Summer 1992

Runge-Kutta Upwind Multigrid Multi-Block Three-Dimensional Thin Layer Navier-Stokes Solver

Frank E. Cannizzaro
Old Dominion University

RUNGE-KUTTA UPWIND MULTIGRID MULTI-BLOCK THREE-DIMENSIONAL THIN LAYER NAVIER-STOKES SOLVER

Frank E. Cannizzaro

B.S. May 1986, Old Dominion University

M.S. August 1988, Old Dominion University

A Dissertation submitted to the Faculty of Old Dominion University

in Partial Fulfillment of the Requirement for the Degree of

DOCTOR OF PHILOSOPHY

MECHANICAL ENGINEERING

OLD DOMINION UNIVERSITY

AUGUST 1992

Approved by:

Robert L. Ash (Director)

Osama A. Kandil

Oktay Baysal

Arthur C. Taylor

Manuel D. Salas

ABSTRACT

A state-of-the-art computer code has been developed that incorporates a modified Runge-Kutta time integration scheme, Upwind numerical techniques, Multigrid acceleration, and Multi-block capabilities (RUMM). A three-dimensional thin-layer formulation of the Navier-Stokes equations is employed. For turbulent flow cases, the Baldwin-Lomax algebraic turbulence model is used. Two different upwind techniques are available, van Leer's flux-vector splitting and Roe's flux-difference splitting. Full approximation multigrid plus implicit residual and corrector smoothing were implemented to enhance the rate of convergence. Multi-block capabilities were developed to provide geometric flexibility. This feature allows the developed computer code to accommodate any grid topology or grid configuration with multiple topologies. The results shown in this dissertation were chosen to validate the computer code and display is geometric flexibility, which is provided by the multi-block structure.

DEDICATION

To my family, who has always supported me and cheered me on through all of my endeavors. I dedicate this work to my parents, who gave me the best start in life anyone could ask for. They have provided unyielding support and encouragement.

ACKNOWLEDGMENT

I would like to thank Dr. Ash for taking over as director of my graduate committee, after my original advisor left. Dr. Ash accepted the position even though it was a political nightmare. I would also like to thank my other faculty advisors, Dr. Kandil, Dr. Baysal, and Dr. Taylor for being on my graduate committee.

My thanks to Manuel Salas, my first branch head at NASA Langley Research Center, and also one of my graduate committee advisors. I really enjoyed being a part of his branch and having him as my branch head. Many people have helped me along the way. One has been my assistant branch head, James Keller. He always looked out for me and did what he could to help me. From the beginning Duane Melson, of NASA, has been a great support and friend and got me started in the correct direction in CFD research. Also, Joseph Morrison, of Analytical Services and Materials, Inc., has been a good friend and provided good advice.

A special thanks to Alaa Elmiligui, who has been a Ph.D. student with me, in the exact same situation, and my officemate. I have the deepest respect and appreciation for Alaa. He gave me a lot of support and has been a great friend.

This work was supported by NASA Langley Research Center Grant No. NAG1–633.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	flux Jacobian $\frac{\partial F}{\partial Q}$
a	local speed of sound
A^+	turbulence constant, 26.
c	chord or body length
C_{KLEB}	Klebanoff constant, 0.3
C_{cp}	turbulence constant, 1.6
C_p	coefficient of pressure
$\check{c}_p$	dimensional specific heat at constant pressure
C_{wk}	turbulence constant, 1.0 {value used for transonic flow}
$\check{c}_v$	dimensional specific heat at constant volume
CFL	Courant-Friedrichs-Lewy stability limit
CFL^*	standard CFL with no residual smoothing
$\hat{e}_i$	dimensional internal energy
E	total internal energy per unit volume
f	forcing function in finite-difference problem used in multigrid process
$f,\ f_v$	Eulerian and viscous Reynolds Favré density averaged dimensional flux vectors in the x-direction
FMG	full multigrid cylces
F, F_v	nondimensional Eulerian and viscous flux vectors in the ξ-direction
$\check{F},\ \tilde{F}$	dimensional and nondimensional Eulerian flux vectors in the x-direction
$\check{F}_v,\ \tilde{F}_v$	dimensional and nondimensional viscous flux vectors in the x-direction
FS	indicates Fuselage Station location on the afterbody
g	grid
$g,\ g_v$	Eulerian and viscous Reynolds Favre density averaged dimensional flux vectors in the y-coordinate direction
G, G_v	nondimensional Eulerian and viscous flux vectors in the η-direction
$\check{G},\ \tilde{G}$	dimensional and nondimensional Eulerian flux vector in the y-direction

$\check{G}_v$, $\tilde{G}_v$	dimensional and nondimdnsional viscous flux vectors in the y-direction
H	total enthalpy per unit volume
h	specific enthalpy per unit volume
h, h_v	Eulerian and viscous Reynolds Favre density averaged dimensional flux vectors in the z-direction
H, H_v	nondimensional Eulerian and viscous flux vectors in the ζ-direction
$\check{H}$, $\tilde{H}$	dimensional and nondimensional Eulerian flux vectors in the z-direction
$\check{H}_v$, $\tilde{H}_v$	dimensional and nondimensional viscous flux vectors in the z-direction
I_h^{2h}	restriction operator from h-spacing to 2h-spacing
I_{2h}^h	prolongation operator from 2h-spacing to h-spacing
J	grid cell Jacobian of transformation
K	kinetic energy
k	coefficient of thermal conductivity
k_1	von Karman's constant 0.4
K_2	Clauser constant, 0.0168
k_{T}	coefficient of turbulent thermal conductivity
L	finite-difference operator
M	Mach number
N	n-stage modified Runge-Kutta time integration
$\check{p}$, p	dimensional and nondimentional static pressure
P_r	Prandtl number
$P_{r\mathrm{T}}$	turbulent Prandtl number
PE	potential energy
q	normalize contravariant velocity
q	Reynolds Favré density averaged dimensional vector of conservative variables
Q	nondimensional vector of conservative variables in body-fitted coordinates
$\check{Q}$, $\tilde{Q}$	dimensional and nondimensional vectors of conservative variables in Cartesian coordinates
R	Reynolds number
R^+, R^-	non-negative and non-positive Riemann invariants
S^*	isentropically derived entropy value
S_ℓ	left eigenvector matrix relative to the ℓ-direction

S_ℓ^{-1}	right eigenvector matrix relative to the ℓ-direction
t	nondimensional time
$\breve{t}$	dimensional time
$\breve{T}$, T	dimensional and nondimensional static temperature
U	solution to finite-difference problem used in multigrid process
u	approximate solution to finite-difference problem used in multigrid process
U, V, W	contravariant velocities in the ξ-, η-, and ζ-coordinate directions
$\overline{u}_\ell$	local scaled contravariant velocity in the ℓ-direction
$\breve{u}$, u	dimensional and nondimensional Cartesian velocity in the x-Cartesian coordinate direction
V	error in approximate solution to finite-diffference problem used in multigrid process
$\breve{v}$, v	dimensional and nondimensional Cartesian velocity in the y-Cartesian coordinate direction
Vol	volume of cell
$\breve{w}$, w	dimensional and nondimensional Cartesian velocity in the z-Cartesian coordinate direction
x, y, z	nondimensional Cartesian coordinate directions
$\breve{x}$, $\breve{y}$, $\breve{z}$	dimensional Cartesian coordinate directions
y^+	length scale for law of the wall
α	angle of attack
β	γ-1
β_ℓ	variable coefficient for residual smoothing in the ℓ-direction
γ	ratio of specific heats c_p/c_v
λ	indicates eigenvalue
Λ	diagonal eigenvalue matrix
μ_T	turbulent eddy viscosity
$\breve{\mu}$, μ	dimensional and nondimensional molecular viscosity
ω	vorticity
$\breve{\rho}$, ρ	dimensional and nondimensional density
σ	$\frac{1}{\beta(\mu+\mu_T)}\left(\frac{\mu}{P_r}+\frac{\mu_T}{P_{rT}}\right)$
τ	shear stress

