A Semi-Implicit Navier-Stokes Solver and its Application to a Study of Separated Flow about Blunt Delta Wings

by

Bernard Loyd

S.B. Aeronautics and Astronautics, Massachusetts Institute of Technology, 1985
S.M. Aeronautics and Astronautics, Massachusetts Institute of Technology, 1985

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy
in
Computational Fluid Dynamics
Department of Aeronautics and Astronautics
at the
Massachusetts Institute of Technology

February 1989

Signature of Author _____
Department of Aeronautics and Astronautics
10 February 1989

Certified by _____
Professor Earll M. Murman
Thesis Supervisor, Department of Aeronautics and Astronautics

Certified by _____
Professor Eugene E. Covert
Department of Aeronautics and Astronautics

Certified by _____
Dean Wesley L. Harris, Jr.
University of Connecticut at Storrs, School of Engineering

Accepted by _____
~~Professor~~ Harold Y. Wachman
Chairman, Department Graduate Committee

A Semi-Implicit Navier-Stokes Solver and its Application to a Study of Separated Flow about Blunt Delta Wings

by

Bernard Loyd

Submitted to the Department of Aeronautics and Astronautics on
February 1988
in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in Computational Fluid Dynamics

Abstract

A novel *semi-implicit* scheme for the Navier-Stokes equations is presented and evaluated. The semi-implicit scheme combines an implicit temporal integration in the body-normal directions with explicit temporal integrations in the streamwise and cross stream directions. Thus, advantages of both explicit and implicit schemes are retained in the semi-implicit scheme. Numerical stiffness due to disparate physical scales in the normal direction is eliminated, since stability of the algorithm depends only on relatively coarse streamwise and cross stream grid spacing, not on the typically fine normal spacing. Approximate factorization is unnecessary and only one matrix inversion per multi-stage time step is required. Computations show that while a explicit scheme employing multi-grid and residual smoothing and a fully implicit scheme are competitive for inviscid calculations, the semi-implicit scheme is superior for viscous flow calculations.

Efficiency of the semi-implicit scheme is exploited in a study of flow separation around delta wings with blunt leading edges. Three-dimensional laminar vortical flows over two 65° swept semi-infinite elliptical wings of thickness to chord ratio 1 : 11.55 and 1 : 20 at $M_\infty = 1.6$, $Re_L = 10^6$, and angles of attack of 4°, and 8°, and a 60° swept elliptical wing with $t/c = 1 : 11.55$ at $M_\infty = 1.4$, $Re_L = 2 \times 10^6$ and $\alpha = 14°$ are considered. In these flow cases, separation line locations are fixed not by a particular geometric factor (eg. sharp leading edge), but by interaction of physical and geometric factors. Solutions with the semi-implicit scheme are shown to be significantly more efficient than solutions with a corresponding explicit scheme. Two distinct leading edge separation processes are identified: separation due to shock-less flow recompression leeward of the leading edge expansion in the $t/c = 11.55$, $\alpha = 4°$ case and separation involving a leading edge shock in the remaining cases.

Thesis Supervisor: Earll M. Murman
Title: Professor of Aeronautics and Astronautics

Acknowledgements

The last five-and-a-half years or so of graduate work, two-thirds of it spent in connection with this dissertation, have been literally a whirlwind of activity. I've grown and learned so much, and it has prepared me well for what is to come. I want to thank the many individuals who have contributed to my growth and to the research activities leading to this document.

To Earll Murman, super guy and manager par excellence, thanks for everything and especially for your unwavering support! Thank you also to Gene Covert and Wes Harris for helping shape this project and thesis. Wes, you were always an inspiration. Thanks go to my thesis reader, Mark Drela, and a special thank you to two soon-to-be Ph.D.'s, Aga Goodsell and Steve Ruffin, for their friendship, and technical consultation and collaboration which contributed greatly to this work.

I want to thank all the members of the CFDL "gang" for the always stimulating technical discussions, late night Coffee House runs, and occasional rowdiness. To Teva Regule and, especially, Bob Bruen for keeping the computer beasties contained. Thanks go also to Bob Melnik and Frank Marconi at Grumman Aerospace Corporation for their assistance and stimulating technical discussions during my stay at Grumman.

Nights were long and sleep was short due to work done on this project, but also due to a number of "activities" that tried to turn into full-time occupations. I want to thank everyone connected with the MIT Men's Basketball program, especially my coaches, Fran O'Brien and Leo Osgood, for the many sweet memories. And I want to thank my colleagues and friends at the *Black Graduate Student Association*, especially Dean John Turner, and the *Black Graduate Coalition*.

Some special people inspired me along the way. This is to the enterprising spirit of St. Patrick's High School Class of '79, to Jeremiah Mendscole '79 for his confidence in me, and to our math and science guru extraordinaire, Brother Thomas, who continues to inspire generations of students. Jim McCune first taught me about vortices and suggested numerical fluid dynamics. A special thanks goes to Laverne Gibson, and to a pair of friends and collegues, Cathy Mavriplis and Patrick Hanley,

This research was partially supported by the Air Force Office of Scientific Research under contract AFOSR-87-0218 monitored by J. Wilson, the Office of Naval Research grant N00014-86-K-0288 monitored by S. Lekoudis, and a fellowship from the National Science Foundation. Additional support was provided by Grumman Aerospace Corporation. CRAY2 time was provided by the NAS facility at NASA Ames.

This dissertation is dedicated to my family: to Mamuschka and Holler and all of my crazy aunts, uncles, and cousins who put up with the quirks of my frantic schedule to whisk me off to beautiful Alpine peaks; to Pat and family, Darnell, and Darlene in California; to my sister; and, especially, to my mother who, by her example, showed us the meaning of love and courage.

To My Family

Contents

List of Figures

List of Tables

This page intentionally left blank.

Chapter 1

Introduction

1.1 Background

Flight vehicles have played an important role in human society for the better part of this century. Especially in the last fifty years, increases in our understanding of the technology of air transportation and the physics of fluid flow have contributed to a rapid decrease in the cost of air travel and an associated explosion of growth in transportation of humans and freight by air.

Acquisition of this knowledge, especially in the field of fluid dynamics, has been and continues to be a difficult process. While the physical laws governing the dynamics of a fluid were formulated in mathematical terms many years ago [Navier 1823] [Stokes 1845] and are well known today, analytical solutions to these differential relations for general flow cases have eluded generations of mathematicians and physicists. Moreover, the degree of mathematical complexity of the Navier-Stokes relations is such that the probability of finding meaningful analytical solutions in the future is minimal.

Coupled with this is the fact that flight testing as a means of advancing the science of aeronautics or even exploring the characteristics of a specific vehicle has always been a precarious and expensive undertaking. Instead, early aerodynamicists resorted to testing scale models of flight vehicles in earth bound wind tunnels and attempting

to analytically scale those results to obtain forces and moments on the actual vehicle at flight conditions. The successes of wind tunnel testing are tremendous and this technology continues to be a mainstay of aerodynamic design and development. However, wind tunnel testing has many limitations: extreme, even normal, flow conditions are difficult to simulate; flow data is difficult to acquire and subject to experimental error; and, the cost of manufacturing models and running the tests is high.

The numerical approach to investigating the aerodynamics of flight vehicles has recently emerged as an alternative. In this approach, the physical region of interest is divided into many small computational volumes in each of which the governing differential relations are approximated by discrete equations. Numerical constraints that model physical boundary conditions are imposed at boundary cells adjacent to the body and at the outer farfield surface of the computational domain. Computers are then used to numerically integrate the discrete equations from an assumed set of initial conditions to a set of invariant flow conditions (for steady flow). Forces and moments acting on a body can be deduced from this solution. For unsteady flow, the boundary conditions and numerical integration must be time accurate.

The computational approach has its own set of fundamental problems. First, because both consistency and accuracy of the numerical approximation to the governing equations are often difficult to determine, numerical algorithms must be validated, by comparison of their output to accepted "baseline" numerical results or to experimental data. Even getting to this point can be difficult since convergence of the temporal integration is not guaranteed.

Second, much like a wind tunnel, the computer model necessarily imposes artificial limits on the flow field. Although the limits of the computational domain can extend

much further away from the computational model than the walls of the wind tunnel extend from the scale model, inaccuracies in the flow conditions imposed at the farfield boundaries will affect the solution.

Finally, the cost of computer time limits the types of flow problems that can reasonably be modeled and the level of accuracy that can be achieved in a simulation. This is because some of the flow phenomena of interest occur at a physical scale much smaller than the overall dimensions of the test object. Since the computational grid must contain cells small enough to resolve these features, the total number of cells in a grid becomes large. Solutions with 100,000+ cell grids are not uncommon in 1988. The extreme computational cost of these solutions can be reduced by improving the efficiency of the numerical solver. This thesis reports one such attempt.

1.2 Overview

This thesis describes a new "semi-implicit" method for solving the Navier-Stokes equations and its application to calculation of flow about delta wings with rounded leading edges. In the semi-implicit approach the spatial discretization in the direction normal to a body is treated implicitly in time, while the streamwise and cross stream directions are treated explicitly. The approach eliminates the numerical "stiffness" that is introduced into the governing equations by the small grid spacing in the normal direction necessary to resolve viscous regions. The goals of the thesis are, first, to demonstrate the efficiency of the semi-implicit approach compared to conventional explicit and implicit approaches and, second, to investigate the phenomenon of flow separation around blunt leading edges.

This chapter serves as an introduction to the remainder of the thesis. Section 1.3

presents the Navier-Stokes equations and some derivative equation sets thereof. Section 1.4 describes some of the difficult issues associated with obtaining numerical solutions to these equations, and Section 1.5 gives a brief survey of existing solution algorithms. In the final sections, the aims of this research effort are discussed in more detail, the semi-implicit algorithm is described, and the remainder of the thesis is outlined.

1.3 Navier-Stokes and Derivative Equations

A continuum fluid is governed by the principles of classical mechanics and thermodynamics which require the conservation of mass, momentum and energy. Stated in mathematical terms, these principles yield a coupled system of five partial differential equations known as the Navier-Stokes equations[1]. Due to their complexity and mixed mathematical character, these equations have not yielded to analytical investigations. Numerical solution of the equations is possible, however, it can be prohibitively expensive. For this reason a number of simplified equation sets have been used in numerical simulations.

Solution schemes based on simplified equations usually require less computational grid resolution, fewer arithmetic operations on a given grid, and/or fewer iterations for convergence, than solutions based on the full Navier-Stokes equations. Thus, the cost of obtaining a solution is usually lower. In general, derivatives of the Navier-Stokes equations are based on the assumption that the overall flow field is not significantly affected by one or more physical characteristics of the fluid, thus allowing elimination of the relevant mathematical terms from the equations. Several simplified inviscid and

[1]Strictly speaking the Navier-Stokes equations are a mathematical formulation of the law of conservation of momentum only. This definition is broadened here to include the laws of mass and energy conservation.

viscous equation sets are discussed below.

1.3.1 Inviscid Flow Equations

The Euler equations are derived by eliminating viscosity from the Navier-Stokes equations, and thus should not be used to model flows in which viscous effects are important. Solution of these equations is relatively efficient because viscous terms need not be calculated and because the absence of shear layers reduces the number of grid points needed to resolve the flowfield.

The potential equation can be derived from the Euler equations with the additional assumption that the flow is irrotational. Solutions to this scalar equation can be obtained at relatively small cost and this equation is the basis of many aerodynamic design and analysis programs. However, the equation is valid only in flows where shock waves are weak and where the separation line can be determined *a priori* such that "equivalent" vorticity in the form of some kind of singularity can be introduced into the simulation. For these reasons, the potential equation is unsuited for simulation of most separated flows.

1.3.2 Conical Flow Equations

Conical flow equations can be used to calculate three dimensional flows if the body of interest is "conical" and no length scale exists in the radial direction from the apex. In these cases, the radial dimension can be eliminated from the governing equations, and numerical solution is relatively inexpensive.

Inviscid supersonic flow about conical bodies with attached shocks falls into this

category. Viscous flows have a radial length scale dependence due to the Reynolds number which precludes true conical flow. Nonetheless, a number of investigators (e.g. [McRae 76], [Thomas & Newsome 86], [Bluford 79], [Ruffin 87]) have justified use of the conical Navier-Stokes equations with the observation that the viscous regions in supersonic flow around conical bodies are dominated by the surrounding approximately conical inviscid flow [Cross 71]. The validity of this assumption is especially questionable in separated flows because of their inherent elliptical nature. This question will be explored in Chapter 6.

1.3.3 Parabolized Navier-Stokes Equations

An attractive approach to reducing the cost of computing steady viscous dominated flows is to alter the mixed mathematical character of the governing equations to something more amenable to numerical solution. The partially elliptic nature of the Navier-Stokes equations is eliminated in the "parabolized" Navier-Stokes equations by deleting the streamwise diffusion terms and prescribing the streamwise pressure gradient terms in the momentum equations while retaining the body-normal terms as unknowns. Solutions can then be obtained by "marching" downstream *sans* global iteration [Rudman & Rubin 68]. The boundary layer equations [Prandtl 04] assume, in addition, that momentum transfer in the direction normal to a body is negligible so that the normal momentum equation can be replaced by a condition of zero normal pressure gradient.

Solutions to the parabolized Navier-Stokes equations give a large improvement in computational efficiency compared to solution of the full equations, but require *a priori* knowledge of initial conditions and streamwise pressure distributions. If this information is not available, the parabolized equations must be coupled with a solution scheme that,

by solving the outer inviscid flow, provides it.

Implementation of this coupling for complex three dimensional geometries is difficult and the coupling often does not work well in flow cases in which there is a strong interaction between the viscous and inviscid flow regions [Davis & Rubin 80]. In any case, the parabolized equations are not valid in flows with substantial separation which, due to their upstream regions of influence, violate the assumption of parabolicity.

1.3.4 Use of the Simplified Equations

The price of simplifications is a loss of generality. The simplified systems do not have the capability to predict a full range of flow characteristics and may require *a priori* information about certain flow features. In addition, some simplified relations such as the potential equation may permit non-unique solutions [Salas et al. 83]. Thus, any use of a simplified system of equations must be justified by the particular characteristics of the flow being investigated.

None of the simplified systems described above retains the capability of modeling the massively separated flows that are of interest in this thesis. For this reason the Navier-Stokes equations are preferred. Some problems and issues related to their solution are examined below.

1.4 Numerical Solution – Problems and Issues

Several fundamental issues associated with the numerical approach to solving the Navier-Stokes equations – consistency and accuracy of the discretization, effect of boundary conditions, cost – were introduced in Section 1.1. In this section, four practical

problems that stem from these issues are discussed. They are turbulence modelling, spatial accuracy, grid generation, and numerical boundary conditions at farfield boundaries. These problems result result from a need to minimize computational expense and are inherent in a numerical modelling approach.

1.4.1 Modelling Turbulence and Transition to Turbulence

Although all flow calculations described in this thesis are laminar, flows of aeronautical interest have streamwise Reynolds numbers of $10^5 - 10^7$ and generally are turbulent. The present calculations are of intrinsic interest and represent a first step towards calculating turbulent flows. Because turbulence can greatly affect the character of the flow, the ability to accurately simulate it is essential. Thus, a central problem in solving the Navier-Stokes equations for high Reynolds number flow is that of modelling turbulence and transition to turbulence.

Present modelling capabilities fall short of that task. While, in principle, turbulent flow can be simulated by direct solution of the unsteady Navier-Stokes equations, in practice, the temporal and spatial scales associated with turbulent fluctuations in a flow are orders of magnitude smaller than its gross time and length scales. Resolving them would require commitment of excessive computational resources. [Chapman 79] and others estimate that a several order of magnitude improvement in speed and memory size of current supercomputers is necessary before flow at microscopic scales such as the Kolmogoroff scale can be directly simulated.

Instead, turbulent flows are usually treated by time-averaging the variables in the governing equations (e.g. [Anderson et al. 84]). The "Reynolds averaged" equations that are obtained are identical to the original equations except for the presence of ap-

parent stress gradients and heat flux terms that embody the effect of sub-scale turbulent motion. A solution can be sought on the scale of the laminar flow problem if the new terms are related to the mean flow variables through some *ad hoc* model of turbulence.

Numerous turbulence models ranging from algebraic (e.g. [Cebeci 74], [Baldwin & Lomax 78] and, recently, [Johnson & King 84]), to two-equation (e.g. $\overline{k}-\epsilon$ [Harlow & Nakayama 68]), and multi-equation Reynolds stress models such as those based on [Rotta 51], have been reported in the scientific literature. These models are at various levels of sophistication and empiricism. Some of them have become widely used. However, the models are all based on an incomplete understanding of the physical processes and are wed to simplifying assumptions about the nature of the flowfield. Most work well for a particular class of flows for which the free numerical parameters were "tuned", but none is successful for a large range of flow conditions.

Modelling the transition from laminar to turbulent flow is an equally difficult and poorly understood task. It is often ignored by numerical analysts who, for simplicity, assume an "instantaneous" transition to turbulent flow. The task is composed of two separate problems: predicting onset of transition and calculating turbulence quantities in the transition region. There are a number of numerical transition models, albeit with limited applicability, described in the scientific literature. The Orr-Sommerfeld equation (e.g. [Obremski et al. 69]) is often used to define the onset of transition. A number of approaches have been used to then formulate governing equations for the flow in the transition region (see, e.g. [Drela 86]).

1.4.2 Farfield Boundary Conditions

Farfield flow conditions must be imposed at the finite bounds of the computational domain. In the case of inviscid flows calculated with the unsteady Euler equations, the theory of characteristics mandates the type and number of boundary conditions to be specified. At a freestream boundary in three-dimensional subsonic flow, for example, four characteristics enter the domain from the exterior and one exits the domain from the interior. Characteristics theory decrees that, correspondingly, four characteristic variables be specified, with the fifth being extrapolated from the interior.

A similar well established boundary treatment does not exist for use in Navier-Stokes simulations. The characteristic treatment for the Euler equations is based on their hyperbolic nature in time and does not extend to time-dependent viscous flows, which have mixed parabolic/hyperbolic character. A number of investigations of the subject have been published (e.g. [Ababarnel et al. 86]), however research in this field is not yet conclusive.

Instead, investigators solving the Navier-Stokes equations have fashioned a variety of *ad hoc* treatments to find values of variables at farfield boundaries. For example, in this thesis a simple extrapolation of all variables from the interior of the domain to the outflow boundaries is used. While many of these treatments seem to work well, further research in this area is desirable.

1.4.3 Spatial Accuracy

Numerical solutions to the NS equations are useful only if they accurately represent the real flow field. Thus, for a steady flow, spatial accuracy is of prime importance[2]. One means of quantifying the accuracy of a solution is to examine the truncation error of the discretization. Unfortunately, even though the truncation error of a spatial discretization can usually be derived by Taylor series analysis of the discrete operator on idealized Cartesian grids, the actual truncation error for non-Cartesian grids is difficult to obtain and inevitably larger. The accuracy of several kinds of spatial operators in common use is discussed in general terms below.

Central difference approximations to the first and second derivative terms in the Navier-Stokes equations are in widespread use. These approximations are generally implemented in finite volume form thus ensuring conservation of mass, momentum, and energy and enabling accurate modeling of shock waves [Lax & Wendroff 60]. The approximations are termed "cell centered" or "vertex based" depending on whether the variables are stored at the centers or vertices of each grid cell.

In a cell centered discretization (e.g. [Beam & Warming 76], [Jameson et al. 81]), fluxes of mass, momentum, and energy from one cell to the next are calculated by surface integration of values at adjacent cell centers averaged to the cell faces. This results in a second order truncation error on Cartesian grids (see Appendix D). On stretched grids the accuracy of the first and second derivative approximations decreases to one and zero[3], respectively.

[2] In this discussion, the flow problem is assumed to be well posed and the discrete approximation consistent and stable [Richtmyer & Morton 67].

[3] Although the formal order of accuracy of the second derivative approximation is zero, numerical experiments in Appendix D show that the variation of truncation error with number of grid points is

Vertex based central difference approximations such as those of [Ni 81] and [Jameson 86] allow trapezoidal integration of the *inviscid* fluxes across cell faces. This portion of the integration promises higher accuracy. However, since the second derivative terms must still be computed at the vertices by central differences, vertex based Navier-Stokes schemes have the same formal order of accuracy as cell centered scheme

Artificial dissipation terms add to the truncation error in both cell centered and vertex based central difference schemes. These terms are needed to stabilize the otherwise neutrally stable temporal integration. Their detrimental effect on solution accuracy is difficult to quantify (see, eg., [Lindquist 88]), but can be substantial and should be monitored during a calculation.

Upwind difference discretizations for the inviscid terms (e.g. [Thomas & Newsome 86], [Roe 86]), have emerged as viable alternatives to central differencing[4]. These schemes are an attempt to model the physics of the fluid more closely by aligning the differences with the characteristic directions of the signal propagation. Upwind difference schemes require no artificial dissipation *per se* and can be constructed to contain only terms which have second and higher order truncation error on Cartesian grids[5]. Two disadvantages of upwind schemes are that a dissipative first order form often must be used in high gradient flow areas to obtain convergence, and that the leading truncation error term, which has the form of an artificial viscosity term [Pulliam 85], may introduce excessive dissipation in viscous regions [Hänel et al. 87].

A number of other approaches to obtaining accurate viscous solutions are reported

almost linear, ie. in practice the terms are nearly first order accurate

[4]Most upwind schemes use the same centered differences for the viscous terms as do central difference schemes.

[5]The *coefficients* of the truncation terms of a second order accurate upwind scheme are larger than those of a second order accurate centered discretization, as can be shown by a Taylor series analysis.

in the literature. Schemes based on triangular or, in three dimensions, tetrahedral unstructured grids promise less grid distortion at wing-body junctions and other regions of high surface curvature and, possibly, easier grid generation [Baker 87], [Löhner 88]. A review of some of this work is given by [Baker 88]. The accuracy of these discretizations, especially of second derivative terms, on these grids is difficult to assess. A number of researchers have combined unstructured grids in the external inviscid flow with structured (quadrilateral or hexahedral) grids in viscous zones.

In "box"-type schemes the governing equations are reduced to a coupled system of first order equations which is discretized with a vertex based spatial operator. This discretization allows a trapezoidal interation that gives grid independent second order accuracy. Box schemes have been applied to Prandtl's boundary layer equations (e.g. see [Keller 70], [Drela 83], [Loyd & Murman 86]) where they result in implicit block tridiagonal systems for the incompressible and compressible equations, of block size three and five, respectively, which can be solved efficiently by direct matrix inversion. [Allmaras & Giles 89] have applied features of this work to their algorithm for the two-dimensional thin shear layer (TSL) Navier-Stokes equations. These equations differ from the Prandtl boundary layer equations due to their retention of a normal momentum equation (inviscid terms only). Thus, they are intermediate in complextiy between the boundary layer equations and the thin layer Navier-Stokes equations. Discretization of these equations results in a block tridiagonal system of block size six.

Box discretizations of the full or thin layer Navier-Stokes equations would result in an implicit system with significantly more first-order equations and thus a larger block size than that of box discretizations of the boundary layer or TSL Navier-Stokes equations. This is due to the presence of numerous second order terms in the Navier-Stokes equations. These larger systems would present a challenging and costly inversion prob-

lem since the number of operations associated with the Gaussian elimination used to invert the component block matrices in the tridiagonal system increases with approximately the third power (n^3) of the block size. The author is not aware of any work in this area.

Other strategies for obtaining high accuracy include instituting adaptive gridding procedures and increasing the overall grid density. The former approach increases the number of grid cells in flow regions that contain high gradients *via* some kind of automated error minimizing procedure. Although these types of scheme can be complicated and may require significant computational overhead they have been used by a number of investigators (e.g. [Kallinderis & Baron 88] and [Davis & Dannenhoffer 89]). The latter grid refinement approach is simple but often very expensive. Regardless of which of the above approaches is employed, careful design of the inital grid can significantly reduce overall computational cost and improve accuracy.

1.4.4 Grid Generation

The discussion in Section 1.4.3 points out the vital role of the computational grid in determining spatial accuracy. The order of the truncation error of each of the schemes discussed, with the exception of the box scheme, is grid dependent, and the magnitude of the coefficients of the truncation terms in all cases depends on grid smoothness[6].

The difficulty of generating appropriate grids for complex configurations is, however, in itself a major impediment to the use of numerical techniques. Text books by [Anderson et al. 84] and [Thompson et al. 85] give a summary of available grid gener-

[6]The decrease in accuracy due to grid stretching or skewness is independent of whether discretization is performed in physical space or whether the physical space is transformed to computational coordinates before discretization (see e.g. [Peyret & Taylor 83] pp. 108-112 or, for 1–D analysis, Appendix D).

ation techniques. While many of the methods described in these references work well for simple geometries, none comes close to being applicable to general geometries. Most investigators attempting to model complex configurations resort to using a patchwork of grids generated by one or more methods. Because of this difficulty flow calculations are often limited to relatively simple problems.

The focus in this thesis is on the numerics and physics of a complicated flowfield around a simple conical geometry. Grids for the conical geometry are calculated by simply stacking a number of two-dimensional sections. Since the emphasis is on algorithm development it is important to review existing numerical algorithms for the Navier-Stokes equations.

1.5 Survey of Navier-Stokes Solution Methods

A large number of Navier-Stokes solution methods have been developed in the last 20 years. A primary focus of this research has been the development of efficient cost-effective algorithms. This brief survey describes some of those approaches that have been or could straightforwardly be applied to compressible three dimensional flows. The schemes are grouped by their use of explicit or implicit temporal integration.

1.5.1 Explicit Schemes

In an explicit scheme the unknown solution at each cell at time t^{n+1} depends on the known values of variables in surrounding cells at the previous time step t^n only (Figure 1.1). This decoupling of time levels t^n and t^{n+1} yields a temporal operator that can be readily combined with different spatial operators. It also results in a restriction

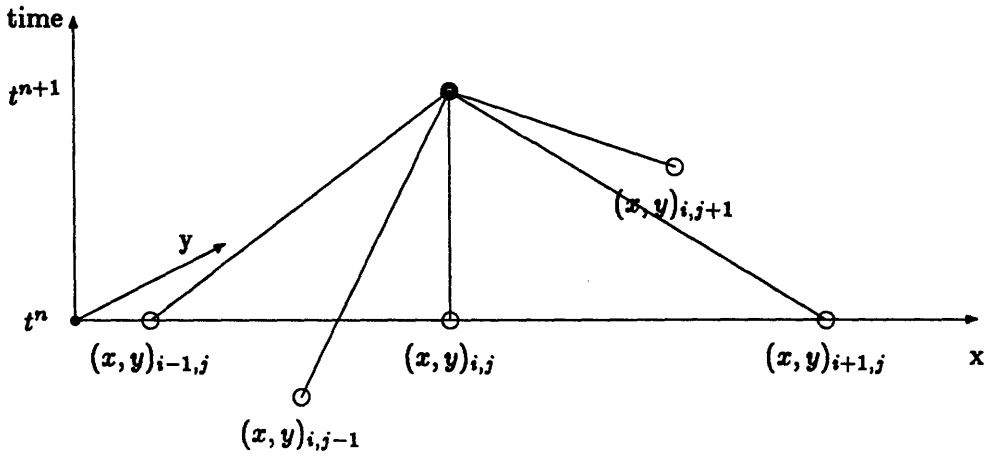

Figure 1.1: Difference Stencil in Explicit Scheme

of the maximum size of the time step by the Courant-Friedrichs-Levy (CFL) criterion, which requires that the numerical zone of dependence must include the physical zone of dependence.

The CFL criterion describes information propagation requirements in inviscid flow. Consider one spatial dimension in Figure 1.1. According to characteristics theory the solution of the inviscid flow described by the model wave equation, $u_{tt} - c^2 u_{xx} = 0$, at grid point x_i and time t^{n+1} depends on information in the region enclosed by the forward and backward sloping characteristics $u + c$ and $u - c$. Thus the CFL condition requires that for numerical stability the time step must be chosen so that all this information is used in the numerical prediction of u_i^{n+1}; that is,

$$\Delta t \leq \frac{\Delta x}{|u| + |c|}\,, \tag{1.1}$$

where $\Delta t = t^{n+1} - t^n$, $\Delta x = x_{i+1} - x_i$. and the vertical bars denote absolute values. Although this criterion is derived here for a model equation representing inviscid one-dimensional flow, numerical calculations show that it holds approximately for viscous

multi-dimensional flows (see also Section A.5 in Appendix A).

In practice, Equation (1.1) is a mild constraint if the spatial dimensions of the cells as represented by cell aspect ratios $AR_{xy} = \Delta x/\Delta y$ and $AR_{zy} = \Delta z/\Delta y$ are roughly uniform, that is AR_{xy}, $AR_{zy} \cong 1$. This usually is the case in Euler computations. In viscous flows, however, the small body normal grid spacing (Δy, say) necessary to resolve the viscous gradients near a body results in $AR_{xy}, AR_{zy} \gg 1$, which severely restricts the size of the time step. The resulting large number of iterations required for convergence is indicative of what is commonly referred to as a "stiff" system of equations.

Nonetheless, many investigators have had considerable success in using explicit Euler solvers as "stepping stones" to developing schemes for the Navier-Stokes equations, with the added viscous terms typically approximated by central differences. The Euler scheme of [Jameson et al. 81] combines a central difference spatial discretization and a blend of second and fourth order artificial viscosity with a multistage time integration to achieve a flexible numerical algorithm. Variants of this scheme have been extended to the Navier-Stokes equations by several investigators (eg. [Agarwal & Deese 85], [Martinelli 86], [Swanson & Turkel 85], [Müller & Rizzi 87], [Bussing 85], [Jayaram & Jameson 88]). The explicit two-stage scheme of [MacCormack 69] (e.g. [Deiwert 75]) and various versions of Lax-Wendroff integrators (e.g. [Ni 81], [Davis et al. 87], [Kallinderis & Baron 87]) have also been used. Because of the long computational times and correspondingly high CPU costs of explicit solvers, convergence acceleration techniques such as local time stepping, implicit residual smoothing, and multigrid are often used. With these schemes, viscous solutions in a wide range of Mach and Reynolds numbers have been obtained.

Explicit solution methods based on unstructured grids promise easier adaptability to complex configurations than those based on structured grids and have recently been applied to the Navier-Stokes equations (e.g. [Peraire et al. 87], [Kallinderis & Baron 88], [Davis & Dannenhoffer 89], [Mavriplis et al 89]). Kallinderis and Baron use a variant of the Lax-Wendroff scheme due to [Ni 81] with an adaptive gridding algorithm. [Peraire et al. 87] use a Taylor-Galerkin method for the spatial discretization with a four stage Runge-Kutta temporal integrator and an adaptive gridding algorithm.

A number of researchers have combined structured and unstructured grid methods in an attempt to capitalize on the strengths of each approach. A promising recent development is the structured/unstructured approach of [Nakahashi 88], who uses prismatic elements with triangular bases as grid cells. The triangular bases of the elements cover body surfaces in an unstructured manner, thus providing geometric flexibility. The sides of the elements are quadrilaterals, thus giving the grid structure in the body-normal direction. Because of this grid structure, multi-grid and implicit residual smoothing can be easily implemented in the body-normal direction to relax the severe Δt constraints, and the thin-layer approximation and Baldwin-Lomax turbulence model can be straightforwardly applied.

Despite this myriad of approaches, computation times for explicit solutions are usually long at Reynolds numbers of interest. With increasing Reynolds number and hence decreasing body-normal spacing the computation times become longer and implicit methods become competitive.

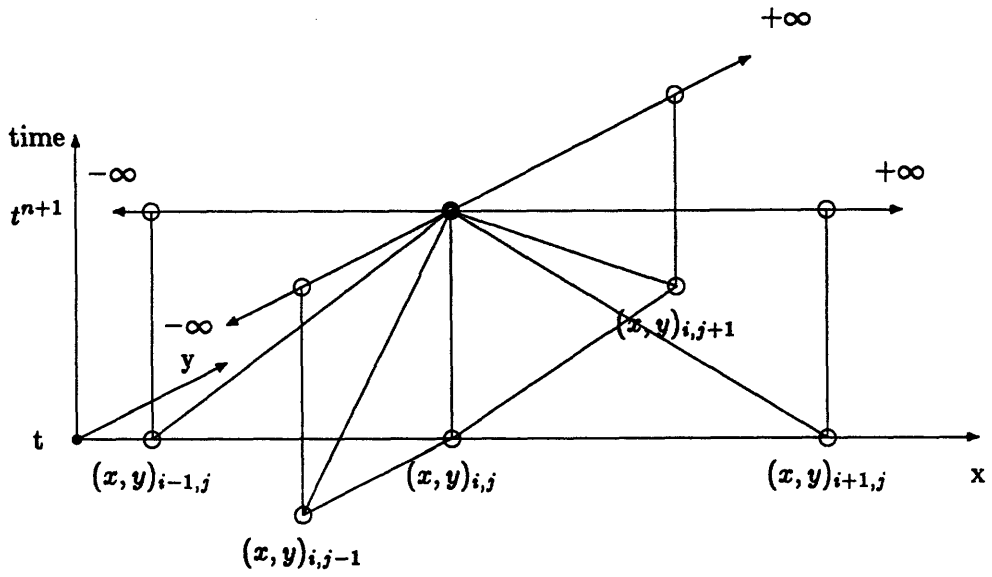

Figure 1.2: Schematic of Difference Stencil in Implicit Scheme

1.5.2 Implicit Schemes

Implicit schemes eliminate the numerical stiffness due to the small cells needed in shear layers by solving the equations implicitly in time. Unlike explicit schemes, which use only functions of the known solution at t^n to calculate the solution at t^{n+1}, implicit schemes base that calculation both on the known solution at t^n and the unknown solution at all (x, y, z) locations at t^{n+1}. Figure 1.2 shows a schematic at $(x, y)_{i,j}$ of a 2–D implicit difference stencil. Because the node points at t^{n+1} are linked, unknowns at any spatial location at t^{n+1} receive information from all unknowns in the x and y directions at that time level and from the known quantities at t^n.

The implicit discretization gives a coupled system of equations for the unknowns at t^{n+1} that, from linear stability analysis, is stable regardless of the size of the time step. Non-linear effects such as the movement of a shock wave over several cells in one time step may lead to instability even with implicit methods. Since direct solution of the

coupled system is prohibitively expensive, the system is usually approximately factored before inversion or solved via a relaxation algorithm. Both of these processes result in a limit in the size of the time step that can actually be used, although that limitation is usually much less restrictive than the CFL condition. Even if an approximate factorization or a relaxation algorithm is used, one implicit iteration is still considerably more expensive than an explicit iteration. Thus, implicit schemes are more efficient than explicit schemes only if the smaller number of iterations necessary for convergence of the implicit solution outweighs their higher computational cost per iteration.

In the alternating directing implicit (ADI) method (e.g. [McDonald & Briley 75], [Beam & Warming 76]) the coupled system is approximately factored along the spatial dimensions of the difference operators. Thus the full matrix is replaced by a product of three block tridiagonal matrices, one for each coordinate direction. The block tridiagonal nature of the new system is due to the use (or assumption of use of) three point central difference stencils in each coordinate direction[7]. Unfortunately, the approximate factorization introduces a slight instability in three dimensions [Dwoyer & Thomas 81] which must be suppressed, and the size of the optimal time step for convergence is limited by growth of the approximate factorization error. The ADI method has found widespread use in the work of [Steger 77], the zonal approaches of [Norton et al. 84], [Benek 87], [Krouthen 88], and [Holst et al. 87], and in the ARC2D and ARC3D codes of [Pulliam 84].

A number of implicit upwind methods have been extended from the Euler to the Navier-Stokes equations. These discretizations result in systems with increased diagonally dominance. Thus, the systems may be solved with ADI methods [Thomas et al. 87]

[7]The block tridiagonal system may be diagonalized for more efficient solution [Pulliam & Chaussee 81].

or relaxation [Thomas & Walters 85].

[Fuji & Obayashi 86] have introduced an LU-ADI method in which the ADI equations of [Beam & Warming 76] are decomposed into upper and lower bidiagonal matrices using the flux vector splitting technique. This ensures diagonal dominance of the implicit system. The explicit portion of the operator is the same as that used in [Beam & Warming 76].

While implicit schemes tend to require less CPU time than explicit schemes for high Reynolds number cases, the cost of solutions can still be large. Thus, new algorithm development is a priority and is the focus of this thesis.

1.6 The Semi-Implicit Approach

An alternative semi-implicit approach to solving the Navier-Stokes equations is described in this thesis. The semi-implicit algorithm combines elements of the explicit and implicit approaches by treating body normal terms in the equations implicitly, but the streamwise and cross stream terms explicitly. The concept is illustrated in Figure 1.3, which shows the schematic at $(x, y)_{i,j}$ of an semi-implicit difference stencil. It is a composite of the fully explicit and implicit stencils shown in Figures 1.1 and 1.2.

The semi-implicit concept is based on the observation that resolution for Navier-Stokes simulations is often necessary only in the direction normal to a body. At the ordinary differential equation level this implies that only boundary layer like viscous terms need to be retained in addition to the normal and cross stream momentum equations, leading to the so-called thin layer Navier Stokes equations. At the discrete level it implies that mesh cells will have a much smaller dimension in the body normal di-

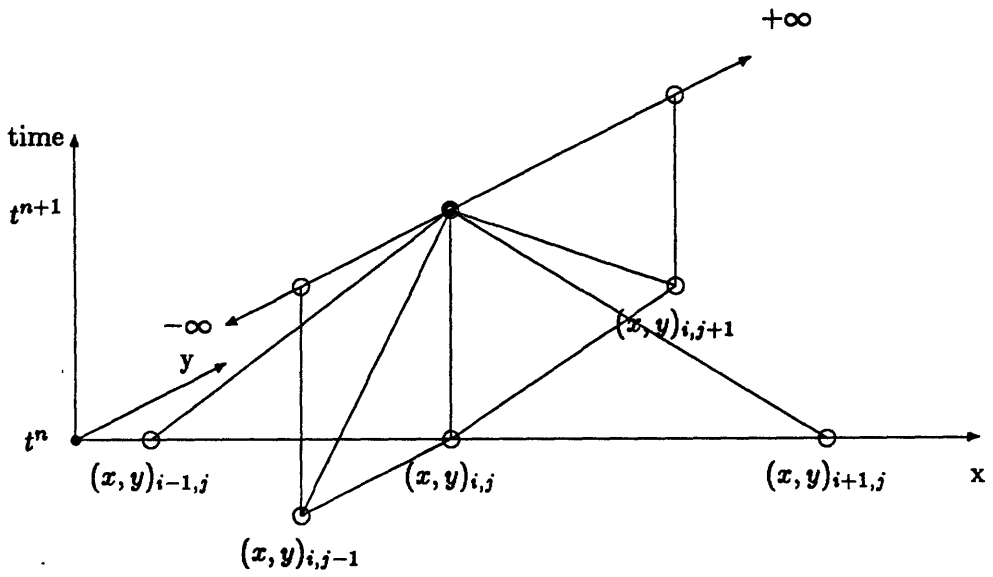

Figure 1.3: Schematic of Difference Stencil in Semi-Implicit Scheme

rection compared to the streamwise and cross stream directions. Thus, the stability restriction for explicit schemes in body fitted grid systems is usually dominated by the small normal spacing and the resulting numerical "stiffness" in the equations.

The scheme is an attempt to exploit the advantages of explicit and implicit schemes while minimizing their disadvantages. The semi-implicit algorithm eliminates the stability restriction due to the normal spacing by solving the flow equations implicitly in the normal direction. However, the solver is explicit in the tangential (flow) and cross stream direction, thereby avoiding factorization schemes, *CFL* limitations associated with the approximate factorization error, and a costly second or second and third block-tridiagonal inversions in two or three dimensions, respectively. The scheme is incorporated in a multi-stage time-stepping algorithm which, as a result of the implicit treatment of the body-normal terms, has a stability restriction that depends only on the grid spacing in the streamwise and cross stream directions.

A related concept was developed by [Rizk & Chausee 83] and used subsequently by [Müller & Rues 86] and [Riedelbauch & Müller 87]. In their approach the block tridiagonal matrices corresponding to the streamwise and cross stream directions of a Beam & Warming solver were simply eliminated. This produces a "forward time centered space" algorithm in the explicit directions that can easily be shown to be unconditionally unstable for the inviscid equations. Nevertheless, these authors report stable calculations.

1.7 Thesis Outline

Roughly the first half of this thesis (Chapters 1 – 5) presents a discussion of the characteristics and efficiency of a semi-implicit algorithm as compared to two popular conventional schemes. Chapter 2 presents the Navier-Stokes equations and discusses implementation of the thin layer approximation. Chapter 3 describes the finite volume discretization of the spatial derivatives, and Chapter 4 introduces the semi-implicit temporal integration. Stability of the scheme is shown to be dependent on the spacing in the explicit direction(s) only. In Chapter 5 the semi-implicit scheme is evaluated by comparison of two dimensional results with those obtained from fully explicit (with multigrid and residual smoothing) and implicit schemes.

The second half of this thesis describes application of the semi-implicit scheme to the calculation of separated flow about delta wings with rounded leading edges. Chapter 6 describes the geometry and boundary conditions of the flow case, and validation of the numerical algorithm. Chapter 7 displays characteristic features of the flow about delta wings with rounded leading edges, and a discussion of specific features of flow separation in given in Chapter 8. Conclusions and recommendations for further research are found

in Chapter 9.

Chapter 2

3–D Navier-Stokes Equations

The Navier-Stokes equations are a mathematical formulation of the laws of conservation of mass, momentum, and energy applied to a continuum fluid. Since mass and energy are scalar quantities and momentum is a vector quantity this gives in three dimensions a system of five equations. These five equations contain eight unknowns: the thermodynamic variables density, pressure, and temperature, three velocity components, the transport coefficient of viscosity, and the Prandtl number. A sixth equation is obtained from the equation of state which relates the thermodynamic properties of the fluid. The viscosity μ is obtained from Sutherland's formula and the Prandtl number is set to a constant, thus closing the system of equations.

The Navier-Stokes equations may be derived by applying the conservation laws to an arbitrary control volume. To conserve mass, the rate of mass flux passing into the control volume must equal the time rate change of mass in the control volume. Conservation of momentum requires that the rate of momentum gained by convection through the control surface, plus surface and body forces must equal the rate of increase of momentum within the control volume. Finally, conservation of energy requires that energy convection and heat conduction at the surfaces and addition through external agents balance the rate of growth of total energy.

The three-dimensional Navier–Stokes equations integrated in Cartesian coordinates

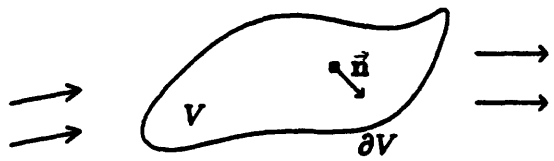

Figure 2.1: Control Volume

over a control volume V with boundary ∂V (see Fig. 2.1) are:

$$\frac{\partial}{\partial t}\iiint_V \mathbf{W}\, dV + \iint_{\partial V} \vec{\mathbf{F}}(\mathbf{W}) \cdot \vec{\mathbf{n}} dS = 0. \tag{2.1}$$

The vector of state variables is $\mathbf{W} = (\rho \;\; \rho u \;\; \rho v \;\; \rho w \;\; \rho E)^T$, where ρ is the density, u, v, w are x, y, and z components of velocity, and E is energy per unit mass. The flux vector $\vec{\mathbf{F}}$ is composed of inviscid and viscous parts

$$\vec{\mathbf{F}} = (\mathbf{F}_I - \mathbf{F}_V)\vec{i} + (\mathbf{G}_I - \mathbf{G}_V)\vec{j} + (\mathbf{H}_I - \mathbf{H}_V)\vec{k} \tag{2.2}$$

in the x, y, and z coordinate directions denoted by the unit vectors $\vec{i}$, $\vec{j}$, and $\vec{k}$, respectively.

The inviscid fluxes are

$$\mathbf{F}_I = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \rho uH \end{pmatrix}, \quad \mathbf{G}_I = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ \rho vH \end{pmatrix}, \quad \mathbf{H}_I = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ \rho wH \end{pmatrix}, \tag{2.3}$$

where H is the stagnation enthalpy, $H = E + P/\rho$, and P is the static pressure.

In a viscous fluid the rate of increase of the state variables in a control volume will also be affected by shearing stresses and heat conduction on the surface of the volume.

These viscous fluxes are given by

$$\mathbf{F}_V = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{pmatrix}, \quad \mathbf{G}_V = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \end{pmatrix}, \tag{2.4}$$

$$\text{and} \qquad \mathbf{H}_V = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{pmatrix}.$$

Components of the viscous stress tensor are,

$$\begin{aligned} \tau_{xx} &= \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) & \tau_{xy} &= \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \tau_{yy} &= \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) & \tau_{xz} &= \tau_{zx} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \tau_{zz} &= \frac{2}{3}\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) & \tau_{yz} &= \tau_{zy} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right). \end{aligned} \tag{2.5}$$

The heat flux terms are

$$q_x = -k\frac{\partial T}{\partial x} \qquad q_y = -k\frac{\partial T}{\partial y} \qquad q_z = -k\frac{\partial T}{\partial z}, \tag{2.6}$$

or, using the perfect gas relation $H = C_pT + .5(u^2+v^2+w^2)$ to eliminate the temperature T, and introducing the Prandtl number $Pr = \frac{\mu C_p}{k}$,

$$\begin{aligned} q_x &= -\frac{\mu}{Pr}\left(\frac{\partial H}{\partial x} - u\frac{\partial u}{\partial x} - v\frac{\partial v}{\partial x} - w\frac{\partial w}{\partial x}\right), \\ q_y &= -\frac{\mu}{Pr}\left(\frac{\partial H}{\partial y} - u\frac{\partial u}{\partial y} - v\frac{\partial v}{\partial y} - w\frac{\partial w}{\partial y}\right), \\ q_z &= -\frac{\mu}{Pr}\left(\frac{\partial H}{\partial z} - u\frac{\partial u}{\partial z} - v\frac{\partial v}{\partial z} - w\frac{\partial w}{\partial z}\right). \end{aligned} \tag{2.7}$$

The value of the Prandtl number Pr was taken as 0.72 for the laminar flow solutions presented in this thesis.

The coefficient of viscosity is related to the local enthalpy by the Sutherland formula [Cebeci 74]

$$\frac{\mu}{\mu_\infty} = \left(\frac{h}{h_\infty}\right)^{3/2} \frac{h_\infty + h_1}{h + h_1} , \tag{2.8}$$

where h_1 is a dimensional constant (= 111421 in *SI* units), and ∞ refers to free stream conditions. This relation closes the system of equations.

Equations 2.1 are written with the assumption that body forces such as gravity or electromagnetic forces are negligible and that there are no internal heat or mass sources. Also, the Stokes hypothesis is assumed to hold for the viscous terms. The form of (2.1) indicates that the change of the state vector **W** with time in a given volume is due to a net vector flux of **F** into that volume. Although the steady solutions to these equations are sought, one retains the unsteady terms to allow a time-like integration to steady state. There is no guarantee that steady solutions exist and/or are unique.

2.0.1 Nondimensionalization

Nondimensionalization of variables in the governing equations allows characteristic flow parameters such as the Mach number and Reynolds number to be identified and varied independently. The free stream values of density, speed of sound, and viscosity ρ_∞, a_∞, μ_∞, and a characteristic length c were chosen as reference values. Introducing these quantities gives the nondimensional variables:

$$\begin{array}{lll}
x' = \dfrac{x}{c} & y' = \dfrac{y}{c} & z' = \dfrac{z}{c} \\
u' = \dfrac{u}{a_\infty} & v' = \dfrac{v}{a_\infty} & w' = \dfrac{w}{a_\infty} \\
\rho' = \dfrac{\rho}{\rho_\infty} & p' = \dfrac{p}{\rho_\infty a_\infty^2} & t' = \dfrac{t}{c/a_\infty} \\
\mu' = \dfrac{\mu}{\mu_\infty}\dfrac{M_\infty}{Re_\infty} & E' = \dfrac{E}{a_\infty^2} & H' = \dfrac{H}{a_\infty^2} .
\end{array} \tag{2.9}$$

The forms of the nondimensional governing equations, equation of state, and viscosity model are identical to that of the dimensional equations. The free stream vector in terms of nondimensional variables is

$$\mathbf{W}_\infty = \begin{pmatrix} 1 \\ M_\infty \cos\alpha \\ 0 \\ M_\infty \sin\alpha \\ \dfrac{1}{\gamma(\gamma-1)} + \dfrac{M_\infty^2}{2} \end{pmatrix}, \tag{2.10}$$

where α is the angle of attack and the yaw angle is assumed zero. The free stream pressure is $p_\infty = 1/\gamma$. For convenience the primes are dropped and reference is henceforth made to the nondimensional variables only.

2.1 Thin Layer Approximation

The viscous terms in the governing equations can be simplified if one asssumes that gradients of viscous stress and heat flux in directions parallel to the body are negligible compared to those gradients in the direction normal to the body. All viscous terms that contain derivatives parallel to the body surface can then be eliminated, giving the thin layer form of the Navier-Stokes equations [Baldwin & Lomax 78]. In a real flow this assumption holds only in attached, high Reynolds number regimes. However, in a numerical solution the assumption is convenient, regardless of Reynolds number or whether the flow is attached or separated, if the computational grid does not adequately resolve gradients of viscous stress and heat flux in the streamwise or cross stream directions. The latter justification of the thin layer approximation is usually appropriate since computer memory restrictions and high *CPU* costs limit the total number of cells that can be employed in a computation.

For a computational grid with a given number of grid cells, grid surfaces are typically clustered in the normal direction in order to resolve the rapid flow changes through the boundary layer. This strategy is appropriate since quantities such as the skin friction can be predicted accurately only if the boundary layer is well resolved. However, the strategy leaves few grid surfaces available to resolve gradients in the cross stream or streamwise directions.

Because the thin layer form of the Navier Stokes equations retains the viscous terms in the direction normal to a body only, and these terms are discretized along grid lines, the computational grid must be constructed such that it contains a family of lines that is body normal. In addition, the viscous stresses must be transformed from the Cartesian x, y, z system to body normal ξ, η, and ζ coordinates, since in general no Cartesian coordinate direction corresponds to the normal direction η. This transformation is described below.

Derivative terms with respect to x, y, and z may be expressed in ξ, η, and ζ coordinates via the chain rule of differentiation:

$$\begin{aligned}
\frac{\partial}{\partial x} &= \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta} = \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial y} &= \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial}{\partial \zeta} = \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \\
\frac{\partial}{\partial z} &= \frac{\partial \xi}{\partial z}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial z}\frac{\partial}{\partial \zeta} = \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta} .
\end{aligned} \tag{2.11}$$

By the thin layer assumption, derivatives in the downstream ξ and cross stream ζ directions are small compared to derivatives in the η direction; that is,

$$\frac{\partial}{\partial \eta} >> \frac{\partial}{\partial \xi}, \frac{\partial}{\partial \zeta}, \tag{2.12}$$

so that all ξ and ζ derivative terms may be dropped from (2.11) to obtain

$$\frac{\partial}{\partial x} \cong \eta_x \frac{\partial}{\partial \eta}, \qquad \frac{\partial}{\partial y} \cong \eta_y \frac{\partial}{\partial \eta}, \text{ and } \qquad \frac{\partial}{\partial z} \cong \eta_z \frac{\partial}{\partial \eta} . \tag{2.13}$$

Inserting (2.13) into (2.5) and (2.7) gives the thin layer shear stresses

$$\begin{aligned}
\tau_{xx} &\cong \frac{2}{3}\mu\left(2\eta_x\frac{\partial u}{\partial \eta} - \eta_y\frac{\partial v}{\partial \eta} - \eta_z\frac{\partial w}{\partial \eta}\right) & \tau_{xy} = \tau_{yx} &\cong \mu\left(\eta_y\frac{\partial u}{\partial \eta} + \eta_x\frac{\partial v}{\partial \eta}\right) \\
\tau_{yy} &\cong \frac{2}{3}\mu\left(2\eta_y\frac{\partial v}{\partial \eta} - \eta_x\frac{\partial u}{\partial \eta} - \eta_z\frac{\partial w}{\partial \eta}\right) & \tau_{xz} = \tau_{zx} &\cong \mu\left(\eta_z\frac{\partial u}{\partial \eta} + \eta_x\frac{\partial w}{\partial \eta}\right) \\
\tau_{zz} &\cong \frac{2}{3}\mu\left(2\eta_z\frac{\partial w}{\partial \eta} - \eta_x\frac{\partial u}{\partial \eta} - \eta_y\frac{\partial v}{\partial \eta}\right) & \tau_{yz} = \tau_{zy} &\cong \mu\left(\eta_z\frac{\partial v}{\partial \eta} + \eta_y\frac{\partial w}{\partial \eta}\right),
\end{aligned} \tag{2.14}$$

and heat fluxes

$$\begin{aligned}
q_x &\cong -\frac{\mu}{Pr}\left(\eta_x\frac{\partial H}{\partial \eta} - u\eta_x\frac{\partial u}{\partial \eta} - v\eta_x\frac{\partial v}{\partial \eta} - w\eta_x\frac{\partial w}{\partial \eta}\right) \\
q_y &\cong -\frac{\mu}{Pr}\left(\eta_y\frac{\partial H}{\partial \eta} - u\eta_y\frac{\partial u}{\partial \eta} - v\eta_y\frac{\partial v}{\partial \eta} - w\eta_y\frac{\partial w}{\partial \eta}\right) \\
q_z &\cong -\frac{\mu}{Pr}\left(\eta_z\frac{\partial H}{\partial \eta} - u\eta_z\frac{\partial u}{\partial \eta} - v\eta_z\frac{\partial v}{\partial \eta} - w\eta_z\frac{\partial w}{\partial \eta}\right).
\end{aligned} \tag{2.15}$$

While these expressions seem no simpler than the original stress tensor and heat flux terms, they in fact allow considerable computational savings, since derivatives in this approximation need be evaluated only on the two faces of each volume that are parallel to the body. Evaluating the thin layer terms on the other four faces is unnecessary since, by assumption, face-to-face changes in the streamwise and cross stream viscous terms are small.

While the thin layer simplifications result in a saving of *CPU* time and in memory required, they do not change the character of the governing equations. Unlike the boundary layer equations, the normal momentum equation is retained. Unlike "parabolized" sets of the Navier-Stokes equations, the pressure gradient in the streamwise momentum equation is retained and no assumption is made about the pressure variation in the streamwise direction.

2.2 Solutions to the Navier-Stokes Equations

While the Navier-Stokes equations were formulated more than a century ago, solutions to them have been obtained only for very special cases. The complexity of th.. nonlinear coupled system of equations and their mixed mathematical character – they are hyperbolic-parabolic for unsteady flow and elliptic-parabolic for steady flow – make future analytical solutions to general problems unlikely. The next two chapters explore numerical means of efficiently solving the equations for cases of real interest.

Chapter 3

Spatial Discretization

The governing equations must be discretized before numerical solution can be attempted. Discretization involves identification of a computational region around the body of interest, division of that region into many cells or volumes, and approximation of the governing equations on each of these control volumes by discrete representations. Boundary conditions representing conditions in the physical flow field must be imposed at the outer surface of the computational region and at body surfaces. An initial state for each of the finite cells is assumed and the approximation in each is integrated forward in time. The final solution is obtained when the temporal changes of the values of the state vector become appropriately small.

A number of problems complicate the process of discretization. Some of these are: generation of an "acceptable" computational grid around the body of interest, stability of the temporal integration, efficiency of the numerical calculation, accuracy of the spatial approximation, and, for time-accurate calculations, accuracy of the temporal integration. Some of these problems will be addressed in the remainder of this thesis.

Discretization may proceed in two steps, spatial and temporal. Even though the two steps are connected due to dependence of the stability of the temporal integration on the form of the spatial operator, it is convenient to consider them separately. This chapter presents the spatial discretization, treatment of flow conditions at grid boundaries, and added artificial viscosity. The next chapter gives an explicit temporal

discretization used in many conventional schemes and derives from it a semi-implicit temporal discretization. Appendix A combines temporal and spatial operators in an analysis of the temporal stability.

3.1 Spatial Difference Operators

The area integrals in (2.1) represent the flux of mass, momentum, and energy into and out of a given control volume. While the equations hold regardless of the shape of this volume, the formulation in this thesis assumes that the flow area of interest is divided *a priori* into hexahedral volumes organized in a logical $(I : I_{max}, J : J_{max}, K : K_{max})$ matrix, commonly referred to as a "structured" grid, and that the vectors of state variables are stored at the centers of each of these volumes.

The discretization scheme assumes that the flux at each face of a volume is constant on that face. The area integrals (2.1) are then the sums of the products of the flux vector $\vec{\mathbf{F}}$ with the projected surface areas

$$\begin{aligned} \iint_{\partial V} \vec{\mathbf{F}}(\mathbf{W}) \cdot \hat{\mathbf{n}} dS &= \sum_{f=1}^{6} \vec{\mathbf{F}}(\mathbf{W})_f \cdot \vec{S}_f \\ &= \sum_{f=1}^{6} (\mathbf{F}S_x + \mathbf{G}S_y + \mathbf{H}S_z)_f \end{aligned} \tag{3.1}$$

over the six cell faces (see Figure 3.1). In particular, the flux out of a cell in the y direction is given by

$$\sum_{f=1}^{6} (\mathbf{G}S_y)_f = (\mathbf{G}S_y)_1 + (\mathbf{G}S_y)_2 + (\mathbf{G}S_y)_3 + (\mathbf{G}S_y)_4 + (\mathbf{G}S_y)_5 + (\mathbf{G}S_y)_6 \,, \tag{3.2}$$

where S_y is the projected area on the xz plane of a cell face. In the x and z directions the summations are obtained by replacing $(\mathbf{G}S_y)$ by $(\mathbf{F}S_x)$ and $(\mathbf{H}S_z)$, respectively. Because it is written in finite volume form and the flux out of any given cell goes directly

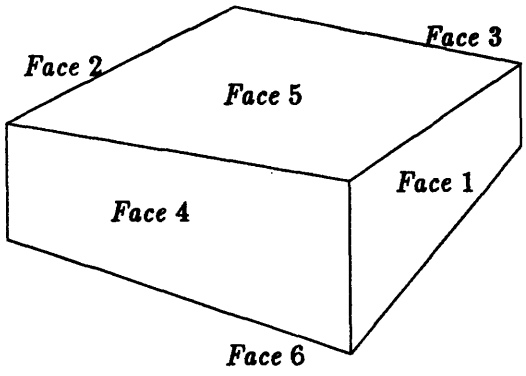

Figure 3.1: Cell Nomenclature (Body Surface is Tangent to *Face* 6)

into neighboring cells this cell-centered discretization scheme identically conserves mass, momentum, and energy.

For the inviscid portions of the flux vectors (2.3) a simple average of the values at the cell centers of the two adjoining cells is used to calculate the face centered flux:

$$\begin{aligned} \mathbf{G}_1 &= .5(\mathbf{G}_{I_{i+1,j,k}} + \mathbf{G}_{I_{i,j,k}}) & \mathbf{G}_2 &= .5(\mathbf{G}_{I_{i-1,j,k}} + \mathbf{G}_{I_{i,j,k}}) \\ \mathbf{G}_3 &= .5(\mathbf{G}_{I_{i,j+1,k}} + \mathbf{G}_{I_{i,j,k}}) & \mathbf{G}_4 &= .5(\mathbf{G}_{I_{i,j-1,k}} + \mathbf{G}_{I_{i,j,k}}) \\ \mathbf{G}_5 &= .5(\mathbf{G}_{I_{i,j,k+1}} + \mathbf{G}_{I_{i,j,k}}) & \mathbf{G}_6 &= .5(\mathbf{G}_{I_{i,j,k-1}} + \mathbf{G}_{I_{i,j,k}}) \, . \end{aligned} \tag{3.3}$$

The values of the viscous terms are evaluated in finite difference form as either

$$\frac{\partial(\,)}{\partial\eta} = \frac{(\,)_{j+1} - (\,)_j}{\Delta\eta} \quad \text{or} \quad \frac{\partial(\,)}{\partial\eta} = \frac{(\,)_j - (\,)_{j-1}}{\Delta\eta} \, , \tag{3.4}$$

depending on whether the viscous flux at *Face 5* or at *Face 6* is desired. The definitions (3.3) and (3.4) are convenient because they give a compact three point difference stencil for the thin layer Navier Stokes equations. The appropriate prefix η_x, η_y, or η_z is defined by centered differences (see Appendix E). Consistent with the thin layer approximation (2.12), the viscous fluxes across *Faces* 1 and 2, and 3 and 4 are discarded since these correspond to changes in the ξ and ζ directions respectively, which are, by assumption, small compared to changes in the direction normal to the body.

3.1.1 Accuracy of the Discretization

The discretizations (3.1 - 3.4) reduce to second order accurate central difference approximations for first derivatives (inviscid terms) and second derivatives (viscous terms) on uniform grids. However, that order of accuracy is not maintained on arbitrary grids. A simple analysis of the accuracy of these and several alternate operators on stretched grids is given in Appendix D. It is easy to show that the *maximum* order of accuracy for a three point approximation of first and second derivatives on non-uniform grids is two and one, respectively, regardless of whether discretization is performed in physical or mapped (transformed) coordinates[1].

The discretization for inviscid terms described in the previous section has a first order truncation error but is conservative. While three point discretizations of inviscid terms with second order truncation error can be constructed, these discretizations are necessarily non-conservative.

The finite volume approximation to the viscous terms is not formally first order accurate even on a stretched one-dimensional grid. However, computational results show that the magnitude of its truncation error is almost identical to that of the non-conservative centered finite difference approximation. Thus, in practice, the scheme is first order accurate. The analysis also shows that it is important that grids vary smoothly to limit degradation in accuracy.

Higher order accurate difference approximations to 1^{st} and 2^{nd} order derivatives using a cell-centered approach can be constructed by using more points in the difference stencil. This is usually not practical since the higher order of accuracy is achievable

[1] While discretization in computational space is nominally second order accurate when $\Delta\xi = \Delta\eta = \Delta\varsigma$, the *numerical* mapping of variables to and from that space is not.

only on the grid for which the approximation is designed. For example, a higher order accurate representation can be designed for a Cartesian grid with a given stretching. However, when the stretching varies non-analytically or when the grid becomes skewed (due, say, to a third dimension) the difference approximation reverts to lower order. Also, each additional point in the difference stencil increases the bandwidth of the matrix that must be solved in an implicit sweep, thus increasing the computational cost of inversion.

3.2 Boundary Conditions

Flow conditions at grid boundaries must be imposed to fully specify any flow problem. The boundary conditions are implemented in the computer code by creating a surface of "ghost" cells outside the boundary and setting the state vector in those cells to such values that Equation 3.3 gives the correct values at boundary faces. For example, $W_{ghost_{i,k}}$ is set to $-W_{i,1,k}$ if a body is located adjacent to the $j = 1$ surface of cells and $W = 0$ is desired on those boundary faces. In general, $W_{ghost_{i,k}} = 2W_{spec} - W_{i,1,k}$ where W_{spec} is the value to be specified at the boundary. Different kinds of boundary conditions are applied at solid walls, in the far field, and at symmetry planes. They are discussed below.

3.2.1 Solid Wall

At solid boundaries it is experimentally observed that the flow of a continuum fluid obeys a "no-slip" condition of zero velocity: $u = v = w = 0$. The value of the pressure in the ghost cells is calculated by assuming $\partial p/\partial n = 0$, consistent with classical boundary layer treatment. Isothermal or adiabatic boundary conditions are implemented by

setting the appropriate values of total enthalpy in the ghost cells.

3.2.2 Far Field Boundary Conditions

Two kinds of boundary conditions are used at far field boundaries of the grid, depending on whether the flow through the boundary contains viscous-dominated or inviscid flow. For inviscid flow a Riemann invariant boundary condition treatment is used following [Jameson & Baker 83]. These Riemann conditions are based on the theory of characteristics which describes the information transfer in the one dimensional flow normal to a boundary. For subsonic flow, the incoming and outgoing Riemann invariants are

$$R_- = \vec{W}_\infty \cdot \vec{n} - 2\frac{a_\infty}{\gamma - 1} = u_n - 2\frac{a}{\gamma - 1} \tag{3.5}$$

and

$$R_+ = \vec{W}_{ex} \cdot \vec{n} + 2\frac{a_{ex}}{\gamma - 1} = u_n + 2\frac{a}{\gamma - 1} \tag{3.6}$$

where the subscript ∞ identifies freestream values and ex denotes values extrapolated from the first interior cell. The vector $\vec{n}$ is the outward pointing normal to the cell.

The normal velocity and speed of sound are obtained by adding and subtracting the invariants,

$$u_n = \frac{1}{2}(R_- + R_+) \,, \tag{3.7}$$

and

$$a = \frac{\gamma - 1}{4}(R_- + R_+) \,. \tag{3.8}$$

The velocity vector can then be calculated for an outflow boundary by extrapolating the tangential component from the interior solution,

$$\vec{W} = \vec{W}_{ex} + (u_n - \vec{W}_{ex} \cdot \vec{n})\vec{n} \tag{3.9}$$

and, for an inflow boundary by specifying it from the freestream,

$$W = W_\infty + (u_n - W_\infty \cdot \vec{n})\vec{n} \ . \tag{3.10}$$

Similarly, the entropy at an outflow boundary is extrapolated

$$s = s_{ex} = \frac{p_{ex}}{\rho_{ex}^\gamma} \ , \tag{3.11}$$

and specified at an inflow boundary

$$s = s_\infty = 1/\gamma \ . \tag{3.12}$$

For supersonic flow, all characteristics point in the direction of the flow and all quantities are correspondingly extrapolated from the interior at an outflow boundary and set to freestream at an inflow boundary.

The Riemann invariant treatment is not appropriate for boundaries that cut through regions of viscous-dominated flow. Boundaries downstream of solid bodies fall in this category since they usually contain wakes or boundary layers. At these boundaries all flow variables were extrapolated from the interior. This treatment is consistent with the parabolic nature of most boundary layer flows, but may lead to some wave reflection in regions of the outflow that contain subsonic inviscid flow. Numerical models that are consistent with the dual character of the outflow (e.g. [Ababarnel et al. 86]) are a matter of current research and were not considered for this investigation.

3.2.3 Symmetry Planes

Several of the test cases shown in later chapters will make use of one or more symmetry conditions. These are grid planes at which the exterior flow is assumed to be the mirror image of the interior flow. Symmetry boundary conditions are easily

implemented by setting all variables in the plane of ghost cells equal to the corresponding variables at the first interior grid plane, except for the component of the velocity normal to the boundary which is reflected, $\vec{W}_{ghost} \cdot \vec{n} = -\vec{W}_{interior} \cdot \vec{n}$.

3.3 Artificial Viscosity

Artificial viscosity is added to the physical fluxes to damp out non-physical odd-even oscillations and to stabilize the integration in areas of discontinuos flows. The discretization for the inviscid terms described above allows odd-even decoupling of flux values at adjacent points. This decoupling may lead to aliasing errors that inhibit convergence of the temporal integration. Also, the truncation terms of central difference-type discretizations on a Cartesian grid are not dissipative and, in inviscid regions, one finds that discontinuities in the flow field may cause divergence of the algorithm. For both these reasons artificial viscosity is added to the solution. While its presence is essential, it is important that the artificial viscosity terms be kept as small as possible to minimize degradation of solution accuracy.

A blend of nonlinear second and fourth difference terms that has been established as particularly effective in damping out nonphysical oscillations [Jameson 83] is used in this thesis. Consider a 3-D damping operator:

$$\mathbf{D}(\mathbf{W}) = \mathbf{D}_\xi(\mathbf{W}) + \mathbf{D}_\zeta(\mathbf{W}) + \mathbf{D}_\eta(\mathbf{W}), \tag{3.13}$$

where $\mathbf{D}_\xi$, $\mathbf{D}_\zeta$, and $\mathbf{D}_\eta$ are undivided differences across the cell in the streamwise, cross

stream, and normal directions, respectively:

$$\begin{aligned} \mathbf{D}_\xi(\mathbf{W}) &= \mathbf{d}_{i+1/2,j,k} - \mathbf{d}_{i-1/2,j,k} \\ \mathbf{D}_\zeta(\mathbf{W}) &= \mathbf{d}_{i,j,k+1/2} - \mathbf{d}_{i,j,k-1/2} \\ \mathbf{D}_\eta(\mathbf{W}) &= \mathbf{d}_{i,j+1/2,k} - \mathbf{d}_{i,j-1/2,k} \cdot \end{aligned} \tag{3.14}$$

The damping flux terms $\mathbf{d}_{i+1/2,j,k}, \mathbf{d}_{i-1/2,j,k}$, etc. are constructed such that $\mathbf{D}(\mathbf{W})$ is a sum of 2^{nd} and 4^{th} differences across the cell. Each $\mathbf{d}$ is the sum of a first and third difference, for example,

$$\begin{aligned} \mathbf{d}_{i+1/2} = \left(\tfrac{V}{\Delta t}\right)_{i+1/2} [\quad & \epsilon^{(2)}_{i+1/2}(\mathbf{W}_{i+1} - \mathbf{W}_i) - \\ & \epsilon^{(4)}_{i+1/2}(\mathbf{W}_{i+2} - 3\mathbf{W}_{i+1} + 3\mathbf{W}_i - \mathbf{W}_{i-1})] \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} \mathbf{d}_{i-1/2} = \left(\tfrac{V}{\Delta t}\right)_{i-1/2} [\quad & \epsilon^{(2)}_{i-1/2}(\mathbf{W}_i - \mathbf{W}_{i-1}) - \\ & \epsilon^{(4)}_{i-1/2}(\mathbf{W}_{i+1} - 3\mathbf{W}_i + 3\mathbf{W}_{i-1} - \mathbf{W}_{i-2})] \end{aligned} \tag{3.16}$$

where the indices j, k have been suppressed for clarity. Analogous difference stencils are used for $\mathbf{d}_{i,j+1/2,k}$, $\mathbf{d}_{i,j-1/2,k}$, $\mathbf{d}_{i,j,k+1/2}$, and $\mathbf{d}_{i,j,k-1/2}$. With this formulation and the factor of $V/\Delta t$, the artificial viscosity terms are conservative.

The coefficients $\epsilon^{(2)}$ and $\epsilon^{(4)}$ contain pressure weighting and on/off switches. They are defined as:

$$\epsilon^{(2)}_{i+1/2} = \kappa^{(2)}(\nu_{\xi_{i+1}} + \nu_{\xi_i})/2 \,, \tag{3.17}$$

$$\epsilon^{(4)}_{i+1/2} = \max(0, \kappa^{(4)} - \epsilon^{(2)}_{i+1/2}) \,. \tag{3.18}$$

The factor ν provides scaling by the local pressure gradient:

$$\nu_{\xi_i} = \frac{|P_{i+1} - 2P_i + P_{i-1}|}{P_{i+1} + 2P_i + P_{i-1}} \,. \tag{3.19}$$

With the above switches, 2^{nd} and 4^{th} difference smoothing terms are nearly mutually exclusive: At shocks lying in the (ξ, η) plane, $\nu_{\xi_{i,j}}$, for example, is large; thus, $\epsilon^{(4)} =$

$0, \epsilon^{(2)} \neq 0$; in regions of small gradients $\epsilon^{(2)} \approx 0, \epsilon^{(4)} \neq 0$. Stencils at the other cell faces are defined similarly. The coefficients $\kappa^{(2)}$ and $\kappa^{(4)}$ for each calculation are given in the results sections.

In addition to the scaling done via the pressure weighting and on/off switches above, a boundary layer scaling of the artificial viscosity was implemented in two-dimensional test cases. In this scaling, the normal component of the artificial viscosity $\mathbf{D}_\eta$ was set to zero to avoid contamination of the "real" physical viscosity of the fluid with the artificial viscosity of the numerical scheme in boundary layer regions. The scaling region was defined by an *a posteriori* inspection of the flowfield.

3.3.1 Boundary Formulation

The dissipation operators above require special treatment at the boundaries of the computational grid since, at the boundary cell, the second difference operator extends one cell and the fourth difference extends two cells out of the computational domain. Two different approaches were used in this thesis and are described below. Two-dimensional solutions obtained with the two approaches showed no significant differences.

Eriksson Boundary Treatment

A boundary treatment that guarantees a globally dissipative artificial dissipation operator was developed by [Eriksson & Rizzi 83] and is applied for three-dimensional calculations this thesis. In Eriksson's treatment the second difference operator is adjusted by setting the coefficient $\epsilon^{(2)}$ at boundary faces to zero. This changes the character of this artificial viscosity term at the boundary from dissipative to convective, but eliminates

the need for any sort of extrapolation.

For the fourth difference operator, values of the state vector are obtained by linearly extrapolating the values from the first two cells inside the domain to two dummy cells outside the body

$$\begin{aligned} \mathbf{W}_0 &= 2\mathbf{W}_1 - \mathbf{W}_2 \\ \mathbf{W}_{-1} &= 3\mathbf{W}_1 - 2\mathbf{W}_2 \,, \end{aligned} \tag{3.20}$$

where the subscript $_0$ indicates the plane of ghost cells adjacent to the boundary and $_{-1}$ denotes the next plane of ghost cells.

"Zero Smoothing" Formulation

A simple means of eliminating the boundary formulation problem is to set the artificial viscosity terms normal to that boundary to zero. This can be easily done by setting $\epsilon^{(2)}$ to zero at the first boundary cell face normal to the body and $\epsilon^{(4)}$ to zero at the first two cell faces normal to the body. This approach, together with the boundary layer scaling described earlier, was used for two-dimensional calculations.

Chapter 4

Semi-Implicit Temporal Integration

This chapter describes the semi-implicit temporal discretization of the governing equations in semi-discrete form[1]:

$$V\frac{d\mathbf{W}}{dt} = -\sum_{f=1}^{6}\left(\mathbf{F}S_x + \mathbf{G}S_y + \mathbf{H}S_z\right)_f \ , \tag{4.1}$$

Conventional approaches integrate (4.1) explicitly or implicitly in time. For a model equation $u_t = g(u)$, where $u_t = du/dt$ and $g(u)$ is a semi-discrete term, the explicit and implicit integrations can be represented by, respectively,

$$\begin{aligned} u_t = \ & g(u^n) && \text{(Explicit)} \\ u_t = \ & g(u^n, u^{n+1}) && \text{(Implicit)} \ . \end{aligned} \tag{4.2}$$

Explicit methods use functions of u at time t^n only to compute f_t, while implicit methods base that calculation on functions of u at both t^n and t^{n+1}.

In the semi-implicit approach, the semi-discrete right-hand side of (4.1) is divided into two parts, a part containing streamwise and cross stream flux terms, and a part containing body normal flux terms. Integration of the streamwise and cross stream terms is explicit, since grid spacing in these directions imposes relatively mild restrictions on the time step. However, integration of the flux terms in the normal direction is implicit, since the small normal grid spacing necessary in viscous calculations would otherwise impose severe limits on the size of the time step.

[1] The left hand side of Equation 2.1 has now been integrated over the volume V with the assumption that the state vector in each cell is constant throughout the cell.

The next section presents a widely used explicit multistage approach and its stability constraints, followed by derivation of the semi-implicit scheme in Section 4.2. Analyses of the stability characteristics of the semi-implicit scheme and comparison with the characteristics of both the explicit and a fully implicit scheme are given in Appendix A.

4.1 Explicit Multi-Stage Integration

A popular multistage algorithm for fluid dynamic calculations [Jameson et al. 81] is the four stage scheme given by

$$\begin{aligned}
\mathbf{W}^0 &= \mathbf{W}^n \\
\mathbf{W}^1 &= \mathbf{W}^0 - \alpha_1 \tfrac{\Delta t}{V} \left[\textstyle\sum_{f=1}^{6} \left(\mathbf{F}^0 S_x + \mathbf{G}^0 S_y + \mathbf{H}^0 S_z \right)_f - \mathbf{D}^0 \right] \\
\mathbf{W}^2 &= \mathbf{W}^0 - \alpha_2 \tfrac{\Delta t}{V} \left[\textstyle\sum_{f=1}^{6} \left(\mathbf{F}^1 S_x + \mathbf{G}^1 S_y + \mathbf{H}^1 S_z \right)_f - \mathbf{D}^0 \right] \\
\mathbf{W}^3 &= \mathbf{W}^0 - \alpha_3 \tfrac{\Delta t}{V} \left[\textstyle\sum_{f=1}^{6} \left(\mathbf{F}^2 S_x + \mathbf{G}^2 S_y + \mathbf{H}^2 S_z \right)_f - \mathbf{D}^0 \right] \\
\mathbf{W}^4 &= \mathbf{W}^0 - \alpha_4 \tfrac{\Delta t}{V} \left[\textstyle\sum_{f=1}^{6} \left(\mathbf{F}^3 S_x + \mathbf{G}^3 S_y + \mathbf{H}^3 S_z \right)_f - \mathbf{D}^0 \right] \\
\mathbf{W}^{n+1} &= \mathbf{W}^4 \, ,
\end{aligned} \tag{4.3}$$

where the temporal level is identified by superscripts and the multistage coefficients used are

$$\alpha_1 = \frac{1}{4}, \quad \alpha_2 = \frac{1}{3}, \quad \alpha_3 = \frac{1}{2}, \quad \alpha_4 = 1 \, . \tag{4.4}$$

The terms $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ are flux vectors in the x, y, and z directions, respectively. The artificial viscosity $\mathbf{D}$ is calculated at the first stage and frozen for the remainder of that iteration to minimize computational expense. The temporal integration scheme is explicit, since, at each stage, values of the state vector used in the flux calculation are the known values calculated in the previous stage.

The time step Δt that may be taken with (4.3) is limited by the Courant-Friedrichs-Lewy condition. This condition limits the propagation rate of numerical information in explicit schemes for hyperbolic equations to ensure that the numerical domain of dependence of the difference scheme includes the physical domain of dependence. The time step is also restricted by a physical limit on the numerical diffusion time.

The stability criterion for the explicit multi-stage central difference scheme can be derived as (see Appendix A.5)

$$\Delta t \cong \lambda \frac{V}{|\mathbf{W} \cdot \mathbf{S}|_{max} + C|\mathbf{S}_{max}| + \frac{4\mu}{\Delta l_n} S_n} , \tag{4.5}$$

where C is speed of sound, V is volume of the cell, and $\mathbf{W}$ and $\mathbf{S}$ are vector velocity and projected surface area, $\mathbf{W} = u\vec{i} + v\vec{j} + w\vec{k}$ and $\mathbf{S} = S_x\vec{i} + S_y\vec{j} + S_z\vec{k}$, respectively. Δl_n is the "height" of a cell in the direction normal to the body and S_n is the projected area of the cell normal to l_n. The coefficient λ for a four stage scheme may be taken as $\lambda \leq 2\sqrt{2}$.

Examination of the time step constraint (4.5) immediately provides incentive to search for a temporal integration scheme that allows larger steps in time. For simplicity, assume a two-dimensional viscous flow on a rectangular grid with aspect ratio $A_R = \Delta x/\Delta y$ where Δy is the spacing in the direction normal to a body. Expanding the denominator of (4.5) gives approximately

$$\Delta t \cong \lambda \frac{\Delta x}{|u| + C + A_R(|v| + C + |4\mu/\Delta y|)} . \tag{4.6}$$

As the Reynolds number of the flow increases, grids with smaller and smaller normal grid spacing Δy must be used to resolve the boundary layer. Thus, the aspect ratio A_R of the cells increases, the magnitude of the denominator increases and the magnitude of the time step rapidly decreases. Since aspect ratios of 10^4 and even 10^5 are not uncommon in turbulent flows, restrictions on the size of the time steps can be severe.

4.2 Semi-Implicit Integration

The stability restriction due to small grid spacing in the normal direction can be eliminated by treating fluxes in that direction implicitly, while retaining the explicit character of fluxes in streamwise and cross stream directions. Consider the first stage of (4.3) with implicit treatment of the normal component of the flux vectors

$$\begin{aligned}\mathbf{W}^1 - \mathbf{W}^0 = -\alpha_1 \frac{\Delta t}{V} [\quad & (\mathbf{F}^0 S_x + \mathbf{G}^0 S_y + \mathbf{H}^0 S_z)_1 + (\mathbf{F}^0 S_x + \mathbf{G}^0 S_y + \mathbf{H}^0 S_z)_2 + \\ & (\mathbf{F}^0 S_x + \mathbf{G}^0 S_y + \mathbf{H}^0 S_z)_3 + (\mathbf{F}^0 S_x + \mathbf{G}^0 S_y + \mathbf{H}^0 S_z)_4 + \\ & (\mathbf{F}^1 S_x + \mathbf{G}^1 S_y + \mathbf{H}^1 S_z)_5 + (\mathbf{F}^1 S_x + \mathbf{G}^1 S_y + \mathbf{H}^1 S_z)_6 - \mathbf{D}^0)] \ . \end{aligned} \tag{4.7}$$

Each term in parentheses in (4.7) gives the flux across one of six cell grid faces (see Figure 3.1); however, while fluxes through *Faces* 1,2,3,4 are based on values of the state vector at time t^0, fluxes through *Faces* 5 and 6 are based on the ***unknown*** values of the state vector at time t^1. Thus, the discretization of the fluxes is explicit in the cross stream and streamwise directions, but implicit in the normal direction. Numerical experimentation shows that it is not necessary to treat the body normal artificial viscosity terms implicitly.

Since the flux vectors (2.3) – (2.4) are nonlinear with respect to the state vector $\mathbf{W}$, the fluxes at *Faces* 5 and 6 must be linearized in time to allow calculation of $\mathbf{W}^1$. Newton type linearization for $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ gives,

$$\begin{aligned} \mathbf{F}^1 = \ & \mathbf{F}^0 + \frac{\partial \mathbf{F}^0}{\partial t}\Delta t + O(\Delta t^2) = \ \mathbf{F}^0 + [A]^0 \Delta \mathbf{W}^1 + O(\Delta t^2) \\ \mathbf{G}^1 = \ & \mathbf{G}^0 + \frac{\partial \mathbf{G}^0}{\partial t}\Delta t + O(\Delta t^2) = \ \mathbf{G}^0 + [B]^0 \Delta \mathbf{W}^1 + O(\Delta t^2) \\ \mathbf{H}^1 = \ & \mathbf{H}^0 + \frac{\partial \mathbf{H}^0}{\partial t}\Delta t + O(\Delta t^2) = \ \mathbf{H}^0 + [C]^0 \Delta \mathbf{W}^1 + O(\Delta t^2) \end{aligned} \tag{4.8}$$

where $[A]$, $[B]$, and $[C]$ are the 5×5 Jacobian matrices $[\partial \mathbf{F}/\partial \mathbf{W}]$, $[\partial \mathbf{G}/\partial \mathbf{W}]$, and $[\partial \mathbf{H}/\partial \mathbf{W}]$, respectively, and the vector of changes $\Delta \mathbf{W}$ is defined as $\Delta \mathbf{W}^1 \equiv \mathbf{W}^1 - \mathbf{W}^0$.

Inserting (4.8) into (4.7) gives

$$\Delta\mathbf{W} = -\alpha_1 \tfrac{\Delta t}{V} \Big\{ \Big(\textstyle\sum_{f=1}^{6} (\mathbf{F}S_x + \mathbf{G}S_y + \mathbf{H}S_z)_f - \mathbf{D} \Big)^0 + [[A]_5 S_{x_5} + [A]_6 S_{x_6} + [B]_5 S_{y_5} + [B]_6 S_{y_6} + [C]_5 S_{z_5} + [C]_6 S_{z_6}] \Delta\mathbf{W} \Big\} , \tag{4.9}$$

or

$$\Big[[I] + \alpha_1 \tfrac{\Delta t}{V} ([A]_5 S_{x_5} + [A]_6 S_{x_6} + [B]_5 S_{y_5} + [B]_6 S_{y_6} + [C]_5 S_{z_5} + [C]_6 S_{z_6}) \Big] \Delta\mathbf{W} = -\alpha_1 \tfrac{\Delta t}{V} \Big[\textstyle\sum_{f=1}^{6} (\mathbf{F}S_x + \mathbf{F}S_y + \mathbf{H}S_z)_f - \mathbf{D} \Big]^0 , \tag{4.10}$$

where the quantity in the square brackets [ ] prefacing $\Delta\mathbf{W}$ is a matrix which operates on the $5 \times J_{max}$ vector of changes $\Delta\mathbf{W}$ at a given (i,k). Thus, (4.10) is an implicit matrix equation for the changes in the state vector $\mathbf{W}$ at a line of cells (i,k). The right-hand side is the usual semi-discrete form of the residual, while the left-hand side differs from the identity matrix $[I]$ due to introduction of the terms from the time linearization of $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$.

Subsequent steps in a multistage scheme have the same form. For a four stage scheme:

$$\begin{aligned} \mathbf{W}^0 &= \mathbf{W}^n \\ [LHS]^0 \Delta\mathbf{W}^1 &= \mathbf{Res}^0 \\ [LHS]^1 \Delta\mathbf{W}^2 &= \mathbf{Res}^1 - \Delta\mathbf{W}^1 \\ [LHS]^2 \Delta\mathbf{W}^3 &= \mathbf{Res}^2 - (\Delta\mathbf{W}^2 + \Delta\mathbf{W}^1) \\ [LHS]^3 \Delta\mathbf{W}^4 &= \mathbf{Res}^3 - (\Delta\mathbf{W}^3 + \Delta\mathbf{W}^2 + \Delta\mathbf{W}^1) \\ \mathbf{W}^{n+1} &= \mathbf{W}^3 + \Delta\mathbf{W}^4 \end{aligned} \tag{4.11}$$

where

$$\mathbf{Res}^n = -\alpha_n \frac{\Delta t}{V} \left[\sum_{f=1}^{6} (\mathbf{F}S_x + \mathbf{G}S_y + \mathbf{H}S_z)_f^n - \mathbf{D}^0 \right] , \tag{4.12}$$

$$[LHS]^n = [I] + \alpha \frac{\Delta t}{V} \left([A]_5 S_{x_5} + [A]_6 S_{x_6} + [B]_5 S_{y_5} + [B]_6 S_{y_6} + [C]_5 S_{z_5} + [C]_6 S_{z_6} \right)^n \ , \tag{4.13}$$

and

$$\Delta \mathbf{W}^S = \mathbf{W}^S - \mathbf{W}^{S-1} \ . \tag{4.14}$$

Although the *RHS* of stages two and three contain more than one vector $\Delta \mathbf{W}$, only one $\Delta \mathbf{W}$ needs to be stored. Subsequent $\Delta \mathbf{W}$'s are simply added to the stored vector to give $\sum_{i=1}^{s-1} \Delta \mathbf{W}$.[2] The explicit scheme is recovered when $[LHS] = [I]$.

The *LHS* contains dependent variables at $(j-1, j, j+1)$, only, due to the compact three point stencil used in the spatial discretization of the viscous and inviscid terms. Thus, (4.11) is a block tridiagonal system of equations, which can be efficiently inverted with a Gauss elimination routine. Setup of the matrices and solution of the matrix system is discussed in Appendix B.

The time step in the semi-implicit integration is limited only by the tangential and cross stream flux terms, which are treated explicitly. It is derived in Appendix A and given by

$$\Delta t \leq 2\sqrt{2} \frac{Vol}{(|\mathbf{V} \cdot \mathbf{S}|)_{max_{1,2}} + (|\mathbf{V} \cdot \mathbf{S}|)_{max_{3,4}} + C|\mathbf{S}_{max}|_{1,2,3,4}} \ . \tag{4.15}$$

Faces 1 and 2, and 3 and 4 are in the tangential and cross stream directions, respectively. As with the explicit scheme, λ may be taken as $2\sqrt{2}$.

[2]Equations 4.11 imply that we use $\mathbf{G}^2 = \mathbf{G}^1 + [B](\mathbf{W}^2 - \mathbf{W}^1)$ for the second stage, and similarly for stages three and four. Alternatively, one might might use $\mathbf{G}^2 = \mathbf{G}^0 + [B](\mathbf{W}^2 - \mathbf{W}^0)$, where the temporal linearization at each stage refers back to time level 0. Using the latter form eliminates the need for ΔW's on the *RHS* and sets $\mathbf{W}^{n+1} = \mathbf{W}^n + \Delta \mathbf{W}^4$. However, numerical experimentation shows this form results in slower convergence. It was therefore discarded.

4.3 Matrix Conditioning

For grids with high aspect ratio cells the matrix (4.13) becomes increasingly ill conditioned. Consider, for example, a rectangular mesh with $\Delta x = Const.$ and $A_R = \Delta x/\Delta y$, where AR_{xy} is the aspect ratio. Then,

$$\begin{aligned}[LHS]=&[I] + \alpha\frac{\Delta t AR_{xy}}{\Delta x^3}\left[[B]_5\Delta x^2 - [B]_6\Delta x^2\right] \\ =&[I] + \alpha\frac{\Delta t AR_{xy}}{\Delta x}\left[[B]_{j+1} - [B]_{j-1}\right] ,\end{aligned} \tag{4.16}$$

and diagonal dominance is lost as the off diagonal terms increase with A_R, while Δt, which is not a function of A_R, remains constant. The loss of diagonal dominance may result in instability of the numerical calculation.

To increase the diagonal dominance of $[LHS]$ implicit smoothing can be added. It is similar to the implicit residual smoothing that is often used with explicit schemes (see Section 5.1.2). If $[D]$ is the diagonal block and $[LD]$ and $[RD]$ are the off-diagonal blocks in $[LHS]$ the implicit smoothing is added as a nonlinear second difference operator:

$$\begin{aligned}[D]_{i,i} = &\quad [D]_{i,i} + \alpha\frac{\Delta t_{SI,j}}{V_j} \times \mu_{IS}\left(\frac{V_j}{\Delta t_{exp,j}} + \frac{V_{j+1}}{\Delta t_{exp,j+1}}\right) \\ [LD]_{i,i} = &\quad [LD]_{i,i} - \alpha\frac{\Delta t_{SI,j}}{V_j} \times \mu_{IS}\frac{V_j}{\Delta t_{exp,j}} \\ [RD]_{i,i} = &\quad [RD]_{i,i} - \alpha\frac{\Delta t_{SI,j}}{V_j} \times \mu_{IS}\frac{V_{j+1}}{\Delta t_{exp,j+1}}\end{aligned} \tag{4.17}$$

The subscripts SI and exp denote semi-implicit and explicit, and i and j denote the matrix element and normal (to the body) grid line counter, respectively. This form is chosen because the ratio $\Delta t_{SI}/\Delta t_{exp}$ varies inversely with the normal dimension of computational cells[3]. Thus the implicit smoothing is largest in the boundary layer region where the matrix would otherwise be very poorly conditioned, and is $O(\mu_{IS})$ in coarse grid regions. The implicit smoothing does not affect the steady state solution;

[3]The factor of V is included because it is the ratio $\Delta t/V$ that is computed in the code, not just Δt

however, large μ_{IS} may decrease the rate of convergence to steady state. Although this option was implemented in the numerical code, it was not used for any of the final calculations.

4.4 Code Validation

Consistency of the finite volume approximation to the thin layer Navier-Stokes equations with the governing equations can be investigated by computing a well known test case and comparing with results obtained by other investigators. This is also a valuable test for coding errors in the computer program and a means of validating numerical results. Computational efficiency will be addressed in the next chapter.

The viscous, $M_\infty = 0.5$, $Re = 8065$ flow in a channel with a bump on the lower wall was chosen as the test case. Mach contours of this problem are shown in Figure 4.4. The flow field begins at $x = -1$. A 5% (of channel height) circular arc bump begins at $x = 0$ and ends at $x = 1$. The chanel extends to $x = 2$. Symmetry boundary conditions are applied at $y = 0$ $(-1 < x < 0)$ and $y = 1$ $(-1 < x < 2)$. This case has been calculated by [Chima 85], [Whitfield 86], and [Davis et al. 87] and others.

