PICTURE R = 6.5 × 10^5, M = 2.0, α = 6.0°
(AGRELL & WHITE, 1974)
SIMULATION OF BOATTAIL AFTER BODY FLOW FIELD
R = 2.9 × 10^6 M = 0.9, α = 6° (DEIWERT, 1980)
INTERACTION ZONE
COMPARISON OF SURFACE PRESSURE DISTRIBUTION
○□△ EXPERIMENT (SHREWSBURY, 1968)
COMPUTATION
LEE SIDE
Cp
.2
.6
1.0
1.4
x ÷ 0.636
SURFACE FLOW PATTERN AND PRESSURE MAP
LEE
WIND
Cp
RN
RS
SS
SN
φ

FIGURE 34. Boattail Study at $M_\infty = 2.0, \alpha = 6°, Re = 6.5 \times 10^5$.

The boattail study was undertaken as a first step toward the simulation of boattail exhaust plumes which has been carried out in a preliminary stage by Deiwert [65]. In this simulation the boattail sting is eliminated and a conical exhaust jet is added at the base region. Figures 35 and 36 show comparisons of density contours and streamlines with Schlieren photographs from Agrell and White [67]. In Fig. 35 the pressure ratio for the jet was 3 and we see an expanded exhaust plume and a complicated shock shear flow pattern (depicted by the bold lines). The qualitative comparison with the photograph is quite remarkable. At a higher exhaust ratio, Fig. 36 the jet is tighter and the shock shear surface pattern more complicated. Again the qualitative comparison with the photograph is quite good. I refer the reader to the original papers and one by Nakahashi and Deiwert [60] for a more detailed analysis of these flowfields.



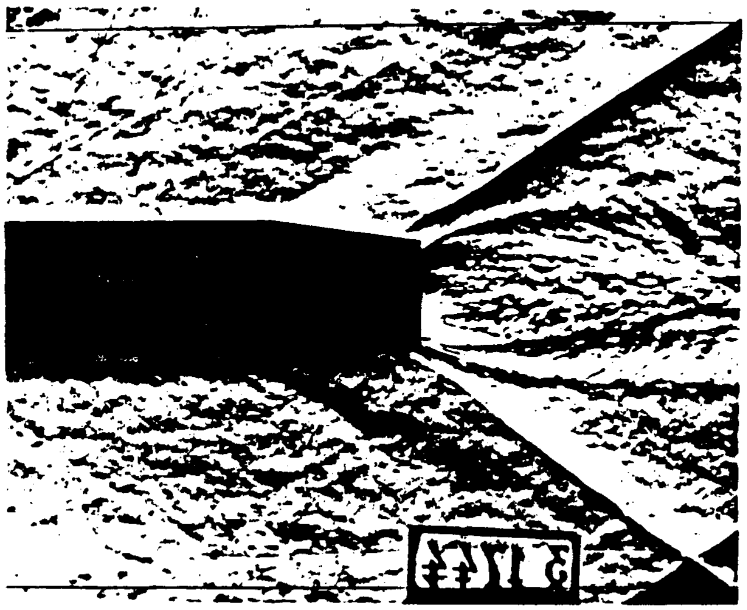
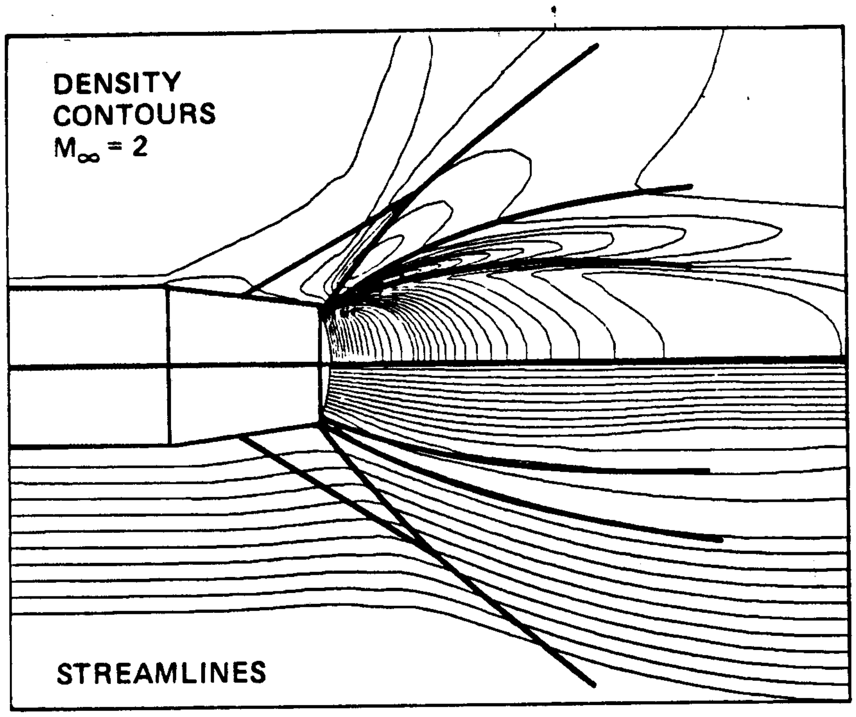