τ^R	Reynolds stress tensor
τ^T	combined stress tensor of $\tau^R + \tau$
ε	$\frac{1}{(\gamma-1)P_r}$ for laminar equations or small parameter
ξ, η, ζ	nondimensional body-fitted coordinate directions
∂	partial derivative
$\Re$	gas constant
∇	backward differencing or gradient
Δ	forward differencing

Subscripts

g	ghost cell
int	interior cell
ℓ	represents ξ, η, ζ
L	values obtained from the left side of a cell face
n	normal
R	values obtained from the right side of a cell face
ref	reference
x, y, z	indicates derivatives relative to these Cartesian directions
∞	free-stream value

Superscripts

h	grid spacing
n	time level
v	indicates viscous-dependent values
$\breve{}$	dimensional Cartesian value
$\hat{}$	normalized
$-$	non-positive eigenvalues
$+$	non-negative eigenvalues
$\tilde{}$	nondimensional Cartesian value or Roe averaged variable
$\ddot{}$	Favre density-averaged fluctuating value
$\sim$	Favre density-averaged time-independent value

Chapter 1 INTRODUCTION

1.1 Historical Perspective

Over the past thirty to forty years the advancement in computer resources and capabilities has grown exponentially. Along with this advancement has been the development and implementation of many numerical techniques for predicting fluid flows for different geometrical configurations. As both hardware and software improve, more and more complicated problems are being analyzed. Aerodynamic flows ranging from continuum to rarefied flows are being simulated with computers. One of the first major developments for continuum flow in computational aerodynamics was the boundary integral method, also known as the panel method. Its initial use was for the solution of subsonic linearized potential flows. This method was first employed by Hess and Smith in 1962 [1] for computing flows for three-dimensional, non-lifting bodies. Panel methods were extended to lifting flows [2] for inviscid, low Mach numbers, where compressibility effects were small, and supersonic flows [3] (presently there exists integral equation methods for solving transonic flows, such as those by Kandil and Yates [4] and Kandil and Hong [5]). Transonic flows presented difficulties because the subsonic flow regions required elliptic solution techniques, and the supersonic flow regions required hyperbolic solution techniques.

In 1970, Murman and Cole [6] successfully solved the transonic small disturbance equations for transonic aerodynamic flow fields using a successive line over relaxation (SLOR) algorithm, with a scheme known as the "type difference scheme". They used central differencing for the subsonic regions and upwind, one sided differencing, for the

supersonic regions. This provided the proper biasing for the numerical differentiation of the discretized governing equations relative to the characteristic directions for information propagation. For many flow configurations the potential flow equations are sufficient, but they assume isentropic, irrotational flow which is not valid for flows containing shock waves. However, for weak shock waves, these assumptions are valid up to second-order approximations. A more precise solution of inviscid transonic flows can be obtained by using the Euler equations.

Solving the Euler equations requires more memory and computational time than the potential equations. Another landmark in 1970 was the work of Magnus and Yoshihara [7], who produced one of the first Euler computer codes accepted for computing transonic flows. They used the Lax-Wendroff scheme, which required an added artificial viscosity to remain stable. Artificial viscosity, also called artificial dissipation and numerical dissipation, is a numerical term that is related to the type of technique used in approximating the governing equations of motion, and should not be mistaken with the physical viscosity of a fluid. It will be referred to as numerical dissipation for the remainder of this dissertation. Numerical dissipation is required to stabilize numerical schemes, and is not a physical phenomena of the flow field. Another significant contribution came from MacCormack in 1969 [8], where he introduced a two stage predictor-corrector explicit scheme for iteratively solving inviscid flows about three-dimensional bodies. In 1972, MacCormack and Paullay [9] developed the rationale used for applying space discretization to the Euler equations. Another major contribution came from Beam and Warming [10], who used an implicit finite-difference algorithm to solve the conservative form of the Euler equations. A trapezoidal formula was chosen to integrate the unknown conserved variables, which produced an implicit difference equation. An Alternating Direction Implicit (ADI) method, based on those introduced by

Douglas [11], Peaceman and Rachford [12], and Douglas and Gunn [13], was used to execute the integration in time. Beam and Warming [10] incorporated a hybrid scheme which switched from central differencing, for subsonic regions, to upwind differencing whenever the local characteristic speeds were of the same sign, as is the case for supersonic regions. This is similar to the technique used by Murman and Cole [6]. The reduction of an implicit scheme into an alternating direction scheme was originally introduced by Gourlay and Mitchell in 1966 [14]. Briley and MacDonald [15] developed an equivalent alternating direction technique for solving non-linear hyperbolic equations, and applied it to the three-dimensional, compressible Navier-Stokes equations in 1974. They applied central differences to compute the spatial flux derivatives. The alternating direction method was also used by Steger [16] in 1978. He incorporated it into general curvilinear coordinates for computing transonic flow about arbitrary two-dimensional geometries using a finite difference scheme. It was also employed by Pulliam and Steger [17] in the same year for computing transonic, three-dimensional inviscid and viscous flows using a finite-difference method, with central differencing applied to the spatial flux derivatives. Beam and Warming [18] again used the alternating direction method in their finite difference scheme for computing the solution to the compressible Navier-Stokes equations, where central differencing was applied to the spatial flux derivatives.

Expanding on the concept of biasing the type of numerical differencing used on the flux vectors, were the biasing depends upon whether the flow field surrounding the point of interest is subsonic or supersonic, Steger [19] introduced the concept of splitting each flux-vector, from the conservation law form of the governing equations, into two flux-vectors. The vectors were chosen so that the Jacobian matrix of one of the split flux-vectors would contain only positive real eigenvalues, and the other only negative real eigenvalues. This type of separation allows for upwind differencing to be used for

each point of interest: backward differences would be used for terms associated with positive eigenvalues and forward differences would be used for terms associated with negative eigenvalues. Executing this type of splitting of the flux-vectors removes the need to switch between central differencing and upwind differencing, where the type of differencing chosen was based upon the local Mach number at the point of interest. Steger and Warming [20] developed this technique in what is known as flux-vector splitting in their paper in 1981. Upwind differencing is an attempt to model the directions of signal propagation. Two schemes that closely model the characteristic propagation directions are the λ-scheme, by Moretti [21], and the Split Coefficient Matrix (SCM) scheme, by Chakravarthy, Anderson, and Salas [22]. Unfortunately, these two schemes can only be applied to the non-conservative form of the governing equations, and therefore require shock fitting techniques to locate the shock waves. In 1982, van Leer [23] introduced another type of flux-vector splitting that provided smooth transitions between the split fluxes when the eigenvalues changed signs, and good shock capturing capabilities. This approach is considered a pseudo particle approach. Other types of upwind methods are those that iteratively solve the Riemann problem, such as Godunov [24] proposed in 1959. This approach is based on the shock tube membrane rupturing problem, and solves the Riemann problem at every cell face of the physical domain, searching for a shock, expansion, and/or contact wave. Another approach is to approximate the Riemann problem to a set of equations that can be solved exactly. Schemes that solve the approximate Riemann problem are known as flux-difference splitting schemes. Examples of these types of methods have been developed by Roe [25, 26], Lombard, Oliger, and Yang [27], and Engquist and Osher [28]. Both flux-vector splitting and flux-difference splitting can be applied to the conservative form of the governing equations, and therefore,

as shown by Lax [29], Lax and Richtmyer [30], and Lax and Wendroff [31], capture a shock implicitly by solving the governing equations.

A method introduced in 1981 by Jameson, Schmidt, and Turkel [32], applies central differencing to the spatial flux derivatives and used the classical four-stage Runge-Kutta time-stepping scheme for solving the governing equations. This approach was modified to require less computer memory, as shown by Jameson and Baker [33], with the knowledge that the coefficients for each time stage could be modified to provide various stability and amplification characteristics. In 1985, Jameson [34] explained that adjusting the time integration coefficients and the number of stages would alter the stability and amplification regions. He also revealed the benefits of evaluating the numerical dissipation at various stages, using a different set of coefficients than that of the time integration. This work provided a significant step for the central difference computer programs. The use of multistage Runge-Kutta methods with modified time integration coefficients has also been investigated for upwind solvers [35–37]. The flux-vector splitting and flux-difference splitting methods are becoming more popular and the multistage Runge-Kutta scheme is widely accepted.