The case was computed on a rectangular grid, with 65 evenly spaced streamwise nodes. Thirty-three nodes were placed in the cross-stream direction, with the first node at $y_1 = 1.4 \times 10^{-3}$ and subsequent node locations defined by $\Delta y_{j+1} = 1.16 \Delta y_j$. This ensures adequate resolution of the boundary layer. Artificial viscosity effects were minimized by choosing very small smoothing coefficients: $\nu_2 = 0.0, \nu_4 = 0.002$.

Normalized pressure distribution and skin friction results for this case are shown in Figures 4.2 and 4.3, respectively, along with numerical data from [Chima 85], [Davis et al. 87],

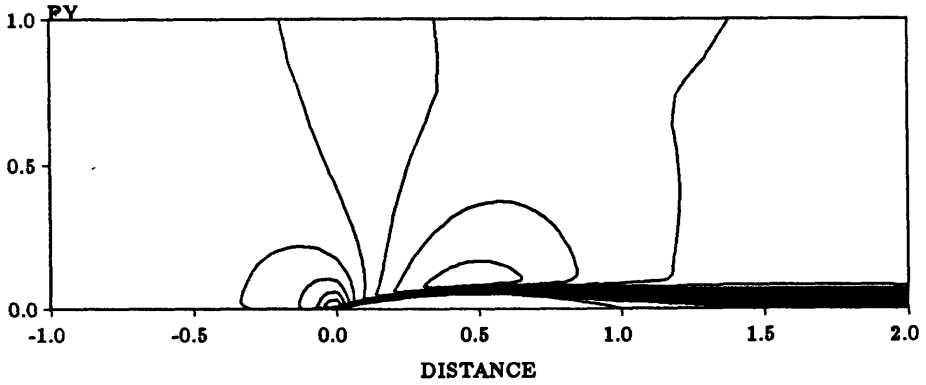

Figure 4.1: Flow geometry and Mach contours for code validation

and [Whitfield 86]. All three calculations predict a small separation zone near the cor-

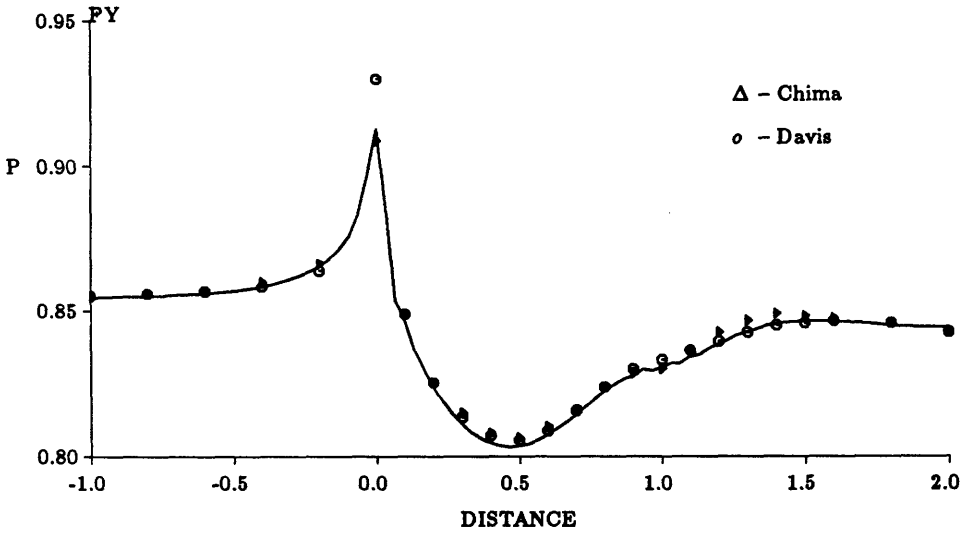

Figure 4.2: Normalized pressure at lower surface

ner at $x = 1$. Aside from differences near the discontinuity and a small discrepency in the pressure near the inlet, agreement is excellent.

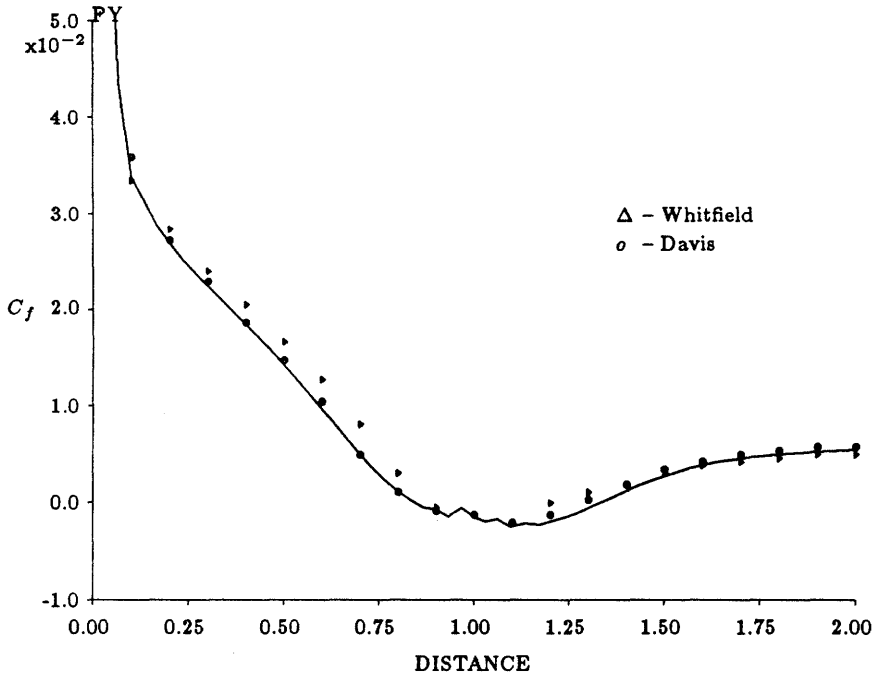

Figure 4.3: Skin friction

Chapter 5

Computational Efficiency of the Temporal Integration

This chapter compares the efficiency of the semi-implicit scheme to that of an explicit four stage multi-stage scheme [Jameson et al. 81] and a fully implicit algorithm [Beam & Warming 76]. The Beam & Warming scheme [Anderson et al. 84] uses a three point backward temporal integration, although backward Euler temporal integration was also tried with similar results. The measure of efficiency used is *CPU* time required for the solution to reach a measure of convergence. The three schemes were compared by calculating several inviscid and viscous problems in two dimensions only, since comparisons in three dimensions are prohibitively expensive.

Considerable care was taken to ensure a fair comparison. All of the computer programs were written "from scratch" by the author. Boundary condition, artificial viscosity, and flux balance modules in the codes were identical, and the *CFL* numbers that resulted in the quickest convergence were chosen. The explicit form of the time step (Equation 4.5) was used for the fully implicit scheme, together with the largest coefficient λ consistent with stability. Implicit smoothing was used for the fully implicit scheme. The magnitudes of the implicit smoothing factor and time step factor λ were optimized for each case calculated with the implicit scheme.

The explicit scheme was obtained by setting the left hand side of *SINSS* (4.11)

to the identity matrix. Since explicit schemes often use one or more techniques to accelerate convergence of the solution to steady state, a number of these techniques were implemented to ensure a fair comparison. Various implementations of the acceleration techniques were examined to determine the optimal conditions.

The next section describes the convergence acceleration techniques that were implemented for each of the schemes, and the following section gives the results of applying the schemes to the model problems. Despite efforts at creating a fair comparison, because of possible differences between actual and "ideal" coding, machine efficiency and utilization, etc., the numerical comparisons shown below should be considered accurate to no more than approximately 25% of stated *CPU* times.

5.1 Convergence Acceleration Techniques

Straightforward solution of the discrete governing equations usually results in very long computation times due either to the large number of iterations needed to achieve steady state, to the large number of operations needed per iteration, or to a combination of these two factors. Because large *CPU* times translate directly into a high dollar cost of computations and long turn-around times, numerous techniques to speed convergence have been proposed for Navier-Stokes calculations. Below is a list of some acceleration techniques along with a brief description of each and a literature reference:

1. **Implicit residual smoothing**: The residuals of the discrete equations are smoothed implicitly giving the algorithm a larger effective difference stencil, and, correspondingly, a less stringent CFL condition [Jameson & Baker 83], [Lerat & Sides 79].

2. **Grid sequencing**: Preliminary solutions are calculated on coarse grids to supply improved initial conditions to the fine grid calculation.

3. **Multigrid**: Calculations are cycled between a fine and one or more coarse grids in an attempt to attain the rapid information transfer characteristics of the coarse grid and achieve optimum error smoothing rates while maintaining the spatial accuracy of the fine grid [Brandt 80], [Jameson 83], [Ni 81].

4. **Matrix diagonalization**: Matrices in implicit methods are diagonalized to reduce computational effort of the matrix inversion [Pulliam & Chaussee 81].

5. **Wynn's ϵ Algorithm**: Approximate solutions at different iteration levels are used to predict the final result [Hafez et al. 87].

6. **GMRES**: Generalized minimal residual algorithm that uses a conjugate gradient approach to accelerate convergence of nonlinear systems [Wigton 85].

7. **Local time stepping**: Each grid cell is advanced by its own (maximum) time step thus giving a higher average time step for the entire grid [Eriksson & Rizzi 83], [Jameson et al. 81].

Each of these techniques has a unique numerical character which may or may not be compatible with the host algorithm and the nature of the desired solution. For example, several of these techniques apply specifically to explicit or implicit schemes, only. All of the techniques give the same steady state solution as the original discrete equations if the solution converges at all, but some destroy its temporal accuracy. Many were developed to be used with Euler methods and work very well for inviscid solutions, but perform poorly on the highly stretched grids commonly used for viscous calculations. Because of these and other reasons implementation and use of the techniques is, unfortunately, often as much art as it is science.

Local time stepping, residual smoothing and multigrid are three acceleration techniques that have come into widespread use with explicit schemes, and were therefore included in the comparison. Implementation of these techniques results in a scheme that is not time accurate but, one hopes, has better convergence properties. A more detailed description of each technique is given below.

5.1.1 Local Time Stepping

Local time stepping is implemented by applying the stability criterion (4.5), in the explicit and implicit schemes, or (4.15) in the semi-implicit scheme, locally in each cell. Thus, each cell is advanced at a time step corresponding to the local stability restriction, and large cells are not unduly restricted by the stability limit due to the smallest cell in the grid. Temporal accuracy is lost, but the improvement in convergence rate can be impressive, especially if the computational grid contains cells of disparate sizes. Local time stepping is very simple to implement, and can be used with explicit and implicit schemes. All results shown in this thesis were calculated with local time stepping.

5.1.2 Implicit Residual Smoothing

The *CFL* number limitation for explicit and semi-implicit schemes can be relaxed by implicitly smoothing the residuals [Jameson & Baker 83]. In one dimension residual smoothing is applied via

$$(1 - \mu_R \delta_{\eta\eta})\mathbf{Res}' = \mathbf{Res} \ , \tag{5.1}$$

where **Res** is the vector of residuals in (4.12), $\delta_{\eta\eta}$ is the undivided second difference operator $\delta_{\eta\eta} = (\ )_{j+1} - 2(\ )_j + (\ )_{j-1}$, and μ_R is a smoothing constant. Application of this operator increases the stencil of influence of the difference scheme and thus increases

the permissible time step. The original residuals **Res** are replaced in the computation by the vector of smoothed residuals **Res'**. The left hand side of Equation 5.1 gives a scalar tridiagonal matrix. Details of matrix setup and inversion are given in Appendix C.

In multiple dimensions Equation 5.1 is applied successively in each direction. For example in three dimensions (ξ, η, ς):

$$\begin{aligned} (1 - \mu_R \delta_{\xi\xi})\mathbf{Res}' &= \mathbf{Res} \\ (1 - \mu_R \delta_{\eta\eta})\mathbf{Res}'' &= \mathbf{Res}' \\ (1 - \mu_R \delta_{\varsigma\varsigma})\mathbf{Res}''' &= \mathbf{Res}'' \,, \end{aligned} \tag{5.2}$$

and **Res** is now replaced by $\mathbf{Res}'''$ in the calculation. Residual smoothing in a given direction is superfluous if the discrete fluxes are already treated implicitly as in a semi-implicit or fully implicit scheme. In a semi-implicit scheme it may still be profitably applied in the explicit direction(s).

Residual smoothing (RS) may be implemented in a number of different ways, each of which has unique convergence characteristics. A large number of versions were implemented and tested on a viscous channel flow as part of this study. It was found for the explicit scheme that computation was most efficient if residual smoothing was applied after alternate stages and run with a smoothing coefficient of 1.75 and a *CFL* number of approximately 5. In the semi-implicit solver RS was applied in the explicit direction(s) only, *after* the implicit portion of the integration of the fourth stage. Values of the smoothing coefficient and CFL number were approximately 2.0 and 3.5, respectively.

5.1.3 Multigrid

Multigrid was originally developed for systems of elliptic equations by [Brandt 80] and others, but has recently been applied to systems of equations with mixed character such as the Euler or Navier-Stokes equations (e.g. [Ni 81], [Jameson 83]). It is generally used for explicit calculations, and thus was implemented only in the explicit multistage scheme. Multigrid accelerates the propagation of information across a grid and dissipates low frequency error by solving the conservation equations on successively coarser grids. Solution corrections on the coarse grids are intermittently interpolated up to the fine grid solution which drives the scheme.

Any number of multigrid cycling strategies is possible. In the comparisons to follow, a simple V-type strategy was used. In this strategy, one multistage iteration on the fine grid is followed by one iteration on each of a number of successively coarser grid. After completing calculation on the coarsest grid, corrections on that grid are interpolated back up to the finest grid without further iteration.

In the following discussion, h denotes the finest level, $2h$ the first coarsening, $4h$ the next, etc., so that at a level nh the next coarser level is $2nh$. A coarse $(2nh)$ grid is defined by dropping alternate grid lines in the next (nh) grid. The areas of $2nh$ cells must be defined in terms of areas of nh cells,

$$A_{i,j}^{2nh} = A_{i,j}^{nh} + A_{i+1,j}^{nh} + A_{i,j+1}^{nh} + A_{i+1,j+1}^{nh} \,, \tag{5.3}$$

and state variables at $2nh$ are inferred by an area weighted average of the included nh cells:

$$\mathbf{W}_{i,j}^{2nh} = \left(\mathbf{W}_{i,j}^{nh} A_{i,j}^{nh} + \mathbf{W}_{i+1,j}^{nh} A_{i+1,j}^{nh} + \mathbf{W}_{i,j+1}^{nh} A_{i,j+1}^{nh} + \mathbf{W}_{i+1,j+1}^{nh} A_{i+1,j+1}^{nh}\right) / A_{i,j}^{2nh} \,. \tag{5.4}$$

Each stage of the multistage scheme at coarse levels uses a restriction operator in

place of the customary residual. The restriction operator is a forcing function that uses the fine grid solution and previous coarse solutions to drive the solution at the current coarse level. For the first coarse stage the restriction operator is defined as

$$\mathbf{P}^{2h} = \sum \mathbf{Res}^{h}(\mathbf{W}^{h}) - \mathbf{Res}^{2h}(\mathbf{W}^{0^{2nh}}) \tag{5.5}$$

where **Res** is the residual and the sum is over the 4 included cells. For subsequent coarse level the restriction operator is

$$\mathbf{P}^{2nh} = \sum \left[\mathbf{Res}^{nh}(\mathbf{W}^{nh}) + \mathbf{P}^{nh}\right] - \mathbf{Res}^{2nh}(\mathbf{W}^{0^{2h}}) \; . \tag{5.6}$$

At coarse grid levels it is important to damp out errors at short wave lengths, such as the odd-even decoupling allowed by the difference scheme. For this reason 2^{nd} and 4^{th} difference artificial viscosity is added to the fluxes at the coarse grids, which does not affect the steady state solution. The artificial viscosity is calculated before the second stage[1] and then frozen.

After calculation on the coarsest grid, repeated bilinear interpolation is used to transfer changes at level $2nh$ to level nh and eventually to level h. Contributions are taken from the host and the nearest three coarse cells weighed as $\frac{9}{16}, \frac{3}{16}, \frac{3}{16}, \frac{1}{16}$. At grid boundaries, interior cell changes are reflected to ghost cells to allow this interpolation.

[1] This prevents the artificial viscosity from being identically zero at coarse levels due to the definition of the restriction operator.

The solution is updated via the modified multi-stage scheme:

$$\begin{aligned}
\mathbf{W}^1 &= \mathbf{W}^0 - \alpha_1 \tfrac{\Delta t}{S}\left[\mathbf{Res}^{2nh^0} + \mathbf{P}^{2nh}\right] \\
\mathbf{W}^2 &= \mathbf{W}^0 - \alpha_2 \tfrac{\Delta t}{S}\left[\mathbf{Res}^{2nh^1} + \mathbf{P}^{2nh}\right] \\
\mathbf{W}^3 &= \mathbf{W}^0 - \alpha_3 \tfrac{\Delta t}{S}\left[\mathbf{Res}^{2nh^2} + \mathbf{P}^{2nh}\right] \\
\mathbf{W}^4 &= \mathbf{W}^0 - \tfrac{\Delta t}{S}\left[\mathbf{Res}^{2nh^3} + \mathbf{P}^{2nh}\right] \\
\mathbf{W}^{n+1} &= \mathbf{W}^4
\end{aligned} \tag{5.7}$$

where the coefficients $\alpha_1 = 0.25$, $\alpha_2 = 0.5$, and $\alpha_3 = 0.55$ are chosen to optimize the temporal damping properties of the scheme [Jameson 83]. When residuals on the fine grid are zero, the right hand sides of the first and all subsequent stages are zero. Thus, the steady state solution is unaffected by anything done on coarser grids.

5.2 Results

Figures 5.1 and 5.2 give geometries and typical Mach number contours of the two sets of cases considered below. The first is inviscid flow at $M_\infty = 0.5$ in a channel over a circular arc bump with thickness/chord ratio $t/c = 0.1$. The second is laminar viscous flow at $M_\infty = 0.5$ in a channel, with a lower wall beginning 1/3 of the way into the channel. In both cases, symmetry is assumed at the upper boundaries. Because the flows are subsonic no second difference smoothing is added to the spatial operator. A small amount of fourth difference smoothing ($\nu_4 = 0.002$) is added.

Grids for both cases were calculated algebraically, with constant spacing in the streamwise direction. A stretching factor $A = \Delta y_{j+1}/\Delta y_j$ was used to cluster grid lines in the normal direction.

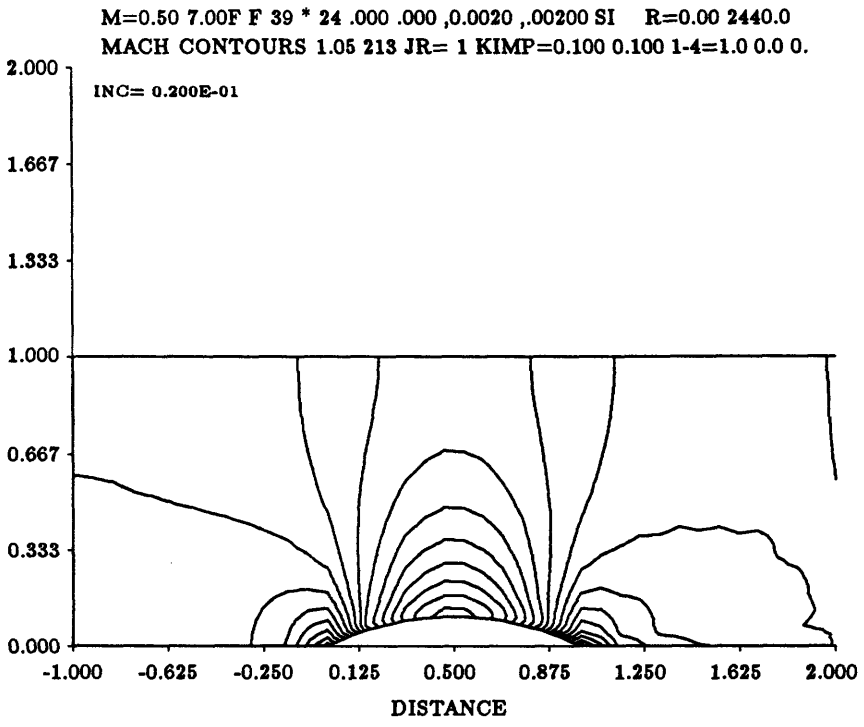

Figure 5.1: Mach Contours, $M_\infty = 0.5$

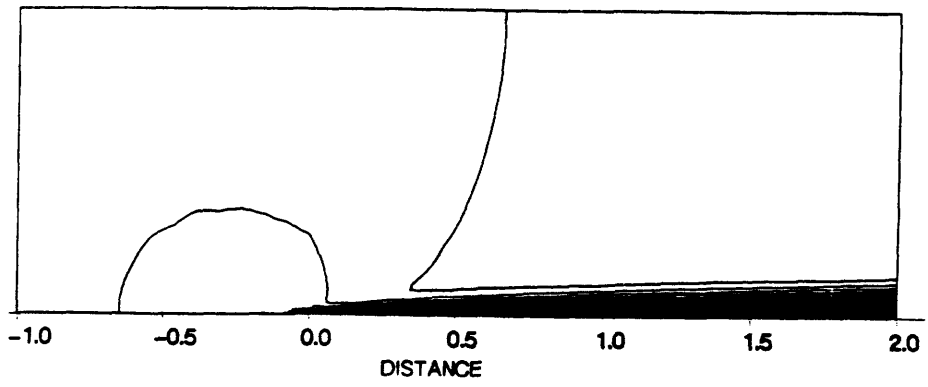

Figure 5.2: Mach Contours in Viscous Channel Flow $Re = 2000$

5.2.1 Convergence Criteria

The convergence criterion used in inviscid cases was

$$\frac{1}{I \times J} \sum_{i,j=1}^{I,J} \left\{ \left| \frac{\Delta \rho u}{\Delta t} \right| + \left| \frac{\Delta \rho v}{\Delta t} \right| + \left| \frac{\Delta \rho E / E_\infty}{\Delta t} \right| \right\}_{i,j} \leq 5 \times 10^{-4} , \tag{5.8}$$

where $I = I_{max} - 1$, $J = J_{max} - 1$, and $\Delta \rho u$, $\Delta \rho v$, and $\Delta \rho E$ are the changes in ρu, ρv, and ρE calculated during one iteration. This criterion ensures that the solution is globally converged and allows fair comparison of convergence histories calculated with different methods and time steps.

In viscous flow cases, the accurate prediciton of skin friction is of importance. To ensure that this quantity is converged the percentage change in skin friction coefficient over time was required to satisfy the criterion

$$\sum_{N=n}^{n-2} \frac{1}{\Delta t_{ave}} \left[\frac{C_f^n - C_f^{n-1}}{C_f^n} \right]^n \leq 5 \cdot 10^{-4} \tag{5.9}$$

was used in addition to (5.8). Summing over three iterations in (5.9) helps eliminate spurious small values of $C_f^n - C_f^{n-1}$ that may be due to oscillatory convergence of the skin friction coefficient. Δt_{ave} is the average time step of the row of cells adjacent to the body.

5.2.2 Inviscid Flow Cases

Table 5.1 gives the iteration histories for inviscid cases calculated with the explicit scheme. Each calculation was made on a grid with 48 streamwise cells and 16, 24, or 32 cells in the cross stream direction, with a stretching factor, $A = 1.05$. The second and third columns give the *MicroVAX II CPU* time (in minutes) and iteration count for the "standard" scheme with only local time stepping. The fourth and fifth give those results for the scheme with multigrid (MG) and residual smoothing (RS).

Table 5.1: Inviscid Solution with Explicit Scheme

	Standard		MG & RS	
# cells	CPU(m)	ITER	CPU(m)	ITER
16	291	4177	19	155
24	754	7295	37	208
32	1370	9970	71	306

Explicit solutions without multigrid and residual smoothing are characterized by a very slow convergence rate, due, in part, to slow damping of pressure waves. Multigrid and residual smoothing is very effective for these cases, resulting in an iteration count reduction of up to a factor of 35 giving a factor of 20 savings in *CPU* time.

Table 5.2 gives convergence histories for the same flow cases using the semi-implicit and fully implicit schemes. The semi-implicit scheme uses residual smoothing in the streamwise direction. The iteration count is much smaller with both the semi-implicit and the fully implicit algorithm than with the explicit scheme. However, each iteration takes proportionally more *CPU* time, resulting in an efficiency similar to that of the explicit scheme with multigrid and residual smoothing. Note that the semi-implicit scheme's iteration counts appear to be unaffected by the number of normal grid cells. This is a result of the instantaneous information transfer in the body-normal direction due to the implicit treatment of the discrete operator in that direction.

Table 5.2: Inviscid Solutions with semi-implicit and Beam & Warming Scheme

	Semi-Implicit		B & W	
# cells	CPU(m)	ITER	CPU(m)	ITER
16	42	118	30	190
24	63	117	62	264
32	83	115	63	203

5.2.3 Viscous Flow Cases

Table 5.3 gives the iteration histories for the viscous cases at Reynolds numbers (based on channel height and U_∞) of 2×10^3, 10^4, and 10^5. All cases have 48 cells in the streamwise direction and 24, 32, or 32 cells across the half-channel. The grids were generated with stretchings of 1.12, 1.15, and 1.18, respectively. The second column gives the acceleration mechanism: residual smoothing (RS) and/or multigrid (MG) for the explicit code. The semi-implicit code used only residual smoothing (in the streamwise direction). The implicit code used neither multigrid nor residual smoothing.

The semi-implicit scheme does significantly better than the explicit or the implicit scheme at all Reynolds numbers. The implicit scheme converged with difficulty at the highest Reynolds number, despite attempts with a variety of parameter values. The grid stretching at high Reynolds numbers renders the explicit solver increasingly stiff and decreases the effectiveness of multigrid. The Reynolds number 10^5 case converged only *without* multigrid.

Table 5.3: Viscous Channel Flow

		Explicit		Semi-Implicit		B & W	
Re #	Accel.	CPU(m)	ITER	CPU(m)	ITER	CPU(m)	ITER
2×10^3	MG & RS	99	493	–	–	–	–
	RS	–	–	59	97	–	–
	Standard	219	1678	76	127	75	298
1×10^4	MG & RS	366	1385	–	–	–	–
	RS	–	–	72	89	–	–
	Standard	574	3345	101	125	142	421
1×10^5	MG & RS	DNC	DNC	–	–	–	–
	RS	548	2820	59	73	–	–
	Standard	758	4418	82	102	550	1500

5.3 Discussion

The semi-implicit scheme is superior to both the explicit and fully implicit schemes for all viscous flow cases in these comparisons. In three dimensions, the relative efficiency of semi-implicit to fully implicit scheme is expected to increase, since the Beam & Warming scheme then requires one additional block tridiagonal inversion while the semi-implicit scheme requires no additional inversions.

The results above show that while the conventional explicit scheme with acceleration techniques works well in inviscid calculations, where speedup factors of over 30 (in terms of iterations) were achieved, it does not perform nearly as well in viscous cases. Less than a factor of 2 improvement was achieved in the highest Reynolds number case.

It is important to note, however, that viscosity in or of itself is not the culprit. If it were, then differences between the efficencies of viscous and inviscid calculations would decrease with increasing Reynolds number. The opposite is true. Instead, the decrease in efficiency seems to be due to the presence of very high aspect ratio $AR_{xy} = \Delta x/\Delta y$ cells. The convergence rate decreases as the Reynolds number increases and grids with higher and higher aspect ratios are used to provide adequate resolution of the boundary layer.

It is these high aspect ratio cells that cause a numerical "stiffness" in the discrete equations [Eriksson & Rizzi 83] which results in a drastically reduced convergence rate. Although little analysis of acceleration techniques on non-Cartesian grids has been done, it is clear that they do not perform well on stretched grids. Other investigators have observed that multigrid techniques fail when the aspect ratios are too large (e.g. [Melnik 87], [Shmilovich & Caughey 81]) and the nature of the grid appears to have the same detrimental effect on residual smoothing.

A concern with the semi-implicit and fully implicit schemes is the poorly conditioned left hand side matrix (B.13). For a given grid this matrix becomes more ill-conditioned as the stabilizing effect of the viscous terms decreases with increasing Reynolds number. If the matrix is sufficiently ill-conditioned the temporal iteration may diverge. Although it is always possible to stabilize the matrix by adding an appropriate amount of implicit smoothing, this may reduce the convergence rate of the solution. An alternate solution to this problem is use a spatial discretization for the inviscid terms such as an upwind method which results in a more diagonally dominant system [MacCormack 84].

5.3.1 Effect of Computer Architecture on Efficiency

While the calculations above were made on a scalar computer with a single processor, performance of the schemes on vector machines with multiprocessors is an important issue since computational efficiency of any numerical algorithm is affected by the architecture of the computer system used. Thus, the flexibility with which an algorithm can be adapted to different architectures is an important factor in its use. In general, the simpler a scheme is the more likely it is that it will run efficiently on vector and parallel processors as well as on more conventional scalar computers.

The explicit scheme maps easily onto vector and/or parallel processors, since each cell at a given time level is independent of all other cells at that time-level, and all cells may be operated on in parallel. The semi-implicit and implicit schemes can also be efficiently run on vector and/or parallel processors, despite the recursive nature of the matrix inversions that the require. The Gaussian elimination of the left hand side matrices must be structured such that a plane of grid cells is treated simultaneously, rather than the inversion being line-by-line of grid cells. Because the inner loop can then be run in vector and/or parallel mode, the recursive nature of the algorithm is circumvented.

When the computer coding is done in this manner the relative efficiencies of the three methods remains constant[2] unless the computer system is so massively parallel, or has such a long optimal vector length, that the speedup of treating *all* cells in parallel exceeds that of treating them plane by plane. If this were the case the explicit scheme's efficiency relative to semi-implicit and fully implicit schemes would improve.

[2]This was verified by running the codes on a *Cray XMP*.

This page intentionally left blank.

Chapter 6

Blunt Leading Edge Delta Wing Calculations

This chapter and the following two describe three dimensional calculations of flow past delta wings with blunt leading edges. The following introductory section presents a description of the flow physics and a brief survey of relevant computational research on delta wings with sharp and rounded leading edges. After this, the wing geometry, computational grids, and test parameters are presented, the computational code is described, and initial and boundary conditions are dicussed. Finally, numerical aspects of the solutions are discussed and their accuracy is determined by comparison with experiment and by examining the effect of various numerical parameters. Chapter 7 describes features of the flow solutions, and Chapter 8 examines separation processes at the rounded leading edges.

Semi-implicit calculations of *separated* flows with local time steps were found to diverge. The computations converged only when the time step was set equal to a constant at each streamwise station. Thus, the numerical time step that is used in a typical cell is smaller than if true local time-stepping were used. Nevertheless, comparison of the explicit (with local time-stepping) and semi-implicit schemes in terms of CPU time shows that semi-implicit integration is considerably more efficient than explicit integration for all cases considered.

The effect of numerical parameters on global features of the solution was found to be small. Grid resolution, artificial viscosity, and location of the starting solution were

examined. In each case, the effect of varying the parameter on the computed pressure distribution was noticeable; however, no fundamental changes in the flow character were detected. Comparison with experiment established the accuracy of the numerical solution.

6.1 Introduction

Flows about delta wings contain a variety of interesting and intricate patterns. At low angles of attack the flow is usually attached to the body and can be modelled with inviscid potential aerodynamic theory. At sufficiently high angles of attack the flow separates at or near the leading edge to form cores of spiralling fluid – leading edge vortices – that lie above the wing. The adverse pressure gradient induced by the recompression just inboard of the expansion around the leading edge may cause the outboard flowing fluid beneath the vortex to separate, resulting, in turn, in secondary and even tertiary vortices. Shock waves that interact with the boundary layer and vortices may also complicate the flow picture. The intricacy of possible flow patterns makes vortical delta wing flows fascinating to examine. However, the complex interaction between inviscid and viscous effects makes this examination difficult.

Apart from the esthetic qualities of these patterns, vortical flow around delta wings is of interest for several reasons. If the vortices are symmetric and stable they result in pressure suction peaks on the leeward side that lend high lift characteristics to the wing. If, on the other hand, the vortices become asymmetric or unstable while above the wing, they may induce catastrophic spin or stall. Finally, if the vortices remain strong and coherent even in the wake of the wing they pose a hazard to following aircraft or interact with tail surfaces.

6.1.1 Flow about Delta Wings with Sharp Leading Edges

Flow about delta wings with *sharp* leading edges has been the subject of numerous numerical investigations (eg. [Murman et al. 85], [Powell 87], [Thomas & Newsome 86], [Rizzi 86]). Flow separation in these cases occurs at the sharp leading edges of the wing where, it is argued, it is triggered by a Kutta condition enforced by truncation error and/or artificial viscosity effects. Thus, these investigations have typically been made with the Euler or conical Euler equations, whose solution is less expensive than that of the Navier Stokes equations. While these inviscid solutions are admissible solutions to the discrete equations and often show reasonable agreement with experiment, they do not capture real viscous effects and may model the physical character of the separation process incorrectly.

For flows about delta wings with sharp leading edges [Stanbrook & Squire 64] collapsed the three parameter space $(\Lambda, \alpha, M_\infty)$ into a (M_N, α_N) plane defined by

$$M_N = M_\infty \cos\Lambda \sqrt{1 + \sin^2\alpha \tan^2\Lambda} \tag{6.1}$$

and

$$\alpha_N = \tan^{-1}\left(\frac{\tan\alpha}{\cos\Lambda}\right) . \tag{6.2}$$

These parametes give the Mach number and angle of attack of the flow in the direction normal to the leading edge.

[Miller & Wood 83] used experimental data and previous results from [Stanbrook & Squire 64] to postulate a flow region diagram (Figure 6.1) that characterizes sharp leading edge flow cases by vortex and shock structure. To the left of the $M_N \simeq 1$ line the leading edge is contained in the Mach cone and flow separates at the leading edge. The separation at the sharp leading edge is thought to be due to the equivalent of a Kutta condition (eg. [Powell 87]). For normal Mach numbers greater than approximately one,

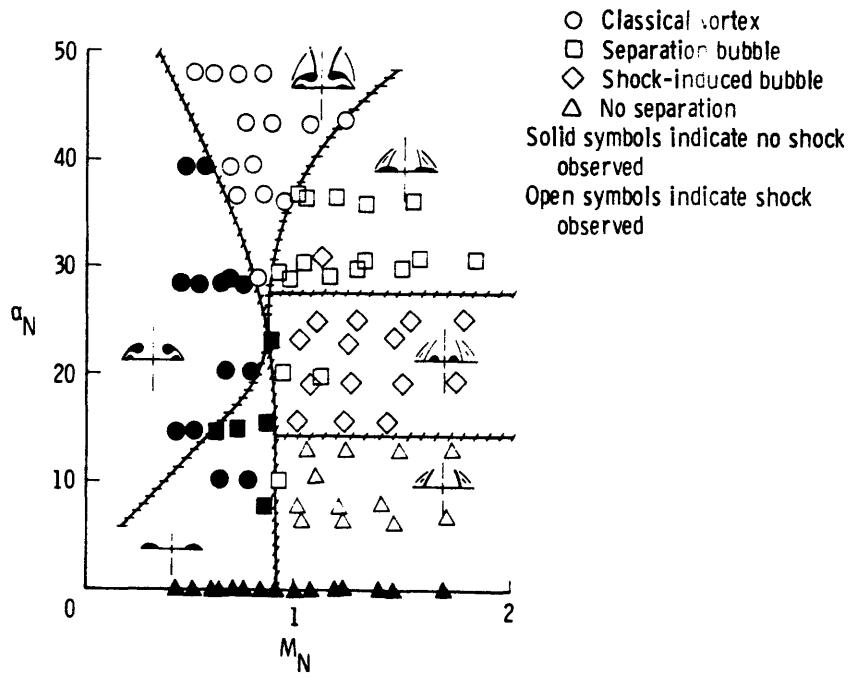

Figure 6.1: Flow regime character for flat plate delta wings

the flow is attached at the leading edge and goes through a Prandtl-Meyer expansion [Salas & Daywitt 79].

6.1.2 Delta Wings with Blunt Leading Edges

Unlike flow about delta wings with sharp leading edges, the locations of separation lines in flow about wings with blunt leading edges are not well defined. Table 6.1 gives flow parameters and values of M_N and α_N for each of the cases considered as well as M_N and α_N. All of the cases being considered have subsonic flow normal to the leading edge.

While the parameter and flow regime descriptions in Table 6.1 and Figure 6.1 give useful characterizations of flows about delta wings with *sharp* leading edges, they are less useful for flows about delta wings with rounded leading edges. This is because the location of the primary flow separation line in flows around blunt leading edges is not

Table 6.1: Flow Parameters

Test Case	Re_L	M_∞	α	Λ	M_N	α_N
Wing F & T	10^6	1.6	4°	65°	0.68	9.4°
Wing F & T	10^6	1.6	8°	65°	0.71	18.4°
Rogers wing	2×10^6	1.41	14°	60°	0.76	26.5°

fixed as is the separation line at a sharp leading edge. Depending on flow conditions and curvature of the leading edge, separation may occur at the leading edge, at some location inboard of the leading edge, or not at all. The location of the separation lines is determined by a complex interaction of viscous and inviscid effects that can not be modeled with the Euler or other inviscid sets of equations[1]

Similarly, two viscous equation sets that are alternatives to the full Navier-Stokes equations, the parabolized Navier-Stokes (PNS) and the conical Navier-Stokes (CNS) equations, are of questionable validity when applied to study the flow separation around blunt leading edges. These equation sets are less expensive to solve than the full Navier-Stokes equations, but have limited generality. PNS schemes require *apriori* knowledge of streamwise pressure gradients in the flow, which must be supplied from experimental data or from an inviscid solver coupled to the PNS scheme. Alternatively, the pressure gradient terms in the original equations must be modified. These modifications, however, lack rigorous theoretical basis. Moreover, PNS schemes tend to be poorly conditioned in flows with extensive separation or strong viscous-inviscid interaction.

Schemes based on the conical Navier-Stokes (CNS) equations have been used to

[1][Newsome 85], for example, finds that, for wings with blunt leading edges, the conical Euler equations admit grid-dependent solutions that are inconsistent with the real flow.

predict overall features of supersonic flow over conical delta wings (e.g. [McRae 76], [Bluford 79], [Ruffin 87]). However, the conical self similarity assumption used to derive these equations is not valid in viscous regions, solutions obtained with these equations lack rigorous justification, and the class of flow cases that can be investigated is small. Nevertheless the CNS equations can give a useful approximation.

For these reasons, examination of the flow separation process at the rounded leading edges of delta wings requires use of the full three-dimensional Navier-Stokes equations. Primarily because of the high computational cost of these solutions, limited numerical research has been done in this area (e.g. [Müller & Rizzi 87], [Newsome & Thomas 85], [Fuji & Kutler 83]). [Müller & Rizzi 87] investigated the laminar transonic flow over a cropped delta wing with an explicit Runge-Kutta solver. A flow case at angle of attack of 10°, $M_\infty = 0.85$, and $Re_L = 2.4 \times 10^6$ was calculated on a 400,000 point grid. The solution was compared to experimental data and Euler results, however convergence of the viscous calculation was questionable despite use of local time steps and grid sequencing.

[Newsome & Thomas 85] used an upwind-difference method to compare Euler and thin-layer Navier-Stokes solutions for the flow about rounded and sharp leading edge delta wings.

[Fuji & Kutler 83] solved the three-dimensional thin-layer Navier-Stokes equations with an implicit Beam & Warming solver. They use this scheme to to calculate the $M_\infty = 0.5$ flow about a delta wing and a strake delta wing with rounded leading edges [Fuji & Kutler 83], and, subsequently, to perform a more realistic calculation about the previously used 76° swept delta wing [Fuji & Kutler 84]. While the first set of calculations showed only primary separation, both primary and secondary separation

was observed in the latter computation which utilized an improved computational grid. However, comparison of the results with experimenal data showed discrepancies on the leeward side, notably in the prediction of a sharp negative pressure peak at the leading edge which was not evident in the experiment. This was ascribed to the fact that the numerical model of the wing leading edge was not as sharp as that of the experimental model.

Two goals of this research program were, first, to examine and characterize the flow separation processes at blunt leading edges, and, second, to decrease the cost of these calculations by using the semi-implicit approach. Thus, the semi-implicit Navier-Stokes solver was implemented in an existing 3–D Euler scheme, and a number of flow calculations were performed to gain an understanding of the flow physics and, especially, of the separation phenomena at the leading edge. The following four sections present the computational code, wing geometry and grid generation, and boundary and initial conditions of the test cases. The final section describes the accuracy of the code by comparing its numerical results with experimental data and by examining the effect of various numerical parameters on the solution. The next two chapters discuss the features of the flow solution in more detail and explore the phenomena that characterize flow separation near the blunt leading edges.

6.2 Computational Code

The computer code used in the 3–D delta wing calculations was adopted from a 3–D Euler solver written by [Roberts 86] and modified by [Goodsell 87]. The original solver combined a multistage scheme with central differences as described in Sections 4.1 and 3.1. The inviscid scheme was extensively used, validated, and compared with

experiments in the above references.

The code was converted to a semi-implicit Navier-Stokes solver by adding thin layer Navier-Stokes terms (Section 2.1), boundary conditions appropriate for viscous flow (Section 3.2), and the *LHS* matrix (Equation 4.11) for the semi-implicit temporal integration. In addition, the residual smoothing found in the original code was adapted for semi-implicit operation (Section 5.1.2), and the "enthalpy damping" convergence acceleration mechanism found in the original code, which is unsuited for calculation of viscous flows, was removed.

6.3 Wing Geometry & Test Parameters

Two idealized semi-infinite wings with elliptical cross sections were chosen as test cases. "Wing T(hin)" has a thickness to span ratio of 1 : 20, giving an eccentricity of $\epsilon = 0.99875$. Test cases involving Wing T were constructed to compare with the conical Navier-Stokes solutions generated by Naomi McMillan at NASA Langley using Thomas' code[2]. "Wing F(at)" has a thickness to span ratio of 11.55 giving an eccentricity of $\epsilon = 0.99624$. Wing F was designed to allow examination of the effect of leading edge curvature via comparison of solutions obtained with Wing F and Wing T. Both wings have 65° sweep.

A number of laminar flows at a Reynolds number of 10^6 (based on chord), Mach number of 1.6, and angles of attack of 4, 8, and 12 were considered. The computational domain was limited to the volume between $x = 0.1$[3] and $x = 1.0$. All calculations were made at zero yaw; therefore, only half of the wing was computed and is shown in plots

[2]At the time of this writing, no results were available for comparison.

[3]See discussion of initial conditions in Sec. 6.5.

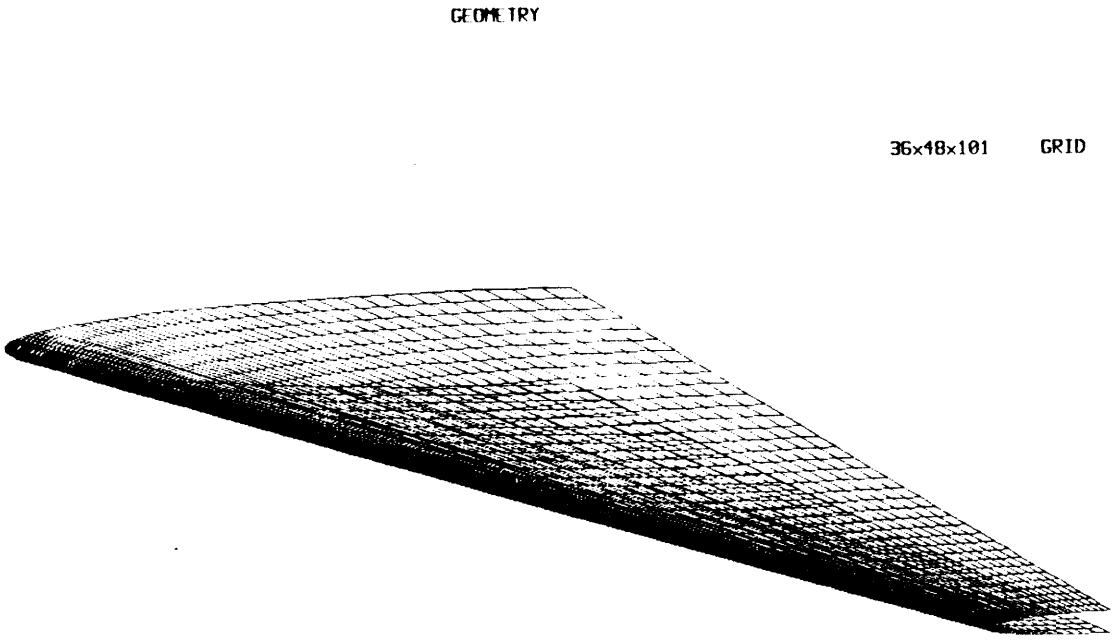

Figure 6.2: Idealized Delta Wing with Elliptical Cross Sections

of the results. Wing F as defined by a surface grid is shown in Figure 6.2.

6.4 Grid Generation

Three-dimensional grids around the wing were constructed by "stacking" two-dimensional grids at a number of streamwise stations. At each station the inner boundary of the grid is defined by elliptical cross sections and the outer boundary by ellipses centered some distance z' above the wing centerline. Each 2–D grid is calculated with the conformal transformation

$$y + iz = \varsigma = \frac{a^2 + b^2}{2\varsigma} \tag{6.3}$$

which transforms the elliptical inner and outer grid boundaries

$$\left(\frac{y}{a_i}\right)^2 + \left(\frac{z}{b_i}\right)^2 = 1 \text{ and } \left(\frac{y}{a_o}\right)^2 + \left(\frac{z}{b_o}\right)^2 = R_o \tag{6.4}$$

into circles of radius $1/2(a_i + b_i)$ and $R_o/2(a_o + b_o)$ in the complex ς plane. The outer boundaries of the grid and the center point z' are chosen such that the Mach cone emanating from the wing apex is captured. The grid is generated around the circle in the ς computational plane by forming a mesh of rays normal to the circle, with concentric circles at appropriate radial locations, and is transformed back into physical (y, z) coordinates via (6.3). Details of the grid generation process may be found in the Fortran code (Appendix G).

The (k), (j), and (i) indices are assigned to the circumferential, normal, and streamwise directions, respectively. With the minimum and maximum values of the indices being 1 and K_{max}, J_{max}, and I_{max}, respectively, the computational cells in the domain are indexed by $k = 1, K_{max} - 1$, $j = 1, J_{max} - 1$, and $i = 1, I_{max} - 1$.

Grid line families in each $2 - D$ section (j, k) of the 3-D grid are orthogonal, due to the conformal nature of the transformation. However, near the leading edges, the (j) and (k) families are not orthogonal to the (i) family of lines, since the wing sweep is not equal 90°. Because the derivation of the thin layer viscous terms assumes grid orthogonality, this may affect the accuracy of the solution. Unlike the extreme non-orthogonality that often occurs at complicated shapes such as wing tips, blunt trailing edges and wing-body junctions, the small degree of non-orthogonality of the present grid makes accuracy problems unlikely.

6.4.1 Normal, Circumferential, and Streamwise Grid Definition

Proper choice of radial (body-normal) location of the concentric circles is essential for constructing grids that resolve flow gradients in the wing boundary layer, in the vortex core, and in the farfield. The radial locations are assigned *via* an exponential

stretching in the computational plane:

$$\begin{aligned} rad_{j+1} = \;& rad_j + \Delta r_j \;, \quad \text{where} \\ \Delta r_j = \;& (1+\epsilon)\Delta r_{j-1} \;. \end{aligned} \tag{6.5}$$

The user defined stretching factor ϵ gives grid clustering near the boundary at $j = 1$ for $\epsilon > 0$. It should be large enough to provide adequate grid resolution in the boundary layer, but not so large that the high stretching seriously degrades solution accuracy.

Control of grid point location is enhanced by allowing ϵ to vary. In the calculations, ϵ was set at

$$\begin{aligned} \epsilon \;&= \epsilon_1 \qquad 1 \le j \le 15 \\ &= \epsilon_2 \qquad 15 < j \le J_{max} \;. \end{aligned} \tag{6.6}$$

These regions correspond approximately to viscous and inviscid flow zones. Typical values of the stretching coefficients were $\epsilon_1 = 0.2$ and $\epsilon_2 = 0.15$.

The circumferential distribution of grid points should result in high grid point density in regions of high flow gradients. In the vortical flows studied, this implies high grid point density at the leading edge and near the primary and secondary vortices and relatively low grid point density on most of the windward side of the wing. The circumferential spacing was controlled by separating the wing into windward and leeward sections and defining independent exponential stretchings, following (6.5), from the leading edge to the windward and leeward symmetry planes.

Spacing of streamwise stations was defined such that the ratio of spacing to streamwise distance remains constant:

$$\frac{\Delta x_i}{x_i} = \frac{\Delta x_{i+1}}{x_{i+1}} \;. \tag{6.7}$$

The stations were calculated by an iterative solution of Equation 6.7. With this definition the fractional streamwise spacing at any given station is invarient with respect

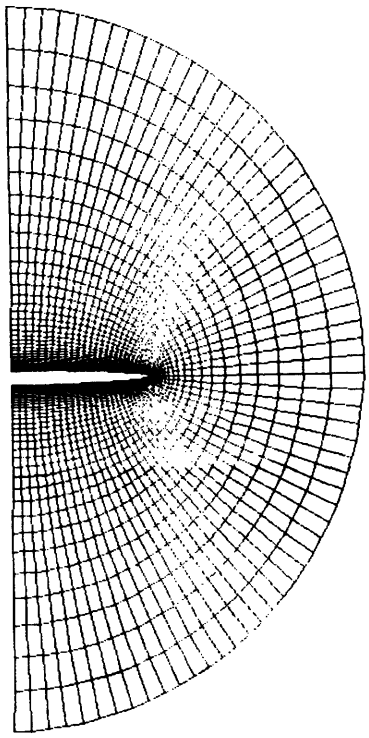

Figure 6.3: "Medium" Grid

to translation in the streamwise direction. The spacing is designed to maintain, for a fictitious conical flow, the conical nature of a numerical three dimensional solution.

One of two grids, "medium" (110,600 cells) or "fine" (172,800 cells), was used in the calculations. Both grids have 36 streamwise stations and 48 grid points in the body normal directions, computed with $\epsilon_1 = 0.2$ and $\epsilon_2 = 0.15$. The grids differ in the number and location of grid points in the circumferential direction. The medium grid has 65 points in the circumferential direction (Figure 6.3), spaced evenly in the computational plane. The fine grid has 101 points in the circumferential direction (Figure 6.4), of which 40 points are distributed with exponential stretching ϵ_w of 0.08 on the windward side (including a point at the leading edge) and 61 points with $\epsilon_l = 0.04$ on the leeward side. A plot of the surface fine grid about Wing F is given in Figure 6.2. Due to the perspective view, the wing dimensions are somewhat distorted. This grid gives excellent

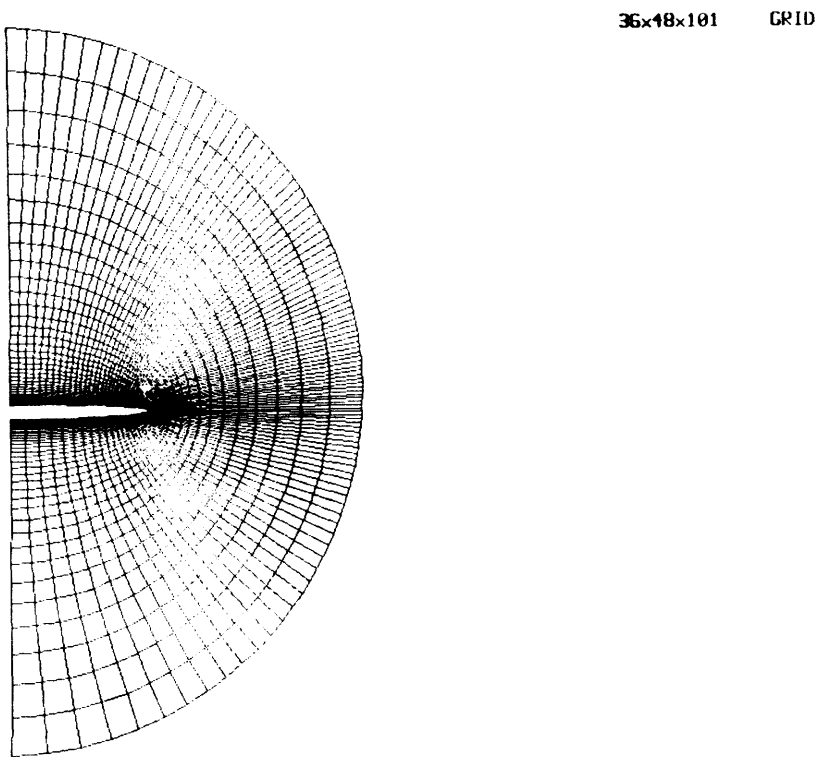

Figure 6.4: "Fine" Grid

resolution of the leading edge region, with 60% of circumferential grid points located in the outer 10% of span. Grid resolution at the leading edge of the medium and fine grids is shown in Figures 6.5 and 6.6, respectively. Locations of streamwise and circumferential node points on the body are given in Appendix F.

This choice of grid refinement was made after examination of solutions calculated on the medium grid. Velocity vectors in cross stream and streamwise planes demonstrated that the boundary layer is adequately resolved on the medium grid. Because streamwise variation of flow quantities is relatively small, the streamwise grid density is satisfactory. However, large flow gradients in the circumferential direction, especially at the leading edge, indicated a need for further refinement in this direction. Despite these conclusions, an attempt to further refine the body-normal grid was made. Due to computer resource limitations, calculations on this "extra fine" grid could not be completed.

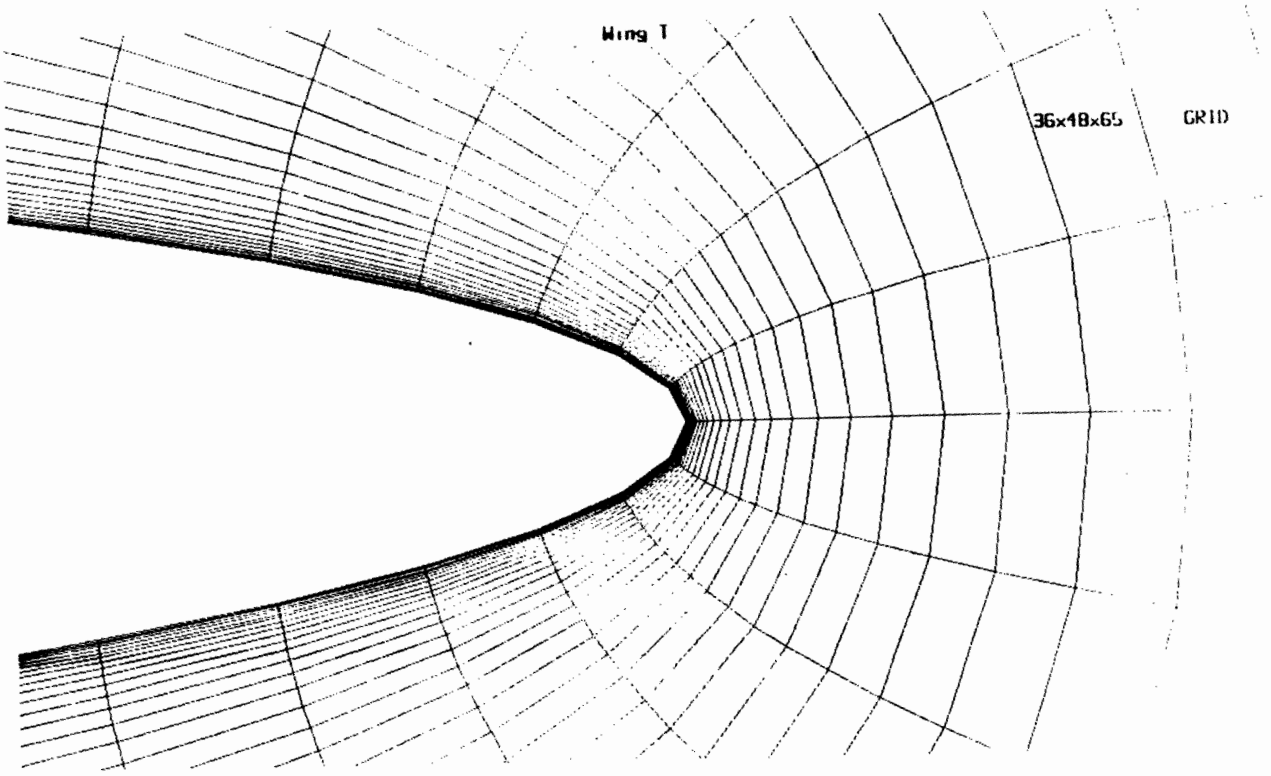

Figure 6.5: Leading edge area (95% span) of medium grid (T)

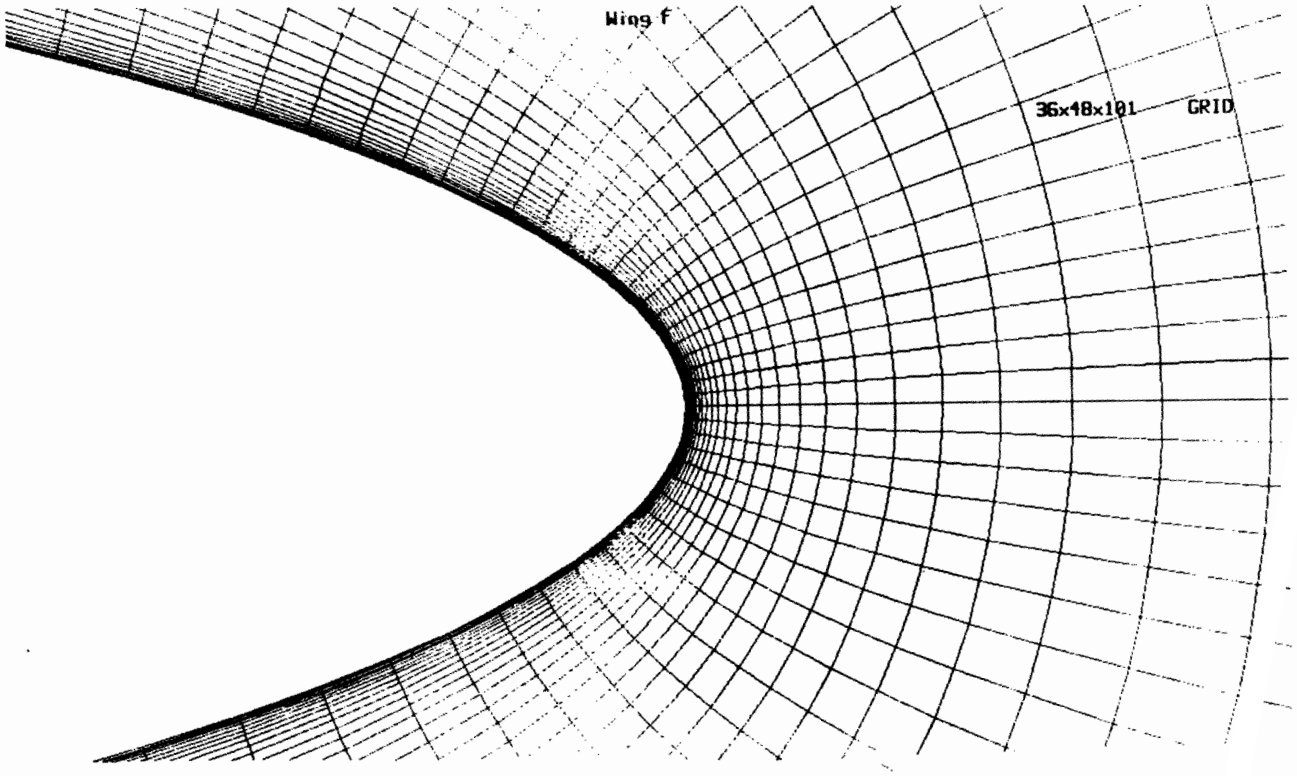

Figure 6.6: Leading edge area (97% span) of fine grid (F)

6.5 Boundary and Initial Conditions

Flow conditions at grid boundaries at the body and farfield ($j = 1, J_{max}$), first and final stations ($i = 1, I_{max}$), and at windward and leeward ($k = 1, K_{max}$) symmetry planes must be prescribed in a numerical calculation. Conditions at the body were discussed in Section 3.2. The symmetry boundary conditions implemented at $k = 1$ and $k = K_{max}$ do not require further explanation. Treatment of boundaries at $j = J_{max}$, and $i = 1, I_{max}$ is discussed below.

Characteristics theory dictates that flow variables in the inviscid supersonic flow at the outer surface ($J = J_{max}$) are imposed by the free stream. This condition is implemented by computing values of the state vector at a dummy layer of cells at J_{max}, $\mathbf{W}_{J_{max}} = 2\mathbf{W}_\infty - \mathbf{W}_{J_{max}-1}$, such that, upon forming the linear average with the variables at $J_{max} - 1$, the correct free stream values are obtained.

For computational efficiency, the idealized semi-infinite wings were truncated at $x = 0.1$ and $x = 1.0$, and inflow and outflow boundary conditions, respectively, were applied at these stations. The location of initial and final stations is arbitrary. Of importance is the ratio x_{max}/x_1, which defines the Reynolds number range of the solution. It is also important that the flow domain be large enough so that the effect of potential inaccuracies in the boundary conditions at the inflow is limited to a small percentage of the flow region.

The conical Navier-Stokes (CNS) code of [Ruffin 87] was used to calculate approximate flow conditions at the inflow. Ruffin's code solves the thin layer form of the equations with van Leer flux splitting and MUSCL differencing [van Leer 82]. It uses a pressure weighted flux limiter which makes the solution nominally (on Cartesian grids) second order accurate in smooth flow regions. However, [van Leer et al. 87] note that

this difference scheme is excessively diffusive in the boundary layer, which may result in inaccurate modelling of the primary and secondary vortices.

Two alternative methods of treating the upstream boundary conditions were considered but discarded. The "standard" appproach of modelling the wing from *apex* to trailing edge and of applying free stream conditions at some surface upstream of the apex was discarded because this approach requires a sophisticated grid generation procedure and is computationally more expensive due to the need to resolve high flow gradients at the wing apex. Even with sophisticated techniques, the requirement that a body-normal family of grid lines exist near the apex may be violated, and subsequent inaccuracies due to the thin layer Navier-Stokes approximation may be appreciable. Alternatively, free stream conditions could be applied directly at $x = 0.1$. While this option is simple, the boundary conditions are a poor model of the real flow at that station.

The grid at Station 1 ($x = 0.1$) was used as input geometry for the CNS solver. A CNS solution at the local Reynolds number was then obtained on the equivalent spherical computational surface. In most cases, a two or more order decrease in the magnitude of the residuals was achieved. However, in some cases convergence to steady state was questionable [4]. A similar procedure to calculate initial conditions was used by [Thomas & Newsome 86].

The results from the CNS calculation were used directly as boundary conditions at the planar grid at the inflow of the three dimensional domain. While the extrapolation from spherical to planar grid introduces numerical error, the use of the CNS solution clearly results in higher accuracy than the alternate assumption of freestream flow at

[4] Solutions at 12° and 14° did not converge further than 2E-4 after several thousand CNS iterations.

$x = 0.1$. In addition, the solution was used as initial conditions throughout the flow field. This removes the need to start the calculation impulsively, and thereby reduces computational cost.

The treatment of outflow boundary conditions for the Navier-Stokes equations is a topic of current research (eg. [Ababarnel et al. 86]). This research effort has not yet resulted in a "standard" approach comparable to the use of characteristics theory for Euler computations. The *ad hoc* approach used in this thesis was to compute values at outflow boundaries by extrapolation of the interior solution. Inspection of the flow solution near the outflow boundary showed that the linear extrapolation $\mathbf{W}_{Imax} = 2\mathbf{W}_{Imax-1} - \mathbf{W}_{Imax-2}$ modelled the variation of the flow variables more accurately than the first order approximation, $\mathbf{W}_{Imax} = \mathbf{W}_{Imax-1}$. It was therefore used in all flow calculations. This extrapolation is theoretically valid in supersonic zones of inviscid flow since it maintains the physical region of dependence inherent in this flow. It is also valid in the boundary layer if no reverse flow in the streamwise direction occurs. No significant retardation of the convergence rate of the algorithm due to the boundary conditions was observed.

6.6 Post-Processing and Graphical Display

Display and visualization of numerical flow data is crucial to gaining an understanding of the flow features and phenomena of any complex flowfield. The primary tool used in this respect was PLOT3D, a post-processing/graphics program developed at NASA Ames Research Center by [Buning & Steger 85]. Its capability of generating a variety of 2–D and 3–D line and color contour plots was employed to generate many of the plots presented below. A number of 2–D line plots of the pressure coefficients are also shown.

These were generated with the QPLOT program of [Kennelly 87].

A very useful feature of PLOT3D is its ability to generate 3–D particle paths. When the flow is steady, these particle paths are equivalent to streamlines. They are generated in PLOT3D by using trilinear interpolation of values of the vector velocity inside a computational cell and second order Runge-Kutta steps to advance the particle in time. In addition, projections of the velocity vectors onto a surface may be used to create "sectional" streamlines. This capability can be used to generate flow patterns on a surface that mimic oil flows used in experimental work. Both of these features were used extensively to analyze the flow field.

The structure of the flow field can also be explored by examining vector plots of the cross stream velocity (v, w). Although plots of the two-dimensional projections of three-dimensional vectors can be misleading, these plots are particularly useful in the present flow cases because the cross stream planes intersect the vortical structures at nearly 90°. Thus, most features of the vortices are visible and relatively free of distortion. Primary and secondary separation points can be easily located. A three dimensional "look" of the plots is obtained if the vectors are colored by a local scalar quantity such as the Mach number or pressure coefficient.

PLOT3D requires flow data at points corresponding to grid point locations. For this reason the numerical solution was pre-processed to transfer the solution from the centers of computational cells to cell nodes. In the interior of the computational domain this was achieved by forming a linear average of state vector values at the eight cells adjacent to each node. At boundaries of the domain, appropriate one-sided averages were computed. No pre-processing was done before application of the QPLOT program. Instead, the values of the pressure coefficient plotted were calculated directly from values of the state

vectors at cell centers. Due to the zero normal pressure gradient boundary condition (see Section 3.2) used at the body surface these values give the surface pressure.

6.7 Code validation

Validity of the three dimensional semi-implicit solutions was established by studying the convergence histories of the solutions, by comparing computed results with experimental data from [Rogers & Berry 55], and by considering the effect of various numerical parameters on the solution. To examine the influence of numerical parameters, the flow around Wing T at $M = 1.6$, $Re = 10^6$, and $\alpha = 4°$ was calculated with a number of different parameters, and compared to baseline results calculated on the medium grid with coefficients of artificial viscosity of $\kappa_2 = 0.1$ and $\kappa_4 = 0.01$.

All cases were run on a Cray-2 supercomputer of the National Aerodynamic Simulation (NAS) Program at NASA Ames Research Center. CPU time requirements for the fully vectorized codes on the fine grid was 21 seconds per iteration for the semi-implicit and 5 seconds per iteration for the explicit scheme.

6.7.1 Convergence to Steady State

Convergence of the solution to steady state was verified by examining the iteration history of the sum of residuals of the five discrete equation and by comparing pressure coefficients calculated at different iteration levels. Other measures of convergence, such as convergence of skin friction values, can be defined and are often used. However, previous two-dimensional numerical experiments showed that convergence of the skin friction was comparable to that of RMS residuals. Thus, the above measures of convergence

were deemed to be sufficient.

Both the explicit and semi-implicit schemes were run at the maximum CFL number for stability. However, it was found that convergence for separated flows ($\alpha \geq 4$) could be attained in the semi-implicit mode only if the time step used was set to a constant at each streamwise station. This phenomenon may be due to nonlinear behaviour of the separated flow which is not captured in the linear stability analysis given in Appendix A. Residual smoothing (in the (i) and (k) directions) was used in the semi-implicit scheme, but not in the explicit scheme where it did not seem to have much effect.

Semi-implicit calculations for Wing T at $\alpha = 4°$ on medium and fine grids were run 450 and 1300 iterations, respectively, and the explicit calculation on the medium grid was run 3200 iterations (Figure 6.7). Despite the high CPU cost of the semi-implicit (approximately 4.2 times higher per iteration than the explicit scheme), it outperforms the explicit scheme by a significant margin. As might be expected, the fine grid calculation with the semi-implicit scheme takes substantially more iterations to converge than the medium grid case. Because of its low convergence rate in medium grid calculations, the explicit scheme was not applied to any fine grid case.

Convergence of the iterations was verified by comparing solutions at several different iteration levels. Profiles of the pressure coefficient at the final station (35) after 200, 350, and 450 semi-implicit iterations (Figure 6.8) show a suction peak on the leeward side that indicates the presence of a primary vortex. The small and decreasing differences between the profiles indicate convergence of the solution.

Further comparisons of the efficiency of semi-implicit and explicit calculations can be made by considering the convergence histories of the $\alpha = 8°$ and $\alpha = 12°$ cases. Figures 6.9 and 6.10 confirm the improvement in efficiency given by the semi-implicit

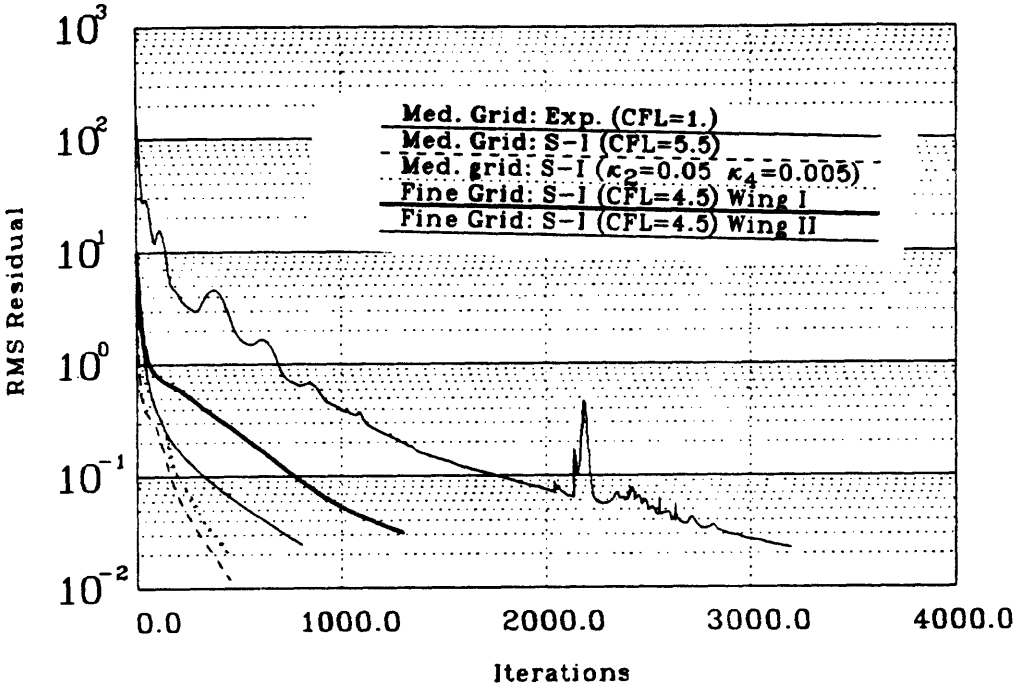

Figure 6.7: Semi-implicit and Explicit Convergence Histories, $\alpha = 4°$

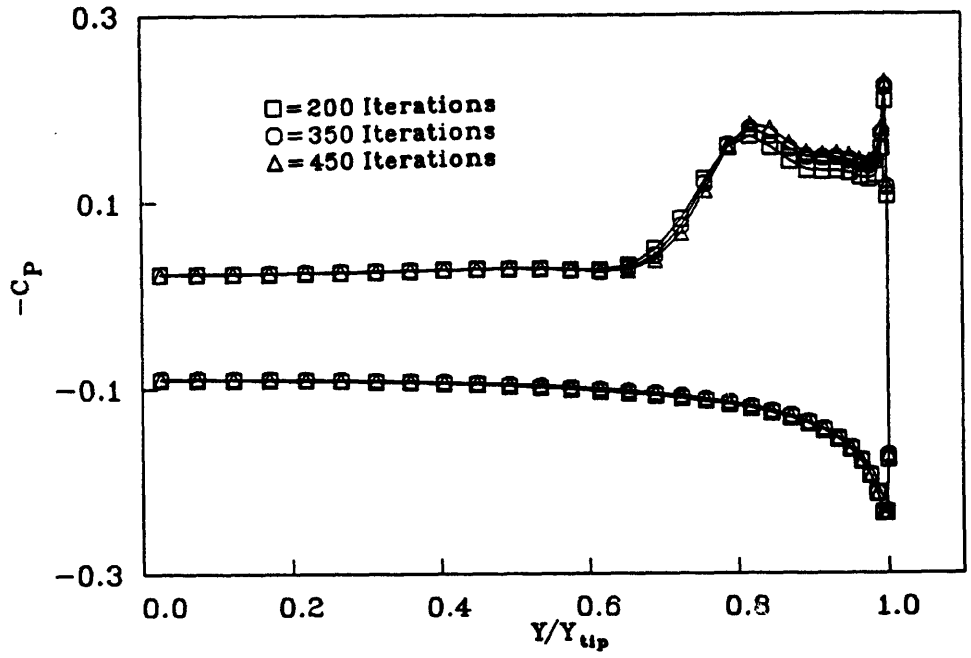

Figure 6.8: Convergence of Cp at Station 35, $M = 1.6$, $\alpha = 4°$, Med. Grid

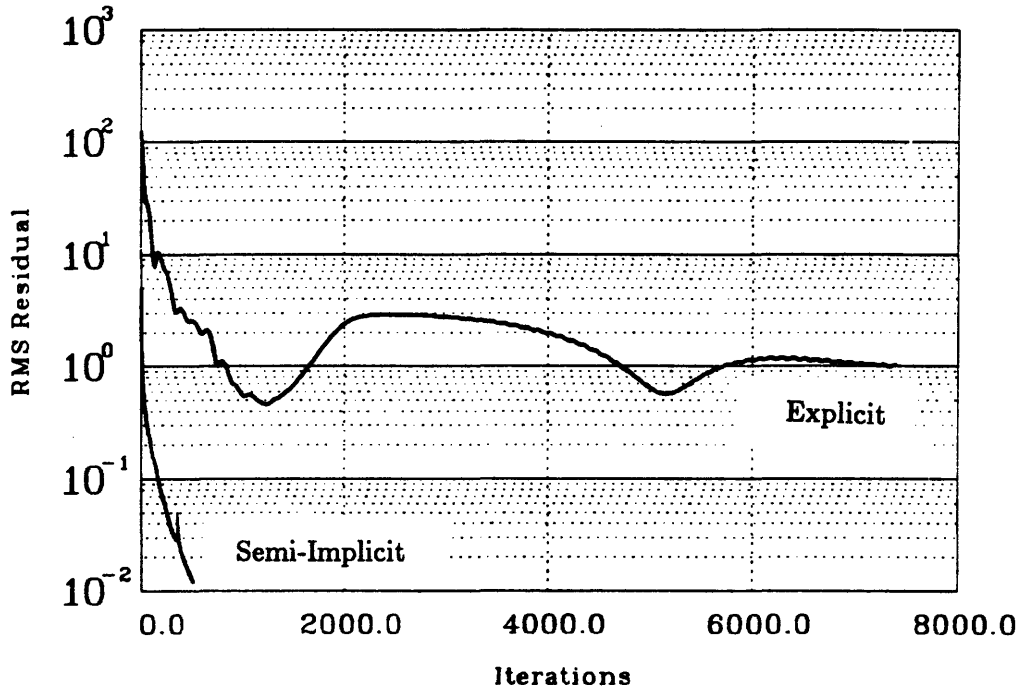

Figure 6.9: Semi-implicit and Explicit Convergence Histories, $\alpha = 8$, Med. Grid

scheme. In fact, the explicit scheme's convergence difficulties seem to increase as the angle of attack is increased. Thus, for $\alpha = 8°$ and 12°, the semi-implicit scheme is the only viable mode of calculation, both for medium and fine grid cases.

6.7.2 Comparison with Experiment

An experimental test case used by [Rogers & Berry 55] was used to examine the accuracy of the Navier-Stokes scheme. The geometry of the Rogers and Berry experiments corresponds to Wing F with 65° sweep. Mach number of the experiment was 1.41 and free stream Reynolds number was 2×10^6. An angle of attack of 14° was chosen for comparison. Unfortunately the experimental data indicated significant regions of turbulent flow which was not modeled in the calculations.

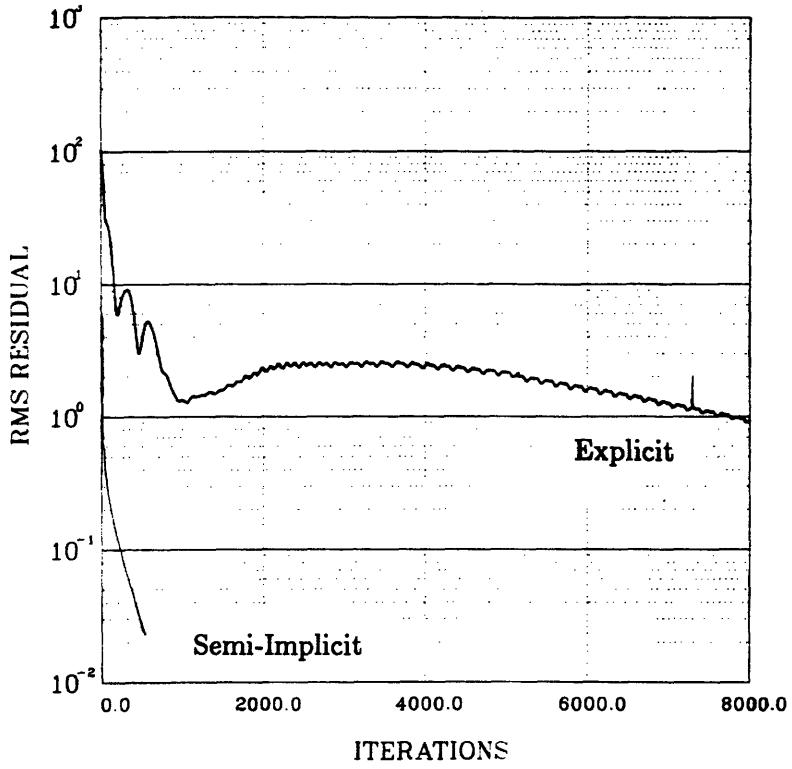

Figure 6.10: Semi-implicit and Explicit Convergence Histories, $\alpha = 12$, Med. Grid

A vector plot of the velocity in the cross stream plane (Figure 6.11) shows an elongated primary vortex that lies very close to the leeside boundary layer. The calculation was performed on the fine grid. Boundary layer separation occurs almost directly underneath the primary vortex and results in the formation of a secondary vortex outboard of its center. This secondary vortex appears to be absent in the experiments [Rogers & Berry 55]. It is likely that the turbulent mixing in the boundary layer prevented boundary layer separation in the experiment and the subsequent appearance of a secondary vortex. [Thomas & Newsome 86] note that the effect of turbulence on a flow about a elliptic cone at $M_\infty = 1.8$, $\alpha = 10°$, and $Re_L = 2.1 \times 10^6$, calculated with a CNS solver is to eliminate a tertiary vortex and to decrease the strength of the secondary vortex.

The effect of this discrepancy between experiment and computation is also seen in

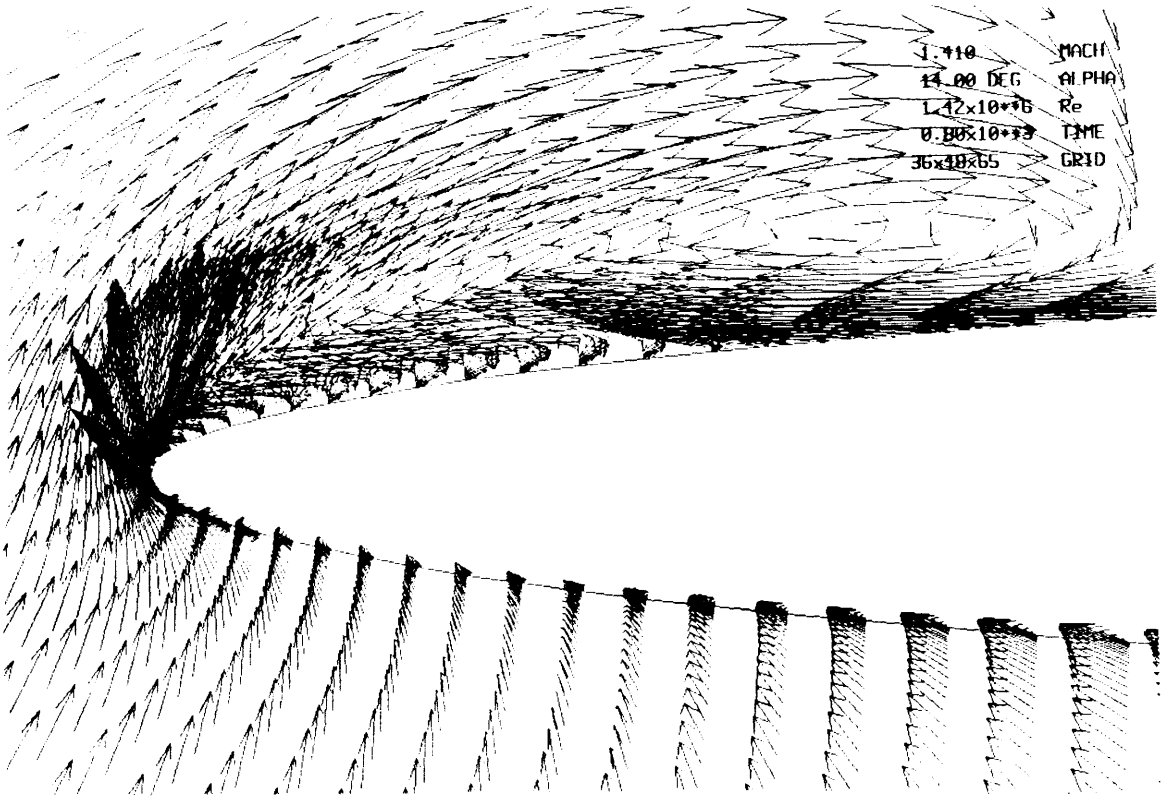

Figure 6.11: Velocity vectors for Rogers and Berry case[1]

comparison of pressure coefficients (Figure 6.12). Agreement is good on the windward side; however, the laminar computation overpredicts the pressure suction peak on the leeward side. Despite this, the location of the primary vortex and the behaviour of the pressure at the leading edge are predicted fairly well.

6.7.3 Effect of Artificial Viscosity

Artificial viscosity in a numerical scheme can have a significant effect on solutions of rapidly varying flowfields such as vortex flows. This effect is not physical because the artificial viscosity terms, unlike the "real" viscous terms contained in the Navier-Stokes equations, have no physical basis. The effect of artificial viscosity was examined by reducing the smoothing coefficients used in the baseline calculation by a factor of 2 from $\kappa_2 = 0.1$ to $\kappa_2' = 0.05$ and from $\kappa_4 = 0.01$ to $\kappa_4' = 0.005$.

[1]Med. grid solution. Subsequent plots are of fine grid solution.

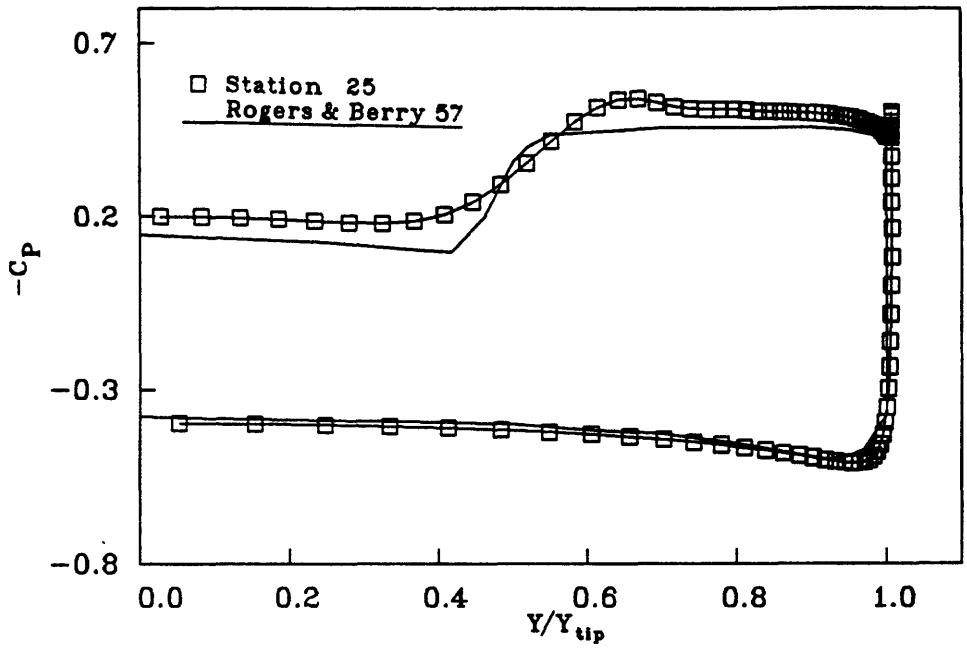

Figure 6.12: Pressure coefficients for Rogers and Berry case

Comparison of pressure coefficient profiles at Station 35 shows that the pressure suction peak is higher in the reduced smoothing solution than in the baseline solution. This indicates that the additional artificial viscosity in the basline solution acts to "smooth out" high velocity gradients, thus resulting in a weaker primary vortex. The profiles are virtually identical in other regions of the flow field.

6.7.4 Effect of Starting Solution

The effect of the starting solution was investigated by comparing the baseline solution with a solution calculated on a truncated wing (Figure 6.14), which begins at Station 11 ($x = 0.206$) of the baseline wing. Thus, the ratio x_{max}/x_1 for the truncated wing is approximately half that of the baseline wing. A starting solution at that station was calculated with the CNS scheme and the semi-implicit flow calculation was

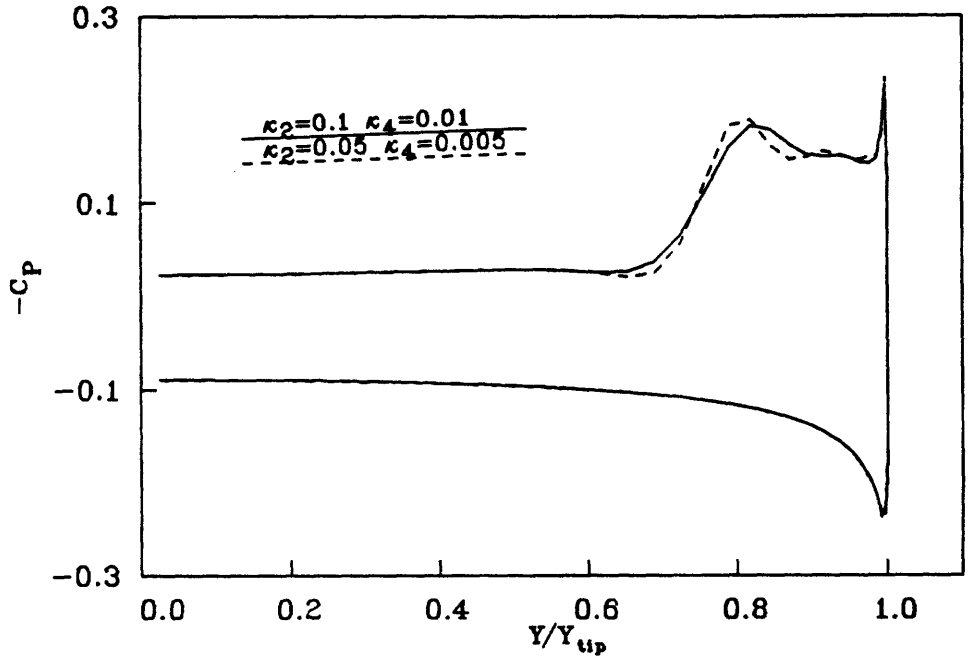

Figure 6.13: Effect of Artificial Viscosity on Solution

performed on the final 25 stations.

Differences between the two solutions are quantified by plotting the streamwise pressure coefficient profiles (Figure 6.15) at circumferential locations k=40 and 45. These locations are beneath the vortex in a region that is characterized by rapid surface pressure changes. There are consistent differences between the two sets of profiles. Errors in the starting solution can be expected to persist because, although numerical errors in viscous regions of the CNS solution are naturally dissipated due to the elliptic nature of the flow, errors in inviscid regions are propagated along characteristic lines without dissipation. The maximum difference between the two sets of profiles is approximately 2%.

The effects of artificial viscosity and location of the starting solution can be compared by superimposing profiles of the pressure coefficients. Figure 6.16 shows that the effect

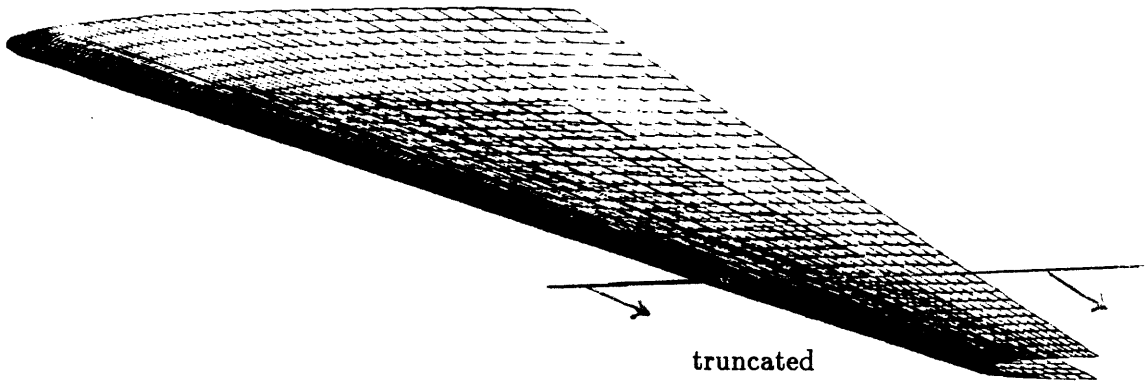

Figure 6.14: Baseline and truncated wings

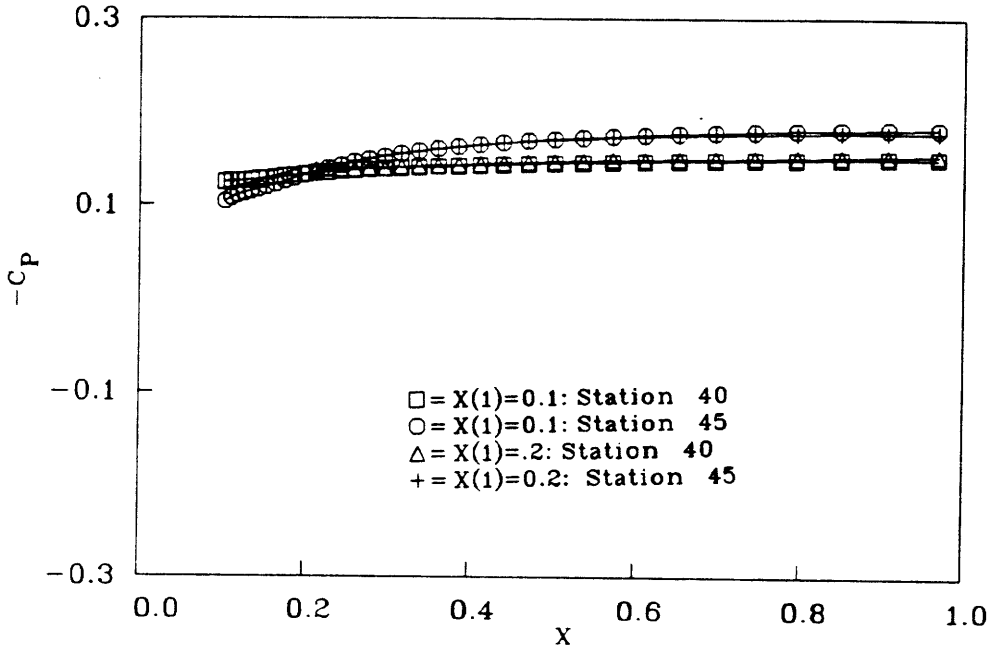

Figure 6.15: Effect of initial conditions: Streamwise C_P's at K=40,45

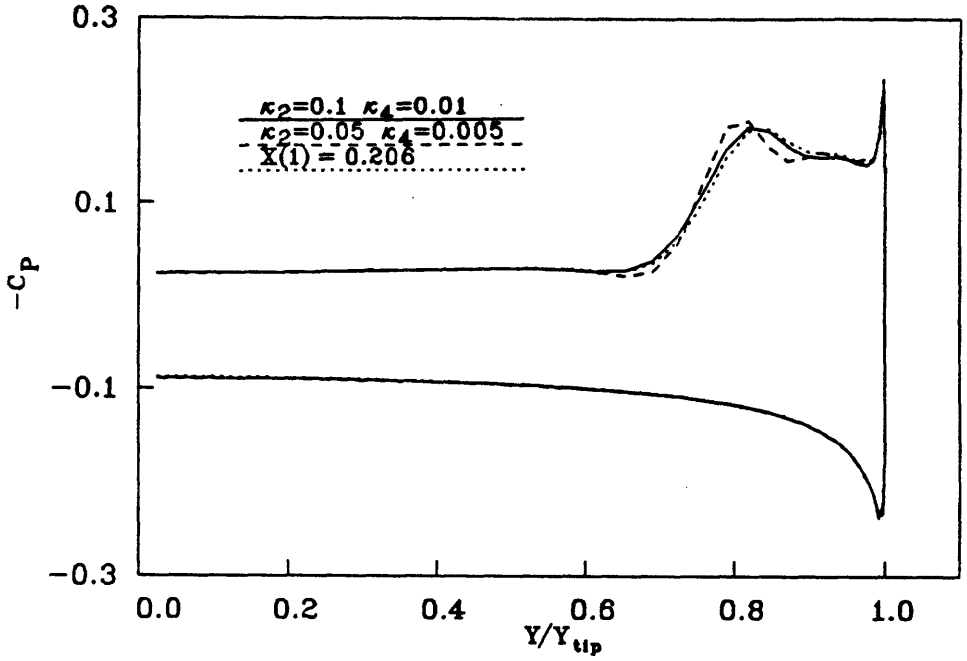

Figure 6.16: Effect of artificial visc. & init. conditions: C_p's at Station 35:

of a change the starting location is significantly smaller than the effect of reducing the artificial viscosity by a factor of two.