Figure 35. Boattail Exhaust Plume Flow Details at Pressure Ratio = 3.



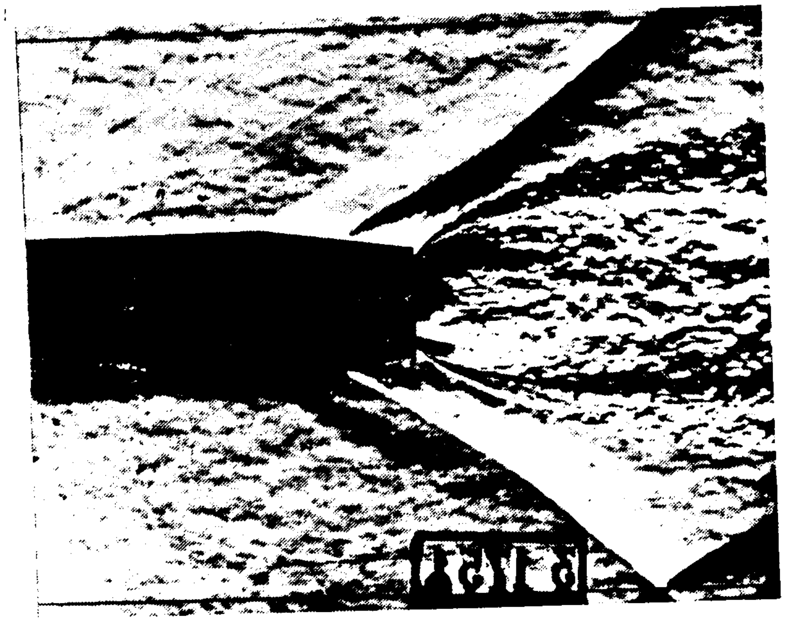


FIGURE 36. Boattail Exhaust Plume Flow Details at Pressure Ratio = 9.

Summary

In summary, the development of some computational algorithms in two- and three-dimensions have been presented. Details of two computational codes, ARC2D and ARC3D have been presented. The basic algorithm used is the Beam and Warming implicit approximate factorization scheme or variants of that scheme such as the diagonalization. The codes employ improvements to enhance accuracy, (grid refinement, better boundary conditions, more versatile artificial dissipation model) and efficiency (diagonal algorithm, implicit treatment of artificial dissipation terms, variable time steps). Results for a wide variety of cases substantiate the accuracy and efficiency claims.

Future work is required to address improvement of boundary conditions, examining stability questions, eliminating cross term errors and more. We are also interested in developing new grid generation concepts both in terms of generation and grid quality. In three dimensions we see the area of zonal concepts as the newest horizon and envision substantial gains in solution capability as a result.

References

1 Beam, R. and Warming, R. F. *An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form*, J. Comp. Phys., Vol. 22 1976, pp. 87-110

2 Steger, J. L. *Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils* , AIAA Paper 77-665, 1977

3 Pulliam, T. H. and Steger, J. L. *Implicit Finite- Difference Simulations of Three Dimensional Compressible Flow*, AIAA J Vol. 18 1980 page 159

4 MacCormack, R. W. *The Effect of Viscosity in Hypervelocity Impact Cratering*, AIAA Paper 69-354, 1969.

5 Rizzi, A. and Eriksson, L. *Explicit Multistage Finite Volume Procedure to Solve the Euler Equations for Transonic Flow Lecture* , Series on Computational Fluid Dynamics, von Kárm án Institute, Rhode-St-Genese, Belgium 1983

6 Jameson, A., Schmidt, W. and Turkel, E. *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge- Kutta Time- Stepping Schemes*, AIAA paper 81-1259 AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto 1981

7 Yee, H. C., and Harten, A. *Implicit TVD Schemes for Hyperbolic Conservation Laws in Curvilinear Coordinates*, AIAA-85-1513 AIAA 7th Computational Fluid Dynamics Conference, Cincinatti, Oh., 1985