The solution of realistic fluid dynamic problems can become CPU intensive, and as more grid points or cells are added, the amount of CPU time increases in a non-linear fashion. One approach being used to accelerate the convergence rate of iterative schemes was introduced in 1964 by Fedorenko [38]. He presented a technique called multigrid. This process was further developed by Brandt [39] for boundary value problems and applied to the small disturbance equation by South and Brandt [40] for transonic flow calculations. It was later applied to the Euler equations by Ni [41] and Jameson [34, 42]. This method has become an integral part of many steady-state flow solvers for Euler and Navier-Stokes equations, for both central-difference and upwind-differencing schemes.

As the numerical techniques advanced beyond the geometrically uncomplicated test configurations to multiple element airfoils, internal/external engine designs, and entire aircraft configurations, the grid generation process became much more difficult. Much research has been directed toward this problem [43–46]. It became apparent that developing the grid topologies for these configurations would be easier if the geometries were divided into sections or blocks and then joined together in a fashion that a computer program could analyze. This approach is often referred to as domain decomposition.

There are different types of domain decomposition. One approach is to have the grid patches or blocks overlap and/or be embedded with each other. This is also referred to as the Chimera grid scheme. One of the first to use this technique was Boppe in 1977 [47] for transonic wing flows. Others were Hedman [48] and Thompson[49]. Atta [50] and Atta and Vadyak [51] used this approach to solve the transonic full potential equations. Benek, Steger, and Dougherty [52] and Benek, Buning, and Steger [53] used the Chimera approach with the Euler equations for transonic airfoil and wing/body configurations, respectively. Eberhardt and Baganoff [54] used the embedded grid approach for a supersonic blunt cylindrical body configuration, and tried to address the problem of properly maintaining flow discontinuities across grid boundary interfaces. Maintaining conservation across grid interface boundaries is a difficult problem for the Chimera grid embedding scheme, especially for higher order extrapolations.

Another type of domain decomposition is grid or block patching. This approach does not allow the grid patches or blocks to overlap. The blocks interface along the same surface. Grid or block patching was done by Chambier, Ghazzi, Veuillot, and Viviand [55] in 1981, for a system of hyperbolic equations. They used compatibility equations to develop the interface boundary approach, which provided good results for transonic channel flows. Unfortunately this approach is not conservative and therefore unsuitable

where flow discontinuities cross block interfaces. This problem was approached by Rai, Chakravarthy, and Hessenius [56] and Rai [57]. Mastin and McConnaughey [58] investigated the issue of higher order solutions on block patched grids with C^1 continuity (lines meet one to one) at the interface with no grid discontinuities and compared these results to configurations where the blocks overlapped. A number of researchers have used the block patching method without the C^1 continuity condition [59–66]. Others, while still maintaining the C^1 continuity were able to handle a variety of complex configurations [67–70]. It should be noted that only C^1 continuity provides both higher-order extrapolations and conservation of fluxes across block interfaces.

1.2 Physical Problems of Interest

Many present day Computational Fluid Dynamics (CFD) computer codes have state-of-the-art solvers. The current effort being put forth is to take these numerically advanced computer codes and use them on more physically demanding geometries. It is no longer sufficient for a computer code to only provide analysis for a wing; it needs to be capable of including a fuselage and a nacelle. In analyzing such a configuration, one can see that these geometries cannot be accommodated with only one grid or mesh. In many cases, different grid topologies should be used for different parts of the configuration. For example, a C-O mesh may be desired for the wing, an H-O mesh for the fuselage and a polar grid inside the nacelle. To accommodate these different components, even if they were of the same mesh topology, a computer code capable of handling the different sections of the configuration separately with sufficient communication between the sections is required. This computer code must either be specially designed for this particular type of problem, or be what is generally called a multi-block computer code. A multi-block computer code would be the most accommodating. It should be flexible

enough to allow the user to divide a given configuration into many different sections, or blocks, and use a mesh topology that best suits each particular section or block. Plus, if required, each block could be handled differently, in terms of the flow solvers or governing equations. This approach is often referred to as domain decomposition.

Another configuration that requires multiple block capabilities is afterbodies, with internal nozzles. These configurations also require different mesh topologies. For the external body a polar grid is used. For the internal nozzle, a polar grid is best suited at the wall of the body for two dependent reasons. One is that at the end of the body, where the external and internal blocks meet, the meshes need to match with C^1 continuity, due to the interface constraints of the current multi-block computer code. Therefore, the best way for the grid lines to match up at interfaces is to use the same mesh topology, and due to the external geometry constraints, a polar grid is best suited. Unfortunately a polar grid in the internal nozzle is extremely demanding because the nozzle is not axisymmetric, but rectangular; therefore two mesh topologies are used in the internal nozzle. A polar mesh is used at the wall region, because it will match the external geometry with C^1 continuity and it requires packing in only one direction. This is beneficial in using Navier-Stokes equations, especially if using an algebraic turbulence model, because there will be only one coordinate direction necessary for a length scale. The second mesh topology is an H-H mesh, which will interface with the polar grid and fill in the remainder of the interior of the nozzle. This case will be explained in more detail in the results section.

Other configurations of current interest, such as the National Aerospace Plane, with its multiple scram jets, and Advanced Tactical Fighters with thrust vectoring/thrust reversing nozzles definitely require a multi-block computer code for computational analysis. In analyzing scram jets and thrust vectoring/thrust reversing nozzles, a multi-block computer code is needed to handle the different grids and the multiple boundary conditions a

particular block face might encounter. It would not be uncommon for a block face to have two or more different boundary conditions.

1.3 Objective of Dissertation

The objective of this dissertation was to develop a state-of-the-art computer code that was capable of handling all of the aforementioned configurations. The efforts were directed toward steady-state solutions, which allowed for different types of convergence accelerators to be used. The main thrust of this project was to develop a computer code that was capable of handling many different problems of various mesh topologies, configurations, and boundary conditions, without requiring any changes to the basic computer code. To achieve this, the computer code had to have grid independent subroutines that were adaptable to various boundary conditions. It had to allow for more than one type of boundary condition on a given grid surface, such as for the surface of the grid wrapping around a wing and forming a wake line or wake cut. Mesh topology independence pertains not only to one type of mesh at a time, but also the ability to handle multiple mesh topologies at the same time. To accommodate all of these desired qualities it was determined that a multi-block computer code was required. A multi-block computer code provides the flexibility of handling all of the different grid configurations, plus the interaction of cases that if not analyzed with such a computer code, would require a computer program modified just for the one specific configuration. Also, a multi-block computer code can provide the flexibility of having different mesh topologies interact. Knowing that this type of computer code is going to be used provides more flexibility for the grid generation process, by allowing the best grid topology for each particular section of the configuration being studied, without being overly concerned with what topologies the other grid sections are going to posses. This remains true even for a multi-block

computer code that requires C^1 continuity at the block interfaces. C^1 continuity does not put a restriction on the types of meshes that can be used.

1.4 Requirements of Proposed Computational Fluid Dynamics Computer Code

The proposed Computational Fluid Dynamics (CFD) computer code was to be capable of accommodating internal/external flows, wing body configurations, and afterbodies with internal nozzles. This computer code was to be mesh topology independent, allowing it to handle C-O, C-H, O-O, H-H, and H-O mesh topologies and their interactions. Although today's state-of-the-art computers permit very large memory requirements, a judicious use of computer memory is still required. As more complex configurations are tested, it is easy to have a half million to one and a half million points in a grid. In developing this computer program consideration was given to its readability.

There is a trade off at some point between legibility and computational efficiency. An example of this is in having a three-dimensional array in a corresponding set of nested "do loops". If the array is kept as a three-dimensional array, then when it is compiled only the inner "do loop" will be vectorized, but if the three-dimensional array is collapsed into a one-dimensional array then only one "do loop" is needed and instead of vectorizing a line at a time, the computer will vectorize a volume at a time. This would greatly decrease the amount of CPU time required per calculation. Unfortunately working with and manipulating a collapsed three-dimensional array can become quite cumbersome. The method adopted in the present work was to maintain the arrays in their three-dimensional form in the subroutines, attain vectorized inner "do loops" where ever possible, and write the computer code in such a manner that would allow the rearranging of the arrays in a subroutine from three-dimensional to either two-dimensional or one-dimensional, without affecting the main program or the other subroutines.