6.8 Effect of Grid Density

The effect of grid density on solution accuracy can be significant, especially when the flow field contains multiple spatial scales which must be resolved. In the present delta wing calculations, identifiable length scales are those of the body and displacement of the bow shock wave from the body, the thickness of the attached boundary layer, the dimensions of the primary and secondary vortex cores, and the spatial scales associated with location of the primary and secondary separation lines. In addition, other length scales not easily identified *a priori* may be present in a given flow and must be resolved.

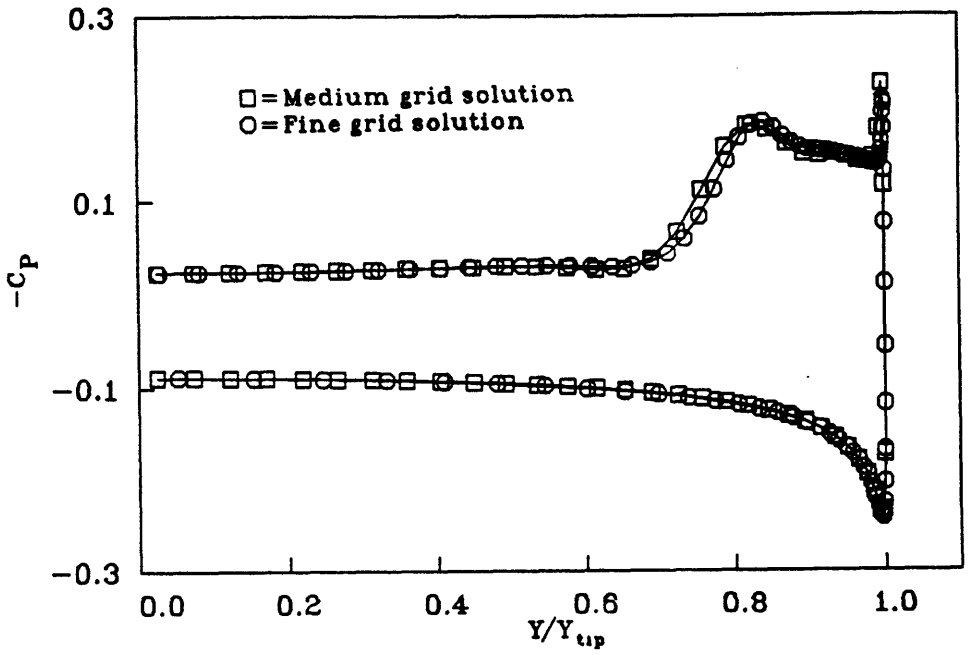

Figure 6.17: Pressure coefficients on medium and fine grid: Station 35

To determine whether the medium grid resolves the important features and scales in the flow field, the baseline flow calculation was repeated on the fine grid. The high resolution at the leading edge of the fine grid enables a more accurate modeling of the leading edge flow physics than is possible on the medium grid. However, profiles of the pressure coefficients at Station 35 calculated from the medium and fine grid solutions (Figure 6.17) show only small differences. The vortex in the medium grid solution is located slightly inboards of the vortex in the fine grid calculation. Both solutions indicate a leading edge negative pressure spike. The spike in the fine grid solution has smaller magnitude but, as might be expected, is more sharply defined. These flow structures at the leading edge are examined in detail in Chapter 8.

Chapter 7

Features of the Flow Solution

This chapter describes characteristic features of the laminar supersonic flow about delta wings with blunt leading edges. Flows over two 65° swept elliptical wings with thickness to chord ratios of 1:20 (Wing T) and 1:11.55 (Wing F) are considered at $M_\infty = 1.6$, $Re_L = 10^6$, and $\alpha = 4°$ and 8°. Solutions for these four cases were computed on a grid with 36, 101, and 48 points in the streamwise, cross stream and body-normal directions, respectively. Contour and vector plots of various solution variables are examined in Sections 7.1 and 7.2 to give insight into the flow topology.

Dominating feature of the flow is the massive separation and subsequent rollup of the boundary layer near the leading edge. This separated fluid forms elongated vortical structures – leading edge vortices – that lie very close to the leeward body surface. Separation of the boundary layer beneath the primary vortex, resulting in the formation of a secondary vortex, occurs in the $\alpha = 4°$ Wing T case and the $\alpha = 8°$ Wing F and Wing T cases. In the first and second of these cases secondary separation occurs some distance downstream of the inflow station, while in the third, it begins at the inflow station. No secondary separation is observed in the $\alpha = 4°$ Wing F case. The *mechanism* of leading edge separation – of primary interest in this thesis – appears to differ among the test cases. This topic is discussed in Chapter 8.

In the final section (7.3), accuracy and relevance of the conical assumption for the Navier-Stokes equations is discussed. For three of the cases studied, $\alpha = 8°$ Wing F

and $\alpha = 4°$ Wing T and Wing F cases, flow profiles vary considerably with streamwise location. This variation is due in part to the development of a secondary vortex, a process that can not be predicted by a solution of the CNS equations at a single station. For this and other reasons it is suggested that the CNS equations are a not very useful tool if details are to be adequately resolved.

7.1 Flow About Wing F and Wing T at $\alpha = 4°$

The flow about Wing F and Wing T at 4° angle of attack is considered in this section. First, the vortical structure of the flow field is investigated by examination of two- and three-dimensional particle paths. Then, variation of flow variables at several streamwise stations is shown in contours of Mach number and pressure and stagnation pressure coefficients. Finally, the topology of the flow as indicated by vector plots of the cross flow velocity is studied.

7.1.1 Vortical Structures

Vortical structures in a flow field can be identified and examined by plotting paths of particles released in the flowfield. If the initial locations of these particles are well chosen, the resulting patterns give a good sense of overall flow features. The 3–D particle plots below (Figures 7.1, 7.3) were generated by tracking particles released near the leading edge and in the boundary layer on the leeward side of Wing F and Wing T, respectively[1]. Primary vortices on the leeward sides of Wing F and Wing T are visible.

[1]Particles in Figure 7.1 were released at alternate streamwise stations and $k = 40$ and 55. Figure 7.3 was generated by releasing colored particles at alternate streamwise stations at the circumferential stations $k = 40$ (green) and $k = 64$ and 68 (blue or red).

No secondary separation in the flow about Wing F can be seen[2]. Secondary separation does occur in the flow about Wing T (Figure 7.3). It results in a secondary vortex which lies within the leeside boundary layer,

These three dimensional plots are complemented by particle paths projected on the cross stream (y-z) plane at Station 36 of each flow case (Figures 7.2, 7.4). The patterns simulate cross stream streamlines. The plots are generated by releasing particles at a number of body normal and circumferential grid point locations and integrating the projection of their spatial motion onto the y-z plane. The local slope of each path at any y-z location gives the 2–D flow direction at that point.

The vortex above Wing F lies very close to the body. It is weak, and can barely be seen in 2–D particle particle paths (Figure 7.2). Flow acceleration due to the vortex area is small and the peak Mach number (1.82) of the flow is only slightly higher than that of the freestream.

The higher leading edge curvature of Wing T results in a stronger leading edge vortex (Figures 7.3, 7.4) than found in the Wing F flow. The core of this vortex is well defined and lies just above the leeside boundary layer at approximately 82% chord.

Boundary layer separation and the secondary vortex in the Wing T case develop at approximately Station 30. This is highlighted in Figure 7.3 by use of the color red for particles released at $k = 64\&68$ downstream of Station 32.

Another view of the flow topology in two cases can be obtained by examining plots of numerical "oil flow" calculations. These plots are generated by projecting paths of particles released just above the wing surface onto that surface. (The same process in the

[2]This was verified by scanning the leeside flow field by releasing particles at a number of circumferential locations.

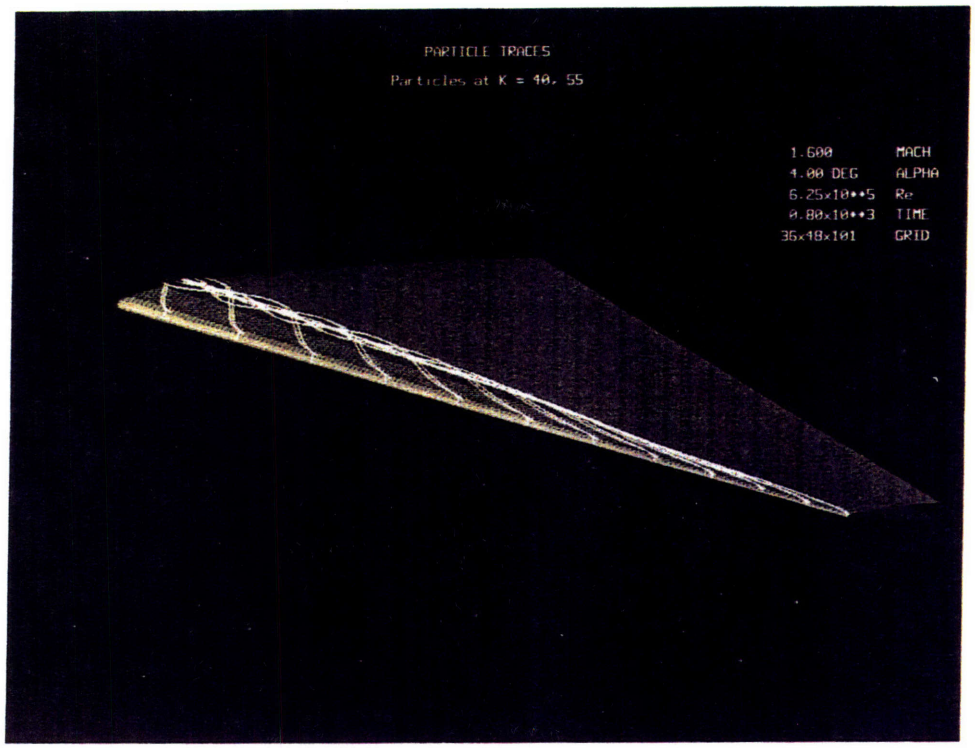

Figure 7.1: Vortical Structures: Wing F, $\alpha = 4°$

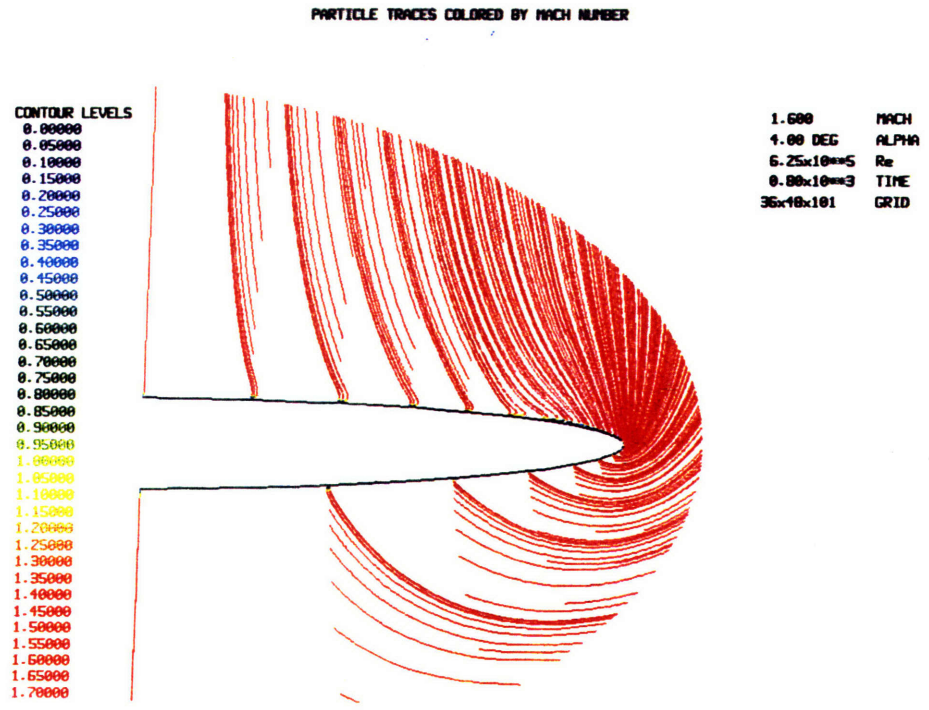

Figure 7.2: Primary vortex in cross flow plane: Wing F, $\alpha = 4°$

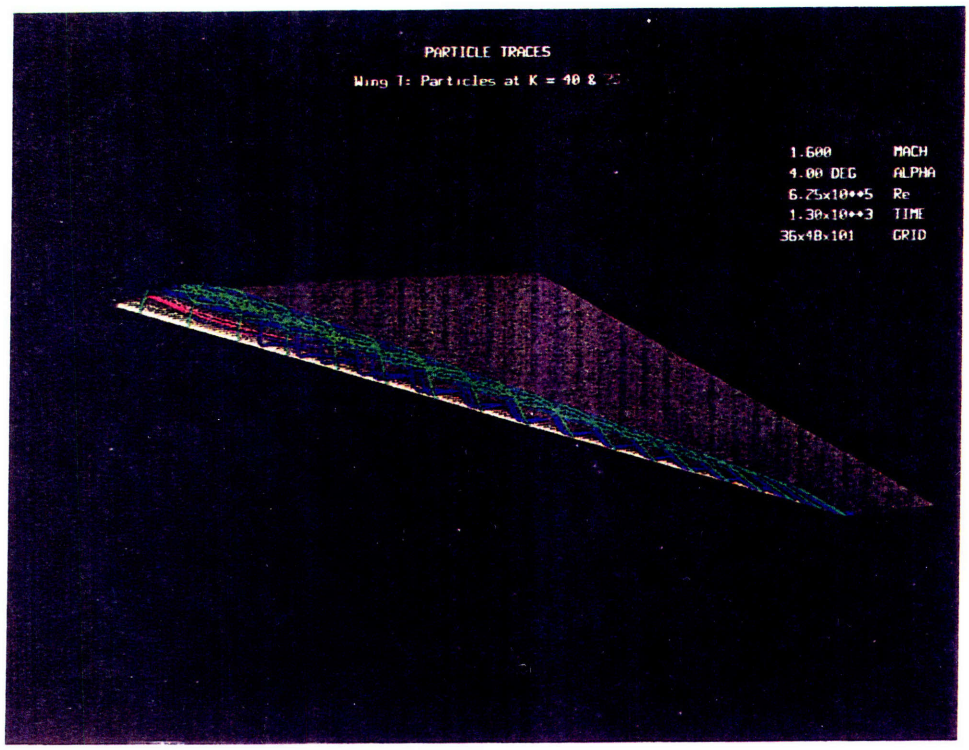

Figure 7.3: Vortical Structure: Wing T, $\alpha = 4°$

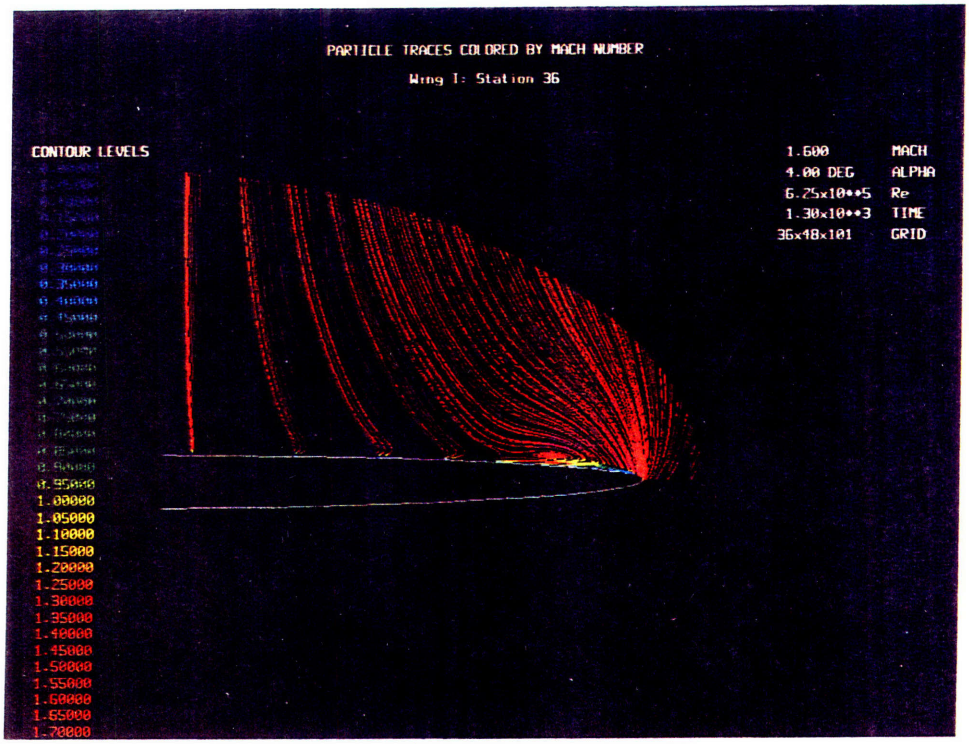

Figure 7.4: Primary vortex in cross flow plane: Wing T, $\alpha = 4°$

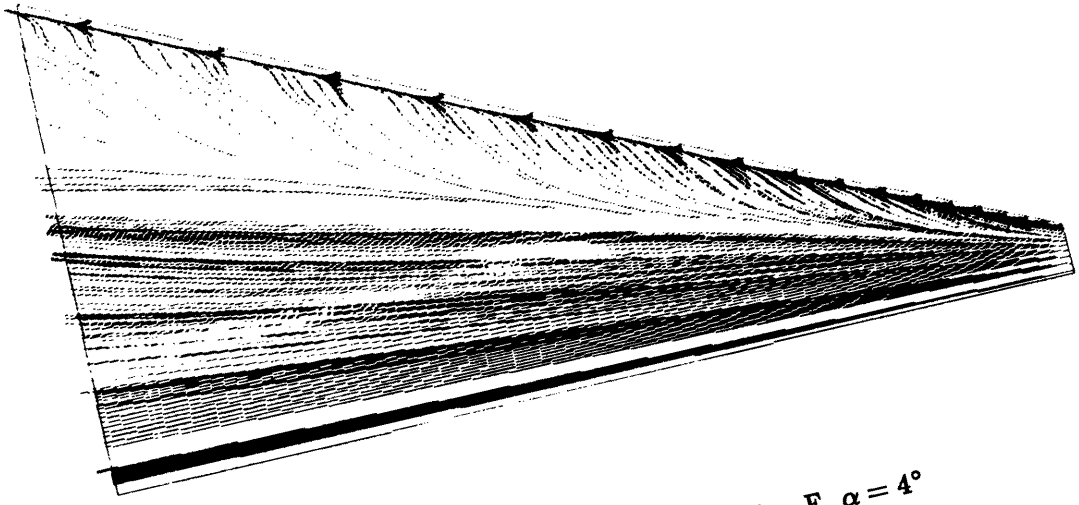

Figure 7.5: Numerical Oil Flow: Wing F, $\alpha = 4°$

y-z plane was used to generate Figures 7.4 and 7.2.) These "sectional" streamlines mimic oil flow patterns often used for surface flow visualization in wind tunnel experiments. Boundary layer separation is shown by a confluence of surface lines such as those just inboard of the leading edge of Wing F (Figure 7.5), and those at the leading edge and at 85% chord (starting at approximately 60% span) of Wing T (Figure 7.6). In addition, the streamwise flow inboards of the primary vortex reattachment line and the vortex-induced outboard-spanwise flow can be identified. However, numerical oil flow calculations do not respond to pressure fluctuations as do real oil flows. Thus, some features such as location of the primary vortex core are not indicated in these plots.

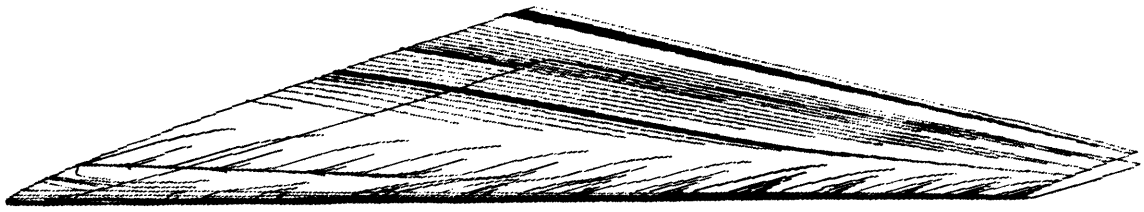

Figure 7.6: Numerical Oil Flow: Wing T, $\alpha = 4°$

7.1.2 Mach and Pressure Contours

Contour plots of solution quantities such as the Mach number and pressure and stagnation pressure coefficients complement the global view of a flow field given by the two- and three-dimensional particle paths shown above and allow close examination of flow features. Figures 7.7 and 7.8 show global and local contours of Mach number at the Station 36 ($x = 1$) of Wing F. The weak bow shock visible on the windward side (Figure 7.7) begins to turn the incoming $\alpha = 4°$ flow towards a direction parallel to the body. The shock develops into an expansion fan on the leeward side. The "local" contours display the solution for $k = 40,101$, $j = 1,40$. Strong Mach number gradients are evident at the leading edge and, of course, in the boundary layer.

Contours of the pressure and stagnation pressure contours at Station 36 of Wing

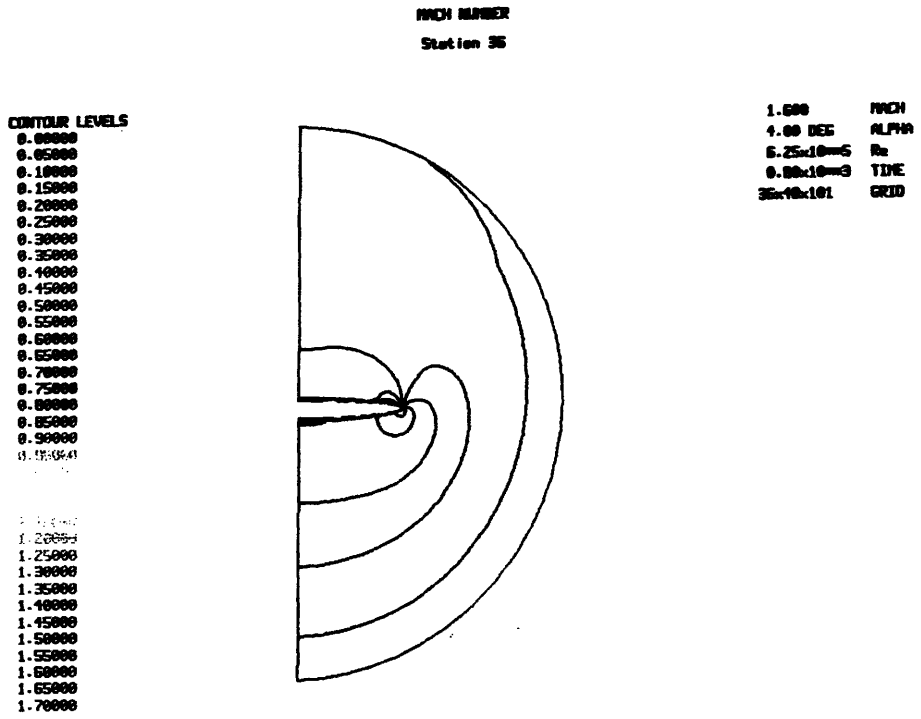

Figure 7.7: Mach number contours at Station 36: Wing F, $\alpha = 4°$

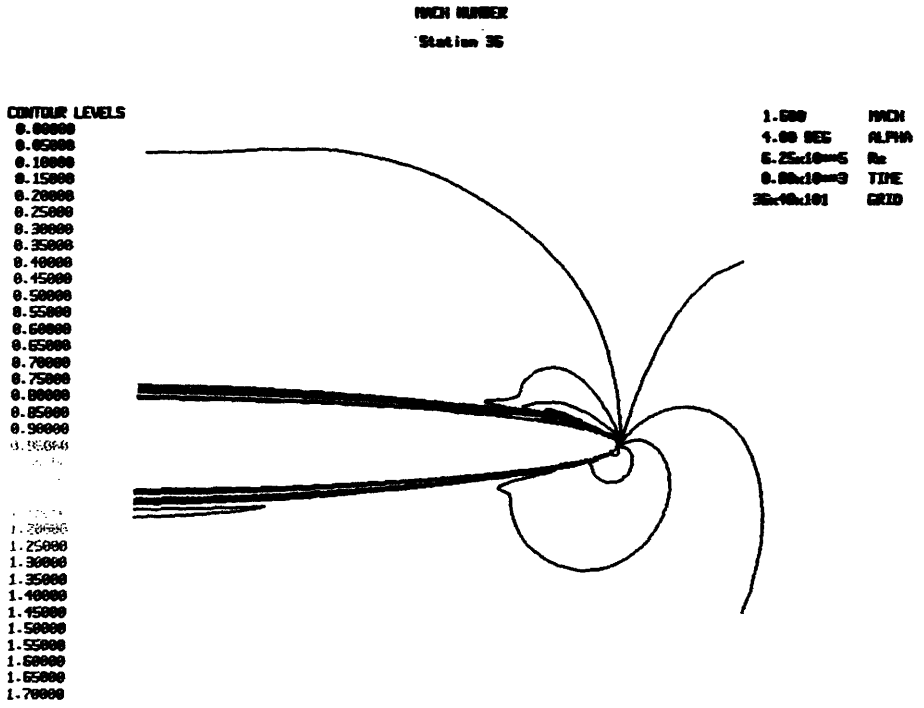

Figure 7.8: Close-up of Mach number contours: Wing F, $\alpha = 4°$

F are given in Figures 7.9 and 7.10, respectively. The high concentration of pressure coefficient contours near the leading edge indicate high pressure gradients in this region. Stagnation pressure losses help identify regions of vortical flow, such as found in boundary layers and vortices[3]. In this case (Figure 7.10), the primary vortex is too weak to be clearly identified, although a small bulge in the leeside contours can be seen.

Similar results for Wing T are given in Figures 7.11 (Stations 5, 15, 25, 35) and 7.12 (Stations 36). The maximum Mach number in this flow solution is 1.85 and the minimum pressure coefficient is -0.26.

[3]The definition of the stagnation pressure coefficient is identical to that of the pressure coefficient with the pressures replaced by stagnation pressures.

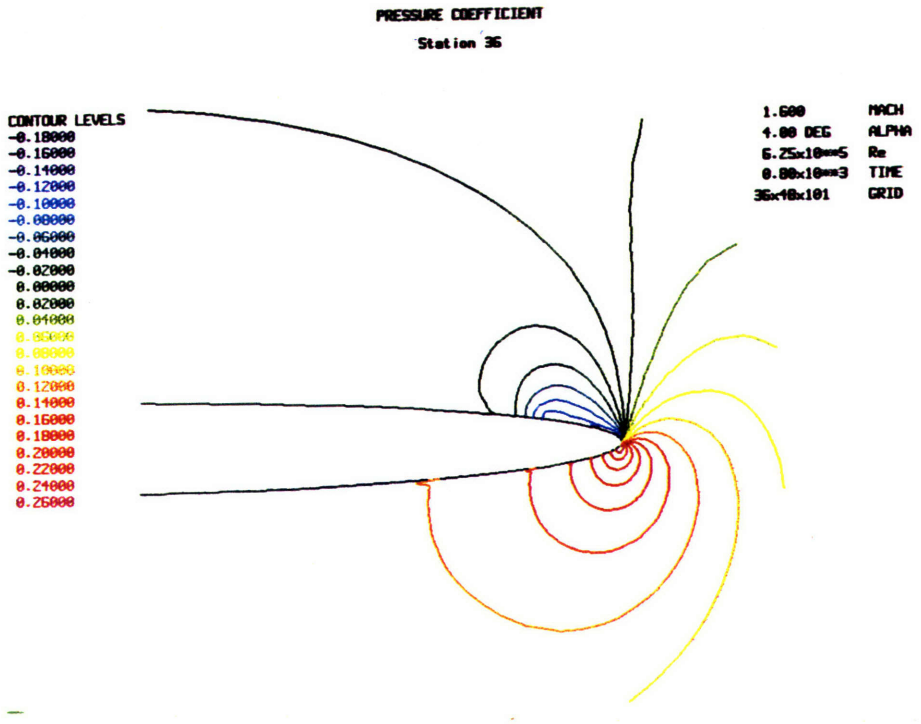

Figure 7.9: Pressure coefficient contours: Wing F, $\alpha = 4°$

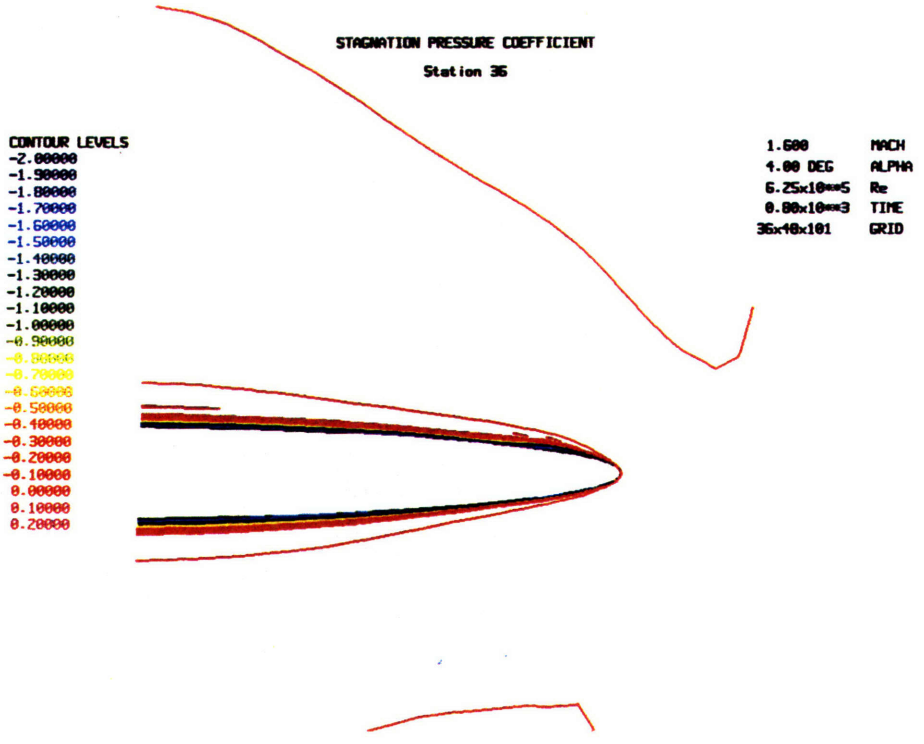

Figure 7.10: Stagnation pressure coefficient contours: Wing F, $\alpha = 4°$

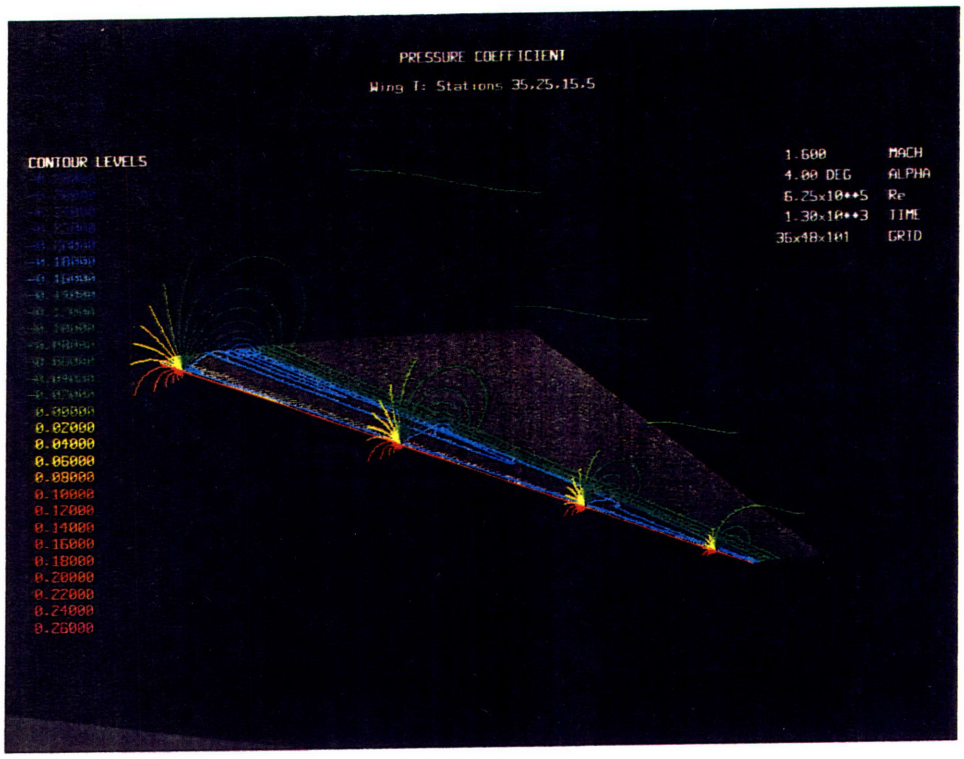

Figure 7.11: Pressure coefficient contours: Wing T, $\alpha = 4°$

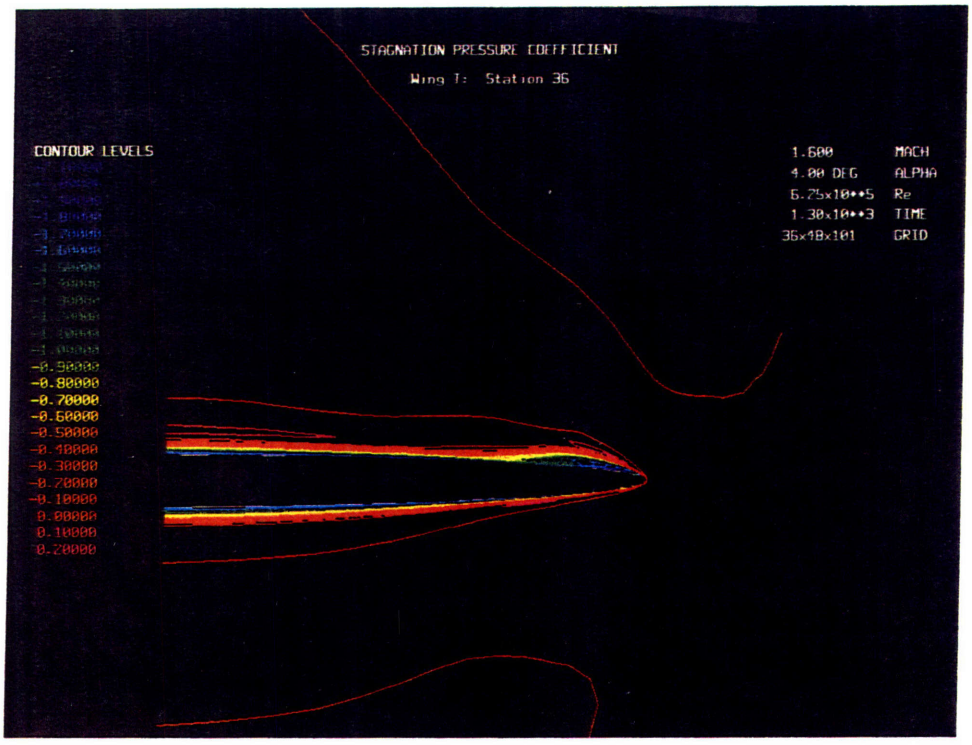

Figure 7.12: Stagnation pressure coefficient contours: Wing T, $\alpha = 4°$

7.1.3 The Cross Stream Velocity Field

Although velocity fields in all of these solutions are clearly three-dimensional, useful insights can be obtained by examining cross stream velocities. A convenient way of doing so is to graph $v - w$ velocity vectors in the region of interest. It is important to remember, however, that in most cases the streamwise component of the velocity is substantially larger than the displayed cross stream vectors. A sense of three-dimensionality is obtained when the velocity vectors are colored by a local scalar quantity such as the Mach number.

Figure 7.13 gives a global view of the cross stream velocity vector field, colored by Mach number, at Station 36. Large cross stream velocities near the leading edge are evident. However, as shown by the small size of the differences between the maximum local (1.82) and freestream (1.6) Mach numbers, the cross stream velocity is significantly smaller than the streamwise component of the velocity.

Velocity vectors at the leading edge and in the vortex area are given in Figure 7.14. They are colored by the local pressure coefficient. Large gradients of the velocity and pressure coefficient are again evident at the leading edge. As noted above, secondary boundary layer separation does not occur in Wing F flow at $\alpha = 4°$, and indeed no reverse flow in the boundary layer can be seen. This is not the case in the $\alpha = 4°$ Wing T solution (Figure 7.15), where a very small reverse flow region can be detected outboard of the primary vortex.

Formation of the secondary vortex in the case of Wing T may be explained by using a two-dimensional analogy in cross flow planes. The thinner wing has a relatively strong vortex which causes a high pressure peak and adverse pressure gradient immediately

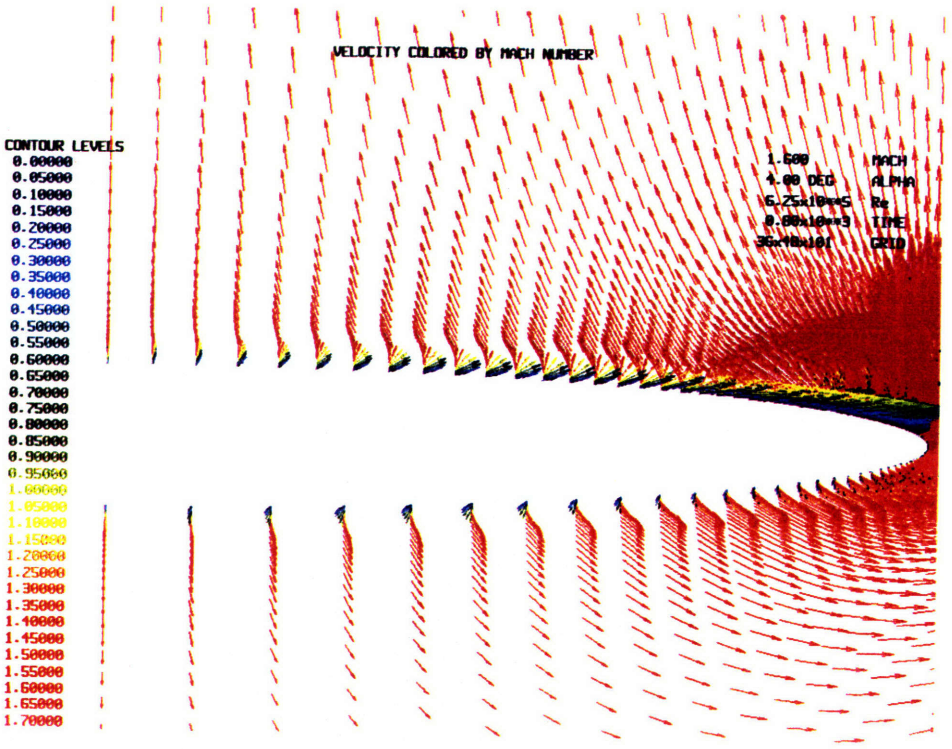

Figure 7.13: Cross stream Velocity Vectors: Wing F, $\alpha = 4°$

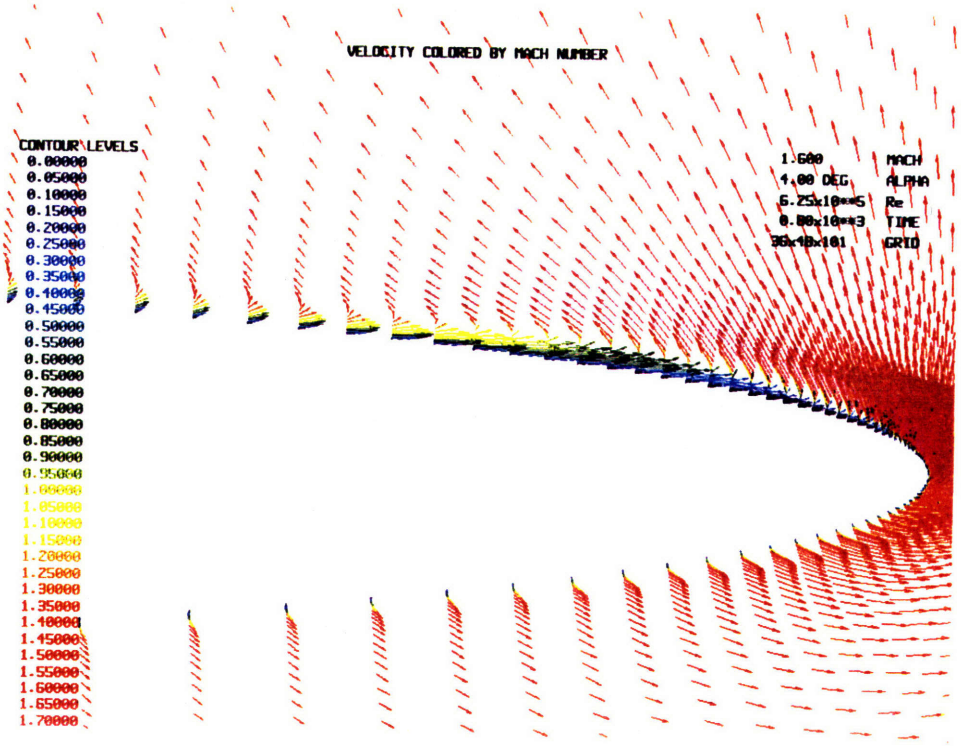

Figure 7.14: Velocity Vectors near Leading Edge: Wing F, $\alpha = 4°$

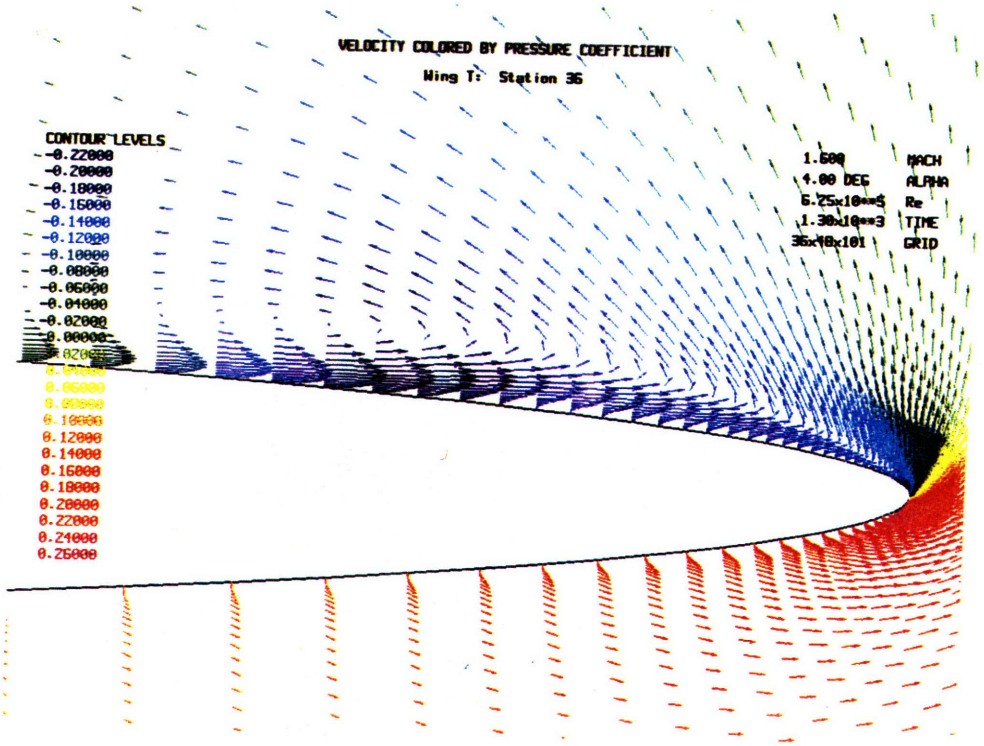

Figure 7.15: Velocity Vectors near Leading Edge: Wing T, $\alpha = 4°$

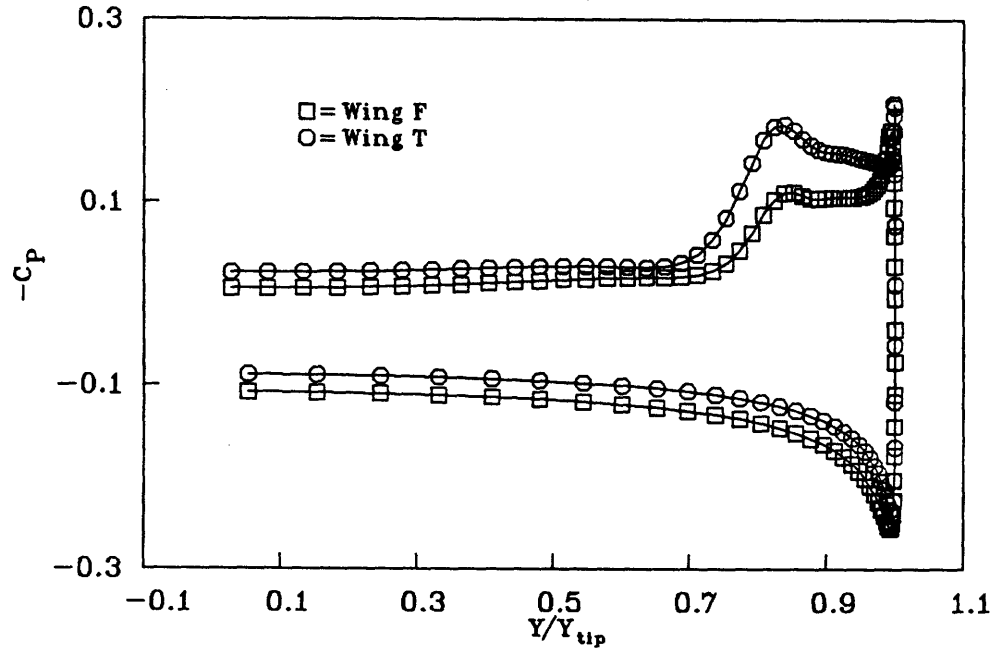

Figure 7.16: C_p profiles at Station 35: Wing T & F, $\alpha = 4°$

outboard of the primary vortex (Figure 7.16). The outward flowing fluid can not negotiate this pressure rise and separates to form a secondary vortex. The adverse pressure gradient on Wing F, on the other hand, is too mild to induce secondary separation. Separation processes at the leading edge will be discussed in Chapter 8.

7.2 Solutions for Wing T and Wing F at $\alpha = 8°$

In general, the flows around Wing F and Wing T have stronger, better developed vortex systems than the corresponding flows at $\alpha = 4°$. Both of the 8° cases have well defined primary and secondary vortices. Flow topology and characteristics are discussed in a manner similar to Section 7.1, below.

7.2.1 Vortical Structures

The vortex system on the leeward side of Wing F at 8° (Figure 7.17) has a primary vortex present over the entire chord of the wing and a secondary vortex beginning at approximately 30% chord[4]. The core of the leading edge vortex is located just above the leeside boundary layer. The sense of rotation with respect to an upstream view point of the left portion of the wing is clockwise for the primary vortex and counterclockwise for the secondary vortex.

The core of the primary vortex is clearly visible at approximately 70% chord in the sectional streamlines (Figure 7.18). A maximum Mach number of 2.01 is at $(j,k) = (24,81)$ in the feeding sheet just above it. The secondary vortex can not be seen clearly, because it is much weaker than the primary vortex and located in the boundary layer very close to the wing.

Three-dimensional particle paths and two-dimensional particle path projections are combined in Figure 7.19 to give a composite picture of the vortical structure and surface streamlines. The dark blue particle paths correspond to low Mach number flow and are

[4]Particles in this plot were released at the $k = 40$ (leading edge) and $k = 75$ circumferential locations at every third streamwise station.

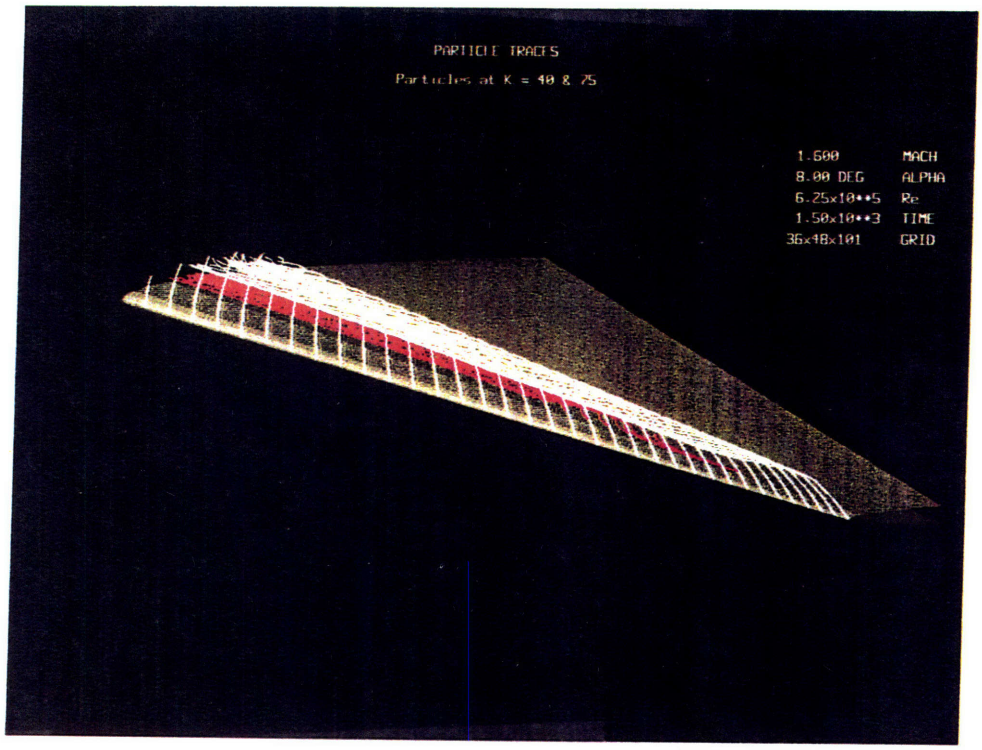

Figure 7.17: Vortical Structures: Wing F, $\alpha = 8°$

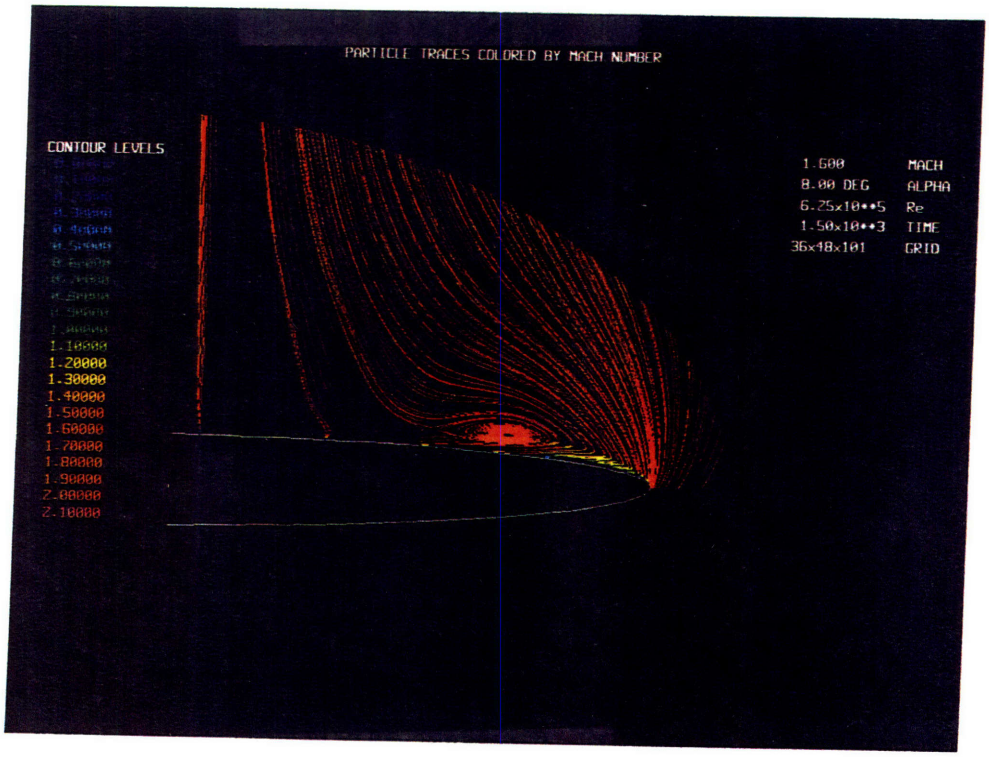

Figure 7.18: Primary vortex in cross flow plane: Wing F, $\alpha = 8°$

due to the particle path integrations constrained to the $j = 2$ surface ("oil flows"). Convergence of these lines shows that secondary separation on Wing F occurs just outboard of the primary vortex core. The core of the secondary vortex (as indicated by green and light blue particle paths) is located few points outboard of the secondary separation line. Although the separation line is present even at the first stations of the flowfield no secondary vortex appears to be present at this point.

An interesting feature of the flow in all of these cases is its directionality in the windward boundary layer. Oil flow lines on the windward surface (Figure 7.20) indicate that the direction of the flow in the windward boundary layer is towards the wing symmetry plane. This is a result a slightly decreasing pressure towards the windward symmetry plane (Figure 7.21). However, flow outside of the boundary layer actually has a slight directionality towards the leading edge, as might be expected with this geometry and angle of attack. On the leeward side (eg. Figures 7.5 & 7.6) the flow in the boundary layer inboard of the primary vortex is approximately parallel to the body centerline.

Figure 7.22 shows the vortex system of the Wing T solution at $\alpha = 8°$. The effect of the higher leading edge curvature at $\alpha = 8°$ is similar to that at $\alpha = 4°$. The leading edge vortex is somewhat stronger than that in the Wing F flow, and secondary separation begins a short distance downstream of the inflow solution.

7.2.2 Mach and Pressure Contours

As might be expected, vortices in flows around Wing F and Wing T at 8° angle of attack are stronger than those in the $\alpha = 4°$ flow. Mach and pressure coefficient contour plots for Wing F (Figure 7.23, 7.24) and Wing T (Figure 7.25, 7.26) show higher peak

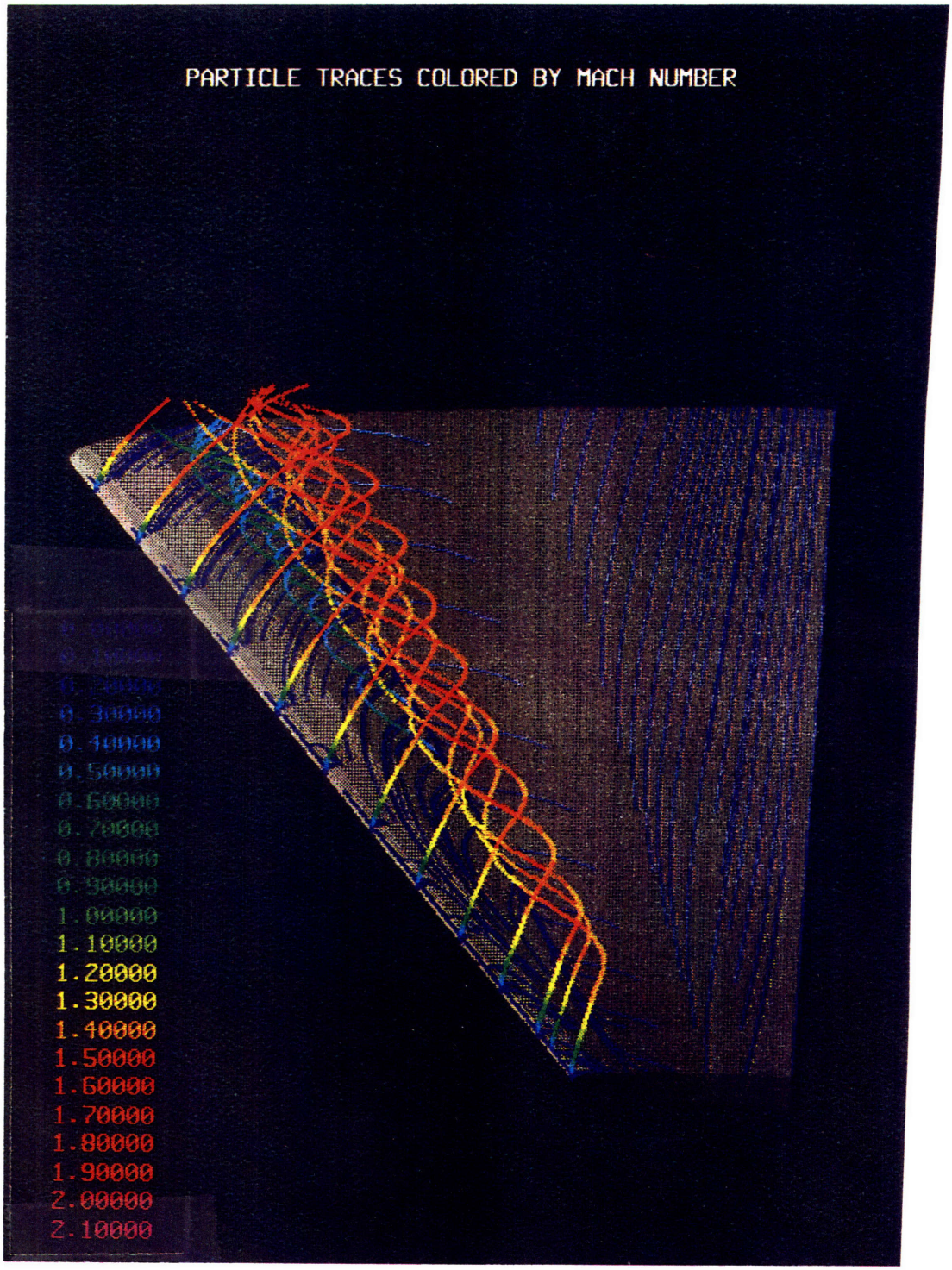

Figure 7.19: 3–D/2–D Composite particle paths: Wing F, $\alpha = 8°$

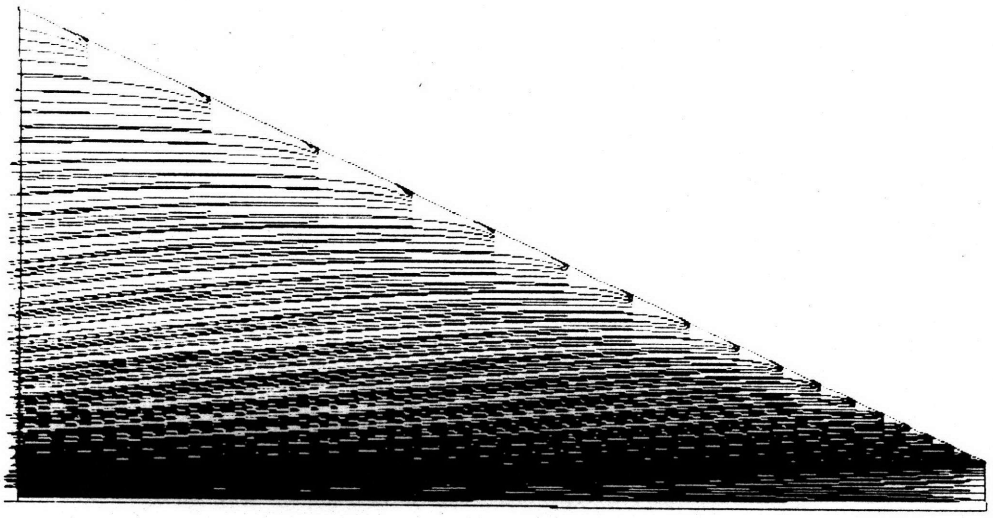

Figure 7.20: Numerical oil flow on windward surface: Wing F, $\alpha = 8°$

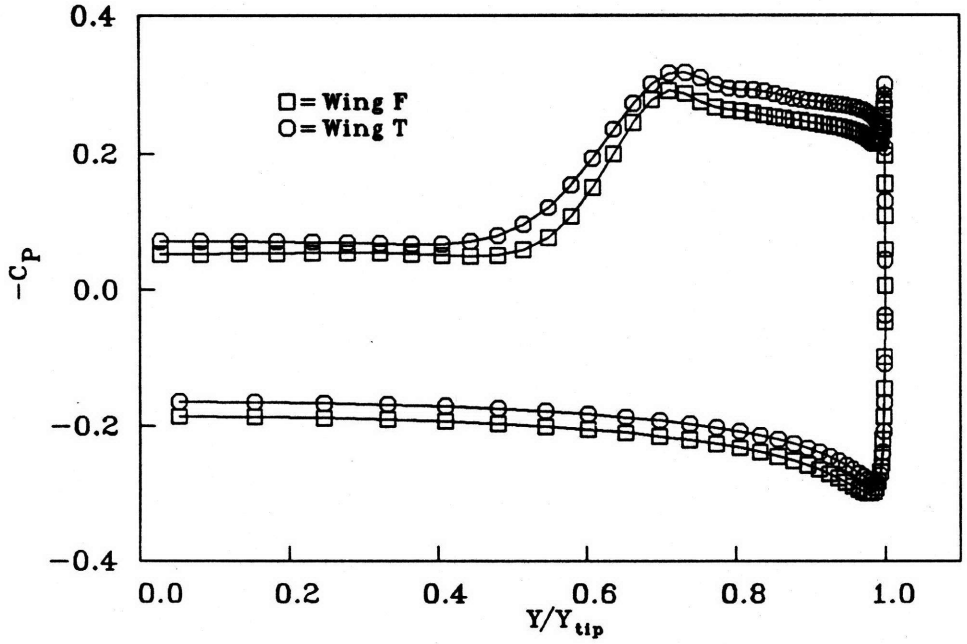

Figure 7.21: Pressure profile at Station 35: Wing F, $\alpha = 8°$

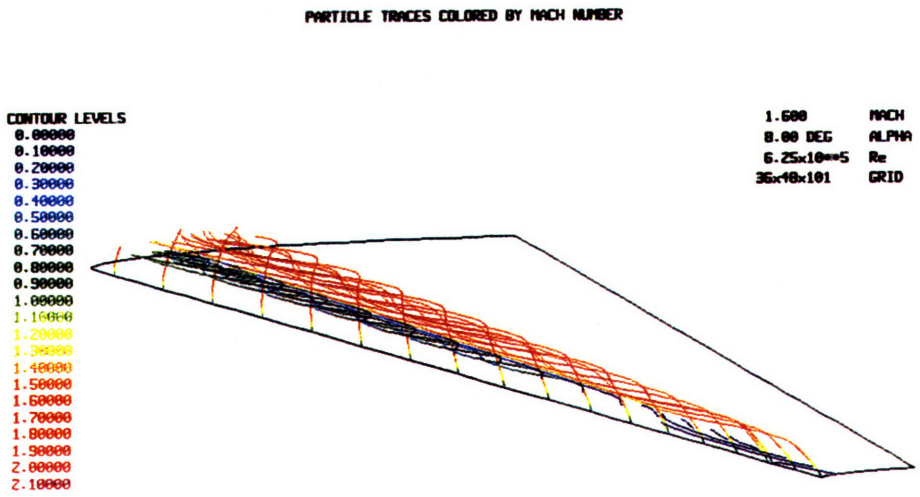

Figure 7.22: Vortical Structures: Wing T, $\alpha = 8°$

Mach numbers and steeper pressure gradients in the vortical region than seen in the $\alpha = 4$ solutions.

Similar results for Wing T are given in Figures 7.25 and 7.26.

7.2.3 Cross Flow Velocity Vectors

Figure 7.27 gives a global view of the velocity vectors at the leading edge of Wing F and near the primary and secondary vortices. The vectors are colored by the local pressure coefficient. The cross stream velocities at the leading edge are large and the feeding sheet is visible. The core of the primary vortex and the location of the separated boundary layer that constitutes the secondary vortex are seen. A closer view of the leading edge region and of the primary separation point is given in Figure 7.28. Because

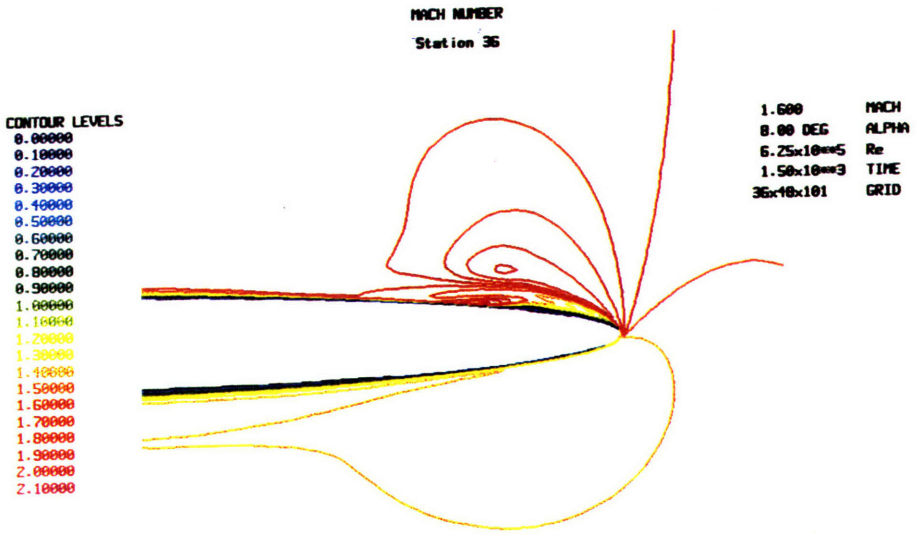

Figure 7.23: Mach number contours: Wing F, $\alpha = 8°$

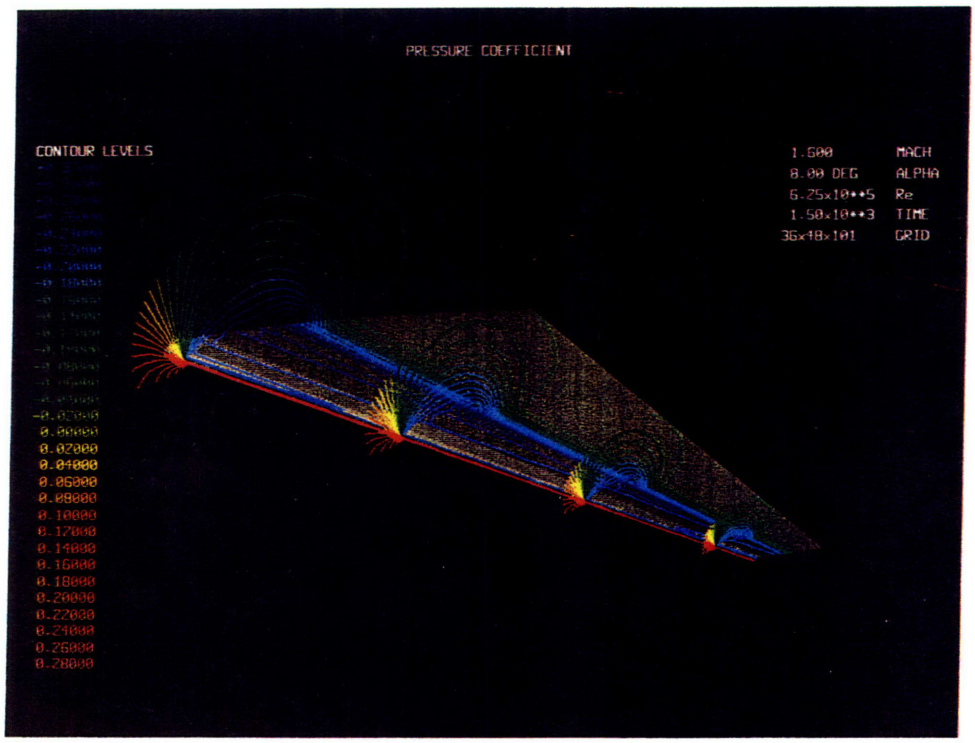

Figure 7.24: Pressure coefficient contours: Wing F, $\alpha = 8°$

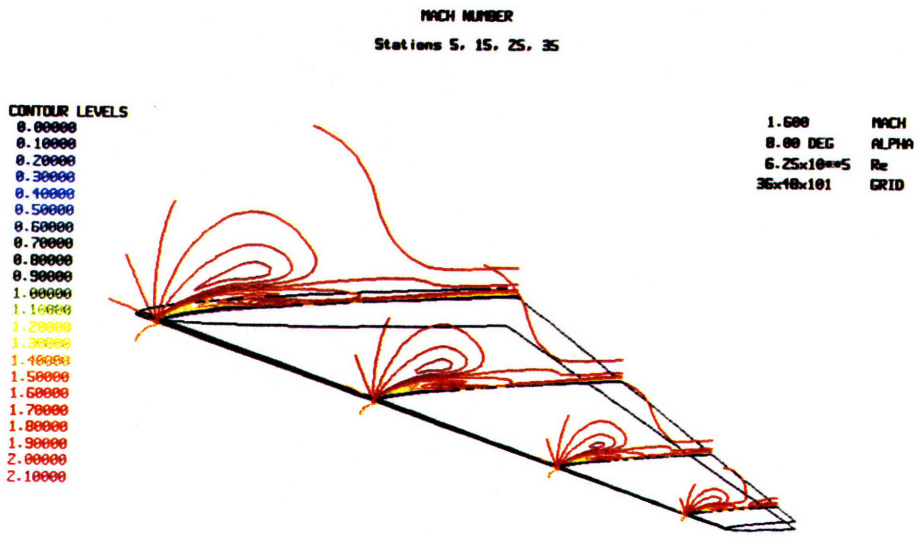

Figure 7.25: Mach number contours: Wing T, $\alpha = 8°$

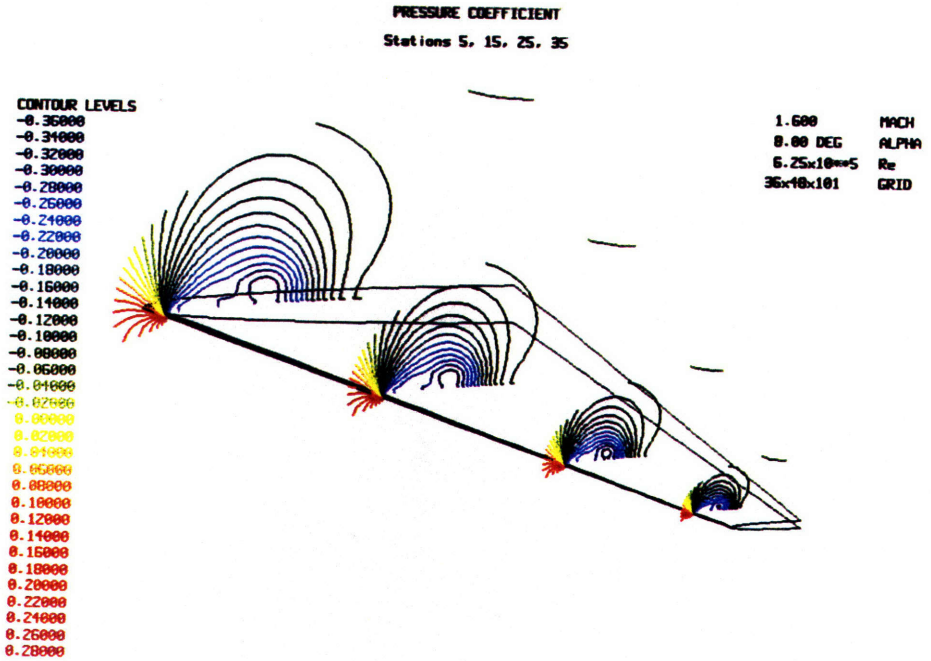

Figure 7.26: Pressure coefficient contours: Wing T, $\alpha = 8°$

of the curvature of the leading edge, location of the separation line is several grid points leeward of the leading edge.

Similar results are seen in plots of vector velocities around Wing T (Figures 7.29 & 7.30).

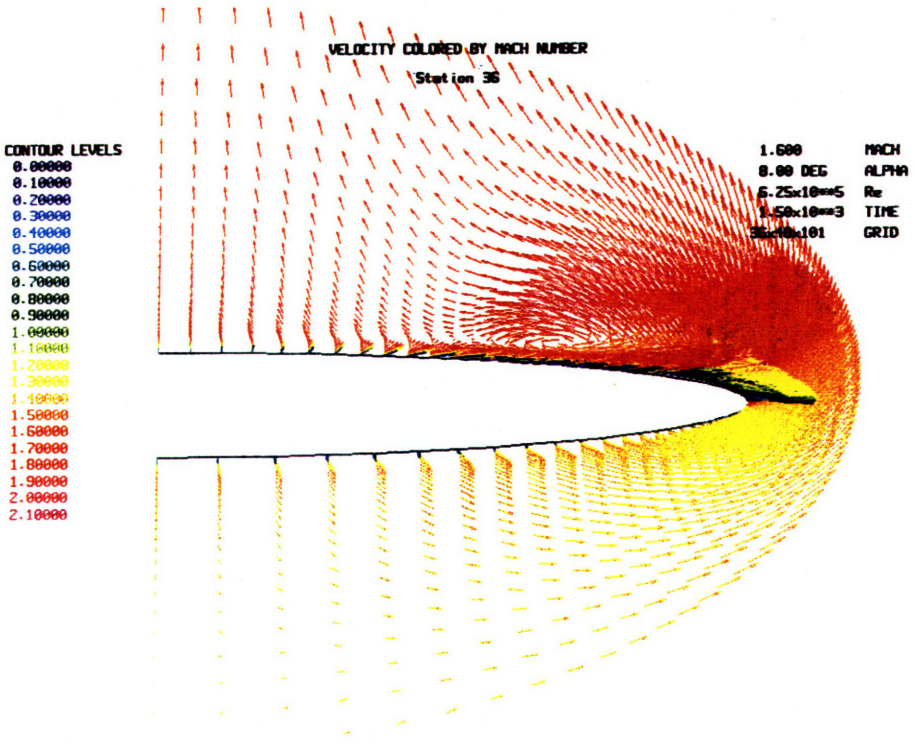

Figure 7.27: Cross stream velocity vectors at Station 36: Wing F, $\alpha = 8°$

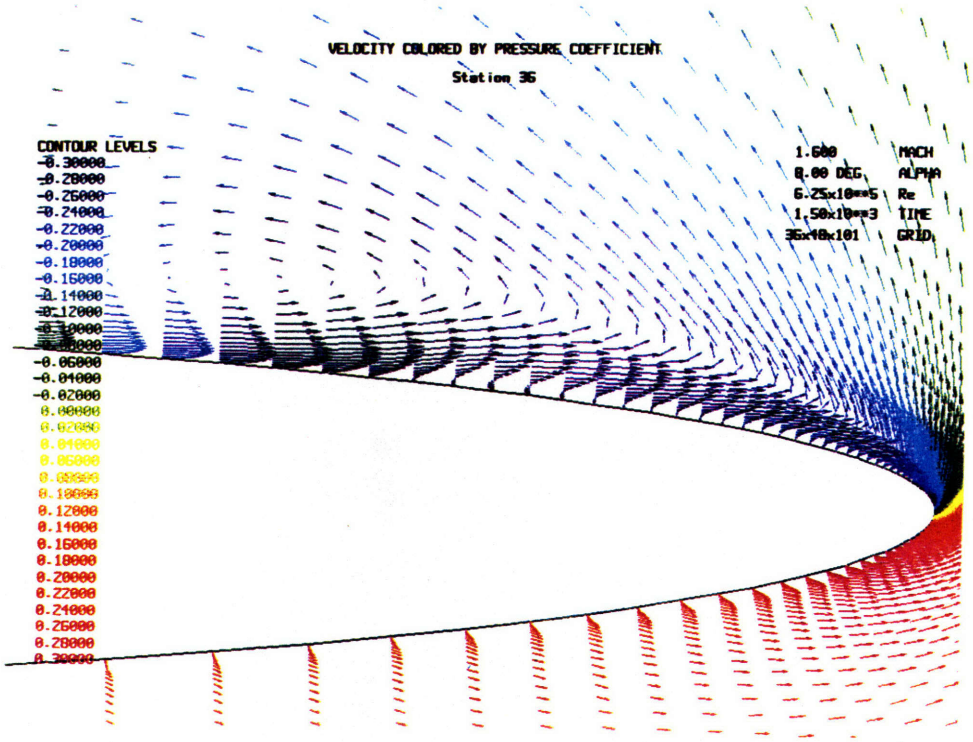

Figure 7.28: Close-up of cross stream velocity vectors: Wing F, $\alpha = 8°$

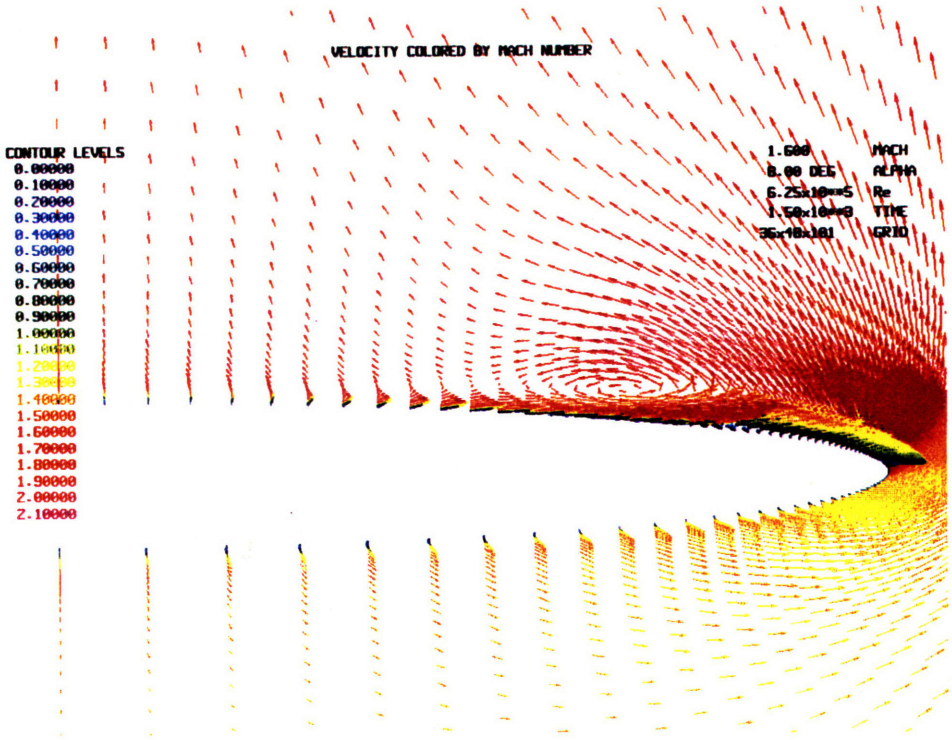

Figure 7.29: Cross stream velocity vectors: Wing T, $\alpha = 8°$

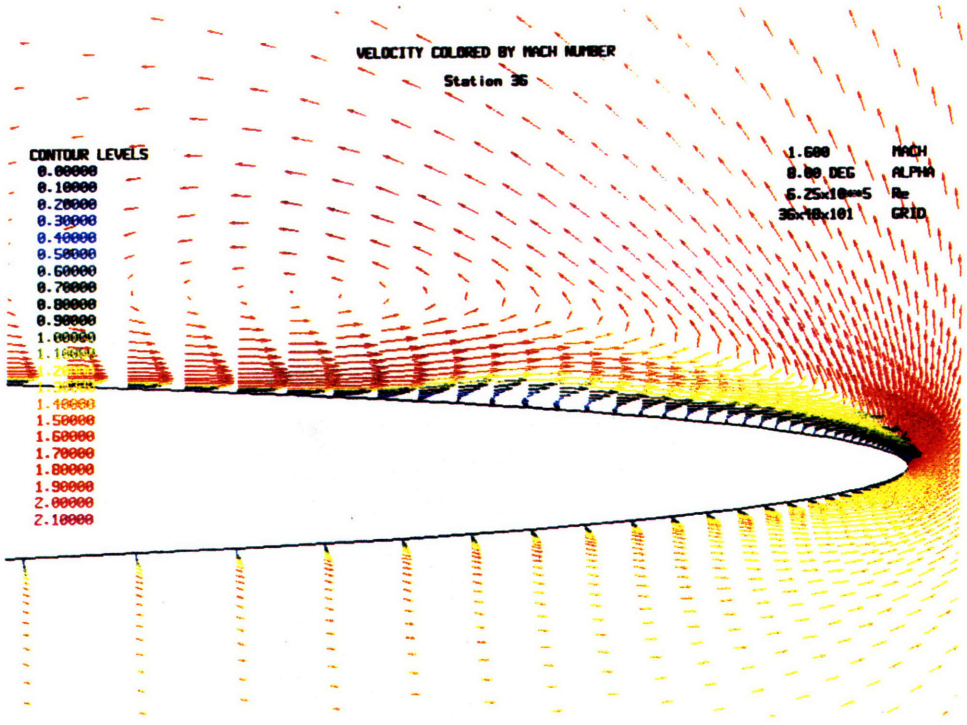

Figure 7.30: Close-up of cross stream velocity vectors: Wing T, $\alpha = 8°$

7.3 Flow Conicality

The question of whether viscous supersonic flows over conical geometries can be considered locally conical has been an issue of contention in the literature, and will be addressed briefly here. As discussed in the introduction to Chapter 6 a number of investigators have used the conical assumption to justify application of the so-called conical Navier-Stokes equations with a "local" Reynolds number. Others (e.g. [Rogers & Berry 55]) have used experimental data to argue to the contrary.

From a theoretical standpoint the conical assumption has limited validity. Derivatives in the streamwise direction are certainly non-zero in the boundary layer, as any elementary viscous fluids textbook will attest to. This includes first and, in the case of the full Navier-Stokes equations, second (viscous) derivatives. Because viscous layers tend to grow in the streamwise direction, the external inviscid flow necessarily varies with streamwise distance. Both of these factors are assumed to be negligible in CNS solutions. While numerous comparisons have been made between CNS and NS solutions, and varied conclusions have been drawn, there appears to be a lack of rigorous analysis of the effects of discarding these terms.

It is clear from the solutions above that the streamwise variation can be substantial. In three of four cases, the variation is due in part to the development of secondary separation downstream of the inflow station. This development as well as the normal growth of the boundary layer can not be predicted by a CNS computation at a single station. At best, it may be predicted by solving the CNS equations at several streamwise stations. For example, secondary and even tertiary vortical structures were found to develop with streamwise distance by [Thomas & Newsome 86], who solved the CNS equations at several Reynolds numbers.

While solving the CNS equations at several Reynolds numbers is an option, this approach has a serious theoretical deficiency: three-dimensional solutions composed of several CNS calculations at different values of Re_L do not, in the limit of $\Delta Re_L \rightarrow 0$ (that is, # of stations $\rightarrow$ 0), reduce to the corresponding Navier Stokes solution. Moreover, the cost of solution increases with every station added thus reducing the comparative efficiency of CNS solutions. For these reasons, the author feels that use of the conical Naver-Stokes equations for research purposes is not to be recommended. Instead, solution of the three-dimensional Navier-Stokes equations on a grid in which the density of streamwise stations is determined by the level of streamwise gradients expected in a flow field is the preferred approach.

Chapter 8

Flow Separation Mechanisms at Blunt Leading Edges

At sufficiently high Mach numbers and angles of attack the flow about delta wings separates and forms a system of primary, and, in some cases, secondary and tertiary vortices. For wings with *sharp* leading edges, separation lines are located at the sharp edges and fluid mechanic processes leading to separation have been fairly well characterized. Leading edge separation in flows around delta wings with *rounded* leading edges, is less well understood. It is clear, however, that the locations of separation lines on blunt delta wings are not fixed by geometry, but are determined by an interaction of geometric and fluid dynamic factors.

Since the general features of vortical flows about blunt delta wings were described in Chapters 6 and 7, this chapter examines details of the separation mechanisms. The flows about the 65° swept Wing F and Wing T at 4° and 8° angle of attack and about the 60° swept Wing F [Rogers & Berry 55] discussed in Chapters 6 and 7 are considered. Wing F and Wing T are elliptical wings with thickness-to-chord ratios of 1 : 11.55 and 1 : 20, respectively. The flow physics is examined through pressure and density profiles, Mach number contours, and projected streamline patterns near the leading edge.

Two distinct flow separation mechanisms are observed:

1. Separation near the leading edge due, at least in part, to a shock-induced pressure rise; and,

2. Separation due to shock-less flow recompression just leeward of a leading edge expansion.

The first mechanism occurs in flows about the [Rogers & Berry 55] case, Wing F and Wing T at 8°, and Wing T at 4° and is discussed in Section 8.2. The second mechanism is observed in flow about Wing F at 4° and is considered in Section 8.3. In the final section, the characterization of flows around rounded leading edges is discussed. While flows about delta wings with sharp leading edges can be usefully described by Mach number and angle of attack normal to the leading edge, this is not the case for flows about wings with blunt leading edges. In the latter cases factors such as leading edge curvature and flow Reynolds number must be considered.

8.1 Flow Structure Identification

In most flow cases, examination of Mach number and pressure profiles suffices to identify shock structures and shear layers. Because of the physical proximity of shock and boundary layer in the present case, the conventional shock indicators – steep negative gradients of Mach number linked with positive gradients of pressure – are not sufficient. This is because any coherent shear layer displays a strong Mach number gradient across it; therefore, a sharp drop in the Mach number even when accompanied with a pressure increase (which may be due to shock-less recompression) does not necessarily imply existance of a shock[1]. Rather, the existance of a sharp density rise

[1]Similarly, changes in entropy or stagnation pressure are ambiguous since these quantities increase and decrease, respectively, through *both* shear layers and shock waves.

in addition to appropriate Mach number and pressure gradients is required to identify shock locations[2], and changes must correspond to the Rankine-Hugeniot shock jump relations. Identification of a shock wave may also complicated by its small size.

To examine the flow structures flow variables are plotted against a conveniently defined arclength. Each profile in the figures below plots a scalar quantity at a given streamwise station along a particular (j) family of (k) points, where (j) is the coordinate direction away from the body, and (k) runs cross stream from the windward to the leeward symmetry planes.

The initial value of the arclength is set to $\simeq 1$ and subsequent values are defined by subtracting the arclength between two adjacent (k) points from previous values: $s_{l+1} = s_l - \Delta s$. In all cases the initial cross stream station $l = 0$ is fixed at $k = 40$ (leading edge) and $l_{max} = k = 75$,[3] although, ideally, a station normal to the leading edge would be considered. Thus, quantities plotted are in the vicinity of the leading edge, corresponding to $0.9 \leq t/c \leq 1.0$. Values plotted were taken directly from the flow solution without any post-processing.

[2]The density in the shear layer usually decreases as $M \to 0$ since temperature increases and pressure is relatively constant. Thus a sharp increase in ρ is an unambiguous indication of presence of a shock layer if it occurs together with gradient in the Mach number and pressure.

[3]The initial value s_o is defined as

$$s_o = \frac{y_{i,j,40}}{y_{i,1,40}} + \frac{\Delta s_0}{2} , \tag{8.1}$$

and Δs is calculated along the $x = const.$ station as

$$\Delta s = \sqrt{(y_{i,j,k+1} - y_{i,j,k})^2 + (z_{i,j,k+1} - z_{i,j,k})^2} , \tag{8.2}$$

8.2 "Shock-Induced" Separation

Cross stream pressure coefficient profiles of flow past Wings T and F at 8° angle of attack, Wing T at 4°, and the Rogers and Berry wing at 14° (Figures 7.21, 7.16, 6.12) all show sharp negative pressure coefficient spikes that appear to indicate a shock wave lying in the vicinity of the leading edge. Examination of the results reveals a weak shock that impinges on the boundary layer and extends a small distance into the outer flow. The sharp pressure gradient due to the shock wave apparently induces boundary layer separation near the leading edge.

This section identifies the shock structure and describes interaction of the shock wave and shear layers and the separation process. The flow phenomena in the cases are similar, but, are especially well defined in the 14° Rogers and Berry case. Thus, this case is used to describe the flow physics.

8.2.1 Rogers & Berry Case

The [Rogers & Berry 55] wing case will be examined first because, due to its higher angle of attack, it displays a number of flow features relatively clearly. The flow about the Rogers wing has a characteristic vortical pattern: a primary vortex due to boundary layer separation near the leading edge and a developing secondary vortex underneath and somewhat outboards the core of the primary vortex. As in all cases considered here, the primary vortex shows a very flat elongated structure on the leeward side of the wing.

The flow at the leading edge is shown in the Mach and stagnation pressure contours of Figures 8.1 and 8.2. Separation of the shear layer is marked by divergence of contours

from the body on the leeward side of the wing just inboard of the leading edge. It is seen that the flow immediately inboard and outboard of the stem of this shear layer is supersonic, reaching values of approximately 2. The fluid originating from the inner region of the boundary layer has the highest stagnation pressure loss and is entrained into the core of the vortical layer. Although stagnation pressure changes are usually a clue to the location of shock layers this clearly is not the case here[4].