8 Chakravarthy, S., and Osher, S. *A New Class of High Accuracy TVD Schemes for Hyperbolic Conservation Laws*, AIAA-85-0363 AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada., 1985

9 Anderson, W. K., Thomas, J. L., and Van Leer, B. *A Comparison of Finite Volume Flux Vector Splittings for The Euler Equations*, AIAA-85-0122, AIAA 23rd Aerospace Sciences Meeting, Reno, NV, 1985

10 Viviand, H. *Conservative Forms of Gas Dynamics Equations*, La Recherche Aerospatiale 1974 page 65

11 Vinokur, M. *Conservation Equations of Gas-Dynamics in Curvilinear Coordinate Systems*, J. Comp. Phys. pp. 105-125 Vol. 14, 1974

aa Korn, G. and Korn, T., *Mathemaical Handbook for Scientists and Engineers*, Mc Graw Hill Book Co, New York, 1961

12 Hindman, R. *Geometrically Induced Errors and Their Relationship to the Form of the Governing Equations and the Treatment of Generalized Mappings*, AIAA Paper 81-1008 1981

13 Flores, J., Holst, T., Kwak, D., and Batiste, D. *A New Consistent Spatial Differencing Scheme for the Transonic Full Potential Equation* , AIAA Paper 83-073 1983

14 Baldwin, B. S. and Lomax, H. *Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows,* , AIAA Paper No. 78-257 1978

15 Lomax H. and Pulliam T. H. *A Fully Implicit Factored Code for Computing Three Dimensional Flows on the ILLIAC IV*, Parallel Computations, G. Rodrigue, Ed., Academic Press, New York 1982 pp. 217-250

16 Barth, T. J. and Steger, J. L. *A Fast Efficient Implicit Scheme for the Gasdynamics Equations Using A Matrix Reduction Technique* , AIAA-85-0439, AIAA 23rd Aerospace Sciences Meeting, Reno, NV, 1985

17 Warming, R. F., and Beam, R. M. *On the Construction and Application of Implicit Factored Schemes for Conservation Laws*, SIAM-AMS Proceedings Vol. 11 1978 pp. 85–129

18 Jespersen, D. C. *Recent Developments in Multigrid Methods for the Steady Euler Equations*, Lecture Notes for Lecture Series on Computational Fluid Dynamics 1984, von Kárm án Institute, Rhode-St-Genese, Belgium

19 Pulliam, T. H. and Chaussee, D. S. *A Diagonal Form of an Implicit Approximate Factorization Algorithm*, J C P Vol. 39 1981 page 347

20 Warming, R. F., Beam, R., and Hyett, B. J. *Diagonalization and Simultaneous Symmetrization of the Gas-Dynamic Matrices*, Math Comp, Vol. 29 1975 page 1037

21 Steger, J. *Coefficient Matrices for Implicit Finite Difference Solution of the Inviscid Fluid Conservation Law Equations*, Computer Methods In Applied Mechanics and Engineering Vol. 13 pp. 175-188 1978

22 Beam, R. and Bailey, H *Private Communication*

23 Steger J. L. and Warming, R. F. *Flux Vector Splitting of the Inviscid Gas Dynamic Equations with Applications to Finite Difference Methods*, J. Comp. Phys. Vol. 40 pp. 263-293 1981

24 Roe, P. L. *The Use of the Riemann Problem in Finite Difference Schemes*, presented at the Seventh International Conference on Numerical Methods in Fluid Dynamics, Stanford, CA 1980

25 van Leer, B. *Flux-Vector Splitting for the Euler Equations* Eighth International Conference on Numerical Methods in Fluid Dynamics, Springer Lecture Notes in Physics no. 170, ed. E. Krause 1983

26 Osher S. and Chakravarthy, S. *Upwind Schemes and Boundary Conditions with Applications to Euler Equations in General Geometries*, J. Comp. Phys. Vol. 50 pp. 447–481 1983

27 Harten, A. *A High Resolution Scheme for the Computation of Weak Solutions of Hyperbolic Conservation Laws*, J. Comp. Phys. Vol. 49 1983 pp. 357-393

28 Pulliam, T. H. *Artificial Dissipation Models For the Euler Equations* , AIAA 85-0438 AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada., 1985

29 Abarbanel, S., Dwoyer, D. and Gottlieb, D. *Improving the Convergence Rate of Parabolic ADI Methods*, ICASE Report 82-28 1982