1.5 Numerical Analysis

Much has been accomplished in using the central difference operators with explicitly added dissipation for the solution of transonic flows, but as with all approaches there are a few drawbacks. One is that central difference computer codes require added second and fourth order dissipation for stability and to reduce oscillations. This requires a certain amount of numerical experimentation for tuning the coefficients. Another common problem is that due to the numerical operator's nature, it tends to smear contact discontinuities and shocks. The central-difference operators generally require more points at and in the shock so as to produce a sharp shock. A significant advantage of the central-difference methods is that they require less logic in evaluating the fluxes than the current upwind solvers; where upwind is defined as using only one-sided differences. Also, central-difference operators are generally set up with enough damping to provide sufficient smoothing, even in an explicit time integration approach. Thus, they again can save CPU effort by not requiring an implicit method just to damp the high frequency errors generated during the iteration process.

Some of the advantages of an upwind scheme are that its damping characteristics are built into the flux evaluator. There is no need to add explicit second and fourth order dissipation for stability; therefore there is no fine tuning of the dissipation. Also, by evaluating the fluxes with an upwind approach there is less smearing of shocks and contact discontinuities, and fewer points, as few as two or three, are required to adequately capture a shock than are required for a central-difference scheme. This can be very helpful in making calculations on a preliminary grid which is constructed without prior knowledge of where certain physical changes of the flow are going to occur, and yet sufficient results may still be obtained because the upwind schemes are more forgiving in areas where central-difference schemes would require more points. A possible disadvantage

of upwind schemes is that since their dissipation is built in to their formulation, there generally is no mechanism to directly decrease the amount of dissipation introduced by the scheme for a given flow condition. Different upwind schemes can have different amounts of dissipation.

Most upwind schemes have implicit time marching. So even though they may capture all of the desired features on a coarser grid than the central-difference computer codes, the central-difference computer codes can execute on a finer grid, which can provide sufficient resolution, in less time than the implicit, upwind computer code can on a coarser grid [71]. It was on this basis that an explicit upwind code was to be developed. The choice was made to use a modified Runge-Kutta explicit time integration method, very similar to that used in most central-difference computer codes. If modifying the coefficients used in the Runge-Kutta method provides sufficient damping for the general cases of interest, then the project will be considered successful.

The first topic covered in the dissertation is the development of the governing equations. Starting with the Navier-Stokes equations in dimensional Cartesian form. These equations are converted to nondimensional body-fitted coordinates, and are modified to a thin layer formulation, for all three coordinate directions. Turbulent equations are developed by executing the Favré density averaging on the Cartesian form of the full Navier-Stokes equations. The Favré density averaging was chosen because the governing equations are for compressible flow. Algebraic turbulence modelling was chosen to resolve the turbulent flows. The turbulence equations were written in thin-layer body-fitted coordinates. The Baldwin-Lomax [72] algebraic turbulence model was chosen to provide the eddy viscosity values used in the turbulence equations. An explanation of this model and the equations that are used is provided.

The two different types of upwind solvers are presented in the Numerical Schemes chapter. A brief explanation of van Leer's [23] flux-vector splitting and Roe's [25, 26] flux-difference splitting are presented. These two methods are dependent upon the method used to provide values for the splitting of the fluxes. The extrapolation method utilized in this work is the Monotone Upstream-centered Schemes for Conservative Laws (MUSCL) type differencing, which is subsequently explained. Then the time integration method used to advance the solution is provided, followed by a definition of the time step, Δt, and the different types of implicit residual smoothing, and implicit corrector smoothing.

Chapters 2 and 3 provide the governing equations and the solution methods employed in this work. Chapter 4 explains multigrid acceleration and how it is employed in accelerating a numerical solution to a steady-state. It also provides a schematic of a computer listing to show an efficient method of programming this acceleration technique that allows grid independent subroutines. This is important because it sets the foundation for expanding the computer program to have multi-block capabilities.

The multi-block structure developed for this computer program is also presented in Chapter 4. How it is implemented and a schematic of its incorporation into the computer listing is provided. The same chapter addresses the issue of the multi-block multigrid interactions, as well as provides the sequence for one multigrid cycle incorporating smoothers and multiple blocks.

The boundary conditions used for this computer program are explained in Chapter 5. Along with the standard boundary conditions, the interface requirements are provided as well. The cases studied are provided in Chapter 6. They were chosen to validate the computer program and show its flexibility to handle different geometrical configurations. The final chapter contains concluding remarks about the cases studied and the success of the computer code.

Chapter 2 GOVERNING EQUATIONS

2.1 Full Navier-Stokes Equations

This work is directed toward solving compressible fluid flows for various configurations and physical cases. Many global properties of fluid flow can be obtained from simplified equation sets, but as the interests focus to areas closer to the body and global properties are not the only interest, more physics are needed in the governing equations. This is especially true where there are geometry changes in a body or in investigating wake regions and shear layers. It is these cases that require equations which include viscous phenomenon. For early researchers in CFD, the equations of choice were the potential equations coupled with boundary-layer equations. As computer equipment improved the more versatile Navier-Stokes equations became the governing equations of choice, and they are chosen as the governing equations of fluid motion for this work. Although they are computationally more demanding, they eliminate the approximations and restrictions implied in the older potential flow-boundary-layer approaches.

The full Navier-Stokes Equations in dimensional Cartesian coordinate notation can be written as;

$$\frac{\partial \breve{Q}}{\partial \breve{t}} + \frac{\partial\left\{\breve{F} - \breve{F}_v\right\}}{\partial \breve{x}} + \frac{\partial\left\{\breve{G} - \breve{G}_v\right\}}{\partial \breve{y}} + \frac{\partial\left\{\breve{H} - \breve{H}_v\right\}}{\partial \breve{z}} \tag{2.1.1}$$

where,

$$\breve{Q} = \begin{Bmatrix} \breve{\rho} \\ \breve{\rho}\breve{u} \\ \breve{\rho}\breve{v} \\ \breve{\rho}\breve{w} \\ \breve{E} \end{Bmatrix}, \quad \breve{F} = \begin{Bmatrix} \breve{\rho}\breve{u} \\ \breve{\rho}\breve{u}^2 + \breve{p} \\ \breve{\rho}\breve{u}\breve{v} \\ \breve{\rho}\breve{u}\breve{w} \\ \left(\breve{E} + \breve{p}\right)\breve{u} \end{Bmatrix}, \quad \breve{F}_v = \begin{Bmatrix} 0 \\ \breve{\tau}_{\breve{x}\breve{x}} \\ \breve{\tau}_{\breve{x}\breve{y}} \\ \breve{\tau}_{\breve{x}\breve{z}} \\ \breve{u}\breve{\tau}_{\breve{x}\breve{x}} + \breve{v}\breve{\tau}_{\breve{x}\breve{y}} + \breve{w}\breve{\tau}_{\breve{x}\breve{z}} - \breve{q}_{\breve{x}} \end{Bmatrix} \tag{2.1.2}$$

$$\breve{G} = \begin{Bmatrix} \breve{\rho}\breve{v} \\ \breve{\rho}\breve{u}\breve{v} \\ \breve{\rho}\breve{v}^2 + \breve{p} \\ \breve{\rho}\breve{v}\breve{w} \\ \left(\breve{E} + \breve{p}\right)\breve{v} \end{Bmatrix}, \quad \breve{G}_v = \begin{Bmatrix} 0 \\ \breve{\tau}_{\breve{x}\breve{y}} \\ \breve{\tau}_{\breve{y}\breve{y}} \\ \breve{\tau}_{\breve{y}\breve{z}} \\ \breve{u}\breve{\tau}_{\breve{x}\breve{y}} + \breve{v}\breve{\tau}_{\breve{y}\breve{y}} + \breve{w}\breve{\tau}_{\breve{y}\breve{z}} - \breve{q}_{\breve{y}} \end{Bmatrix} \tag{2.1.3}$$

and

$$\breve{H} = \begin{Bmatrix} \breve{\rho}\breve{w} \\ \breve{\rho}\breve{u}\breve{w} \\ \breve{\rho}\breve{v}\breve{w} \\ \breve{\rho}\breve{w}^2 + \breve{p} \\ \left(\breve{E} + \breve{p}\right)\breve{w} \end{Bmatrix}, \quad \breve{H}_v = \begin{Bmatrix} 0 \\ \breve{\tau}_{\breve{x}\breve{z}} \\ \breve{\tau}_{\breve{y}\breve{z}} \\ \breve{\tau}_{\breve{z}\breve{z}} \\ \breve{u}\breve{\tau}_{\breve{x}\breve{z}} + \breve{v}\breve{\tau}_{\breve{y}\breve{z}} + \breve{w}\breve{\tau}_{\breve{z}\breve{z}} - \breve{q}_{\breve{z}} \end{Bmatrix} \tag{2.1.4}$$