To characterize the Mach number, pressure coefficient, and density variations in this region, plots of these quantities were made along a path at constant (j) values, tangent to the leading edge (Figure 8.3). Figures 8.4, 8.5, and 8.6 contain the sum of information necessary to identify the shock structure. In this case a shock wave of very limited extent is present near the leading edge. Although this shock is weak it apparently induces separation by turning the leeward flowing boundary layer fluid at the leading edge away from the body.

A schematic of the flow structure near the leading edge is given in Figure 8.7. A shock wave inclined at approximately 30° to the vertical lies parallel to the leading edge. It extends approximately from gridlines $j = 8$ to $j = 20$, and, at Station 36, intersects the computational grid at $(j,k) \simeq (10,44),(15,45)$ and $(20,47)$. This shock layer is indicated by sharp increases in the density and decreases in Mach number and negative pressure coefficient at those (j,k) locations.

Although the *total* Mach number deficit just leeward of this region is substantial, density and pressure ratios across the shock at $j = 10$ of 8% and 12%, respectively, indicate a relatively weak shock with Mach number normal to the shock of approximately 1.05. This is approximately consistent with the Rankine-Hugeniot relations. The bulk

[4]As shown by [Powell 87], the vortex itself also gives rise to a stagnation pressure loss.

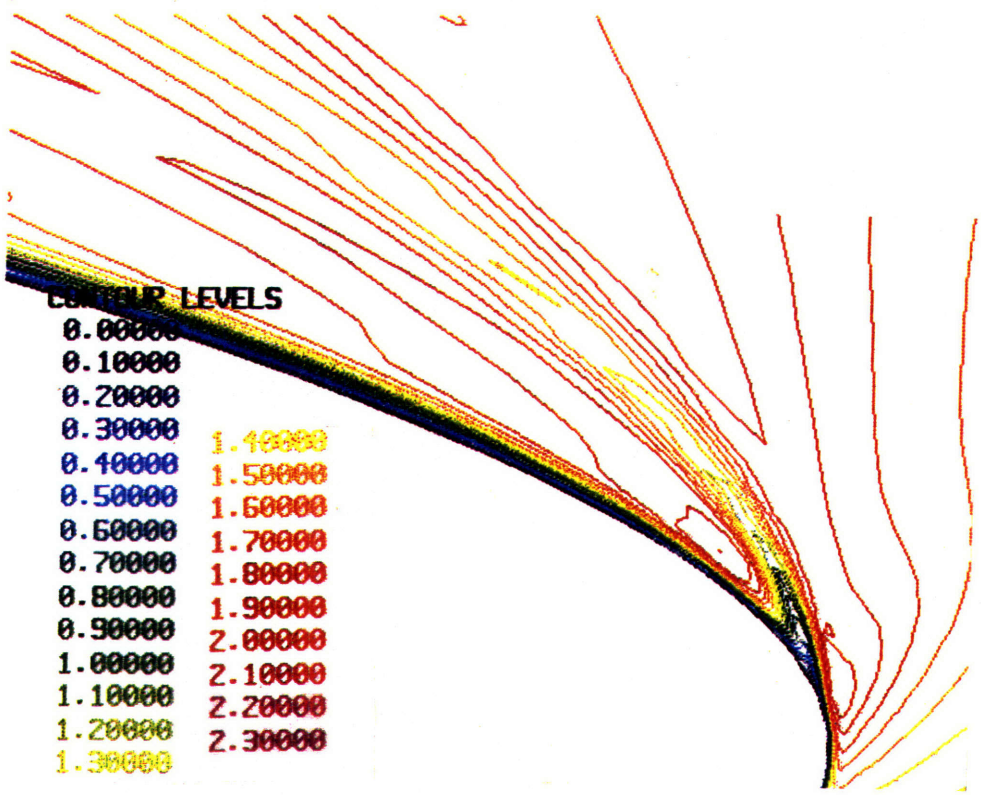

Figure 8.1: Mach number contours: Rogers Wing, Station 36

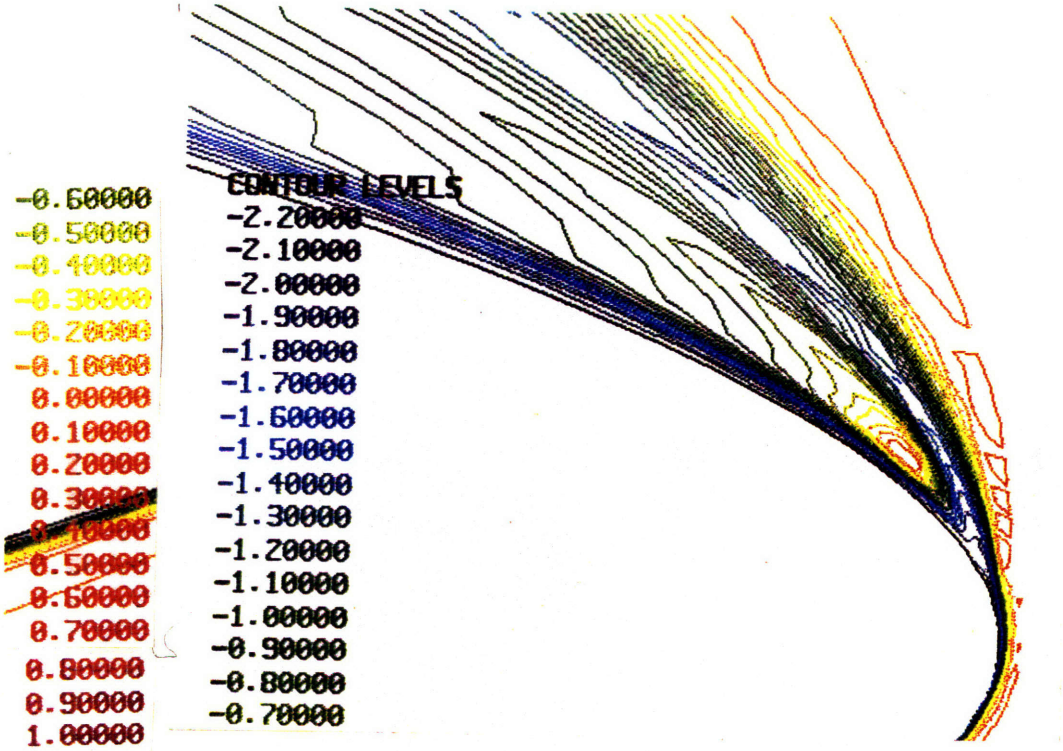

Figure 8.2: Stagnation pressure coefficient contours: Rogers Wing, Station 36

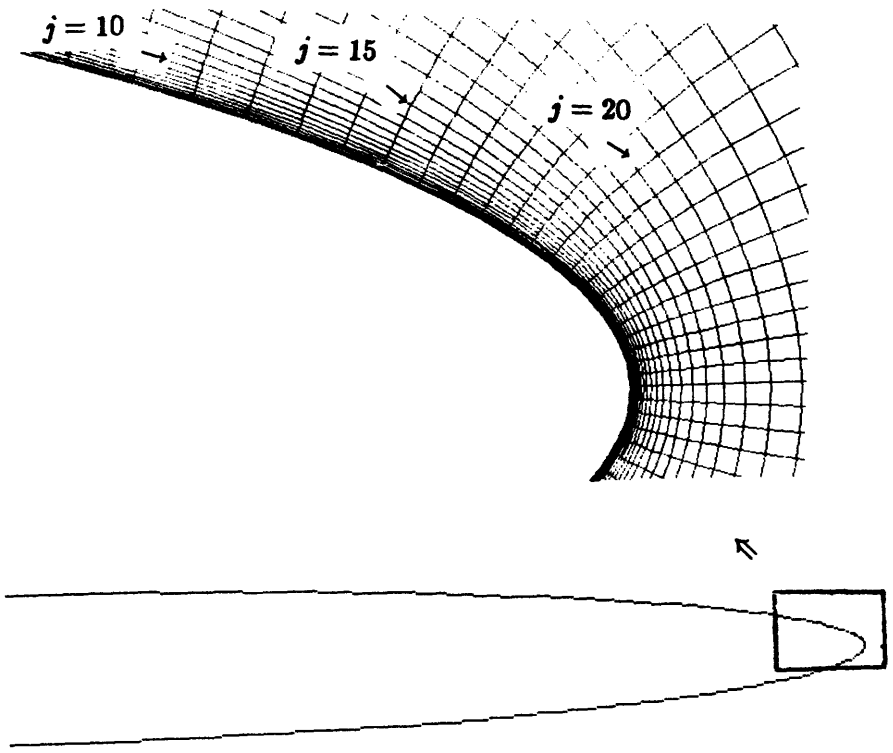

Figure 8.3: Location of profiles: Rogers Wing, Station 36

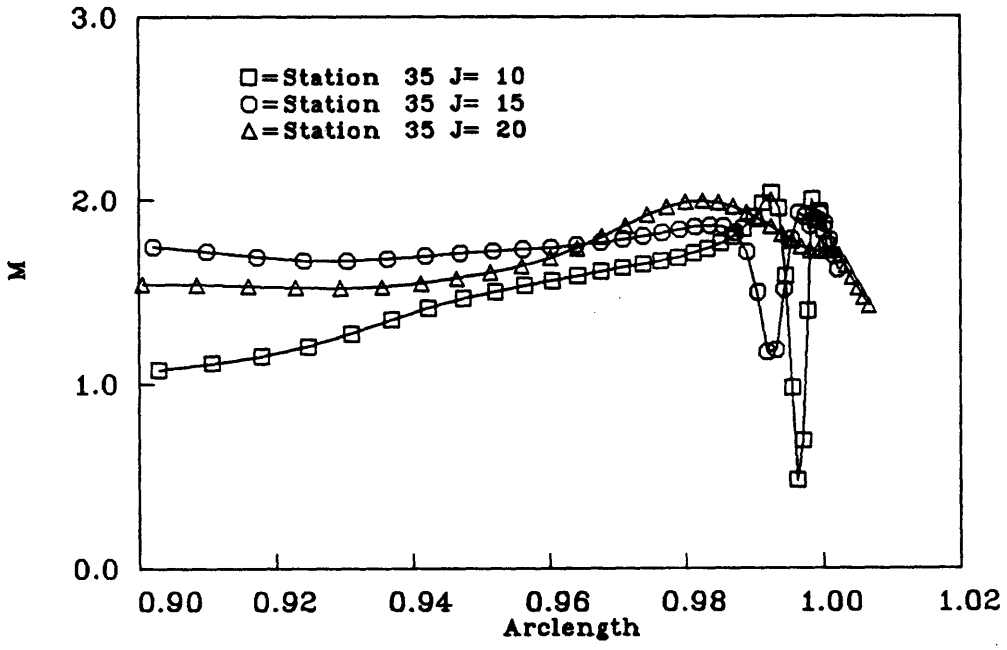

Figure 8.4: Mach number profiles: Rogers Wing, Station 36

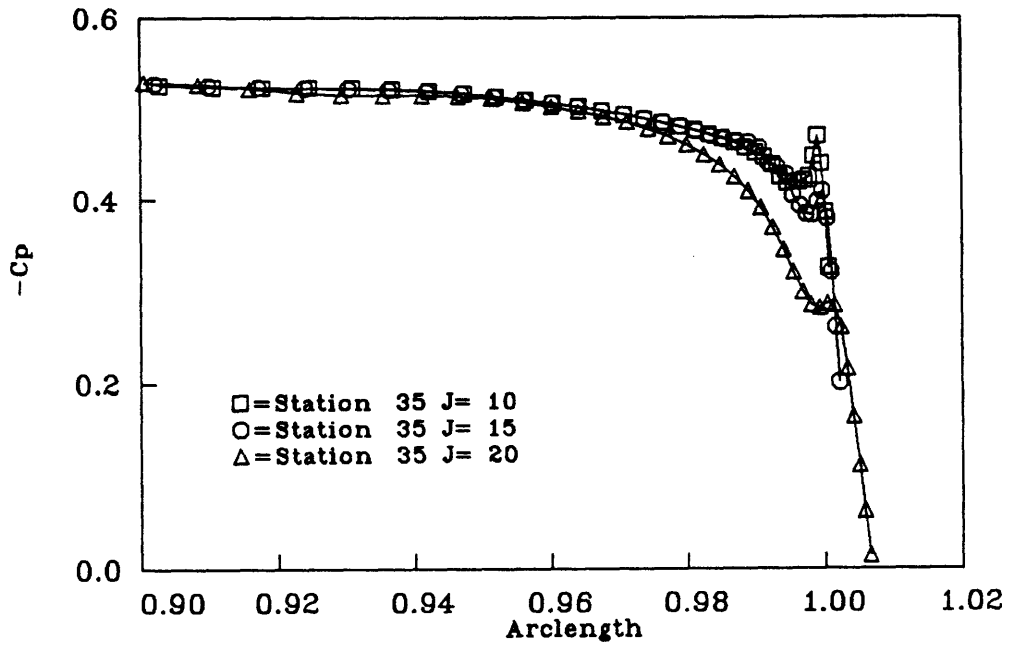

Figure 8.5: Pressure coefficient profiles: Rogers Wing, Station 36

of the Mach number deficit can be attributed to the velocity defect in the separated shear layer leeward of the shock. However, the inertia of the boundary layer fluid in the cross stream direction is sufficiently small so that the adverse pressure gradient appears to be sufficient to induce separation of the flow.

Existance of a leading edge shock is also indicated by curvature of projections of the three-dimensional streamlines on the $y-z$ plane in this area (Figure 8.8). This figure contains grid lines at $j = 1, 11, 21$ and all (k) lines at the leading edge. Particles were released at $j = 2$ and all k from $k = 40$ (leading edge) to $k = 60$. The outbound flow on the leeward side acts as a displacement surface which deflects streamlines originating near the leading edge from the body. The concavity of the streamlines in the flow direction indicates that flow compression takes place. Because the component of the flow in the $y-z$ plane is supersonic a leading edge shock develops. By Crocco's theorem, the

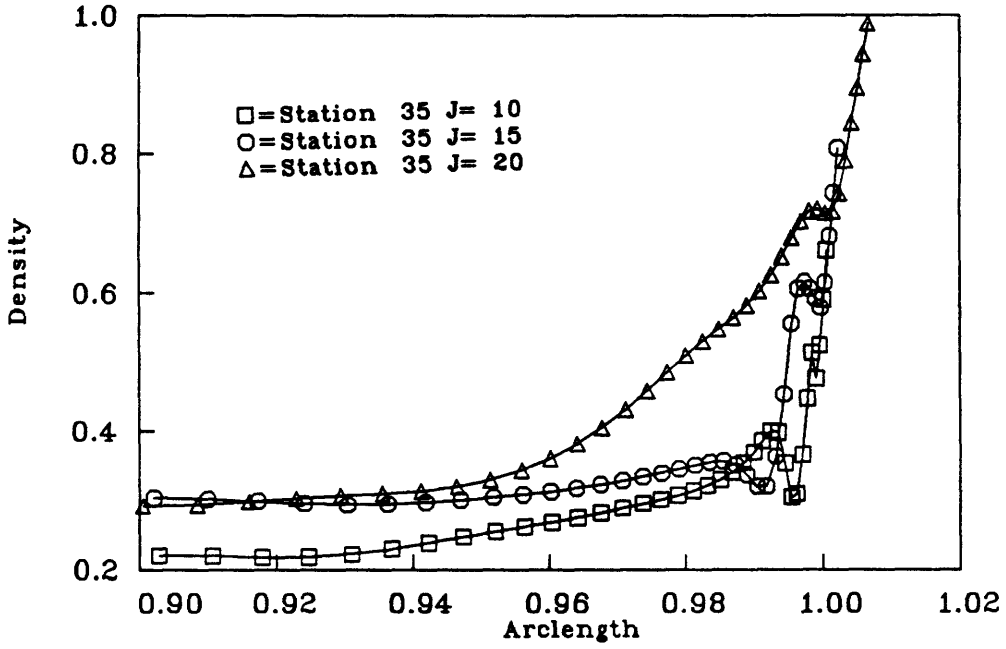

Figure 8.6: Density profiles: Rogers Wing, Station 36

varying strength of the shock results in a variation of vorticity normal to the streamlines, which may aid the separation process. Curvature of streamlines ahead of the shock may be due to the upstream influence of the pressure rise that is transmitted in the subsonic portion of the boundary layer.

Immediately after the leading edge expansion and shock wave, a second region of rapid variation of density, Mach number, and pressure coefficient can be seen at $j = 10$ and $j = 15$. In this region, the flow is headed outboards and is about to be entrained with the separated windward side boundary layer. Although a compression is evident, especially at $j = 10$, the relatively slow variation of the density and pressure coefficient indicates that this occurs without the presence of a shock.

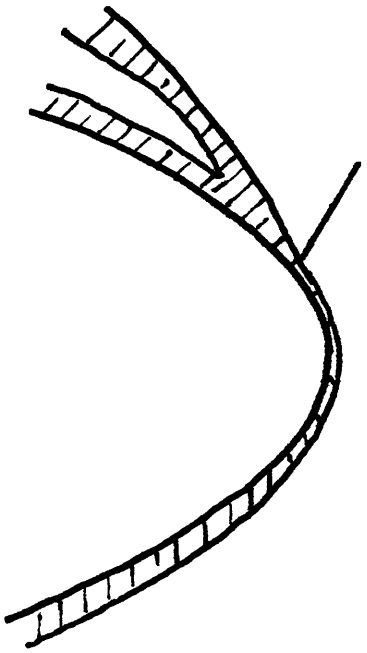

Figure 8.7: Schematic of Proposed Leading Edge Flow Structure

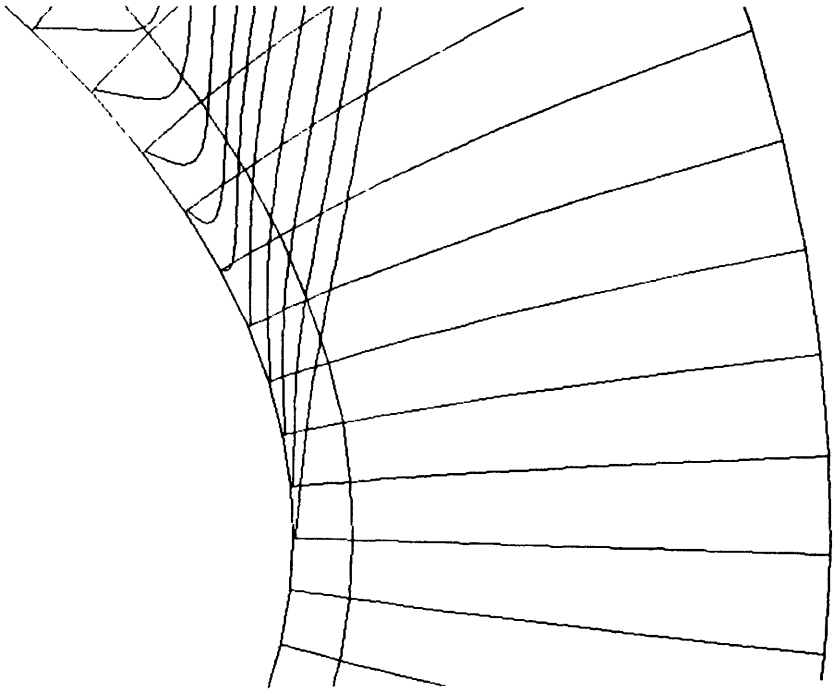

Figure 8.8: Streamlines at the leading edge: Rogers Wing, Station 36

8.2.2 Wing F & Wing T – 8 Degrees

Mach contours near the leading edge of Wing F and Wing T at 8° angle of attack are given in Figures 8.9 and 8.10. Even though the curvature of the leading edges of the two wings are quite different the flow structures near the leading edge appear very similar. The shock structures are very similar to the structure described above.

Profiles of the pressure coefficients given in Figure 7.21 show sharp negative pressure coefficient spikes at the leading edges. Figures 8.11 and 8.12 show density variation as a function of arclength in the vicinity of the leading edges. The shock wave at the leading edge of Wing F intersects Station 35 at approximately $(j,k) = (8,45),(12,45)$ and $(16,46)$. The shock at the leading edge of Wing T is of somewhat more limited extent than that in the Wing F solution, where no evidence of a shock compression is seen at $j = 16$. The shock wave intersects Station 35 at approximately $(j,k) = (8,42)$ and $(12,48)$.

The characteristic bending of streamlines at the leading edge seen in Figure 8.8 can be also observed in Figures 8.13 and 8.14. An effect of the higher leading edge curvature is a movement of the separation line towards the leading edge. While the separation point at Station 36 of Wing F is at $k = 47$ it is at approximately $k = 43$ on Wing T. However, the fundamental flow structure is not altered.

8.2.3 Wing T – 4 Degrees

Mach contours for the leading edge flow for Wing T and Wing F at 4° angle of attack are given in Figures 8.16 and 8.17. (The latter will be discussed in the next section.) Because of the low angle of attack the maximum Mach number in this case

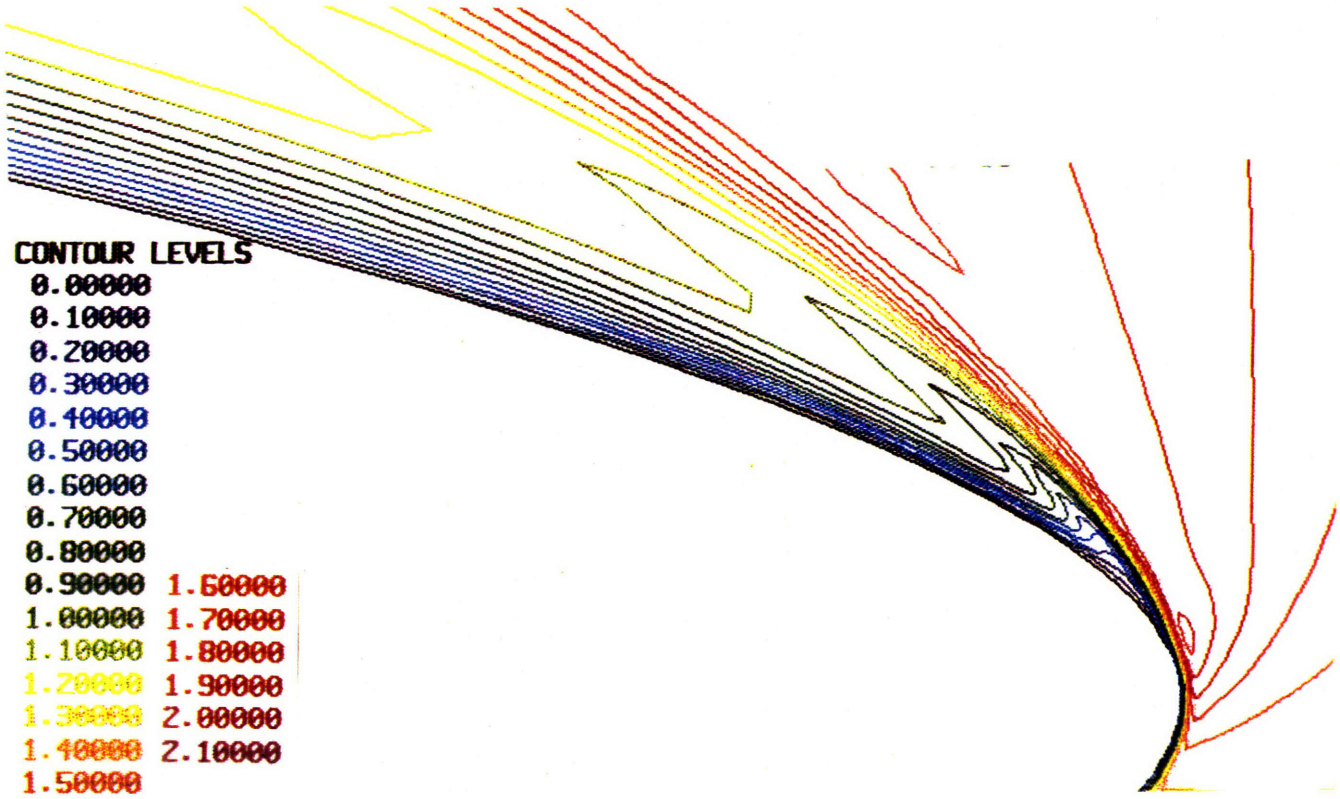

Figure 8.9: Mach number contours: Wing F, 8°_i Station 36

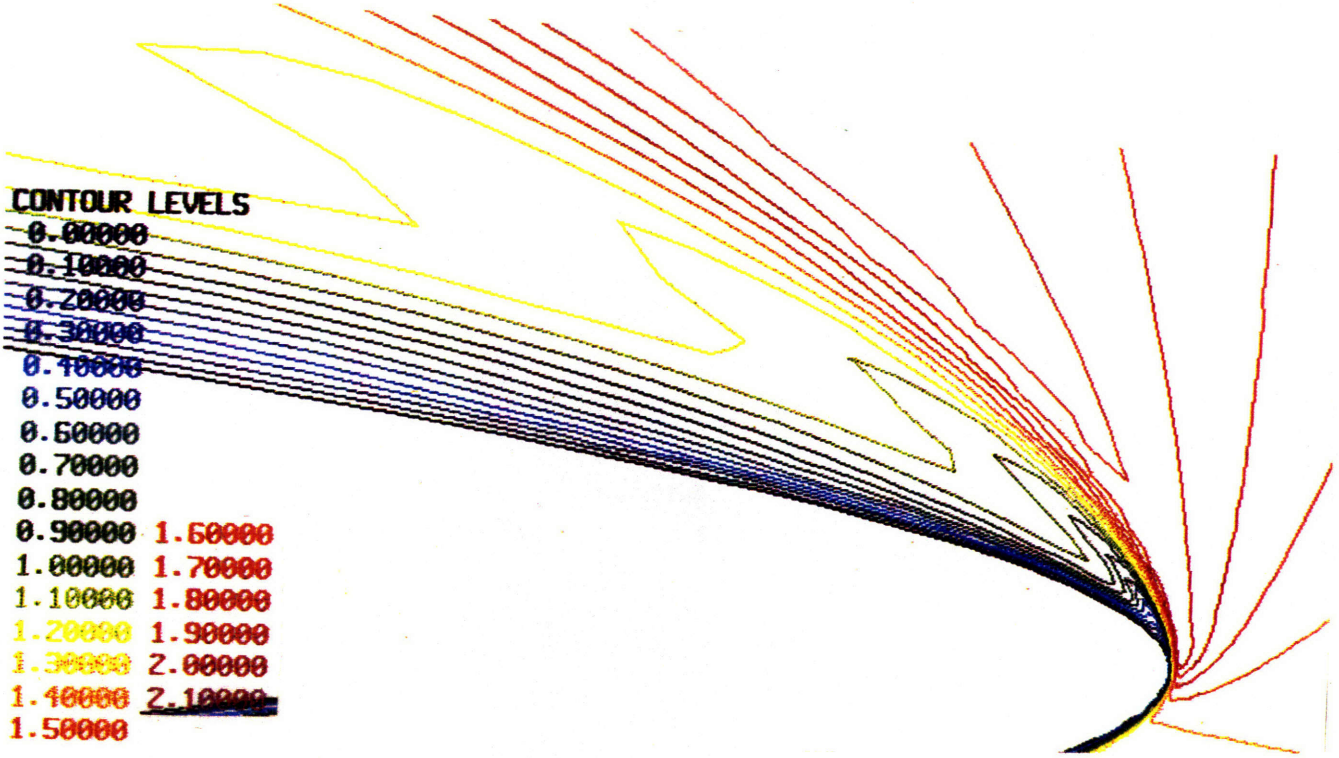

Figure 8.10: Mach number contours: Wing T, 8°_i Station 36

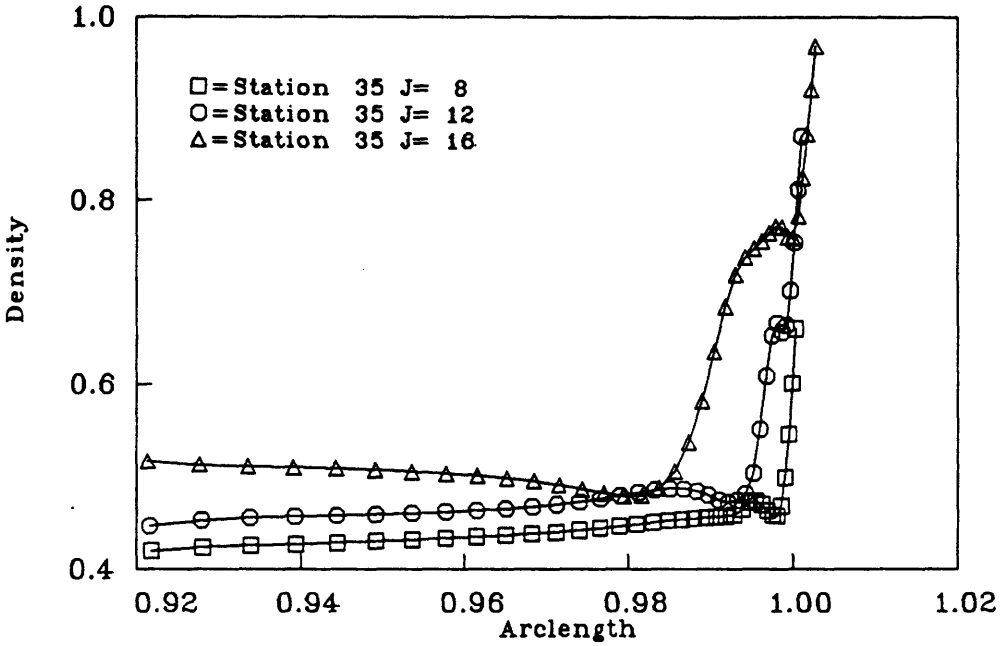

Figure 8.11: Density profiles: Wing F, $\alpha = 8°$, Station 36

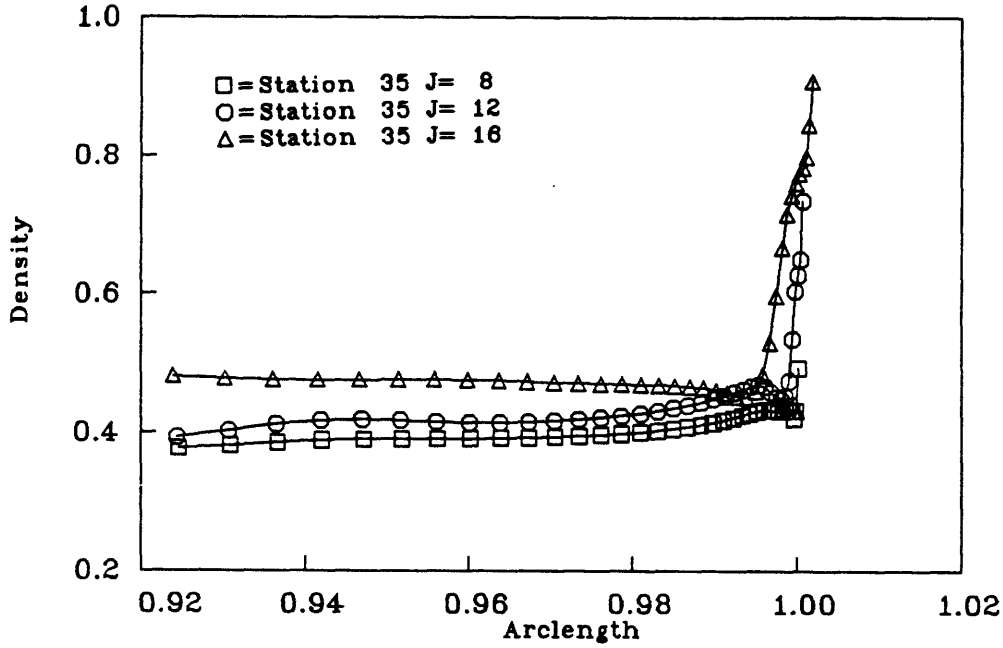

Figure 8.12: Density profiles: Wing T, $\alpha = 8°$, Station 36

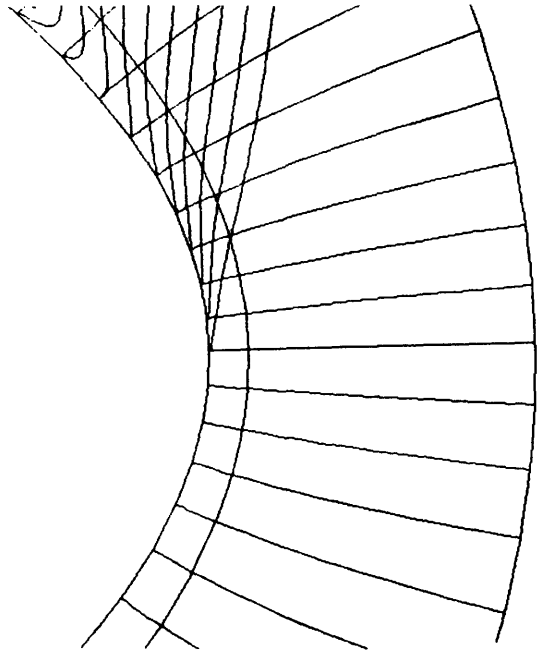

Figure 8.13: Streamlines at the leading edge: Wing F, $\alpha = 8°$, Station 36

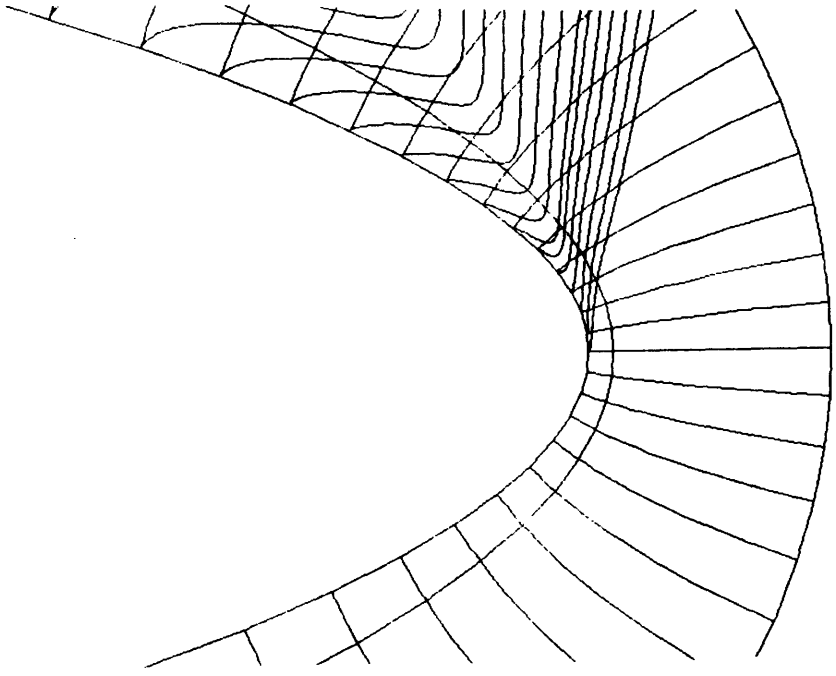

Figure 8.14: Streamlines at the leading edge: Wing T, $\alpha = 8°$, Station 36

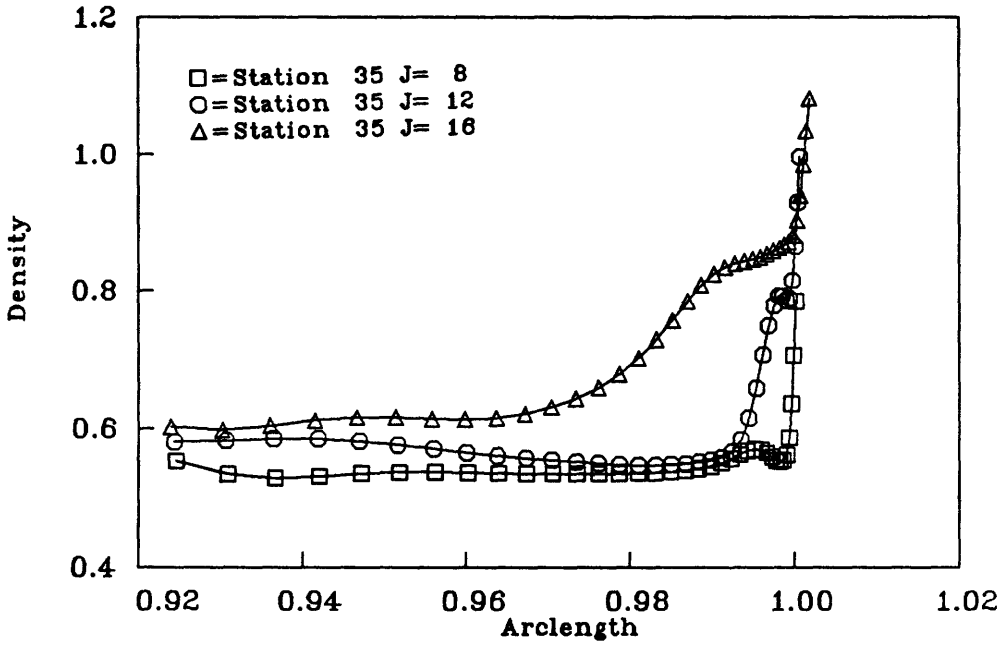

Figure 8.15: Density profiles: Wing T, $\alpha = 4°$, Station 36

is substantially smaller than those found in the above cases. Thus, the existance of a leading edge shock and shock-induced separation becomes less likely. Nevertheless, Wing T displays a flow structure, complete with shock, similar to the previously discussed cases.

Profiles of the density versus arclength in the leading edge area of Wing T are given in Figure 8.15. A weak shock is located at $(j,k) = (8,45)$ and $(12,45)$. The streamlines pattern near the leading edge of the Wing T solution (Figure 8.18) is similar to patterns observed above. The primary separation line at Station 36 is located at $k = 47$.

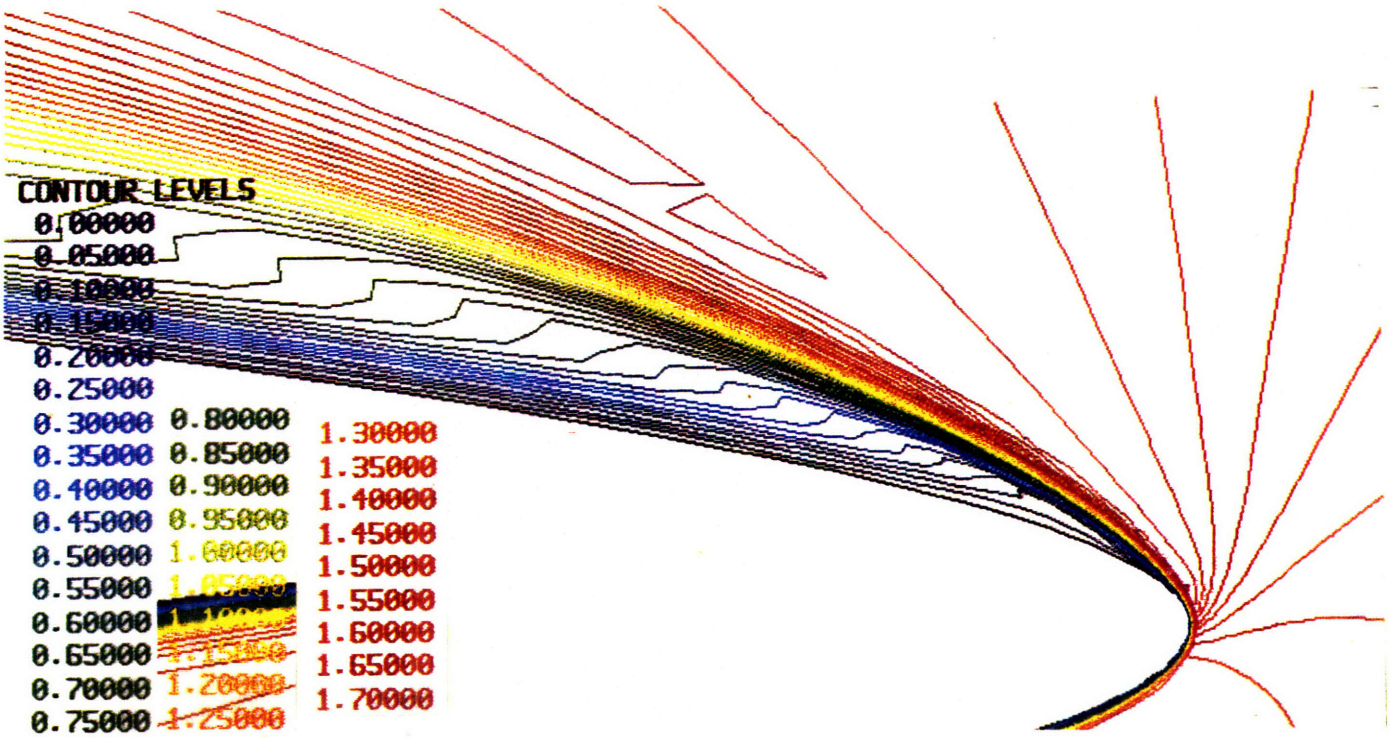

Figure 8.16: Mach number contours: Wing T, $\alpha = 4°$, Station 36

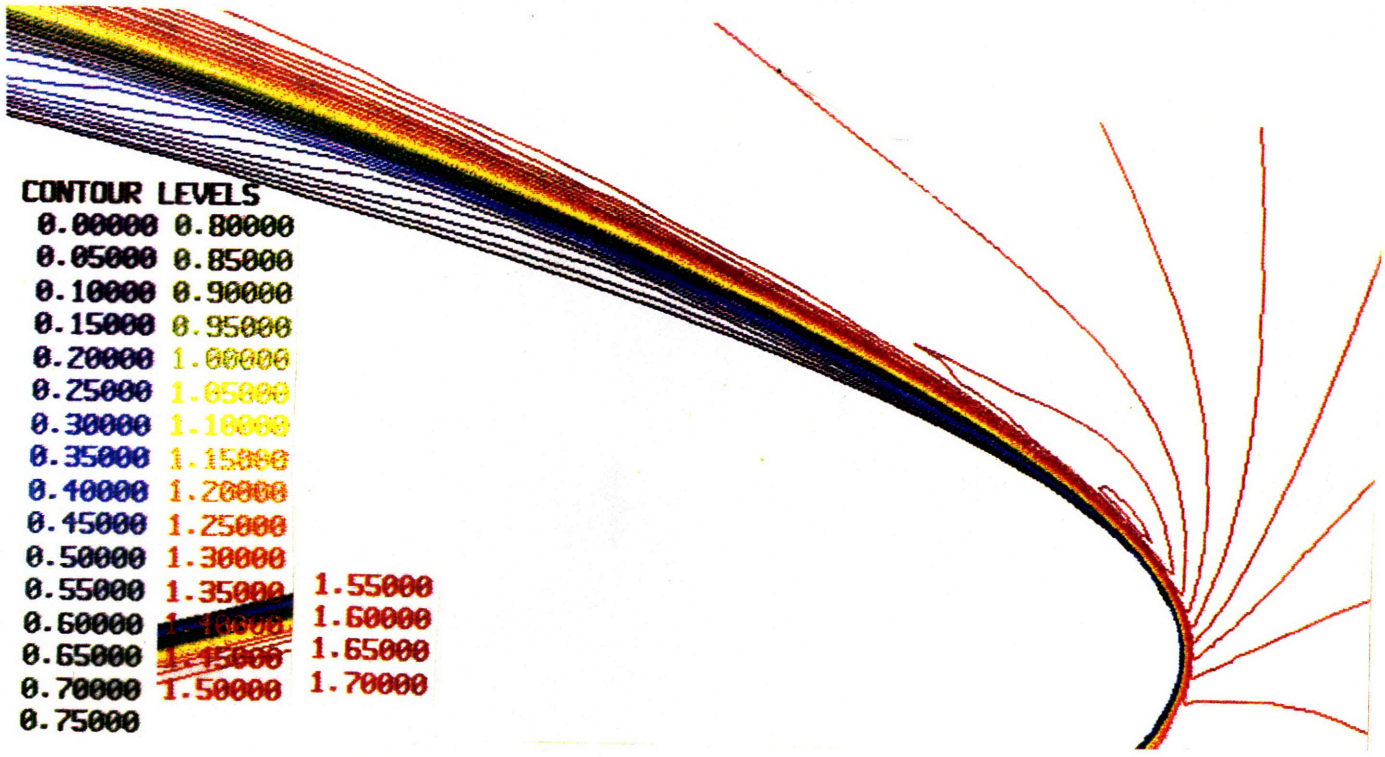

Figure 8.17: Mach number contours: Wing F, $\alpha = 4°$, Station 36

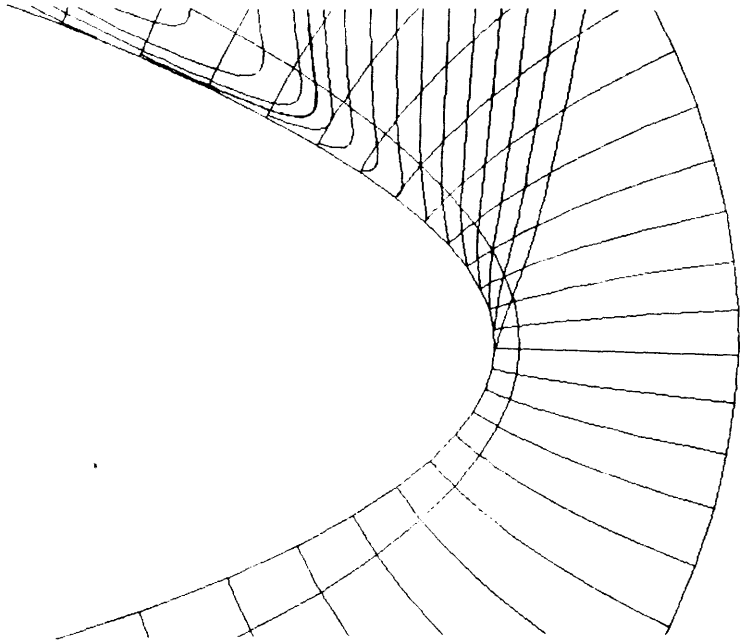

Figure 8.18: Streamlines at the leading edge: Wing T, $\alpha = 4°$, Station 36

8.3 Separation Due to Shock-Less Recompression

In viscous calculations a shock wave or Kutta condition are not necessary to induce separation, since vorticity is introduced by the no-slip condition at body surfaces. The adverse pressure gradient that is required by classical boundary layer theory for separation is, in these cases, supplied by flow recompression inboards of the rapid leading edge expansion. This separation mechanism is found in the Wing F, $\alpha = 4°$ solution.

8.3.1 Wing F – 4 Degrees

Pressure profiles of the 4° Wing F solution indicate a strong recompression just inboard of the leading edge (Figure 8.19). This recompression is not, however, a shock wave. Figure 8.20 "unwraps" the wing to show Mach number profiles at Station 35 and $j = 10, 15, 20, 25$ versus y that, for $j \geq 15$, are entirely supersonic in the leading edge area, and that have no rapid gradients that might indicate a cross flow shock. The flow at $j = 10$ is in the boundary layer, which thickens for a short distance just leeward of the leading edge. The Mach number in the inviscid flow increases monotonically just leeward of the leading edge, indicative of a smooth shock-less expansion of the flow around the blunt leading edge. Profiles of the density (Figure 8.21) at $j = 8, 12, 16$ also show no rapid shock-induced increases of that quantity. Separation is due to the inability of the boundary layer at the leading edge to negotiate the adverse pressure gradient which is due to flow recompression following the leading edge expansion.

The streamline pattern around the leading edge of Wing F (Figure 8.22 is significantly different from any of the previously examined patterns. Because the adverse pressure gradient at the leading edge of this case is relatively benign (Figure 7.16) the flow remains attached remains attached longer – at Station 36 up to $k = 55$ or 99.2%

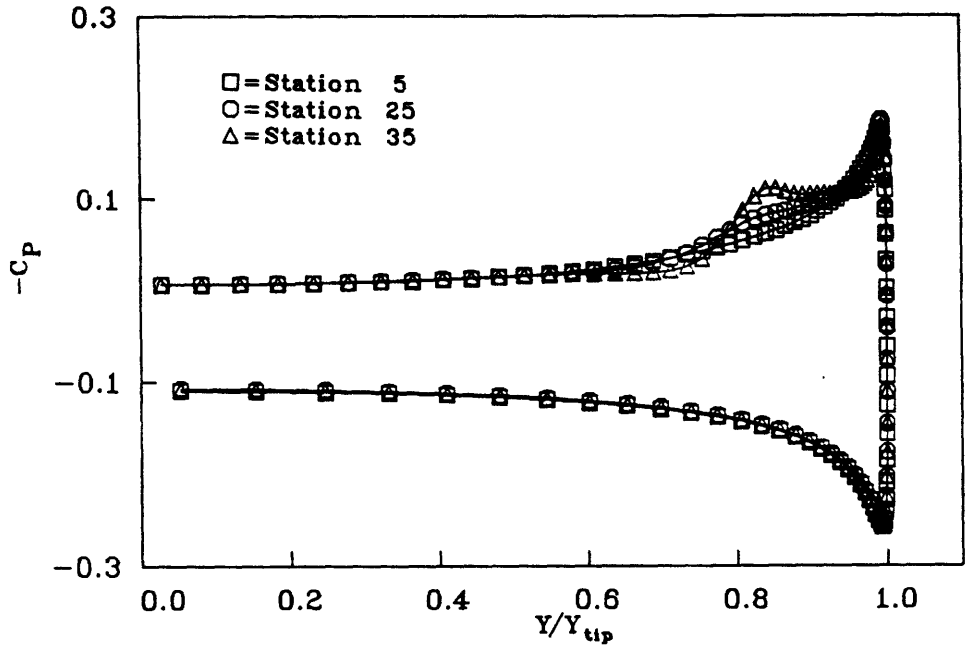

Figure 8.19: Pressure coefficient profiles at Station 35: Wing F, $\alpha = 4°$

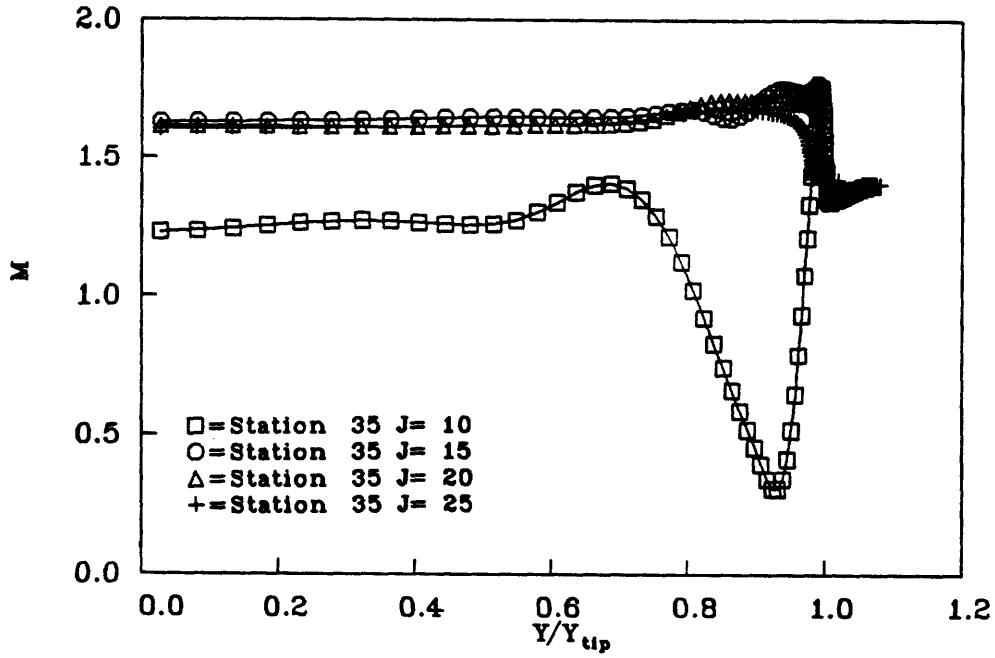

Figure 8.20: Mach number profiles at Station 35 (j=10,15,20,25): Wing F, $\alpha = 4°$

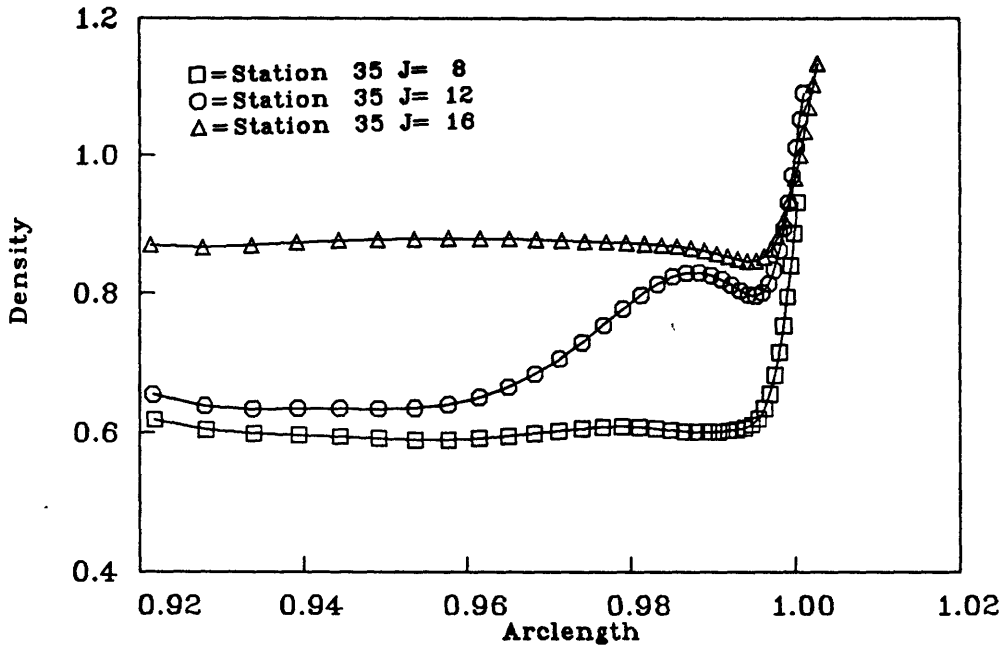

Figure 8.21: Density profiles: Wing F, $\alpha = 4°$, Station 36

span. An outward displacement of the projected streamlines is seen, which is similar to that seen in the previous cases. Apparently, the lower Mach numbers in this area prevent the development of a shock.

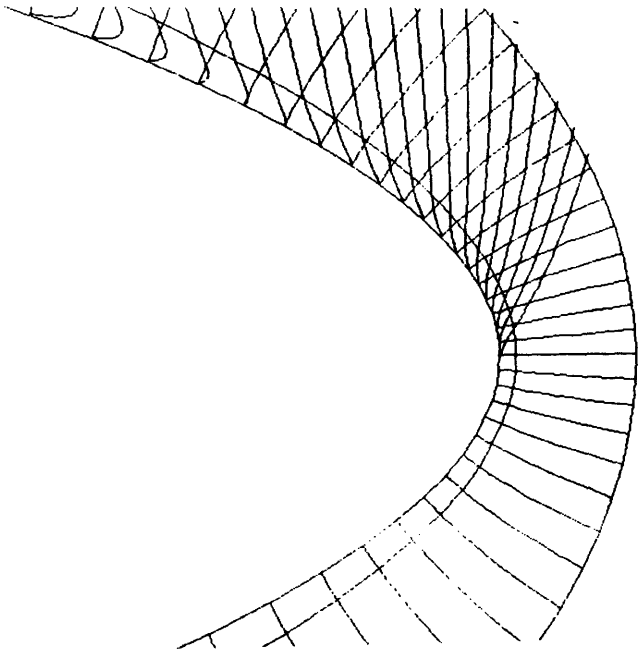

Figure 8.22: Streamlines at the leading edge: Wing F,$\alpha = 4°$, Station 36

8.4 Discussion

The preceding sections have examined the flow separation processes at the leading edges of several test cases. In each of the cases examined, the flow separated near the leading edge of the elliptical delta wing. Location of the separation line and flow phenomena at the leading edge varied from case to case, with a weak shock wave appearing near the leading edge in four of five cases. These results indicate that it is not possible to characterize flows around delta wings with blunt leading edges on the basis of only Mach number and angle of attack normal to the leading edge (eg. see discussion for delta wings with sharp leading edges in Chapter 6). Additional factors such as leading edge curvature and flow Reynolds numbers must be considered. Further discussion of the leading edge flow phenomena is given below.

Table 8.1 presents a summary of leading edge flow information at the final station (Station 35) of the test cases. M_{shock} and k_{sep} are the approximate Mach number normal to the shock wave and the circumferential station at which separation takes place, respectively. The final two columns give the pressure ratios corresponding to M_{shock} and the pressure ratios actually observed. Values of M_{shock} were calculated from the density jump data in Figures 8.6, 8.12, 8.11, 8.15, and 8.21. The separation point was identified by inspection of particle paths of the $v - w$ velocity (Figures 8.8, 8.14, 8.13, 8.18, 8.22).

Table 8.1 indicates that wing geometry and flow conditions influence flow separation phenomena for delta wings with blunt leading edges. Four of the five cases in Table 8.1, the Rogers wing at 14°, Wing T & F at 8° and Wing T at 4° angle of attack, appear to involve a shock wave as part of the separation process. Separation in the Wing F flow at 4° angle of attack is due to a shock-less recompression near the leading edge.

Table 8.1: Summary of leading edge flow data

Test Case	α	k_{sep}	M_{shock}	$(P_2/P_1)_{shock}$	$(P_2/P_1)_{actual}$
Rogers wing	14°	45	1.05	1.12	1.16
Wing T	8°	43	1.03	1.07	1.25
Wing F	8°	47	1.02	1.05	1.21
Wing T	4°	47	1.02	1.05	1.20
Wing F	4°	55	N/A	N/A	1.14

Separation occurs relatively close to the leading edge for Wing T compared to Wing F, the latter having a more blunt leading edge. This behaviour is expected since, in the limit of infinite leading edge curvature, separation is fixed at the leading edge. An increase in the angle of attack of the wing results in an increase of the local turning angles near the leading edges and, thus, in a movement of the separation lines move towards the leading edges ($k = 40$). This is observed in both the Wing T and Wing F cases, where k_{sep} moves from 47 to 43 and 55 to 47, respectively, as the angle of attack increases.

The extent to which the shock waves "induce" flow separation is unclear. In classical boundary layer theory, an adverse pressure gradient is a necessary though not sufficient criterion for separation. The magnitude of the adverse pressure gradient required in this theory depends on flow conditions and body geometry, and usually can not be determined *apriori*. Certainly, the proximity of the shock wave with the separation point indicates that the shock is at least in part the cause of separation. Because the shock waves in these cases are quite weak, it seems clear, however, that the shock wave induced pressure rises are at the lower limits of what might be required. Moreover, the

actual pressure rise at the leading edges of three of the four cases that contain shock waves is significantly higher than that calculated from the Rankine-Hugeniot relations. The pressure increase not due to the shock might be ascribed to flow recompression just leeward of the leading edge that results when the outward flowing fluid under the primary vortex slows down as it is turned and entrained by the separated boundary layer at the leading edge.

Chapter 9

Concluding Remarks

The goals of the research effort described in this dissertation were to develop an efficient numerical algorithm method for solving the Navier-Stokes equations, to use the new method to study vortical flow about delta wings with rounded leading edges, and, in particular, to examine flow separation processes near the leading edges of these wings. Thus, a novel *semi-implicit* temporal integration algorithm was developed and applied to the Navier-Stokes equations. Efficiency of the algorithm was established by comparison with conventional explicit and implicit methods. Subsequently, the scheme was used to calculate the laminar vortical flow about several delta wings with rounded leading edges. Flow separation occured near the leading edges of the five cases examined. In four of these cases, a shock wave contributed to the pressure rise that induced flow separation. In the fifth case, flow separation was due to shock-less recompression leeward of the leading edge expansion. The following three sections describe these contributions and offer recommendations for future research.

9.1 Development of a Semi-Implicit Navier-Stokes Solver

The goal of developing an efficient algorithm for the Navier-Stokes equations is approached with introduction of a semi-implicit temporal integration scheme in this dissertation. In the semi-implicit approach, temporal integration of discrete terms in

the body-normal direction is treated implicitly thus ensuring that information transfer in this direction is instantaneous, much like information transfer across shear layers in "classical" boundary layer theory. Terms in the streamwise and cross stream directions are treated explicitly, thus modeling the hyperbolic nature of the information transfer in those directions.

Linear stability analysis of the scheme shows that this approach results in an elimination of that portion of the stability restriction that is due to grid spacing in the body-normal direction. Instead, the maximum time step is restricted only by grid spacing in the streamwise and cross stream directions. Thus, the numerical stiffness that is due to disparate physical scales found in viscous flows is eliminated, and the governing partial differential equations can be integrated with relative efficiency.

The scheme is applied to several inviscid and viscous problems, and its performance is compared to that of a fully-implicit scheme and an explicit scheme equipped with multigrid and residual smoothing. Results confirm the efficiency of this approach. While the explicit scheme is slightly more efficient for the inviscid solutions, the semi-implicit algorithm performed better than the explicit and implicit schemes on the model viscous problem, where highly stretched grids are required. Compared to the explicit method, the semi-implicit method was $3 - 10$ times faster for the three Reynolds numbers investigated. Similar comparisons with the implicit method gave up to seven times faster calculations.

9.2 Study of Flow Separation near the Blunt Leading Edges of Delta Wings

The second goal of this research, to examine and identify flow separation mechanisms at rounded leading edges, is achieved by studying leading edge flows at several test conditions. Three-dimensional laminar vortical flows over two 65° swept semi-infinite elliptical wings of thickness to chord ratio 1 : 11.55 and 1 : 20 at $M_\infty = 1.6$, $Re_L = 10^6$, and angles of attack of 4°, and 8°, and a 60° swept elliptical wing with $t/c = 1 : 11.55$ at $M_\infty = 1.4$, $Re_L = 2 \times 10^6$ and $\alpha = 14°$ are considered. The elliptical wings are truncated at $x = 0.1$ and $x = 1.0$ and a conical Navier-Stokes solver is used to supply boundary conditions at $x = 0.1$ and initial conditions.

The thin layer form of the three-dimensional Navier-Stokes equations is used to calculate solutions on computational grids with 112000 and 175000 grid points. A nonlinear instability is found for semi-implicit calculations of separated flows. Calculations of separated flows diverge when local time-stepping is used. The instability is not well understood. It appears to be numerical in nature. Nevertheless, convergence is obtained if time steps are set to a constant at each streamwise station, and the semi-implicit scheme is significantly more efficient than the corresponding explicit scheme. Computations are validated by a examination of the effect of numerical parameters such as artificial viscosity coefficients, grid refinement, and location of the starting solution, and by comparison of numerical results to experimental data. Overall features of the flow are discussed and a consistent description of the flow field is given.

The leading edge flow is examined with the aid of velocity vector plots, particle path integrations, Mach number and stagnation pressure contour plots and profiles of pressure, Mach number, and density near the leading edges. Primary and, in some

cases, secondary separation is observed In all cases, primary separation occurs on the leeward side of the wing somewhat inboard of the leading edge. Separation lines are found to move towards the leading edges as the angle of attack is increased and the curvature of the leading edge is decreased.

Two distinct leading edge separation mechanisms are identified. The flow at the leading edges of the $t/c = 20$ ($\alpha = 8°$) and $t/c = 11.55$ ($\alpha = 4°$, $8°$, $14°$) cases is characterized by a weak shock wave that impinges on the boundary layer just leeward of the leading edge and extends a short distance into the flow field. The shock wave is precipitated by a wedge-like displacement of the supersonic leading edge flow, which is due to recirculating flow underneath and outbords of the primary vortex core. A pressure jump associated with the shock wave and, in some cases, with additional flow recompression in this region induces boundary layer separation.

In the $t/c = 11.55$ ($\alpha = 4°$) solution, separation is caused by an adverse pressure gradient due to flow shock-less recompression inboards of the leading edge expansion. This case differs from the $t/c = 20$ test cases in its lower leading edge curvature, and corresponding lower maximum Mach number near the leading edge. Thus, although cross flow streamlines at a typical wing station still show a wedge-like flow diversion near the leading edge, no shock wave forms. The separation is located several cross stream stations inboard of the separation lines in the shock-induced separation cases.

9.2.1 Recommendations for Future Research

A number of areas of research merit further attention:

- Although the semi-implicit scheme showed substantial improvement in efficiency compared to conventional schemes, further improvement of its performance seems possible. Different implementations of the semi-implicit time stage should be investigated, especially for steady state calculations. For example, performing a semi-implicit integration on every second or third stage, or "freezing" values of the implicit matrix for several stages may save CPU expenses. Use of the semi-implicit scheme in viscous regions of the flow field only might be advantageous. Exploratory research in these areas by the author has given promising results.

- The nonlinear instability of the scheme found to occur in calculations of separated flows should be examined, and, if possible, removed.

- Other "acceleration" mechanisms such as grid-sequencing and matrix diagonalization might be implemented to attain additional improvements in efficiency.

- Use of the semi-implicit approach in conjunction with other spatial approximations should be considered. In particular, use with a higher order difference scheme or an upwind scheme is of interest. Issues to be considered in this context are computational efficiency of the matrix inversion and possible use of an approximate linearization for the implicit terms to reduce matrix bandwidth.

- Characterization of flow separation processes for blunt delta wings should be continued. Of special interest would be a study of flow cases with leading edge curvatures ranging between that of the current "Wing T" and infinity (flat plate). A parametric computational and experimental study similar to that of [Miller & Wood 83] but with wings of several different leading edge curvature and different Reynolds numbers would be expensive but could be very instructive.

- Extension of the classical separation criterion by examination of the full or thin layer Navier-Stokes equations could prove to be fruitful.

- The effect of turbulence on vortical flow features and flow separation process should be considered.

This page intentionally left blank.

Bibliography

[Ababarnel et al. 86] Ababarnel, S., Bayliss, A., and Lustman, L., "Non-Reflecting Boundary Conditions for Compressible Navier-Stokes Equations", ICASE Report 86-9, 1986.

[Acton 70] Acton, F., Numerical Methods That Work, Harper and Row, New York, 1970.

[Agarwal & Deese 85] Agarwal, R.K. and Deese, J.E., "Numerical Solution of Shock/Boundary Layer Interaction Using a Finite-Volume Runge-Kutta Time-Stepping Scheme", paper presented at 3^{rd} Symposium on Numerical and Physical Aspects of Aerodynamic Flows, Long Beach, California, 20–24 January 1985.

[Allmaras & Giles 89] Allmaras, S. R. and Giles, M. B., "A Coupled Euler/Navier-Stokes Algorithm for 2–D Unsteady Transonic Flows", *AIAA* Paper # 89-0556.

[Anderson et al. 84] Anderson, D.A., Tannehill, J.C., and Pletcher, R.H., Computational Fluid Mechanics and Heat Transfer, McGraw-Hill Book Company, New York, 1984.

[Baldwin & Lomax 78] Baldwin, B.S. and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows", *AIAA* Paper 78-257, Huntsville, Alabama, 1978.

[Baker 87] Baker, T.J., "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", *AIAA* Paper 87-1124, 1987.

[Baker 88] Baker, T.J., "Developments and Trends in Three Dimensioal Mesh Generation", Review paper presented at the Transonic Symposium, NASA Langley Research Center, Hampton, Virginia, April 19-21, 1988.

[Beam & Warming 76] Beam, R.W. and Warming, R.F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", *AIAA* Journal Vol. 16, pp. 393-401.

[Benek 87] Benek, J.A., Donegan, T.L., and Suhs, N.E., "Extended Chimera Grid Embedding Scheme With Application to Viscous Flows", *AIAA* Paper # 87-1126, 1987.

[Bluford 79] Bluford, G. S., Jr., "Numerical Solutions of the Supersonic and Hypersonic Viscous Flows Around Thin Delta Wings", *AIAA* Journal, Vol. 17, No. 9, Article n0. 78-1136R, pp. 942-949, September, 1979.

[Brandt 80] Brandt, A., "Multi-Level Adaptive Computations in Fluid Dynamics", *AIAA Journal* Vol. 18, No. 10, October 1980, pp. 1165-1172.

[Buning & Steger 85] Buning, G. P. and Steger, J. L., "Graphics and Flow Visualization in Computational Fluid Dynamics", *AIAA* Paper 85-1507, July 1985.

[Bussing 85] Bussing, T. R. A., "A Finite Volume Method for the Navier-Stokes Equations with Finite Rate Chemistry", Ph.D. thesis,

Department of Aeronautics and Astronautics, MIT, August 1985.

[Cebeci 74] Cebeci, T. and Smith, A. M. O., Analysis of Turbulent Boundary Layers, Academic Press, New York, 1974.

[Chapman 79] Chapman, D. R., "Computational Aerodynamics Development and Outlook", *AIAA Journal*, Vol. 17, 1979, pp. 1293-1313.

[Choo et al. 87] Choo, Y.K., Yoon, S., Civinskas, K. "Composite Grid and Finite Volume LU Implicit Scheme for Turbine Flow Analysis", *AIAA* Paper 87-1129, 1987.

[Chima 85] Chima, R.V., "Efficient Solution of the Euler and Navier-Stokes Equations with a Vectorized Multiple-Grid Algorithm", *AIAA* Journal, 23:23-32, Jan. 1985.

[Cross 71] Cross, E.J., "Experimental and Analytical Investigation of the Expansion Flow Field over a Delta Wing at Hypersonic Speeds", Aerospace Reaserch Laboratories, ARL 68-0027, Wright-Patterson Air Force Base, Ohio, Aug. 1971.

[Davis & Rubin 80] Davis, R.T. and Rubin, S.G., "Non-Navier-Stokes Viscous Flow Computations, *Comput. Fluids*, vol. 8, pp. 101-131.

[Davis & Dannenhoffer 89] Davis, R.L. and Dannenhoffer, J., "Adaptive Grid Embedding Navier-Stokes Technique for Cascade Flows", *AIAA* Paper 89-0204, 1989.

[Davis et al. 87] Davis, R.L., Ni, R., and Carter, J.E., "Cascade Viscous Flow

Analysis Using the Navier-Stokes Equations", *AIAA* Paper 86-0033, 1986.

[Deiwert 75] Deiwert, G. S., "Numerical Simulation of High Reynolds Number Transonic Flows", *AIAA Journal* Vol. 13, 1975, pp. 1354-1359.

[Drela 83] Drela, M., "A New Transformation and Integration Scheme for the Compressible Boundary Layer Equations, and Solution Behaviour at Separation", GTL Report 172, M. I. T., 1983.

[Drela 86] Drela, M., "Two-Dimensional Transonic Aerodynamic Design and Analysis Using the Euler Equations", GTL Report 187, M. I. T., 1986.

[Dwoyer & Thomas 81] Dwoyer, D. L. and Thomas, F. C., "Temporal and Spatial Inconsistencies of Time-Split Finite-Difference Schemes", *NASA* TP-1790, 1981.

[Eriksson & Rizzi 83] Eriksson, L.-E. and Rizzi, A., "Analysis by Computer of the Convergence to Steady State of Discrete Approximations to the Euler Equations", *AIAA* Paper 83-1951 in *Proceedings of AIAA Computational Fluid Dynamics Conference*, Danvers, 1983.

[Fuji & Obayashi 86] Fuji, K. and Obayashi, S., "Practical Applications of New LU-ADI Scheme for the Three-Dimensional Navier-Stokes Computation of Transonic Viscous Flow", *AIAA* Paper # 86-0513, 1985.

[Fuji & Obayashi 85] Fuji, K. and Obayashi, S., "Evaluation of Euler and Navier-Stokes Solutions for Leading Edge and Shock-Induced Sepa-

rations", *AIAA* Paper # 85-1563, 1985.

[Fuji & Kutler 83] Fuji, K. and Kutler, P., "Numerical Simulation of the Leading-Edge Vortex for a Wing and Strake-Wing Configuration", *AIAA* Paper # 83-1908, 1983.

[Fuji & Kutler 84] Fuji, K. and Kutler, P., "Numerical Solution of the Viscous Flow Fields over Three-Dimensional Complicated Geometries", *AIAA* Paper # 84-1550, 1984.

[Giles 88] Giles, M. "Accuracy of Node-Based Solutions on Irregular Meshes", to be presented at 11^{th} International Conference on Numerical Methods in Fluid Dynamics, June 1988.

[Goodsell 87] Goodsell, A., "3-D Euler Calculations of Vortex Flows over Delta Wings", *CFDL*-TR-87-6, MIT Dept. of Aeronautics and Astronautics, July 1987.

[Hafez et al. 87] Hafez, M., Palaniswamy, S., Kuruvila G., Salas, M.D., "Application of Wynn's ϵ-Algorithm to Transonic Flow Calculations", *AIAA* Paper 87-1143, 1987.

[Hänel et al. 87] Hänel, D., Schwane, R., and Seider, G., "On the Accuracy of Upwind Schemes for the Solution of the Navier-Stokes Equations", *AIAA* Paper 87-1105, 1987.

[Harlow & Nakayama 68] Harlow, F.H. and Nakayama, P.I., "Transport of Turbulence Energy Decay Rate", Los Alamos Scientific Laboratory Report LA-3854, Los Alamos, New Mexico, 1968.

[Holst et al. 87] Holst, T.L., Kaynak, U., Gundy, K.L., Thomas, S.D., Flores, J., and Chaderjian, N.M., "Transonic Wing Flows Using an

Euler/Navier-Stokes Zonal Approach", *J. Aircraft*, Vol. 24, No. 1, Jan. 1987.

[Jameson et al. 81] Jameson, A., Schmidt, W., and Turkel, E. "Numerical Solutions of the Euler Equations Using Runge-Kutta Time Stepping Schemes", *AIAA* Paper 81-1259, 1981.

[Jameson 83] Jameson, A., "Solution of the Euler Equations by a Multigrid Method", *Applied Math. and Computation*,13:327-356, June 1983.

[Jameson & Baker 83] Jameson, A. and Baker, T. J., "Solution of the Euler Equations for Complex Configurations", *AIAA* Paper 83-1929CP, in *Proceedings of AIAA Computational Fluid Dynamics Conference*, Danvers, 1983.

[Jameson 86] Jameson, A., "A Vertex Based Multigrid Algorithm for Three Dimensional Compressible Flow Calculations", *ASME* Symposium on Numerical Methods for Compressible Flows, Annaheim, Dec. 1986.

[Jayaram & Jameson 88] Jayaram, M. and Jameson, A., "Multi-Grid Solution of the Navier-Stokes Equations for Flow over Wings", AIAA-88-0705, 1988.

[Johnson & King 84] Johnson, D. A. and King, L. S., "A New Turbulence Closure Model for Boundary Layer Flows with Strong Adverse Pressure Gradients and Separation", AIAA-84-0175, Reno, Nevada, 1984.

[Kallinderis & Baron 87] Kallinderis, J. G. and Baron, J. R., "Adaptation Methods for a New Navier-Stokes Algorithm", *AIAA* Paper # 87-1167.

[Kallinderis & Baron 88] Kallinderis, J. G. and Baron, J. R., "Unsteady and Turbulent Flow using Adaptation Methods", paper presented at 11^{th} International Conference on Numerical Methods in Fluid Dynamics, Williamsburg, Virginia, June 27 – July 1, 1988.

[Keller 70] Keller, H.B., "Accurate Numerical Methods for Boundary Layer Flows: I. Two Dimensional Laminar Flows", *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*, Von Karman Institute for Fluid Dynamics, Sept. 15-19, 1970.

[Kennelly 87] Kennelly, R. A., "QPLOT", Users manual published by Sterling Software/NASA Ames, 19 May 1987.

[Krouthen 88] Krouthen, B., "A Numerical Investigation of Hot Streaks in Turbines", MIT Computational Fluid Dynamics Laboratory, CFDL-TR-88-9, 1988.

[Lax & Wendroff 60] Lax, P.D. and Wendroff, B., *Commun. Pure Appl. Math.* Vol. 13, 217-237, 1960.

[Lerat & Sides 79] Lerat, A. and Sides, J., "Finite Volume Methods for the Solution of the Euler Equations", ONERA TP No. 1979-155, 1979.

[Lindquist 88] Lindquist, D.R., "A Comparison of Numerical Schemes on Triangular and Quadrilateral Meshes", CFDL-TR-88-6, M.I.T., 1986.

[Löhner 88] Löehner, R., "Generation of Three-Dimensional Unstructured grids by the Advancing Front Method", *AIAA* Paper 88-0515, January 1988.

[Loyd & Murman 86] Loyd, B. and Murman, E. M., "Finite Volume Solution of the Compressible Boundary Layer Equations", *NASA* CR 4013, October 1986.

[Loyd et al. 86] Loyd, B., Murman, E. M., and Abarbanel, S., "Semi-Implicit Solution of the Navier-Stokes Equations", presented at the *Society for Industrial and Applied Mathematics* 1986 National Meeting, Boston, July 21-25, 1986.

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

[MacCormack 81] MacCormack, R. W., "A Numerical Method for Solving the Equations of Compressible Viscous Flows", *AIAA* Paper # 81-110.

[MacCormack 84] MacCormack, R. W., "Numerical Methods for the Navier-Stokes Equations", in *Progress and Supercomputing in Computational Fluid Dynamics*, proceedings of U.S.–Israel Workshop, 1984. # 81-110.

[Marconi 83] Marconi, F., "The Spiral Singularity in the Supersonic Inviscid Flow Over a Cone", *AIAA* Paper #83-1665.

[Martinelli 86] Martinelli, L., Jameson, A., and Grasso, F., "A Multigrid Method fot the Navier-Stokes Equations", *AIAA* Paper 86-0208.

[Mavriplis et al 89] Mavriplis, D. J., Jameson A., and Martinelli, L., "Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes", *AIAA* Paper 89-0120.

[McDonald & Briley 75] McDonald, H. and Briley, W.R., "Three Dimensional Supersonic Flow of a Viscous or Inviscid Gas", *J. Comp. Phys.* Vol. 19, pp. 150-178, 175.

[McRae 76] McRae, D. S., "A Numerical Study of Supersonic Viscous Cone Flows at High Angle of Attack", *AIAA* Paper #76-77, June 1976.

[Melnik 87] Melnik, R., Personal communication.

[Miller & Wood 83] Miller, D. S. and Wood, R. M., "An Investigation of Wing Leading-Edge Vortices at Supersonic Speeds", *AIAA* Paper # 83-1816, July, 1983.

[Müller & Rues 86] Müller, B. and Rues, D., "Finite-Difference Method for the Simulation of Three-Dimensional Laminar Supersonic Flow over Blunt Bodies", DFVLR-IB, Göttingen, 1986.

[Müller & Rizzi 87] Müller, B. and Rizzi, A., "Navier-Stokes Calculations of Transonic Vortices over a Round Leading Edge Delta Wing", *AIAA* Paper 87-1227, 1987.

[Murman et al. 85] Murman, E. M., Rizzi, A., and Powell, K. G., "High Resolution Solutions of the Euler Equations for Vortex Flows", In *Progress and Supercomputing in Computational Fluid Dynamics*, Birkhauser-Boston, 1985, pp. 93-113.

[Nakahashi 88] Nakahashi, K., "A Finite Element Method on Prismatic Elements for the Three Dimensional Navier-Stokes Equations", 11^{th} International Conference on Numerical Methods in Fluid Dynamics, Williamsburg, Virginia, June 1988.

[Navier 1823] Navier, M., "Memoires sur les lois du mouvement des Fluids", Mem. Acad. Sci., Vol. 6, pp. 389-416, 1823.

[Newsome & Thomas 85] Newsome, R. W. and Thomas, J. L., "Computation of Leading-Edge Vortex Flows", NASA CP-2416, pp. 305-330, Oct. 1985.

[Newsome 85] Newsome, R. W., "A Comparison of Euler and Navier-Stokes Solutions for Supersonic Flow Over a Conical Wing", *AIAA* Paper 85-0111, 1985.

[Ni 81] Ni, R.-H., "A Multiple Grid Scheme for Solving the Euler Equations", *AIAA Journal*, 20:1565-1571, Nov. 1981.

[Norton et al. 84] Norton, R. J. G., Thompkins, W. T., Haimes, R., "Implicit Finite Difference Schemes with Non-Simply Connected Grids – A Novel Approach", *AIAA* Paper # 84-0003, 1984.

[Obremski et al. 69] Obremski, H.J., Morovkin, M.V., Landahl, M.T., "A Portfolio of Stability Characteristics of Incompressible Boundary Layers", AGARDograph 134, Neuilly Sur Seine, France, 1969.

[Peraire et al. 87] Peraire, J., Morgan., K., Peiro, J., and Zienkiewicz, O. C., "An Adaptive Finite Element Method for High Speed Flows", *AIAA* Paper # 87-0558, 1987.

[Peyret & Taylor 83] Peyret, R. and Taylor, T. D., Computational Methods for Fluid Flow, Springer-Verlag, New York.

[Powell 87] Powell, K. G., "Vortical Solutions of the Conical Euler Equations", Ph. D thesis, M.I.T., 1987.

[Prandtl 04] Prandtl, L., "Uber Flussigkeitsbewegung bei sehr kleiner Reibung", Proc. 3^{rd} Int. Math. Congr., Heidelberg, 1904.

[Pulliam & Steger 80] Pulliam, T. H. and Steger, J., L., "Implicit Finite Difference Simulations of Three-Dimensional Compressible Flow", *AIAA* Journal, Vol. 18, No. 2, pp. 159-167, Feb. 1980.

[Pulliam & Chaussee 81] Pulliam, T.H. and Chaussee, D.S., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm", *Journal of Computational Physics* 37,347-383, 1981.

[Pulliam 84] Pulliam, T. H., "Euler and Thin Layer Navier-Stokes Codes: ARC2D, ARC3D", Computational Fluid Dynamics Workshop, UTSI, Mar. 1984, pp 15.1 - 15.85.

[Pulliam 85] Pulliam, T.H., "Artificial Dissipation Models for the Euler Equations", *AIAA* Paper # 85-0438, 1985.

[Richtmyer & Morton 67] Richtmeyer, R. D. and Morton, K. W., Difference Methods of Initial Value Problems, Interscience Publishers, New York, 1967.

[Riedelbauch & Müller 87] Riedelbauch, S. and Müller, B., "The Simulation of Three-Dimensional Viscous Supersonic Flow Past Blunt Bodies with a Hybrid Implicit/Explicit Finite-Difference Method", DFVLR-FB 87-32.

[Rizetta & Shang 86] Rizetta, D. P. and Shang, J. S., "Numerical Simulation of Leading-Edge Vortex Flows", *AIAA Journal* Vol. 24, No. 2, February 1986.

[Rizk & Chausee 83] Rizk, Y. M. and Chaussee, D. S., "Three Dimensional Viscous-

Flow Computation Using a Directionally Hybrid Implicit-Explicit Procedure", AIAA Paper 83-1910-CP, 1983.

[Rizzi 86] Rizzi, A., "Three-Dimensional Solutions to the Euler Equations with One Million Grid Points", *Journal of Fluid Mechanics*, Vol. 148, November 1984.

[Roberts 86] Roberts, T. W., "Euler Equation Computations for the Flow Over a Hovering Helicopter Rotor", Ph.D thesis, MIT 1986.

[Roe 86] Roe, P., "Characteristic-Based Schemes for the Euler Equations", *Ann. Rev. Fluid Mech.* 18:337-65, 1986.

[Rogers & Berry 55] Rogers, E.W.E. and Berry, C.J., "Experiments at M=1.41 on Elliptic Cones with Subsonic Leading Edge", A.R.C.R.&M. No. 3042, Oct. 1955.

[Rotta 51] Rotta, J., "Statistische Theorie nichthomogener Turbulenz", *Z. Phys.*, vol. 1, (B.E. Launder, ed.), Academic, New York, 1951.

[Rudman & Rubin 68] Rudman, S. and Rubin S. G., "Hypersonic Viscous Flow over Slender Bodies with Sharp Leading Edges", *AIAA Journal*, vol. 6, pp. 1883-1889, 1968.

[Ruffin 87] Ruffin, S. M., "Solutions for Hypersonic Viscous Flow Over Delta Wings", S.M. Thesis, Massachusetts Institute of Technology, 1987.

[Salas & Daywitt 79] Salas, M. D. and Daywitt, J., "Structure of the Conical Flow about External Axial Corners", *AIAA* Journal, vol. 17, no. 1, pp. 41-47, Jan. 1979.

[Salas et al. 83] Salas, M. D., Jameson, A., Melnik, R. E., "A Comparative Study of the Nonuniqueness Problem of the Potential Equation", *AIAA* Paper # 83-1888, 1983.

[Schlichting 68] Schlichting, H., Boundary Layer Theory, Sixth ed. McGraw Hill Book Co. Inc., New York, c. 1968.

[Shmilovich & Caughey 81] Shmilovich, A. and Caughey, D. A., "Application of the Multi-Grid Method to Calculations of Transonic Potential Flow about Wing-Fuselage Combinations", in Multigrid Methods, NASA Conference Publication 2202, 1981.

[Stanbrook & Squire 64] Stanbrook, A. and Squire, L. C., "Possible Types of Flos at Swept Leading Edges", *Aeronautical Quarterly*, Vol. 15, No. 1, pp. 72-82, February 1964.

[Steger 77] Steger, J. L., "Implicit Finite-Difference Simulation of Flow about Arbitrary Geometries with Application to Airfoils", *AIAA* Paper 77-665, Albuquerque, New Mexico, 1977.

[Stokes 1845] Stokes, G. G., "On the Theories of Internal Friction of Fluids in Motion," Trans. Camb. Philos. Soc. Vol. 8, pp. 287-305, 1845.

[Swanson & Turkel 85] Swanson, R. C. and Turkel, E., "A Multistage Time-Stepping Scheme for the Navier-Stokes Equations", *AIAA* Paper 85-0035.

[Thomas & Walters 85] Thomas, J. L. and Walters, R. W., "Upwind Relaxation Algorithms for the Navier-Stokes Equations", *AIAA* Paper # 85-1501.

[Thomas & Newsome 86] Thomas, J. L. and Newsome, R. W., "Navier-Stokes Computations of Lee-Side Flows over Delta Wings", *AIAA* Paper # 86-1049, 1986.

[Thomas et al. 87] Thomas, J. L., Taylor S. L., Anderson, W.K., "Navier-Stokes Computations of Vortical Flows over Low Aspect Ratio Wings", AIAA Paper 87-0207, January, 1987.

[Thompson et al. 85] Thompson, J. F., Warsi, Z. U. A., and Mastin, W. C., Numerical Grid Generation – Foundations and Applications, Elsevier Science Publishing Co., Inc., New York, 1985.

[van Leer 82] van Leer, B., "Flux Vector Splitting for the Euler Equations", ICASE Report No. 82-30, September, 1982.

[van Leer et al. 87] van Leer, B., Thomas, J.L., and Roe, P.L., "A Cmparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations", *AIAA* Paper 87-1104, June 1987.

[Warming & Beam 79] Warming, R. F. and Beam, R. M., "Extension of *A*-Stability to ADI Methods", *BIT* Vol. 19, p. 395-417, 1979.

[Whitfield 86] Davis, R.L., Personal communication to Y. Kallinderis, 1986.

[Wigton 85] Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes", *AIAA* Paper # 85-1494, 1984.

Appendix A

Stability Characteristics of the Temporal Integration

This appendix presents derivations and comparisons of the stability characteristics of the semi-implicit scheme with those of the four stage explicit scheme given in Chapter 4, and of the fully implicit algorithm of [Beam & Warming 76]. The scalar $2-D$ convection/diffusion equation

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \mu \frac{\partial^2 u}{\partial y^2} \,, \tag{A.1}$$

is used to model the thin layer Navier-Stokes equations. The constant μ is the coefficient of viscosity and, consistent with the thin layer assumption, the viscous terms in the tangential x direction are assumed to be zero. Von Neumann stability analyses [Richtmyer & Morton 67] are performed to to evaluate and compare the three schemes.

The analysis is done in two dimensions to allow straightforward graphical representation of the results. The conclusions, however, apply as well to the three dimensional case, except for the approximate factorization used with the fully implicit scheme. Although stability analyses of the explicit and fully implicit schemes can be found in the technical literature, a review of that analysis is given below for background and to allow comparison with the semi-implicit scheme.

A.1 Stability of the Explicit Multistage Scheme

The four stage explicit scheme of Chapter 4 applied to Equation A.1 can be written as

$$\begin{aligned} u^0_{i,j} &= u^n_{i,j} \\ u^1_{i,j} &= u^0_{i,j} + \alpha_1 \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^0_{i,j} \\ u^2_{i,j} &= u^0_{i,j} + \alpha_2 \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^1_{i,j} \\ u^3_{i,j} &= u^0_{i,j} + \alpha_3 \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^2_{i,j} \\ u^4_{i,j} &= u^0_{i,j} + \quad \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^3_{i,j} \\ u^{n+1}_{i,j} &= u^4_{i,j} \ , \end{aligned} \tag{A.2}$$

with the discrete operators, δ_x, δ_y, and δ_{yy} defined as

$$\begin{aligned} \delta_x u^n_{i,j} &= \frac{u^n_{i+1,j} - u^n_{i-1,j}}{2\Delta x} \\ \delta_y u^n_{i,j} &= \frac{u^n_{i,j+1} - u^n_{i,j-1}}{2\Delta y} \\ \delta_{yy} u^n_{i,j} &= \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2} \ . \end{aligned} \tag{A.3}$$

By implication, the computational grid is rectangular and has constant spacing in the x and y directions.

The stages in (A.2) may be combined to give

$$\begin{aligned} u^{n+1}_{i,j} = \ & [\, 1 + \Delta t(\delta_x + \delta_y + \mu\delta_{yy}) + \alpha_3(\Delta t)^2(\delta_x + \delta_y + \mu\delta_{yy})^2 \\ & + \alpha_3\alpha_2(\Delta t)^3(\delta_x + \delta_y + \mu\delta_{yy})^3 + \alpha_3\alpha_2\alpha_1(\Delta t)^4(\delta_x + \delta_y + \mu\delta_{yy})^4]\, u^n_{i,j} \ . \end{aligned} \tag{A.4}$$

The algorithm is stable if, for all possible values of u, the magnitude of the amplification factor is less than unity[1]:

$$|G| = |u^{n+1}/u^n| \leq 1 \ . \tag{A.5}$$

[1] The strict criterion due to von Neumann is that $|u^{n+1}/u^n| \leq 1 + O(\Delta t)$. This allows the discrete solution to model problems whose exact solutions grow exponentially in time. It does not apply to steady-state problems.

To explore this condition, assume that an admissible solution can be represented by a sum of sinusoidal error modes[2]

$$\epsilon = \sum_{\theta,\phi=0}^{\pi} e^{\hat{i}i\theta} e^{\hat{i}j\phi} , \tag{A.6}$$

and examine the decay or amplification of each of these modes separately.

Now, set

$$u_{i,j}^n = \hat{u}_{i,j}^n e^{\hat{i}i\theta} e^{\hat{i}j\phi} , \tag{A.7}$$

where the Fourier coefficient $\hat{u}_{i,j}^n$ is the amplitude of a particular mode at time t^n, $\hat{i} = \sqrt{-1}$, and θ and ϕ are the phase angles in the x and y directions, respectively. Inserting (A.7) into (A.4) gives the amplification factor

$$G = 1 - z + \alpha_3 z^2 - \alpha_3\alpha_2 z^3 + \alpha_3\alpha_2\alpha_1 z^4 , \tag{A.8}$$

where the complex expression z is

$$z = \hat{i}(\lambda_x \sin\theta + \lambda_y \sin\phi) - \lambda_y \left(\frac{4\mu}{\Delta y} \sin^2(\phi/2)\right) , \tag{A.9}$$

and $\lambda_x = \Delta t/\Delta x$, $\lambda_y = \Delta t/\Delta y$. After some algebra, one finds that for the "classical" choice of integration constants

$$\alpha_1 = 1/4 \qquad \alpha_2 = 1/3 \qquad \alpha_3 = 1/2 , \tag{A.10}$$

the scheme is stable for $|z| \leq 2\sqrt{2}$, that is, $|G| \leq 1$ if

$$\left[(\lambda_x + \lambda_y)^2 + \lambda_y^2 \left(\frac{4\mu}{\Delta y}\right)^2\right]^{1/2} \leq 2\sqrt{2} , \tag{A.11}$$

or

$$\Delta t \leq 2\sqrt{2} \frac{1}{\left[\left(\frac{1}{\Delta x} + \frac{1}{\Delta y}\right)^2 + \frac{1}{\Delta y^2}\left(\frac{4\mu}{\Delta y}\right)^2\right]^{1/2}} . \tag{A.12}$$

[2]This Fourier approach also implies that boundary conditions of the solution are also periodic.