30 Shang, J. and Hankey, W. , *Numerical Solution of the Compressible Navier-Stokes Equations for a Three-Dimensional Corner* , AIAA paper 77-169 1977

31 McDonald, H. and Briley, W. R. *Computational Fluid Dynamic Aspects of Internal Flows* , AIAA paper 79-1445, Proceedings of AIAA Computational Fluid Dynamics Conference, Williamsburg, Va. 1979

32 Srinivasan, G., Chyu, W. , and Steger, J. *Computation of Simple Three- Dimensional Wing- Vortex Interaction in Transonic Flow* , AIAA paper 81-1206 1981

33 Coakley, T. J. *Numerical Method for Gas Dynamics Combining Characteristic and Conservation Concepts* , AIAA Paper 81-1257 1981

34 Yee, H. C. *Numerical Approximation of Boundary Conditions with Applications to Inviscid Equations of Gas Dynamics*, NASA TM-81265 1981

35 Chakravarthy, S. *Euler Equations – Implicit Schemes and Implicit Boundary Conditions* , AIAA Paper 82-0228, Orlando, Fla. 1982

36 Salas, M. , Jameson, A., and Melnik, R. *A Comparative Study of the Nonuniqueness Problem of the Potential Equation* , AIAA paper 83-1888, AIAA 6th Computational Fluid Dynamics Conference 1983

37 Eiseman, P. *Geometric Methods in Computational Fluid Dynamics*, ICASE Report 80-11 1980

38 Thompson, J. , Thames, F. and Mastin, C. *Automatic Numerical Generation of Body Fitted Curvilinear Coordinate Systems for Field Containing Any Number of Arbitrary Two Dimensional Bodies*, J. Comp. Phy. Vol. 15 pp. 299-319 1974

39 Thompson, J. *Numerical Grid Generation*, J. Thompson Ed. North Holland, 1982

40 Steger, J. L. and Bailey, H. E. *Calculation of Transonic Aileron Buzz* , AIAA Jour. 18 pp. 249-55 1980

41 Steger, J. L., Pulliam, T. H., and Chima, R. V. *An Implicit Finite Difference Code for Inviscid and Viscous Cascade Flow* , AIAA paper 80–1427, AIAA

13th Fluid and Plasma Dynamics Conference, Snowmass, Colorado 1980

42 Chaussee, D. S. and Pulliam, T. H. *A Diagonal Form of an Implicit Approximate Factorization Algorithm with Application to a Two Dimensional Inlet* , AIAA J. Vol. 19 1981 page 153

43 Barth, T. J., Pulliam, T. H., and Buning, P. G. *Navier- Stokes Computations For Exotic Airfoils* , AIAA-85-0109 AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada., 1985

44 Pulliam, T. H., Jespersen, D. C., and Barth, T. J. *Navier- Stokes Computations For Circulation Controlled Airfoils*, AIAA-85-1587 AIAA 7th Computational Fluid Dynamics Conference, Cincinatti, Oh., 1985

45 Kutler, P., Chakravarthy, S. , and Lombard, C. *Supersonic Flow Over Ablated Nosetips Using an Unsteady Implicit Numerical Procedure* , AIAA Paper 78–213 1978

46 Pan, D. and Pulliam, T. H. *The Computation of Steady 3-D Separated Flows Over Aerodynamic Bodies At High Incidence and Yaw*, AIAA 86-0109, AIAA 24rd Aerospace Sciences Meeting, Reno, NV, 1986

47 Pulliam, T. H., Jespersen, D. C.,and Childs, R., E. *An Enhanced Version of an Implicit Code for the Euler Equations* , AIAA paper 83-0344 AIAA 21st Aerospace Sciences Meetings, Reno, NV. 1983

48 Jameson, A. *Solution of the Euler Equations for Two-Dimensional Transonic Flow by a Multigrid Method*, Appl. Math. and Computation Vol. 13 pp. 327–355 1983

49 *The 1980-81 AFOSR-HTTM-Stanford Conference on Complex Turbulent Flows: Comparison of Computation and Experiment*, Vol. 2, Taxonomies, Reporters' Summaries, Evaluation and Conclusions, Eds. S. J. Kline, B. J. Cantwell and G. M. Lilley, Thermoscience Division, Stanford University, California 1982

50 Cook, P. , McDonald, M. and Firmin, M. *Aerofoil RAE 2822 - Pressure Distributions, and Boundary layer and Wake Measurements*, AGARD- AR- 138 1979