The dimensional quantities, $\breve{\rho}$, $\breve{u}$, $\breve{v}$, $\breve{w}$, $\breve{p}$, and $\breve{E}$ are the density, Cartesian velocities, pressure, and total energy per unit volume, respectively. The dimensional shear stress is represented by $\breve{\tau}$, and $\breve{q}$ represents the dimensional heat flux. Invoking Stokes hypothesis, the shear stress terms become

$$\begin{aligned}
&\breve{\tau}_{\breve{x}\breve{x}} = \frac{2}{3}\breve{\mu}\left(2\frac{\partial\breve{u}}{\partial\breve{x}} - \frac{\partial\breve{v}}{\partial\breve{y}} - \frac{\partial\breve{w}}{\partial\breve{z}}\right), \quad \breve{\tau}_{\breve{y}\breve{y}} = \frac{2}{3}\breve{\mu}\left(2\frac{\partial\breve{v}}{\partial\breve{y}} - \frac{\partial\breve{u}}{\partial\breve{x}} - \frac{\partial\breve{w}}{\partial\breve{z}}\right), \\
&\breve{\tau}_{\breve{z}\breve{z}} = \frac{2}{3}\breve{\mu}\left(2\frac{\partial\breve{w}}{\partial\breve{z}} - \frac{\partial\breve{u}}{\partial\breve{x}} - \frac{\partial\breve{v}}{\partial\breve{y}}\right), \\
&\breve{\tau}_{\breve{x}\breve{y}} = \breve{\mu}\left(\frac{\partial\breve{u}}{\partial\breve{y}} + \frac{\partial\breve{v}}{\partial\breve{x}}\right) = \breve{\tau}_{\breve{y}\breve{x}} \qquad \breve{\tau}_{\breve{y}\breve{z}} = \breve{\mu}\left(\frac{\partial\breve{u}}{\partial\breve{z}} + \frac{\partial\breve{w}}{\partial\breve{x}}\right) = \breve{\tau}_{\breve{z}\breve{y}}, \\
&\breve{\tau}_{\breve{y}\breve{z}} = \breve{\mu}\left(\frac{\partial\breve{v}}{\partial\breve{z}} + \frac{\partial\breve{w}}{\partial\breve{y}}\right) = \breve{\tau}_{\breve{z}\breve{y}},
\end{aligned} \tag{2.1.5}$$

where $\breve{\mu}$ is the dynamic viscosity. The formulation for the heat flux term is

$$\breve{q} = -\breve{k}\breve{\nabla}\breve{T} \tag{2.1.6}$$

where $\breve{T}$ is the temperature, and the coefficient of thermal conductivity, $\breve{k}$, is given by

$$\breve{k} = \frac{\breve{c}_p\breve{\mu}}{P_r} \tag{2.1.7}$$

where $\breve{c}_p$ and P_r are the specific heat at constant pressure and the Prandtl number, respectively. These equations are coupled with the perfect gas equation of state:

$$\breve{p} = \breve{\rho}\breve{\Re}\breve{T} \tag{2.1.8}$$

where $\breve{\Re}$ is the gas constant. The assumption of a calorically perfect gas is made allowing

$$\breve{c}_v \breve{T} = \breve{e}_i \tag{2.1.9}$$

where $\breve{c}_v$ is the specific heat at constant volume, and $\breve{e}_i$ is the internal energy. Thus,

$$\breve{E} = \breve{\rho}\left[\breve{e}_i + \frac{1}{2}(\breve{u}^2 + \breve{v}^2 + \breve{w}^2) + \mathrm{PE} + \ldots\right] \tag{2.1.10}$$

where the potential energy (PE) and other terms are assumed to be small or constant and have a negligible effect; therefore

$$\begin{aligned} \breve{p} &= (\gamma - 1)\breve{\rho}\breve{e}_i = (\gamma - 1)\left[\breve{E} - \frac{1}{2}\breve{\rho}(\breve{u}^2 + \breve{v}^2 + \breve{w}^2)\right] \\ \breve{E} &= \frac{\breve{p}}{\gamma - 1} + \frac{1}{2}\breve{\rho}(\breve{u}^2 + \breve{v}^2 + \breve{w}^2) \end{aligned} \tag{2.1.11}$$

and $\gamma = \frac{\breve{c}_p}{\breve{c}_v}$ which is the ratio of specific heats.

It is more convenient to have the equations in nondimensional form, because it allows for easier scaling of the flow case of interest, especially in investigating viscous flows. Doing so allows the independent variation of such parameters as the Mach number, Reynolds number, and Prandtl number, so that the computational results can be generalized and not be restricted to a specific geometrical configuration. Also, the flow variables are normalized so that their values will be between certain limits. Defining $\breve{L}$, $\breve{\rho}_{ref}$, $\breve{T}_{ref}$, $\breve{a}_{ref}$, and $\breve{\mu}_{ref}$ to be the reference length, density, temperature, speed of sound, and dynamic viscosity, the following nondimensionalizations were employed;

$$\begin{gathered} x = \frac{\breve{x}}{\breve{L}}, \quad y = \frac{\breve{y}}{\breve{L}}, \quad z = \frac{\breve{z}}{\breve{L}}, \quad t = \frac{\breve{t}\,\breve{a}_{ref}}{\breve{L}}, \\ u = \frac{\breve{u}}{\breve{a}_{ref}}, \quad v = \frac{\breve{v}}{\breve{a}_{ref}}, \quad w = \frac{\breve{w}}{\breve{a}_{ref}}, \quad \mu = \frac{\breve{\mu}}{\breve{\mu}_{ref}}, \\ \rho = \frac{\breve{\rho}}{\breve{\rho}_{ref}}, \quad p = \frac{\breve{p}}{\breve{\rho}_{ref}\breve{a}_{ref}^2}, \quad T = \frac{\breve{T}}{\breve{T}_{ref}}, \\ E = \frac{\breve{E}}{\breve{a}_{ref}^2}, \quad H = \frac{\breve{H}}{\breve{a}_{ref}^2} \end{gathered} \tag{2.1.12}$$

These nondimensionalizations allowed the governing equations to be rewritten as:

$$\frac{\partial \tilde{Q}}{\partial t} + \frac{\partial\left\{\tilde{F} - \tilde{F}_v\right\}}{\partial x} + \frac{\partial\left\{\tilde{G} - \tilde{G}_v\right\}}{\partial y} + \frac{\partial\left\{\tilde{H} - \tilde{H}_v\right\}}{\partial z} \tag{2.1.13}$$

where

$$\tilde{Q} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{Bmatrix}, \quad \tilde{F} = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \end{Bmatrix}, \quad \tilde{F}_v = \frac{M_{ref}}{R_{ref}} \begin{Bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \frac{\mu}{\beta P_r}\frac{\partial T}{\partial x} \end{Bmatrix} \tag{2.1.14}$$

and likewise for $\tilde{G}$, $\tilde{G}_v$, $\tilde{H}$, and $\tilde{H}_v$. Here H, M_{ref}, R_{ref}, and β are the total Enthalpy per unit volume, reference Mach number, reference Reynolds number, and γ-1, respectively.