The factor multiplying this expression, commonly referred to as the *CFL* number, gives the maximum Δt for which the numerical domain of dependence of the scheme includes the physical domain of dependence. The viscous portion of (A.12) ensures that the numerical diffusion time does not exceed the physical diffusion time.

The Δt limitation can result in a high cost of computation. While each cell may be advanced by its own time step for steady-state solutions, the minimum $\Delta t_{i,j}$ must be used in the temporal integration of all cells in time-accurate calculations. In either case, severe restrictions on the size of the time step occur if the computational grid is fine such that Δx, $\Delta y \ll 1$, or if grid spacing in one coordinate direction is much smaller than spacing in another. The latter condition is usually present in Navier-Stokes calculations because rapid changes of velocity and temperature through the boundary layer must be resolved. Thus, the time steps in explicit Navier-Stokes algorithms are usually smaller than needed for temporal accuracy and result in large iteration counts and correspondingly high CPU costs.

A.1.1 Graphical Analysis

Study of the variation of amplification factor with error phase angles can give additional insight into the temporal characteristics of the scheme. In steady-state calculations large areas of $|G(\theta, \phi)| \ll 1$ are desirable since this implies that errors at most frequencies will be rapidly damped. All stability graphs shown in this appendix will assume an infinite Reynolds number. This eliminates the problem of having to choose a particular value. Since viscous effects due to a finite Reynolds number tend to stabilize calculations, this assumption presents a worst case scenario.

Figure A.1 shows contours of the amplification factor for the four stage explicit

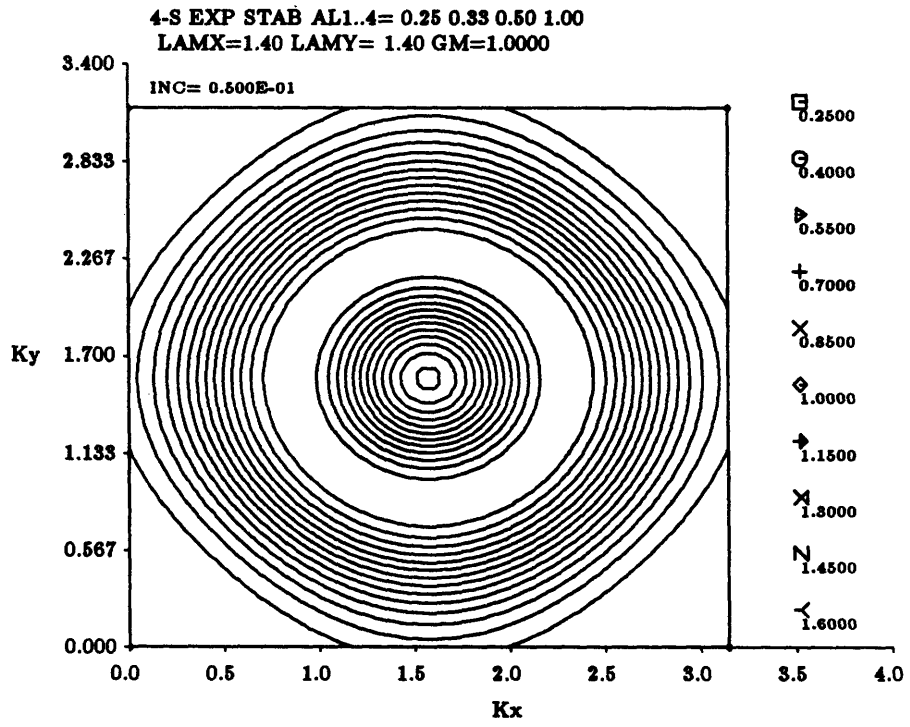

Figure A.1: Stability Contours of Explicit Scheme

scheme for $\lambda = \lambda_x + \lambda_y = 2.8$ and $\mu = 0$. The value of the contour at each pair of wave numbers is the magnitude of G. The wave numbers are the frequencies of the error modes, and the magnitude of the amplification factor $|G(k_x, k_y)|$ is the rate of temporal growth of that error mode. Modes corresponding to $|G|$ greater than one will grow and eventually cause divergence of the solution algorithm.

Contours of $|G|$ are concentric circles[3], with $|G| = 1$ at $(k_x, k_y) = (0,0), (\pi, \pi)$, as required for consistency of the scheme, and $|G| = 1$ at $(k_x, k_y) = (\pi/2, \pi/2)$ as well.

[3]In this case, $|G|$ is only a function of one variable (z) and could be displayed as a line plot. For consistency with later stability contours we show it as as contour plot. The distortion is due to the plotting package.

A.2 Stability of a Fully Implicit Scheme

The general form of the Beam & Warming temporal discretization can be written as [Beam & Warming 76]

$$\Delta^n u = \frac{\theta_1}{1+\theta_2}\Delta t\frac{\partial}{\partial t}(\Delta^n u) + \frac{\Delta t}{1+\theta_2}\frac{\partial}{\partial t}(u^n) + \frac{\theta_2}{1+\theta_2}\Delta^{n-1}u\ , \tag{A.13}$$

where $\Delta^n u = u^{n+1} - u^n$. Setting $\theta_1 = 1$ and $\theta_2 = 0$ in Equation A.13 gives the implicit backward Euler algorithm

$$\Delta^n u = \Delta t\frac{\partial}{\partial t}(\Delta^n u) + \Delta t\frac{\partial}{\partial t}(u^n)\ . \tag{A.14}$$

Using (A.14) and (A.3) to discretize Equation A.1 gives

$$[1 - \Delta t(\delta_x + \delta_y + \mu\delta_{yy})]\,\Delta^n u = \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^n\ . \tag{A.15}$$

Equation (A.15) has a matrix with large bandwidth on the left hand side which is very expensive to invert. Because of this, Equation A.15 is usually approximately factored to give:

$$[1 - \Delta t\delta_x]\,[1 - \Delta t(\delta_y + \mu\delta_{yy})]\,\Delta^n u \simeq \Delta t(\delta_x + \delta_y + \mu\delta_{yy})u^n \tag{A.16}$$

While this approximately factored (AF) scheme is used in nearly all computational work, it is instructive to examine first the stability characteristics of the unfactored algorithm.

Transposing the LHS of Equation A.15 and using the definition of $\Delta^n u$ gives the amplification factor for the unfactored algorithm

$$\begin{aligned} G &= 1 + \frac{\Delta t(\delta_x + \delta_y + \mu\delta_{yy})}{1 - \Delta t(\delta_x + \delta_y + \mu\delta_{yy})} \\ &= \frac{1}{1 - \Delta t(\delta_x + \delta_y + \mu\delta_{yy})}\ , \end{aligned} \tag{A.17}$$

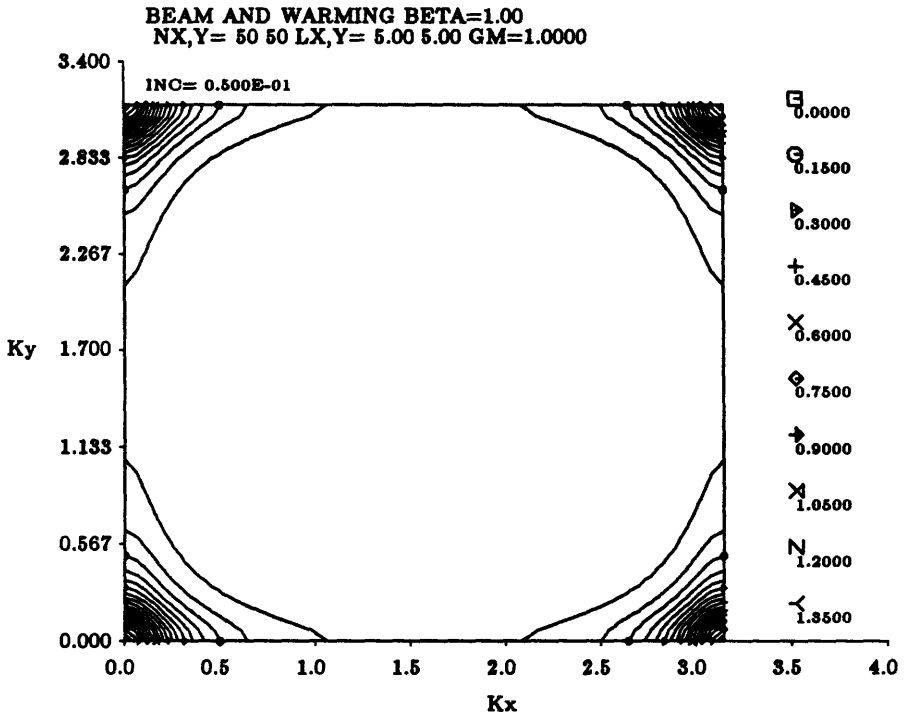

Figure A.2: Stability Contours of Un-Factored Backward Euler Scheme

or, after Fourier decomposing and reordering

$$G = \frac{1}{1 + y_\mu - i(x+y)}\,, \tag{A.18}$$

where $x = \lambda_x \sin\theta$, $y = \lambda_y \sin\phi$, and $y_\mu = \lambda_y(4\mu/\Delta y)\sin^2(\phi/2)$. Since the real part of the denominator of (A.18) alone is always greater than unity ($y_\mu \geq 0$), the amplification factor $|G| \leq 1$ and the scheme is unconditionally stable.

The contour plot of $|G|$ is given in Figure A.2. The magnitude of G in the unfactored algorithm approaches zero as Δt increases, thus giving the unfactored scheme excellent damping properties.

The approximately factored scheme is solved in two stages

$$\begin{aligned} [1 - \Delta t \delta_x] \Delta^n u' &= \Delta t(\delta_x + \delta_y + \mu\delta_{yy}) u^n \\ [1 - \Delta t(\delta_y + \mu\delta_{yy})] \Delta^n u &= \Delta^n u' \ , \end{aligned} \tag{A.19}$$

which may be combined to give

$$\begin{aligned} G &= 1 + \frac{\Delta t(\delta_x + \delta_y + \mu\delta_{yy})}{[1 - \Delta t\delta_x][1 - \Delta t(\delta_y + \mu\delta_{yy})]} \\ &= \frac{1 + \Delta t^2 \delta_x(\delta_y + \mu\delta_{yy})}{[1 - \Delta t\delta_x][1 - \Delta t(\delta_y + \mu\delta_{yy})]} \ . \end{aligned} \tag{A.20}$$

Fourier decomposition gives the AF amplification factor

$$G = \frac{1 - xy - ixy_\mu}{(1 - ix)(1 - iy + y_\mu)} \ , \tag{A.21}$$

Contours of $|G|$ for $y_\mu = 0$ ($Re \to \infty$) are shown for $\lambda = 5$ in Figure A.3. Unlike the unfactored scheme, $|G| \to 1$ over most of the domain as $\Delta t \to \infty$. This is due to the factorization error which grows at Δt^2. Thus, large Δt do not necessarily lead to fast convergence in an approximately factored implicit scheme. Numerical experiments show that there is in fact an optimal time step λ_{opt} which, for a given problem, gives the fastest convergence.

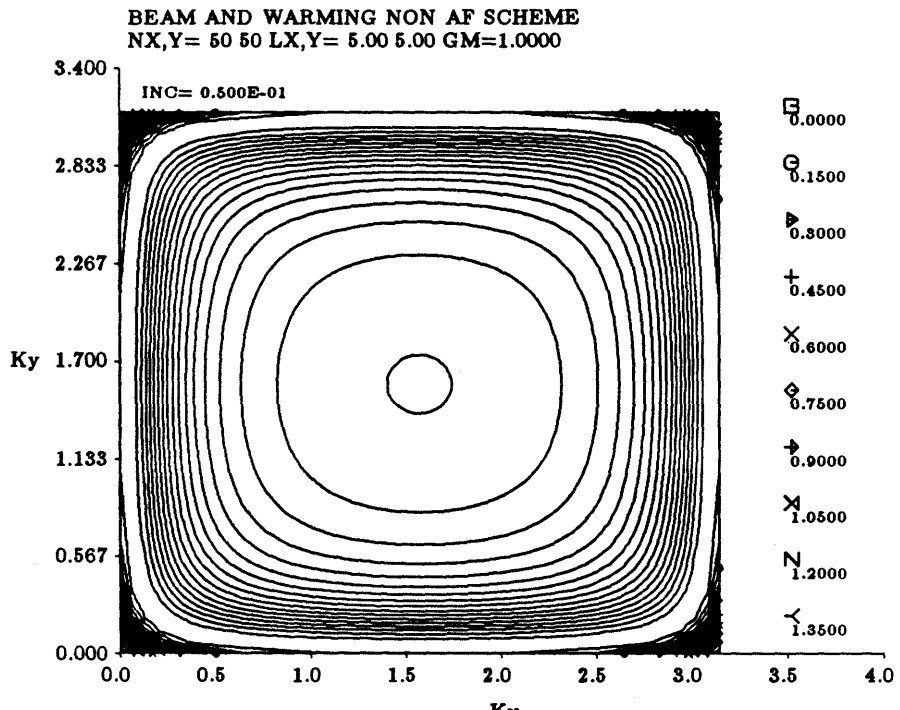

Figure A.3: Stability Contours of Factored Backward Euler Scheme

A.3 Stability of the Semi-Implicit Scheme

The semi-implicit 4-stage algorithm may be written in delta form as:

$$
\begin{aligned}
u^0_{i,j} &= u^n_{i,j} \\
(1-\alpha_1\Delta\delta_y)\Delta u^1_{i,j} &= \alpha_1\Delta t[\delta_x+\delta_y+\mu\delta_{yy}]u^0_{i,j} \\
(1-\alpha_2\Delta\delta_y)\Delta u^2_{i,j} &= \alpha_2\Delta t[\delta_x+\delta_y+\mu\delta_{yy}]u^1_{i,j} - \Delta u^1 \\
(1-\alpha_3\Delta\delta_y)\Delta u^3_{i,j} &= \alpha_3\Delta t[\delta_x+\delta_y+\mu\delta_{yy}]u^2_{i,j} - (\Delta u^2+\Delta u^1) \\
(1-\Delta\delta_y)\Delta u^4_{i,j} &= \quad \Delta t[\delta_x+\delta_y+\mu\delta_{yy}]u^3_{i,j} - (\Delta u^3+\Delta u^2+\Delta u^1) \\
u^{n+1}_{i,j} &= u^3_{i,j} + \Delta u^4
\end{aligned}
\tag{A.22}
$$

where $\Delta u^n = u^n - u^{n-1}$. Equations A.22 can be simplified by inserting the appropriate definitions for Δu, giving the system of equations:

$$\begin{aligned}
u^0_{i,j} &= u^n_{i,j} \\
(1 - \alpha_1 \Delta t(\delta_y + \mu\delta yy))u^1_{i,j} &= u^0_{i,j} + \alpha_1 \Delta t \delta_x u^0_{i,j} \\
(1 - \alpha_2 \Delta t(\delta_y + \mu\delta yy))u^2_{i,j} &= u^0_{i,j} + \alpha_2 \Delta t \delta_x u^1_{i,j} \\
(1 - \alpha_3 \Delta t(\delta_y + \mu\delta yy))u^3_{i,j} &= u^0_{i,j} + \alpha_3 \Delta t \delta_x u^2_{i,j} \\
(1 - \quad \Delta t(\delta_y + \mu\delta yy))u^4_{i,j} &= u^0_{i,j} + \quad \Delta t \delta_x u^3_{i,j} \\
u^{n+1}_{i,j} &= u^4_{i,j}
\end{aligned} \tag{A.23}$$

Inserting each stage in turn gives

$$\begin{aligned}
u^{n+1}_{i,j} = \Big[& \frac{1}{1 - \Delta t(\delta_y + \mu\delta yy)} + \frac{\Delta t \delta_x}{(1 - \Delta t(\delta_y + \mu\delta yy))(1 - \alpha_3 \Delta t(\delta_y + \mu\delta yy))} + \\
& \frac{\alpha_3 \Delta t^2 \delta_x^2}{(1 - \Delta t(\delta_y + \mu\delta yy))(1 - \alpha_3 \Delta t(\delta_y + \mu\delta yy))(1 - \alpha_2 \Delta t(\delta_y + \mu\delta yy))} + \\
& \frac{\alpha_3\alpha_2 \Delta t^3 \delta_x^3 + \alpha_3\alpha_2\alpha_1 \Delta t^4 \delta_x^4}{(1 - \Delta t(\delta_y + \mu\delta yy))(1 - \alpha_3 \Delta t(\delta_y + \mu\delta yy)(1 - \alpha_2 \Delta t(\delta_y + \mu\delta yy))(1 - \alpha_1 \Delta t(\delta_y + \mu\delta yy))} \Big] u^n_{i,j}
\end{aligned} \tag{A.24}$$

Fourier decomposing this equation as before gives the amplification factor:

$$\begin{aligned}
G = & \frac{1}{1 - iy + y_\mu} + \frac{ix}{(1 - iy + y_\mu)(1 - \alpha_3(iy - y_\mu)} + \\
& \frac{\alpha_3(ix)^2}{(1 - iy + y_\mu))(1 - \alpha_3(iy - y_\mu))(1 - \alpha_2(iy - y_\mu))} + \\
& \frac{\alpha_3\alpha_2(ix)^3 + \alpha_3\alpha_2\alpha_1(ix)^4}{(1 - iy + y_\mu))(1 - \alpha_3(iy - y_\mu))(1 - \alpha_2(iy - y_\mu))(1 - \alpha_1(iy - y_\mu))} ,
\end{aligned} \tag{A.25}$$

where x, y, and y_μ are defined as before.

The "worst case" $|G|$ occurs when at $y = y_\mu = 0$, such that the denominator of each term in (A.25) is unity, giving

$$G = 1 + ix + \alpha_3(ix)^2 + \alpha_3\alpha_2(ix)^3 + \alpha_3\alpha_2\alpha_1(ix)^4 . \tag{A.26}$$

The form of expression A.26 is identical to the form of the amplification factor of the explicit case and the magnitudes of the two factors are identical if z in (A.8) is replaced by ix. The stability is limited only by spacing in the x direction, which is treated explicitly, and a maximum time step $\lambda_x \leq 2\sqrt{2}$ or

$$\Delta t \leq 2\sqrt{2}\Delta x \tag{A.27}$$

may be taken.

Equation (A.27) shows that only the characteristic in the x-direction limits the propagation of numerical information. The physical domain of dependence corresponding to the characteristic in the y-direction is instantaneously accounted for by the simultaneous solution of the governing equations (A.24). The criterion is also independent of the level of viscosity. It is less restrictive than the explicit criterion (A.12) even for inviscid cases that have grid cells with $\Delta x = \Delta y$. When λ_y is large stability is enhanced, unlike the fully implicit AF scheme, since λ_y appears as a factor in the denominator of (A.25) only.

Amplification contours of the four stage semi-implicit scheme are given in Figure A.4, for $\lambda_x = 2.8$ and $\lambda_y = 50$. The contours are similar to contours of the unfactored implicit scheme in that $|G| \approx 0$ almost everywhere. The one dimensional nature of the temporal discretization is evident.

A.4 Extension to Three Dimensions

Analysis for the three dimensional versions of the explicit, semi-implicit, and fully implicit schemes proceeds much like the analyses above. In the explicit and semi-implicit cases the stability criterion is altered only by inclusion of appropriate terms correspond-

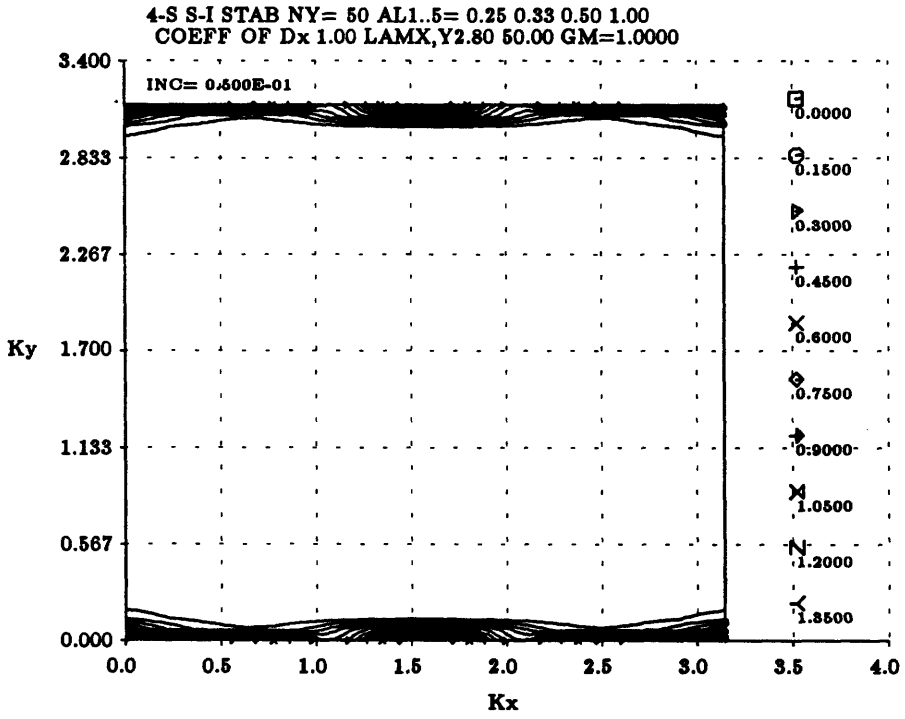

Figure A.4: Stability Contours of Four Stage Semi-Implicit Scheme $\lambda_y = 50$

ing to the third dimension. The terms δ_x and λ_x are replaced by the sums $\delta_x + \delta z$ and $\lambda_x + \lambda_z$, respectively. The forms of the stability criteria are unchanged. For the semi-implicit case, stability is still independent of spacing in the y-direction. However, stability analysis of the implicit scheme reveals the well known unconditional instability of approximately factored schemes in three dimensions [Warming & Beam 79], [Dwoyer & Thomas 81]. Although the instability is weak and can be suppressed by artificial viscosity, it merits concern.

A.5 Time Step Definition for the Thin Layer Navier-Stokes Equations

The stability constraints for explicit and semi-implicit schemes for the thin layer Navier stokes equations in generalized coordinates can be derived from their Cartesian counterparts for the convection/diffusion equation. The primary difference between the two, apart from the differing coordinate systems is in the number and form of the characteristics of the inviscid equations. The following two sections describe how the preceding results for the model equation may be used to deduce the form of the stability constraint for the thin layer Navier-Stokes equations.

A.5.1 Explicit Time Step

The explicit time step can be derived by requiring that the numerical domain of dependence of the scheme include the physical domain of dependence. This domain can be estimated by considering the Euler equations written in coordinates normal and tangential to a cell face. In the analysis that follows, ξ, ς, and η are coordinates

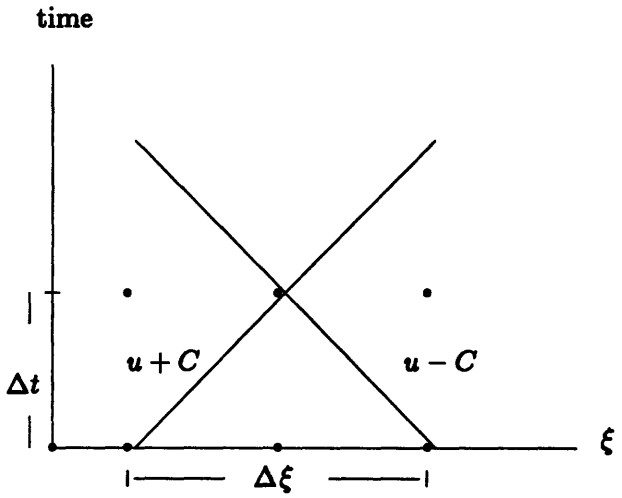

Figure A.5: One-Dimensional Domains of Dependence

in physical space with the body of interest located at a $\eta = Const.$ surface. The coordinates ξ and ς are in the streamwise and cross stream directions, respectively.

One can show (e.g. [Roberts 86]) that the Euler equations have five eigenvalues $(u_\xi, u_\xi, u_\xi, u_\xi + C, u_\xi - C)$, where u_ξ is the velocity in the ξ direction (normal to a $\varsigma\eta$ cell face), and C is the speed of sound. The physical domain of the solution is contained within the characteristics of this equation (see Fig. A.5). Thus, in one dimension the CFL condition requires that

$$\begin{aligned} \Delta t &= \lambda \frac{1}{\frac{u_\xi + C}{\Delta\xi}} \\ &= \lambda \frac{\Delta\xi}{u_\xi + C} \end{aligned} \tag{A.28}$$

The constant λ is the *CFL* number that was determined to be $\leq 2\sqrt{2}$ from the stability analysis.

In multiple spatial dimensions the CFL condition is somewhat of an approximation.

One assumes that the CFL condition must be satisfied in each of the dimensions and writes

$$\begin{aligned}\Delta t &\simeq \lambda \frac{1}{\frac{u_\xi + C}{\Delta\xi} + \frac{u_\eta + C}{\Delta\eta} + \frac{u_\zeta + C}{\Delta\zeta}} \\ &= \lambda \frac{\Delta\xi\Delta\eta\Delta\zeta}{u_\xi\Delta\eta\Delta\zeta + u_\eta\Delta\xi\Delta\zeta + u_\zeta\Delta\xi\Delta\eta + C(\Delta\eta\Delta\zeta + \Delta\xi\Delta\zeta + \Delta\xi\Delta\eta)}\end{aligned} \tag{A.29}$$

Each of the velocity terms in the denominator is nothing but the flux of velocity through one of the faces of the volume and the upper bound for the three terms may be written as

$$u_\xi\Delta\zeta\Delta\eta + u_\eta\Delta\xi\Delta\zeta + u_\zeta\Delta\xi\Delta\eta = |uS_x|_{max} + |vS_y|_{max} + |wS_z|_{max} \tag{A.30}$$

where S_x, S_y, and S_z are the projected areas of the cell in the x, y, and z directions, respectively. The product $\Delta\xi\Delta\eta\Delta\zeta$ is approximately the volume of the cell, and the speed of sound terms can be safely taken as

$$C(\Delta\eta\Delta\zeta + \Delta\xi\Delta\zeta + \Delta\xi\Delta\eta) = C|\mathbf{S}_{max}| \tag{A.31}$$

All of this can be combined in a somewhat *ad hoc* fashion with the viscous restriction to obtain

$$\Delta t \cong \lambda \frac{V}{|\mathbf{V}\cdot\mathbf{S}|_{max} + C|\mathbf{S}_{max}| + \frac{4\mu}{\Delta l_n}S_n} . \tag{A.32}$$

A.6 Semi-Implicit Time Step

The three dimensional semi-implicit stability limit can be derived from the two-dimensional explicit criterion. The convective and the viscous restrictions in the direction normal to a body are eliminated due to the implicit treatment of those fluxes.

Consider the 2–D criterion written in body tangential and cross stream coordinates

$$\begin{aligned}\Delta t &= \lambda \frac{1}{\frac{u_\xi + C}{\Delta\xi} + \frac{u_\zeta + C}{\Delta\zeta}} \\ &= \lambda \frac{\Delta\xi\Delta\zeta}{(u_\xi + C)\Delta\zeta + (u_\zeta + C)\Delta\xi} .\end{aligned} \tag{A.33}$$

where ξ corresponds to the streamwise direction and ς to the cross-stream direction. Multiplying the top and bottom by the cell's normal dimension $\Delta\eta$

$$\begin{aligned}\Delta t &= \lambda \frac{\Delta\xi\Delta\varsigma\Delta\eta}{(u_\xi + c)\Delta\varsigma\Delta\eta + (u_\varsigma + c)\Delta\xi\Delta\eta} \\ &= \lambda \frac{Vol}{u_\xi\Delta\varsigma\Delta\eta + u_\varsigma\Delta\xi\Delta\eta + C(\Delta\xi + \Delta\varsigma)\Delta\eta}\end{aligned} \tag{A.34}$$

and realizing that $u_\xi\Delta\varsigma\Delta\eta$ is just the flux of velocity through *Face* 1 or 2, while $u_\varsigma\Delta\xi\Delta\eta$ is the flux through *Face* 3 or 4:

$$\begin{aligned}u_\xi\Delta\varsigma\Delta\eta &= MAX(|\mathbf{V}\cdot\mathbf{S}_1|, |\mathbf{V}\cdot\mathbf{S}_2|) \\ u_\varsigma\Delta\xi\Delta\eta &= MAX(|\mathbf{V}\cdot\mathbf{S}_3|, |\mathbf{V}\cdot\mathbf{S}_4|)\end{aligned} \tag{A.35}$$

and

$$C(\Delta\xi + \Delta\varsigma)\Delta\eta) = C|\mathbf{S}_{max}|_{1,2,3,4} \tag{A.36}$$

Combining all these terms gives

$$\Delta t \cong 2\sqrt{2}\frac{Vol}{(|\mathbf{V}\cdot\mathbf{S}|)_{max_{1,2}} + (|\mathbf{V}\cdot\mathbf{S}|)_{max_{3,4}} + C|\mathbf{S}_{max_{1,2,3,4}}|}\,. \tag{A.37}$$

Appendix B

Implicit Formulation

This appendix describes in detail the system of equations that results from semi-implicit treatment of the governing equations and the efficient solution of that system. Each stage of *SINSS* has the form

$$\left[[I] + \alpha_s \frac{\Delta t}{V}\left([A]_5 S_{x_5} + [A]_6 S_{x_6} + [B]_5 S_{y_5} + [B]_6 S_{y_6} + [C]_5 S_{z_5} + [C]_6 S_{z_6}\right)^{s-1}\right] \Delta \mathbf{W}^s = -\alpha_s \frac{\Delta t}{V}\left[\sum_{f=1}^{6} \left(\mathbf{F} S_x + \mathbf{G} S_y + \mathbf{H} S_z\right)_f^{s-1} - \mathbf{D}^0\right] - \Delta \mathbf{W}^{s-1} . \tag{B.1}$$

This is a matrix equation for the changes $\Delta \mathbf{W}^s$ in the state vector from stage $s-1$ to stage s. The first two sections below present derivations of the terms on the implicit left hand side due to linearization of inviscid and viscous flux terms, respectively. The third and final section presents the variant of the Thomas algorithm that is used to invert the system.

B.1 Formulation of Left Hand Side for Inviscid Fluxes

The Jacobian matrices of the inviscid fluxes, $[A_I] = \partial \mathbf{F}_I / \partial W$, $[B_I] = \partial \mathbf{G}_I / \partial W$, and $[C_I] = \partial \mathbf{H}_I / \partial W$, must be defined and discretized. It is convenient to define character-

istic variables $U_1 \cdots U_5$

$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{pmatrix}, \tag{B.2}$$

so that the flux vectors $\mathbf{F}_I$, $\mathbf{G}_I$, $\mathbf{H}_I$ can be written as

$$\mathbf{F}_I = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \rho u(E + P/\rho) \end{pmatrix} = \begin{pmatrix} U_2 \\ \dfrac{U_2 U_2}{U_1} + P \\ \dfrac{U_2 U_3}{U_1} \\ \dfrac{U_2 U_4}{U_1} \\ \dfrac{U_2 U_5}{U_1} + \dfrac{U_2}{U_1} P \end{pmatrix}, \tag{B.3}$$

$$\mathbf{G}_I = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ \rho v(E + P/\rho) \end{pmatrix} = \begin{pmatrix} U_3 \\ \dfrac{U_2 U_3}{U_1} \\ \dfrac{U_3 U_3}{U_1} + P \\ \dfrac{U_3 U_4}{U_1} \\ \dfrac{U_3 U_5}{U_1} + \dfrac{U_3}{U_1} P \end{pmatrix}, \tag{B.4}$$

and

$$\mathbf{H}_I = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ \rho w(E + P/\rho) \end{pmatrix} = \begin{pmatrix} U_4 \\ \dfrac{U_2 U_4}{U_1} \\ \dfrac{U_3 U_4}{U_1} \\ \dfrac{U_4 U_4}{U_1} + P \\ \dfrac{U_4 U_5}{U_1} + \dfrac{U_4}{U_1} P \end{pmatrix}. \tag{B.5}$$

The pressure in terms of the characteristic variables (B.2) is

$$P = (\gamma - 1) \left(U_5 - \frac{1}{2} \frac{U_2^2 + U_3^2 + U_4^2}{U_1} \right). \tag{B.6}$$

The Jacobians $\partial \mathbf{F}_I/\partial \mathbf{W}$, $\partial \mathbf{G}_I/\partial \mathbf{W}$, and $\partial \mathbf{H}_I/\partial \mathbf{W}$, are found by performing the indicated differentiation,

$$[A_I] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -u^2 + \frac{\partial P}{\partial U_1} & 2u + \frac{\partial P}{\partial U_2} & \frac{\partial P}{\partial U_3} & \frac{\partial P}{\partial U_4} & \frac{\partial P}{\partial U_5} \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ -uH + u\frac{\partial P}{\partial U_1} & H + u\frac{\partial P}{\partial U_2} & u\frac{\partial P}{\partial U_3} & u\frac{\partial P}{\partial U_4} & u(1 + \frac{\partial P}{\partial U_5}) \end{bmatrix} \tag{B.7}$$

$$[B_I] = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -uv & v & u & 0 & 0 \\ -v^2 + \frac{\partial P}{\partial U_1} & \frac{\partial P}{\partial U_2} & 2v + \frac{\partial P}{\partial U_3} & \frac{\partial P}{\partial U_4} & \frac{\partial P}{\partial U_5} \\ -vw & 0 & w & v & 0 \\ -vH + v\frac{\partial P}{\partial U_1} & v\frac{\partial P}{\partial U_2} & H + v\frac{\partial P}{\partial U_3} & v\frac{\partial P}{\partial U_4} & v(1 + \frac{\partial P}{\partial U_5}) \end{bmatrix} \tag{B.8}$$

$$[C_I] = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ -w^2 + \frac{\partial P}{\partial U_1} & \frac{\partial P}{\partial U_2} & \frac{\partial P}{\partial U_3} & 2w + \frac{\partial P}{\partial U_4} & \frac{\partial P}{\partial U_5} \\ -wH + w\frac{\partial P}{\partial U_1} & w\frac{\partial P}{\partial U_2} & w\frac{\partial P}{\partial U_3} & H + w\frac{\partial P}{\partial U_4} & w(1 + \frac{\partial P}{\partial U_5}) \end{bmatrix} \tag{B.9}$$

For clarity, partial derivatives with respect to P were not expanded in (B.7) - (B.9). They are

$$\begin{aligned} \frac{\partial P}{\partial U_1} &= \frac{\gamma - 1}{2}(u^2 + v^2 + w^2) & \frac{\partial P}{\partial U_2} &= -(\gamma - 1)u \\ \frac{\partial P}{\partial U_3} &= -(\gamma - 1)v & \frac{\partial P}{\partial U_4} &= -(\gamma - 1)w \\ \frac{\partial P}{\partial U_5} &= (\gamma - 1) \end{aligned} \tag{B.10}$$

In a computer code it is convenient to define these terms in similar fashion to minimize coding complexity and computational cost.

To examine the structure of Equation B.1 ignoring, for now, linearization of the boundary points $j = 1$ and $j = JM$, expand (B.1) for each control volume (i, j, k)

$$[LHS_I] = \frac{\alpha\Delta t}{V}\{ \begin{array}{l} \frac{[A_I]_{j+1}+[A_I]_j}{2}S_{x_5} + \frac{[A_I]_j+[A_I]_{j-1}}{2}S_{x_6} \\ \frac{[B_I]_{j+1}+[B_I]_j}{2}S_{x_5} + \frac{[B_I]_j+[B_I]_{j-1}}{2}S_{x_6} \\ \frac{[C_I]_{j+1}+[C_I]_j}{2}S_{x_5} + \frac{[C_I]_j+[C_I]_{j-1}}{2}S_{x_6} \}+[I] \end{array}, \tag{B.11}$$

where the i and k indices and temporal superscript s have been suppressed for clarity. Rearranging slightly gives

$$[LHS_I] = \frac{\alpha\Delta t}{2V}\{ \begin{array}{l} [A_I]_{j+1}S_{x_5} + [A_I]_j(S_{x_5}+S_{x_6}) + [A_I]_{j-1}S_{x_6} \\ [B_I]_{j+1}S_{y_5} + [B_I]_j(S_{y_5}+S_{y_6}) + [B_I]_{j-1}S_{y_6} \\ [C_I]_{j+1}S_{z_5} + [C_I]_j(S_{z_5}+S_{z_6}) + [C_I]_{j-1}S_{z_6} \}+[I] \end{array} \tag{B.12}$$

Equations B.11 and B.12 are simply the implicit analog of the finite volume flux integration over *Faces* 5 and 6 found on the right hand side of Equation B.1.

Writing (B.12) at all cells and ordering in the index j gives a block tridiagonal matrix of equations:

$$\begin{bmatrix} [M_I]_1 & [M_I^+]_1 & & 0 & 0 & 0 \\ [M_I^-]_2 & [M_I]_2 & [M_I^+]_2 & & 0 & 0 \\ & \cdots & \cdots & \cdots & & 0 \\ & & [M_I^-]_j & [M_I]_j & [M_I^+]_j & \\ 0 & & & \cdots & \cdots & \cdots \\ 0 & 0 & & & \cdots & \cdots \\ 0 & 0 & 0 & & [M_I^-]_{JM} & [M_I]_{JM} \end{bmatrix} \times \begin{bmatrix} \Delta\mathbf{W}_1 \\ \Delta\mathbf{W}_2 \\ .. \\ \Delta\mathbf{W}_j \\ .. \\ .. \\ \Delta\mathbf{W}_{JM} \end{bmatrix} = \begin{bmatrix} \mathbf{Res}_1 \\ \mathbf{Res}_2 \\ .. \\ \mathbf{Res}_j \\ .. \\ .. \\ \mathbf{Res}_{JM} \end{bmatrix} \tag{B.13}$$

Each vector **Res** contains the residual of the five conservation equations in a cell (i,j,k) given by the right hand side of Equation B.1. The vectors $\Delta\mathbf{W}$ are the unknown changes in the state vector; and, $[M_I]$ are 5×5 matrices composed of the matrices $[A_I]$, $[B_I]$, and $[C_I]$ and the identity matrix $[I]$:

$$
\begin{aligned}
[M_I^+]_j = &\ \frac{\alpha\Delta t}{2V}\left[[A_I]_{j+1}S_{x_5} + [B_I]_{j+1}S_{y_5} + [C_I]_{j+1}S_{z_5}\right] && 1 \le j < JM \\
[M_I]_j = &\ [I] + \frac{\alpha\Delta t}{2V}\left[[A_I]_j(S_{x_5}+S_{x_6}) + [B_I]_j(S_{y_5}+S_{y_6}) + [C_I]_j(S_{z_5}+S_{z_6})\right] && 1 < j < JM \\
[M_I^-]_j = &\ \frac{\alpha\Delta t}{2V}\left[[A_I]_{j-1}S_{x_6} + [B_I]_{j-1}S_{y_6} + [C_I]_{j-1}S_{z_6}\right] && 1 < j \le JM
\end{aligned}
\tag{B.14}
$$

B.1.1 Boundary Conditions for the Inviscid Terms

At boundary cells ($j = 1$, $j = JM$), boundary conditions affect the definitions in (B.14). If a time-accurate solution is sought it is important to linearize these boundary fluxes accurately. If only a steady-state solution is required, an exact boundary condition treatment is necessary only if its absence causes a decay in the convergence rate. Below are described implicit treatment of boundary conditions at a wall and at a supersonic outer boundary. Other boundary conditions can be treated by writing out the appropriate discrete terms at the boundary and linearizing. In some cases not linearizing the boundary conditions does not appear to decrease the convergence rate of the calculation.

While the implementation of implicit boundary conditions in general can be complicated, implicit treatment of a wall boundary condition is straightforward. At a walls only the pressure flux is retained in the Euler equations, all other fluxes are set to zero. Moreover the pressure at the wall is calculated by simple extrapolation, $P_0 = P_1$ if the wall is at $j = 0$. Thus, $[M_I]_1$ contains all terms corresponding to fluxes through *Face* 5

but only the pressure fluxes across *Face* 6

$$[M_I]_1 = \frac{\alpha\Delta t}{2V}[[A_I]_1 S_{x_6} + [B_I]_1 S_{y_6} + [C_I]_1 S_{z_6} + 2([A_p]_1 S_{x_6} + [B_p]_1 S_{y_6} + [C_p]_1 S_{z_6})] + [I] \ . \tag{B.15}$$

The matrices $[A_p]$, $[B_p]$, and $[C_p]$ are equivalent to $[A_I]$, $[B_I]$, and $[C_I]$ with all terms except the pressure terms in the x, y, and z momentum equations, respectively, deleted.

Linearization of supersonic boundary conditions is even simpler since the values of the variables at the boundary are constant in time. Therefore, no additional terms are added to $[M_I]_{JM}$.

B.2 Linearization of Viscous Fluxes

The linearization process described in the previous section for the inviscid flux terms must also be applied to the viscous fluxes,

$$\mathbf{F}_V = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ F_{V_6} \end{pmatrix}, \quad \mathbf{G}_V = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ G_{V_6} \end{pmatrix}, \quad \text{and} \quad \mathbf{H}_V = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ H_{V_6} \end{pmatrix}, \tag{B.16}$$

where

$$\begin{aligned} F_{V_6} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \\ G_{V_6} &= u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \\ H_{V_6} &= u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \ . \end{aligned} \tag{B.17}$$

As before, the fluxes are written in terms of the characteristic variables (B.2) and

the Jacobians can be easily calculated. A cumulative "Jacobian" is given by

$$[M_V]_j = \frac{\partial}{\partial \mathbf{W}}(\mathbf{F}_V S_x + \mathbf{G}_V S_y + \mathbf{H}_V S_z)$$

$$=$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
\frac{\partial \tau_{xx}}{\partial(U1)}S_x + \frac{\partial \tau_{xy}}{\partial(U1)}S_y + \frac{\partial \tau_{xz}}{\partial(U1)}S_z & \frac{\partial \tau_{xx}}{\partial(U2)}S_x + \frac{\partial \tau_{xy}}{\partial(U2)}S_y + \frac{\partial \tau_{xz}}{\partial(U2)}S_z & \frac{\partial \tau_{xx}}{\partial(U3)}S_x + \frac{\partial \tau_{xy}}{\partial(U3)}S_y & \frac{\partial \tau_{xx}}{\partial(U4)}S_x + \frac{\partial \tau_{xz}}{\partial(U4)}S_z & 0 \\
\frac{\partial \tau_{xy}}{\partial(U1)}S_x + \frac{\partial \tau_{yy}}{\partial(U1)}S_y + \frac{\partial \tau_{yz}}{\partial(U1)}S_z & \frac{\partial \tau_{xy}}{\partial(U2)}S_x + \frac{\partial \tau_{yy}}{\partial(U2)}S_y & \frac{\partial \tau_{xy}}{\partial(U3)}S_x + \frac{\partial \tau_{yy}}{\partial(U3)}S_y + \frac{\partial \tau_{yz}}{\partial(U3)}S_z & \frac{\partial \tau_{yy}}{\partial(U4)}S_y + \frac{\partial \tau_{yz}}{\partial(U4)}S_z & 0 \\
\frac{\partial \tau_{xz}}{\partial(U1)}S_x + \frac{\partial \tau_{yz}}{\partial(U1)}S_y + \frac{\partial \tau_{xz}}{\partial(U1)}S_z & \frac{\partial \tau_{xz}}{\partial(U2)}S_x + \frac{\partial \tau_{xz}}{\partial(U2)}S_z & \frac{\partial \tau_{yz}}{\partial(U3)}S_y + \frac{\partial \tau_{xz}}{\partial(U3)}S_z & \frac{\partial \tau_{xz}}{\partial(U4)}S_x + \frac{\partial \tau_{yz}}{\partial(U4)}S_y + \frac{\partial \tau_{xz}}{\partial(U4)}S_z & 0 \\
V_{5,1} & V_{5,2} & V_{5,3} & V_{5,4} & V_{5,5}
\end{bmatrix}_j \tag{B.18}$$

where

$$\begin{aligned}
V_{5,1} = \; & (\frac{\partial \tau_{xx}}{\partial(U1)}u_{ave} + \frac{\partial \tau_{xy}}{\partial(U1)}v_{ave} + \frac{\partial \tau_{xz}}{\partial(U1)}w_{ave} - \Lambda\frac{\partial q}{\partial U1}\eta_x)S_x + \\
& (\frac{\partial \tau_{xy}}{\partial(U1)}u_{ave} + \frac{\partial \tau_{yy}}{\partial(U1)}v_{ave} + \frac{\partial \tau_{yz}}{\partial(U1)}w_{ave} - \Lambda\frac{\partial q}{\partial U1}\eta_y)S_y + \\
& (\frac{\partial \tau_{xz}}{\partial(U1)}u_{ave} + \frac{\partial \tau_{yz}}{\partial(U1)}v_{ave} + \frac{\partial \tau_{xz}}{\partial(U1)}w_{ave} - \Lambda\frac{\partial q}{\partial U1}\eta_z)S_z \;,
\end{aligned} \tag{B.19}$$

$$\begin{aligned}
V_{5,2} = \; & \frac{\partial \tau_{xx}}{\partial(U2)}u_{ave} + \frac{\partial \tau_{xy}}{\partial(U2)}v_{ave} + \frac{\partial \tau_{xz}}{\partial(U2)}w_{ave} - \Lambda\frac{\partial q}{\partial U2}\eta_x)S_x + \\
& (\frac{\partial \tau_{xy}}{\partial(U2)}u_{ave} + \frac{\partial \tau_{yy}}{\partial(U2)}v_{ave} - \Lambda\frac{\partial q}{\partial U2}\eta_y)S_y + \\
& (\frac{\partial \tau_{xz}}{\partial(U2)}u_{ave} + \frac{\partial \tau_{xz}}{\partial(U2)}w_{ave} - \Lambda\frac{\partial q}{\partial U2}\eta_z)S_z \;,
\end{aligned} \tag{B.20}$$

$$\begin{aligned}
V_{5,3} = \; & (\frac{\partial \tau_{xz}}{\partial(U3)}u_{ave} + \frac{\partial \tau_{xy}}{\partial(U3)}v_{ave} - \Lambda\frac{\partial q}{\partial U3}\eta_x)S_x + \\
& (\frac{\partial \tau_{xy}}{\partial(U3)}u_{ave} + \frac{\partial \tau_{yy}}{\partial(U3)}v_{ave} + \frac{\partial \tau_{yz}}{\partial(U3)}w_{ave} - \Lambda\frac{\partial q}{\partial U3}\eta_y)S_y + \\
& (\frac{\partial \tau_{yz}}{\partial(U3)}v_{ave} + \frac{\partial \tau_{xz}}{\partial(U3)}w_{ave} - \Lambda\frac{\partial q}{\partial U3}\eta_z)S_z \;,
\end{aligned} \tag{B.21}$$

$$\begin{aligned}
V_{5,4} = \; & (\frac{\partial \tau_{xz}}{\partial(U4)}u_{ave} + \frac{\partial \tau_{xz}}{\partial(U4)}w_{ave} - \Lambda\frac{\partial q}{\partial U4}\eta_x)S_x + \\
& (\frac{\partial \tau_{yy}}{\partial(U4)}v_{ave} + \frac{\partial \tau_{yz}}{\partial(U4)}w_{ave} - \Lambda\frac{\partial q}{\partial U4}\eta_y)S_y + \\
& (\frac{\partial \tau_{xz}}{\partial(U4)}u_{ave} + \frac{\partial \tau_{yz}}{\partial(U4)}v_{ave} + \frac{\partial \tau_{xz}}{\partial(U4)}w_{ave} - \Lambda\frac{\partial q}{\partial U4}\eta_z)S_z \;,
\end{aligned} \tag{B.22}$$

and

$$V_{5,5} = \quad -\Lambda\frac{\partial q}{\partial U5}\eta_x S_x - \Lambda\frac{\partial q}{\partial U5}\eta_y S_y - \Lambda\frac{\partial q}{\partial U5}\eta_z S_z \tag{B.23}$$

The heat flux terms in (B.19 – B.23) have been premultiplied by

$$\Lambda = (\mu_{l_{i,j,k}}/Pr + \mu_{t_{i,j,k}}/Pr_t)/(\mu_{l_{i,j,k}} + \mu_{t_{i,j,k}}) \tag{B.24}$$

to allow eventual multiplication of B.18 by μ.

The derivatives of the stress and heat flux terms are defined as

$$\begin{aligned}
\frac{\partial \tau_{xx}}{\partial(U1)} &= \frac{2}{3}(2\frac{\partial(\Delta u)}{\partial(U1)}\eta_x - \frac{\partial(\Delta v)}{\partial(U1)}\eta_y - \frac{\partial(\Delta w)}{\partial(U1)}\eta_z) & \frac{\partial \tau_{xx}}{\partial(U2)} &= \frac{4}{3}\frac{\partial(\Delta u)}{\partial(U2)}\eta_x \\
\frac{\partial \tau_{xx}}{\partial(U3)} &= -\frac{2}{3}\frac{\partial(\Delta v)}{\partial(U3)}\eta_y & \frac{\partial \tau_{xx}}{\partial(U4)} &= -\frac{2}{3}\frac{\partial(\Delta w)}{\partial(U4)}\eta_z \\
\frac{\partial \tau_{yy}}{\partial(U1)} &= \frac{2}{3}(2\frac{\partial(\Delta v)}{\partial(U1)}\eta_y - \frac{\partial(\Delta u)}{\partial(U1)}\eta_x - \frac{\partial(\Delta w)}{\partial(U1)}\eta_z) & \frac{\partial \tau_{yy}}{\partial(U2)} &= -\frac{2}{3}\frac{\partial(\Delta u)}{\partial(U2)}\eta_x \\
\frac{\partial \tau_{yy}}{\partial(U3)} &= \frac{4}{3}\frac{\partial(\Delta v)}{\partial(U3)}\eta_y & \frac{\partial \tau_{yy}}{\partial(U4)} &= -\frac{2}{3}\frac{\partial(\Delta w)}{\partial(U4)}\eta_z \\
\frac{\partial \tau_{zz}}{\partial(U1)} &= \frac{2}{3}(2\frac{\partial(\Delta w)}{\partial(U1)}\eta_z - \frac{\partial(\Delta v)}{\partial(U1)}\eta_y - \frac{\partial(\Delta u)}{\partial(U1)}\eta_x) & \frac{\partial \tau_{zz}}{\partial(U2)} &= -\frac{2}{3}\frac{\partial(\Delta u)}{\partial(U2)}\eta_x \\
\frac{\partial \tau_{zz}}{\partial(U3)} &= -\frac{2}{3}\frac{\partial(\Delta v)}{\partial(U3)}\eta_y & \frac{\partial \tau_{zz}}{\partial(U4)} &= \frac{4}{3}\frac{\partial(\Delta w)}{\partial(U4)}\eta_z\,,
\end{aligned} \tag{B.25}$$

$$\begin{aligned}
\frac{\partial \tau_{xy}}{\partial(U1)} &= \frac{\partial(\Delta u)}{\partial(U1)}\eta_y + \frac{\partial(\Delta v)}{\partial(U1)}\eta_x & \frac{\partial \tau_{xy}}{\partial(U2)} &= \frac{\partial(\Delta u)}{\partial(U2)}\eta_y & \frac{\partial \tau_{xy}}{\partial(U3)} &= \frac{\partial(\Delta v)}{\partial(U3)}\eta_x \\
\frac{\partial \tau_{xz}}{\partial(U1)} &= \frac{\partial(\Delta u)}{\partial(U1)}\eta_z + \frac{\partial(\Delta w)}{\partial(U1)}\eta_x & \frac{\partial \tau_{xz}}{\partial(U2)} &= \frac{\partial(\Delta u)}{\partial(U2)}\eta_z & \frac{\partial \tau_{xz}}{\partial(U4)} &= \frac{\partial(\Delta w)}{\partial(U4)}\eta_x \\
\frac{\partial \tau_{yz}}{\partial(U1)} &= \frac{\partial(\Delta v)}{\partial(U1)}\eta_z + \frac{\partial(\Delta w)}{\partial(U1)}\eta_y & \frac{\partial \tau_{yz}}{\partial(U3)} &= \frac{\partial(\Delta v)}{\partial(U3)}\eta_z & \frac{\partial \tau_{yz}}{\partial(U4)} &= \frac{\partial(\Delta w)}{\partial(U4)}\eta_y\,,
\end{aligned} \tag{B.26}$$

and

$$\begin{aligned}
\frac{\partial q}{\partial U1} &= \frac{\partial H}{\partial(U1)_{k,j}} - u_{ave}\frac{\partial(\Delta u)}{\partial(U1)} - v_{ave}\frac{\partial(\Delta v)}{\partial(U1)} - w_{ave}\frac{\partial(\Delta w)}{\partial(U1)} \\
\frac{\partial q}{\partial U2} &= \frac{\partial H}{\partial(U2)_{k,j}} - u_{ave}\frac{\partial(\Delta u)}{\partial(U2)} \\
\frac{\partial q}{\partial U3} &= \frac{\partial H}{\partial(U3)_{k,j}} - v_{ave}\frac{\partial(\Delta v)}{\partial(U3)} \\
\frac{\partial q}{\partial U4} &= \frac{\partial H}{\partial(U4)_{k,j}} - w_{ave}\frac{\partial(\Delta w)}{\partial(U4)} \\
\frac{\partial q}{\partial U5} &= \frac{\partial H}{\partial(U5)_{k,j}}\,.
\end{aligned} \tag{B.27}$$

Derivatives of the total enthalpy are

$$\begin{aligned}
\frac{\partial H}{\partial(U1)} &= \frac{1}{\rho}\left(\frac{\partial P}{\partial(U1)} - H\right) & \frac{\partial H}{\partial(U2)} &= \frac{1}{\rho}\frac{\partial P}{\partial(U2)} \\
\frac{\partial H}{\partial(U3)} &= \frac{1}{\rho}\frac{\partial P}{\partial(U3)} & \frac{\partial H}{\partial(U4)} &= \frac{1}{\rho}\frac{\partial P}{\partial(U4)} \\
\frac{\partial H}{\partial(U5)} &= \frac{1}{\rho}\left(\frac{\partial P}{\partial(U5)} + 1\right) .
\end{aligned} \tag{B.28}$$

The definitions above utilize the fact that $\Delta\eta$ may be set to unity arbitrarily, such that

$$\Delta u \Leftrightarrow \frac{\partial u}{\partial \eta} , \tag{B.29}$$

for example, and, therefore,

$$\begin{aligned}
\frac{\partial(\Delta u)}{\partial(U1)} &= -\frac{U_2}{\rho^2} & \frac{\partial(\Delta v)}{\partial(U1)} &= -\frac{U_3}{\rho^2} & \frac{\partial(\Delta w)}{\partial(U1)} &= -\frac{U_4}{\rho^2} \\
\frac{\partial(\Delta u)}{\partial(U2)} &= \frac{1}{\rho} & \frac{\partial(\Delta v)}{\partial(U3)} &= \frac{1}{\rho} & \frac{\partial(\Delta w)}{\partial(U4)} &= \frac{1}{\rho} .
\end{aligned} \tag{B.30}$$

The metric terms are defined as

$$\begin{aligned}
S_x &= .5(S_{x_{2,j}} + S_{x_{2,j-1}}) & \eta_x &= .5(\eta_{x_j} + \eta_{x_{j-1}}) \\
S_y &= .5(S_{y_{2,j}} + S_{y_{2,j-1}}) & \eta_y &= .5(\eta_{y_j} + \eta_{y_{j-1}}) \\
S_z &= .5(S_{z_{2,j}} + S_{z_{2,j-1}}) & \eta_z &= .5(\eta_{z_j} + \eta_{z_{j-1}}) ,
\end{aligned} \tag{B.31}$$

where i and k indices have been omitted for clarity.

The definition of the viscous contribution to $[LHS]$ is completed by setting $[M] = [M_I] + [M_V]$ in (B.13) where

$$\begin{aligned}
[M_V]_j^+ &= -\alpha\Delta t_j [M_V]_{j+1} \\
[M_V]_j &= 2\alpha\Delta t_j [M_V]_j \\
[M_V]_j^- &= -\alpha\Delta t_j [M_V]_{j-1} .
\end{aligned} \tag{B.32}$$

Several simplifying assumptions were made to obtain the linearizations that lead to (B.32). Implicit in the definitions above is the assumption that, for a discrete quantity Λ,

$$\frac{\partial \Lambda}{\partial \mathbf{W}_{j+1}} = \pm \left.\frac{\partial \Lambda}{\partial \mathbf{W}_j}\right|_{j+1} \quad \text{and} \quad \frac{\partial \Lambda}{\partial \mathbf{W}_{j-1}} = \pm \left.\frac{\partial \Lambda}{\partial \mathbf{W}_j}\right|_{j-1} \tag{B.33}$$

This is true for stress terms with constant coefficients such as $\frac{\partial u}{\partial \eta}$ or $\frac{\partial v}{\partial \eta}$, but not for most terms that are the product of a sum and a difference such as $v_{ave}\frac{\partial u}{\partial \eta}$. To handle these terms with the above treatment one must assume that v does not vary from $j-1$ to $j+1$. The difference in the linearization is slight for most terms and numerical experiments with the 2D scheme did not show appreciable differences in convergence. In addition, the temporal variation of viscosity with the state vector was assumed to be negligible, an assumption that usually holds in laminar flow, and the metrics defined in (B.31) are an approximation to those on the right hand side of Equation B.1.

B.3 Solution of Block Tridiagonal System

For a computational grid with JM cells in the normal direction (B.13) is a matrix of size $5JM \times 5JM$. Each unknown is the change of some quantity, $\delta(\rho U)$ for example, over one stage in a cell. Due to the compact nature of the discretization, each unknown depends only on the unknowns in its own and in the two neighboring cells at $(j-1)$ and $(j+1)$. Those unknowns, in turn depend on their immediate neighbors, thus establishing a domain of dependence from $j=1$ to $j=JM$. This gives the matrix a block tridiagonal structure, each block having dimension five, equal to the number of conservation equations. It is essential that the inversions be computationally efficient on computers with scalar or parallel architectures.

Block tridiagonal matrices can be inverted very efficiently. Their solution is equivalent to solution of a scalar tridiagonal matrix, with scalar operations replaced by matrix operations. A specialized form of Gaussian elimination known as the Thomas algorithm

can be employed. Form the recurrence relations

$$\begin{aligned} [D]_1 &= [M]_1 \\ [E]_j &= [D]_j^{-1}[M^+]_j \qquad 1 \le j \le J \\ [D]_j &= [M]_j - [M^-]_j[E]_{j-i} \qquad 2 \le j \le J. \end{aligned} \tag{B.34}$$

The 5×5 matrix equation $[E]_j = [D]_j^{-1}[M^+]_j$ can be solved by Gaussian elimination.

Applying the same operations to the right hand side gives

$$\begin{aligned} \mathbf{Z}_1 &= [D]_1^{-1}\mathbf{Res}_1 \\ \mathbf{Z}_j &= [D]_j^{-1}(\mathbf{Res}_j - [M^-]_j\mathbf{Z}_{j-i}) \qquad 2 \le j \le J \end{aligned} \tag{B.35}$$

These operations require little additional memory since $[E]_j$ can be stored in $[C_j]$ and $\mathbf{Z}_j$ in $\mathbf{Res}_j$.

Combining left and right hand sides gives the intermediate result

$$\begin{bmatrix} [I] & [E]_1 & & & \\ & [I] & [E]_2 & & \\ & \cdots & \cdots & \cdots & \\ & & & [I] & \end{bmatrix} \begin{bmatrix} \Delta\mathbf{W}_1 \\ \Delta\mathbf{W}_2 \\ .. \\ \Delta\mathbf{W}_{JM} \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \\ .. \\ \mathbf{Z}_J \end{bmatrix} \tag{B.36}$$

where $[I]$ is the identity matrix.

Finally, to eliminate the upper band of E matrices subtract $[E]_{j-1}\times$ row j from row $j-1$ to get

$$\begin{bmatrix} \Delta\mathbf{W}_1 \\ \Delta\mathbf{W}_2 \\ .. \\ \Delta\mathbf{W}_J \end{bmatrix} = \begin{bmatrix} \mathbf{ZZ}_1 \\ \mathbf{ZZ}_2 \\ .. \\ \mathbf{ZZ}_J \end{bmatrix} \tag{B.37}$$

where,

$$\begin{aligned} \mathbf{ZZ}_J &= \mathbf{Z}_J \\ \mathbf{ZZ}_{j-1} &= \mathbf{Z}_j - [E]_{j-1}\mathbf{ZZ}_j \qquad 2 \le j < J\ . \end{aligned} \tag{B.38}$$

B.3.1 CPU Requirements

While this algorithm is relatively efficient it is not well suited to vector processing because, as written above, it is recursive in the index j, and thus can not be vectorized. This problem can be circumvented by inverting (B.13) "plane by plane", rather than one (i,k) location at a time by taking either $K = 1, KM$ or $I = 1, IM$ as an inner loop in (B.34) – (B.38). With this approach the sections of computer code that perform the inversions can be completely vectorized. The price in memory required paid for this inversion is small if the Thomas algorithm is integrated into the Fortran routines that setup the matrix B.13 (see Appendix H).

Appendix C

Residual Smoothing Implementation

In three dimensions residual smoothing can be written as:

$$\begin{aligned}(1-\mu_R\delta_{\eta\eta})\mathbf{Res}' &= \mathbf{Res} \\ (1-\mu_R\delta_{\xi\xi})\mathbf{Res}'' &= \mathbf{Res}' \\ (1-\mu_R\delta_{\zeta\zeta})\mathbf{Res}''' &= \mathbf{Res}''\end{aligned} \tag{C.1}$$

The *LHS* of each equation is a scalar tridiagonal matrix. The difference operator $\delta_{\eta\eta}$, for example, is here taken to be $\delta_{\eta\eta}(\ ) = (\ )_{j+1} - 2(\ )_j + (\ )_{j-1}$. At grid boundaries a "ghost" value of the residual must be defined. Two possible boundary conditions are to set the ghost residual to zero or to the value of the first residual inside the domain. Both boundary conditions were tried and no significant difference in convergence acceleration was found. The first boundary condition (ghost residual equal to zero) was used since it leaves the diagonal of the implicit residual smoothing matrix constant:

$$\begin{bmatrix} 1+2\mu_R & -\mu_R & & 0 & 0 & 0 \\ -\mu_R & 1+2\mu_R & -\mu_R & & 0 & 0 \\ & \cdots & \cdots & \cdots & & 0 \\ & & -\mu_R & 1+2\mu_R & -\mu_R & \\ 0 & & & \cdots & \cdots & \cdots \\ 0 & 0 & & & \cdots & \cdots \\ 0 & 0 & 0 & & -\mu_R & 1+2\mu_R \end{bmatrix}_{i,k} \times \begin{bmatrix} \mathbf{Res}'_1 \\ \mathbf{Res}'_2 \\ .. \\ \mathbf{Res}'_j \\ .. \\ .. \\ \mathbf{Res}'_{JM} \end{bmatrix}_{i,k} = \begin{bmatrix} \mathbf{Res}_1 \\ \mathbf{Res}_2 \\ .. \\ \mathbf{Res}_j \\ .. \\ .. \\ \mathbf{Res}_{JM} \end{bmatrix}_{i,k} \tag{C.2}$$

This system can be inverted very efficiently. The solution algorithm is [Acton 70]

$$\begin{aligned} \mathbf{Res}'_N &= \frac{S_N}{\alpha_N} \\ \mathbf{Res}'_i &= \frac{S_i + \mu_R \mathbf{Res}'_{i+1}}{\alpha_i} \qquad 1 \le i \le N-1 \end{aligned} \tag{C.3}$$

where

$$\begin{aligned} \alpha_1 &= 1+2\mu_R & \alpha_i &= 1+2\mu_R - \tfrac{\mu_R^2}{\alpha_{i-1}} \\ S_1 &= \mathbf{Res}_1 & S_i &= \mathbf{Res}_i + \tfrac{\mu_R S_{i-1}}{\alpha_{i-1}} \end{aligned} \tag{C.4}$$

Since α_i are invariant, they must be calculated only once (for the largest dimension) in the entire simulation.

Appendix D

Accuracy of the Difference Operators

This appendix explores the accuracy of the difference operator used in Chapter 3 and of several other difference operators. Because accuracy of approximations to the full governing equations is difficult to assess, representative first derivative and second derivative terms on a one dimensional stretched grid are examined. The purpose of the analysis is to provide insight into the accuracy of popular discretizations on Navier-Stokes type grids, without having to consider complications due to factors such as artificial viscosity, solution convergence, grid skewness in multiple dimensions, etc. As such, this analysis gives an upper bound of attainable accuracy and results given should be interpreted cautiously.

This appendix examines first the accuracy of approximations to df/dx, corresponding to an inviscid term, and then to the "viscous" term d^2f/dx^2. In each case three methods are considered: the finite volume, finite difference, and box-type approaches. The one-dimensional test geometry is given in Figure D.1.1. Nodes points are denoted by bullets, cell centers by $\otimes$. The grid is defined by the coordinates of the nodes. Differences $(\Delta x, \Delta y)$ are calculated with a counterclockwise orientation. The magnitude Δy is taken as unity. Spacing in x is computed from $x_{i+1} = x_i + \Delta x_i$ where $\Delta x_{i+1} = a\Delta x_i$, and a is a stretching parameter defined by $a = 1 + \epsilon$. This formulation is commonly used in generating Navier-Stokes grids about simple geometries (see 5.2).

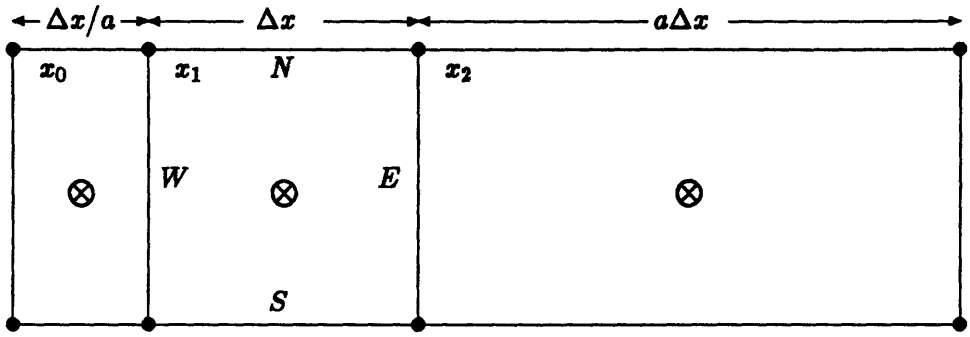

Figure D.1: One Dimensional Grid

D.1 Approximations to Inviscid Terms

D.1.1 The Standard Finite Volume Approach

The "standard" finite volume approach is a cell centered conservative discretization. In this formulation flow variables are stored at the centers of each cell. Fluxes through each of the cell sides are defined by some combination of the variables at nearby cell centers. The difference is formed via a line integration of the fluxes around the cell:

$$\begin{aligned} \frac{df}{dx} &= \frac{1}{Area_1}(F_E\Delta y_E + F_N\Delta y_N + F_W\Delta y_W + F_S\Delta y_S) \\ &= \frac{F_E - F_W}{\Delta x} \end{aligned} \tag{D.1}$$

since $\Delta y_E = 1 = -\Delta y_W$, $\Delta y_S = \Delta y_N = 0$, and the area of *Cell 1* is $\Delta y \times \Delta x = \Delta x$. Equation D.1 is written in finite difference form to allow comparison with formulae constructed with a finite difference approach. The discretization is conservative because the flux out of each cell flows entirely into the adjacent cell.

Discretizations differ by their definition of fluxes and accuracy of the resulting discretization. Two methods of defining the fluxes are "straight" averaging and "weighted" averaging.

Finite Volume Straight Averaging

In this widely-used formulation (used in this thesis and many others) the flux is approximated by a simple average of the variables at the adjacent cell centers:

$$\begin{aligned} F_E &\cong .5(f_2 + f_1) \\ F_W &\cong .5(f_1 + f_0) \ ; \end{aligned} \tag{D.2}$$

or, with [D.1]

$$\frac{df}{dx} \cong \frac{f_2 - f_0}{\Delta x} \ . \tag{D.3}$$

Finite Volume Weighed Averaging

Instead of using a simple average, this method interpolates linearly between the cell centers to get the fluxes F_E and F_W:

$$\begin{aligned} F_E &\cong \frac{f_2(\Delta x/2) + f_1(a\Delta x/2)}{\Delta x/2 + a\Delta x/2} \\ F_W &\cong \frac{f_0(\Delta x/2) + f_1(\Delta x/(2a))}{\Delta x/2 + \Delta x/(2a)} \ , \end{aligned} \tag{D.4}$$

giving

$$\frac{df}{dx} = \frac{f_2 + (a-1)f_1 - af_0}{(a+1)\Delta x} \ . \tag{D.5}$$

D.1.2 Box-Type Finite Volume Integration

In a box-type scheme the governing equations are written as a system of first order partial differential equations, with flow variables stored at the nodes of each cell. The system of equations is typically solved simultaneously via a Newton-type algorithm[1].

[1]See, for example, boundary layer solvers by [Keller 70], [Loyd & Murman 86], and the Euler solver of [Drela 83].

Although this approach has not (so far) proved practical for solution of the Navier-Stokes equations, it is included here because it gives very accurate results.

Fluxes in a box-type scheme are calculated by averaging variables at adjacent *node* points. This results in a second order accurate trapezoidal integration on arbitrary grids. In one dimension this corresponds to knowing F_E and F_W exactly in Equation D.1. Thus the box scheme is second order accurate regardless of the grid.

D.1.3 Finite Difference Approximations

Finite difference approximations are easily derived from Taylor series expansions. The resulting expressions also give immediate notice of their order of accuracy. Consider expansions about cell center 1:

$$\begin{aligned} f_2 &= f_1 + \Delta x^+ f_1' + \frac{\Delta x^{+2}}{2} f_1'' + \frac{\Delta x^{+3}}{6} f_1''' + \ldots \\ f_1 &= f_1 \\ f_0 &= f_1 - \Delta x^- f_1' + \frac{\Delta x^{-2}}{2} f_1'' - \frac{\Delta x^{-3}}{6} f_1''' + \ldots \end{aligned} \tag{D.6}$$

where the prime ($'$) denotes differentiation with respect to x and where $\Delta x^+ = \Delta x/2 + a\Delta x/2$ and $\Delta x^- = \Delta x/2 + \Delta x/(2a)$. Approximations to arbitrary order of accuracy can be attained by appropriate combinations of these and Taylor series at other adjacent points. Consider centered first and second order accurate discretizations to f_1'.

Finite Difference First Order Accurate

Only two points are necessary for a first order accurate discretization. Set $Af_2 + Bf_0 = f_1'$. Then by requiring that the coefficient of f_1 term be zero, and coefficient of f_1' term

be 1 one gets

$$\frac{df}{dx} = \frac{f_2 - f_0}{\Delta x} + O(\Delta x) \tag{D.7}$$

$(A = 1, B = -1)$ as expected. This operator is identical to the standard finite volume operator [D.3].

Finite Difference Second Order Accurate

Second order accuracy can be attained with knowledge of f_1, since, with a third available equation, one may also require the coefficient of the f_1'' terms to vanish. The three equations in three unknowns can be solved to give:

$$\frac{df}{dx} = Af_2 + Bf_1 + Cf_0 + O(\Delta x^2) \tag{D.8}$$

where

$$\begin{aligned} A &= \frac{\Delta x^-}{\Delta x^+(\Delta x^+ + \Delta x^-)} \\ C &= \frac{-\Delta x^+}{\Delta x^-(\Delta x^+ + \Delta x^-)} \\ B &= -(A + C) \end{aligned} \tag{D.9}$$

Since this solution is unique, no other combination of f_2, f_1, and f_0 can be second order accurate.[2] Thus the finite volume weighted averaging formulation (D.5) is not, in general, second order accurate. Conversely, the second order accurate finite difference form is not conservative.

[2] Of course, second order accurate up or downwind discretisations can be constructed which involve three different points. While these can also be made second order accurate, in general, the centered formulation has the smallest truncation error.

D.2 Approximations to Viscous Terms

Viscous terms present a more difficult discretization problem then inviscid terms because for the same order of accuracy (if it is at all obtainable) they generally require more points. The section begins with examination of the finite difference approach, which quickly illustrates the problem.

D.2.1 Finite Difference Viscous Derivatives

Finite Difference First Order Accurate

Set $Af_2+Bf_1+Cf_0 = f_1''$, where f_2, f_1, and f_0 are defined as the Taylor series expansions D.6. This equation implies three relations: the coefficients of coefficients of f and f' must vanish, and the coefficient of f'' must be equal one. Solving the three equations for A, B, and C gives:

$$\frac{d^2f}{dx^2} = Af_2 + Bf_1 + Cf_0 + O(\Delta x) \tag{D.10}$$

where

$$\begin{aligned} A &= \frac{2}{\Delta x^+(\Delta x^+ + \Delta x^-)} \\ C &= \frac{2}{\Delta x^-(\Delta x^+ + \Delta x^-)} \\ B &= -(A+C) \end{aligned} \tag{D.11}$$

Second order accuracy would require the satisfaction of a fourth constraint (the coefficient of f''' equals zero) which is impossible with a three point stencil.

A Simple Minded Approach

An unthinking approach is to take the approximation

$$\frac{df}{dx} = \frac{f_2 - 2f_1 + f_0}{\Delta x^2} \tag{D.12}$$

which is second order accurate on a *Cartesian* grid and apply it to the *stretched* grid. Insertion of the Taylor series D.6 into (D.12) shows immediately that this is an inconsistent approximation.

D.2.2 Finite Volume Viscous Derivative

A finite volume discretization for f'' consistent with the discretization of f' is to treat the second derivative as the difference of two first derivatives. This is the approach used in Chapter 3. Define

$$\begin{aligned} F'_E &\cong \frac{f_2 - f_1}{\Delta x^+} \\ F'_W &\cong \frac{f_1 - f_0}{\Delta x^-}, \end{aligned} \tag{D.13}$$

and then simply apply Equation D.1 to get in finite difference form

$$f'' = \frac{f_2 \Delta x^- - f_1(\Delta x^- + \Delta x^+) + f_0 \Delta x^+}{\Delta x \Delta x^+ \Delta x^-}. \tag{D.14}$$

Note that this form is equivalent to the first order accurate finite difference operator constructed previously only in the limit of $a = 1$ (ie. no stretching).

D.2.3 Box Scheme for Viscous Terms

In a box-type scheme the governing equations are written as a system of *first* order equations, so that the first difference terms are obtained from solution of the governing equations. A numerical first difference approximation only must be constructed to get

the second derivatives. Because of this, second derivatives obtained with a box scheme are necessarily second order accurate on arbitrary grids. The procedure and accuracy is identical to that of the box scheme for inviscid terms discussed above.

D.2.4 Numerical Study of Accuracy

The analyses above give some indication of the relative accuracy of a number of difference operators. It is of interest, however, to examine a numerical comparison of the operators to quantify the differences between them. For this study the function

$$f(x) = sin(\frac{\pi}{2}x) \tag{D.15}$$

was chosen. Equation D.15 is smooth, infinitely differentiable, and a good approximation to the velocity profile in a laminar boundary layer [Schlichting 68]. To compare the difference operators, we construct exponentially stretched grids from $x = 0$ to $x = 1$. The grids start with $NX = 5$ ($\Delta x = .2$) and are successively refined by a factor of 2. Different values of the stretching factor a were chosen to simulate stretching in a boundary layer type grid. The first and second differences were calculated analytically and with the above operators at each cell center. The root mean square of the error

$$RMS\ Error = \frac{1}{Jmax}\sqrt{\sum_{j=1}^{j=Jmax}\left(\frac{Exact - Approx}{Exact}\right)^2} \tag{D.16}$$

was used to measure the deviation from the exact solution. For a second order accurate scheme this quantity should decrease quadratically with x, for a first order scheme, linearly.

Table D.1 and D.2 show the effect of stretching and grid line density on the accuracy of the four discrete operators for inviscid terms. Table D.1 gives the accuracy on a *Cartesian* grid. The first column gives the magnitude of Δx. The second through fourth

columns give the RMS error of the standard *FV* or first order accurate *FD* approach, the weighted *FV* method, the second order accurate *FD* approach and the trapezoidal integration.

Table D.1: First Derivative Accuracy on Cartesian Grid ($a = 1.0$)

Δx	FV/FD 1	FV Mod.	FD 2^{nd} O.	Box Sch.
0.2	0.007320	0.007320	0.007320	0.001837
0.1	0.001299	0.001299	0.001299	0.000325
0.05	0.000229	0.000229	0.000229	0.000058
0.025	0.000040	0.000040	0.000040	0.000010
0.0125	0.000009	0.000009	0.000009	0.000002

Because the grid is unstretched the first three schemes are identical and yield second order accuracy with a low truncation error. Even so their accuracy is surpassed by the trapezoidal scheme. In this idealized survey almost twice the number of grid points are required by the non-trapezoidal schemes to duplicate its accuracy.

This disparity increases on a stretched grid. Table D.2 gives results computed on a grid with stretching factor of 1.2. A factor of this magnitude is often used to construct grids for Navier-Stokes and boundary layer equation simulations (see (5.2)). Accuracy of the *FV*/*FD*1 (first order accurate finite volume and finite difference) schemes decreases significantly. The truncation error is much larger, especially for the standard scheme and the order of accuracy has reverted to first order for both schemes. The second order *FD* scheme and the trapezoidal scheme are unaffected by the stretching.

Tables D.3 and D.4 give the RMS errors for approximations to the second derivative

Table D.2: First Derivative Accuracy on Stretched Grid ($a = 1.2$)

Δx	FV/FD 1	FV Mod.	FD 2^{nd} O.	Box Sch.
0.2	0.041028	0.024457	0.00717	0.001837
0.1	0.019239	0.010656	0.001287	0.000325
0.05	0.009481	0.005126	0.000229	0.000058
0.025	0.004763	0.002541	0.000041	0.000010
0.0125	0.002438	0.001265	0.000007	0.000002

of (D.15). Table D.3 gives results for an unstretched grid where the schemes uniformly exhibit second order accuracy and a small truncation error. Table D.4 gives results on

Table D.3: Second Derivative Accuracy on Cartesian Grid ($a = 1.0$)

Δx	Simple FD	FD 1^{st} O.	FV Stan.	Box Sch.
0.2	0.003666	0.003666	0.003666	0.001837
0.1	0.000650	0.000650	0.000650	0.000325
0.05	0.000115	0.000115	0.000115	0.000058
0.025	0.000021	0.000021	0.000021	0.000010
0.0125	0.000024	0.000032	0.000024	0.000002

a grid with stretching factor $A = 1.2$. Column 2 displays the expected characteristics of an inconsistant scheme: as Δx is decreased accuracy decreases. As expected the first order accurate *FD* scheme (Column 3) shows linear increase in accuracy as Δx is halved. The weighted *FV* scheme (which is not formally first order accurate) results are only

slightly less accurate than the *FD* results.

Table D.4: Second Derivative Accuracy on Stretched Grid ($a = 1.2$)

Δx	Simple FD	FD 1^{st} O.	FV Stan.	Box Sch.
0.2	.739131	0.023144	0.025629	0.001837
0.1	1.58466	0.012835	0.014489	0.000325
0.05	3.24706	0.006648	0.007713	0.000058
0.025	6.554687	0.003369	0.004068	0.000010
0.0125	13.1607	0.001704	0.002177	0.000002

D.3 Accuracy in Generalized Coordinates

A popular approach to discretizing a set of equations is to first transform the equations to general curvilinear form by writing them in body normal and tangential coordinates (η, ξ). This is done by application of the chain rule. In one dimension $x \rightarrow \xi$ via:

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} = \xi_x \frac{\partial}{\partial \xi} \,. \tag{D.17}$$

Discretization of $\partial/\partial\xi$ now proceeds as before with $\partial/\partial x$. In addition, the metric term ξ_x must be defined. For consistency the metric term must be defined precisely where the difference is defined: in a centered difference about point i, ξ_x must also be computed at i. Comparison with (D.3) shows that this procedure results in a conservative discretization. Second differences should be defined as the difference of two first differences with ξ_x defined appropriately.

The accuracy of the first derivative written in generalized coordinate form depends on the accuracy with which both the metric and the actual difference terms are evaluated. Since the computational grid (ξ, η) may be defined as Cartesian, the centered approximations to the first and second derivatives are second order accurate. However, to maintain that order of accuracy, the metric term ξ_x must also be evaluated with second (or higher) order accuracy. This is usually possible only for analytically defined grids[3]. If the metrics are computed with the same difference formulae as the flow derivatives, the truncation error of the scheme will be identical to that of the scheme written in physical coordinates.

[3]These are notoriously scarce in two and three dimensions.

Appendix E

Calculation of Metrics

The metric coefficients η_x, η_y, η_z found in Equations 2.14 and 2.15 may be found by considering the differential expressions from the chain rule

$$\begin{aligned} d\xi &= \xi_x dx + \xi_y dy + \xi_z dz \\ d\eta &= \eta_x dx + \eta_y dy + \eta_z dz \\ d\varsigma &= \varsigma_x dx + \varsigma_y dy + \varsigma_z dz \ , \end{aligned} \tag{E.1}$$

that is

$$\begin{bmatrix} \Delta\xi \\ \Delta\eta \\ \Delta\varsigma \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \varsigma_x & \varsigma_y & \varsigma_z \end{bmatrix} \times \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} , \tag{E.2}$$

and similarly

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \begin{bmatrix} x_\xi & x_\eta & x_\varsigma \\ y_\xi & y_\eta & y_\varsigma \\ z_\xi & z_\eta & z_\varsigma \end{bmatrix} = \begin{bmatrix} \Delta\xi \\ \Delta\eta \\ \Delta\varsigma \end{bmatrix} . \tag{E.3}$$

Inverting (E.3) by Cramer's rule and equating with (E.2) gives

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \varsigma_x & \varsigma_y & \varsigma_z \end{bmatrix} = J \begin{bmatrix} y_\eta z_\varsigma - y_\varsigma z_\eta & -(x_\eta z_\varsigma - x_\varsigma z_\eta) & x_\eta y_\varsigma - x_\varsigma y_\eta \\ -(y_\xi z_\varsigma - y_\varsigma z_\xi) & x_\xi z_\varsigma - x_\varsigma z_\xi & -(x_\xi y_\varsigma - x_\varsigma y_\xi) \\ y_\xi z_\eta - y_\eta z_\xi & -(x_\xi z_\eta - x_\eta z_\xi) & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \tag{E.4}$$

where J is the Jacobian of the transformation

$$
\begin{aligned}
J &= \begin{vmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{vmatrix} \\
&= 1/\left[x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)\right]
\end{aligned}
\tag{E.5}
$$

and, from (E.2)

$$
\begin{aligned}
\eta_x &= -J(y_\xi z_\zeta - y_\zeta z_\xi) \\
\eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\
\eta_z &= -J(x_\xi y_\zeta - x_\zeta y_\xi)
\end{aligned}
\tag{E.6}
$$

In the Fortan code the derivatives with respect to Cartesian coordinates are needed at the center of each face. They are obtained by computing values at the center of each edge with centered differences and averaging to obtain the face centered values required in (2.14) and (2.15).

Appendix F

Wing Grid Definition

Below are coordinates of the streamwise and circumferential (normalized by half span) grid point locations of the medium and fine grids. The grid was generated using the Fortran code in Appendix G.