51 Mehta, U. *Reynolds Averaged Navier–Stokes Computations of Transonic Flows Around Airfoils*, Presented at Second Symposium on Numerical and Physical Aspects of Aerodynamic Flows, Long Beach, Calif, 1983

52 Erikson, A. L. and Stephenson, J. D. *A Suggested Method of Analyzing for Transonic Flutter of Control Surfaces Based on Available Experimental Evidence*, NACA RM A7F30 1947

53 Barton, J. T. and Pulliam, T. H. *Airfoil Computation at High Angles of Attack, inviscid and Viscous Phenomena* , AIAA 84–0524, AIAA 22nd Aerospace Science Meeting, Reno, Nevada 1984

54 Anderson, W., Thomas, J. , and Rumsey, C. *Application of Thin Layer Navier Stokes Equations Near Maximum Lift* , AIAA 84–0049, AIAA 22nd Aerospace science Meeting, Reno, Nevada 1984

55 Holst, T. L. *Implicit Algorithm for the Conservative Transonic Full-Potential Equation Using an Arbitrary Mesh* , AIAA J. 17, 1979 pp. 1038–1045

56 McCroskey, W., McAlister, K., Carr, L., and Pucci, S. *An Experimental Study of Dynamic Stall on Advanced Airfoil Sections*, Vol. 1, Summary of Experiment NASA TM 84245 1982

57 Rai, M. M. *A Conservative Treatment of Zonal Boundaries for Euler Equations Calculations*, AIAA paper 84–0164, AIAA 22nd Aerospace Science Meeting, Reno, NV 1984

58 Hessenius, K. E. and Pulliam, T. H. *A Zonal Approach to Solution of the Euler Equations* , AIAA paper 82-0969 1982

59 Berger,M. *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*, Ph. D. Thesis, Department of Computer Science, Stanford University 1982.

60 Nakahashi, K. and Deiwert, G. S. *A Practical Adaptive Grid Method for Complex Fluid Flow Problems*, Submitted to Ninth International Conference on Numerical Methods in Fluid Dynamics, Saclay, France 1984

61 Hsieh, T. *An Investigation of Separated Flow About a Hemisphere–Cylinder at 0- to 90-deg Incidence in the Mach Number Range from 0.6 to 1.5*, AEDC- TR-76- 112 1976

62 Schiff, L. B., and Steger, J. L. *Numerical Simulation of Steady Supersonic Viscous Flow* , AIAA paper 79-0130, AIAA 17th Aerospace Sciences Meeting, New Orleans, La. 1979

63 Chaussee D. S., Patterson J. L., Kutler P. , Pulliam T. H. and Steger J. L. *A Numerical Simulation of Hypersonic Viscous Flows over Arbitrary Geometries at High Angle of Attack* , AIAA Paper no. 81-0050 1981

64 Deiwert, G. S. *Numerical Simulation of Three- Dimensional Boattail Afterbody Flowfields* , AIAA J. Vol. 19 1981 pp. 582-588

65 Deiwert, G. S. *A Computational Investigation of Supersonic Axisymmetric Flow Over Boattails Containing a Centered Propulsive Jet* , AIAA Paper 83-0462 1983

66 Shrewsbury, G. D. *Effect of Boattail Juncture Shape on Pressure Drag Coefficients of Isolated Afterbodies*, NASA TM X-1517 1968

67 Agrell, J. and White, R. A. *An Experimental Investigation of Supersonic Axisymmetric Flow over Boattails Containing a Centered* , Propulsive Jet FFA Tech. Note AU-913 1974

Appendix

The flux Jacobian matrices of Eq. (5.4) have real eigenvalues and a complete set of eigenvectors. The similarity transforms are

$$\widehat{A} = T_\xi \Lambda_\xi T_\xi^{-1} \quad \text{and} \quad \widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1} \tag{A.1}$$

where

$$\Lambda_\xi = \begin{bmatrix} U & & & \\ & U & & \\ & & U + a\sqrt{\xi_x^2 + \xi_y^2} & \\ & & & U - a\sqrt{\xi_x^2 + \xi_y^2} \end{bmatrix}, \tag{A.2a}$$

$$\Lambda_\eta = \begin{bmatrix} V & & & \\ & V & & \\ & & V + a\sqrt{\eta_x^2 + \eta_y^2} & \\ & & & V - a\sqrt{\eta_x^2 + \eta_y^2} \end{bmatrix}, \tag{A.2b}$$