A Cartesian coordinate system may be ill-suited for many types of geometries. It can cause an inefficient use of points, and it can be very cumbersome to implement the proper boundaries for solving a given configuration. Using a curvilinear grid permits a better fitting grid around the body, and a more efficient use of cells. The cell sizes can change so that in regions of very small gradients, away from the body, large cells can be used. This allows more cells to be used near the body where the flow gradients are larger and require more cells for accuracy. The body-fitted coordinates can then be transformed into a computational domain that has equal sized cells. There are advantages to this, such as having the body surface selected as a boundary in the computational domain, allowing for easy application of the boundary conditions. Performing the transformation from Cartesian coordinates to curvilinear or body-fitted coordinates, ξ, η, and ζ is

accomplished by the following formulations:

$$\xi = \xi\{x,y,z\}, \quad \eta = \eta\{x,y,z\}, \quad \zeta = \zeta\{x,y,z\}$$
$$\begin{aligned}\frac{\partial}{\partial x} &= \xi_x\frac{\partial}{\partial \xi} + \eta_x\frac{\partial}{\partial \eta} + \zeta_x\frac{\partial}{\partial \zeta}\\ \frac{\partial}{\partial y} &= \xi_y\frac{\partial}{\partial \xi} + \eta_y\frac{\partial}{\partial \eta} + \zeta_y\frac{\partial}{\partial \zeta}\\ \frac{\partial}{\partial z} &= \xi_z\frac{\partial}{\partial \xi} + \eta_z\frac{\partial}{\partial \eta} + \zeta_z\frac{\partial}{\partial \zeta}\end{aligned} \tag{2.1.15}$$

It can be shown that the metrics are:

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

(2.1.16)

where the sub-characters x, y, and z on ξ, η, and ζ represent partial derivatives of the body-fitted coordinates relative to the sub-characters. Here J is the Jacobian of transformation;

$$J = \frac{\partial\{\xi,\eta,\zeta\}}{\partial\{x,y,z\}} = \begin{vmatrix}\xi_x & \xi_y & \xi_z\\ \eta_x & \eta_y & \eta_z\\ \zeta_x & \zeta_y & \zeta_z\end{vmatrix} \tag{2.1.17}$$

These formulations allow the governing equations to be written as;

$$\frac{\partial Q}{\partial t} + \frac{\partial\{F - F_v\}}{\partial \xi} + \frac{\partial\{G - G_v\}}{\partial \eta} + \frac{\partial\{H - H_v\}}{\partial \zeta} \tag{2.1.18}$$

where

$$\begin{aligned}Q &= \frac{\tilde{Q}}{J}, & F &= \frac{1}{J}\left(\xi_x\tilde{F} + \eta_x\tilde{G} + \zeta_x\tilde{H}\right), & F_v &= \frac{1}{J}\left(\xi_x\tilde{F}_v + \eta_x\tilde{G}_v + \zeta_x\tilde{H}_v\right)\\ & & G &= \frac{1}{J}\left(\xi_y\tilde{F} + \eta_y\tilde{G} + \zeta_y\tilde{H}\right), & G_v &= \frac{1}{J}\left(\xi_y\tilde{F}_v + \eta_y\tilde{G}_v + \zeta_y\tilde{H}_v\right)\\ & & H &= \frac{1}{J}\left(\xi_z\tilde{F} + \eta_z\tilde{G} + \zeta_z\tilde{H}\right), & H_v &= \frac{1}{J}\left(\xi_z\tilde{F}_v + \eta_z\tilde{G}_v + \zeta_z\tilde{H}_v\right)\end{aligned}$$

(2.1.19)

with

$$F = \frac{1}{J}\begin{Bmatrix} \rho U \\ \rho u U + p\xi_x \\ \rho v U + p\xi_y \\ \rho w U + p\xi_z \\ (E+p)U \end{Bmatrix}, \quad G = \frac{1}{J}\begin{Bmatrix} \rho V \\ \rho u V + p\eta_x \\ \rho v V + p\eta_y \\ \rho w V + p\eta_z \\ (E+p)V \end{Bmatrix}$$

$$H = \frac{1}{J}\begin{Bmatrix} \rho W \\ \rho u W + p\zeta_x \\ \rho v W + p\zeta_y \\ \rho w W + p\zeta_z \\ (E+p)W \end{Bmatrix} \tag{2.1.20}$$

where U, V, and W are the contravariant velocities defined as,

$$\begin{aligned} U &= u\xi_x + v\eta_x + w\zeta_x \\ V &= u\xi_y + v\eta_y + w\zeta_y \\ W &= u\xi_z + v\eta_z + w\zeta_z \end{aligned} \tag{2.1.21}$$

Due to their complexity, F_v, G_v, and H_v, are provided in Appendix A.

2.2 Thin-Layer Navier-Stokes Equations in Body-Fitted Coordinates

Many fluid flow cases do not require the full Navier-Stokes equations. Generally there is a surface that has its normal perpendicular to the main streamwise flow. It is on this surface that a boundary layer will develop; therefore the cross flow and cross derivative viscous terms may be so small as to be considered negligible in comparison to the other flow terms. To sufficiently capture the boundary layer, many points are required near the boundary, thus allowing the governing equations to accurately predict the large gradients in the boundary layer. In some cases the physics of the fluid flow may require the full Navier-Stokes equations, but if there is not a dense packing of points in all directions the gradients of the flow will still not be captured, thus producing the same solution as that of the thin-layer equations. A compromise is to have the thin-layer approximation to the Navier-Stokes equations in all three coordinate directions. One reason is that it allows for generality in different flow cases, meaning that whichever direction is going to have

a boundary layer develop, the computer program is already capable of accommodating it. Plus, if there is significant cross flow and cross derivative flow characteristics, then the computer code is ready to accommodate them as well, provided there is sufficient packing of points in the required directions.

By examining each viscous flux individually and eliminating the cross derivative terms in that flux, the three viscous, thin-layer Navier-Stokes fluxes in body-fitted nondimensional form are:

$$F_v = \frac{M_{ref}\mu}{J\,R_{ref}} \begin{Bmatrix} 0. \\ u_\xi(\phi^2) + \xi_x\Phi \\ v_\xi(\phi^2) + \xi_y\Phi \\ w_\xi(\phi^2) + \xi_z\Phi \\ \phi^2(uu_\xi + vv_\xi + ww_\xi) + \Phi U + \varepsilon T_\xi \phi^2 \end{Bmatrix} \tag{2.2.1}$$

where, $\phi^2 = \xi_x^2 + \xi_y^2 + \xi_z^2$, and, $\Phi = \frac{1}{3}(u_\xi\xi_x + v_\xi\xi_y + w_\xi\xi_z)$

$$G_v = \frac{M_{ref}\mu}{J\,R_{ref}} \begin{Bmatrix} 0. \\ u_\eta(\theta^2) + \eta_x\Theta \\ v_\eta(\theta^2) + \eta_y\Theta \\ w_\eta(\theta^2) + \eta_z\Theta \\ \theta^2(uu_\eta + vv_\eta + ww_\eta) + \Theta V + \varepsilon T_\eta \theta^2 \end{Bmatrix} \tag{2.2.2}$$

where, $\theta^2 = \eta_x^2 + \eta_y^2 + \eta_z^2$, and, $\Theta = \frac{1}{3}(u_\eta\eta_x + v_\eta\eta_y + w_\eta\eta_z)$

and

$$H_v = \frac{M_{ref}\mu}{J\,R_{ref}} \begin{Bmatrix} 0. \\ u_\zeta(\varphi^2) + \zeta_x\psi \\ v_\zeta(\varphi^2) + \zeta_y\psi \\ w_\zeta(\varphi^2) + \zeta_z\psi \\ \varphi^2(uu_\zeta + vv_\zeta + ww_\zeta) + \psi W + \varepsilon T_\zeta \varphi^2 \end{Bmatrix} \tag{2.2.3}$$

where, $\varphi^2 = \zeta_x^2 + \zeta_y^2 + \zeta_z^2$, and, $\psi = \frac{1}{3}(u_\zeta\zeta_x + v_\zeta\zeta_y + w_\zeta\zeta_z)$

and $\varepsilon = \frac{1}{(\gamma-1)P_r}$. These three fluxes can replace the F_v, G_v, and H_v in equation (2.1.18), which is the full Navier-Stokes equation in body-fitted nondimensional form. All of the other terms in that equation will remain the same.