Location of streamwise stations:

Streamwise Station	X coordinate
i=1	0.1000000
2	0.1068000
3	0.1140625
4	0.1218188
5	0.1301025
6	0.1389496
7	0.1483982
8	0.1584893
9	0.1692667
10	0.1807769
11	0.1930698
12	0.2061986
13	0.2202202
14	0.2351953
15	0.2511887
16	0.2682696
17	0.2865120
18	0.3059950
19	0.3268028
20	0.3490255
21	0.3727594
22	0.3981072
23	0.4251787
24	0.4540910
25	0.4849694
26	0.5179476
27	0.5531682
28	0.5907840
29	0.6309575
30	0.6738629

31	0.7196859
32	0.7686249
33	0.8208917
34	0.8767127
35	0.9363296
36	1.000000

Medium grid circumferential locations:

Circumferential loc.	Y Coordinate	Z Coordinate
1	0.0000000E+00	-5.0000504E-02
2	4.9067654E-02	-4.9940266E-02
3	9.8017134E-02	-4.9759798E-02
4	0.1467305	-4.9459293E-02
5	0.1950902	-4.9039796E-02
6	0.2429802	-4.8502065E-02
7	0.2902846	-4.7847454E-02
8	0.3368899	-4.7077641E-02
9	0.3826834	-4.6194427E-02
10	0.4275551	-4.5199927E-02
11	0.4713967	-4.4096500E-02
12	0.5141027	-4.2886861E-02
13	0.5555702	-4.1573882E-02
14	0.5956993	-4.0160805E-02
15	0.6343933	-3.8650859E-02
16	0.6715589	-3.7047926E-02
17	0.7071068	-3.5355635E-02
18	0.7409512	-3.3578303E-02
19	0.7730104	-3.1719983E-02
20	0.8032075	-2.9785288E-02
21	0.8314696	-2.7778795E-02
22	0.8577287	-2.5705412E-02
23	0.8819213	-2.3570077E-02
24	0.9039893	-2.1377981E-02
25	0.9238795	-1.9134356E-02
26	0.9415441	-1.6844654E-02
27	0.9569404	-1.4514369E-02
28	0.9700313	-1.2149133E-02
29	0.9807853	-9.7546075E-03
30	0.9891766	-7.3365844E-03
31	0.9951847	-4.9009109E-03
32	0.9987954	-2.4534089E-03
33	1.000000	0.0000000E+00
34	0.9987954	2.4534089E-03
35	0.9951847	4.9009109E-03
36	0.9891766	7.3365844E-03
37	0.9807853	9.7546075E-03
38	0.9700313	1.2149133E-02
39	0.9569404	1.4514369E-02
40	0.9415441	1.6844654E-02

41	0.9238795	1.9134356E-02
42	0.9039893	2.1377981E-02
43	0.8819213	2.3570077E-02
44	0.8577287	2.5705412E-02
45	0.8314696	2.7778795E-02
46	0.8032075	2.9785288E-02
47	0.7730104	3.1719983E-02
48	0.7409512	3.3578303E-02
49	0.7071068	3.5355635E-02
50	0.6715589	3.7047926E-02
51	0.6343933	3.8650859E-02
52	0.5956993	4.0160805E-02
53	0.5555702	4.1573882E-02
54	0.5141027	4.2886861E-02
55	0.4713967	4.4096500E-02
56	0.4275551	4.5199927E-02
57	0.3826834	4.6194427E-02
58	0.3368899	4.7077641E-02
59	0.2902846	4.7847454E-02
60	0.2429802	4.8502065E-02
61	0.1950902	4.9039796E-02
62	0.1467305	4.9459293E-02
63	9.8017134E-02	4.9759798E-02
64	4.9067654E-02	4.9940266E-02
65	0.0000000E+00	5.0000504E-02

Fine grid circumferential locations:

Circumferential loc.	Y Coordinate	Z Coordinate
k= 1	0.0000000E+00	-5.0000504E-02
2	0.1221366	-4.9626183E-02
3	0.2336354	-4.8616704E-02
4	0.3342313	-4.7125015E-02
5	0.4241185	-4.5280814E-02
6	0.5037954	-4.3191597E-02
7	0.5739482	-4.0944993E-02
8	0.6353631	-3.8611036E-02
9	0.6888653	-3.6244839E-02
10	0.7352764	-3.3888750E-02
11	0.7753860	-3.1574685E-02
12	0.8099346	-2.9326305E-02
13	0.8396040	-2.7160212E-02
14	0.8650135	-2.5087669E-02
15	0.8867190	-2.3115689E-02
16	0.9052153	-2.1247901E-02
17	0.9209397	-1.9485427E-02
18	0.9342766	-1.7827630E-02
19	0.9455620	-1.6272269E-02
20	0.9550884	-1.4816190E-02
21	0.9631100	-1.3455539E-02
22	0.9698462	-1.2185992E-02
23	0.9754869	-1.1002984E-02

24	0.9801953	-9.9017732E-03
25	0.9841117	-8.8776452E-03
26	0.9873562	-7.9259165E-03
27	0.9900325	-7.0420532E-03
28	0.9922282	-6.2216623E-03
29	0.9940189	-5.4604942E-03
30	0.9954687	-4.7545689E-03
31	0.9966323	-4.1000778E-03
32	0.9975563	-3.4934350E-03
33	0.9982802	-2.9312631E-03
34	0.9988374	-2.4104011E-03
35	0.9992564	-1.9278848E-03
36	0.9995613	-1.4809514E-03
37	0.9997723	-1.0670180E-03
38	0.9999065	-6.8368023E-04
39	0.9999784	-3.2869991E-04
40	1.000000	0.0000000E+00
41	0.9999800	3.1604434E-04
42	0.9999169	6.4471620E-04
43	0.9998054	9.8650414E-04
44	0.9996399	1.3419170E-03
45	0.9994140	1.7114739E-03
46	0.9991213	2.0957130E-03
47	0.9987540	2.4951908E-03
48	0.9983044	2.9104769E-03
49	0.9977636	3.3421551E-03
50	0.9971218	3.7908396E-03
51	0.9963688	4.2571523E-03
52	0.9954931	4.7417120E-03
53	0.9944825	5.2451780E-03
54	0.9933233	5.7682167E-03
55	0.9920012	6.3115028E-03
56	0.9905001	6.8757092E-03
57	0.9888027	7.4615260E-03
58	0.9868904	8.0696819E-03
59	0.9847429	8.7008551E-03
60	0.9823383	9.3557714E-03
61	0.9796527	1.0035128E-02
62	0.9766603	1.0739590E-02
63	0.9733334	1.1469876E-02
64	0.9696416	1.2226655E-02
65	0.9655526	1.3010557E-02
66	0.9610311	1.3822161E-02
67	0.9560394	1.4662084E-02
68	0.9505364	1.5530799E-02
69	0.9444786	1.6428791E-02
70	0.9378185	1.7356504E-02
71	0.9305055	1.8314075E-02
72	0.9224851	1.9301843E-02
73	0.9136992	2.0319769E-02
74	0.9040854	2.1367794E-02
75	0.8935770	2.2445738E-02
76	0.8821033	2.3553042E-02
77	0.8695885	2.4689123E-02
78	0.8559526	2.5852967E-02
79	0.8411106	2.7043436E-02
80	0.8249727	2.8258907E-02
81	0.8074445	2.9497487E-02
82	0.7884266	3.0756779E-02
83	0.7678154	3.2033868E-02
84	0.7455025	3.3325493E-02
85	0.7213761	3.4627546E-02

86	0.6953208	3.5935350E-02
87	0.6672184	3.7243534E-02
88	0.6369491	3.8545687E-02
89	0.6043923	3.9834738E-02
90	0.5694283	4.1102458E-02
91	0.5319400	4.2339578E-02
92	0.4918149	4.3535478E-02
93	0.4489476	4.4678412E-02
94	0.4032431	4.5755036E-02
95	0.3546200	4.6750974E-02
96	0.3030155	4.7649808E-02
97	0.2483891	4.8433479E-02
98	0.1907293	4.9082577E-02
99	0.1300590	4.9575854E-02
100	6.6444002E-02	4.9890015E-02
101	0.0000000E+00	5.0000504E-02

Appendix G

Wing Grid Generation Code

Below is the Fortran code (and a sample input file) used to generate computational grids around the wings.

```
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C     THIS PROGRAM GENERATES A 3-D O-H GRID FOR AN ELLIPTICAL DELTA WING
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

      PARAMETER(ISIZ=85,JSIZ=85,KSIZ=111)
      IMPLICIT REAL (A-H,O-Z)
      REAL AOA,MACH,SWDEG,DX,DXO
      DIMENSION X(ISIZ,JSIZ,KSIZ),Y(ISIZ,JSIZ,KSIZ),Z(ISIZ,JSIZ,KSIZ),
     &          YOUTER(KSIZ),ZOUTER(KSIZ),YCON(JSIZ,KSIZ),
     &          ZCON(JSIZ,KSIZ),XX(ISIZ),PSI(KSIZ)
      COMMON/GRID/IMAX,JMAX,KMAX,XLEN,YCON,ZCON,YOUTER,ZOUTER,PSI
      COMMON/ANGLES/SWEEPR,CONEANGR,SYM,ECC
      COMMON/CHANGE/IVALUE
      COMMON/VISC/VISSTRI,JVIS,VISSTRO
      CHARACTER*30 GRNAME
      LOGICAL SYM
      OPEN(UNIT=15,STATUS='OLD',FORM='FORMATTED',FILE='wingn.inp')
      OPEN(UNIT=4,STATUS='UNKNOWN',FORM='FORMATTED',FILE='wing.dat')

      READ(15,*) IMAX
      READ(15,*) JMAX
      READ(15,*) KMAX
      READ(15,*) ECC
      READ(15,*) SYM
      READ(15,*) SWEEP
      READ(15,*) AMACH
      READ(15,*) ALPHA
      READ(15,*) AFAC
      READ(15,*) BFAC
      READ(15,*) ZFAC
      READ(15,*) ITE
```

```
      READ(15,*) XSTART,XEND
      read(15,*) VISSTRI,JVIS,VISSTRO
      READ(15,*) KHALF,TTOP,TBOT
      READ(15,*) GRNAME


      PI = 4.DO*DATAN(1.DO)
      RAD = PI/180.DO
      TANANG = TAN(ALPHA*RAD)
      SWEEPR = SWEEP * RAD
      CONEANGR = PI*0.5DO-SWEEPR
      ALPHAR = ALPHA * RAD
      ANGM = DASIN(1.DO/AMACH)

C....  angles with respect to freestream dir
      SHCKANGW = ANGM + ALPHAR
      SHCKANGL = ANGM

      ILE = 1
      DELX = 2.0/REAL(IMAX-1)
      IMAXN = IMAX
      ITEN = ITE

cray      WRITE(10) IMAXN,JMAX,KMAX,ILE,ITEN
c      WRITE(4,*) 'ijk,ile,ite=',IMAXN,JMAX,KMAX,ILE,ITEN
      IF(SYM) THEN
        ISYM=1
      ELSE
        ISYM=0
      ENDIF
      AOA=ALPHA
      MACH = AMACH
      SWDEG = SWEEP
cray      WRITE(10) MACH,AOA,SWDEG,ISYM
c      WRITE(4,*) 'MACH,AOA=',MACH,AOA
c      WRITE(4,*) 'SWEEP,SYM',SWDEG,ISYM

C..... set up x-distribution ( dx_(i+1)/dx_i = x_(i+1)/x_i )
      DXO = .01
      URELAX = 0.5
      XX(1) = XSTART
      KITER = 0
  9   DX = DXO
      KITER = KITER + 1
      DO 10 I = 1,IMAX
         XX(I+1) = XX(I) + DX
 10      DX = (XX(I+1)/XX(I))*DX
```

```
C
C---- Check whether iteration has converged
      IF (KITER .GT. 1000) THEN
         WRITE(4,*) 'X calculation did not converge -- Stop'
         STOP
      ENDIF
      write(4,*) 'xx(imax),kiter=',xx(imax),kiter
      IF (ABS(XX(IMAX) - XEND) .LT. .000001) GOTO 8
      DXO = DXO*((XEND-XSTART)/(XX(IMAX)-XSTART))**URELAX
      GOTO 9
 8    CONTINUE
      WRITE(4,*) 'KITER = ',KITER
C
C---- Calculate circumferential spacing:  Allow assymetric point distribution
C---- and clustering for symmetric sections
      IF(.NOT. SYM) GOTO 95
      PSI(1) = -PI/2
      PSI(KMAX) = PI/2
C
C---- Top half of symmetric body
      DPSI = 1.
      DO 90 K = KHALF,KMAX-1
         PSI(K+1) = PSI(K) + DPSI
 90      DPSI = DPSI*TTOP
      DO 91 K = KHALF,KMAX
 91      PSI(K) = (PSI(K)/PSI(KMAX))*(PI/2)
C
C---- Bottom half
      DPSI = 1.
      DO 92 K = KHALF,2,-1
         PSI(K-1) = PSI(K) - DPSI
 92      DPSI = DPSI*TBOT
      DO 93 K = KHALF,1,-1
 93      PSI(K) = (PSI(K)/PSI(1))*(-PI/2)
C
C---- Check on "smoothness" of transition from bottom to top
      SMOOTH = ABS(PSI(KHALF+1)/PSI(KHALF-1))
      WRITE(6,*) 'DPSI+/DPSI- =',SMOOTH
C
C---- Limit transition to 1.5 times stretching max.
      TSTRETCH = .5*(TTOP+TBOT)
      SMO = 1 + 1.5*(TSTRETCH-1)
      IF (SMOOTH .GT. SMO .OR. SMOOTH .LT. 1/SMO) THEN
         WRITE(6,*) 'Transition is too rough!!  Stop.'
         STOP
      END IF
      GOTO 97
```

```
C
C---- If not symmetric, do even circumferential spacing
 95     DO 96 K = 1,KMAX
CSYMMETRIC            PSI(K)=PI*REAL(K-1)/(KMAX-1)-0.5*PI
           PSI(K)=2.0*REAL(K-1)/(KMAX-1)-0.50*PI
 96     CONTINUE
 97     CONTINUE

      DO 110 I = 1,IMAX
      write(4,*) '\> ',i,' \>', xx(i),'\\'
      DO 110 J = 1,JMAX
      DO 110 K = 1,KMAX
110      X(I,J,K) = XX(I)

      DO 11 I=1,IMAX
         XLEN = X(I,1,1)
C....  major axis of ellipse which forms outer boundary
        AMAJ = AFAC*XLEN*TAN(SHCKANGL)
C...  minor axis of ellipse which forms outer boundary
        SAFE1 = 2.05
        B1 = SAFE1*ANGM
        BMIN = BFAC*XLEN*TAN(ANGM)
        ZWING = ZFAC*3.DO*XLEN*TAN(ALPHAR)

        DO 3 K=1,KMAX

          T = PSI(K)
          AA = SIN(T)**2/AMAJ**2+COS(T)**2/BMIN**2
          BB = -2.0*ZWING*SIN(T)/AMAJ**2
          CC = ZWING**2/AMAJ**2-1.0

          RAD1 = (-BB+SQRT(BB**2-4.0*AA*CC))/(2.0*AA)
          RAD2 = (-BB-SQRT(BB**2-4.0*AA*CC))/(2.0*AA)
          RAD = MAX(RAD1,RAD2)
          YOUTER(K) = RAD*COS(T)
          ZOUTER(K) = RAD*SIN(T)
          kk= k
  3     CONTINUE

        IVALUE = I
        CALL GENGRD

        chord = ycon(1,40)
        IF (KMAX .LT.70)chord = ycon(1,(KMAX-1)/2+1)
        if(i.eq.1)write(4,*)'i,chord=',i,chord
        if(i.eq.imax)write(4,*)'i,chord=',i,chord
        DO 15 J=1,JMAX
```

```
          DO 16 K=1,KMAX
            Y(I,J,K) = YCON(J,K)
            Z(I,J,K) = ZCON(J,K)
       if(i.eq.1 .and.j.eq.1)
     &write(4,*) '\> ',k,' \>',ycon(j,k)/chord,'\>',zcon(j,k)/chord,'\\'
  16      CONTINUE
  15    CONTINUE

  11  CONTINUE
C
C---- Check whether the grid extends beyond the body
      KMID = (KMAX+1)/2
      RAT1 = Y(1,JMAX,KMID)/Y(1,1,KMID)
      RAT2 = Z(1,JMAX,KMAX)/Y(1,1,KMID)
      RAT3 = ABS(Z(1,JMAX,1)/Y(1,1,KMID))
      WRITE(6,*) 'RATIO AT TIP=',RAT1
      WRITE(6,*) 'RATIO AT TOP=',RAT2
      WRITE(6,*) 'RATIO AT BOT=',RAT3
      IF(RAT1 .LT. 1. .OR. RAT2 .LT. 1. .OR. RAT3 .LT. 1.)THEN
        WRITE(6,*) 'Grid outer boundary intersects body:  STOP'
        WRITE(6,*) 'Adjust AFAC, BFAC, and/or ZFAC and re-run!'
      ENDIF
C
      OPEN(UNIT=10,STATUS='UNKNOWN',FORM='UNFORMATTED',FILE=GRNAME)
      WRITE(10) IMAX,JMAX,KMAX
      WRITE(10) (((X(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX),
     &          (((Y(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX),
     &          (((Z(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX)
      STOP
      END


      SUBROUTINE GENGRD
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$                                                                  $$$$$
C$$$$$   This subroutine generates the grid at each chordwise          $$$$$
C$$$$$   station given the location of the outer boundary and          $$$$$
C$$$$$   body geometry.                                                $$$$$
C$$$$$                                                                  $$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
      PARAMETER(ISIZ=85,JSIZ=85,KSIZ=111)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION X(ISIZ),YCON(JSIZ,KSIZ),ZCON(JSIZ,KSIZ),
```

```
     &          YOUTER(KSIZ),ZOUTER(KSIZ),Y(JSIZ,KSIZ),
     &          Z(JSIZ,KSIZ),PSI(KSIZ)
      COMMON/GRID/IMAX,JMAX,KMAX,XLEN,YCON,ZCON,YOUTER,ZOUTER,PSI
      COMMON/ANGLES/SWEEPR,CONEANGR,SYM,ECC
      COMMON/CHANGE/IVALUE
      LOGICAL SYM
      COMPLEX ZETATMP,ZETAOUT,ZETABOD,ZETA,ZETAV(JSIZ)
      COMMON/VISC/VISSTRI,JVIS,VISSTRO

C
C
C.....  Major and minor axes of wing
C
      PI = 4.0*ATAN(1.)
      A = XLEN*TAN(CONEANGR)
      B = A*SQRT(1.-ECC**2)
C
C.....  Parameters for Joukowski transformation.
C
      C=.5*(A+B)
      S=.5*SQRT(A**2-B**2)
      CS = CMPLX(S,0.0)
C
C.....  Loop over rays.
C
      DO 2 K=1,KMAX
C
C.....  Generate transformed body pts
C
C          IF(SYM) THEN
C              T=PI*REAL(K-1)/(KMAX-1)-PI*0.5
C          ELSE
C              T=2.*PI*REAL(K-1)/(KMAX-1)-PI*0.5
C          ENDIF
          T = PSI(K)
C
          ZETABOD=C*CMPLX(COS(T),SIN(T))
C
C.....  Transform outer boundary points
C
          ZETATMP = CMPLX(YOUTER(K),ZOUTER(K))
          IF(K.EQ.1) THEN
            ZETABOD=CMPLX(0.0,-C)
            ZETATMP=CMPLX(0.0,ZOUTER(K))
          ELSEIF(K.EQ.KMAX) THEN
            ZETABOD=CMPLX(0.0,C)
            ZETATMP=CMPLX(0.0,ZOUTER(K))
```

```
          ENDIF
C
C.....  Choose root by quadrant of shock point.
C
           IF(K.GT.1) THEN
              ZETAOUT = .50*ZETATMP+.50*CSQRT(ZETATMP**2-4.0*CS**2)
          ELSE
              ZETAOUT = .50*ZETATMP-.50*CSQRT(ZETATMP**2-4.0*CS**2)
          ENDIF
C
C----- Viscous stretching in JVIS points ala BLoyd
C          JVIS = 15
C          JVIS = jmax
          DVIS = 1.
          ZETAV(1) = 0.
          DO 11 J = 1,JMAX-1
             ZETAV(J+1) = ZETAV(J) + DVIS
      if(ivalue .eq. 25.and.k.eq.kmax)write(4,*)zetav(j+1),j+1
             IF (J .LT. JVIS)DVIS = DVIS*VISSTRI
 11          IF (J .ge. JVIS)DVIS = DVIS*VISSTRO
          DO 12 J = 1,JMAX
 12          ZETAV(J) = ZETABOD +
     &         (ZETAV(J)/ZETAV(JMAX))*(ZETAOUT-ZETABOD)
c       if(ivalue .eq.25) write(4,*)'zb,zo=',zetabod,zetaout
C
C.....  Divide ray into segments, with exponential stretching.
C
          DO 1 J=1,JMAX
c               COMPX=REAL(J-1.)/(JMAX-1.)
c              RBAR=-LOG(1.0-(1.0-EXP(-BETA))*COMPX)/BETA
c              ZETATMP=ZETABOD+(ZETAOUT-ZETABOD)*RBAR
c              ZETA=ZETATMP+(S**2)/ZETATMP
              ZETA=ZETAV(J)+(S**2)/ZETAV(J)
C
              Z(J,K)=REAL((0.,-1.)*ZETA)
              Y(J,K)=REAL(ZETA)
C
 1        CONTINUE
 2    CONTINUE
        DO 21 J=1,JMAX
          Y(J,1) = 0.0
          Y(J,KMAX) = 0.0
  21    CONTINUE
c        DO 22 K=1,KMAX
c          Z(1,K) = 0.0
c  22    CONTINUE
```

```
      DO 3 J=1,JMAX
      DO 4 K=1,KMAX
         YCON(J,K) = Y(J,K)
         ZCON(J,K) = Z(J,K)
 4    CONTINUE
 3    CONTINUE

 99   RETURN
      END
```

Sample input file:

```
36                      IMAX
64                      JMAX
101                     KMAX
.9987492                ECC - Eccentricity (1=line,0=circle)
.TRUE.                  SYM
65.                     SWEEP
1.6                     AMACH
8.0                     ALPHA
1.5                     AFAC: Makes up and down larger
1.5                     BFAC Makes grid wider
0.175                   ZFAC When increased, moves grid up
36                      ITE
0.1 1.                  XSTART,Xend
1.20 15 1.15            VISSTR
40 1.04 1.08            Grid circumferential definition
'test.gri'              GRNAME
```

Appendix H

Computer Program Listings

SINSS – Semi-Implicit Navier-Stokes Solver

For further information, a listing, or a tape of **SINSS**, please contact

M.I.T. Software Center
Technology Licensing Office
M.I.T. Room E19-722
77 Massachusetts Avenue
Cambridge, MA 02139.