with

$$T_\kappa = \begin{bmatrix} 1 & 0 & \alpha & \alpha \\ u & \widetilde{\kappa}_y \rho & \alpha(u + \widetilde{\kappa}_x a) & \alpha(u - \widetilde{\kappa}_x a) \\ v & -\widetilde{\kappa}_x \rho & \alpha(v + \widetilde{\kappa}_y a) & \alpha(v - \widetilde{\kappa}_y a) \\ \frac{\phi^2}{(\gamma-1)} & \rho(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v) & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} + a\widetilde{\theta}\right] & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} - a\widetilde{\theta}\right] \end{bmatrix} \tag{A.3}$$

$$T_\kappa^{-1} = \begin{bmatrix} (1 - \phi^2/a^2) & (\gamma-1)u/a^2 & (\gamma-1)v/a^2 & -(\gamma-1)/a^2 \\ -(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x/\rho & 0 \\ \beta(\phi^2 - a\widetilde{\theta}) & \beta[\widetilde{\kappa}_x a - (\gamma-1)u] & \beta[\widetilde{\kappa}_y a - (\gamma-1)v] & \beta(\gamma-1) \\ \beta(\phi^2 + a\widetilde{\theta}) & -\beta[\widetilde{\kappa}_x a + (\gamma-1)u] & -\beta[\widetilde{\kappa}_y a + (\gamma-1)v] & \beta(\gamma-1) \end{bmatrix} \tag{A.4}$$

and $\alpha = \rho/(\sqrt{2}a)$, $\beta = 1/(\sqrt{2}\rho a)$, $\widetilde{\theta} = \widetilde{\kappa}_x u + \widetilde{\kappa}_y v$, and, forexample, $\widetilde{\kappa}_x = \kappa_x/\sqrt{\kappa_x^2 + \kappa_y^2}$.

Relations exist between T_ξ and T_η of the form

$$\widehat{N} = T_\xi^{-1} T_\eta, \quad \widehat{N}^{-1} = T_\eta^{-1} T_\xi \tag{A.5}$$

where

$$\widehat{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & -\mu m_2 & \mu m_2 \\ 0 & \mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & -\mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix} \qquad (A.6a)$$

and

$$\widehat{N}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & \mu m_2 & -\mu m_2 \\ 0 & -\mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & \mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix} \qquad (A.6b)$$

with $m_1 = \left(\widetilde{\xi}_x\ \widetilde{\eta}_x + \widetilde{\xi}_y \widetilde{\eta}_y\right)$, $m_2 = \left(\widetilde{\xi}_x\ \widetilde{\eta}_y - \widetilde{\xi}_y \widetilde{\eta}_x\right)$ and $\mu = 1/\sqrt{2}$.

It is interesting to note that the matrix $\widehat{N}$ is only a function of the metrics and not the flow variables.

In three dimensions the Jacobian matrices $\widehat{A}$ or $\widehat{B}$ or $\widehat{C}$ =

$$\begin{bmatrix} \kappa_t & \kappa_x & \kappa_y & \kappa_z & 0 \\ \kappa_x \phi^2 - u\theta & \kappa_t + \theta - \kappa_x(\gamma-2)u & \kappa_y u - \kappa_x(\gamma-1)v & \kappa_z u - \kappa_x(\gamma-1)w & \kappa_x(\gamma-1) \\ \kappa_y \phi^2 - v\theta & \kappa_x v - \kappa_y(\gamma-1)u & \kappa_t + \theta - \kappa_y(\gamma-2)v & \kappa_z v - \kappa_y(\gamma-1)w & \kappa_y(\gamma-1) \\ \kappa_z \phi^2 - w\theta & \kappa_x w - \kappa_z(\gamma-1)u & \kappa_y w - \kappa_z(\gamma-1)v & \kappa_t + \theta - \kappa_z(\gamma-2)w & \kappa_z(\gamma-1) \\ -\theta\left(\gamma e/\rho - 2\phi^2\right) & \kappa_x\left(\gamma e/\rho - \phi^2\right) - (\gamma-1)u\theta & \kappa_y\left(\gamma e \rho^{-1} - \phi^2\right) - (\gamma-1)v\theta & \kappa_z\left(\gamma e \rho^{-1} - \phi^2\right) - (\gamma-1)w\theta & \kappa_t + \gamma\theta \end{bmatrix} \qquad (A.7)$$

where

$$\theta = \kappa_x u + \kappa_y v + \kappa_z w$$

$$\phi^2 = (\gamma-1)(\frac{u^2+v^2+w^2}{2})$$

with $\kappa = \xi$, or η or ζ for $\widehat{A}, \widehat{B}$, or, $\widehat{C}$ respectively.