2.3 Turbulence

There are very few flows, outside of academic cases, that are laminar and attached. Most laminar cases are separated. The majority of flow cases are turbulent, which may have attached flow even though the laminar case would be separated. In other cases the turbulent flow may be separated as well. The larger the separation region, the greater the difficulty in resolving the flow. There are two ways turbulent flows can be resolved. One is by direct simulation, which uses the Navier-Stokes equations and directly solves for the different turbulent scales, down to the mean free path. This requires a tremendous number of points and has thus far been limited to a very small set of problems, for which the Reynolds numbers are in the range of one to three thousand [73]. The second approach is to use a turbulence model that will account for all of the different turbulent scales. This approach requires the use of the Reynolds averaged Navier-Stokes equations. Turbulence models can range from algebraic eddy viscosity models, which are the simplest, to second order closure models involving a minimum of seven additional differential equations which must be solved simultaneously with the original governing flow equations.

Turbulence modeling was the approach chosen for the current work. The first step was to obtain the Reynolds averaged Navier-Stokes equations. Since the problems of interest are for compressible flows, the Favré density averaging approach was selected [73, 74].

The following formulations were used to perform the Favré density averaging:

$$\begin{aligned}
&\widetilde{f} = \frac{\overline{\rho f}}{\overline{\rho}}, \quad \widetilde{u} = \frac{\overline{\rho u}}{\overline{\rho}}, \quad \widetilde{T} = \frac{\overline{\rho T}}{\overline{\rho}}, \quad \widetilde{H} = \frac{\overline{\rho H}}{\overline{\rho}} \\
&\text{where} \quad u = \widetilde{u} + \ddot{u}, \quad T = \widetilde{T} + \ddot{T}, \quad H = \widetilde{H} + \ddot{H} \\
&\text{note} \quad p = \overline{p} + \acute{p}, \quad \rho = \overline{\rho} + \acute{\rho} \\
&\text{and} \quad \overline{(\overline{\rho} + \acute{\rho})\ddot{f}} = 0.0 \\
&\text{but} \quad \overline{\ddot{f}} \neq 0.0
\end{aligned} \tag{2.3.1}$$

Using this formulation allows the continuity equation to be written as

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial(\overline{\rho}\widetilde{u}_i)}{\partial x_i} = 0. \tag{2.3.2}$$

The momentum equations become

$$\frac{\partial\{\overline{\rho}\widetilde{u}_i\}}{\partial t} + \frac{\partial\{\overline{\rho}\widetilde{u}_i\widetilde{u}_j\}}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial\{\overline{\tau}_{ij} - \overline{\rho\ddot{u}_i\ddot{u}_j}\}}{\partial x_j} + \overline{\rho}\frac{\partial \overline{f}}{\partial x_i} \tag{2.3.3}$$

while the energy equation evolves to

$$\frac{\partial\left\{\widetilde{E}\right\}}{\partial t} + \frac{\partial\left\{\overline{\rho}\widetilde{u}_i\widetilde{H} + \overline{\rho\ddot{u}_i\ddot{H}} - \widetilde{u}_j\overline{\tau}_{ij} - \overline{\tau_{ij}\ddot{u}_j} - k\frac{\partial \overline{T}}{\partial x_i}\right\}}{\partial x_i} = 0. \tag{2.3.4}$$

Utilizing the average turbulent kinetic energy, K, and the fluctuating kinetic energy, $\ddot{K}$, where

$$\overline{\rho}K = \frac{\overline{\rho\ddot{u}_i\ddot{u}_i}}{2} = \overline{\rho\ddot{K}}, \quad \text{and} \quad \ddot{K} \equiv \frac{\ddot{u}_i\ddot{u}_i}{2} \tag{2.3.5}$$

it can be shown that

$$\widetilde{H} = \widetilde{h} + \widetilde{K} + K \tag{2.3.6}$$

and

$$\ddot{H} = \ddot{h} + \widetilde{u}_i\ddot{u}_i + \ddot{K} - K \tag{2.3.7}$$

This in turn yields

$$\overline{\rho\ddot{u}_i\ddot{H}} = \overline{\rho\ddot{u}_i\ddot{h}} - \tau_{im}^R\widetilde{u}_m + \overline{\rho\ddot{u}_i\ddot{K}} \tag{2.3.8}$$

where

$$\tau_{im}^R \equiv -\overline{\rho\ddot{u}_i\ddot{u}_m} \tag{2.3.9}$$

and is defined as the Reynolds stress tensor. Implementing the previous relation into the energy equation produces

$$\frac{\partial\left\{\widetilde{E}\right\}}{\partial t}+\frac{\partial}{\partial x_i}\left\{\overline{\rho}\,\widetilde{u}_i\widetilde{H} + \overline{\rho\ddot{u}_i\ddot{h}} - k\frac{\partial\overline{T}}{\partial x_i} - \overline{\tau}_{ij}\widetilde{u}_j - \tau_{im}^R\widetilde{u}_m - \overline{\tau_{ij}\ddot{u}_j} + \overline{\rho\ddot{u}_i\ddot{K}}\right\} = 0. \tag{2.3.10}$$

The following terms need to be accommodated in order to solve the governing equations

$$\overline{\rho\ddot{u}_i\ddot{h}},\quad \tau_{ij}^R\widetilde{u}_j,\quad \overline{\tau_{ij}\ddot{u}_i},\quad \overline{\rho\ddot{u}_i\ddot{K}} \tag{2.3.11}$$

The first term, $\overline{\rho\ddot{u}_i\ddot{h}}$, is generally considered the turbulent heat flux, and is often modelled as

$$\overline{\rho\ddot{u}_i\ddot{h}} = -k_{\mathrm{T}}\nabla\overline{T} \tag{2.3.12}$$

where k_{T} is the coefficient of turbulent thermal conductivity [73]. The second term, τ_{ij}^R, will be modelled using the Boussinesq's assumption [73, 75]:

$$\tau_{ij}^R = -\overline{\rho\ddot{u}_i\ddot{u}_j} = \mu_{\mathrm{T}}\left[\frac{\partial\widetilde{u}_j}{\partial x_i} + \frac{\partial\widetilde{u}_i}{\partial x_j} - \frac{2}{3}\frac{\partial\widetilde{u}_m}{\partial x_m}\delta_{ij}\right] - \frac{2}{3}\overline{\rho}K\delta_{ij} \tag{2.3.13}$$

where μ_{T} is the turbulent eddy viscosity and is related to k_{T} by

$$k_{\mathrm{T}} = \frac{c_p\mu_{\mathrm{T}}}{P_{r\mathrm{T}}} \tag{2.3.14}$$

with $P_{r\mathrm{T}}$ being the turbulent Prandtl number. It is common practice to combine τ_{ij}^R with $\widetilde{\tau}_{ij}$ to give

$$\widetilde{\tau}_{ij}^{\mathrm{T}} = (\mu+\mu_{\mathrm{T}})\left[\frac{\partial\widetilde{u}_j}{\partial x_i} + \frac{\partial\widetilde{u}_i}{\partial x_j} - \frac{2}{3}\frac{\partial\widetilde{u}_m}{\partial x_m}\delta_{ij}\right] - \frac{2}{3}\overline{\rho}K\delta_{ij} \tag{2.3.15}$$

The remaining two terms $\overline{\tau_{ij}\ddot{u}_j}$ and $\overline{\rho\ddot{u}_i\ddot{K}}$ are either considered negligible and therefore accommodated by the turbulence model, or they are solved for as dependent variables in higher order schemes. For the present work these terms are assumed to be accommodated by the model. In executing the Favré density averaging, the equation of state needs to be included, which in turn becomes

$$\overline{p} = (\gamma - 1)\left[\widetilde{E} - \frac{\overline{\rho}\,\widetilde{u}_i\widetilde{u}_i}{2} - \overline{\rho}K\right] \tag{2.3.16}$$