A listing of **SINSS** may be obtained for $15.00. A 9 track tape copy written in the **TAR** format may be obtained for $200.00. Make checks payable to MIT and send to the above address.

```
C*********************************************************************
C                                                                     *
      PROGRAM SINSS
C                                                                     *
C     Semi-Implicit solution of the thin layer Navier-Stokes equations *
C                                                                     *
C          Bernard Loyd                                               *
C          Computational Fluid Dynamics Laboratory                    *
C          Massachusetts Institute of Technology                      *
C          Cambridge, Massachusetts USA                               *
C          6 February 1989                                            *
C                                                                     *
C   Fortran code based on 3-D Euler equations code by Roberts/Goodsell *
C*********************************************************************
      include 'euler.inc'
      include 'blvisc.inc'
      COMMON/SHCK/ JSM,ADDSMO
      OPEN(UNIT=15,STATUS='OLD',FORM='FORMATTED',FILE='sinss.inp')
C
C---  VERSION BL1 CREATED  8/25/88 AT 03:10
C      WRITE(6,*) '*****  VERSION  BL1  *****'
C---  VERSION BL2 (TEST) CREATED  9/15/88 AT 15:00
C      DT  set to minimum at each streamwise station (S-I only)
      WRITE(6,*) '*****  VERSION  BL2 (TEST) *****'

C
      READ(15,*) DATE   ! date
      READ(15,*) MACH   ! mach number
      READ(15,*) AOA  ! angle of attack
      READ(15,*) YAW    ! yaw angle
      READ(15,*) GAM    ! gas constant
      READ(15,*) KAP2   ! 2nd order art.visc. constant
      READ(15,*) KAP4   ! 4th order art.visc. constant
      READ(15,*) ADDSMO,JOUT !Add'l shock smoothing faired in @jsm=jmax-jout
      READ(15,*) CFL    ! cfl number
      READ(15,*) AENTH  ! enthalpy damping coef.
      READ(15,*) ITMAX  ! maximum number of iterations
      READ(15,*) ITCOEF ! number of iter for force calculations
      READ(15,*) ITPRIN ! number of iter for printing
      READ(15,*) BIN    ! if .true. then output is given in binary form
      READ(15,*) CFBIN  ! if .true. then force coefficients in binary
      READ(15,*) ITER
C     If the code is being restarted from a previous solution, the value
C     input for ITER must be > 1 (assign input to unit 8);
C     if you're starting from scratch, ITER = 1
      READ(15,*) ICON   ! used only with supersonic flows
C     if ICON = 0 then initial conditions are set to the conical soln
```

```
C     if ICON = 1 then initial conditions are the soln to a lower alpha
C     (assign input to unit 12) ICON=1 OPTION NOT IMPLEMENTED -- USE RESTART
C***** assign conical solution to unit 11 ******
      READ(15,*) ILE
      READ(15,*) ITE
      READ(15,*) RE
      READ(15,*) EPSR
      READ(15,*) GRNAME
      READ(15,*) RSNAME
      READ(15,*) STNODE
      READ(15,*) SVNAME
      READ(15,*) CFNAME
      READ(15,*) RESTRT
      READ(15,*) INPDAT
      READ(15,*) LOWANG
      READ(15,*) OLDRES
      READ(15,*) CONSOLN
C
CViscous
      READ(15,*) SEMIIMP
      READ(15,*) MUSI
      READ(15,*) REYNUM
      READ(15,*) TINF,TWALL
      READ(15,*) PR,PRT
      RE = REYNUM
C
C---- Consistent units here...
      RGAS = 287.
      TCONST = 110.
      ACONST = (GAM*RGAS*TINF)**.5
      HWALL = (GAM*RGAS/(GAM-1))*TWALL/ACONST**2
      SHCONST = (GAM*RGAS/(GAM-1))*TCONST/ACONST**2
       write(6,*) 'ac,sh,hwALL',ACONST,SHCONST,HWALL
CViscous
C
      OPEN(UNIT=1,STATUS='OLD',FORM='UNFORMATTED',FILE=GRNAME)
      OPEN(UNIT=2,STATUS='NEW',FORM='FORMATTED',FILE=RSNAME)
      OPEN(UNIT=4,STATUS='NEW',FORM='UNFORMATTED',FILE=SVNAME)
      CLOSE(4)
      IF(CFBIN) THEN
        OPEN(UNIT=7,STATUS='unknown',FORM='UNFORMATTED',FILE=CFNAME)
      ELSE
        OPEN(UNIT=7,STATUS='unknown',FORM='FORMATTED',FILE=CFNAME)
      ENDIF

      PI = 4.*ATAN(1.)
C
```

```
C---  Read in the coordinates of the grid cell vertices and initialize
C     the grid from a binary data file.  An O-H mesh topology is assumed.
C
      READ (1) IMAX,JMAX,KMAX
      IM = IMAX - 1
      JM = JMAX - 1
      KM = KMAX - 1
      WRITE(6,*) MACH,AOA
      WRITE(6,*) IM,JM,KM,ITE
      NCELLS = IM*JM*KM
      READ(1) (((X(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX),
     1        (((Y(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX),
     2        (((Z(I,J,K),I=1,IMAX),J=1,JMAX),K=1,KMAX)
      CLOSE (1)
C
      write(6,*) 'imax,isiz=',imax,isiz
      IF(IMAX .GT. ISIZ .OR. JMAX .GT. JSIZ .OR. KMAX .GT. KSIZ)THEN
        WRITE(6,*) 'DIMENSIONS ARE IMPROPER -- STOP!'
        STOP
      ENDIF
      SYM = .FALSE.
c      IF(YAW .EQ. 0.) SYM = .TRUE.
      IF(YAW .EQ. 0. .and. grname .ne. 'ni.gri') SYM = .TRUE.
      IF(GRNAME .EQ. 'ni.gri')then
        write(6,*) 'Reactivate commented out lines in DTSIZE
     & (twice)'
        STOP
      ENDIF
C
C---- Define location for additional shock smoothing:
      JSM = JMAX - JOUT
      IF(1.*JSM .LT. 1.*(JMAX/2))THEN
        WRITE(6,*) 'Shock smoothing too near body: STOP!'
        WRITE(6,*) 'Decrease JOUT'
        STOP
      ENDIF

      CALL PROUT

      CALL NORMAL

      CALL AMEAN
CViscous
      IF (REYNUM .GT. 0. .OR. SEMIIMP) CALL BLETA
CViscous

      ALPHA1 = 0.25
```

```
      ALPHA2 = 1./3.
      ALPHA3 = 0.5
C
C--- initialize the state vector;  note that density and speed of sound
C    are non-dimensionalized by their freestream values
C
      PIFAC = PI/180.
      ALPHA = AOA
      AOA = AOA*PIFAC
      YAW = YAW*PIFAC
      UIN = MACH*COS(AOA)                  ! free stream x-velocity
      WIN = MACH*SIN(AOA)                  ! free stream z-velocity
      PIN = 1./GAM                         ! free stream pressure
      EIN = PIN/(GAM-1.) + 0.5*MACH*MACH   ! freestream energy
      HIN = EIN + PIN                      ! free stream total enthalpy
      CIN = 1.

C---  DELNORM is used to normalize the residuals later
      DELNORM(1) = 1.
      DELNORM(2) = 1./UIN
      DELNORM(3) = 1./UIN
      DELNORM(4) = 1./UIN
      DELNORM(5) = 1./EIN

      IF (ITER.GT.1) THEN
        CALL RESTART
      ELSE IF(MACH.GT.1.0) THEN
        CALL SUPERIC
      ELSE
        CALL SUBSONIC
      ENDIF
      IF(ITMAX.EQ.0) GO TO 11
C
C---  start Euler solution procedure;  establish boundary conditions,
C     dissipation, determine initial value of residuals, and determine
C     size of local time step
C
  1   CALL BNDRYC
      CALL FLUX
      CALL DTSIZE
      CALL DISSIP
C
C---  call four-stage time stepping procedure
C
      CALL TIMSTP
C
C--- calculate force and moment coefficients every ITCOEF iterations,
```

```
C     and when done.
C
      IF(MOD(ITER,ITCOEF).EQ.0 .OR. ITER .EQ. ITMAX) THEN
        CALL COEF
      ENDIF
C
C---  write out solution (restart file) every ITPRIN iterations,
C     or when done (ITER = ITMAX)
C
      IF (MOD(ITER,ITPRIN). EQ .0. OR .ITER. EQ .ITMAX) THEN
        CALL SAVESET
      ENDIF
C
      ITER = ITER + 1
      IF (ITER. GT .ITMAX) GO TO 10
      GO TO 1
   10 CONTINUE

C---  calculate the state vector at the nodes for plotting
   11 CALL BNDRYC
      CALL UNODES
C
      STOP
      END
```

```
C
C
C  SUBROUTINE AMEAN:  determines mean projected area in x,y,and z
C  directions of each grid cell for use in calculation of time step
C
C  For semi-implicit runs, this subroutine is repeated so that semi-implicit
C  and explicit (for use in LHSSMO) can be calculated...
C
      SUBROUTINE AMEAN
      include 'euler.inc'
      include 'blvisc.inc'
      LOGICAL SEMISTO
      COMMON/AEXP/AXEXP(ISIZ,JSIZ,KSIZ),AYEXP(ISIZ,JSIZ,KSIZ),
     &            AZEXP(ISIZ,JSIZ,KSIZ)
C
C---- Determine whether to run twice
      SEMISTO = .FALSE.
      IF (SEMIIMP) THEN
         SEMIIMP = .FALSE.
         SEMISTO = .TRUE.
      ENDIF
 1    CONTINUE
C
C---- Do interior and i=1,IM
      DO 30 I = 1,IM
        II = I - 1
        DO 20 J = 2,JM
          JJ = J - 1
          DO 10 K = 2,KM
            KK = K-1
            A1 = ABS(FACEX(1,I,J,K))
            A2 = ABS(FACEX(2,I,J,K))
            A3 = ABS(FACEX(3,I,J,K))
            A4 = ABS(FRONT(J,K))
            IF(II.GT.0) A4 = ABS(FACEX(1,II,J,K))
            A5 = ABS(FACEX(2,I,JJ,K))
            A6 = ABS(FACEX(3,I,J,KK))
            AX = 0.
            IF (SEMIIMP) THEN
              A2 = 0.
               A5 = 0.
            ENDIF
            AX = MAX(AX,(A1 + A2 + A3))
            AX = MAX(AX,(A2 + A3 + A4))
            AX = MAX(AX,(A2 + A4 + A6))
            AX = MAX(AX,(A1 + A2 + A6))
            AX = MAX(AX,(A1 + A3 + A5))
```

```
      AX = MAX(AX,(A3 + A4 + A5))
      AX = MAX(AX,(A4 + A5 + A6))
      AX = MAX(AX,(A1 + A5 + A6))
      AXM(I,J,K) = AX
C
      A1 = ABS(FACEY(1,I,J,K))
      A2 = ABS(FACEY(2,I,J,K))
      A3 = ABS(FACEY(3,I,J,K))
      A4 = 0.
      IF(II.GT.0) A4 = ABS(FACEY(1,II,J,K))
      A5 = ABS(FACEY(2,I,JJ,K))
      A6 = ABS(FACEY(3,I,J,KK))
      IF (SEMIIMP) THEN
         A2 = 0.
         A5 = 0.
      ENDIF
      AY = 0.
      AY = MAX(AY,(A1 + A2 + A3))
      AY = MAX(AY,(A2 + A3 + A4))
      AY = MAX(AY,(A2 + A4 + A6))
      AY = MAX(AY,(A1 + A2 + A6))
      AY = MAX(AY,(A1 + A3 + A5))
      AY = MAX(AY,(A3 + A4 + A5))
      AY = MAX(AY,(A4 + A5 + A6))
      AY = MAX(AY,(A1 + A5 + A6))
      AYM(I,J,K) = AY
C
      A1 = ABS(FACEZ(1,I,J,K))
      A2 = ABS(FACEZ(2,I,J,K))
      A3 = ABS(FACEZ(3,I,J,K))
      A4 = 0.
      IF(II.GT.0) A4 = ABS(FACEZ(1,II,J,K))
      A5 = ABS(FACEZ(2,I,JJ,K))
      A6 = ABS(FACEZ(3,I,J,KK))
      IF (SEMIIMP) THEN
         A2 = 0.
         A5 = 0.
      ENDIF
      AZ = 0.
      AZ = MAX(AZ,(A1 + A2 + A3))
      AZ = MAX(AZ,(A2 + A3 + A4))
      AZ = MAX(AZ,(A2 + A4 + A6))
      AZ = MAX(AZ,(A1 + A2 + A6))
      AZ = MAX(AZ,(A1 + A3 + A5))
      AZ = MAX(AZ,(A3 + A4 + A5))
      AZ = MAX(AZ,(A4 + A5 + A6))
      AZ = MAX(AZ,(A1 + A5 + A6))
```

```
            AZM(I,J,K) = AZ
C
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---- Do k = 1
      DO 50 I = 1,IM
          II = I - 1
        DO 40 J = 2,JM
          JJ = J - 1
          A1 = ABS(FACEX(1,I,J,1))
          A2 = ABS(FACEX(2,I,J,1))
          A3 = ABS(FACEX(3,I,J,1))
          A4 = FRONT(J,1)
          IF(II.GT.0) A4 = ABS(FACEX(1,II,J,1))
          A5 = ABS(FACEX(2,I,JJ,1))
          A6 = 0.
            IF (SEMIIMP) THEN
               A2 = 0.
               A5 = 0.
            ENDIF
          AX = 0.
          AX = MAX(AX,(A1 + A2 + A3))
          AX = MAX(AX,(A2 + A3 + A4))
          AX = MAX(AX,(A2 + A4 + A6))
          AX = MAX(AX,(A1 + A2 + A6))
          AX = MAX(AX,(A1 + A3 + A5))
          AX = MAX(AX,(A3 + A4 + A5))
          AX = MAX(AX,(A4 + A5 + A6))
          AX = MAX(AX,(A1 + A5 + A6))
          AXM(I,J,1) = AX
C
          A1 = ABS(FACEY(1,I,J,1))
          A2 = ABS(FACEY(2,I,J,1))
          A3 = ABS(FACEY(3,I,J,1))
          A4 = 0.
          IF(II.GT.0) A4 = ABS(FACEY(1,II,J,1))
          A5 = ABS(FACEY(2,I,JJ,1))
          A6 = ABS(WALLB(I,J))
            IF (SEMIIMP) THEN
               A2 = 0.
               A5 = 0.
            ENDIF
          AY = 0.
          AY = MAX(AY,(A1 + A2 + A3))
          AY = MAX(AY,(A2 + A3 + A4))
```

```
          AY = MAX(AY,(A2 + A4 + A6))
          AY = MAX(AY,(A1 + A2 + A6))
          AY = MAX(AY,(A1 + A3 + A5))
          AY = MAX(AY,(A3 + A4 + A5))
          AY = MAX(AY,(A4 + A5 + A6))
          AY = MAX(AY,(A1 + A5 + A6))
          AYM(I,J,1) = AY
C
          A1 = ABS(FACEZ(1,I,J,1))
          A2 = ABS(FACEZ(2,I,J,1))
          A3 = ABS(FACEZ(3,I,J,1))
          A4 = 0.
          IF(II.GT.0) A4 = ABS(FACEZ(1,II,J,1))
          A5 = ABS(FACEZ(2,I,JJ,1))
          A6 = 0.
            IF (SEMIIMP) THEN
               A2 = 0.
               A5 = 0.
            ENDIF
          AZ = 0.
          AZ = MAX(AZ,(A1 + A2 + A3))
          AZ = MAX(AZ,(A2 + A3 + A4))
          AZ = MAX(AZ,(A2 + A4 + A6))
          AZ = MAX(AZ,(A1 + A2 + A6))
          AZ = MAX(AZ,(A1 + A3 + A5))
          AZ = MAX(AZ,(A3 + A4 + A5))
          AZ = MAX(AZ,(A4 + A5 + A6))
          AZ = MAX(AZ,(A1 + A5 + A6))
          AZM(I,J,1) = AZ
   40   CONTINUE
   50 CONTINUE
C
C---- Do j = 1
      DO 70 I = 1,IM
        II = I-1
        DO 60 K = 1,KM
          KK = K - 1
          A1 = ABS(FACEX(1,I,1,K))
          A2 = ABS(FACEX(2,I,1,K))
          A3 = ABS(FACEX(3,I,1,K))
          A4 = FRONT(1,K)
          IF(II.GT.0) A4 = ABS(FACEX(1,II,1,K))
          A5 = ABS(FACEX(2,I,0,K))
          A6 = 0.
          IF (KK. GT .0) A6 = ABS(FACEX(3,I,1,KK))
            IF (SEMIIMP) THEN
               A2 = 0.
```

```
          A5 = 0.
        ENDIF
      AX = 0.
      AX = MAX(AX,(A1 + A2 + A3))
      AX = MAX(AX,(A2 + A3 + A4))
      AX = MAX(AX,(A2 + A4 + A6))
      AX = MAX(AX,(A1 + A2 + A6))
      AX = MAX(AX,(A1 + A3 + A5))
      AX = MAX(AX,(A3 + A4 + A5))
      AX = MAX(AX,(A4 + A5 + A6))
      AX = MAX(AX,(A1 + A5 + A6))
      AXM(I,1,K) = AX
C
      A1 = ABS(FACEY(1,I,1,K))
      A2 = ABS(FACEY(2,I,1,K))
      A3 = ABS(FACEY(3,I,1,K))
      A4 = 0.
      IF(II.GT.0) A4 = ABS(FACEY(1,II,1,K))
      A5 = ABS(FACEY(2,I,0,K))
      A6 = ABS(WALLB(I,1))
      IF (KK. GT .0) A6 = ABS(FACEY(3,I,1,KK))
        IF (SEMIIMP) THEN
          A2 = 0.
          A5 = 0.
        ENDIF
      AY = 0.
      AY = MAX(AY,(A1 + A2 + A3))
      AY = MAX(AY,(A2 + A3 + A4))
      AY = MAX(AY,(A2 + A4 + A6))
      AY = MAX(AY,(A1 + A2 + A6))
      AY = MAX(AY,(A1 + A3 + A5))
      AY = MAX(AY,(A3 + A4 + A5))
      AY = MAX(AY,(A4 + A5 + A6))
      AY = MAX(AY,(A1 + A5 + A6))
      AYM(I,1,K) = AY
C
      A1 = ABS(FACEZ(1,I,1,K))
      A2 = ABS(FACEZ(2,I,1,K))
      A3 = ABS(FACEZ(3,I,1,K))
      A4 = 0.
      IF(II.GT.0) A4 = ABS(FACEZ(1,II,1,K))
      A5 = ABS(FACEZ(2,I,0,K))
      A6 = 0.
      IF (KK. GT .0) A6 = ABS(FACEZ(3,I,1,KK))
        IF (SEMIIMP) THEN
          A2 = 0.
          A5 = 0.
```

```
            ENDIF
           AZ = 0.
           AZ = MAX(AZ,(A1 + A2 + A3))
           AZ = MAX(AZ,(A2 + A3 + A4))
           AZ = MAX(AZ,(A2 + A4 + A6))
           AZ = MAX(AZ,(A1 + A2 + A6))
           AZ = MAX(AZ,(A1 + A3 + A5))
           AZ = MAX(AZ,(A3 + A4 + A5))
           AZ = MAX(AZ,(A4 + A5 + A6))
           AZ = MAX(AZ,(A1 + A5 + A6))
           AZM(I,1,K) = AZ
   60    CONTINUE
   70 CONTINUE
C
C---- Store explicit stuff for use in calculating explicit dt's
      IF(SEMISTO) THEN
        SEMIIMP = .TRUE.
        SEMISTO = .FALSE.
        DO 100 I = 1,IM
        DO 100 J = 1,JM
        DO 100 K = 1,KM
           AXEXP(I,J,K) = AXM(I,J,K)
           AYEXP(I,J,K) = AYM(I,J,K)
 100       AZEXP(I,J,K) = AZM(I,J,K)
        GOTO 1
      ENDIF


c      do 80 j=1,10
c      do 80 k=1,km
c 80        write(6,81) axm(1,j,k),aym(1,j,k),azm(1,j,k),j,k
c 81   format('ax,ay,az=',3E18.3E2,' jk=',2i3)
      RETURN
      END
```

```
C
C
C  SUBROUTINE BNDRYC:  sets the far-field and solid wall boundary
C  conditions for Jameson's explicit four-stage scheme.  At the
C  wing surface, the pressure is extrapolated from interior values;
C  far-field conditions are determined by assuming one-dimensional
C  flow normal to the boundary, and using Riemann invariants in
C  the normal direction.
C
      SUBROUTINE BNDRYC
      include 'euler.inc'

C
C---  first, determine the pressure field
C
      DO 30 I = 0,IM
        DO 20 J = 1,JM
          DO 10 K = 1,KM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            EN  = U(5,I,J,K)
            P(I,J,K) = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*
     &       (GAM - 1.)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      IPNEG = 0
      IDNEG = 0
C
C---- Prevent pressure from going negative during iteration
      DO 21 I = 0,IM
      DO 21 J = 1,JM
      DO 21 K = 1,KM
         IF(P(I,J,K) .LE. 0.00000001) THEN
            P(I,J,K) = 0.000000001
            IPNEG = 1
         ENDIF
         IF(U(1,I,J,K) .LE. 0.000000001) THEN
            U(1,I,J,K) = 0.000000001
            IDNEG = 1
         ENDIF
 21   CONTINUE
      IF (IPNEG .EQ. 1)WRITE(6,*) 'Pressure wants to go negative'
      IF (IDNEG .EQ. 1)WRITE(6,*) 'Density wants to go negative'
      IF(.NOT. SYM) THEN
```

```
        DO 31 I = 1,IM
          P(I,1,0) = P(I,1,KM)
          P(I,1,KMAX) = P(I,1,1)
   31   CONTINUE
      ENDIF

      IF(MACH.GT.1.0) GO TO 99
C
C---  first, determine the far-field conditions at the upstream boundary
C
      DO 62 J = 1,JM
        DO 52 K = 1,KM
C
C---  find density, velocity, and pressure at the first point
C     inside the computational domain, (1,j,k)
C
          RHOEX = U(1,1,J,K)
          UEX   = U(2,1,J,K)/RHOEX
          VEX   = U(3,1,J,K)/RHOEX
          WEX   = U(4,1,J,K)/RHOEX
          PEX   =   P(1,J,K)
C          IF(PEX.LT.0.) GO TO 200
C          IF(RHOEX.LT.0.) GO TO 201
C
C---  find the outward unit normal to the computational domain
C     and determine the normal component of the free stream and
C     extrapolated velocities and sonic speeds
C
          ONX = -FRONT(J,K)
          ONY = 0.
          ONZ = 0.
          ON  = SQRT(ONX**2 + ONY**2 + ONZ**2)
          ONX = ONX/ON
          ONY = ONY/ON
          ONZ = ONZ/ON
C
          UINN = ONX*UIN +           ONZ*WIN
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---  compute the incoming and outgoing Riemann invariants
C
         RIN = UINN - 2.*CIN/(GAM - 1.)
         REX = UEXN + 2.*CEX/(GAM - 1.)
C
```

```
C---  supersonic inflow:  all variables specified

         IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)

C---  supersonic outflow:  all variables extrapolated

         IF (EV5. GT .0.) RIN = UEXN - 2.*CEX/(GAM - 1.)

C---  these two lines are the Cray vectorizable statements that do the
C     same thing as the above IF statements
C
C          REX = CVMGM((UINN + 2.*CIN/(GAM - 1.)),(UEXN + 2.*CEX/(GAM -
C     &     1.)),EV4)
C          RIN = CVMGP((UEXN - 2.*CEX/(GAM - 1.)),(UINN - 2.*CIN/(GAM -
C     &     1.)),EV5)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (1,j,k)
C
          UN = (REX + RIN)/2.
          C  = (REX - RIN)*(GAM - 1.)/4.
          IF (UN. GT .0.) THEN

C---  outflow boundary:  extrapolate tangential velocity, entropy

           UP = UEX + (UN - UEXN)*ONX
           VP = VEX + (UN - UEXN)*ONY
           WP = WEX + (UN - UEXN)*ONZ
           S = PEX/RHOEX**GAM
          ELSE

C---  inflow boundary:  free stream tangential velocity, entropy
C
           UP = UIN + (UN - UINN)*ONX
           VP =       (UN - UINN)*ONY
           WP = WIN + (UN - UINN)*ONZ
           S = PIN
          END IF
C
C---  these four lines are the Cray vectorizable statements that do the
C     same thing as the above IF THEN ELSE block
C
C          UP = CVMGP((UEX + (UN - UEXN)*ONX),(UIN + (UN - UINN)*ONX),
C     &     UN)
C          VP = CVMGP((VEX + (UN - UEXN)*ONY),((UN - UINN)*ONY),
C     &     UN)
C          WP = CVMGP((WEX + (UN - UEXN)*ONZ),(WIN + (UN - UINN)*ONZ),
```

```
C     &     UN)
C          S = CVMGP((PEX/RHOEX**GAM),PIN,UN)
C
C
C---  determine density, energy;  set conservation variables at boundary
C     cell
C
          ENER = (UP**2 + VP**2 + WP**2)/2. + C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
cbl          U(1,0,J,K) = RHO
cbl          U(2,0,J,K) = RHO*UP
cbl          U(3,0,J,K) = RHO*VP
cbl          U(4,0,J,K) = RHO*WP
cbl          U(5,0,J,K) = RHO*ENER
cbl          P(0,J,K) = RHO*C**2/GAM
          U(1,0,J,K) = 2*RHO    - U(1,1,J,K)
          U(2,0,J,K) = 2*RHO*UP - U(2,1,J,K)
          U(3,0,J,K) = 2*RHO*VP - U(3,1,J,K)
          U(4,0,J,K) = 2*RHO*WP - U(4,1,J,K)
          U(5,0,J,K) = 2*RHO*ENER - U(5,1,J,K)
          P(0,J,K) = 2*RHO*C**2/GAM - P(1,J,K)
   52   CONTINUE
   62 CONTINUE
C       GO TO 99
C  200 WRITE(6,*) 'UPSTREAM BOUNDARY CONDITION'
C       WRITE(6,*) 'NEGATIVE PEX AT I,J,K = 1,',J,K
C       STOP
C  201 WRITE(6,*) 'UPSTREAM BOUNDARY CONDITION'
C       WRITE(6,*) 'NEGATIVE RHOEX AT I,J,K = 1,',J,K
C       STOP
   99 CONTINUE
C
C---  next, determine the far-field conditions at the downstream boundary
C
C
C---- For viscous flow, extrapolate linearly (this is generally much better
C     than assuming constant):
      IF (RE .GT. 0.) THEN
       DO 55 J=1,JM
       DO 55 K=1,KM
          U(1,IMAX,J,K) = 2*U(1,IM,J,K) - U(1,IM-1,J,K)
          U(2,IMAX,J,K) = 2*U(2,IM,J,K) - U(2,IM-1,J,K)
          U(3,IMAX,J,K) = 2*U(3,IM,J,K) - U(3,IM-1,J,K)
          U(4,IMAX,J,K) = 2*U(4,IM,J,K) - U(4,IM-1,J,K)
          U(5,IMAX,J,K) = 2*U(5,IM,J,K) - U(5,IM-1,J,K)
 55       P(IMAX,J,K) = P(IM,J,K)
```

```
C
C---- For inviscid flows use Riemann invarients
      ELSE
        DO 61 J = 1,JM
        DO 51 K = 1,KM
C
C---  find density, velocity, and pressure at the last point
C     inside the computational domain, (I-1,j,k)
C
          RHOEX = U(1,IM,J,K)
          UEX   = U(2,IM,J,K)/RHOEX
          VEX   = U(3,IM,J,K)/RHOEX
          WEX   = U(4,IM,J,K)/RHOEX
          PEX   =   P(IM,J,K)
C          IF(PEX.LT.0.) GO TO 210
C          IF(RHOEX.LT.0.) GO TO 211
C
C---  find the outward unit normal to the computational domain
C     and determine the normal component of the free stream and
C     extrapolated velocities and sonic speeds
C
          ONX = FACEX(1,IM,J,K)
          ONY = FACEY(1,IM,J,K)
          ONZ = FACEZ(1,IM,J,K)
          ON  = SQRT(ONX**2 + ONY**2 + ONZ**2)
          ONX = ONX/ON
          ONY = ONY/ON
          ONZ = ONZ/ON
C
          UINN = ONX*UIN +           ONZ*WIN
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---  compute the incoming and outgoing Riemann invariants
C
          RIN = UINN - 2.*CIN/(GAM - 1.)
          REX = UEXN + 2.*CEX/(GAM - 1.)
C
C---  supersonic inflow:  all variables specified
C
          IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)
C
C---  supersonic or viscous outflow:  all variables extrapolated
C
          IF (EV5. GT .0.)
```

```
     &       RIN = UEXN - 2.*CEX/(GAM - 1.)
C
C---  these two lines are the Cray vectorizable statements that do the
C     same thing as the above IF statements (NOT FOR VISCOUS FLOW!)
C
C          REX = CVMGM((UINN + 2.*CIN/(GAM - 1.)),(UEXN + 2.*CEX/(GAM -
C     &     1.)),EV4)
C          RIN = CVMGP((UEXN - 2.*CEX/(GAM - 1.)),(UINN - 2.*CIN/(GAM -
C     &     1.)),EV5)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (I,j,k)
C
          UN = (REX + RIN)/2.
          C  = (REX - RIN)*(GAM - 1.)/4.
          IF (UN. GT .0.) THEN
C
C---  outflow boundary:  extrapolate tangential velocity, entropy
C
           UP = UEX + (UN - UEXN)*ONX
           VP = VEX + (UN - UEXN)*ONY
           WP = WEX + (UN - UEXN)*ONZ
           S = PEX/RHOEX**GAM
          ELSE
C
C---  inflow boundary:  free stream tangential velocity, entropy
C
           UP = UIN + (UN - UINN)*ONX
           VP =       (UN - UINN)*ONY
           WP = WIN + (UN - UINN)*ONZ
           S = PIN
          END IF
C
C---  these four lines are the Cray vectorizable statements that do the
C     same thing as the above IF THEN ELSE block
C
C          UP = CVMGP((UEX + (UN - UEXN)*ONX),(UIN + (UN - UINN)*ONX),
C     &     UN)
C          VP = CVMGP((VEX + (UN - UEXN)*ONY),((UN - UINN)*ONY),
C     &     UN)
C          WP = CVMGP((WEX + (UN - UEXN)*ONZ),(WIN + (UN - UINN)*ONZ),
C     &     UN)
C          S = CVMGP((PEX/RHOEX**GAM),PIN,UN)

C
C---  determine density, energy;  set conservation variables at boundary
C     cell
```

```
C
          ENER = (UP**2 + VP**2 + WP**2)/2. + C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
CBL          U(1,IMAX,J,K) = RHO
CBL          U(2,IMAX,J,K) = RHO*UP
CBL          U(3,IMAX,J,K) = RHO*VP
CBL          U(4,IMAX,J,K) = RHO*WP
CBL          U(5,IMAX,J,K) = RHO*ENER
CBL          P(IMAX,J,K) = RHO*C**2/GAM
          U(1,IMAX,J,K) = 2*RHO    - U(1,IM,J,K)
          U(2,IMAX,J,K) = 2*RHO*UP - U(2,IM,J,K)
          U(3,IMAX,J,K) = 2*RHO*VP - U(3,IM,J,K)
          U(4,IMAX,J,K) = 2*RHO*WP - U(4,IM,J,K)
          U(5,IMAX,J,K) = 2*RHO*ENER - U(5,IM,J,K)
          P(IMAX,J,K) = 2*RHO*C**2/GAM - P(IM,J,K)

C        if (j.eq.1 .and. (k .gt.19 .and. k .lt.26))then
C         WRITE(6,*) J,K
C         WRITE(6,*) 'ex:',RHO,UEX,VEX,WEX,PEX
c         write(6,*) 'im:'
c         WRITE(6,*) U(1,IM,J,K),U(2,IM,J,K),U(3,IM,J,K),
c     &              U(4,IM,J,K),U(5,IM,J,K)
c         write(6,*) 'imax:'
c         WRITE(6,*) U(1,IMAX,J,K),U(2,IMAX,J,K),U(3,IMAX,J,K),
c     &              U(4,IMAX,J,K),U(5,IMAX,J,K)
c       endif
   51    CONTINUE
   61 CONTINUE
      ENDIF
C       GO TO 299

C  210 WRITE(6,*) 'DOWNSTREAM BOUNDARY CONDITION'
C      WRITE(6,*) 'NEGATIVE PEX AT I,J,K = ',IM,J,K
C      STOP
C  211 WRITE(6,*) 'DOWNSTREAM BOUNDARY CONDITION'
C      WRITE(6,*) 'NEGATIVE RHOEX AT I,J,K =',IM,J,K
C      STOP
C  299 CONTINUE
C
C---  next, determine the far-field conditions on the outer boundary
C
         DO 60 I = 1,IM
         DO 50 K = 1,KM
C
C-BL--- FOR NI BUMP ONLY*****************************************************
         IF(grname .NE. 'ni.gri') goto 1001
```

```
          U(1,I,JMAX,K) = U(1,I,JM,K)
          U(2,I,JMAX,K) = U(2,I,JM,K)
          U(3,I,JMAX,K) =-U(3,I,JM,K)
          U(4,I,JMAX,K) = U(4,I,JM,K)
          U(5,I,JMAX,K) = U(5,I,JM,K)
          P(I,JMAX,K) = P(I,JM,K)
          goto 50
1001     continue
C-BL--- ABOVE IS FOR NI BUMP ONLY*********************************************
C
C---   find density, velocity, and pressure at the last point
C      inside the computational domain, (i,J-1,k)
C
          RHOEX = U(1,I,JM,K)
          UEX   = U(2,I,JM,K)/RHOEX
          VEX   = U(3,I,JM,K)/RHOEX
          WEX   = U(4,I,JM,K)/RHOEX
          PEX   =   P(I,JM,K)
C          IF(PEX.LT.0.) GO TO 220
C          IF(RHOEX.LT.0) GO TO 221
C
C---   find the outward unit normal to the computational domain
C      and determine the normal component of the free stream and
C      extrapolated velocities and sonic speeds
C
          ONX = FACEX(2,I,JM,K)
          ONY = FACEY(2,I,JM,K)
          ONZ = FACEZ(2,I,JM,K)
          ON  = SQRT(ONX**2 + ONY**2 + ONZ**2)
          ONX = ONX/ON
          ONY = ONY/ON
          ONZ = ONZ/ON
C
          UINN = ONX*UIN +           ONZ*WIN
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---   compute the incoming and outgoing Riemann invariants
C
          RIN = UINN - 2.*CIN/(GAM - 1.)
          REX = UEXN + 2.*CEX/(GAM - 1.)
C
C---   supersonic inflow:  all variables specified
C
          IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)
```

```
C
C---  supersonic outflow:  all variables extrapolated
C
          IF (EV5. GT .0.) RIN = UEXN - 2.*CEX/(GAM - 1.)
C
C---  these two lines are the Cray vectorizable statements that do the
C     same thing as the above IF statements
C
C           REX = CVMGM((UINN + 2.*CIN/(GAM - 1.)),(UEXN + 2.*CEX/(GAM -
C     &      1.)),EV4)
C           RIN = CVMGP((UEXN - 2.*CEX/(GAM - 1.)),(UINN - 2.*CIN/(GAM -
C     &      1.)),EV5)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (i,J,k)
C
          UN = (REX + RIN)/2.
          C  = (REX - RIN)*(GAM - 1.)/4.
          IF (UN. GT .0.) THEN
C
C---  outflow boundary:  extrapolate tangential velocity, entropy
C
           UP = UEX + (UN - UEXN)*ONX
           VP = VEX + (UN - UEXN)*ONY
           WP = WEX + (UN - UEXN)*ONZ
           S = PEX/RHOEX**GAM
          ELSE
C
C---  inflow boundary:  free stream tangential velocity, entropy
C
           UP = UIN + (UN - UINN)*ONX
           VP =       (UN - UINN)*ONY
           WP = WIN + (UN - UINN)*ONZ
           S = PIN
          END IF
C
C---  these four lines are the Cray vectorizable statements that do the
C     same thing as the above IF THEN ELSE block
C
C           UP = CVMGP((UEX + (UN - UEXN)*ONX),(UIN + (UN - UINN)*ONX),
C     &      UN)
C           VP = CVMGP((VEX + (UN - UEXN)*ONY),((UN - UINN)*ONY),
C     &      UN)
C           WP = CVMGP((WEX + (UN - UEXN)*ONZ),(WIN + (UN - UINN)*ONZ),
C     &      UN)
C           S = CVMGP((PEX/RHOEX**GAM),PIN,UN)
```

```
C
C---  determine density, energy;  set conservation variables at boundary
C     cell
C
          ENER = (UP**2 + VP**2 + WP**2)/2. + C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
CBL          U(1,I,JMAX,K) = RHO
CBL          U(2,I,JMAX,K) = RHO*UP
CBL          U(3,I,JMAX,K) = RHO*VP
CBL          U(4,I,JMAX,K) = RHO*WP
CBL          U(5,I,JMAX,K) = RHO*ENER
CBL          P(I,JMAX,K) = RHO*C**2/GAM
          U(1,I,JMAX,K) = 2*RHO - U(1,I,JM,K)
          U(2,I,JMAX,K) = 2*RHO*UP - U(2,I,JM,K)
          U(3,I,JMAX,K) = 2*RHO*VP - U(3,I,JM,K)
          U(4,I,JMAX,K) = 2*RHO*WP - U(4,I,JM,K)
          U(5,I,JMAX,K) = 2*RHO*ENER - U(5,I,JM,K)
          P(I,JMAX,K) = 2*RHO*C**2/GAM - P(I,JM,K)

C         WRITE(34,*) I,K
C         WRITE(34,*) ONX,ONY,ONZ
C         WRITE(34,*) UINN,UN,ENER
C         WRITE(34,*) U(1,I,JMAX,K),U(2,I,JMAX,K),U(3,I,JMAX,K),
C     &               U(4,I,JMAX,K),U(5,I,JMAX,K)
   50   CONTINUE
   60 CONTINUE
C      GO TO 399

C 220  WRITE(6,*) 'OUTER BOUNDARY'
C      WRITE(6,*) 'NEGATIVE PEX AT I,J,K = ',I,JM,K
C      STOP
C 221  WRITE(6,*) 'OUTER BOUNDARY'
C      WRITE(6,*) 'NEGATIVE RHOEX AT I,J,K =',I,JM,K
C      STOP
C
C 399  CONTINUE

      IF(SYM) THEN
C
C---  determine the pressure at the symmetry plane by extrapolation
C     from the interior--set (dP/dn) = 0 at the symmetry plane;
C
C---  First, do the symmetry plane
C
      DO 80 J = 1,JM
        DO 70 I = 1,IM
```

```
          PRESYMB(I,J) = P(I,J,1)
          PRESYMT(I,J) = P(I,J,KM)
   70   CONTINUE
   80 CONTINUE
      ENDIF
C
C---  Next, do the wing surface (pressure b.c.'s using normal momentum
C     equation is in PRESSURE.SAV). The normal momentum equation is used to
C     determine the pressure on the wing.  On the wing, the local
C     coordinates xi,eta,zeta correspond to the indice k,i,j,
C     respectively (that is not a typo).
C
      DO 100 I = ILE,ITE-1
        DO 90 K = 1,KM
          PRESWG(I,K) = P(I,1,K)
   90 CONTINUE
  100 CONTINUE

      RETURN
      END
```

```
C
C  SUBROUTINE COEF:  this subroutine calculates the coefficients of
C  lift, drag, root bending moment, and moment about the apex for
C  the whole wing.
C
      SUBROUTINE COEF
      include 'euler.inc'

C--- initialize coefficients for summing
      CZ = 0.
      CX = 0.
      CMR = 0.
      CMLE = 0.
      S = 0.

C--- calculate pressure coefficient at each cell on wing surface and
C    use for force calculations
      KTIP = (KMAX+1)/2
      DO 10 I=ILE,ITE-1
        DO 11 K=1,KM
          CP = 2.*(PRESWG(I,K) - PIN)/(MACH*MACH)

          CZ = CZ - CP*FACEZ(2,I,0,K)
          CX = CX - CP*FACEX(2,I,0,K)

          CMR = CZ*0.5*(Y(I,1,K) + Y(I,1,K+1))
          CMLE = CZ*0.5*(X(I+1,1,K) + X(I,1,K+1))

          IF(K.LT.KTIP) S = S + ABS(FACEZ(2,I,0,K))

  11    CONTINUE
  10  CONTINUE

      IF (S .NE. 0.) THEN
      CL = (CZ*COS(AOA) - CX*SIN(AOA))/S
      CD = (CZ*SIN(AOA) + CX*COS(AOA))/S

      CMROOT = CMR/S
      CMAPEX = CMLE/S

      IF(CFBIN) THEN
        WRITE(7) ITER,CL,CD,CMROOT,CMAPEX
      ELSE
        WRITE(7,*) ITER,CL,CD,CMROOT,CMAPEX
      ENDIF
      ENDIF
      RETURN
```

```
END
```

```
C
C
C  SUBROUTINE DISSIP:  calculates second- and fourth-order
C  dissipation terms in Jameson's explicit, multi-stage Euler
C  scheme
C
      SUBROUTINE DISSIP
      include 'euler.inc'
      COMMON/SHCK/ JSM,ADDSMO
      DIMENSION EP2(0:JSIZ,0:KSIZ), EP4(0:JSIZ,0:KSIZ),
     &          UCON(5,JSIZ,KSIZ),PCON(JSIZ,KSIZ)
      REAL NUM, NUP, NUMM, NUPM, MM, MP
      IF(IMAX.GT.JSIZ .OR. JMAX.GT.JSIZ .OR. KMAX.GT.KSIZ)THEN
         WRITE(6,*) '[DISSIP]DIMENSIONS ARE IMPROPER'
         STOP
      ENDIF
C
C---  The value for the dissipation is found from the operator
C
C                      D  + D  + D  = D(u)
C                       X    Y    Z       ijk
C
C     as defined by Jameson, yielding second- and fourth-order
C     differences.  The operator in the X,Y,Z (I,J,K) coordinate
C     directions are determined in order.  Eriksson's treatment of the
C     smoothing at the solid wall and far-field boundaries is used.
C
C---  store upstream boundary condition in temporary array
C
      DO 52 J=1,JM
        DO 62 K=1,KM
          UCON(1,J,K) = U(1,0,J,K)
          UCON(2,J,K) = U(2,0,J,K)
          UCON(3,J,K) = U(3,0,J,K)
          UCON(4,J,K) = U(4,0,J,K)
          UCON(5,J,K) = U(5,0,J,K)
          PCON(J,K)   = P(0,J,K)
  62    CONTINUE
  52  CONTINUE

C
C---  set-up dummy points in the i-, j-, and k-directions.
C
      DO 10 I = ILE,ITE-1
        DO 20 K = 1,KM
        if(grname .ne. 'ni.gri')THEN
          U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
```

```
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
            P(I,JMAX+1,K) =   3.*P(I,JM,K) -   2.*P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
            P(I,JMAX,K) =   2.*P(I,JM,K) -   P(I,JM-1,K)
C
        ELSE
C
CVIS --- PERIODICITY CONDITION
          U(1,I,JMAX,K) = U(1,I,JM,K)
          U(2,I,JMAX,K) = U(2,I,JM,K)
          U(3,I,JMAX,K) =-U(3,I,JM,K)
          U(4,I,JMAX,K) = U(4,I,JM,K)
          U(5,I,JMAX,K) = U(5,I,JM,K)
            P(I,JMAX,K) =   P(I,JM,K)
          U(1,I,JMAX+1,K) = U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) =-U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = U(5,I,JM-1,K)
            P(I,JMAX+1,K) = P(I,JM-1,K)
        ENDIF
C
          U(1,I,0,K) = 2.*U(1,I,1,K) - U(1,I,2,K)
          U(2,I,0,K) = 2.*U(2,I,1,K) - U(2,I,2,K)
          U(3,I,0,K) = 2.*U(3,I,1,K) - U(3,I,2,K)
          U(4,I,0,K) = 2.*U(4,I,1,K) - U(4,I,2,K)
          U(5,I,0,K) = 2.*U(5,I,1,K) - U(5,I,2,K)
            P(I,0,K) =   2.*P(I,1,K) -   P(I,2,K)
C
          U(1,I,-1,K) = 3.*U(1,I,1,K) - 2.*U(1,I,2,K)
          U(2,I,-1,K) = 3.*U(2,I,1,K) - 2.*U(2,I,2,K)
          U(3,I,-1,K) = 3.*U(3,I,1,K) - 2.*U(3,I,2,K)
          U(4,I,-1,K) = 3.*U(4,I,1,K) - 2.*U(4,I,2,K)
          U(5,I,-1,K) = 3.*U(5,I,1,K) - 2.*U(5,I,2,K)
            P(I,-1,K) =   3.*P(I,1,K) -   2.*P(I,2,K)
C
          DT(I,JMAX,K) = DT(I,JM,K)
          DT(I,0,K) = DT(I,1,K)
   20 CONTINUE
```

```
   10   CONTINUE

        IF(MACH .GT. 1.) THEN
        DO 12 I=1,ILE-1
C       CVD$ NODEPCHK
C       CVD$ NOSYNC
          DO 11 K=1,KM
            U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
            U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
            U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
            U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
            U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
              P(I,JMAX+1,K) = 3.*  P(I,JM,K) - 2.*  P(I,JM-1,K)
C
            U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
            U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
            U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
            U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
            U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
              P(I,JMAX,K) = 2.*  P(I,JM,K) -   P(I,JM-1,K)
C
            U(1,I,0,K) = U(1,I,1,KMAX-K)
            U(2,I,0,K) = U(2,I,1,KMAX-K)
            U(3,I,0,K) = U(3,I,1,KMAX-K)
            U(4,I,0,K) = U(4,I,1,KMAX-K)
            U(5,I,0,K) = U(5,I,1,KMAX-K)
              P(I,0,K) =   P(I,1,KMAX-K)
C
            U(1,I,-1,K) = U(1,I,2,KMAX-K)
            U(2,I,-1,K) = U(2,I,2,KMAX-K)
            U(3,I,-1,K) = U(3,I,2,KMAX-K)
            U(4,I,-1,K) = U(4,I,2,KMAX-K)
            U(5,I,-1,K) = U(5,I,2,KMAX-K)
              P(I,-1,K) =   P(I,2,KMAX-K)
C
            DT(I,0,K) = DT(I,1,KMAX-K)
            DT(I,JMAX,K) = DT(I,JM,K)
   11     CONTINUE
   12   CONTINUE

        ELSE
        DO 140 I=1,ILE-1
C       CVD$ NODEPCHK
C       CVD$ NOSYNC
          DO 141 K=1,KM
          if(grname .ne. 'ni.gri')THEN
            U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
```

```
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
            P(I,JMAX+1,K) = 3.*  P(I,JM,K) - 2.*  P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
            P(I,JMAX,K) = 2.*  P(I,JM,K) -   P(I,JM-1,K)
        ELSE
C
CVIS --- PERIODICITY CONDITION
          U(1,I,JMAX,K) = U(1,I,JM,K)
          U(2,I,JMAX,K) = U(2,I,JM,K)
          U(3,I,JMAX,K) =-U(3,I,JM,K)
          U(4,I,JMAX,K) = U(4,I,JM,K)
          U(5,I,JMAX,K) = U(5,I,JM,K)
            P(I,JMAX,K) =   P(I,JM,K)
          U(1,I,JMAX+1,K) = U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) =-U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = U(5,I,JM-1,K)
            P(I,JMAX+1,K) = P(I,JM-1,K)
        ENDIF
C
          U(1,I,0,K) =  U(1,I,1,KMAX-K)
          U(2,I,0,K) =  U(2,I,1,KMAX-K)
          U(3,I,0,K) = -U(3,I,1,KMAX-K)
          U(4,I,0,K) =  U(4,I,1,KMAX-K)
          U(5,I,0,K) =  U(5,I,1,KMAX-K)
            P(I,0,K) =   P(I,1,KMAX-K)
C
          U(1,I,-1,K) =  U(1,I,2,KMAX-K)
          U(2,I,-1,K) =  U(2,I,2,KMAX-K)
          U(3,I,-1,K) = -U(3,I,2,KMAX-K)
          U(4,I,-1,K) =  U(4,I,2,KMAX-K)
          U(5,I,-1,K) =  U(5,I,2,KMAX-K)
            P(I,-1,K) =   P(I,2,KMAX-K)
C
          DT(I,0,K) = DT(I,1,KMAX-K)
          DT(I,JMAX,K) = DT(I,JM,K)
  141    CONTINUE
  140  CONTINUE
```

```
      ENDIF

      DO 14 I=ITE,IM
C     CVD$ NODEPCHK
C     CVD$ NOSYNC
        DO 13 K=1,KM
          U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
            P(I,JMAX+1,K) = 3.*  P(I,JM,K) - 2.*  P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
            P(I,JMAX,K) = 2.*  P(I,JM,K) -   P(I,JM-1,K)
C
          U(1,I,0,K) = U(1,I,1,KMAX-K)
          U(2,I,0,K) = U(2,I,1,KMAX-K)
          U(3,I,0,K) = U(3,I,1,KMAX-K)
          U(4,I,0,K) = U(4,I,1,KMAX-K)
          U(5,I,0,K) = U(5,I,1,KMAX-K)
            P(I,0,K) =   P(I,1,KMAX-K)
C
          U(1,I,-1,K) = U(1,I,2,KMAX-K)
          U(2,I,-1,K) = U(2,I,2,KMAX-K)
          U(3,I,-1,K) = U(3,I,2,KMAX-K)
          U(4,I,-1,K) = U(4,I,2,KMAX-K)
          U(5,I,-1,K) = U(5,I,2,KMAX-K)
            P(I,-1,K) =   P(I,2,KMAX-K)
C
          DT(I,0,K) = DT(I,1,KMAX-K)
          DT(I,JMAX,K) = DT(I,JM,K)
  13    CONTINUE
  14  CONTINUE
C
      IF(SYM) THEN
      DO 40 I = 1,IM
        DO 30 J = 1,JM
          U(1,I,J,KMAX) =  U(1,I,J,KMAX-1)
          U(2,I,J,KMAX) =  U(2,I,J,KMAX-1)
          U(3,I,J,KMAX) = -U(3,I,J,KMAX-1)
          U(4,I,J,KMAX) =  U(4,I,J,KMAX-1)
          U(5,I,J,KMAX) =  U(5,I,J,KMAX-1)
```

```
            P(I,J,KMAX) =    P(I,J,KMAX-1)
C
          U(1,I,J,KMAX+1) =  U(1,I,J,KMAX-2)
          U(2,I,J,KMAX+1) =  U(2,I,J,KMAX-2)
          U(3,I,J,KMAX+1) = -U(3,I,J,KMAX-2)
          U(4,I,J,KMAX+1) =  U(4,I,J,KMAX-2)
          U(5,I,J,KMAX+1) =  U(5,I,J,KMAX-2)
            P(I,J,KMAX+1) =    P(I,J,KMAX-2)
C
          U(1,I,J,0) =  U(1,I,J,1)
          U(2,I,J,0) =  U(2,I,J,1)
          U(3,I,J,0) = -U(3,I,J,1)
          U(4,I,J,0) =  U(4,I,J,1)
          U(5,I,J,0) =  U(5,I,J,1)
            P(I,J,0) =    P(I,J,1)
C
          U(1,I,J,-1) =  U(1,I,J,2)
          U(2,I,J,-1) =  U(2,I,J,2)
          U(3,I,J,-1) = -U(3,I,J,2)
          U(4,I,J,-1) =  U(4,I,J,2)
          U(5,I,J,-1) =  U(5,I,J,2)
            P(I,J,-1) =    P(I,J,2)
C
          DT(I,J,0) = DT(I,J,1)
          DT(I,J,KMAX) = DT(I,J,KM)
   30   CONTINUE
   40 CONTINUE

        ELSE
        DO 51 I=1,IM
          DO 61 J=1,JM
            U(1,I,J,KMAX+1) = U(1,I,J,2)
            U(2,I,J,KMAX+1) = U(2,I,J,2)
            U(3,I,J,KMAX+1) = U(3,I,J,2)
            U(4,I,J,KMAX+1) = U(4,I,J,2)
            U(5,I,J,KMAX+1) = U(5,I,J,2)
              P(I,J,KMAX+1) =   P(I,J,2)

            U(1,I,J,KMAX) = U(1,I,J,1)
            U(2,I,J,KMAX) = U(2,I,J,1)
            U(3,I,J,KMAX) = U(3,I,J,1)
            U(4,I,J,KMAX) = U(4,I,J,1)
            U(5,I,J,KMAX) = U(5,I,J,1)
              P(I,J,KMAX) =   P(I,J,1)

            U(1,I,J,0) = U(1,I,J,KM)
            U(2,I,J,0) = U(2,I,J,KM)
```

```
          U(3,I,J,0) = U(3,I,J,KM)
          U(4,I,J,0) = U(4,I,J,KM)
          U(5,I,J,0) = U(5,I,J,KM)
            P(I,J,0) =   P(I,J,KM)

          U(1,I,J,-1) = U(1,I,J,KM-1)
          U(2,I,J,-1) = U(2,I,J,KM-1)
          U(3,I,J,-1) = U(3,I,J,KM-1)
          U(4,I,J,-1) = U(4,I,J,KM-1)
          U(5,I,J,-1) = U(5,I,J,KM-1)
            P(I,J,-1) =   P(I,J,KM-1)

          DT(I,J,0) = DT(I,J,KM)
          DT(I,J,KMAX) = DT(I,J,1)
 61     CONTINUE
 51   CONTINUE
      ENDIF
C
      IF (GRNAME .ne. 'ni.gri')then
      DO 50 J = 1,JM
        DO 60 K = 1,KM
          U(1,IMAX+1,J,K) = 3.*U(1,IM,J,K) - 2.*U(1,IM-1,J,K)
          U(2,IMAX+1,J,K) = 3.*U(2,IM,J,K) - 2.*U(2,IM-1,J,K)
          U(3,IMAX+1,J,K) = 3.*U(3,IM,J,K) - 2.*U(3,IM-1,J,K)
          U(4,IMAX+1,J,K) = 3.*U(4,IM,J,K) - 2.*U(4,IM-1,J,K)
          U(5,IMAX+1,J,K) = 3.*U(5,IM,J,K) - 2.*U(5,IM-1,J,K)
            P(IMAX+1,J,K) =   3.*P(IM,J,K) -   2.*P(IM-1,J,K)
C
          U(1,IMAX,J,K) = 2.*U(1,IM,J,K) - U(1,IM-1,J,K)
          U(2,IMAX,J,K) = 2.*U(2,IM,J,K) - U(2,IM-1,J,K)
          U(3,IMAX,J,K) = 2.*U(3,IM,J,K) - U(3,IM-1,J,K)
          U(4,IMAX,J,K) = 2.*U(4,IM,J,K) - U(4,IM-1,J,K)
          U(5,IMAX,J,K) = 2.*U(5,IM,J,K) - U(5,IM-1,J,K)
            P(IMAX,J,K) =   2.*P(IM,J,K) -   P(IM-1,J,K)
C
          U(1,0,J,K) = 2.*U(1,1,J,K) - U(1,2,J,K)
          U(2,0,J,K) = 2.*U(2,1,J,K) - U(2,2,J,K)
          U(3,0,J,K) = 2.*U(3,1,J,K) - U(3,2,J,K)
          U(4,0,J,K) = 2.*U(4,1,J,K) - U(4,2,J,K)
          U(5,0,J,K) = 2.*U(5,1,J,K) - U(5,2,J,K)
            P(0,J,K) =   2.*P(1,J,K) -   P(2,J,K)
C
          U(1,-1,J,K) = 3.*U(1,1,J,K) - 2.*U(1,2,J,K)
          U(2,-1,J,K) = 3.*U(2,1,J,K) - 2.*U(2,2,J,K)
          U(3,-1,J,K) = 3.*U(3,1,J,K) - 2.*U(3,2,J,K)
          U(4,-1,J,K) = 3.*U(4,1,J,K) - 2.*U(4,2,J,K)
          U(5,-1,J,K) = 3.*U(5,1,J,K) - 2.*U(5,2,J,K)
```

```
            P(-1,J,K) =   3.*P(1,J,K) -   2.*P(2,J,K)
C
          DT(IMAX,J,K) = DT(IM,J,K)
          DT(0,J,K) = DT(1,J,K)
   60   CONTINUE
   50 CONTINUE
      ELSE
c
C---- Ni bump....
      DO 15 J = 1,JM
        DO 16 K = 1,KM
          U(1,IMAX+1,J,K) = U(1,IM,J,K)
          U(2,IMAX+1,J,K) = U(2,IM,J,K)
          U(3,IMAX+1,J,K) = U(3,IM,J,K)
          U(4,IMAX+1,J,K) = U(4,IM,J,K)
          U(5,IMAX+1,J,K) = U(5,IM,J,K)
            P(IMAX+1,J,K) = P(IM,J,K)
C
          U(1,IMAX,J,K) = U(1,IM,J,K)
          U(2,IMAX,J,K) = U(2,IM,J,K)
          U(3,IMAX,J,K) = U(3,IM,J,K)
          U(4,IMAX,J,K) = U(4,IM,J,K)
          U(5,IMAX,J,K) = U(5,IM,J,K)
            P(IMAX,J,K) =   P(IM,J,K)
C
c          U(1,0,J,K) = 2.*U(1,1,J,K) - U(1,2,J,K)
c          U(2,0,J,K) = 2.*U(2,1,J,K) - U(2,2,J,K)
c          U(3,0,J,K) = 2.*U(3,1,J,K) - U(3,2,J,K)
c          U(4,0,J,K) = 2.*U(4,1,J,K) - U(4,2,J,K)
c          U(5,0,J,K) = 2.*U(5,1,J,K) - U(5,2,J,K)
c            P(0,J,K) =   2.*P(1,J,K) -   P(2,J,K)
C
c          U(1,-1,J,K) = 3.*U(1,1,J,K) - 2.*U(1,2,J,K)
c          U(2,-1,J,K) = 3.*U(2,1,J,K) - 2.*U(2,2,J,K)
c          U(3,-1,J,K) = 3.*U(3,1,J,K) - 2.*U(3,2,J,K)
c          U(4,-1,J,K) = 3.*U(4,1,J,K) - 2.*U(4,2,J,K)
c          U(5,-1,J,K) = 3.*U(5,1,J,K) - 2.*U(5,2,J,K)
c            P(-1,J,K) =   3.*P(1,J,K) -   2.*P(2,J,K)
C
          DT(IMAX,J,K) = DT(IM,J,K)
          DT(0,J,K) = DT(1,J,K)
   16   CONTINUE
   15 CONTINUE
      ENDIF

      DO 100 J = 1,JM
        DO 110 I = 0,IM
```

```
C
C---  begin with the i-direction:  find the values of the
C     dissipation coefficients
C
          DO 70 K = 1,KM
            NUM = ABS((P(I+1,J,K) - 2.*P(I,J,K) + P(I-1,J,K))/
     &                (P(I+1,J,K) + 2.*P(I,J,K) + P(I-1,J,K)))
            NUP = ABS((P(I+2,J,K) - 2.*P(I+1,J,K) + P(I,J,K))/
     &                (P(I+2,J,K) + 2.*P(I+1,J,K) + P(I,J,K)))
            DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
            EP = KAP2 * 0.5*(NUM+NUP)
            EP2(I,K) = EP*DTH
            EP4(I,K) = MAX(0.,(KAP4 - EP))*DTH
   70     CONTINUE
  110   CONTINUE
          DO 770 K=1,KM
            EP2(0,K) = 0.
            EP2(IM,K)= 0.
  770     CONTINUE

c         IF(AOA .EQ. 0.) GO TO 73
c         IF(J.EQ.1) THEN
c           DO 71 I=ILE,ITE-1
c              KTIP = (KMAX+1)/2

c              K = KTIP+2
c              DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c              EP2(I,KTIP+2) = KAP2*DTH
c              EP4(I,KTIP+2) = 0.

c              K = KTIP + 1
c              DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c              EP2(I,KTIP+1) = KAP2*DTH
c              EP4(I,KTIP+1) = 0.

c              K = KTIP
c              DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c              EP2(I,KTIP) = KAP2*DTH
c              EP4(I,KTIP) = 0.

c              K = KTIP-1
c              DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c              EP2(I,KTIP-1) = KAP2*DTH
c              EP4(I,KTIP-1) = 0.

c              K = KTIP-2
c              DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
```

```
c             EP2(I,KTIP-2) = KAP2*DTH
c             EP4(I,KTIP-2) = 0.
c  71      CONTINUE
c         ENDIF

c         IF(J.EQ.2) THEN
c           DO 72 I=ILE,ITE-1
c             KTIP = (KMAX+1)/2

c             K = KTIP+2
c             DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c             EP2(I,KTIP+2) = KAP2*DTH
c             EP4(I,KTIP+2) = 0.

c             K = KTIP+1
c             DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c             EP2(I,KTIP+1) = KAP2*DTH
c             EP4(I,KTIP+1) = 0.

c             K = KTIP
c             DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c             EP2(I,KTIP) = KAP2*DTH
c             EP4(I,KTIP) = 0.
c    72     CONTINUE
c         ENDIF

cblvis          IF(J.EQ.1) THEN
c         IF(J.EQ.1 .AND. ITE .LT. IMAX) THEN
c         KTIP = (KMAX + 1)/2
c         DO 725 K=KTIP,KM
c           I=ITE-1
c           DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c           EP2(I,K) = KAP2*DTH
c           EP4(I,K) = 0.
c           I=ITE-2
c           DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c           EP2(I,K) = KAP2*DTH
c           EP4(I,K) = 0.
c           I=ITE
c           DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c           EP2(I,K) = KAP2*DTH
c           EP4(I,K) = 0.
c           I=ITE+1
c           DTH = CFL/(0.5*(DT(I+1,J,K) + DT(I,J,K)))
c           EP2(I,K) = KAP2*DTH
c           EP4(I,K) = 0.
c 725     CONTINUE
```

```
c        ENDIF

C
C---  calculate dissipation d(i,j,k)
C
c  73  CONTINUE

C
C---- For ni Bump zero at inflow
      IONE = 1
      IF (GRNAME .EQ. 'ni.gri') IONE = 3
c      DO 111 I=1,IM
      DO 111 I=IONE,IM
          DO 80 K = 1,KM
            DS1 = EP2(I,K)*(U(1,I+1,J,K) - U(1,I,J,K))
            DF1 = EP4(I,K)*(U(1,I+2,J,K) - 3.*(U(1,I+1,J,K) -
     &            U(1,I,J,K)) - U(1,I-1,J,K))
            DS2 = EP2(I,K)*(U(2,I+1,J,K) - U(2,I,J,K))
            DF2 = EP4(I,K)*(U(2,I+2,J,K) - 3.*(U(2,I+1,J,K) -
     &            U(2,I,J,K)) - U(2,I-1,J,K))
            DS3 = EP2(I,K)*(U(3,I+1,J,K) - U(3,I,J,K))
            DF3 = EP4(I,K)*(U(3,I+2,J,K) - 3.*(U(3,I+1,J,K) -
     &            U(3,I,J,K)) - U(3,I-1,J,K))
            DS4 = EP2(I,K)*(U(4,I+1,J,K) - U(4,I,J,K))
            DF4 = EP4(I,K)*(U(4,I+2,J,K) - 3.*(U(4,I+1,J,K) -
     &            U(4,I,J,K)) - U(4,I-1,J,K))
            DS5 = EP2(I,K)*(U(5,I+1,J,K) - U(5,I,J,K) + P(I+1,J,K) -
     &            P(I,J,K))
            DF5 = EP4(I,K)*(U(5,I+2,J,K) - 3.*(U(5,I+1,J,K) -
     &            U(5,I,J,K)) - U(5,I-1,J,K) + P(I+2,J,K) - 3.*
     &            (P(I+1,J,K) - P(I,J,K)) - P(I-1,J,K))
C
            D(1,I,J,K) = DS1 - DF1
            D(2,I,J,K) = DS2 - DF2
            D(3,I,J,K) = DS3 - DF3
            D(4,I,J,K) = DS4 - DF4
            D(5,I,J,K) = DS5 - DF5

   80     CONTINUE
C
          DO 90 K = 1,KM
            DS1M = EP2(I-1,K)*(U(1,I,J,K) - U(1,I-1,J,K))
            DF1M = EP4(I-1,K)*(U(1,I+1,J,K) - 3.*(U(1,I,J,K) -
     &             U(1,I-1,J,K)) - U(1,I-2,J,K))
            DS2M = EP2(I-1,K)*(U(2,I,J,K) - U(2,I-1,J,K))
            DF2M = EP4(I-1,K)*(U(2,I+1,J,K) - 3.*(U(2,I,J,K) -
     &             U(2,I-1,J,K)) - U(2,I-2,J,K))
```

```
            DS3M = EP2(I-1,K)*(U(3,I,J,K) - U(3,I-1,J,K))
            DF3M = EP4(I-1,K)*(U(3,I+1,J,K) - 3.*(U(3,I,J,K) -
     &             U(3,I-1,J,K)) - U(3,I-2,J,K))
            DS4M = EP2(I-1,K)*(U(4,I,J,K) - U(4,I-1,J,K))
            DF4M = EP4(I-1,K)*(U(4,I+1,J,K) - 3.*(U(4,I,J,K) -
     &             U(4,I-1,J,K)) - U(4,I-2,J,K))
            DS5M = EP2(I-1,K)*(U(5,I,J,K) - U(5,I-1,J,K) + P(I,J,K) -
     &             P(I-1,J,K))
            DF5M = EP4(I-1,K)*(U(5,I+1,J,K) - 3.*(U(5,I,J,K) -
     &             U(5,I-1,J,K)) - U(5,I-2,J,K) + P(I+1,J,K) - 3.*
     &             (P(I,J,K) - P(I-1,J,K)) - P(I-2,J,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M

   90     CONTINUE
  111 CONTINUE
  100 CONTINUE
C
C---  next do the j-direction:  find the values of the
C     dissipation coefficients
C
      DO 121 I=1,IM
        DO 122 J=0,JM
          DO 120 K = 1,KM
            NUM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &                (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
            NUP = ABS((P(I,J+2,K) - 2.*P(I,J+1,K) + P(I,J,K))/
     &                (P(I,J+2,K) + 2.*P(I,J+1,K) + P(I,J,K)))
            DTH = CFL/(0.5*(DT(I,J+1,K) + DT(I,J,K)))
            EP = KAP2 * 0.5*(NUM+NUP)
            EP2(J,K) = EP*DTH
            EP4(J,K) = MAX(0.,(KAP4 - EP))*DTH
  120     CONTINUE
  122   CONTINUE

          IF(I.GE.ILE .OR. I.LT.ITE) THEN
          DO 1200 K=1,KM
            EP2(0,K) = 0.
            EP2(JM,K) = 0.
 1200     CONTINUE
          ENDIF
C
C---- Put in additional (faired) smoothing in j-direction to stabilize shock
```

```
      DO 1201 K = 1,KM
         EP2(JSM-4,K) = EP2(JSM-4,K) + EP2(JSM-4,K)*.2*ADDSMO
         EP2(JSM-3,K) = EP2(JSM-3,K) + EP2(JSM-3,K)*.4*ADDSMO
         EP2(JSM-2,K) = EP2(JSM-2,K) + EP2(JSM-2,K)*.6*ADDSMO
         EP2(JSM-1,K) = EP2(JSM-1,K) + EP2(JSM-1,K)*.8*ADDSMO
         DO 1201 J = JSM,JM
         EP2(J,K) = EP2(J,K) + EP2(J,K)*ADDSMO
 1201    CONTINUE
c           IF(AOA .EQ. 0.) GO TO 74

C           IF(I.LT.ILE .OR. I.GT.ITE-1) GO TO 74
C           KTIP = (KMAX+1)/2

C           K = KTIP-2
C           DTH = CFL/(0.5*(DT(I,2,K) + DT(I,1,K)))
C           EP2(1,KTIP-2) = KAP2*DTH
C           EP4(1,KTIP-2) = 0.

C           K = KTIP-1
C           DTH = CFL/(0.5*(DT(I,2,K) + DT(I,1,K)))
C           EP2(1,KTIP-1) = KAP2*DTH
C           EP4(1,KTIP-1) = 0.

C           K = KTIP
C           DTH = CFL/(0.5*(DT(I,2,K) + DT(I,1,K)))
C           EP2(1,KTIP) = KAP2*DTH
C           EP4(1,KTIP) = 0.

C           K = KTIP+2
C           DTH = CFL/(0.5*(DT(I,2,K) + DT(I,1,K)))
C           EP2(1,KTIP+2) = KAP2*DTH
C           EP4(1,KTIP+2) = 0.

C           K = KTIP+1
C           DTH = CFL/(0.5*(DT(I,2,K) + DT(I,1,K)))
C           EP2(1,KTIP+1) = KAP2*DTH
C           EP4(1,KTIP+1) = 0.

C           K = KTIP
C           DTH = CFL/(0.5*(DT(I,3,K) + DT(I,2,K)))
C           EP2(2,KTIP) = KAP2*DTH
C           EP4(2,KTIP) = 0.

C           K = KTIP+2
C           DTH = CFL/(0.5*(DT(I,3,K) + DT(I,2,K)))
C           EP2(2,KTIP+2) = KAP2*DTH
C           EP4(2,KTIP+2) = 0.
```

```
C            K = KTIP+1
C            DTH = CFL/(0.5*(DT(I,3,K) + DT(I,2,K)))
C            EP2(2,KTIP+1) = KAP2*DTH
C            EP4(2,KTIP+1) = 0.

C
C---  calculate dissipation d(i,j,k)
C
  74    CONTINUE
C
CVIS*** Erricson's treatment seems to add too much smoothing near lower
Cvis*BL boundary, so set smoothing there to zero
        JONE = 1
        IF (RE .GT. 1.)THEN
        JONE = 3
        ENDIF
C        DO 132 J=1,JM
        DO 132 J=JONE,JM
          DO 130 K = 1,KM
            DS1 = EP2(J,K)*(U(1,I,J+1,K) - U(1,I,J,K))
            DF1 = EP4(J,K)*(U(1,I,J+2,K) - 3.*(U(1,I,J+1,K) -
     &            U(1,I,J,K)) - U(1,I,J-1,K))
            DS2 = EP2(J,K)*(U(2,I,J+1,K) - U(2,I,J,K))
            DF2 = EP4(J,K)*(U(2,I,J+2,K) - 3.*(U(2,I,J+1,K) -
     &            U(2,I,J,K)) - U(2,I,J-1,K))
            DS3 = EP2(J,K)*(U(3,I,J+1,K) - U(3,I,J,K))
            DF3 = EP4(J,K)*(U(3,I,J+2,K) - 3.*(U(3,I,J+1,K) -
     &            U(3,I,J,K)) - U(3,I,J-1,K))
            DS4 = EP2(J,K)*(U(4,I,J+1,K) - U(4,I,J,K))
            DF4 = EP4(J,K)*(U(4,I,J+2,K) - 3.*(U(4,I,J+1,K) -
     &            U(4,I,J,K)) - U(4,I,J-1,K))
            DS5 = EP2(J,K)*(U(5,I,J+1,K) - U(5,I,J,K) + P(I,J+1,K) -
     &            P(I,J,K))
            DF5 = EP4(J,K)*(U(5,I,J+2,K) - 3.*(U(5,I,J+1,K) -
     &            U(5,I,J,K)) - U(5,I,J-1,K) + P(I,J+2,K) - 3.*
     &            (P(I,J+1,K) - P(I,J,K)) - P(I,J-1,K))
C
           D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
           D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
           D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
           D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
           D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  130     CONTINUE
C
          DO 150 K = 1,KM
            DS1M = EP2(J-1,K)*(U(1,I,J,K) - U(1,I,J-1,K))
```

```
            DF1M = EP4(J-1,K)*(U(1,I,J+1,K) - 3.*(U(1,I,J,K) -
     &             U(1,I,J-1,K)) - U(1,I,J-2,K))
            DS2M = EP2(J-1,K)*(U(2,I,J,K) - U(2,I,J-1,K))
            DF2M = EP4(J-1,K)*(U(2,I,J+1,K) - 3.*(U(2,I,J,K) -
     &             U(2,I,J-1,K)) - U(2,I,J-2,K))
            DS3M = EP2(J-1,K)*(U(3,I,J,K) - U(3,I,J-1,K))
            DF3M = EP4(J-1,K)*(U(3,I,J+1,K) - 3.*(U(3,I,J,K) -
     &             U(3,I,J-1,K)) - U(3,I,J-2,K))
            DS4M = EP2(J-1,K)*(U(4,I,J,K) - U(4,I,J-1,K))
            DF4M = EP4(J-1,K)*(U(4,I,J+1,K) - 3.*(U(4,I,J,K) -
     &             U(4,I,J-1,K)) - U(4,I,J-2,K))
            DS5M = EP2(J-1,K)*(U(5,I,J,K) - U(5,I,J-1,K) + P(I,J,K) -
     &             P(I,J-1,K))
            DF5M = EP4(J-1,K)*(U(5,I,J+1,K) - 3.*(U(5,I,J,K) -
     &             U(5,I,J-1,K)) - U(5,I,J-2,K) + P(I,J+1,K) - 3.*
     &             (P(I,J,K) - P(I,J-1,K)) - P(I,J-2,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M

  150     CONTINUE
  132   CONTINUE
  121  CONTINUE
C
       DO 230 I = 1,IM
         DO 220 J = 1,JM
C
C---   next do the k-direction:  find the values of the
C      dissipation coefficients
C
           DO 180 K = 0,KM
             NUM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                 (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
             NUP = ABS((P(I,J,K+2) - 2.*P(I,J,K+1) + P(I,J,K))/
     &                 (P(I,J,K+2) + 2.*P(I,J,K+1) + P(I,J,K)))
             DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
             EPP = KAP2 * 0.5*(NUM+NUP)
             EP2(J,K) = EPP*DTH
             EP4(J,K) = MAX(0.,(KAP4 - EPP))*DTH
  180      CONTINUE

C            IF(AOA .EQ. 0.) GO TO 75
C            IF(I.LT.ILE .OR. I.GT.ITE-1) GO TO 75
C            KTIP = (KMAX+1)/2
```

```
C            IF(J.EQ.1) THEN

C              K=KTIP-2
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(1,KTIP-2) = KAP2*DTH
C              EP4(1,KTIP-2) = 0.

C              K=KTIP-1
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(1,KTIP-1) = KAP2*DTH
C              EP4(1,KTIP-1) = 0.

C              K=KTIP
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(1,KTIP) = KAP2*DTH
C              EP4(1,KTIP) = 0.

C              K=KTIP +1
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(1,KTIP+1) = KAP2*DTH
C              EP4(1,KTIP+1) = 0.

C              K=KTIP+2
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(1,KTIP+2) = KAP2*DTH
C              EP4(1,KTIP+2) = 0.
C            ENDIF

C            IF(J.EQ.2) THEN
C              K=KTIP
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(2,KTIP) = KAP2*DTH
C              EP4(2,KTIP) = 0.

C              K=KTIP+1
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(2,KTIP+1) = KAP2*DTH
C              EP4(2,KTIP+1) = 0.

C              K=KTIP+2
C              DTH = CFL/(0.5*(DT(I,J,K+1) + DT(I,J,K)))
C              EP2(2,KTIP+2) = KAP2*DTH
C              EP4(2,KTIP+2) = 0.
C            ENDIF

C
C---  calculate dissipation d(i,j,k)
```

```
C
 75      DO 190 K = 1,KM
           DS1 = EP2(J,K)*(U(1,I,J,K+1) - U(1,I,J,K))
           DF1 = EP4(J,K)*(U(1,I,J,K+2) - 3.*(U(1,I,J,K+1) -
     &           U(1,I,J,K)) - U(1,I,J,K-1))
           DS2 = EP2(J,K)*(U(2,I,J,K+1) - U(2,I,J,K))
           DF2 = EP4(J,K)*(U(2,I,J,K+2) - 3.*(U(2,I,J,K+1) -
     &           U(2,I,J,K)) - U(2,I,J,K-1))
           DS3 = EP2(J,K)*(U(3,I,J,K+1) - U(3,I,J,K))
           DF3 = EP4(J,K)*(U(3,I,J,K+2) - 3.*(U(3,I,J,K+1) -
     &           U(3,I,J,K)) - U(3,I,J,K-1))
           DS4 = EP2(J,K)*(U(4,I,J,K+1) - U(4,I,J,K))
           DF4 = EP4(J,K)*(U(4,I,J,K+2) - 3.*(U(4,I,J,K+1) -
     &           U(4,I,J,K)) - U(4,I,J,K-1))
           DS5 = EP2(J,K)*(U(5,I,J,K+1) - U(5,I,J,K) + P(I,J,K+1) -
     &           P(I,J,K))
           DF5 = EP4(J,K)*(U(5,I,J,K+2) - 3.*(U(5,I,J,K+1) -
     &           U(5,I,J,K)) - U(5,I,J,K-1) + P(I,J,K+2) - 3.*
     &           (P(I,J,K+1) - P(I,J,K)) - P(I,J,K-1))
C
           D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
           D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
           D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
           D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
           D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
 190     CONTINUE
C
         DO 210 K = 1,KM
           DS1M = EP2(J,K-1)*(U(1,I,J,K) - U(1,I,J,K-1))
           DF1M = EP4(J,K-1)*(U(1,I,J,K+1) - 3.*(U(1,I,J,K) -
     &            U(1,I,J,K-1)) - U(1,I,J,K-2))
           DS2M = EP2(J,K-1)*(U(2,I,J,K) - U(2,I,J,K-1))
           DF2M = EP4(J,K-1)*(U(2,I,J,K+1) - 3.*(U(2,I,J,K) -
     &            U(2,I,J,K-1)) - U(2,I,J,K-2))
           DS3M = EP2(J,K-1)*(U(3,I,J,K) - U(3,I,J,K-1))
           DF3M = EP4(J,K-1)*(U(3,I,J,K+1) - 3.*(U(3,I,J,K) -
     &            U(3,I,J,K-1)) - U(3,I,J,K-2))
           DS4M = EP2(J,K-1)*(U(4,I,J,K) - U(4,I,J,K-1))
           DF4M = EP4(J,K-1)*(U(4,I,J,K+1) - 3.*(U(4,I,J,K) -
     &            U(4,I,J,K-1)) - U(4,I,J,K-2))
           DS5M = EP2(J,K-1)*(U(5,I,J,K) - U(5,I,J,K-1) + P(I,J,K) -
     &            P(I,J,K-1))
           DF5M = EP4(J,K-1)*(U(5,I,J,K+1) - 3.*(U(5,I,J,K) -
     &            U(5,I,J,K-1)) - U(5,I,J,K-2) + P(I,J,K+1) - 3.*
     &            (P(I,J,K) - P(I,J,K-1)) - P(I,J,K-2))
C
           D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
```

```
          D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
          D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
          D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
          D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M

210       CONTINUE
220     CONTINUE
230   CONTINUE

      DO 53 J=1,JM
        DO 63 K=1,KM
          U(1,0,J,K) = UCON(1,J,K)
          U(2,0,J,K) = UCON(2,J,K)
          U(3,0,J,K) = UCON(3,J,K)
          U(4,0,J,K) = UCON(4,J,K)
          U(5,0,J,K) = UCON(5,J,K)
            P(0,J,K) = PCON(J,K)
63      CONTINUE
53    CONTINUE

      RETURN
      END
```

```
C
C
C  SUBROUTINE DTSIZE:  calculates the value of the local time step
C  size based on the local CFL number restriction
C
      SUBROUTINE DTSIZE
      include 'euler.inc'
      INCLUDE 'blvisc.inc'
      REAL DTVISC
      COMMON/DTE/DTEXP(ISIZ,JSIZ,KSIZ)
      COMMON/AEXP/AXEXP(ISIZ,JSIZ,KSIZ),AYEXP(ISIZ,JSIZ,KSIZ),
     &            AZEXP(ISIZ,JSIZ,KSIZ)
C
C---- Check for negative density
      DO 5 I = 1,IM
      DO 5 J = 1,JM
      DO 5 K = 1,KM
         IF (P(I,J,K) .LT. 0.) GO TO 71
 5       IF (U(1,I,J,K) .LT. 0.) GO TO 70
C
C---- Time step...
      DO 10 I = 1,IM
        DO 10 J = 1,JM
          DO 10 K = 1,KM
            DTVISC = 0.
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
C
            C = SQRT(GAM*P(I,J,K)/RHO)
C
C---- (Note:  Semi-implicit stuff is implemented in AMEAN.F)
            AX = AXM(I,J,K)
            AY = AYM(I,J,K)
            AZ = AZM(I,J,K)
            AR = SQRT(AX**2 + AY**2 + AZ**2)
Cdebug for debuggin purposes only
cdb         IF(GRNAME .EQ. 'ni.gri')AR = SQRT(AX**2 + AY**2)
cdb           IF(GRNAME .NE. 'ni.gri') write(6,*)
cdb     &  'dtsize: Remove Ni Bump Fudege!'
cdb           IF(GRNAME .NE. 'ni.gri') stop
Cdebug
            UA = ABS(U1*AX)
            VA = ABS(V1*AY)
            WA = ABS(W1*AZ)
            CA = C*AR
```

```
CViscous
C---- Viscous restriction is eliminated by S-I treatment
      IF (.NOT. SEMIIMP) THEN
         DNN = MAX(1E-16,DNORM(I,J,K))
         DTVISC = 4.*(MUL(I,J,K)+MUT(I,J,K))*NAREA(I,J,K)/DNN
      ENDIF
C
C---- Time step
      DT(I,J,K) = CFL/(UA + VA + WA + CA + DTVISC)
Cviscous
C
C---- Calculate explicit dt's for lhs smoothing if semiimplicit
      IF (SEMIIMP) THEN
            AX = AXEXP(I,J,K)
            AY = AYEXP(I,J,K)
            AZ = AZEXP(I,J,K)
            AR = SQRT(AX**2 + AY**2 + AZ**2)
Cdebug for debuggin purposes only
cdb         IF(GRNAME .EQ. 'ni.gri')AR = SQRT(AX**2 + AY**2)
cdb            IF(GRNAME .NE. 'ni.gri') write(6,*)
cdb     &  'dtsize: Remove Ni Bump Fudege!'
cdb            IF(GRNAME .NE. 'ni.gri') stop
Cdebug
            UA = ABS(U1*AX)
            VA = ABS(V1*AY)
            WA = ABS(W1*AZ)
            CA = C*AR
         DNN = MAX(1E-16,DNORM(I,J,K))
         DTVISC = 4.*(MUL(I,J,K)+MUT(I,J,K))*NAREA(I,J,K)/DNN
         DTEXP(I,J,K) = CFL/(UA + VA + WA + CA + DTVISC)
      ENDIF
   10 CONTINUE
C
C---- Set DT in each row equal to minimum FOR SEMI-IMPLICIT OPERATION
      IF (.NOT. SEMIIMP) GOTO 60
      DO 50 I = 1, IM
        DTMIN = 10E13
        DO 55 J = 1,JM
        DO 55 K = 1,KM
           DT(I,J,K) = DT(I,J,K)*VOL(I,J,K)
 55        DTMIN = MIN(DTMIN,DT(I,J,K))
        DO 56 J = 1,JM
        DO 56 K = 1,KM
           DT(I,J,K) = DTMIN
 56        DT(I,J,K) = DT(I,J,K)/VOL(I,J,K)

 50   CONTINUE
```

```
 60   CONTINUE
      GOTO 99
 70   WRITE(6,*) 'DTSIZE:  Negative density at i,j,k=',i,j,k
      STOP
 71   WRITE(6,*) 'DTSIZE:  Negative pressure at i,j,k=',i,j,k
      STOP
 99   CONTINUE
C
      RETURN
      END

C
C---This would work if FACEX,Y,Z were defined at the boundaries
c           AX = MAX(ABS(FACEX(1,I,J,K)),ABS(FACEX(1,I-1,J,K))) +
c     &           MAX(ABS(FACEX(3,I,J,K)),ABS(FACEX(3,I,J,K-1)))
c           AY = MAX(ABS(FACEY(1,I,J,K)),ABS(FACEY(1,I-1,J,K))) +
c     &           MAX(ABS(FACEY(3,I,J,K)),ABS(FACEY(3,I,J,K-1)))
c           AZ = MAX(ABS(FACEZ(1,I,J,K)),ABS(FACEZ(1,I-1,J,K))) +
c     &           MAX(ABS(FACEZ(3,I,J,K)),ABS(FACEZ(3,I,J,K-1)))
```

```
C
C
C  SUBROUTINE FLUX:  determine the fluxes across each finite volume cell
C  face, and the residual R for each cell.  This is done by averaging
C  the flux vectors between neighboring cells
C
C NOTE:  The form of the flux balance used by Roberts/Goodsell has
C        been replaced with a modified version  that is second order accurate
C        on Cartesian grids (R/G is not).  To recover the R/G version
C        simply comment out the two lines:  QA=.5(QA+QAP) , QAP=QA
C        occuring at several locations in the subroutine. 8/1/88
C
      SUBROUTINE FLUX
      include 'euler.inc'
      include 'blvisc.inc'

C
C---  set-up dummy points at kmax
C
      IF(SYM) GO TO 5
      DO 20 I = 1,IM
        DO 10 J = 1,JM
          U(1,I,J,KMAX) = U(1,I,J,1)
          U(2,I,J,KMAX) = U(2,I,J,1)
          U(3,I,J,KMAX) = U(3,I,J,1)
          U(4,I,J,KMAX) = U(4,I,J,1)
          U(5,I,J,KMAX) = U(5,I,J,1)
            P(I,J,KMAX) =   P(I,J,1)
   10   CONTINUE
   20 CONTINUE

C
C---  first, initialize the residual field
C
 5    DO 70 I = 1,IMAX
        DO 60 J = 1,JMAX
          DO 50 K = 1,KMAX
            R(1,I,J,K) = 0.
            R(2,I,J,K) = 0.
            R(3,I,J,K) = 0.
            R(4,I,J,K) = 0.
            R(5,I,J,K) = 0.
   50     CONTINUE
   60   CONTINUE
   70 CONTINUE

      IF(SYM) THEN
```

```
C
C---  determine the residual at the symmetry plane cells
C
      DO 90 I = 1,IM
        DO 80 J = 1,JM
          R(3,I,J,1) = WALLB(I,J)*PRESYMB(I,J)
          R(3,I,J,KM) = WALLT(I,J)*PRESYMT(I,J)
   80   CONTINUE
   90 CONTINUE
      ENDIF

C
C---  add contribution due to pressures at the wing surface
C
      DO 110 I = ILE,ITE-1
         DO 100 K = 1,KM
          R(2,I,1,K) = R(2,I,1,K) + FACEX(2,I,0,K)*PRESWG(I,K)
          R(3,I,1,K) = R(3,I,1,K) + FACEY(2,I,0,K)*PRESWG(I,K)
          R(4,I,1,K) = R(4,I,1,K) + FACEZ(2,I,0,K)*PRESWG(I,K)
  100   CONTINUE
  110 CONTINUE
C
C---  set up dummy points at j=0 for upstream and downstream pts
C
      DO 21 I=1,ILE-1
        DO 11 K=1,KM
          U(1,I,0,K) =  U(1,I,1,KMAX-K)
          U(2,I,0,K) =  U(2,I,1,KMAX-K)
          U(3,I,0,K) = -U(3,I,1,KMAX-K)
          U(4,I,0,K) =  U(4,I,1,KMAX-K)
          U(5,I,0,K) =  U(5,I,1,KMAX-K)
          P(I,0,K)   =  P(I,1,KMAX-K)
  11    CONTINUE
  21  CONTINUE

      DO 22 I=ITE,IM
        DO 12 K=1,KM
          U(1,I,0,K) = U(1,I,1,KMAX-K)
          U(2,I,0,K) = U(2,I,1,KMAX-K)
          U(3,I,0,K) = U(3,I,1,KMAX-K)
          U(4,I,0,K) = U(4,I,1,KMAX-K)
          U(5,I,0,K) = U(5,I,1,KMAX-K)
          P(I,0,K)   = P(I,1,KMAX-K)
  12    CONTINUE
  22  CONTINUE
C
C---  do the flux across the j=0-face downstream and upstream
```

```
C
          J=0
          DO 132 I = 1,ILE-1
            DO 131 K=1,KM
C
C---  find the flux vector for cells (i,0,k) and (i,1,k)
C
            QA = (U(2,I,J,K)*FACEX(2,I,J,K) + U(3,I,J,K)*FACEY(2,I,J,K)
     &       + U(4,I,J,K)*FACEZ(2,I,J,K))/U(1,I,J,K)
            QAP = (U(2,I,J+1,K)*FACEX(2,I,J,K) + U(3,I,J+1,K)*
     &       FACEY(2,I,J,K) + U(4,I,J+1,K)*FACEZ(2,I,J,K))/U(1,I,J+1,K)
            QA = .5*(QA+QAP)
            QAP = QA
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(2,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(2,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            H1P = U(1,I,J+1,K)*QAP
            H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEX(2,I,J,K)
            H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEY(2,I,J,K)
            H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEZ(2,I,J,K)
            H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,0,k) and (i,1,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---   add flux to cell (i,1,k)
C
            R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
            R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
            R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
            R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
            R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
C
  131     CONTINUE
  132   CONTINUE

          DO 134 I = ITE,IM
            DO 133 K=1,KM
C
```

```
C---  find the flux vector for cells (i,0,k) and (i,1,k)
C
            QA = (U(2,I,J,K)*FACEX(2,I,J,K) + U(3,I,J,K)*FACEY(2,I,J,K)
     &       + U(4,I,J,K)*FACEZ(2,I,J,K))/U(1,I,J,K)
            QAP = (U(2,I,J+1,K)*FACEX(2,I,J,K) + U(3,I,J+1,K)*
     &       FACEY(2,I,J,K) + U(4,I,J+1,K)*FACEZ(2,I,J,K))/U(1,I,J+1,K)
            QA = .5*(QA+QAP)
            QAP = QA
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(2,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(2,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            H1P = U(1,I,J+1,K)*QAP
            H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEX(2,I,J,K)
            H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEY(2,I,J,K)
            H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEZ(2,I,J,K)
            H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,0,k) and (i,1,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  add flux to cell (i,1,k)
C
            R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
            R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
            R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
            R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
            R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
C
  133     CONTINUE
  134   CONTINUE

      I = 0
      DO 161 J = 1,JM
        DO 151 K = 1,KM
C
C---  do the flux across the i=0-face
C---  find the flux vector for cells (0,j,k) and (1,j,k)
C
          QA = (U(2,I,J,K)*FRONT(J,K))/U(1,I,J,K)
```

```
          QAP = (U(2,I+1,J,K)*FRONT(J,K))/U(1,I+1,J,K)
          QA = .5*(QA+QAP)
          QAP = QA
          H1 = U(1,I,J,K)*QA
          H2 = U(2,I,J,K)*QA + P(I,J,K)*FRONT(J,K)
          H3 = U(3,I,J,K)*QA
          H4 = U(4,I,J,K)*QA
          H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
          H1P = U(1,I+1,J,K)*QAP
          H2P = U(2,I+1,J,K)*QAP + P(I+1,J,K)*FRONT(J,K)
          H3P = U(3,I+1,J,K)*QAP
          H4P = U(4,I+1,J,K)*QAP
          H5P = (U(5,I+1,J,K) + P(I+1,J,K))*QAP
C
C---  find the average flux vector between cells (0,j,k) and (1,j,k)
C
          DR1 = .5*(H1 + H1P)
          DR2 = .5*(H2 + H2P)
          DR3 = .5*(H3 + H3P)
          DR4 = .5*(H4 + H4P)
          DR5 = .5*(H5 + H5P)
C
C---  add flux to cell (1,j,k)
C
          R(1,I+1,J,K) = R(1,I+1,J,K) + DR1
          R(2,I+1,J,K) = R(2,I+1,J,K) + DR2
          R(3,I+1,J,K) = R(3,I+1,J,K) + DR3
          R(4,I+1,J,K) = R(4,I+1,J,K) + DR4
          R(5,I+1,J,K) = R(5,I+1,J,K) + DR5
C
  151   CONTINUE
  161 CONTINUE
C
      DO 120 I = 1,IM
        DO 160 J = 1,JM
C
C---  first do the flux across the i-face
C
          DO 150 K = 1,KM
C
C---  find the flux vector for cells (i,j,k) and (i+1,j,k)
C
            QA = (U(2,I,J,K)*FACEX(1,I,J,K) + U(3,I,J,K)*FACEY(1,I,J,K)
     &       + U(4,I,J,K)*FACEZ(1,I,J,K))/U(1,I,J,K)
            QAP = (U(2,I+1,J,K)*FACEX(1,I,J,K) + U(3,I+1,J,K)*
     &       FACEY(1,I,J,K) + U(4,I+1,J,K)*FACEZ(1,I,J,K))/U(1,I+1,J,K)
```

```
            QA = .5*(QA+QAP)
            QAP = QA
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(1,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(1,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(1,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            H1P = U(1,I+1,J,K)*QAP
            H2P = U(2,I+1,J,K)*QAP + P(I+1,J,K)*FACEX(1,I,J,K)
            H3P = U(3,I+1,J,K)*QAP + P(I+1,J,K)*FACEY(1,I,J,K)
            H4P = U(4,I+1,J,K)*QAP + P(I+1,J,K)*FACEZ(1,I,J,K)
            H5P = (U(5,I+1,J,K) + P(I+1,J,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i+1,j,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i+1,j,k)
C
            R(1,I+1,J,K) = R(1,I+1,J,K) + DR1
            R(2,I+1,J,K) = R(2,I+1,J,K) + DR2
            R(3,I+1,J,K) = R(3,I+1,J,K) + DR3
            R(4,I+1,J,K) = R(4,I+1,J,K) + DR4
            R(5,I+1,J,K) = R(5,I+1,J,K) + DR5
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
  150     CONTINUE
C
C---  next do the flux across the j-face
C
          DO 130 K = 1,KM
C
C---  find the flux vector for cells (i,j,k) and (i,j+1,k)
C
            QA = (U(2,I,J,K)*FACEX(2,I,J,K) + U(3,I,J,K)*FACEY(2,I,J,K)
     &       + U(4,I,J,K)*FACEZ(2,I,J,K))/U(1,I,J,K)
```

```
            QAP = (U(2,I,J+1,K)*FACEX(2,I,J,K) + U(3,I,J+1,K)*
     &       FACEY(2,I,J,K) + U(4,I,J+1,K)*FACEZ(2,I,J,K))/U(1,I,J+1,K)
            QA = .5*(QA+QAP)
            QAP = QA
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(2,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(2,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            H1P = U(1,I,J+1,K)*QAP
            H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEX(2,I,J,K)
            H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEY(2,I,J,K)
            H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEZ(2,I,J,K)
            H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j+1,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C            if(i.eq.30 .and. j.eq.1 .and. k.eq.1) then
C       write(6,*) 'dr2,h2,h2p',dr2*4,h2*4,h2p*4
C       write(6,*) 'pdy,uq=',
C     & .5*(p(i,j,k)+p(i,j+1,k))*facex(2,i,j,k)*4,
C     & .5*(u(2,i,j,k)*qa+u(2,i,j+1,k)*qap)*4
C       write(6,*) 'u2*qa,*qap=',u(2,i,j,k)*qa*4,u(2,i,j+1,k)*qap*4
C       write(6,*) 'qa,qap,p(j),p(jp)',qa*2,qap*2,
C     & p(i,j,k)*4,p(i,j+1,k)*4
C       endif
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j+1,k)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
            R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
            R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
            R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
            R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
```

```
C
  130     CONTINUE
  160   CONTINUE
  120 CONTINUE
C
C---  finally do the flux across the k-face
c---  for an asymmetric case you will want to loop from k=1,km
C
      KLOOP = KM
      IF(SYM) KLOOP = KM-1
      DO 152 K=1,KLOOP
        DO 140 I=1,IM
          DO 162 J=1,JM
C
C---  find the flux vector for cells (i,j,k) and (i,j,k+1)
C
            QA = (U(2,I,J,K)*FACEX(3,I,J,K) + U(3,I,J,K)*FACEY(3,I,J,K)
     &       + U(4,I,J,K)*FACEZ(3,I,J,K))/U(1,I,J,K)
            QAP = (U(2,I,J,K+1)*FACEX(3,I,J,K) + U(3,I,J,K+1)*
     &       FACEY(3,I,J,K) + U(4,I,J,K+1)*FACEZ(3,I,J,K))/U(1,I,J,K+1)
            QA = .5*(QA+QAP)
            QAP = QA
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(3,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(3,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(3,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            H1P = U(1,I,J,K+1)*QAP
            H2P = U(2,I,J,K+1)*QAP + P(I,J,K+1)*FACEX(3,I,J,K)
            H3P = U(3,I,J,K+1)*QAP + P(I,J,K+1)*FACEY(3,I,J,K)
            H4P = U(4,I,J,K+1)*QAP + P(I,J,K+1)*FACEZ(3,I,J,K)
            H5P = (U(5,I,J,K+1) + P(I,J,K+1))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j,k+1)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
```

```
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J,K+1) = R(1,I,J,K+1) + DR1
            R(2,I,J,K+1) = R(2,I,J,K+1) + DR2
            R(3,I,J,K+1) = R(3,I,J,K+1) + DR3
            R(4,I,J,K+1) = R(4,I,J,K+1) + DR4
            R(5,I,J,K+1) = R(5,I,J,K+1) + DR5

  162     CONTINUE
  140   CONTINUE
  152 CONTINUE

      IF(.NOT. SYM) THEN
C
C---  fix-up the residuals across the coordinate cut in the
C     k-direction (spanwise)
C
      DO 180 I = 1,IM
C     CVD$ NODEPCHK
        DO 170 J = 1,JM
          R(1,I,J,1) = R(1,I,J,1) + R(1,I,J,KMAX)
          R(2,I,J,1) = R(2,I,J,1) + R(2,I,J,KMAX)
          R(3,I,J,1) = R(3,I,J,1) + R(3,I,J,KMAX)
          R(4,I,J,1) = R(4,I,J,1) + R(4,I,J,KMAX)
          R(5,I,J,1) = R(5,I,J,1) + R(5,I,J,KMAX)
  170   CONTINUE
  180 CONTINUE
      ENDIF
C
C---- Add in viscous fluxes on j faces
CViscous
      IF (REYNUM .GT. 0.) CALL BLVISC
CViscous
C

      RETURN
      END
```

```
C
C
C  SUBROUTINE NORMAL:  This subroutine calculates the cell-face
C  normals of the finite volume cells
C
      SUBROUTINE NORMAL
      include 'euler.inc'
C
C---  determine the outward pointing normal on each cell face.
C     The name of the variable (FACEX, FACEY, FACEZ) identifies the
C     physical space component of the normal, and the first index of
C     the variable (1, 2, 3) identifies the computational space com-
C     ponent of the normal.  For example, FACEX(2,I,J,K) is the
C     x-component of the normal pointing in the j-direction out of cell
C     (I,J,K).  (You'll figure it out, don't worry.)
C
C********** NOTE: THIS SUBROUTINE HAS BEEN MODIFIED SO THAT THE
C**********       COMP. COORDINATES FORM A RIGHT-HANDED SYSTEM
C
C---  first, the I - face
C
      DO 30 I = 1,IM
        DO 20 J = 1,JM
          DO 10 K = 1,KM
            X1 = X(I+1,J+1,K) - X(I+1,J,K+1)
            Y1 = Y(I+1,J+1,K) - Y(I+1,J,K+1)
            Z1 = Z(I+1,J+1,K) - Z(I+1,J,K+1)
            X2 = X(I+1,J+1,K+1) - X(I+1,J,K)
            Y2 = Y(I+1,J+1,K+1) - Y(I+1,J,K)
            Z2 = Z(I+1,J+1,K+1) - Z(I+1,J,K)
C
            FACEX(1,I,J,K) = (Y1*Z2 - Z1*Y2)*0.5
            FACEY(1,I,J,K) = (Z1*X2 - X1*Z2)*0.5
            FACEZ(1,I,J,K) = (X1*Y2 - Y1*X2)*0.5
C
C---  second, the J - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J+1,K)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J+1,K)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J+1,K)
            X2 = X(I+1,J+1,K) - X(I,J+1,K+1)
            Y2 = Y(I+1,J+1,K) - Y(I,J+1,K+1)
            Z2 = Z(I+1,J+1,K) - Z(I,J+1,K+1)
C
            FACEX(2,I,J,K) = (Y1*Z2 - Z1*Y2)*0.5
            FACEY(2,I,J,K) = (Z1*X2 - X1*Z2)*0.5
            FACEZ(2,I,J,K) = (X1*Y2 - Y1*X2)*0.5
```

```
C
C---  third, the K - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J,K+1)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J,K+1)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J,K+1)
            X2 = X(I,J+1,K+1) - X(I+1,J,K+1)
            Y2 = Y(I,J+1,K+1) - Y(I+1,J,K+1)
            Z2 = Z(I,J+1,K+1) - Z(I+1,J,K+1)
C
            FACEX(3,I,J,K) = (Y1*Z2 - Z1*Y2)*0.5
            FACEY(3,I,J,K) = (Z1*X2 - X1*Z2)*0.5
            FACEZ(3,I,J,K) = (X1*Y2 - Y1*X2)*0.5
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE

      IF(SYM) THEN
C
C---  find the normals on the symmetry plane
C
      DO 50 I = 1,IM
        DO 40 J = 1,JM
          X1 = X(I+1,J,1) - X(I,J+1,1)
          Z1 = Z(I+1,J,1) - Z(I,J+1,1)
          X2 = X(I+1,J+1,1) - X(I,J,1)
          Z2 = Z(I+1,J+1,1) - Z(I,J,1)
C
          WALLB(I,J) = (Z1*X2 - X1*Z2)*0.5

          X1 = X(I+1,J+1,KMAX) - X(I,J,KMAX)
          Z1 = Z(I+1,J+1,KMAX) - Z(I,J,KMAX)
          X2 = X(I,J+1,KMAX) - X(I+1,J,KMAX)
          Z2 = Z(I,J+1,KMAX) - Z(I+1,J,KMAX)
C
          WALLT(I,J) = -(Z1*X2 - X1*Z2)*0.5
C....     this is the same as -FACEY(3,I,J,KM)

   40   CONTINUE
   50 CONTINUE
      ENDIF
C
C---  find the normals at the upstream boundary cells (i=1)
C
      DO 51 J=1,JM
        DO 41 K=1,KM
          Y1 = Y(1,J+1,K) - Y(1,J,K+1)
```

```
          Z1 = Z(1,J+1,K) - Z(1,J,K+1)
          Y2 = Y(1,J+1,K+1) - Y(1,J,K)
          Z2 = Z(1,J+1,K+1) - Z(1,J,K)

          FRONT(J,K) = (Y1*Z2 - Y2*Z1)*0.5

  41    CONTINUE
  51  CONTINUE

C
C---  find the normals on the wing
C
      DO 70 I = 1,IM
        DO 60 K = 1,KM
          X1 = X(I+1,1,K+1) - X(I,1,K)
          Y1 = Y(I+1,1,K+1) - Y(I,1,K)
          Z1 = Z(I+1,1,K+1) - Z(I,1,K)
          X2 = X(I,1,K+1) - X(I+1,1,K)
          Y2 = Y(I,1,K+1) - Y(I+1,1,K)
          Z2 = Z(I,1,K+1) - Z(I+1,1,K)
C
          FACEX(2,I,0,K) = -(Y1*Z2 - Z1*Y2)*0.5
          FACEY(2,I,0,K) = -(Z1*X2 - X1*Z2)*0.5
          FACEZ(2,I,0,K) = -(X1*Y2 - Y1*X2)*0.5
   60   CONTINUE
   70 CONTINUE
      RETURN
      END
```

```
C
C      SUBROUTINE PROUT: this subroutine prints out the input data
C      for each case.
C
       SUBROUTINE PROUT
       include 'euler.inc'

       OPEN(UNIT=16,STATUS='unknown',FORM='FORMATTED',FILE=INPDAT)
       KTIP = (KMAX+1)/2
       WRITE(16,*) DATE,' DATE'
       WRITE(16,*) MACH, ' MACH'
       WRITE(16,*) AOA,' ANGLE OF ATTACK'
       WRITE(16,*) YAW,' YAW ANGLE'
       WRITE(16,*) GAM,' GAMMA'
       WRITE(16,*) KAP2,' KAP2'
       WRITE(16,*) KAP4,' KAP4'
       WRITE(16,*) CFL,' CFL'
       WRITE(16,*) AENTH,' AENTH'
       WRITE(16,*) ITMAX,' MAX ITER'
       WRITE(16,*) ITCOEF,' ITCOEF'
       WRITE(16,*) ITPRIN,' ITPRIN'
       WRITE(16,*) BIN,' BIN'
       WRITE(16,*) CFBIN,' CFBIN'
       WRITE(16,*) ITER,' ITER'
C      If the code is being restarted from a previous solution, the value
C      input for ITER must be > 1 (assign input to unit 8);
C      if you're starting from scratch, ITER = 1
       WRITE(16,*) ICON,' ICON'
C      if ICON = 0 then initial conditions are set to the conical soln
C      if ICON = 1 then initial conditions are the soln to a lower alpha
C      (assign input to unit 12)
C***** assign conical solution to unit 11 ******
       WRITE(16,*) ILE,' ILE'
       WRITE(16,*) ITE,' ITE'
       WRITE(16,*) RE,' RE'
       WRITE(16,*) EPSR,' EPSR'
       WRITE(16,*) GRNAME,' GRID'
       WRITE(16,*) RSNAME,' RESIDUAL HISTORY'
       WRITE(16,*) STNODE,' U AT NODES'
       WRITE(16,*) SVNAME,' U AT CENTERS'
       WRITE(16,*) CFNAME,' COEFFICIENTS'
       WRITE(16,*) RESTRT,' RESTART DATA'
       WRITE(16,*) INPDAT,' INPUT'
       WRITE(16,*) LOWANG,' LOWER ANGLE SOLN'
       WRITE(16,*) OLDRES,' OLD RESIDUAL HISTORY'

       RETURN
```

```
END
```

```
C
C  SUBROUTINE RESTART:  initializes the state vector for the Euler
C  solver by reading in old solution to restart a calculation
C
      SUBROUTINE RESTART
      include 'euler.inc'
C
C---  read in old solution
C
cbl      IF(MACH.GT.1.0) THEN
        ISTART = 0
cbl      ELSE
cbl        ISTART = 1
cbl      ENDIF

      IF(BIN) THEN
      OPEN(UNIT=8,STATUS='OLD',FORM='UNFORMATTED',FILE=RESTRT)
      READ(8) DATE
      READ(8) IIN,JIN,KIN
      WRITE(6,*) 'IIN=',IIN,' JIN=',JIN,' KIN=',KIN
      IF(IIN.NE.IM .OR. JIN.NE.JM .OR. KIN.NE.KM) THEN
        WRITE(6,*) 'INPUT ARRAY DOES NOT HAVE THE CORRECT DIMENSIONS'
        STOP
      ENDIF
      DO 13 I=ISTART,IIN+1
        DO 14 J=0,JIN+1
          DO 15 K=0,KIN+1
             READ(8) (U(L,I,J,K),L=1,5)
 15       CONTINUE
 14     CONTINUE
 13   CONTINUE
      ELSE

 25   FORMAT(A20)
      OPEN(UNIT=8,STATUS='OLD',FORM='FORMATTED',FILE=RESTRT)
      READ(4,*) DATE
      READ(8,*) IIN,JIN,KIN
      WRITE(6,*) 'IIN=',IIN,' JIN=',JIN,' KIN=',KIN
      IF(IIN.NE.IM .OR. JIN.NE.JM .OR. KIN.NE.KM) THEN
        WRITE(6,*) 'INPUT ARRAY DOES NOT HAVE THE CORRECT DIMENSIONS'
        STOP
      ENDIF
      DO 10 I=ISTART,IIN+1
        DO 11 J=0,JIN+1
          DO 12 K=0,KIN+1
             READ(8,*) (U(L,I,J,K),L=1,5)
```

```
12      CONTINUE
11    CONTINUE
10  CONTINUE

    ENDIF
    OPEN(UNIT=15,STATUS='OLD',FORM='FORMATTED',FILE=OLDRES)
    DO 26 II=1,ITER-1
      READ(15,28) NN,RMS(II),RMS2(II),RMS3(II),RMS4(II),
   &                 RMS5(II),RESMAX(II)
26  CONTINUE
28  FORMAT(I6,6E12.4E2)
    DO 27 II=1,ITER-1
      WRITE(2,28) II,RMS(II),RMS2(II),RMS3(II),RMS4(II),
   &                 RMS5(II),RESMAX(II)
27  CONTINUE

    RETURN
    END
```

```
C
C
C  SUBROUTINE SAVESET:  stores the state vector in the cell centers
C  to use as a restart file
C
      SUBROUTINE SAVESET
      include 'euler.inc'

      IF(BIN) THEN
        OPEN(UNIT=4,STATUS='OLD',FORM='UNFORMATTED',FILE=SVNAME)
      ELSE
        OPEN(UNIT=4,STATUS='OLD',FORM='FORMATTED',FILE=SVNAME)
      ENDIF

cbl      IF(MACH.GT.1.) THEN

      IF(BIN) THEN
        REWIND(4)
        WRITE(4) DATE
        WRITE(4) IM,JM,KM
        DO 20 I=0,IMAX
          DO 21 J=0,JMAX
            DO 22 K=0,KMAX
              WRITE(4) (U(L,I,J,K),L=1,5)
  22        CONTINUE
  21      CONTINUE
  20    CONTINUE
      ELSE
        REWIND(4)
        WRITE(4,*) DATE
        WRITE(4,*) IM,JM,KM
        DO 23 I=0,IMAX
          DO 24 J=0,JMAX
            DO 25 K=0,KMAX
              WRITE(4,*) (U(L,I,J,K),L=1,5)
  25        CONTINUE
  24      CONTINUE
  23    CONTINUE
      ENDIF

cbl      ELSE

cbl      IF(BIN) THEN
cbl        REWIND(4)
cbl        WRITE(4) DATE
cbl        WRITE(4) IM,JM,KM
cbl        DO 30 I=1,IM
```

```
cbl           DO 31 J=1,JM
cbl             DO 32 K=1,KM
cbl               WRITE(4) (U(L,I,J,K),L=1,5)
cbl  32         CONTINUE
cbl  31       CONTINUE
cbl  30     CONTINUE
cbl       ELSE
cbl         REWIND(4)
cbl         WRITE(4,*) DATE
cbl         WRITE(4,*) IM,JM,KM
cbl         DO 33 I=1,IM
cbl           DO 34 J=1,JM
cbl             DO 35 K=1,KM
cbl               WRITE(4,*) (U(L,I,J,K),L=1,5)
cbl  35         CONTINUE
cbl  34       CONTINUE
cbl  33     CONTINUE
cbl       ENDIF
cbl
cbl       ENDIF

      CLOSE(4)

      RETURN
      END
```

```
C
C   SUBROUTINE SUBSONIC:  initializes the state vector for the Euler
C   solver when running cases which have a subsonic freestream Mach
C   number.
C
      SUBROUTINE SUBSONIC
      include 'euler.inc'

      IF(ICON.EQ.1) THEN
C.... if starting from a lower angle solution then read in data file
        OPEN(UNIT=12,STATUS='OLD',FORM='UNFORMATTED',FILE=LOWANG)
        READ(12) IMO,JMO,KMO
        IF(IMO.NE.IM .OR. JMO.NE.JM .OR. KMO.NE.KM) THEN
          WRITE(6,*) 'IM,JM,KM FROM INPUT DOES NOT MATCH CASE RUNNING'
          STOP
        ENDIF
cbl        DO 20 I=1,IMO
        DO 20 I=0,IMO
          DO 21 J=1,JMO
            DO 22 K=1,KMO
              READ(12) (U(L,I,J,K),L=1,5)
  22        CONTINUE
  21      CONTINUE
  20  CONTINUE

C.... calculate initial pressure field
cbl      DO 23 I=1,IM
      DO 23 I=0,IM
        DO 24 J=1,JM
          DO 25 K=1,KM
            RHO = U(1,I,J,K)
            U1 = U(2,I,J,K)/RHO
            V1 = U(3,I,J,K)/RHO
            W1 = U(4,I,J,K)/RHO
            EN = U(5,I,J,K)
            P(I,J,K) = (EN - .5*(U1**2+V1**2+W1**2)*RHO)*(GAM-1.)
  25      CONTINUE
  24    CONTINUE
  23  CONTINUE
      ELSE

C....initializing to freestream conditions
      DO 10 I=0,IM
        DO 11 J=1,JM
          DO 12 K=1,KM
            U(1,I,J,K) = 1.
            U(2,I,J,K) = UIN
```

```
            U(3,I,J,K) = 0.
            U(4,I,J,K) = WIN
            U(5,I,J,K) = EIN
              P(I,J,K) = PIN
12        CONTINUE
11      CONTINUE
10    CONTINUE
      ENDIF

      RETURN
      END
```

```
C
C
C  SUBROUTINE SUPERIC:  initializes the state vector for the
C  solver by setting the upstream variables at i=0 to a conical (or
C  some other) solution and using that as the
C  initial condition everywhere in the field.
C
      SUBROUTINE SUPERIC
      include 'euler.inc'
      CHARACTER*20 FNAME,TITLE,DATEOLD

C
C---- For zero angle of attack Euler problems or problems where no upstream
C---- solution is available initialize flow to freestream:
      IEULER = 0
      IF(AOA .EQ. 0. .AND. REYNUM .EQ. 0.) IEULER = 1
      IF(CONSOLN .EQ. 'freestream' .OR. IEULER .EQ. 1) THEN
       I=0
       write(6,*) 'Initial conditions are freestream...'
       DO 15 J=1,JM
         DO 16 K=1,KM
            U(1,I,J,K)=1.
            U(2,I,J,K)=UIN
            U(3,I,J,K)=0.
            U(4,I,J,K)=WIN
            U(5,I,J,K)=EIN
              P(I,J,K)=PIN
  16     CONTINUE
  15   CONTINUE
        DO 12 I=1,IM
          DO 13 J=1,JM
            DO 14 K=1,KM
              U(1,I,J,K)=1.
              U(2,I,J,K)=UIN
              U(3,I,J,K)=0
              U(4,I,J,K)=WIN
              U(5,I,J,K)=EIN
                P(I,J,K)=PIN
  14        CONTINUE
  13       CONTINUE
  12     CONTINUE
        RETURN
      ELSE
      write(6,*) 'Conical starting solution'
C
C----- (Assume its unformatted)
        OPEN(UNIT=11,STATUS='OLD',FORM='UNFORMATTED',FILE=CONSOLN)
```

```
      READ(11) TITLE
      READ(11) JIN,KIN
      IF(JIN .NE. JM .OR. KIN .NE. KM) THEN
        WRITE(6,*) 'JIN/KIN DO NOT EQUAL JM/KM'
        STOP
      ENDIF
      DO 10 K=1,KIN
        DO 20 J=1,JIN
          READ(11) (U(L,0,J,K),L=1,5)
 20     CONTINUE
 10   CONTINUE
      CLOSE(11)
C
C---  storing conical flow data as boundary condition at i=0
C
c       write(6,*) 'before 11'
      DO 11 J=1,JM
c       write(6,*) 'U(1,j,1)=',u(2,0,j,1),j
        DO 21 K=1,KM
          UTEMP = U(3,0,J,K)
          U(3,0,J,K) = U(4,0,J,K)
          U(4,0,J,K) = UTEMP
 21     CONTINUE
 11   CONTINUE
C
      DO 41 J=1,JM
c      write(6,*) 'j=',j
       DO 51 K=1,KM
c        write(6,*)'rho,u=',u(1,0,j,k),u(2,0,j,k),' @',j,k
         RHO = U(1,0,J,K)
         UCOMP = U(2,0,J,K)/RHO
         VCOMP = U(3,0,J,K)/RHO
         WCOMP = U(4,0,J,K)/RHO
         EN = U(5,0,J,K)
         P(0,J,K) = (EN - .5*(UCOMP**2 + VCOMP**2 + WCOMP**2)*
     &                  *RHO)*(GAM - 1.)
 51     CONTINUE
 41   CONTINUE
C
C---  using conical flow solution as initial guess everywhere
      DO 30 I=1,IM
        DO 40 J=1,JM
          DO 50 K=1,KM
            U(1,I,J,K) = U(1,0,J,K)
            U(2,I,J,K) = U(2,0,J,K)
            U(3,I,J,K) = U(3,0,J,K)
            U(4,I,J,K) = U(4,0,J,K)
```

```
            U(5,I,J,K) = U(5,0,J,K)
              P(I,J,K) =   P(0,J,K)
50          CONTINUE
40        CONTINUE
30      CONTINUE

      ENDIF
      write(6,*) 'after consoln'

      RETURN
      END
```

```
C
C
C  SUBROUTINE TIMSTP:  advances solution Euler equations by a four-
C  stage time stepping scheme, using a spacially varying time step
C  and frozen dissipative terms in order to speed convergence
C
      SUBROUTINE TIMSTP
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      REAL DWL(5,ISIZ,JSIZ,KSIZ)
C
C--- start four-stage time stepping procedure
C
      DO 10 I = 1,IM
      DO 10 J = 1,JM
      DO 10 K = 1,KM
         DTLOC = ALPHA1*DT(I,J,K)
         R(1,I,J,K) = DTLOC*(R(1,I,J,K) + D(1,I,J,K))
         R(2,I,J,K) = DTLOC*(R(2,I,J,K) + D(2,I,J,K))
         R(3,I,J,K) = DTLOC*(R(3,I,J,K) + D(3,I,J,K))
         R(4,I,J,K) = DTLOC*(R(4,I,J,K) + D(4,I,J,K))
         R(5,I,J,K) = DTLOC*(R(5,I,J,K) + D(5,I,J,K))
 10   CONTINUE
C
C---- Semi-implicit:
      IF (SEMIIMP) CALL LHSINV(ALPHA1)
C
      DO 14 I = 1,IM
      DO 14 J = 1,JM
      DO 14 K = 1,KM
         DWL(1,I,J,K) = R(1,I,J,K)
         DWL(2,I,J,K) = R(2,I,J,K)
         DWL(3,I,J,K) = R(3,I,J,K)
         DWL(4,I,J,K) = R(4,I,J,K)
         DWL(5,I,J,K) = R(5,I,J,K)
 14   CONTINUE
      call dbg
      DO 16 I = 1,IM
      DO 16 J = 1,JM
      DO 16 K = 1,KM
         U(1,I,J,K) = U(1,I,J,K) + R(1,I,J,K)
         U(2,I,J,K) = U(2,I,J,K) + R(2,I,J,K)
         U(3,I,J,K) = U(3,I,J,K) + R(3,I,J,K)
         U(4,I,J,K) = U(4,I,J,K) + R(4,I,J,K)
         U(5,I,J,K) = U(5,I,J,K) + R(5,I,J,K)
 16   CONTINUE
      CALL BNDRYC
```

```
      CALL FLUX

C
C---  second stage
C
      DO 20 I = 1,IM
      DO 20 J = 1,JM
      DO 20 K = 1,KM
         DTLOC = ALPHA2*DT(I,J,K)
         R(1,I,J,K) = DTLOC*(R(1,I,J,K) + D(1,I,J,K))
         R(2,I,J,K) = DTLOC*(R(2,I,J,K) + D(2,I,J,K))
         R(3,I,J,K) = DTLOC*(R(3,I,J,K) + D(3,I,J,K))
         R(4,I,J,K) = DTLOC*(R(4,I,J,K) + D(4,I,J,K))
         R(5,I,J,K) = DTLOC*(R(5,I,J,K) + D(5,I,J,K))
 20   CONTINUE
C
C---- Residual smoothing?
      IF (EPSR .GT. 0. .AND. .NOT. SEMIIMP)CALL RESMOOTH
C
      DO 22 I = 1,IM
      DO 22 J = 1,JM
      DO 22 K = 1,KM
         R(1,I,J,K) = R(1,I,J,K) - DWL(1,I,J,K)
         R(2,I,J,K) = R(2,I,J,K) - DWL(2,I,J,K)
         R(3,I,J,K) = R(3,I,J,K) - DWL(3,I,J,K)
         R(4,I,J,K) = R(4,I,J,K) - DWL(4,I,J,K)
         R(5,I,J,K) = R(5,I,J,K) - DWL(5,I,J,K)
 22   CONTINUE
C
C---- Semi-implicit:
      IF (SEMIIMP) CALL LHSINV(ALPHA2)
C
      DO 24 I = 1,IM
      DO 24 J = 1,JM
      DO 24 K = 1,KM
         DWL(1,I,J,K) = R(1,I,J,K) + DWL(1,I,J,K)
         DWL(2,I,J,K) = R(2,I,J,K) + DWL(2,I,J,K)
         DWL(3,I,J,K) = R(3,I,J,K) + DWL(3,I,J,K)
         DWL(4,I,J,K) = R(4,I,J,K) + DWL(4,I,J,K)
         DWL(5,I,J,K) = R(5,I,J,K) + DWL(5,I,J,K)
 24   CONTINUE
      call dbg
      DO 26 I = 1,IM
      DO 26 J = 1,JM
      DO 26 K = 1,KM
         U(1,I,J,K) = U(1,I,J,K) + R(1,I,J,K)
         U(2,I,J,K) = U(2,I,J,K) + R(2,I,J,K)
```

```
         U(3,I,J,K) = U(3,I,J,K) + R(3,I,J,K)
         U(4,I,J,K) = U(4,I,J,K) + R(4,I,J,K)
         U(5,I,J,K) = U(5,I,J,K) + R(5,I,J,K)
 26    CONTINUE

       CALL BNDRYC
       CALL FLUX
C
C--- third stage
C

       DO 30 I = 1,IM
       DO 30 J = 1,JM
       DO 30 K = 1,KM
          DTLOC = ALPHA3*DT(I,J,K)
          R(1,I,J,K) = DTLOC*(R(1,I,J,K) + D(1,I,J,K))
          R(2,I,J,K) = DTLOC*(R(2,I,J,K) + D(2,I,J,K))
          R(3,I,J,K) = DTLOC*(R(3,I,J,K) + D(3,I,J,K))
          R(4,I,J,K) = DTLOC*(R(4,I,J,K) + D(4,I,J,K))
          R(5,I,J,K) = DTLOC*(R(5,I,J,K) + D(5,I,J,K))
 30    CONTINUE
C
C---- Residual smoothing?

C
       DO 32 I = 1,IM
       DO 32 J = 1,JM
       DO 32 K = 1,KM
          R(1,I,J,K) = R(1,I,J,K) - DWL(1,I,J,K)
          R(2,I,J,K) = R(2,I,J,K) - DWL(2,I,J,K)
          R(3,I,J,K) = R(3,I,J,K) - DWL(3,I,J,K)
          R(4,I,J,K) = R(4,I,J,K) - DWL(4,I,J,K)
          R(5,I,J,K) = R(5,I,J,K) - DWL(5,I,J,K)
 32    CONTINUE
C
C---- Semi-implicit:
       IF (SEMIIMP) CALL LHSINV(ALPHA3)
C
       DO 34 I = 1,IM
       DO 34 J = 1,JM
       DO 34 K = 1,KM
          DWL(1,I,J,K) = R(1,I,J,K) + DWL(1,I,J,K)
          DWL(2,I,J,K) = R(2,I,J,K) + DWL(2,I,J,K)
          DWL(3,I,J,K) = R(3,I,J,K) + DWL(3,I,J,K)
          DWL(4,I,J,K) = R(4,I,J,K) + DWL(4,I,J,K)
          DWL(5,I,J,K) = R(5,I,J,K) + DWL(5,I,J,K)
 34    CONTINUE
```

```
      call dbg
      DO 36 I = 1,IM
      DO 36 J = 1,JM
      DO 36 K = 1,KM
         U(1,I,J,K) = U(1,I,J,K) + R(1,I,J,K)
         U(2,I,J,K) = U(2,I,J,K) + R(2,I,J,K)
         U(3,I,J,K) = U(3,I,J,K) + R(3,I,J,K)
         U(4,I,J,K) = U(4,I,J,K) + R(4,I,J,K)
         U(5,I,J,K) = U(5,I,J,K) + R(5,I,J,K)
 36   CONTINUE

      CALL BNDRYC
      CALL FLUX
C
C---  final stage
C
      DO 40 I = 1,IM
      DO 40 J = 1,JM
      DO 40 K = 1,KM
         DTLOC = DT(I,J,K)
         R(1,I,J,K) = DTLOC*(R(1,I,J,K) + D(1,I,J,K))
         R(2,I,J,K) = DTLOC*(R(2,I,J,K) + D(2,I,J,K))
         R(3,I,J,K) = DTLOC*(R(3,I,J,K) + D(3,I,J,K))
         R(4,I,J,K) = DTLOC*(R(4,I,J,K) + D(4,I,J,K))
         R(5,I,J,K) = DTLOC*(R(5,I,J,K) + D(5,I,J,K))
 40   CONTINUE
C
C---- Residual smoothing?
      IF (EPSR .GT. 0. .AND. .NOT. SEMIIMP)CALL RESMOOTH
C
      DO 42 I = 1,IM
      DO 42 J = 1,JM
      DO 42 K = 1,KM
         R(1,I,J,K) = R(1,I,J,K) - DWL(1,I,J,K)
         R(2,I,J,K) = R(2,I,J,K) - DWL(2,I,J,K)
         R(3,I,J,K) = R(3,I,J,K) - DWL(3,I,J,K)
         R(4,I,J,K) = R(4,I,J,K) - DWL(4,I,J,K)
         R(5,I,J,K) = R(5,I,J,K) - DWL(5,I,J,K)
 42   CONTINUE
C
C---- Semi-implicit:
      IF (SEMIIMP) CALL LHSINV(1.)
      IF (EPSR .GT. 0. .AND. SEMIIMP)CALL RESMOOTH
C
      DO 44 I = 1,IM
      DO 44 J = 1,JM
      DO 44 K = 1,KM
```

```
         DWL(1,I,J,K) = R(1,I,J,K) + DWL(1,I,J,K)
         DWL(2,I,J,K) = R(2,I,J,K) + DWL(2,I,J,K)
         DWL(3,I,J,K) = R(3,I,J,K) + DWL(3,I,J,K)
         DWL(4,I,J,K) = R(4,I,J,K) + DWL(4,I,J,K)
         DWL(5,I,J,K) = R(5,I,J,K) + DWL(5,I,J,K)
 44    CONTINUE
       call dbg
       DO 46 I = 1,IM
       DO 46 J = 1,JM
       DO 46 K = 1,KM
         U(1,I,J,K) = U(1,I,J,K) + R(1,I,J,K)
         U(2,I,J,K) = U(2,I,J,K) + R(2,I,J,K)
         U(3,I,J,K) = U(3,I,J,K) + R(3,I,J,K)
         U(4,I,J,K) = U(4,I,J,K) + R(4,I,J,K)
         U(5,I,J,K) = U(5,I,J,K) + R(5,I,J,K)
 46    CONTINUE

C
C--- add enthalpy damping and determine the pressure field
C
       IF (AENTH .LE. 0.) GOTO 180
       DO 160 K=1,KM
         DO 160 J=1,JM
           DO 160 I=1,IM
             RHO = U(1,I,J,K)
              U1 = U(2,I,J,K)/RHO
              V1 = U(3,I,J,K)/RHO
              W1 = U(4,I,J,K)/RHO
              EN = U(5,I,J,K)
              PR = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*(GAM-1.)
              HMH = (EN+PR)/RHO - HIN
              H = 1./(1. + AENTH*HMH)
              U(1,I,J,K) = RHO*H
              U(2,I,J,K) = RHO*U1*H
              U(3,I,J,K) = RHO*V1*H
              U(4,I,J,K) = RHO*W1*H
              RH = ((RHO*HMH - PR) - AENTH*PR)/(1. + AENTH)
              U(5,I,J,K) = RH + RHO*H*HIN
              P(I,J,K) = RHO*H*HMH - RH
  160      CONTINUE
  180 CONTINUE
C
C---  calculate the residual, rms(delta-U)
C
       RMS(ITER) = 0.
       RMS2(ITER) = 0.
       RMS3(ITER) = 0.
```

```
      RMS4(ITER) = 0.
      RMS5(ITER) = 0.
      RESMAX(ITER) = 0.
      DO 210 I = 1,IM
        DO 200 J = 1,JM
          DO 190 K = 1,KM
c          DO 191 L=1,5
            DTT = 1./(DT(I,J,K)*vol(i,j,k))
            RES = (DWL(1,I,J,K)*DELNORM(1)*DTT)**2
            RES2 = (DWL(2,I,J,K)*DELNORM(2)*DTT)**2
            RES3 = (DWL(3,I,J,K)*DELNORM(3)*DTT)**2
            RES4 = (DWL(4,I,J,K)*DELNORM(4)*DTT)**2
            RES5 = (DWL(5,I,J,K)*DELNORM(5)*DTT)**2
            RMS(ITER) = RMS(ITER) + RES
            RMS2(ITER) = RMS2(ITER) + RES2
            RMS3(ITER) = RMS3(ITER) + RES3
            RMS4(ITER) = RMS4(ITER) + RES4
            RMS5(ITER) = RMS5(ITER) + RES5
            IF(RES2 .GT. RESMAX(ITER)) THEN
              RESMAX(ITER) = RES2
              IRES = I
              JRES = J
              KRES = K
              LMAX = 2
            ENDIF
c  191     CONTINUE
  190     CONTINUE
  200   CONTINUE
  210 CONTINUE
c      write(6,99) iter,rmsres(iter),dwl(lmax,ires,jres,kres)
      RMS(ITER) = SQRT(RMS(ITER)/(FLOAT(NCELLS)))
      RMS2(ITER) = SQRT(RMS2(ITER)/(FLOAT(NCELLS)))
      RMS3(ITER) = SQRT(RMS3(ITER)/(FLOAT(NCELLS)))
      RMS4(ITER) = SQRT(RMS4(ITER)/(FLOAT(NCELLS)))
      RMS5(ITER) = SQRT(RMS5(ITER)/(FLOAT(NCELLS)))
      RESMAX(ITER) = DELNORM(2)*ABS(DWL(2,IRES,JRES,KRES))
C
C---- Output
      WRITE(6,96) ITER
      WRITE(6,95) RESMAX(ITER),IRES,JRES,KRES
 96   FORMAT('Iter #',I4)
 95   FORMAT('Max=',E12.3E2,'                                    @',3I3)
      WRITE(6,97) RMS(ITER),RMS2(ITER),RMS3(ITER),RMS4(ITER),RMS5(ITER)
C      WRITE(6,*) 'P =',P(IRES,JRES,KRES),' DT =',DT(IRES,JRES,KRES)
C      WRITE(6,98) U(1,IRES,JRES,KRES),U(2,IRES,JRES,KRES),U(3,IRES,JRES,
C     *            KRES),U(4,IRES,JRES,KRES),U(5,IRES,JRES,KRES)
C      WRITE(6,98)D(1,IRES,JRES,KRES),D(2,IRES,JRES,KRES),D(3,IRES,JRES,
```

```
C     *            KRES),D(4,IRES,JRES,KRES),D(5,IRES,JRES,KRES)
C      WRITE(6,98)R(1,IRES,JRES,KRES),R(2,IRES,JRES,KRES),R(3,IRES,JRES,
C     *            KRES),R(4,IRES,JRES,KRES),R(5,IRES,JRES,KRES)
      WRITE(6,*)

      WRITE(2,99) ITER,RMS(ITER),RMS2(ITER),RMS3(ITER),RMS4(ITER),
     *            RMS5(ITER),RESMAX(ITER)
 98   FORMAT(5E13.3E3)
 97   FORMAT('RMS   ',5E10.3E2)
 99   FORMAT(I6,6E12.4E2)
      RETURN
      END
```

```
C
C
C  SUBROUTINE UCALC:  calculates the state vector at the grid points
C
C
      SUBROUTINE UCALC
      include 'euler.inc'

      IF (GRNAME .EQ. 'ni.gri') GOTO 98
C...  interior points
      CON = 1./8.
      DO 10 I=1,IMAX
        DO 20 J=2,JMAX
          DO 30 K=2,KM
            UO(1,I,J,K) = CON*(U(1,I-1,J-1,K-1)+U(1,I,J-1,K-1)+
     &         U(1,I,J,K-1)+U(1,I-1,J,K-1)+U(1,I-1,J-1,K)+U(1,I,J-1,K)
     &         +U(1,I,J,K)+U(1,I-1,J,K))
            UO(2,I,J,K) = CON*(U(2,I-1,J-1,K-1)+U(2,I,J-1,K-1)+
     &         U(2,I,J,K-1)+U(2,I-1,J,K-1)+U(2,I-1,J-1,K)+U(2,I,J-1,K)
     &         +U(2,I,J,K)+U(2,I-1,J,K))
            UO(3,I,J,K) = CON*(U(3,I-1,J-1,K-1)+U(3,I,J-1,K-1)+
     &         U(3,I,J,K-1)+U(3,I-1,J,K-1)+U(3,I-1,J-1,K)+U(3,I,J-1,K)
     &         +U(3,I,J,K)+U(3,I-1,J,K))
            UO(4,I,J,K) = CON*(U(4,I-1,J-1,K-1)+U(4,I,J-1,K-1)+
     &         U(4,I,J,K-1)+U(4,I-1,J,K-1)+U(4,I-1,J-1,K)+U(4,I,J-1,K)
     &         +U(4,I,J,K)+U(4,I-1,J,K))
            UO(5,I,J,K) = CON*(U(5,I-1,J-1,K-1)+U(5,I,J-1,K-1)+
     &         U(5,I,J,K-1)+U(5,I-1,J,K-1)+U(5,I-1,J-1,K)+U(5,I,J-1,K)
     &         +U(5,I,J,K)+U(5,I-1,J,K))
  30      CONTINUE
  20    CONTINUE
  10  CONTINUE

C...  J=1 LINE AND WING SURFACE
      J=1
      IF(MACH.GT.1.) THEN
      DO 12 I=1,ILE-1
        DO 22 K=2,KM
          UO(1,I,J,K) = CON*(U(1,I-1,J,K-1)+U(1,I,J,K-1)+
     &         U(1,I,J,K)+U(1,I-1,J,K)+U(1,I-1,J,KMAX-(K-1))+
     &         U(1,I,J,KMAX-(K-1))+U(1,I,J,KMAX-K)+U(1,I-1,J,KMAX-K))
          UO(2,I,J,K) = CON*(U(2,I-1,J,K-1)+U(2,I,J,K-1)+
     &         U(2,I,J,K)+U(2,I-1,J,K)+U(2,I-1,J,KMAX-(K-1))+
     &         U(2,I,J,KMAX-(K-1))+U(2,I,J,KMAX-K)+U(2,I-1,J,KMAX-K))
          UO(3,I,J,K) = CON*(U(3,I-1,J,K-1)+U(3,I,J,K-1)+
     &         U(3,I,J,K)+U(3,I-1,J,K)+U(3,I-1,J,KMAX-(K-1))+
     &         U(3,I,J,KMAX-(K-1))+U(3,I,J,KMAX-K)+U(3,I-1,J,KMAX-K))
```

```
      UO(4,I,J,K) = CON*(U(4,I-1,J,K-1)+U(4,I,J,K-1)+
     &      U(4,I,J,K)+U(4,I-1,J,K)+U(4,I-1,J,KMAX-(K-1))+
     &      U(4,I,J,KMAX-(K-1))+U(4,I,J,KMAX-K)+U(4,I-1,J,KMAX-K))
      UO(5,I,J,K) = CON*(U(5,I-1,J,K-1)+U(5,I,J,K-1)+
     &      U(5,I,J,K)+U(5,I-1,J,K)+U(5,I-1,J,KMAX-(K-1))+
     &      U(5,I,J,KMAX-(K-1))+U(5,I,J,KMAX-K)+U(5,I-1,J,KMAX-K))
 22   CONTINUE
 12  CONTINUE

     ELSE
     DO 120 I=1,ILE-1
       SUM1 = 0.
       SUM2 = 0.
       SUM3 = 0.
       SUM4 = 0.
       SUM5 = 0.

       DO 121 K=1,KM
         SUM1 = SUM1+1.5*(U(1,I-1,J,K)+U(1,I,J,K))-.5*(U(1,I-1,J+1,K)
     &         +U(1,I,J+1,K))
         SUM2 = SUM2+1.5*(U(2,I-1,J,K)+U(2,I,J,K))-.5*(U(2,I-1,J+1,K)
     &         +U(2,I,J+1,K))
         SUM3 = SUM3+1.5*(U(3,I-1,J,K)+U(3,I,J,K))-.5*(U(3,I-1,J+1,K)
     &         +U(3,I,J+1,K))
         SUM4 = SUM4+1.5*(U(4,I-1,J,K)+U(4,I,J,K))-.5*(U(4,I-1,J+1,K)
     &         +U(4,I,J+1,K))
         SUM5 = SUM5+1.5*(U(5,I-1,J,K)+U(5,I,J,K))-.5*(U(5,I-1,J+1,K)
     &         +U(5,I,J+1,K))
121    CONTINUE

       DO 122 K=1,KMAX
         UO(1,I,J,K) = SUM1/(2.*KM)
         UO(2,I,J,K) = SUM2/(2.*KM)
         UO(3,I,J,K) = SUM3/(2.*KM)
         UO(4,I,J,K) = SUM4/(2.*KM)
         UO(5,I,J,K) = SUM5/(2.*KM)
122    CONTINUE

120  CONTINUE
     ENDIF

     DO 14 I=ITE+1,IMAX
       DO 24 K=2,KM
         UO(1,I,J,K) = CON*(U(1,I-1,J,K-1)+U(1,I,J,K-1)+
     &      U(1,I,J,K)+U(1,I-1,J,K)+U(1,I-1,J,KMAX-(K-1))+
     &      U(1,I,J,KMAX-(K-1))+U(1,I,J,KMAX-K)+U(1,I-1,J,KMAX-K))
         UO(2,I,J,K) = CON*(U(2,I-1,J,K-1)+U(2,I,J,K-1)+
```

```
     &         U(2,I,J,K)+U(2,I-1,J,K)+U(2,I-1,J,KMAX-(K-1))+
     &         U(2,I,J,KMAX-(K-1))+U(2,I,J,KMAX-K)+U(2,I-1,J,KMAX-K))
          UO(3,I,J,K) = CON*(U(3,I-1,J,K-1)+U(3,I,J,K-1)+
     &         U(3,I,J,K)+U(3,I-1,J,K)+U(3,I-1,J,KMAX-(K-1))+
     &         U(3,I,J,KMAX-(K-1))+U(3,I,J,KMAX-K)+U(3,I-1,J,KMAX-K))
          UO(4,I,J,K) = CON*(U(4,I-1,J,K-1)+U(4,I,J,K-1)+
     &         U(4,I,J,K)+U(4,I-1,J,K)+U(4,I-1,J,KMAX-(K-1))+
     &         U(4,I,J,KMAX-(K-1))+U(4,I,J,KMAX-K)+U(4,I-1,J,KMAX-K))
          UO(5,I,J,K) = CON*(U(5,I-1,J,K-1)+U(5,I,J,K-1)+
     &         U(5,I,J,K)+U(5,I-1,J,K)+U(5,I-1,J,KMAX-(K-1))+
     &         U(5,I,J,KMAX-(K-1))+U(5,I,J,KMAX-K)+U(5,I-1,J,KMAX-K))
 24     CONTINUE
 14   CONTINUE

      IF(MACH .GT. 1.) THEN
        IWING = ILE
      ELSE
        IWING = ILE+1
      ENDIF

      CON = 1./4.
      DO 11 I=IWING,ITE
        DO 21 K=2,KM
          UO(1,I,J,K) = CON*(1.5*(U(1,I-1,J,K)+U(1,I,J,K))-
     &       .5*(U(1,I-1,J+1,K)+U(1,I,J+1,K))+1.5*(U(1,I-1,J,K-1)+
     &       U(1,I,J,K-1))-.5*(U(1,I-1,J+1,K-1)+U(1,I,J+1,K-1)))
          UO(2,I,J,K) = CON*(1.5*(U(2,I-1,J,K)+U(2,I,J,K))-
     &       .5*(U(2,I-1,J+1,K)+U(2,I,J+1,K))+1.5*(U(2,I-1,J,K-1)+
     &       U(2,I,J,K-1))-.5*(U(2,I-1,J+1,K-1)+U(2,I,J+1,K-1)))
          UO(3,I,J,K) = CON*(1.5*(U(3,I-1,J,K)+U(3,I,J,K))-
     &       .5*(U(3,I-1,J+1,K)+U(3,I,J+1,K))+1.5*(U(3,I-1,J,K-1)+
     &       U(3,I,J,K-1))-.5*(U(3,I-1,J+1,K-1)+U(3,I,J+1,K-1)))
          UO(4,I,J,K) = CON*(1.5*(U(4,I-1,J,K)+U(4,I,J,K))-
     &       .5*(U(4,I-1,J+1,K)+U(4,I,J+1,K))+1.5*(U(4,I-1,J,K-1)+
     &       U(4,I,J,K-1))-.5*(U(4,I-1,J+1,K-1)+U(4,I,J+1,K-1)))
          UO(5,I,J,K) = CON*(1.5*(U(5,I-1,J,K)+U(5,I,J,K))-
     &       .5*(U(5,I-1,J+1,K)+U(5,I,J+1,K))+1.5*(U(5,I-1,J,K-1)+
     &       U(5,I,J,K-1))-.5*(U(5,I-1,J+1,K-1)+U(5,I,J+1,K-1)))
 21     CONTINUE
 11   CONTINUE

      IF(SYM) THEN
        DO 16 I=0,IMAX
          DO 26 J=1,JMAX
            U(1,I,J,0) =  U(1,I,J,1)
            U(2,I,J,0) =  U(2,I,J,1)
            U(3,I,J,0) = -U(3,I,J,1)
```

```
          U(4,I,J,0) =  U(4,I,J,1)
          U(5,I,J,0) =  U(5,I,J,1)

          U(1,I,J,KMAX) =  U(1,I,J,KM)
          U(2,I,J,KMAX) =  U(2,I,J,KM)
          U(3,I,J,KMAX) = -U(3,I,J,KM)
          U(4,I,J,KMAX) =  U(4,I,J,KM)
          U(5,I,J,KMAX) =  U(5,I,J,KM)
26      CONTINUE
16    CONTINUE

      K=KMAX
      CON=1./8.
      DO 17 I=1,IMAX
        DO 27 J=2,JMAX
          UO(1,I,J,K) = CON*(U(1,I-1,J-1,K-1)+U(1,I,J-1,K-1)+
     &       U(1,I-1,J-1,K)+U(1,I,J-1,K)+U(1,I-1,J,K)+U(1,I,J,K)+
     &       U(1,I-1,J,K-1)+U(1,I,J,K-1))
          UO(2,I,J,K) = CON*(U(2,I-1,J-1,K-1)+U(2,I,J-1,K-1)+
     &       U(2,I-1,J-1,K)+U(2,I,J-1,K)+U(2,I-1,J,K)+U(2,I,J,K)+
     &       U(2,I-1,J,K-1)+U(2,I,J,K-1))
          UO(3,I,J,K) = CON*(U(3,I-1,J-1,K-1)+U(3,I,J-1,K-1)+
     &       U(3,I-1,J-1,K)+U(3,I,J-1,K)+U(3,I-1,J,K)+U(3,I,J,K)+
     &       U(3,I-1,J,K-1)+U(3,I,J,K-1))
          UO(4,I,J,K) = CON*(U(4,I-1,J-1,K-1)+U(4,I,J-1,K-1)+
     &       U(4,I-1,J-1,K)+U(4,I,J-1,K)+U(4,I-1,J,K)+U(4,I,J,K)+
     &       U(4,I-1,J,K-1)+U(4,I,J,K-1))
          UO(5,I,J,K) = CON*(U(5,I-1,J-1,K-1)+U(5,I,J-1,K-1)+
     &       U(5,I-1,J-1,K)+U(5,I,J-1,K)+U(5,I-1,J,K)+U(5,I,J,K)+
     &       U(5,I-1,J,K-1)+U(5,I,J,K-1))

27      CONTINUE
17    CONTINUE

     J=1
     CON = 1./4.
     DO 31 I=IWING,ITE
         UO(1,I,J,K) = CON*(1.5*(U(1,I-1,J,K)+U(1,I,J,K))-
     &      .5*(U(1,I-1,J+1,K)+U(1,I,J+1,K))+1.5*(U(1,I-1,J,K-1)+
     &       U(1,I,J,K-1))-.5*(U(1,I-1,J+1,K-1)+U(1,I,J+1,K-1)))
         UO(2,I,J,K) = CON*(1.5*(U(2,I-1,J,K)+U(2,I,J,K))-
     &       .5*(U(2,I-1,J+1,K)+U(2,I,J+1,K))+1.5*(U(2,I-1,J,K-1)+
     &       U(2,I,J,K-1))-.5*(U(2,I-1,J+1,K-1)+U(2,I,J+1,K-1)))
         UO(3,I,J,K) = CON*(1.5*(U(3,I-1,J,K)+U(3,I,J,K))-
     &       .5*(U(3,I-1,J+1,K)+U(3,I,J+1,K))+1.5*(U(3,I-1,J,K-1)+
     &       U(3,I,J,K-1))-.5*(U(3,I-1,J+1,K-1)+U(3,I,J+1,K-1)))
         UO(4,I,J,K) = CON*(1.5*(U(4,I-1,J,K)+U(4,I,J,K))-
```

```
     &       .5*(U(4,I-1,J+1,K)+U(4,I,J+1,K))+1.5*(U(4,I-1,J,K-1)+
     &       U(4,I,J,K-1))-.5*(U(4,I-1,J+1,K-1)+U(4,I,J+1,K-1)))
          UO(5,I,J,K) = CON*(1.5*(U(5,I-1,J,K)+U(5,I,J,K))-
     &       .5*(U(5,I-1,J+1,K)+U(5,I,J+1,K))+1.5*(U(5,I-1,J,K-1)+
     &       U(5,I,J,K-1))-.5*(U(5,I-1,J+1,K-1)+U(5,I,J+1,K-1)))
   31  CONTINUE

       CON = 1./8.
       DO 40 I=ITE+1,IMAX
          UO(1,I,J,K) = CON*(U(1,I-1,J,K-1)+U(1,I,J,K-1)+
     &         U(1,I,J,K)+U(1,I-1,J,K)+U(1,I-1,J,KMAX-(K-1))+
     &         U(1,I,J,KMAX-(K-1))+U(1,I,J,KMAX-K)+U(1,I-1,J,KMAX-K))
          UO(2,I,J,K) = CON*(U(2,I-1,J,K-1)+U(2,I,J,K-1)+
     &         U(2,I,J,K)+U(2,I-1,J,K)+U(2,I-1,J,KMAX-(K-1))+
     &         U(2,I,J,KMAX-(K-1))+U(2,I,J,KMAX-K)+U(2,I-1,J,KMAX-K))
          UO(3,I,J,K) = CON*(U(3,I-1,J,K-1)+U(3,I,J,K-1)+
     &         U(3,I,J,K)+U(3,I-1,J,K)+U(3,I-1,J,KMAX-(K-1))+
     &         U(3,I,J,KMAX-(K-1))+U(3,I,J,KMAX-K)+U(3,I-1,J,KMAX-K))
          UO(4,I,J,K) = CON*(U(4,I-1,J,K-1)+U(4,I,J,K-1)+
     &         U(4,I,J,K)+U(4,I-1,J,K)+U(4,I-1,J,KMAX-(K-1))+
     &         U(4,I,J,KMAX-(K-1))+U(4,I,J,KMAX-K)+U(4,I-1,J,KMAX-K))
          UO(5,I,J,K) = CON*(U(5,I-1,J,K-1)+U(5,I,J,K-1)+
     &         U(5,I,J,K)+U(5,I-1,J,K)+U(5,I-1,J,KMAX-(K-1))+
     &         U(5,I,J,KMAX-(K-1))+U(5,I,J,KMAX-K)+U(5,I-1,J,KMAX-K))
   40  CONTINUE

       IF(MACH .LT. 1.) THEN
         I=ILE-1
         J=1
         KTIP = (KMAX +1)/2
         KTIPM = KTIP - 1

         SUM1 = 0.
         SUM2 = 0.
         SUM3 = 0.
         SUM4 = 0.
         SUM5 = 0.

         DO 180 K=KTIP,KM
           SUM1 = SUM1 + 1.5*U(1,I,J,K) - .5*U(1,I,J+1,K)
           SUM2 = SUM2 + 1.5*U(2,I,J,K) - .5*U(2,I,J+1,K)
           SUM3 = SUM3 + 1.5*U(3,I,J,K) - .5*U(3,I,J+1,K)
           SUM4 = SUM4 + 1.5*U(4,I,J,K) - .5*U(4,I,J+1,K)
           SUM5 = SUM5 + 1.5*U(5,I,J,K) - .5*U(5,I,J+1,K)
  180    CONTINUE

         UT1 = SUM1/KTIPM
```

```
        UT2 = SUM2/KTIPM
        UT3 = SUM3/KTIPM
        UT4 = SUM4/KTIPM
        UT5 = SUM5/KTIPM

        I=ILE
        J=1
        K=KMAX
        UTT1=.5*(1.5*(U(1,I,J,K)+U(1,I,J,K-1))-.5*(U(1,I,J+1,K)
     &        +U(1,I,J+1,K-1)))
        UTT2=.5*(1.5*(U(2,I,J,K)+U(2,I,J,K-1))-.5*(U(2,I,J+1,K)
     &        +U(2,I,J+1,K-1)))
        UTT3=.5*(1.5*(U(3,I,J,K)+U(3,I,J,K-1))-.5*(U(3,I,J+1,K)
     &        +U(3,I,J+1,K-1)))
        UTT4=.5*(1.5*(U(4,I,J,K)+U(4,I,J,K-1))-.5*(U(4,I,J+1,K)
     &        +U(4,I,J+1,K-1)))
        UTT5=.5*(1.5*(U(5,I,J,K)+U(5,I,J,K-1))-.5*(U(5,I,J+1,K)
     &        +U(5,I,J+1,K-1)))

        DO 182 K=KTIP,KMAX
          UO(1,ILE,1,K) = .5*(UT1 + UTT1)
          UO(2,ILE,1,K) = .5*(UT2 + UTT2)
          UO(3,ILE,1,K) = .5*(UT3 + UTT3)
          UO(4,ILE,1,K) = .5*(UT4 + UTT4)
          UO(5,ILE,1,K) = .5*(UT5 + UTT5)
182    CONTINUE
      ENDIF

      ELSE
        DO 18 I=0,IMAX
          DO 28 J=1,JMAX
            U(1,I,J,0) = U(1,I,J,KM)
            U(2,I,J,0) = U(2,I,J,KM)
            U(3,I,J,0) = U(3,I,J,KM)
            U(4,I,J,0) = U(4,I,J,KM)
            U(5,I,J,0) = U(5,I,J,KM)
 28       CONTINUE
 18     CONTINUE

      ENDIF

C...  J=1 AND WING SURFACE
      J=1
      K=1
      CON =1./8.
      IF(MACH.GT.1.) THEN
      DO 32 I=1,ILE-1
```

```
      UO(1,I,J,K) = CON*(U(1,I-1,J,K-1)+U(1,I,J,K-1)+
     &      U(1,I,J,K)+U(1,I-1,J,K)+U(1,I-1,J,KMAX-(K-1))+
     &      U(1,I,J,KMAX-(K-1))+U(1,I,J,KMAX-K)+U(1,I-1,J,KMAX-K))
      UO(2,I,J,K) = CON*(U(2,I-1,J,K-1)+U(2,I,J,K-1)+
     &      U(2,I,J,K)+U(2,I-1,J,K)+U(2,I-1,J,KMAX-(K-1))+
     &      U(2,I,J,KMAX-(K-1))+U(2,I,J,KMAX-K)+U(2,I-1,J,KMAX-K))
      UO(3,I,J,K) = CON*(U(3,I-1,J,K-1)+U(3,I,J,K-1)+
     &      U(3,I,J,K)+U(3,I-1,J,K)+U(3,I-1,J,KMAX-(K-1))+
     &      U(3,I,J,KMAX-(K-1))+U(3,I,J,KMAX-K)+U(3,I-1,J,KMAX-K))
      UO(4,I,J,K) = CON*(U(4,I-1,J,K-1)+U(4,I,J,K-1)+
     &      U(4,I,J,K)+U(4,I-1,J,K)+U(4,I-1,J,KMAX-(K-1))+
     &      U(4,I,J,KMAX-(K-1))+U(4,I,J,KMAX-K)+U(4,I-1,J,KMAX-K))
      UO(5,I,J,K) = CON*(U(5,I-1,J,K-1)+U(5,I,J,K-1)+
     &      U(5,I,J,K)+U(5,I-1,J,K)+U(5,I-1,J,KMAX-(K-1))+
     &      U(5,I,J,KMAX-(K-1))+U(5,I,J,KMAX-K)+U(5,I-1,J,KMAX-K))
32    CONTINUE
      ENDIF

      DO 34 I=ITE+1,IMAX
        UO(1,I,J,K) = CON*(U(1,I-1,J,K-1)+U(1,I,J,K-1)+
     &      U(1,I,J,K)+U(1,I-1,J,K)+U(1,I-1,J,KMAX-(K-1))+
     &      U(1,I,J,KMAX-(K-1))+U(1,I,J,KMAX-K)+U(1,I-1,J,KMAX-K))
        UO(2,I,J,K) = CON*(U(2,I-1,J,K-1)+U(2,I,J,K-1)+
     &      U(2,I,J,K)+U(2,I-1,J,K)+U(2,I-1,J,KMAX-(K-1))+
     &      U(2,I,J,KMAX-(K-1))+U(2,I,J,KMAX-K)+U(2,I-1,J,KMAX-K))
        UO(3,I,J,K) = CON*(U(3,I-1,J,K-1)+U(3,I,J,K-1)+
     &      U(3,I,J,K)+U(3,I-1,J,K)+U(3,I-1,J,KMAX-(K-1))+
     &      U(3,I,J,KMAX-(K-1))+U(3,I,J,KMAX-K)+U(3,I-1,J,KMAX-K))
        UO(4,I,J,K) = CON*(U(4,I-1,J,K-1)+U(4,I,J,K-1)+
     &      U(4,I,J,K)+U(4,I-1,J,K)+U(4,I-1,J,KMAX-(K-1))+
     &      U(4,I,J,KMAX-(K-1))+U(4,I,J,KMAX-K)+U(4,I-1,J,KMAX-K))
        UO(5,I,J,K) = CON*(U(5,I-1,J,K-1)+U(5,I,J,K-1)+
     &      U(5,I,J,K)+U(5,I-1,J,K)+U(5,I-1,J,KMAX-(K-1))+
     &      U(5,I,J,KMAX-(K-1))+U(5,I,J,KMAX-K)+U(5,I-1,J,KMAX-K))
34    CONTINUE

      CON = 1./4.
      DO 36 I=IWING,ITE
        UO(1,I,J,K) = CON*(1.5*(U(1,I-1,J,K)+U(1,I,J,K))-
     &     .5*(U(1,I-1,J+1,K)+U(1,I,J+1,K))+1.5*(U(1,I-1,J,K-1)+
     &      U(1,I,J,K-1))-.5*(U(1,I-1,J+1,K-1)+U(1,I,J+1,K-1)))
        UO(2,I,J,K) = CON*(1.5*(U(2,I-1,J,K)+U(2,I,J,K))-
     &      .5*(U(2,I-1,J+1,K)+U(2,I,J+1,K))+1.5*(U(2,I-1,J,K-1)+
     &      U(2,I,J,K-1))-.5*(U(2,I-1,J+1,K-1)+U(2,I,J+1,K-1)))
        UO(3,I,J,K) = CON*(1.5*(U(3,I-1,J,K)+U(3,I,J,K))-
     &      .5*(U(3,I-1,J+1,K)+U(3,I,J+1,K))+1.5*(U(3,I-1,J,K-1)+
     &      U(3,I,J,K-1))-.5*(U(3,I-1,J+1,K-1)+U(3,I,J+1,K-1)))
```

```
      UO(4,I,J,K) = CON*(1.5*(U(4,I-1,J,K)+U(4,I,J,K))-
     &     .5*(U(4,I-1,J+1,K)+U(4,I,J+1,K))+1.5*(U(4,I-1,J,K-1)+
     &     U(4,I,J,K-1))-.5*(U(4,I-1,J+1,K-1)+U(4,I,J+1,K-1)))
      UO(5,I,J,K) = CON*(1.5*(U(5,I-1,J,K)+U(5,I,J,K))-
     &     .5*(U(5,I-1,J+1,K)+U(5,I,J+1,K))+1.5*(U(5,I-1,J,K-1)+
     &     U(5,I,J,K-1))-.5*(U(5,I-1,J+1,K-1)+U(5,I,J+1,K-1)))
36  CONTINUE

    K=1
    CON=1./8.
    DO 15 I=1,IMAX
      DO 25 J=2,JMAX
        UO(1,I,J,K) = CON*(U(1,I-1,J-1,K-1)+U(1,I,J-1,K-1)+
     &     U(1,I-1,J-1,K)+U(1,I,J-1,K)+U(1,I-1,J,K)+U(1,I,J,K)+
     &     U(1,I-1,J,K-1)+U(1,I,J,K-1))
        UO(2,I,J,K) = CON*(U(2,I-1,J-1,K-1)+U(2,I,J-1,K-1)+
     &     U(2,I-1,J-1,K)+U(2,I,J-1,K)+U(2,I-1,J,K)+U(2,I,J,K)+
     &     U(2,I-1,J,K-1)+U(2,I,J,K-1))
        UO(3,I,J,K) = CON*(U(3,I-1,J-1,K-1)+U(3,I,J-1,K-1)+
     &     U(3,I-1,J-1,K)+U(3,I,J-1,K)+U(3,I-1,J,K)+U(3,I,J,K)+
     &     U(3,I-1,J,K-1)+U(3,I,J,K-1))
        UO(4,I,J,K) = CON*(U(4,I-1,J-1,K-1)+U(4,I,J-1,K-1)+
     &     U(4,I-1,J-1,K)+U(4,I,J-1,K)+U(4,I-1,J,K)+U(4,I,J,K)+
     &     U(4,I-1,J,K-1)+U(4,I,J,K-1))
        UO(5,I,J,K) = CON*(U(5,I-1,J-1,K-1)+U(5,I,J-1,K-1)+
     &     U(5,I-1,J-1,K)+U(5,I,J-1,K)+U(5,I-1,J,K)+U(5,I,J,K)+
     &     U(5,I-1,J,K-1)+U(5,I,J,K-1))
25    CONTINUE
15  CONTINUE

    IF(MACH .LT. 1.) THEN
      I=ILE-1
      J=1
      KTIP = (KMAX +1)/2
      KTIPM = KTIP - 1

      SUM1 = 0.
      SUM2 = 0.
      SUM3 = 0.
      SUM4 = 0.
      SUM5 = 0.

      DO 181 K=1,KTIPM
        SUM1 = SUM1 + 1.5*U(1,I,J,K) - .5*U(1,I,J+1,K)
        SUM2 = SUM2 + 1.5*U(2,I,J,K) - .5*U(2,I,J+1,K)
        SUM3 = SUM3 + 1.5*U(3,I,J,K) - .5*U(3,I,J+1,K)
        SUM4 = SUM4 + 1.5*U(4,I,J,K) - .5*U(4,I,J+1,K)
```

```
          SUM5 = SUM5 + 1.5*U(5,I,J,K) - .5*U(5,I,J+1,K)
 181    CONTINUE

        UB1 = SUM1/KTIPM
        UB2 = SUM2/KTIPM
        UB3 = SUM3/KTIPM
        UB4 = SUM4/KTIPM
        UB5 = SUM5/KTIPM

        I=ILE
        J=1
        K=1
        UBB1=.5*(1.5*(U(1,I,J,K)+U(1,I,J,K-1))-.5*(U(1,I,J+1,K)
     &         +U(1,I,J+1,K-1)))
        UBB2=.5*(1.5*(U(2,I,J,K)+U(2,I,J,K-1))-.5*(U(2,I,J+1,K)
     &         +U(2,I,J+1,K-1)))
        UBB3=.5*(1.5*(U(3,I,J,K)+U(3,I,J,K-1))-.5*(U(3,I,J+1,K)
     &         +U(3,I,J+1,K-1)))
        UBB4=.5*(1.5*(U(4,I,J,K)+U(4,I,J,K-1))-.5*(U(4,I,J+1,K)
     &         +U(4,I,J+1,K-1)))
        UBB5=.5*(1.5*(U(5,I,J,K)+U(5,I,J,K-1))-.5*(U(5,I,J+1,K)
     &         +U(5,I,J+1,K-1)))

        DO 183 K=1,KTIPM
          UO(1,ILE,1,K) = .5*(UB1 + UBB1)
          UO(2,ILE,1,K) = .5*(UB2 + UBB2)
          UO(3,ILE,1,K) = .5*(UB3 + UBB3)
          UO(4,ILE,1,K) = .5*(UB4 + UBB4)
          UO(5,ILE,1,K) = .5*(UB5 + UBB5)
 183    CONTINUE
      ENDIF

      IF( .NOT. SYM) THEN
      DO 38 I=1,IMAX
        DO 37 J=1,JMAX
          UO(1,I,J,KMAX) = UO(1,I,J,1)
          UO(2,I,J,KMAX) = UO(2,I,J,1)
          UO(3,I,J,KMAX) = UO(3,I,J,1)
          UO(4,I,J,KMAX) = UO(4,I,J,1)
          UO(5,I,J,KMAX) = UO(5,I,J,1)
  37    CONTINUE
  38  CONTINUE
      ENDIF
      GOTO 99
C
C***************   A SIMPLE MINDED APPROACH FOR NI'S BUMP GEOMETRY **********
C
```

```
C---- interior points and 8 faces
 98   CON = 1./8.
      DO 100 I=1,IMAX
        DO 100 J=1,JMAX
          DO 100 K=1,KMAX
            UO(1,I,J,K) = CON*(U(1,I-1,J-1,K-1)+U(1,I,J-1,K-1)+
     &         U(1,I,J,K-1)+U(1,I-1,J,K-1)+U(1,I-1,J-1,K)+U(1,I,J-1,K)
     &         +U(1,I,J,K)+U(1,I-1,J,K))
            UO(2,I,J,K) = CON*(U(2,I-1,J-1,K-1)+U(2,I,J-1,K-1)+
     &         U(2,I,J,K-1)+U(2,I-1,J,K-1)+U(2,I-1,J-1,K)+U(2,I,J-1,K)
     &         +U(2,I,J,K)+U(2,I-1,J,K))
            UO(3,I,J,K) = CON*(U(3,I-1,J-1,K-1)+U(3,I,J-1,K-1)+
     &         U(3,I,J,K-1)+U(3,I-1,J,K-1)+U(3,I-1,J-1,K)+U(3,I,J-1,K)
     &         +U(3,I,J,K)+U(3,I-1,J,K))
            UO(4,I,J,K) = CON*(U(4,I-1,J-1,K-1)+U(4,I,J-1,K-1)+
     &         U(4,I,J,K-1)+U(4,I-1,J,K-1)+U(4,I-1,J-1,K)+U(4,I,J-1,K)
     &         +U(4,I,J,K)+U(4,I-1,J,K))
            UO(5,I,J,K) = CON*(U(5,I-1,J-1,K-1)+U(5,I,J-1,K-1)+
     &         U(5,I,J,K-1)+U(5,I-1,J,K-1)+U(5,I-1,J-1,K)+U(5,I,J-1,K)
     &         +U(5,I,J,K)+U(5,I-1,J,K))
 100  CONTINUE
      DO 150 L = 1,5
C
C---- Do I seams (AVERAGE ADJACENT VOLUMES)
      DO 110 I = 1,IMAX
      UO(L,I,1,1   ) = UO(L,I,1,2)
      UO(L,I,1,KMAX) = UO(L,I,1,KM)
      UO(L,I,JMAX,1   ) = UO(L,I,JMAX,2)
 110  UO(L,I,JMAX,KMAX) = UO(L,I,JMAX,KM)
C
C---- Do j seams
      Do 115 J = 1,JMAX
      UO(L,1,J,1   ) = UO(L,1,J,2)
      UO(L,1,J,KMAX) = UO(L,1,J,KM)
      UO(L,IMAX,J,1   ) = UO(L,IMAX,J,2)
 115  UO(L,IMAX,J,KMAX) = UO(L,IMAX,J,KM)
C
C---- Do k seams
      DO 130 K = 1,KMAX
      UO(L,1   ,1,K) = .5*(U(L,1 ,1,K-1)+U(L,1 ,1,K))
      UO(L,IMAX,1,K) = .5*(U(L,IM,1,K-1)+U(L,IM,1,K))
      UO(L,1   ,JMAX,K) = .5*(U(L,1 ,JM,K-1)+U(L,1 ,JM,K))
 130  UO(L,IMAX,Jmax,K) = .5*(U(L,IM,JM,K-1)+U(L,IM,JM,K))
C
C---- corners (simple extrapolation from interior OR seam)
      UO(L,1,1,1) = UO(L,1,1,2)
      UO(L,1,1,KMAX) = UO(L,1,1,KM)
```

```
      UO(L,1,JMAX,1) = UO(L,1,JMAX,2)
      UO(L,1,JMAX,KMAX) = U(L,1,JMAX,KM)

      UO(L,IMAX,1,1) = UO(L,IMAX,1,2)
      UO(L,IMAX,1,KMAX) = UO(L,IMAX,1,KM)
      UO(L,IMAX,JMAX,1) = U(L,IMAX,JMAX,2)
      UO(L,IMAX,JMAX,KMAX) = U(L,IMAX,JMAX,KM)
150   CONTINUE
 99   RETURN
      END
```

```
C
C
C  SUBROUTINE UNODES:  calculates the state vector at the grid points
C  and writes them to a data file which is used to plot the data
C
      SUBROUTINE UNODES
      include 'euler.inc'
      DIMENSION Q(ISIZ+1,JSIZ+1,KSIZ+1,5)

C---  filling in outer boundary points at the upstream
C---  and downstream bondary used for plotting

      DO 50 K=1,KM
        DO 51 L=1,5
          U(L,0,JMAX,K) = U(L,0,JM,K)
          U(L,IMAX,JMAX,K) = U(L,IMAX,JM,K)
  51    CONTINUE
  50  CONTINUE

C---  calculate the state vector at the nodes
        CALL UCALC

C       IF(BIN) THEN

      DO 14 I=1,IMAX
        DO 24 J=1,JMAX
          DO 34 K=1,KMAX
            DO 44 L=1,5
              Q(I,J,K,L) = UO(L,I,J,K)
  44        CONTINUE
  34      CONTINUE
  24    CONTINUE
  14  CONTINUE

      RTIME=REAL(ITMAX)
      OPEN(UNIT=3,STATUS='NEW',FORM='UNFORMATTED',FILE=STNODE)
      WRITE(3) IMAX,JMAX,KMAX
      WRITE(3) MACH,ALPHA,RE,RTIME
      WRITE(3) ((((Q(I,J,K,L),I=1,IMAX),J=1,JMAX),K=1,KMAX),L=1,5)

C       ELSE
C
C       DO 10 I=1,IMAX
C         DO 23  J=1,JMAX
C           DO 33  K=1,KMAX
C             WRITE(3,*) UO(1,I,J,K),UO(2,I,J,K),UO(3,I,J,K),
C     &                  UO(4,I,J,K),UO(5,I,J,K)
```

```
C  33      CONTINUE
C  23    CONTINUE
C  10  CONTINUE
C
C      ENDIF
       RETURN
       END
```

```
C
C
C  SUBROUTINE RESMOOTH:  implicity smooths the flux part of the
C                        residuals before each time step.
C
      SUBROUTINE RESMOOTH
      include 'euler.inc'
      include 'blvisc.inc'
      DIMENSION B(KSIZ),RR(ISIZ,JSIZ,KSIZ)

      DO 40 L=1,5

C..... Store the flux and dissipation terms in array RR
      DO 50 I=1,IM
        DO 51 J=1,JM
          DO 52 K=1,KM
            RR(I,J,K) = R(L,I,J,K)
  52      CONTINUE
  51    CONTINUE
  50  CONTINUE

C..... Set up the tridiagonal coefficients for all three directions
          A = -EPSR
          C = -EPSR
          AC = EPSR*EPSR
C..... store and forward sweep to find new diagonal coefficient
          B(1) = 1.+2.*EPSR
          IJKMAX = MAX(IM,JM,KM)
          DO 11 I=2,IJKMAX
            B(I) = 1.+2.*EPSR
            B(I) = B(I) - AC/B(I-1)
  11    CONTINUE

C..... First, solve the tridiagonal matrix in the i-direction

C..... Forward sweep to find new right hand side coefficients
      DO 14 I=2,IM
        DO 13 J=1,JM
          DO 15 K=1,KM
            RR(I,J,K) = RR(I,J,K) - A*RR(I-1,J,K)/B(I-1)
  15      CONTINUE
  13    CONTINUE
  14  CONTINUE

C..... Backward sweep to solve for updated residuals
        DO 12 J=1,JM
          DO 18 K=1,KM
```

```
          RR(IM,J,K) = RR(IM,J,K)/B(IM)
 18       CONTINUE
 12     CONTINUE

        DO 16 I=1,IM-1
          II = IM-I
          DO 10 J=1,JM
            DO 17 K=1,KM
              RR(II,J,K) = (RR(II,J,K) - C*RR(II+1,J,K))/B(II)
 17         CONTINUE
 10       CONTINUE
 16     CONTINUE

C........ Second, solve the tridiagonal matrix in the j-direction
C.........  (If not semi-implicit)
       IF (SEMIIMP) GOTO 199
C........ Forward sweep to find new right hand side coefficients
       DO 24 J=2,JM
         DO 23 I=1,IM
           DO 25 K=1,KM
             RR(I,J,K) = RR(I,J,K) - A*RR(I,J-1,K)/B(J-1)
 25        CONTINUE
 23      CONTINUE
 24    CONTINUE

C......... Backward sweep to solve for updated residuals
        DO 22 I=1,IM
          DO 28 K=1,KM
            RR(I,JM,K) = RR(I,JM,K)/B(JM)
 28       CONTINUE
 22     CONTINUE

        DO 26 J=1,JM-1
          JJ = JM-J
          DO 21 I=1,IM
            DO 27 K=1,KM
              RR(I,JJ,K) = (RR(I,JJ,K) - C*RR(I,JJ+1,K))/B(JJ)
 27         CONTINUE
 21       CONTINUE
 26     CONTINUE
199   CONTINUE
C........ Finally, solve the tridiagonal matrix in the k-direction

C........ Forward sweep to find new right hand side coefficients
       DO 34 K=2,KM
         DO 33 I=1,IM
           DO 35 J=1,JM
```

```
            RR(I,J,K) = RR(I,J,K) - A*RR(I,J,K-1)/B(K-1)
 35       CONTINUE
 33     CONTINUE
 34   CONTINUE

C......... Backward sweep to solve for updated residuals
        DO 32 I=1,IM
          DO 38 J=1,JM
            RR(I,J,KM) = RR(I,J,KM)/B(KM)
 38       CONTINUE
 32     CONTINUE

        DO 36 K=1,KM-1
          KK = KM-K
          DO 30 I=1,IM
            DO 37 J=1,JM
              RR(I,J,KK) = (RR(I,J,KK) - C*RR(I,J,KK+1))/B(KK)
 37         CONTINUE
 30       CONTINUE
 36     CONTINUE

C......... Send the updated residual vector back in the flux vector
       DO 41 I=1,IM
         DO 42 J=1,JM
           DO 43 K=1,KM
             R(L,I,J,K) = RR(I,J,K)
 43        CONTINUE
 42      CONTINUE
 41    CONTINUE

 40    CONTINUE

       RETURN
       END
```

```
C
      SUBROUTINE LHS(A1,I,J)
C
C    THIS SUBROUTINE SETS UP THE LHS (KMPLICIT) OF EACH SEMI-KMPLICIT FLUX
C    BALANCE
C
C      |A C      |
C      |B A C    |
C      |  B A C  |  = LHS
C      |   . . . |
C      |      B A|
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      INCLUDE 'invert.inc'
      INCLUDE 'conven.inc'
      REAL COEFF(KSIZ),BCX(KSIZ),BCY(KSIZ),BCZ(KSIZ),
     &     BX(KSIZ),BY(KSIZ),BZ(KSIZ),CX(KSIZ),CY(KSIZ),CZ(KSIZ)
C
C
C---- Calculate all coefficients
      DO 8 K=1,KM
 8       COEFF(K) = A1*DT(I,J,k)
      DO 9 K=1,KM
         BY(K) =-.5*COEFF(K)*FACEY(2,I,J-1,K)
         BX(K) =-.5*COEFF(K)*FACEX(2,I,J-1,K)
         BZ(K) =-.5*COEFF(K)*FACEZ(2,I,J-1,K)

         CY(K) = .5*COEFF(K)*FACEY(2,I,J,K)
         CX(K) = .5*COEFF(K)*FACEX(2,I,J,K)
         CZ(K) = .5*COEFF(K)*FACEZ(2,I,J,K)
c        by(k) = 1.
c        bx(k) = 1.
c        bz(k) = 1.
c        cy(k) =-1.
c        cx(k) =-1.
c        cz(k) =-1.
 9     continue
       DO 7 K=1,KM
         BCY(K) = BY(K)+CY(K)
         BCX(K) = BX(K)+CX(K)
 7       BCZ(K) = BZ(K)+CZ(K)
C
C---- Choose proper forms

      IF (J .EQ. 1) GOTO 1
      IF (J .EQ. JM) GOTO 3
```

```
      GOTO 2

C
C---- Set Matrix elements at lower boundary first -- sweep through all I
 1    DO 10 K=1,KM
C
C
C---- Matrix A at lower boundary:  this is [C] modified by bc's(changes
C----                              in pressure terms) and addition of [I]
      AA(K,1,1) = 1.
      AA(K,1,2) = CX(K)
      AA(K,1,3) = CY(K)
      AA(K,1,4) = CZ(K)
      AA(K,1,5) = 0.
C
      AA(K,2,1) =
     & HW(K,J,2,1)*CZ(K) + GW(K,J,2,1)*CY(K) + FW(K,J,2,1)*CX(K)
     &                                        + 2*DPDU1(K,J)*BX(K)
      AA(K,2,2) = 1. +
     & HW(K,J,2,2)*CZ(K) + GW(K,J,2,2)*CY(K) + FW(K,J,2,2)*CX(K)
     &                                        + 2*DPDU2(K,J)*BX(K)
      AA(K,2,3) =
     &                    GW(K,J,2,3)*CY(K) + FW(K,J,2,3)*CX(K)
     &                                        + 2*DPDU3(K,J)*BX(K)
      AA(K,2,4) =
     & HW(K,J,2,4)*CZ(K)                      + FW(K,J,2,4)*CX(K)
     &                                        + 2*DPDU4(K,J)*BX(K)
      AA(K,2,5) =
     &                                         + FW(K,J,2,5)*CX(K)
     &                                        + 2*DPDU5*BX(K)
C
      AA(K,3,1) =
     & HW(K,J,3,1)*CZ(K) + GW(K,J,3,1)*CY(K) + FW(K,J,3,1)*CX(K)
     &                  + 2*DPDU1(K,J)*BY(K)
      AA(K,3,2) =
     &                     GW(K,J,3,2)*CY(K) + FW(K,J,3,2)*CX(K)
     &                  + 2*DPDU2(K,J)*BY(K)
      AA(K,3,3) = 1. +
     & HW(K,J,3,3)*CZ(K) + GW(K,J,3,3)*CY(K) + FW(K,J,3,3)*CX(K)
     &                  + 2*DPDU3(K,J)*BY(K)
      AA(K,3,4) =
     & HW(K,J,3,4)*CZ(K) + GW(K,J,3,4)*CY(K)
     &                  + 2*DPDU4(K,J)*BY(K)
      AA(K,3,5) =
     &                     GW(K,J,3,5)*CY(K)
     &                  + 2*DPDU5*BY(K)
C
```

```
      AA(K,4,1) =
     & HW(K,J,4,1)*CZ(K) + GW(K,J,4,1)*CY(K) + FW(K,J,4,1)*CX(K)
     &+2*DPDU1(K,J)*BZ(K)
      AA(K,4,2) =
     & HW(K,J,4,2)*CZ(K) +                     FW(K,J,4,2)*CX(K)
     &+2*DPDU2(K,J)*BZ(K)
      AA(K,4,3) =
     & HW(K,J,4,3)*CZ(K) + GW(K,J,4,3)*CY(K)
     &+2*DPDU3(K,J)*BZ(K)
      AA(K,4,4) = 1. +
     & HW(K,J,4,4)*CZ(K) + GW(K,J,4,4)*CY(K) + FW(K,J,4,4)*CX(K)
     &+2*DPDU4(K,J)*BZ(K)
      AA(K,4,5) =
     & HW(K,J,4,5)*CZ(K)
     &+2*DPDU5*BZ(K)
C
      AA(K,5,1) =
     & HW(K,J,5,1)*CZ(K) + GW(K,J,5,1)*CY(K) + FW(K,J,5,1)*CX(K)
      AA(K,5,2) =
     & HW(K,J,5,2)*CZ(K) + GW(K,J,5,2)*CY(K) + FW(K,J,5,2)*CX(K)
      AA(K,5,3) =
     & HW(K,J,5,3)*CZ(K) + GW(K,J,5,3)*CY(K) + FW(K,J,5,3)*CX(K)
      AA(K,5,4) =
     & HW(K,J,5,4)*CZ(K) + GW(K,J,5,4)*CY(K) + FW(K,J,5,4)*CX(K)
      AA(K,5,5) = 1. +
     & HW(K,J,5,5)*CZ(K) + GW(K,J,5,5)*CY(K) + FW(K,J,5,5)*CX(K)
C
C---- MATRIX C at lower boundary

      C(K,J,1,1) = 0.
      C(K,J,1,2) = CX(K)
      C(K,J,1,3) = CY(K)
      C(K,J,1,4) = CZ(K)
      C(K,J,1,5) = 0.
C
C---- d(RUV)/d(U1,U2,U3)
      C(K,J,2,1)=
     & HW(K,J+1,2,1)*CZ(K)+GW(K,J+1,2,1)*CY(K)+FW(K,J+1,2,1)*CX(K)
      C(K,J,2,2)=
     & HW(K,J+1,2,2)*CZ(K)+GW(K,J+1,2,2)*CY(K)+FW(K,J+1,2,2)*CX(K)
      C(K,J,2,3)=
     &                     GW(K,J+1,2,3)*CY(K)+FW(K,J+1,2,3)*CX(K)
      C(K,J,2,4)=
     & HW(K,J+1,2,4)*CZ(K)                   +FW(K,J+1,2,4)*CX(K)
      C(K,J,2,5)=
     &                                        +FW(K,J+1,2,5)*CX(K)
C
```

```
C---- d(RVV+P)/d(U1,U2,U3,U4)
      C(K,J,3,1)=
     & HW(K,J+1,3,1)*CZ(K)+GW(K,J+1,3,1)*CY(K)+FW(K,J+1,3,1)*CX(K)
      C(K,J,3,2)=
     &                     GW(K,J+1,3,2)*CY(K)+FW(K,J+1,3,2)*CX(K)
      C(K,J,3,3)=
     & HW(K,J+1,3,3)*CZ(K)+GW(K,J+1,3,3)*CY(K)+FW(K,J+1,3,3)*CX(K)
      C(K,J,3,4)=
     & HW(K,J+1,3,4)*CZ(K)+GW(K,J+1,3,4)*CY(K)
      C(K,J,3,5)=
     &                     GW(K,J+1,3,5)*CY(K)
C
C---- d(U3H)/d(U1,U2,U3,U4)
      C(K,J,4,1)=
     & HW(K,J+1,4,1)*CZ(K)+GW(K,J+1,4,1)*CY(K)+FW(K,J+1,4,1)*CX(K)
      C(K,J,4,2)=
     & HW(K,J+1,4,2)*CZ(K)+                    FW(K,J+1,4,2)*CX(K)
      C(K,J,4,3)=
     & HW(K,J+1,4,3)*CZ(K)+GW(K,J+1,4,3)*CY(K)
      C(K,J,4,4)=
     & HW(K,J+1,4,4)*CZ(K)+GW(K,J+1,4,4)*CY(K)+FW(K,J+1,4,4)*CX(K)
      C(K,J,4,5)=
     & HW(K,J+1,4,5)*CZ(K)
C
      C(K,J,5,1) =
     & HW(K,J+1,5,1)*CZ(K)+GW(K,J+1,5,1)*CY(K)+FW(K,J+1,5,1)*CX(K)
      C(K,J,5,2) =
     & HW(K,J+1,5,2)*CZ(K)+GW(K,J+1,5,2)*CY(K)+FW(K,J+1,5,2)*CX(K)
      C(K,J,5,3) =
     & HW(K,J+1,5,3)*CZ(K)+GW(K,J+1,5,3)*CY(K)+FW(K,J+1,5,3)*CX(K)
      C(K,J,5,4) =
     & HW(K,J+1,5,4)*CZ(K)+GW(K,J+1,5,4)*CY(K)+FW(K,J+1,5,4)*CX(K)
      C(K,J,5,5) =
     & HW(K,J+1,5,5)*CZ(K)+GW(K,J+1,5,5)*CY(K)+FW(K,J+1,5,5)*CX(K)
C
      DO 11 L=1,5
      DO 11 M=1,5
  11     B(K,L,M) = 0.
 10   CONTINUE
      GOTO 99
C
C---- Regular matrices ( 2 < j < JM-1 )
 2    DO 20 K=1,KM
C
C---------------------
C---- MATRIX A       -
C---------------------
```

```
C
C---- In 1st order acc. discret. interior AA contains smoothing & [I] only.
C---- These are added on at end.
      AA(K,1,1) = 1.
      AA(K,1,2) = BCX(K)
      AA(K,1,3) = BCY(K)
      AA(K,1,4) = BCZ(K)
      AA(K,1,5) = 0.
C
      AA(K,2,1) =
     & HW(K,J,2,1)*BCZ(K) + GW(K,J,2,1)*BCY(K) + FW(K,J,2,1)*BCX(K)
      AA(K,2,2) = 1. +
     & HW(K,J,2,2)*BCZ(K) + GW(K,J,2,2)*BCY(K) + FW(K,J,2,2)*BCX(K)
      AA(K,2,3) =
     &                     GW(K,J,2,3)*BCY(K) + FW(K,J,2,3)*BCX(K)
      AA(K,2,4) =
     & HW(K,J,2,4)*BCZ(K)                     + FW(K,J,2,4)*BCX(K)
      AA(K,2,5) =
     &                                        + FW(K,J,2,5)*BCX(K)
C
      AA(K,3,1) =
     & HW(K,J,3,1)*BCZ(K) + GW(K,J,3,1)*BCY(K) + FW(K,J,3,1)*BCX(K)
      AA(K,3,2) =
     &                     GW(K,J,3,2)*BCY(K) + FW(K,J,3,2)*BCX(K)
      AA(K,3,3) = 1. +
     & HW(K,J,3,3)*BCZ(K) + GW(K,J,3,3)*BCY(K) + FW(K,J,3,3)*BCX(K)
      AA(K,3,4) =
     & HW(K,J,3,4)*BCZ(K) + GW(K,J,3,4)*BCY(K)
      AA(K,3,5) =
     &                     GW(K,J,3,5)*BCY(K)
C
      AA(K,4,1) =
     & HW(K,J,4,1)*BCZ(K) + GW(K,J,4,1)*BCY(K) + FW(K,J,4,1)*BCX(K)
      AA(K,4,2) =
     & HW(K,J,4,2)*BCZ(K) +                      FW(K,J,4,2)*BCX(K)
      AA(K,4,3) =
     & HW(K,J,4,3)*BCZ(K) + GW(K,J,4,3)*BCY(K)
      AA(K,4,4) = 1. +
     & HW(K,J,4,4)*BCZ(K) + GW(K,J,4,4)*BCY(K) + FW(K,J,4,4)*BCX(K)
      AA(K,4,5) =
     & HW(K,J,4,5)*BCZ(K)
C
      AA(K,5,1) =
     & HW(K,J,5,1)*BCZ(K) + GW(K,J,5,1)*BCY(K) + FW(K,J,5,1)*BCX(K)
      AA(K,5,2) =
     & HW(K,J,5,2)*BCZ(K) + GW(K,J,5,2)*BCY(K) + FW(K,J,5,2)*BCX(K)
      AA(K,5,3) =
```

```
     & HW(K,J,5,3)*BCZ(K) + GW(K,J,5,3)*BCY(K) + FW(K,J,5,3)*BCX(K)
      AA(K,5,4) =
     & HW(K,J,5,4)*BCZ(K) + GW(K,J,5,4)*BCY(K) + FW(K,J,5,4)*BCX(K)
      AA(K,5,5) = 1. +
     & HW(K,J,5,5)*BCZ(K) + GW(K,J,5,5)*BCY(K) + FW(K,J,5,5)*BCX(K)
C
C-------------------------------------------------------
C---- "Regular" submatrices [B] & [C] for 1 < j < JM-1  -
C-------------------------------------------------------
C---- MATRIX C

      C(K,J,1,1) = 0.
      C(K,J,1,2) = CX(K)
      C(K,J,1,3) = CY(K)
      C(K,J,1,4) = CZ(K)
      C(K,J,1,5) = 0.
C
C---- d(RUV)/d(U1,U2,U3)
      C(K,J,2,1)=
     & HW(K,J+1,2,1)*CZ(K)+GW(K,J+1,2,1)*CY(K)+FW(K,J+1,2,1)*CX(K)
      C(K,J,2,2)=
     & HW(K,J+1,2,2)*CZ(K)+GW(K,J+1,2,2)*CY(K)+FW(K,J+1,2,2)*CX(K)
      C(K,J,2,3)=
     &                     GW(K,J+1,2,3)*CY(K)+FW(K,J+1,2,3)*CX(K)
      C(K,J,2,4)=
     & HW(K,J+1,2,4)*CZ(K)                   +FW(K,J+1,2,4)*CX(K)
      C(K,J,2,5)=
     &                                        +FW(K,J+1,2,5)*CX(K)
C
C---- d(RVV+P)/d(U1,U2,U3,U4)
      C(K,J,3,1)=
     & HW(K,J+1,3,1)*CZ(K)+GW(K,J+1,3,1)*CY(K)+FW(K,J+1,3,1)*CX(K)
      C(K,J,3,2)=
     &                     GW(K,J+1,3,2)*CY(K)+FW(K,J+1,3,2)*CX(K)
      C(K,J,3,3)=
     & HW(K,J+1,3,3)*CZ(K)+GW(K,J+1,3,3)*CY(K)+FW(K,J+1,3,3)*CX(K)
      C(K,J,3,4)=
     & HW(K,J+1,3,4)*CZ(K)+GW(K,J+1,3,4)*CY(K)
      C(K,J,3,5)=
     &                     GW(K,J+1,3,5)*CY(K)
C
C---- d(U3H)/d(U1,U2,U3,U4)
      C(K,J,4,1)=
     & HW(K,J+1,4,1)*CZ(K)+GW(K,J+1,4,1)*CY(K)+FW(K,J+1,4,1)*CX(K)
      C(K,J,4,2)=
     & HW(K,J+1,4,2)*CZ(K)+                    FW(K,J+1,4,2)*CX(K)
      C(K,J,4,3)=
```

```
     & HW(K,J+1,4,3)*CZ(K)+GW(K,J+1,4,3)*CY(K)
      C(K,J,4,4)=
     & HW(K,J+1,4,4)*CZ(K)+GW(K,J+1,4,4)*CY(K)+FW(K,J+1,4,4)*CX(K)
      C(K,J,4,5)=
     & HW(K,J+1,4,5)*CZ(K)
C
      C(K,J,5,1) =
     & HW(K,J+1,5,1)*CZ(K)+GW(K,J+1,5,1)*CY(K)+FW(K,J+1,5,1)*CX(K)
      C(K,J,5,2) =
     & HW(K,J+1,5,2)*CZ(K)+GW(K,J+1,5,2)*CY(K)+FW(K,J+1,5,2)*CX(K)
      C(K,J,5,3) =
     & HW(K,J+1,5,3)*CZ(K)+GW(K,J+1,5,3)*CY(K)+FW(K,J+1,5,3)*CX(K)
      C(K,J,5,4) =
     & HW(K,J+1,5,4)*CZ(K)+GW(K,J+1,5,4)*CY(K)+FW(K,J+1,5,4)*CX(K)
      C(K,J,5,5) =
     & HW(K,J+1,5,5)*CZ(K)+GW(K,J+1,5,5)*CY(K)+FW(K,J+1,5,5)*CX(K)
C
C---------------------
C---- MATRIX B       -
C---------------------
C
      B(K,1,1) = 0.
      B(K,1,2) = BX(K)
      B(K,1,3) = BY(K)
      B(K,1,4) = BZ(K)
      B(K,1,5) = 0.
C
      B(K,2,1)=
     & HW(K,J-1,2,1)*BZ(K)+GW(K,J-1,2,1)*BY(K)+FW(K,J-1,2,1)*BX(K)
      B(K,2,2)=
     & HW(K,J-1,2,2)*BZ(K)+GW(K,J-1,2,2)*BY(K)+FW(K,J-1,2,2)*BX(K)
      B(K,2,3)=
     &                    GW(K,J-1,2,3)*BY(K)+FW(K,J-1,2,3)*BX(K)
      B(K,2,4)=
     & HW(K,J-1,2,4)*BZ(K)                   +FW(K,J-1,2,4)*BX(K)
      B(K,2,5)=
     &                                        +FW(K,J-1,2,5)*BX(K)
C
      B(K,3,1)=
     & HW(K,J-1,3,1)*BZ(K)+GW(K,J-1,3,1)*BY(K)+FW(K,J-1,3,1)*BX(K)
      B(K,3,2)=
     &                    GW(K,J-1,3,2)*BY(K)+FW(K,J-1,3,2)*BX(K)
      B(K,3,3)=
     & HW(K,J-1,3,3)*BZ(K)+GW(K,J-1,3,3)*BY(K)+FW(K,J-1,3,3)*BX(K)
      B(K,3,4)=
     & HW(K,J-1,3,4)*BZ(K)+GW(K,J-1,3,4)*BY(K)
      B(K,3,5)=
```

```
     &                    GW(K,J-1,3,5)*BY(K)
C
      B(K,4,1)=
     & HW(K,J-1,4,1)*BZ(K)+GW(K,J-1,4,1)*BY(K)+FW(K,J-1,4,1)*BX(K)
      B(K,4,2)=
     & HW(K,J-1,4,2)*BZ(K)+                    FW(K,J-1,4,2)*BX(K)
      B(K,4,3)=
     & HW(K,J-1,4,3)*BZ(K)+GW(K,J-1,4,3)*BY(K)
      B(K,4,4)=
     & HW(K,J-1,4,4)*BZ(K)+GW(K,J-1,4,4)*BY(K)+FW(K,J-1,4,4)*BX(K)
      B(K,4,5)=
     & HW(K,J-1,4,5)*BZ(K)
C
      B(K,5,1) =
     & HW(K,J-1,5,1)*BZ(K)+GW(K,J-1,5,1)*BY(K)+FW(K,J-1,5,1)*BX(K)
      B(K,5,2) =
     & HW(K,J-1,5,2)*BZ(K)+GW(K,J-1,5,2)*BY(K)+FW(K,J-1,5,2)*BX(K)
      B(K,5,3) =
     & HW(K,J-1,5,3)*BZ(K)+GW(K,J-1,5,3)*BY(K)+FW(K,J-1,5,3)*BX(K)
      B(K,5,4) =
     & HW(K,J-1,5,4)*BZ(K)+GW(K,J-1,5,4)*BY(K)+FW(K,J-1,5,4)*BX(K)
      B(K,5,5) =
     & HW(K,J-1,5,5)*BZ(K)+GW(K,J-1,5,5)*BY(K)+FW(K,J-1,5,5)*BX(K)
C
 20   CONTINUE
      GOTO 99
C
C----------------------------
C---- Now do upper boundary -
C----------------------------
 3    DO 30 K=1,KM
C
C---- Matrix A at upper boundary:  for supersonic flow this is =[A]interior
      AA(K,1,1) = 1.
      AA(K,1,2) = BX(K)
      AA(K,1,3) = BY(K)
      AA(K,1,4) = BZ(K)
      AA(K,1,5) = 0.
C
      AA(K,2,1) =
     & HW(K,J,2,1)*BZ(K) + GW(K,J,2,1)*BY(K) + FW(K,J,2,1)*BX(K)
      AA(K,2,2) = 1. +
     & HW(K,J,2,2)*BZ(K) + GW(K,J,2,2)*BY(K) + FW(K,J,2,2)*BX(K)
      AA(K,2,3) =
     &                    GW(K,J,2,3)*BY(K) + FW(K,J,2,3)*BX(K)
      AA(K,2,4) =
     & HW(K,J,2,4)*BZ(K)                    + FW(K,J,2,4)*BX(K)
```

```
      AA(K,2,5) =
     &                                        + FW(K,J,2,5)*BX(K)
C
      AA(K,3,1) =
     & HW(K,J,3,1)*BZ(K) + GW(K,J,3,1)*BY(K) + FW(K,J,3,1)*BX(K)
      AA(K,3,2) =
     &                     GW(K,J,3,2)*BY(K) + FW(K,J,3,2)*BX(K)
      AA(K,3,3) = 1. +
     & HW(K,J,3,3)*BZ(K) + GW(K,J,3,3)*BY(K) + FW(K,J,3,3)*BX(K)
      AA(K,3,4) =
     & HW(K,J,3,4)*BZ(K) + GW(K,J,3,4)*BY(K)
      AA(K,3,5) =
     &                     GW(K,J,3,5)*BY(K)
C
      AA(K,4,1) =
     & HW(K,J,4,1)*BZ(K) + GW(K,J,4,1)*BY(K) + FW(K,J,4,1)*BX(K)
      AA(K,4,2) =
     & HW(K,J,4,2)*BZ(K) +                     FW(K,J,4,2)*BX(K)
      AA(K,4,3) =
     & HW(K,J,4,3)*BZ(K) + GW(K,J,4,3)*BY(K)
      AA(K,4,4) = 1. +
     & HW(K,J,4,4)*BZ(K) + GW(K,J,4,4)*BY(K) + FW(K,J,4,4)*BX(K)
      AA(K,4,5) =
     & HW(K,J,4,5)*BZ(K)
C
      AA(K,5,1) =
     & HW(K,J,5,1)*BZ(K) + GW(K,J,5,1)*BY(K) + FW(K,J,5,1)*BX(K)
      AA(K,5,2) =
     & HW(K,J,5,2)*BZ(K) + GW(K,J,5,2)*BY(K) + FW(K,J,5,2)*BX(K)
      AA(K,5,3) =
     & HW(K,J,5,3)*BZ(K) + GW(K,J,5,3)*BY(K) + FW(K,J,5,5)*BX(K)
      AA(K,5,4) =
     & HW(K,J,5,4)*BZ(K) + GW(K,J,5,4)*BY(K) + FW(K,J,5,4)*BX(K)
      AA(K,5,5) = 1. +
     & HW(K,J,5,5)*BZ(K) + GW(K,J,5,5)*BY(K) + FW(K,J,5,5)*BX(K)

C
C---- Matrix B at upper boundary
C
      B(K,1,1) = 0.
      B(K,1,2) = BX(K)
      B(K,1,3) = BY(K)
      B(K,1,4) = BZ(K)
      B(K,1,5) = 0.
C
      B(K,2,1)=
     & HW(K,J-1,2,1)*BZ(K)+GW(K,J-1,2,1)*BY(K)+FW(K,J-1,2,1)*BX(K)
```

```
      B(K,2,2)=
     & HW(K,J-1,2,2)*BZ(K)+GW(K,J-1,2,2)*BY(K)+FW(K,J-1,2,2)*BX(K)
      B(K,2,3)=
     &                     GW(K,J-1,2,3)*BY(K)+FW(K,J-1,2,3)*BX(K)
      B(K,2,4)=
     & HW(K,J-1,2,4)*BZ(K)                     +FW(K,J-1,2,4)*BX(K)
      B(K,2,5)=
     &                                         +FW(K,J-1,2,5)*BX(K)
C
      B(K,3,1)=
     & HW(K,J-1,3,1)*BZ(K)+GW(K,J-1,3,1)*BY(K)+FW(K,J-1,3,1)*BX(K)
      B(K,3,2)=
     &                     GW(K,J-1,3,2)*BY(K)+FW(K,J-1,3,2)*BX(K)
      B(K,3,3)=
     & HW(K,J-1,3,3)*BZ(K)+GW(K,J-1,3,3)*BY(K)+FW(K,J-1,3,3)*BX(K)
      B(K,3,4)=
     & HW(K,J-1,3,4)*BZ(K)+GW(K,J-1,3,4)*BY(K)
      B(K,3,5)=
     &                     GW(K,J-1,3,5)*BY(K)
C
      B(K,4,1)=
     & HW(K,J-1,4,1)*BZ(K)+GW(K,J-1,4,1)*BY(K)+FW(K,J-1,4,1)*BX(K)
      B(K,4,2)=
     & HW(K,J-1,4,2)*BZ(K)+                    FW(K,J-1,4,2)*BX(K)
      B(K,4,3)=
     & HW(K,J-1,4,3)*BZ(K)+GW(K,J-1,4,3)*BY(K)
      B(K,4,4)=
     & HW(K,J-1,4,4)*BZ(K)+GW(K,J-1,4,4)*BY(K)+FW(K,J-1,4,4)*BX(K)
      B(K,4,5)=
     & HW(K,J-1,4,5)*BZ(K)
C
      B(K,5,1) =
     & HW(K,J-1,5,1)*BZ(K) + GW(K,J-1,5,1)*BY(K) + FW(K,J-1,5,1)*BX(K)
      B(K,5,2) =
     & HW(K,J-1,5,2)*BZ(K) + GW(K,J-1,5,2)*BY(K) + FW(K,J-1,5,2)*BX(K)
      B(K,5,3) =
     & HW(K,J-1,5,3)*BZ(K) + GW(K,J-1,5,3)*BY(K) + FW(K,J-1,5,3)*BX(K)
      B(K,5,4) =
     & HW(K,J-1,5,4)*BZ(K) + GW(K,J-1,5,4)*BY(K) + FW(K,J-1,5,4)*BX(K)
      B(K,5,5) =
     & HW(K,J-1,5,5)*BZ(K) + GW(K,J-1,5,5)*BY(K) + FW(K,J-1,5,5)*BX(K)
      DO 31 L=1,5
      DO 31 M=1,5
         C(K,J,L,M) = 0.
  31     C(K,J+1,L,M) = 0.

  30  CONTINUE
```

```
C
C
 99   CONTINUE
      RETURN
      END
C
      SUBROUTINE LHSSMO(A1,I,J)
C
C     THIS SUBROUTINE ADDS THE IMPLICIT SMOOTHING TERMS TO THE LHS
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      INCLUDE 'invert.inc'
      INCLUDE 'conven.inc'
      COMMON/DTE/DTEXP(ISIZ,JSIZ,KSIZ)
C
      REAL ST(KSIZ)
C
c      write(6,*) 'i,j=',i,j
      DO 10 K=1,KM
        ST(K) = A1*MUSI*(DT(I,J,K)/DTEXP(I,J,K))
 10   CONTINUE
C
      DO 20 K=1,KM
      AA(K,1,1) = AA(K,1,1) + 2*ST(K)
      AA(K,2,2) = AA(K,2,2) + 2*ST(K)
      AA(K,3,3) = AA(K,3,3) + 2*ST(K)
      AA(K,4,4) = AA(K,4,4) + 2*ST(K)
      AA(K,5,5) = AA(K,5,5) + 2*ST(K)
      B(K,1,1) = -ST(K)
      B(K,2,2) = B(K,2,2) - ST(K)
      B(K,3,3) = B(K,3,3) - ST(K)
      B(K,4,4) = B(K,4,4) - ST(K)
      B(K,5,5) = B(K,5,5) - ST(K)
      C(K,J,1,1) = -ST(K)
      C(K,J,2,2) = C(K,J,2,2) - ST(K)
      C(K,J,3,3) = C(K,J,3,3) - ST(K)
      C(K,J,4,4) = C(K,J,4,4) - ST(K)
      C(K,J,5,5) = C(K,J,5,5) - ST(K)
 20   CONTINUE
C
      RETURN
      END
      SUBROUTINE CONVEN(I)
C
C---- Calculate some convenient quantities for all I and J
C-------------------------------------------------------------
```

```
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      INCLUDE 'conven.inc'
      REAL ORHO(KSIZ)
      REAL DUDU1(KSIZ,JSIZ),DUDU2(KSIZ,JSIZ),
     &     DVDU1(KSIZ,JSIZ),DVDU3(KSIZ,JSIZ),OPR,
     &     DRUDU1(KSIZ,JSIZ),DRUDU2(KSIZ,JSIZ),DRUDU3(KSIZ,JSIZ),
     &     UU(KSIZ,JSIZ),VV(KSIZ,JSIZ),WW(KSIZ,JSIZ),H(KSIZ,JSIZ)

C
      GM1 = GAM - 1
      OPR         = 1/PR
      DPDU5       = GM1
C
      DO 1003 J=1,JM
c      write(6,*) '[conven] j=',j
      DO 1001 K=1,KM
         ORHO(K) = 1/U(1,I,J,K)
         UU(K,J) = U(2,I,J,K)*ORHO(K)
         VV(K,J) = U(3,I,J,K)*ORHO(K)
         WW(K,J) = U(4,I,J,K)*ORHO(K)
1001     H(K,J) = (U(5,I,J,K) + P(I,J,K))*ORHO(K)

      DO 1002 K=1,KM
C
         DUDU1(K,J) = -UU(K,J)*ORHO(K)
         DUDU2(K,J) = ORHO(K)
C
         DVDU1(K,J) = -VV(K,J)*ORHO(K)
         DVDU3(K,J) = ORHO(K)
C
C------ !d(RUV)/dR
         DRUDU1(K,J) = -UU(K,J)*VV(K,J)
C------ !d(RUV)/d(U2)
         DRUDU2(K,J) =  VV(K,J)
C------ !d(RUV)/d(U3)
         DRUDU3(K,J) =  UU(K,J)
C
         DPDU1(K,J) = .5*GM1*(UU(K,J)**2+VV(K,J)**2+WW(K,J)**2)
         DPDU2(K,J) = -GM1*UU(K,J)
         DPDU3(K,J) = -GM1*VV(K,J)
1002     DPDU4(K,J) = -GM1*WW(K,J)
C
      DO 1003 K=1,KM
         DHDU1(K,J) = (DPDU1(K,J) - H(K,J))*ORHO(K)
         DHDU2(K,J) =  DPDU2(K,J)*ORHO(K)
```

```
         DHDU3(K,J) =  DPDU3(K,J)*ORHO(K)
         DHDU4(K,J) =  DPDU4(K,J)*ORHO(K)
C------- != GAM/rho(K,J)
1003     DHDU5(K,J) = (DPDU5 + 1)*ORHO(K)

C
C
c      write(6,*) '[conven2] before dgfw'
      DO 10 J=1,JM
      DO 10 K=1,KM
C
C-------------------------------------------------------------
C---- dG/dW
C-------------------------------------------------------------
C
C
C
C---- d(RUV)/d(U1,U2,U3)
      GW(K,J,2,1) =-UU(K,J)*VV(K,J)
      GW(K,J,2,2) = VV(K,J)
      GW(K,J,2,3) = UU(K,J)
      GW(K,J,2,4) = 0.
      GW(K,J,2,5) = 0.
C
C---- d(RVV+P)/d(U1,U2,U3,U4,U5)
      GW(K,J,3,1) = -VV(K,J)**2 + DPDU1(K,J)
      GW(K,J,3,2) = DPDU2(K,J)
      GW(K,J,3,3) = 2*VV(K,J) + DPDU3(K,J)
      GW(K,J,3,4) = DPDU4(K,J)
      GW(K,J,3,5) = DPDU5
C
C---- d(RVW)/d(U1,U2,U3,U4,U5)
      GW(K,J,4,1) =-WW(K,J)*VV(K,J)
      GW(K,J,4,2) = 0.
      GW(K,J,4,3) = WW(K,J)
      GW(K,J,4,4) = VV(K,J)
      GW(K,J,4,5) = 0.
C
C---- d(U3H)/d(U1,U2,U3,U4,U5)
      GW(K,J,5,1) = VV(K,J)*(DPDU1(K,J) - H(K,J))
      GW(K,J,5,2) = VV(K,J)*DPDU2(K,J)
      GW(K,J,5,3) = H(K,J) + VV(K,J)*DPDU3(K,J)
      GW(K,J,5,4) = VV(K,J)*DPDU4(K,J)
      GW(K,J,5,5) = VV(K,J)*(1. + DPDU5)
C
C
C
```

```
C
C-------------------------------------------------------
C---- dF/dW
C-------------------------------------------------------
C
C---- d(RUU+P)/d(U1,U2,U3,U4,U5)
      FW(K,J,2,1) =-UU(K,J)**2 + DPDU1(K,J)
      FW(K,J,2,2) = 2*UU(K,J) + DPDU2(K,J)
      FW(K,J,2,3) = DPDU3(K,J)
      FW(K,J,2,4) = DPDU4(K,J)
      FW(K,J,2,5) = DPDU5
C
C---- d(RUV)/d(U1,U2,U3)
      FW(K,J,3,1) =-UU(K,J)*VV(K,J)
      FW(K,J,3,2) = VV(K,J)
      FW(K,J,3,3) = UU(K,J)
      FW(K,J,3,4) = 0.
      FW(K,J,3,5) = 0.
C
C---- d(RUW)/d(U1,U2,U3)
      FW(K,J,4,1) =-UU(K,J)*WW(K,J)
      FW(K,J,4,2) = WW(K,J)
      FW(K,J,4,3) = 0.
      FW(K,J,4,4) = UU(K,J)
      FW(K,J,4,5) = 0.
C
C---- d((E+P)U)/d(U1,U2,U3,U4)
      FW(K,J,5,1) = UU(K,J)*(DPDU1(K,J) - H(K,J))
      FW(K,J,5,2) = H(K,J) + UU(K,J)*DPDU2(K,J)
      FW(K,J,5,3) = UU(K,J)*DPDU3(K,J)
      FW(K,J,5,4) = UU(K,J)*DPDU4(K,J)
      FW(K,J,5,5) = UU(K,J)*(1. + DPDU5)
C
C-------------------------------------------------------
C---- dH/dW
C-------------------------------------------------------
C
C
C
C---- d(RUW)/d(U1,U2,U3,U4,U5)
      HW(K,J,2,1) =-UU(K,J)*WW(K,J)
      HW(K,J,2,2) = WW(K,J)
      HW(K,J,2,3) = 0.
      HW(K,J,2,4) = UU(K,J)
      HW(K,J,2,5) = 0.
C
C---- d(RVW)/d(U1,U2,U3,U4,U5)
```

```
      HW(K,J,3,1) =-VV(K,J)*WW(K,J)
      HW(K,J,3,2) = 0.
      HW(K,J,3,3) = WW(K,J)
      HW(K,J,3,4) = VV(K,J)
      HW(K,J,3,5) = 0.
C
C---- d(RWW+P)/d(U1,U2,U3,U4,U5)
      HW(K,J,4,1) =-WW(K,J)**2 + DPDU1(K,J)
      HW(K,J,4,2) = DPDU2(K,J)
      HW(K,J,4,3) = DPDU3(K,J)
      HW(K,J,4,4) = 2*WW(K,J) + DPDU4(K,J)
      HW(K,J,4,5) = DPDU5
C
C---- d((E+P)W)/d(U1,U2,U3,U4,U5)
      HW(K,J,5,1) = WW(K,J)*(DPDU1(K,J) - H(K,J))
      HW(K,J,5,2) = WW(K,J)*DPDU2(K,J)
      HW(K,J,5,3) = WW(K,J)*DPDU3(K,J)
      HW(K,J,5,4) = H(K,J) + WW(K,J)*DPDU4(K,J)
      HW(K,J,5,5) = WW(K,J)*(1. + DPDU5)
C
 10   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE NSLHS(A1,I,J)
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      INCLUDE 'invert.inc'
      INCLUDE 'nslhs.inc'
      REAL ODIAG(KSIZ),DIAG(KSIZ)
C
C    THIS SUBROUTINE ADDS THE NAVIER-STOKES TERMS TO THE LHS
C
C      |I C      |
C      |B I C    |
C      |  B I C  |  = LHS
C      |   . . . |
C      |      B I|
C
C---- Calculate all coefficients
      DO 5 K=1,KM
         ODIAG(K) = -A1*DT(I,J,K)*MUL(I,J,K)
 5       DIAG(K)  = -2.*ODIAG(K)
C
C-----------------------------------------
C---- Submatrices [A], [B] & [C] for all J  -
C-----------------------------------------
C
      DO 10 K=1,KM
C
C---- MATRIX C
C
      IF(J .EQ. JM) GOTO 2
      C(K,J,2,1) = C(K,J,2,1) + ODIAG(K)*VS(K,J+1,2,1)
      C(K,J,2,2) = C(K,J,2,2) + ODIAG(K)*VS(K,J+1,2,2)
      C(K,J,2,3) = C(K,J,2,3) + ODIAG(K)*VS(K,J+1,2,3)
      C(K,J,2,4) = C(K,J,2,4) + ODIAG(K)*VS(K,J+1,2,4)
C
      C(K,J,3,1) = C(K,J,3,1) + ODIAG(K)*VS(K,J+1,3,1)
      C(K,J,3,2) = C(K,J,3,2) + ODIAG(K)*VS(K,J+1,3,2)
      C(K,J,3,3) = C(K,J,3,3) + ODIAG(K)*VS(K,J+1,3,3)
      C(K,J,3,4) = C(K,J,3,4) + ODIAG(K)*VS(K,J+1,3,4)
C
      C(K,J,4,1) = C(K,J,4,1) + ODIAG(K)*VS(K,J+1,4,1)
      C(K,J,4,2) = C(K,J,4,2) + ODIAG(K)*VS(K,J+1,4,2)
      C(K,J,4,3) = C(K,J,4,3) + ODIAG(K)*VS(K,J+1,4,3)
      C(K,J,4,4) = C(K,J,4,4) + ODIAG(K)*VS(K,J+1,4,4)
C
      C(K,J,5,1) = C(K,J,5,1) + ODIAG(K)*VS(K,J+1,5,1)
      C(K,J,5,2) = C(K,J,5,2) + ODIAG(K)*VS(K,J+1,5,2)
```

```
      C(K,J,5,3) = C(K,J,5,3) + ODIAG(K)*VS(K,J+1,5,3)
      C(K,J,5,4) = C(K,J,5,4) + ODIAG(K)*VS(K,J+1,5,4)
      C(K,J,5,5) = C(K,J,5,5) + ODIAG(K)*VS(K,J+1,5,5)
C
C-------------------------------------------------
C---- MATRIX A      U(j+1)   - 2*U(j)   + U(j-1)  -
C-------------------------------------------------
 2    CONTINUE
      AA(K,2,1) = AA(K,2,1) + DIAG(K)*VS(K,J,2,1)
      AA(K,2,2) = AA(K,2,2) + DIAG(K)*VS(K,J,2,2)
      AA(K,2,3) = AA(K,2,3) + DIAG(K)*VS(K,J,2,3)
      AA(K,2,4) = AA(K,2,4) + DIAG(K)*VS(K,J,2,4)
C
      AA(K,3,1) = AA(K,3,1) + DIAG(K)*VS(K,J,3,1)
      AA(K,3,2) = AA(K,3,2) + DIAG(K)*VS(K,J,3,2)
      AA(K,3,3) = AA(K,3,3) + DIAG(K)*VS(K,J,3,3)
      AA(K,3,4) = AA(K,3,4) + DIAG(K)*VS(K,J,3,4)
C
      AA(K,4,1) = AA(K,4,1) + DIAG(K)*VS(K,J,4,1)
      AA(K,4,2) = AA(K,4,2) + DIAG(K)*VS(K,J,4,2)
      AA(K,4,3) = AA(K,4,3) + DIAG(K)*VS(K,J,4,3)
      AA(K,4,4) = AA(K,4,4) + DIAG(K)*VS(K,J,4,4)
C
      AA(K,5,1) = AA(K,5,1) + DIAG(K)*VS(K,J,5,1)
      AA(K,5,2) = AA(K,5,2) + DIAG(K)*VS(K,J,5,2)
      AA(K,5,3) = AA(K,5,3) + DIAG(K)*VS(K,J,5,3)
      AA(K,5,4) = AA(K,5,4) + DIAG(K)*VS(K,J,5,4)
      AA(K,5,5) = AA(K,5,5) + DIAG(K)*VS(K,J,5,5)
C
C--------------------
C---- MATRIX B      -
C--------------------
C
      IF (J .EQ. 1) GOTO 3
      B(K,2,1) = B(K,2,1) + ODIAG(K)*VS(K,J-1,2,1)
      B(K,2,2) = B(K,2,2) + ODIAG(K)*VS(K,J-1,2,2)
      B(K,2,3) = B(K,2,3) + ODIAG(K)*VS(K,J-1,2,3)
      B(K,2,4) = B(K,2,4) + ODIAG(K)*VS(K,J-1,2,4)
C
      B(K,3,1) = B(K,3,1) + ODIAG(K)*VS(K,J-1,3,1)
      B(K,3,2) = B(K,3,2) + ODIAG(K)*VS(K,J-1,3,2)
      B(K,3,3) = B(K,3,3) + ODIAG(K)*VS(K,J-1,3,3)
      B(K,3,4) = B(K,3,4) + ODIAG(K)*VS(K,J-1,3,4)
C
      B(K,4,1) = B(K,4,1) + ODIAG(K)*VS(K,J-1,4,1)
      B(K,4,2) = B(K,4,2) + ODIAG(K)*VS(K,J-1,4,2)
      B(K,4,3) = B(K,4,3) + ODIAG(K)*VS(K,J-1,4,3)
```

```
      B(K,4,4) = B(K,4,4) + ODIAG(K)*VS(K,J-1,4,4)
C
      B(K,5,1) = B(K,5,1) + ODIAG(K)*VS(K,J-1,5,1)
      B(K,5,2) = B(K,5,2) + ODIAG(K)*VS(K,J-1,5,2)
      B(K,5,3) = B(K,5,3) + ODIAG(K)*VS(K,J-1,5,3)
      B(K,5,4) = B(K,5,4) + ODIAG(K)*VS(K,J-1,5,4)
      B(K,5,5) = B(K,5,5) + ODIAG(K)*VS(K,J-1,5,5)
C
  3   CONTINUE
 10   CONTINUE

      RETURN
      END
      SUBROUTINE CONVNS(I)
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      INCLUDE 'nslhs.inc'
      INCLUDE 'conven.inc'
      REAL TD,ORHO,MOM,VSX,VSY,VSZ
C
      DO 10 K=1,KM
      DO 10 J=1,JM
C
C
      MOM = (MUL(I,J,K)/PR+MUT(I,J,K)/PRT)/(MUL(I,J,K)+MUT(I,J,K))
      ORHO = 1/U(1,I,J,K)
C
C---- Approximations
      UAVE = U(2,I,J,K)*ORHO
      VAVE = U(3,I,J,K)*ORHO
      WAVE = U(4,I,J,K)*ORHO

      FX = .5*(FACEX(2,I,J,K)+FACEX(2,I,J-1,K))
      FY = .5*(FACEY(2,I,J,K)+FACEY(2,I,J-1,K))
      FZ = .5*(FACEZ(2,I,J,K)+FACEZ(2,I,J-1,K))

      EX = .5*(ETX(I,J,K)+ETX(I,J-1,K))
      EY = .5*(ETY(I,J,K)+ETY(I,J-1,K))
      EZ = .5*(ETZ(I,J,K)+ETZ(I,J-1,K))
C
      DUDU1 = -U(2,I,J,K)*ORHO*ORHO
      DVDU1 = -U(3,I,J,K)*ORHO*ORHO
      DWDU1 = -U(4,I,J,K)*ORHO*ORHO
      DUDU2 = ORHO
      DVDU3 = ORHO
      DWDU4 = ORHO
```

```
C      DUADU1 = .5*DUDU1
C      DUADU2 = .5*DUDU2
C      DVADU1 = .5*DVDU1       This does not result in correct linearization
C      DVADU3 = .5*DVDU3       when terms at j+1,j,j-1 are all calculated in
C      DWADU1 = .5*DWDU1       one sweep.  Better to assume variation of average
C      DWADU4 = .5*DWDU4       quantities is small from iteration to iteration..

       DUADU1 = 0.
       DUADU2 = 0.
       DVADU1 = 0.
       DVADU3 = 0.
       DWADU1 = 0.
       DWADU4 = 0.

       TD = 2./3.
C
       TXYDU1 = DUDU1*EY + DVDU1*EX
       TXYDU2 = DUDU2*EY
       TXYDU3 = DVDU3*EX

       TXZDU1 = DUDU1*EZ + DWDU1*EX
       TXZDU2 = DUDU2*EZ
       TXZDU4 = DWDU4*EX

       TYZDU1 = DVDU1*EZ + DWDU1*EY
       TYZDU3 = DVDU3*EZ
       TYZDU4 = DWDU4*EY

       TXXDU1 = TD*(2*DUDU1*EX - DVDU1*EY - DWDU1*EZ)
       TXXDU2 = TD*(2*DUDU2*EX                      )
       TXXDU3 = TD*(           - DVDU3*EY)
       TXXDU4 = TD*(                      - DWDU4*EZ)

       TYYDU1 = TD*(2*DVDU1*EY - DUDU1*EX - DWDU1*EZ)
       TYYDU2 = TD*(           - DUDU2*EX           )
       TYYDU3 = TD*(2*DVDU3*EY                      )
       TYYDU4 = TD*(                      - DWDU4*EZ)

       TZZDU1 = TD*(2*DWDU1*EZ - DVDU1*EY - DUDU1*EZ)
       TZZDU2 = TD*(                      - DUDU2*EZ)
       TZZDU3 = TD*(           - DVDU3*EY           )
       TZZDU4 = TD*(2*DWDU4*EZ                      )

       VS(K,J,2,1) = TXXDU1*FX + TXYDU1*FY + TXZDU1*FZ
       VS(K,J,2,2) = TXXDU2*FX + TXYDU2*FY + TXZDU2*FZ
       VS(K,J,2,3) = TXXDU3*FX + TXYDU3*FY
```

```
      VS(K,J,2,4) = TXXDU4*FX             + TXZDU4*FZ

      VS(K,J,3,1) = TXYDU1*FX + TYYDU1*FY + TYZDU1*FZ
      VS(K,J,3,2) = TXYDU2*FX + TYYDU2*FY
      VS(K,J,3,3) = TXYDU3*FX + TYYDU3*FY + TYZDU3*FZ
      VS(K,J,3,4) =             TYYDU4*FY + TYZDU4*FZ

      VS(K,J,4,1) = TXZDU1*FX + TYZDU1*FY + TZZDU1*FZ
      VS(K,J,4,2) = TXZDU2*FX             + TZZDU2*FZ
      VS(K,J,4,3) =           + TYZDU3*FY + TZZDU3*FZ
      VS(K,J,4,4) = TXZDU4*FX + TYZDU4*FY + TZZDU4*FZ

        DQDU1 = DHDU1(K,J) - DUADU1*DU - DVADU1*DV - DWADU1*DW
     &                     - UAVE*DUDU1- VAVE*DVDU1- WAVE*DWDU1
        DQDU2 = DHDU2(K,J) - DUADU2*DU
     &                     - UAVE*DUDU2
        DQDU3 = DHDU3(K,J)             - DVADU3*DV
     &                                 - VAVE*DVDU3
        DQDU4 = DHDU4(K,J)                         - DWADU1*DW
     &                                             - WAVE*DWDU4
        DQDU5 = DHDU5(K,J)

        DQXDU1 = DQDU1*EX
        DQXDU2 = DQDU2*EX
        DQXDU3 = DQDU3*EX
        DQXDU4 = DQDU4*EX
        DQXDU5 = DQDU5*EX

        DQYDU1 = DQDU1*EY
        DQYDU2 = DQDU2*EY
        DQYDU3 = DQDU3*EY
        DQYDU4 = DQDU4*EY
        DQYDU5 = DQDU5*EY

        DQZDU1 = DQDU1*EZ
        DQZDU2 = DQDU2*EZ
        DQZDU3 = DQDU3*EZ
        DQZDU4 = DQDU4*EZ
        DQZDU5 = DQDU5*EZ

       VS(K,J,5,1) =
     &  - ( DUADU1*TXX(I,J,K) + DVADU1*TXY(I,J,K) + DWADU1*TXZ(I,J,K)
     &    + TXXDU1*UAVE + TXYDU1*VAVE + TXZDU1*WAVE
     &    - MOM*DQXDU1 )*FX
     &  - ( DUADU1*TXY(I,J,K) + DVADU1*TYY(I,J,K) + DWADU1*TYZ(I,J,K)
     &    + TXYDU1*UAVE + TYYDU1*VAVE + TYZDU1*WAVE
     &    - MOM*DQYDU1 )*FY
```

```
     &  - ( DUADU1*TXZ(I,J,K) + DVADU1*TYZ(I,J,K) + DWADU1*TZZ(I,J,K)
     &    + TXZDU1*UAVE + TYZDU1*VAVE + TZZDU1*WAVE
     &    - MOM*DQZDU1 )*FZ
      VS(K,J,5,2) =
     &  - ( DUADU2*TXX(I,J,K)
     &    + TXXDU2*UAVE + TXYDU2*VAVE + TXZDU2*WAVE
     &    - MOM*DQXDU2 )*FX
     &  - ( DUADU1*TXY(I,J,K)
     &    + TXYDU2*UAVE + TYYDU2*VAVE
     &    - MOM*DQYDU2 )*FY
     &  - ( DUADU2*TXZ(I,J,K)
     &    + TXZDU2*UAVE + TZZDU2*WAVE
     &    - MOM*DQZDU2 )*FZ
c      write(6,*) '** 72'
      VS(K,J,5,3) =
     &  - (                    + DVADU3*TXY(I,J,K)
     &    + TXXDU3*UAVE + TXYDU3*VAVE
     &    - MOM*DQXDU3 )*FX
     &  - (                    + DVADU3*TYY(I,J,K)
     &    + TXYDU3*UAVE + TYYDU3*VAVE + TYZDU3*WAVE
     &    - MOM*DQYDU3 )*FY
     &  - (                    + DVADU3*TYZ(I,J,K)
     &    + TYZDU3*VAVE + TZZDU3*WAVE
     &    - MOM*DQZDU3 )*FZ
      VS(K,J,5,4) =
     &   - (                                        + DWADU4*TXZ(I,J,K)
     &    + TXXDU4*UAVE + TXZDU4*WAVE
     &    - MOM*DQXDU4 )*FX
     &  - (                                       + DWADU4*TYZ(I,J,K)
     &    + TYYDU4*VAVE + TYZDU4*WAVE
     &    - MOM*DQYDU4 )*FY
     &  - (                                       + DWADU4*TZZ(I,J,K)
     &    + TXZDU4*UAVE + TYZDU4*VAVE + TZZDU4*WAVE
     &    - MOM*DQZDU4 )*FZ
      VS(K,J,5,5) = MOM*DQXDU5*FX + MOM*DQYDU5*FY + MOM*DQZDU5*FZ
C
 10   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE INVERT(I,J)
C
C    THIS SUBROUTINE INVERTS THE BLOCK TRIDIAGONAL
C
C
        INCLUDE 'euler.inc'
        INCLUDE 'invert.inc'
        REAL BE(KSIZ,5,5),BZ(5),OD
C
C
C----------------------------------------
C---- Invert block tridiagonal           _
C----------------------------------------
      JP1 = J + 1
      JM1 = J - 1
C
C---- Calculate [A(J,j=1)]
      IF (J .GT. 1) GOTO 15
      GOTO 350
 15   CONTINUE
C
C---- Calculate [A(J,j)] = AA(j) - B(j)*C(J-1)  for j > 1.
      DO 39 M=1,5
      DO 39 L=1,5
      DO 39 K=1,KM
 39      BE(K,L,M) = 0.
      DO 40 KK = 1,5
      DO 40 L = 1,5
        DO 40 M = 1,5
CDIR$ IVDEP
         DO 40 K=1,KM
 40        BE(K,KK,L) = BE(K,KK,L) + B(K,KK,M)*C(K,JM1,M,L)
      DO 50 M = 1,5
CDIR$ IVDEP
      DO 50 K=1,KM
         AA(K,M,1) = AA(K,M,1) - BE(K,M,1)
         AA(K,M,2) = AA(K,M,2) - BE(K,M,2)
         AA(K,M,3) = AA(K,M,3) - BE(K,M,3)
         AA(K,M,4) = AA(K,M,4) - BE(K,M,4)
 50      AA(K,M,5) = AA(K,M,5) - BE(K,M,5)
C
C---- Same for RHS
CDIR$ IVDEP
       DO 51 K=1,KM
       R(1,I,J,K) = R(1,I,J,K)
     &               - B(K,1,1)*R(1,I,JM1,K)- B(K,1,2)*R(2,I,JM1,K)
     &               - B(K,1,3)*R(3,I,JM1,K)- B(K,1,4)*R(4,I,JM1,K)
```

```
     &               - B(K,1,5)*R(5,I,JM1,K)
       R(2,I,J,K) = R(2,I,J,K)
     &               - B(K,2,1)*R(1,I,JM1,K)- B(K,2,2)*R(2,I,JM1,K)
     &               - B(K,2,3)*R(3,I,JM1,K)- B(K,2,4)*R(4,I,JM1,K)
     &               - B(K,2,5)*R(5,I,JM1,K)
       R(3,I,J,K) = R(3,I,J,K)
     &               - B(K,3,1)*R(1,I,JM1,K)- B(K,3,2)*R(2,I,JM1,K)
     &               - B(K,3,3)*R(3,I,JM1,K)- B(K,3,4)*R(4,I,JM1,K)
     &               - B(K,3,5)*R(5,I,JM1,K)
       R(4,I,J,K) = R(4,I,J,K)
     &               - B(K,4,1)*R(1,I,JM1,K)- B(K,4,2)*R(2,I,JM1,K)
     &               - B(K,4,3)*R(3,I,JM1,K)- B(K,4,4)*R(4,I,JM1,K)
     &               - B(K,4,5)*R(5,I,JM1,K)
 51    R(5,I,J,K) = R(5,I,J,K)
     &               - B(K,5,1)*R(1,I,JM1,K)- B(K,5,2)*R(2,I,JM1,K)
     &               - B(K,5,3)*R(3,I,JM1,K)- B(K,5,4)*R(4,I,JM1,K)
     &               - B(K,5,5)*R(5,I,JM1,K)
350      CONTINUE
C
C---- Use Gauss elimination to invert the systems:
C       [Dj][Ej] = [Cj]
C       [Dj] ZV = DW
C
C
C---- Calculate multipliers
      DO 121 KK=1,4
         DO 110 II = KK+1,5
C------- IN COLUMN KK
CDIR$ IVDEP
         DO 110 K=1,KM
  110       AA(K,II,KK) = -AA(K,II,KK)/AA(K,KK,KK)
C
C---- Subtract multiples of the pivot row from lower rows.
         DO 120 M = KK+1,5
         DO 120 N = KK+1,5
CDIR$ IVDEP
         DO 120 K=1,KM
  120       AA(K,M,N) = AA(K,M,N) + AA(K,KK,N)*AA(K,M,KK)
 121  CONTINUE
C
C---- CALCULATE THE NEW RHS
      DO 165 II = 2,5
        DO 165 M = 1,II-1
CDIR$ IVDEP
        DO 165 K=1,KM
          C(K,J,II,1) = C(K,J,II,1) + AA(K,II,M)*C(K,J,M,1)
          C(K,J,II,2) = C(K,J,II,2) + AA(K,II,M)*C(K,J,M,2)
```

```
          C(K,J,II,3) = C(K,J,II,3) + AA(K,II,M)*C(K,J,M,3)
          C(K,J,II,4) = C(K,J,II,4) + AA(K,II,M)*C(K,J,M,4)
 165      C(K,J,II,5) = C(K,J,II,5) + AA(K,II,M)*C(K,J,M,5)
CDIR$ IVDEP
      DO 166 K=1,KM
          R(2,I,J,K) = R(2,I,J,K) + AA(K,2,1)*R(1,I,J,K)
          R(3,I,J,K) = R(3,I,J,K) + AA(K,3,1)*R(1,I,J,K)
     &                            + AA(K,3,2)*R(2,I,J,K)
          R(4,I,J,K) = R(4,I,J,K) + AA(K,4,1)*R(1,I,J,K)
     &                            + AA(K,4,2)*R(2,I,J,K)
     &                            + AA(K,4,3)*R(3,I,J,K)
          R(5,I,J,K) = R(5,I,J,K) + AA(K,5,1)*R(1,I,J,K)
     &                            + AA(K,5,2)*R(2,I,J,K)
     &                            + AA(K,5,3)*R(3,I,J,K)
     &                            + AA(K,5,4)*R(4,I,J,K)
 166  CONTINUE
C
C---- Calculate solution vector and put in place of RHS
      DO 170 M = 5,1,-1
CDIR$ IVDEP
      DO 170 K=1,KM
        C(K,J,5,M) = C(K,J,5,M)/AA(K,5,5)

        C(K,J,4,M) = C(K,J,4,M) - C(K,J,5,M)*AA(K,4,5)
        C(K,J,4,M) = C(K,J,4,M)/AA(K,4,4)

        C(K,J,3,M) = C(K,J,3,M) - C(K,J,4,M)*AA(K,3,4)
     &                          - C(K,J,5,M)*AA(K,3,5)
        C(K,J,3,M) = C(K,J,3,M)/AA(K,3,3)

        C(K,J,2,M) = C(K,J,2,M) - C(K,J,3,M)*AA(K,2,3)
     &                          - C(K,J,4,M)*AA(K,2,4)
     &                          - C(K,J,5,M)*AA(K,2,5)
        C(K,J,2,M) = C(K,J,2,M)/AA(K,2,2)

        C(K,J,1,M) = C(K,J,1,M) - C(K,J,2,M)*AA(K,1,2)
     &                          - C(K,J,3,M)*AA(K,1,3)
     &                          - C(K,J,4,M)*AA(K,1,4)
     &                          - C(K,J,5,M)*AA(K,1,5)
 170    C(K,J,1,M) = C(K,J,1,M)/AA(K,1,1)
      DO 171 K=1,KM
        R(5,I,J,K) = R(5,I,J,K)/AA(K,5,5)

        R(4,I,J,K) = R(4,I,J,K) - R(5,I,J,K)*AA(K,4,5)
        R(4,I,J,K) = R(4,I,J,K)/AA(K,4,4)

        R(3,I,J,K) = R(3,I,J,K) - R(4,I,J,K)*AA(K,3,4)
```

```
     &                           - R(5,I,J,K)*AA(K,3,5)
        R(3,I,J,K) = R(3,I,J,K)/AA(K,3,3)

        R(2,I,J,K) = R(2,I,J,K) - R(3,I,J,K)*AA(K,2,3)
     &                           - R(4,I,J,K)*AA(K,2,4)
     &                           - R(5,I,J,K)*AA(K,2,5)
        R(2,I,J,K) = R(2,I,J,K)/AA(K,2,2)

        R(1,I,J,K) = R(1,I,J,K) - R(2,I,J,K)*AA(K,1,2)
     &                           - R(3,I,J,K)*AA(K,1,3)
     &                           - R(4,I,J,K)*AA(K,1,4)
     &                           - R(5,I,J,K)*AA(K,1,5)
 171    R(1,I,J,K) = R(1,I,J,K)/AA(K,1,1)
C
C---- When you've reached the top, solve.
      IF (J .LT. JM) GOTO 99
C
C---- Eliminate the (upper) off-diagonal of "E" blocks
      JMM = JM - 1
      DO 70 JJ = JMM,1,-1
      JJP1 = JJ + 1
CDIR$ IVDEP
      DO 70 K=1,KM
        R(1,I,JJ,K) = R(1,I,JJ,K) - C(K,JJ,1,1)*R(1,I,JJP1,K)
     &                            - C(K,JJ,1,2)*R(2,I,JJP1,K)
     &                            - C(K,JJ,1,3)*R(3,I,JJP1,K)
     &                            - C(K,JJ,1,4)*R(4,I,JJP1,K)
     &                            - C(K,JJ,1,5)*R(5,I,JJP1,K)
        R(2,I,JJ,K) = R(2,I,JJ,K) - C(K,JJ,2,1)*R(1,I,JJP1,K)
     &                            - C(K,JJ,2,2)*R(2,I,JJP1,K)
     &                            - C(K,JJ,2,3)*R(3,I,JJP1,K)
     &                            - C(K,JJ,2,4)*R(4,I,JJP1,K)
     &                            - C(K,JJ,2,5)*R(5,I,JJP1,K)
        R(3,I,JJ,K) = R(3,I,JJ,K) - C(K,JJ,3,1)*R(1,I,JJP1,K)
     &                            - C(K,JJ,3,2)*R(2,I,JJP1,K)
     &                            - C(K,JJ,3,3)*R(3,I,JJP1,K)
     &                            - C(K,JJ,3,4)*R(4,I,JJP1,K)
     &                            - C(K,JJ,3,5)*R(5,I,JJP1,K)
        R(4,I,JJ,K) = R(4,I,JJ,K) - C(K,JJ,4,1)*R(1,I,JJP1,K)
     &                            - C(K,JJ,4,2)*R(2,I,JJP1,K)
     &                            - C(K,JJ,4,3)*R(3,I,JJP1,K)
     &                            - C(K,JJ,4,4)*R(4,I,JJP1,K)
     &                            - C(K,JJ,4,5)*R(5,I,JJP1,K)
        R(5,I,JJ,K) = R(5,I,JJ,K) - C(K,JJ,5,1)*R(1,I,JJP1,K)
     &                            - C(K,JJ,5,2)*R(2,I,JJP1,K)
     &                            - C(K,JJ,5,3)*R(3,I,JJP1,K)
     &                            - C(K,JJ,5,4)*R(4,I,JJP1,K)
```

```
     &                              - C(K,JJ,5,5)*R(5,I,JJP1,K)
 70   CONTINUE
c      IF (I .NE. 5 .or. iter .ne. 8) GOTO 99
c      k=5
c      do 71 j=1,jm
c      write(6,*) 'invert: residuals  j=',j
c 71   write(6,*)r(1,i,j,k),r(2,i,j,k),r(3,i,j,k),r(4,i,j,k),r(5,i,j,k)
C
 99   RETURN
      END
C
```

```
      SUBROUTINE SUMUL
C
C     THIS SUBROUTINE CALCULATES LAMINAR VISCOSITIES
C
C*********************************************************************
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
C
C*********************************************************************
C
      REAL HOH,HOH3S,M2,SH1,SHO,MUO
C
      SH1 = SHCONST
      SHO = HIN -.5*(UIN*UIN + WIN*WIN)
      MUO = 1/REYNUM
C
      DO 10 K = 1,KM
      DO 10 J = 1,JM
      DO 10 I = 1,IM
         SH = (U(5,I,J,K) + P(I,J,K))/U(1,I,J,K) -
     &   .5*(U(2,I,J,K)*U(2,I,J,K) + U(3,I,J,K)*U(3,I,J,K) +
     &       U(4,I,J,K)*U(4,I,J,K))/(U(1,I,J,K)*U(1,I,J,K))

         HOH = SH/SHO
         HOH3S = SQRT(ABS(HOH*HOH*HOH))
         M2 = (SHO + SH1)/(SH + SH1)
C
         MUL(I,J,K)  = MUO*HOH3S*M2
C
 10   CONTINUE
C
C---- Do boundaries
      DO 20 K = 1,KM
      DO 20 I = 1,IM
         MUL(I,0,K) = MUL(I,1,K)
         MUL(I,JM+1,K) = MUL(I,JM,K)
 20   CONTINUE
      RETURN
      END   !SUMUL
c
       SUBROUTINE BALO
      RETURN
      END
```

```
      SUBROUTINE BLVISC
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
C
      REAL QX(ISIZ,0:JSIZ,KSIZ),QY(ISIZ,0:JSIZ,KSIZ),
     &     QZ(ISIZ,0:JSIZ,KSIZ)
      REAL MULT,MULTPR,RHO,RHOP,UAVE,VAVE,WAVE,DU,DV,DW,DH
C
C---- Molecular viscosity
      CALL SUMUL
C
C---- Turbulent ''viscosity''
c      CALL BALO
C
C---- Define appropriate boundary values for the state vector on airfoil
C---- NOTE:  In Roberts/Goodsell code values at U(i,j=0,k) are not
C----        used for flux calculations and are defined appropriately
C----        just before dissipation calculation.  Thus the definitions
C----        here can be done with impunity.  For other host codes may have
C----        to change values back to original.
      DO 10 K = 1,KM
      DO 10 I = ILE,ITE-1
         U(1,I,0,K) = U(1,I,1,K)
         U(2,I,0,K) =-U(2,I,1,K)
         U(3,I,0,K) =-U(3,I,1,K)
         U(4,I,0,K) =-U(4,I,1,K)
C
C------- Adiabatic conditions (note: this isn't overspecified)
         U(5,I,0,K) = U(5,I,1,K)
           P(I,0,K) =   P(I,1,K)
 10   CONTINUE
C
C---- If not adiabatic, correct the density (This then gives correct enthalpy)
      IF(HWALL .NE. 0.) THEN
      DO 11 K = 1,KM
      DO 11 I = ILE,ITE-1
         U1WALL = (U(5,I,0,K)+P(I,0,K))/HWALL
 11      U(1,I,0,K) = 2*U1WALL-U(1,I,1,K)
      ENDIF
C
C---- Calculate viscous terms
      DO 250 K = 1,KM
      DO 250 J = 0,JM
      DO 250 I = 1,IM
C
C---- Define some convenient quantities
```

```
      RHOP = U(1,I,J+1,K)
      RHO  = U(1,I,J,K)
      UAVE = .5*(U(2,I,J+1,K)/RHOP + U(2,I,J,K)/RHO)
      VAVE = .5*(U(3,I,J+1,K)/RHOP + U(3,I,J,K)/RHO)
      WAVE = .5*(U(4,I,J+1,K)/RHOP + U(4,I,J,K)/RHO)
      DU = U(2,I,J+1,K)/RHOP - U(2,I,J,K)/RHO
      DV = U(3,I,J+1,K)/RHOP - U(3,I,J,K)/RHO
      DW = U(4,I,J+1,K)/RHOP - U(4,I,J,K)/RHO
      DH = (U(5,I,J+1,K)+P(I,J+1,K))/RHOP -
     &     (U(5,I,J  ,K)+P(I,J  ,K))/RHO
C
C---- TXY etc.
      TXY(I,J,K) = DU*ETY(I,J,K) + DV*ETX(I,J,K)
      TXZ(I,J,K) = DU*ETZ(I,J,K) + DW*ETX(I,J,K)
      TYZ(I,J,K) = DV*ETZ(I,J,K) + DW*ETY(I,J,K)
C---- Tyy
      TYY(I,J,K) =   1.3333333*DV*ETY(I,J,K) -
     &              0.6666667*(DU*ETX(I,J,K) + DW*ETZ(I,J,K))
C
      TXX(I,J,K) =   1.3333333*DU*ETX(I,J,K) -
     &              0.6666667*(DV*ETY(I,J,K) + DW*ETZ(I,J,K))
C
      TZZ(I,J,K) =   1.3333333*DW*ETZ(I,J,K) -
     &              0.6666667*(DV*ETY(I,J,K) + DU*ETX(I,J,K))

C---- QX, QY, QZ
      QQ = DH - UAVE*DU - VAVE*DV - WAVE*DW
      QX(I,J,K) = QQ*ETX(I,J,K)
      QY(I,J,K) = QQ*ETY(I,J,K)
      QZ(I,J,K) = QQ*ETZ(I,J,K)

c       if (i.eq.1.and.j.eq.0)write(6,251)
c     & DU,DV,DW,DH,K,QQ
c       if (i.eq.1.and.j.eq.0)write(6,252)
c     & U(2,i,j,k),u(3,i,j,k),u(4,i,j,k),u(5,i,j,k),Kk,u(1,i,j,k)
 250  CONTINUE
 251  FORMAT('DU,V,W,H,K,QQ=',4F8.4,I3,F8.4)
 252  FORMAT(' U,V,W,E,K,rh=',4F8.4,I3,F8.4)
C
C---- Viscous fluxes.  Note:  stresses T(j) defined at j+1/2 face
CPUT THESE VISCOSITIES IN STRESS DEFINITIONS
      DO 260 K = 1,KM
      DO 260 J = 0,JM
      DO 260 I = 1,IM
         MULT   = .5*(MUL(I,J  ,K)    + MUT(I,J  ,K) +
     &                MUL(I,J+1,K)    + MUT(I,J+1,K))
         MULTPR = .5*(MUL(I,J  ,K)/PR + MUT(I,J  ,K)/PRT +
```

```
     &                MUL(I,J+1,K)/PR + MUT(I,J+1,K)/PRT)
      UAVE = .5*(U(2,I,J+1,K)/RHOP + U(2,I,J,K)/RHO)
      VAVE = .5*(U(3,I,J+1,K)/RHOP + U(3,I,J,K)/RHO)
      WAVE = .5*(U(4,I,J+1,K)/RHOP + U(4,I,J,K)/RHO)

         DR2 = -( TXX(I,J,K)*FACEX(2,I,J,K) +
     &            TXY(I,J,K)*FACEY(2,I,J,K) +
     &            TXZ(I,J,K)*FACEZ(2,I,J,K) )*MULT
C
         DR3 = -( TXY(I,J,K)*FACEX(2,I,J,K) +
     &            TYY(I,J,K)*FACEY(2,I,J,K) +
     &            TYZ(I,J,K)*FACEZ(2,I,J,K) )*MULT
C
         DR4 = -( TXZ(I,J,K)*FACEX(2,I,J,K) +
     &            TYZ(I,J,K)*FACEY(2,I,J,K) +
     &            TZZ(I,J,K)*FACEZ(2,I,J,K) )*MULT
C
         DR5 =
     &  - (MULT*(UAVE*TXX(I,J,K) + VAVE*TXY(I,J,K) + WAVE*TXZ(I,J,K))
     &   + MULTPR*QX(I,J,K) )*FACEX(2,I,J,K)
C
     &  - (MULT*(UAVE*TXY(I,J,K) + VAVE*TYY(I,J,K) + WAVE*TYZ(I,J,K))
     &   + MULTPR*QY(I,J,K) )*FACEY(2,I,J,K)
C
     &  - (MULT*(UAVE*TXZ(I,J,K) + VAVE*TYZ(I,J,K) + WAVE*TZZ(I,J,K))
     &   + MULTPR*QZ(I,J,K) )*FACEZ(2,I,J,K)

C
C---- Subtract outgoing viscous flux from cell (i,j,k) and add it to (i,j+1,k)
      R(2,I,J,K) = R(2,I,J,K) - DR2
      R(3,I,J,K) = R(3,I,J,K) - DR3
      R(4,I,J,K) = R(4,I,J,K) - DR4
      R(5,I,J,K) = R(5,I,J,K) - DR5
C
      R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
      R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
      R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
      R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
c      if (i.eq.1.and.j.eq.0)write(6,261)dr2,dr3,dr4,dr5,k
 260  CONTINUE
 261  format('drs='4F12.9,i3)
 262  format('txy,xx,xz=',6F11.3)
 263  format('2,3,4,5,p=',5F11.6)
 264  format(' qx,qy,qz=',3F11.2)
 266  format('facexyz,z=',4F11.6)
c      I=10
c      J=0
```

```
c       DO 265 K=1,KM
c       write(6,*) ' k=',k
c       write(6,262)
c      & txy(i,j,k),txz(i,j,k),tyz(i,j,k),txx(i,j,k),tyy(i,j,k),
c      & tzz(i,j,k)
c       write(6,266)
c      & facex(2,i,j,k),facey(2,i,j,k),facez(2,i,j,k),z(i,2,k)
c         write(6,264)
c      & Qx(i,j,k),qy(i,j,k),qz(i,j,k)
c265    write(6,263)
c      & R(2,i,1,k),R(3,i,1,k),R(4,i,1,k),r(5,i,1,k),p(i,1,k)
C

       RETURN
       END
```

```
       SUBROUTINE BLETA
C
C---- THIS SUBROUTINE CALCULATE THE METRICS FOR THE GENERALIZED
C---- TRANSFORMATION
C
      INCLUDE 'euler.inc'
      INCLUDE 'blvisc.inc'
      COMMON/DEBUGCOM/ ZEX(ISIZ,JSIZ,KSIZ),ZEY(ISIZ,JSIZ,KSIZ),
     &                 ZEZ(ISIZ,JSIZ,KSIZ)
      REAL   XXI(ISIZ,JSIZ,KSIZ),XET(ISIZ,JSIZ,KSIZ),
     &       XZE(ISIZ,JSIZ,KSIZ),
     &       YXI(ISIZ,JSIZ,KSIZ),YET(ISIZ,JSIZ,KSIZ),
     &       YZE(ISIZ,JSIZ,KSIZ),ZXI(ISIZ,JSIZ,KSIZ),
     &       ZET(ISIZ,JSIZ,KSIZ),ZZE(ISIZ,JSIZ,KSIZ),JAC
      REAL UU1,UU2,UU3,VV1,VV2,VV3
C
C---- Calculate metrics:  (DET = 1, DXI  = 1, DZE = 1 BY ASSUMPTION)
C     Note: These are defined along cell "edges".  Thus, no information
C           from outside the grid boundaries is required.
      DO 15 I = 1,IM
      DO 15 J = 1,JMAX
      DO 15 K = 1,KMAX
         XXI(I,J,K) = X(I+1,J,K)-X(I,J,K)
         YXI(I,J,K) = Y(I+1,J,K)-Y(I,J,K)
 15      ZXI(I,J,K) = Z(I+1,J,K)-Z(I,J,K)
      DO 16 I = 1,IMAX
      DO 16 J = 1,JM
      DO 16 K = 1,KMAX
         XET(I,J,K) = X(I,J+1,K)-X(I,J,K)
         YET(I,J,K) = Y(I,J+1,K)-Y(I,J,K)
 16      ZET(I,J,K) = Z(I,J+1,K)-Z(I,J,K)
      DO 17 I = 1,IMAX
      DO 17 J = 1,JMAX
      DO 17 K = 1,KM
         XZE(I,J,K) = X(I,J,K+1)-X(I,J,K)
         YZE(I,J,K) = Y(I,J,K+1)-Y(I,J,K)
 17      ZZE(I,J,K) = Z(I,J,K+1)-Z(I,J,K)

 14   format('yxia,et,ze=',1X,3f8.5)
 141   format('zxia,et,ze=',1X,3f8.5)
 142   format('j,k,jac',1X,2I2,f15.6)
C
C---- calculate volume centered average metrics
      DO 20 I = 1,IM
      DO 20 J = 1,JM
      DO 20 K = 1,KM
      XXIA=.25*(XXI(I,J,K)+XXI(I,J+1,K)+XXI(I,J,K+1)+XXI(I,J+1,K+1))
```

```
      YXIA=.25*(YXI(I,J,K)+YXI(I,J+1,K)+YXI(I,J,K+1)+YXI(I,J+1,K+1))
      ZXIA=.25*(ZXI(I,J,K)+ZXI(I,J+1,K)+ZXI(I,J,K+1)+ZXI(I,J+1,K+1))

      XETA=.25*(XET(I,J,K)+XET(I,J,K+1)+XET(I+1,J,K)+XET(I+1,J,K+1))
      YETA=.25*(YET(I,J,K)+YET(I,J,K+1)+YET(I+1,J,K)+YET(I+1,J,K+1))
      ZETA=.25*(ZET(I,J,K)+ZET(I,J,K+1)+ZET(I+1,J,K)+ZET(I+1,J,K+1))

      XZEA=.25*(XZE(I,J,K)+XZE(I,J+1,K)+XZE(I+1,J,K)+XZE(I+1,J+1,K))
      YZEA=.25*(YZE(I,J,K)+YZE(I,J+1,K)+YZE(I+1,J,K)+YZE(I+1,J+1,K))
      ZZEA=.25*(ZZE(I,J,K)+ZZE(I,J+1,K)+ZZE(I+1,J,K)+ZZE(I+1,J+1,K))
C
C---- Calculate Jacobians and inverse metrics (store volume)
      JAC = 1/(
     &   XXIA*(YETA*ZZEA-YZEA*ZETA)
     & - XETA*(YXIA*ZZEA-YZEA*ZXIA)
     & + XZEA*(YXIA*ZETA-YETA*ZXIA) )
      VOL(I,J,K) = 1/JAC

      ETX(I,J,K)=-(YXIA*ZZEA-YZEA*ZXIA)*JAC
      ETY(I,J,K)= (XXIA*ZZEA-XZEA*ZXIA)*JAC
      ETZ(I,J,K)=-(XXIA*YZEA-XZEA*YXIA)*JAC
C
C---- For debugging purposes only
      ZEX(I,J,K)= (YXIA*ZETA-YETA*ZXIA)*JAC
      ZEY(I,J,K)=-(XXIA*ZETA-XETA*ZXIA)*JAC
      ZEZ(I,J,K)= (XXIA*YETA-XETA*YXIA)*JAC
      kk = k
      ii = i
c      vol=ABS((Y(I,J+1,K)-Y(I,J,K))*(X(I+1,J,K)-X(I,J,K))*
c     & (Z(I,J,K+1)-Z(I,J,K)))
c      volu = facex(1,i,j,k)*(x(i+1,j,k)-x(i,j,k))
c      if(i.eq.1 .and.j.eq.1)write(6,*)'1/JAC,VOL=',j,k,1/jac,VOLu
c      if(i.eq.10.and.j.eq.1)write(6,*)'1/JAC,VOL=',j,k,1/jac,VOLu
c      IF(I.EQ.10.and.j.eq.2)WRITE(6,*) 'K=',KK
c      IF(I.EQ.10.and.j.eq.2)WRITE(6,14)
c     & yxia,yeta,yzea
c      IF(I.EQ.10.and.j.eq.2)WRITE(6,*) 'i,K=',ii,KK
c      IF(I.EQ.10.and.j.eq.2)WRITE(6,141)
c     & zxia,zeta,zzea

 20   CONTINUE
C
C---- Extrapolate at upper and lower boundaries:
      DO 21 I = 1,IM
      DO 21 K = 1,KM
        ETX(I,JMAX,K) = ETX(I,JM,K)
        ETY(I,JMAX,K) = ETY(I,JM,K)
```

```
        ETZ(I,JMAX,K) = ETZ(I,JM,K)
        ETX(I,0,K) = ETX(I,1,K)
        ETY(I,0,K) = ETY(I,1,K)
        ETZ(I,0,K) = ETZ(I,1,K)
 21   CONTINUE

C
C---- Calculate face centered averages at each J surface (FACE 1 & 3 needed)
C     as average of the two adjacent volumes.   Planes (1->IM)*(1->KM)
      DO 41 I = 1,IM
      DO 41 J = 0,JM
      DO 41 K = 1,KM
        ETX(I,J,K) = .5*(ETX(I,J+1,K) + ETX(I,J,K))
        ETY(I,J,K) = .5*(ETY(I,J+1,K) + ETY(I,J,K))
        ETZ(I,J,K) = .5*(ETZ(I,J+1,K) + ETZ(I,J,K))
        ii =i
        KK =K
c      IF(I.EQ.10.and.j.eq.0)WRITE(6,*) 'K=',KK
c      IF(I.EQ.10.and.j.eq.0)WRITE(6,*)
c     & 'EX,Y,Z=',ETX(I,J,K),ETY(I,J,K),ETZ(I,J,K)
c      IF(I.EQ.24.and.j.eq.0)WRITE(6,*) 'i,K=',ii,KK
c      IF(I.EQ.24.and.j.eq.0)WRITE(6,*)
c     & 'EX,Y,Z=',ETX(I,J,K),ETY(I,J,K),ETZ(I,J,K)
 41   CONTINUE
C
C---- Quantities needed for time step calculation:
C
C---- Estimate normal distance as ave of normal distance of corner normal lines
      DO 50 I=1,IM
      DO 50 J=1,JM
      DO 50 K=1,KM
      DNORM(I,J,K) = .25*(SQRT( (Y(I,J+1,K)-Y(I,J,K))**2 +
     &                          (X(I,J+1,K)-X(I,J,K))**2 +
     &                          (Z(I,J+1,K)-Z(I,J,K))**2)   +
     &                   SQRT( (X(I+1,J+1,K+1)-X(I+1,J,K+1))**2 +
     &                         (Y(I+1,J+1,K+1)-Y(I+1,J,K+1))**2 +
     &                         (Z(I+1,J+1,K+1)-Z(I+1,J,K+1))**2)  +
     &                   SQRT( (X(I+1,J+1,K)-X(I+1,J,K))**2 +
     &                         (Y(I+1,J+1,K)-Y(I+1,J,K))**2 +
     &                         (Z(I+1,J+1,K)-Z(I+1,J,K))**2)   +
     &                   SQRT( (X(I,J+1,K+1)-X(I,J,K+1))**2 +
     &                         (Y(I,J+1,K+1)-Y(I,J,K+1))**2 +
     &                         (Z(I,J+1,K+1)-Z(I,J,K+1))**2) )
C
C---- Calculate area of cell face #5 (parallel to body)
      UU1 = X(I+1,J+1,K+1)-X(I,J+1,K)
      UU2 = Y(I+1,J+1,K+1)-Y(I,J+1,K)
```

```
      UU3 = Z(I+1,J+1,K+1)-Z(I,J+1,K)
      VV1 = X(I,J+1,K+1)-X(I+1,J+1,K)
      VV2 = Y(I,J+1,K+1)-Y(I+1,J+1,K)
      VV3 = Z(I,J+1,K+1)-Z(I+1,J+1,K)
C
      NAREA(I,J,K) = .5*( SQRT((UU2*VV3-UU3*VV2)**2 +
     &                         (UU1*VV3-UU3*VV1)**2 +
     &                         (UU1*VV2-UU2*VV1)**2 ))
 50   CONTINUE
      RETURN
      END
```