The viscous flux Jacobian is

$$\widehat{M} = J^{-1} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1 \partial_\zeta(\rho^{-1}) & \alpha_2 \partial_\zeta(\rho^{-1}) & \alpha_3 \partial_\zeta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2 \partial_\zeta(\rho^{-1}) & \alpha_4 \partial_\zeta(\rho^{-1}) & \alpha_5 \partial_\zeta(\rho^{-1}) & 0 \\ m_{41} & \alpha_3 \partial_\zeta(\rho^{-1}) & \alpha_5 \partial_\zeta(\rho^{-1}) & \alpha_6 \partial_\zeta(\rho^{-1}) & 0 \\ m_{51} & m_{52} & m_{53} & m_{54} & \alpha_0 \partial_\zeta(\rho^{-1}) \end{bmatrix} J \qquad (A.8a)$$

where

$$\begin{aligned}
m_{21} =& -\alpha_1 \partial_\zeta(u/\rho) - \alpha_2 \partial_\zeta(v/\rho) - \alpha_3 \partial_\zeta(w/\rho) \\
m_{31} =& -\alpha_2 \partial_\zeta(u/\rho) - \alpha_4 \partial_\zeta(v/\rho) - \alpha_5 \partial_\zeta(w/\rho) \\
m_{41} =& -\alpha_3 \partial_\zeta(u/\rho) - \alpha_5 \partial_\zeta(v/\rho) - \alpha_6 \partial_\zeta(w/\rho) \\
m_{51} =& \alpha_0 \partial_\zeta \left[-(e/\rho^2) + (u^2+v^2+w^2)/\rho\right] \\
& - \alpha_1 \partial_\zeta(u^2/\rho) - \alpha_4 \partial_\zeta(v^2/\rho) - \alpha_6 \partial_\zeta(w^2/\rho) \\
& - 2\alpha_2 \partial_\zeta(uv/\rho) - 2\alpha_3 \partial_\zeta(uw/\rho) - 2\alpha_5 \partial_\zeta(vw/\rho) \qquad (A.8b) \\
m_{52} =& -\alpha_0 \partial_\zeta(u/\rho) - m_{21} \qquad m_{53} = -\alpha_0 \partial_\zeta(v/\rho) - m_{31} \\
m_{54} =& -\alpha_0 \partial_\zeta(w/\rho) - m_{41} \qquad m_{44} = \alpha_4 \partial_\zeta(\rho^{-1}) \\
\alpha_0 =& \gamma\mu Pr^{-1}({\zeta_x}^2 + {\zeta_y}^2 + {\zeta_z}^2) \quad \alpha_1 = \mu[(4/3){\zeta_x}^2 + {\zeta_y}^2 + {\zeta_z}^2], \\
\alpha_2 =& (\mu/3)\zeta_x\zeta_y, \quad \alpha_3 = (\mu/3)\zeta_x\zeta_z, \quad \alpha_4 = \mu[{\zeta_x}^2 + (4/3){\zeta_y}^2 + {\zeta_z}^2], \\
\alpha_5 =& (\mu/3)\zeta_y\zeta_z, \qquad \alpha_6 = \mu[{\zeta_x}^2 + {\zeta_y}^2 + (4/3){\zeta_z}^2],
\end{aligned}$$

The eigensystem decomposition of the three dimensional Jacobians have the form $\widehat{A} = T_\xi \Lambda_\xi T_\xi^{-1}$, $\widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1}$, and $\widehat{C} = T_\zeta \Lambda_\zeta T_\zeta^{-1}$. The eigenvalues are

$$\begin{aligned}
\lambda_1 &= \lambda_2 = \lambda_3 = \kappa_t + \kappa_x u + \kappa_y v + \kappa_z w \\
\lambda_4 &= \lambda_1 + \kappa a \quad \lambda_5 = \lambda_1 - \kappa a \qquad (A.9) \\
\kappa &= \sqrt{\kappa_x^2 + \kappa_y^2 + \kappa_z^2}
\end{aligned}$$