Common blocks:

```
c      PARAMETER (ISIZ = 27, JSIZ = 37, KSIZ = 35,NNRES = 5000)
c      PARAMETER (ISIZ = 39, JSIZ = 49, KSIZ = 67,NNRES = 15000)
      PARAMETER (ISIZ = 39, JSIZ = 65, KSIZ = 103,NNRES = 5000)
      IMPLICIT REAL (A-H,O-Z)
      REAL KAP2,KAP4,MACH
      CHARACTER*20 DATE
      CHARACTER*30 GRNAME,RSNAME,STNODE,SVNAME,CFNAME,RESTRT,INPDAT,
     &             LOWANG,OLDRES,CONSOLN
      CHARACTER*50 GRIDNAME
      LOGICAL SYM,BIN,CFBIN
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+2,JSIZ+2,KSIZ+2),NCELLS,
     &             UO(5,ISIZ+1,JSIZ+1,KSIZ+1),DELNORM(5)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM,
     &             ILE,ITE,SWEEP
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYMB(ISIZ,JSIZ),PRESYMT(ISIZ,JSIZ),
     &             FRONT(JSIZ,KSIZ),WALLB(ISIZ,JSIZ),WALLT(ISIZ,JSIZ),
     &             PRESWG(ISIZ,KSIZ)
      COMMON/DISP/ D(5,ISIZ,JSIZ,KSIZ),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN,PIN,EIN,AOA,ALPHA
      COMMON/TIME/ DT(0:ISIZ+1,0:JSIZ+1,0:KSIZ+1),ALPHA1,ALPHA2,ALPHA3,
     &             ITMAX
      COMMON/AREA/ AXM(ISIZ,JSIZ,KSIZ),AYM(ISIZ,JSIZ,KSIZ),
     &             AZM(ISIZ,JSIZ,KSIZ)
      COMMON/BOU2/ WUN(3,0:ISIZ+1,0:KSIZ+1)
      COMMON/WORDLOG/ SYM,BIN
      COMMON/WORDDAT/ DATE,GRIDNAME
      COMMON/HIST/ ITER,ITCOEF,ITPRIN
      COMMON/RESDATA/ RMS(NNRES),RMS2(NNRES),RMS3(NNRES),RMS4(NNRES),
     &                RMS5(NNRES),RESMAX(NNRES),RE
      COMMON/NAMES/ GRNAME,RSNAME,STNODE,SVNAME,CFNAME,RESTRT,INPDAT,
     &              LOWANG,OLDRES,CONSOLN
      COMMON/COEF1/CFBIN
      COMMON/INITCON/ICON
      COMMON/RSMOOTH/EPSR
```

```
      REAL MUL,MUT
      REAL ETX,ETY,ETZ,NAREA,DNORM,MUSI
      LOGICAL SEMIIMP
      COMMON/VSTUFF/ MUL(ISIZ,0:JSIZ,KSIZ),MUT(ISIZ,0:JSIZ,KSIZ),
     &       ETX(ISIZ,0:JSIZ,KSIZ),ETY(ISIZ,0:JSIZ,KSIZ),
     &       ETZ(ISIZ,0:JSIZ,KSIZ),VOL(ISIZ,JSIZ,KSIZ),
     &       NAREA(ISIZ,JSIZ,KSIZ),DNORM(ISIZ,JSIZ,KSIZ),
     &       SHCONST,ACONST,HWALL,REYNUM,PR,PRT
      COMMON/TAU/ TXX(ISIZ,0:JSIZ,KSIZ),TYY(ISIZ,0:JSIZ,KSIZ),
     &            TZZ(ISIZ,0:JSIZ,KSIZ),TXY(ISIZ,0:JSIZ,KSIZ),
     &            TXZ(ISIZ,0:JSIZ,KSIZ),TYZ(ISIZ,0:JSIZ,KSIZ)
      COMMON/SISTUFF/ SEMIIMP,MUSI
```

```
 COMMON/DIFFER/
&     DHDU1(KSIZ,JSIZ),DHDU2(KSIZ,JSIZ),DHDU3(KSIZ,JSIZ),
&     DHDU4(KSIZ,JSIZ),DHDU5(KSIZ,JSIZ),
&     DPDU1(KSIZ,JSIZ),DPDU2(KSIZ,JSIZ),DPDU3(KSIZ,JSIZ),
&     DPDU4(KSIZ,JSIZ),DPDU5
 COMMON/DFGW/ GW(KSIZ,JSIZ,2:5,5),FW(KSIZ,JSIZ,2:5,5),
&             HW(KSIZ,JSIZ,2:5,5)
```

```
REAL B(KSIZ,5,5),C(KSIZ,JSIZ,5,5),AA(KSIZ,5,5)
COMMON/LHSMAT/ B,C,AA
```

```
COMMON/NSCON/ VS(KSIZ,JSIZ,5,5)
```

Typical input file:

```
'11 June 88'            DATE
1.60                   MACH
0.                      AOA
0.                      YAW
1.4                     GAM
0.02                    KAP2
0.005                    KAP4
0.  0.                 Add'l smoothing at bow shock (set to 0)
1.5                    CFL (5.0)  (.1)
0.0                    AENTH (0.025)
50                     ITMAX
50                       ITCOEF
50                      ITPRIN
.TRUE.                  BIN
.TRUE.                 CFBIN
01                       ITER
0                       ICON
1                       ILE
25                     ITE
0.                      RE
.0                     EPSR (1.0)
'test.gri'              GRNAME - Grid file
'sires.dat'             RSNAME - Residuals
'siiris.dat'            STNODE - Output for Iris
'sisv.dat'              SVNAME - State vectors saved for restart
'sicf.dat'              CFNAME - Force Coeficients
'sisv.dat'              RESTRT - Restart file: old state vectors
'siinp.dat'             INPDAT - Copy of input dat
'junk'                  LOWANG - Lower angle solution -- NOT IMPLEMENTED
'sires.dat'                      OLDRES - Old residuals
'freestream' cones.sol'             CONSOLN- Conical starting solution
.false.                  SEMIIMP- Semi-implicit = true, explicit = false
0.00                   MUSI-Implicit smoothing coefficient ~.001
.625E5         REYNUM - Reynolds number = physical/mach number
298.,  0.              Tinf,Twall - Temp at inf , wall (if 0 -- adiabatic)
.72, .9                 PR,PRT - Laminar, turbulent Reynolds number
```