The matrix T_κ, representing the left eigenvectors, is

$$T_\kappa = \begin{bmatrix}
\widetilde{\kappa}_x & \widetilde{\kappa}_y & \widetilde{\kappa}_z & \alpha & \alpha \\
\widetilde{\kappa}_x u & \widetilde{\kappa}_y u - \widetilde{\kappa}_z \rho & \widetilde{\kappa}_z u + \widetilde{\kappa}_y \rho & \alpha(u + \widetilde{\kappa}_x a) & \alpha(u - \widetilde{\kappa}_x a) \\
\widetilde{\kappa}_x v + \widetilde{\kappa}_z \rho & \widetilde{\kappa}_y v & \widetilde{\kappa}_z v - \widetilde{\kappa}_x \rho & \alpha(v + \widetilde{\kappa}_y a) & \alpha(v - \widetilde{\kappa}_y a) \\
\widetilde{\kappa}_x w - \widetilde{\kappa}_y \rho & \widetilde{\kappa}_y w + \widetilde{\kappa}_x \rho & \widetilde{\kappa}_z w & \alpha(w + \widetilde{\kappa}_z a) & \alpha(w - \widetilde{\kappa}_z a) \\
\left[\widetilde{\kappa}_x \phi^2/(\gamma-1) + \rho(\widetilde{\kappa}_z v - \widetilde{\kappa}_y w)\right] & \left[\widetilde{\kappa}_y \phi^2/(\gamma-1) + \rho(\widetilde{\kappa}_x w - \widetilde{\kappa}_z u)\right] & \left[\widetilde{\kappa}_z \phi^2/(\gamma-1) + \rho(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)\right] & \alpha\left[(\phi^2 + a^2)/(\gamma-1) + \widetilde{\theta} a\right] & \alpha\left[(\phi^2 + a^2)/(\gamma-1) - \widetilde{\theta} a\right]
\end{bmatrix} \qquad (A.10)$$

where

$$\alpha = \frac{\rho}{\sqrt{2}a}, \quad \widetilde{\kappa}_x = \frac{\kappa_x}{\kappa}, \quad \widetilde{\kappa}_y = \frac{\kappa_y}{\kappa}, \quad \widetilde{\kappa}_z = \frac{\kappa_z}{\kappa}, \quad \widetilde{\theta} = \frac{\theta}{\kappa}$$

The corresponding T_κ^{-1} is

$$T_\kappa^{-1} = \begin{bmatrix} \widetilde{\kappa}_x\left(1-\phi^2/a^2\right)-(\widetilde{\kappa}_z v-\widetilde{\kappa}_y w)/\rho & \widetilde{\kappa}_x(\gamma-1)u/a^2 \\ \widetilde{\kappa}_y\left(1-\phi^2/a^2\right)-(\widetilde{\kappa}_x w-\widetilde{\kappa}_z u)/\rho & \widetilde{\kappa}_y(\gamma-1)u/a^2-\widetilde{\kappa}_z/\rho \\ \widetilde{\kappa}_z\left(1-\phi^2/a^2\right)-(\widetilde{\kappa}_y u-\widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_z(\gamma-1)u/a^2+\widetilde{\kappa}_y/\rho \\ \beta(\phi^2-\widetilde{\theta}a) & -\beta[(\gamma-1)u-\widetilde{\kappa}_x a] \\ \beta(\phi^2+\widetilde{\theta}a) & -\beta[(\gamma-1)u+\widetilde{\kappa}_x a] \end{bmatrix.$$
$$\begin{bmatrix.} \widetilde{\kappa}_x(\gamma-1)v/a^2+\widetilde{\kappa}_z/\rho & \widetilde{\kappa}_x(\gamma-1)w/a^2-\widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x(\gamma-1)/a^2 \\ \widetilde{\kappa}_y(\gamma-1)v/a^2 & \widetilde{\kappa}_y(\gamma-1)w/a^2+\widetilde{\kappa}_x/\rho & -\widetilde{\kappa}_y(\gamma-1)/a^2 \\ \widetilde{\kappa}_z(\gamma-1)v/a^2-\widetilde{\kappa}_x/\rho & \widetilde{\kappa}_z(\gamma-1)w/a^2 & -\widetilde{\kappa}_z(\gamma-1)/a^2 \\ -\beta[(\gamma-1)v-\widetilde{\kappa}_y a] & -\beta[(\gamma-1)w-\widetilde{\kappa}_z a] & \beta(\gamma-1) \\ -\beta[(\gamma-1)v+\widetilde{\kappa}_y a] & -\beta[(\gamma-1)w+\widetilde{\kappa}_z a] & \beta(\gamma-1) \end{bmatrix} \qquad (A.11)$$

where

$$\beta = \frac{1}{\sqrt{2}\rho a}$$

Thomas H. Pulliam
Mail Stop T27B-1, NASA Ames Research Center
Moffett Field, California, 94035
Phone : 415 - 604 - 6417
email : pulliam@nas.nasa.gov
January, 1986