Thus, the turbulent full Navier-Stokes equations in dimensional Cartesian form become;

$$\frac{\partial q}{\partial t} + \frac{\partial(f - f_v)}{\partial x} + \frac{\partial(g - g_v)}{\partial y} + \frac{\partial(h - h_v)}{\partial z} = 0. \tag{2.3.17}$$

where

$$q = \begin{Bmatrix} \overline{\rho} \\ \overline{\rho}\,\widetilde{u} \\ \overline{\rho}\,\widetilde{v} \\ \overline{\rho}\,\widetilde{w} \\ \widetilde{E} \end{Bmatrix}, \quad f = \begin{Bmatrix} \overline{\rho}\,\widetilde{u} \\ \overline{\rho}\,\widetilde{u}^2 + \overline{p} \\ \overline{\rho}\,\widetilde{u}\widetilde{v} \\ \overline{\rho}\,\widetilde{u}\widetilde{w} \\ \left(\widetilde{E} + \overline{p}\right)\widetilde{u} \end{Bmatrix},$$

$$f_v = (\mu + \mu_{\mathrm{T}}) \begin{Bmatrix} 0. \\ \frac{2}{3}\left[2\frac{\partial\widetilde{u}}{\partial x} - \frac{\partial\widetilde{v}}{\partial y} - \frac{\partial\widetilde{w}}{\partial z}\right] + \frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} \\ \frac{\partial\widetilde{v}}{\partial x} + \frac{\partial\widetilde{u}}{\partial y} \\ \frac{\partial\widetilde{w}}{\partial x} + \frac{\partial\widetilde{u}}{\partial z} \\ \widetilde{u}\frac{2}{3}\left[2\frac{\partial\widetilde{u}}{\partial x} - \frac{\partial\widetilde{v}}{\partial y} - \frac{\partial\widetilde{w}}{\partial z}\right] + \widetilde{v}\left(\frac{\partial\widetilde{v}}{\partial x} + \frac{\partial\widetilde{u}}{\partial y}\right) + \\ \widetilde{w}\left(\frac{\partial\widetilde{w}}{\partial x} + \frac{\partial\widetilde{u}}{\partial z}\right) + \widetilde{u}\frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} + \\ \frac{(k+k_{\mathrm{T}})}{(\mu+\mu_{\mathrm{T}})}\frac{\partial\overline{T}}{\partial x} \end{Bmatrix} \tag{2.3.18}$$

$$g = \begin{Bmatrix} \overline{\rho}\widetilde{v} \\ \overline{\rho}\widetilde{u}\widetilde{v} \\ \overline{\rho}\widetilde{v}^2 + \overline{p} \\ \overline{\rho}\widetilde{v}\widetilde{w} \\ \left(\widetilde{E} + \overline{p}\right)\widetilde{v} \end{Bmatrix}, \quad g_v = (\mu + \mu_{\mathrm{T}}) \begin{Bmatrix} 0. \\ \frac{\partial \widetilde{u}}{\partial y} + \frac{\partial \widetilde{v}}{\partial x} \\ \frac{2}{3}\left[2\frac{\partial \widetilde{v}}{\partial y} - \frac{\partial \widetilde{u}}{\partial x} - \frac{\partial \widetilde{w}}{\partial z}\right] + \frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} \\ \frac{\partial \widetilde{v}}{\partial z} + \frac{\partial \widetilde{w}}{\partial y} \\ \widetilde{u}\left(\frac{\partial \widetilde{u}}{\partial y} + \frac{\partial \widetilde{v}}{\partial x}\right) + \widetilde{v}\frac{2}{3}\left[2\frac{\partial \widetilde{v}}{\partial y} - \frac{\partial \widetilde{u}}{\partial x} - \frac{\partial \widetilde{w}}{\partial z}\right] + \\ \widetilde{w}\left(\frac{\partial \widetilde{v}}{\partial z} + \frac{\partial \widetilde{w}}{\partial y}\right) + \widetilde{v}\frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} + \\ \frac{(k+k_{\mathrm{T}})}{(\mu+\mu_{\mathrm{T}})}\frac{\partial \overline{T}}{\partial y} \end{Bmatrix} \tag{2.3.19}$$

$$h = \begin{Bmatrix} \overline{\rho}\widetilde{w} \\ \overline{\rho}\widetilde{u}\widetilde{w} \\ \overline{\rho}\widetilde{v}\widetilde{w} \\ \overline{\rho}\widetilde{w}^2 + \overline{p} \\ \left(\widetilde{E} + \overline{p}\right)\widetilde{w} \end{Bmatrix}, \quad h_v = (\mu + \mu_{\mathrm{T}}) \begin{Bmatrix} 0. \\ \frac{\partial \widetilde{u}}{\partial z} + \frac{\partial \widetilde{w}}{\partial x} \\ \frac{\partial \widetilde{v}}{\partial z} + \frac{\partial \widetilde{w}}{\partial y} \\ \frac{2}{3}\left[2\frac{\partial \widetilde{w}}{\partial z} - \frac{\partial \widetilde{u}}{\partial x} - \frac{\partial \widetilde{v}}{\partial y}\right] + \frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} \\ \widetilde{u}\left(\frac{\partial \widetilde{u}}{\partial z} + \frac{\partial \widetilde{w}}{\partial x}\right) + \widetilde{v}\left(\frac{\partial \widetilde{v}}{\partial z} + \frac{\partial \widetilde{w}}{\partial y}\right) + \\ \widetilde{w}\frac{2}{3}\left[2\frac{\partial \widetilde{w}}{\partial z} - \frac{\partial \widetilde{u}}{\partial x} - \frac{\partial \widetilde{v}}{\partial y}\right] + \widetilde{w}\frac{2\overline{\rho}K}{3(\mu+\mu_{\mathrm{T}})} + \\ \frac{(k+k_{\mathrm{T}})}{(\mu+\mu_{\mathrm{T}})}\frac{\partial \overline{T}}{\partial z} \end{Bmatrix} \tag{2.3.20}$$

the "˜" has been removed for clarity, and the lower case letters q, f, f_v, g, g_v, h, and h_v are used to signify a difference between the other equation sets.

2.4 Turbulence Modelling

For the present work an algebraic turbulence model was chosen. Although these models generally do not resolve separated regions very well, they are easy to implement

and provide reasonable results for unseparated flows. Other turbulence models, such as the two equation models $K-\varepsilon$, $K-\omega$, and $K-\tau$, are generally more applicable, especially in wake regions, but they require a certain amount of adjusting for a particular class of problems and are more complicated. The second order closure models are still in their early stages and are not commonly used for complicated geometries and configurations. It was decided that an algebraic turbulence model would be sufficient for the flows that were to be tested, and that it would provide a good starting point for validating the turbulence equations. More complicated models could be incorporated in the future.

In using an algebraic turbulence model, the turbulent kinetic energy term, K, is dropped, because there is no mechanism in an algebraic turbulence model that can account for K. For consistency it is also dropped from the equation of state. Thus, writing the turbulent full Navier-Stokes equations in nondimensional Cartesian form produces;

$$\frac{\partial \tilde{Q}}{\partial t}+\frac{\partial\left(\tilde{F}-\tilde{F}_v\right)}{\partial x}+\frac{\partial\left(\tilde{G}-\tilde{G}_v\right)}{\partial y}+\frac{\partial\left(\tilde{H}-\tilde{H}_v\right)}{\partial z}=0. \tag{2.4.1}$$

$$\tilde{Q}=\left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{array}\right\},\quad \tilde{F}=\left\{\begin{array}{c}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ (E+p)u\end{array}\right\} \tag{2.4.2}$$

$$\tilde{F}_v = \frac{M_{ref}}{R_{ref}}(\mu + \mu_{\mathbf{T}}) \begin{Bmatrix} 0. \\ \frac{2}{3}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ u\frac{2}{3}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) + v\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + \\ w\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) + \sigma\frac{\partial T}{\partial x} \end{Bmatrix} \tag{2.4.3}$$

$$\tilde{G} = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (E+p)v \end{Bmatrix}, \quad \tilde{G}_v = \frac{M_{ref}}{R_{ref}}(\mu + \mu_{\mathbf{T}}) \begin{Bmatrix} 0. \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{2}{3}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ u\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + v\frac{2}{3}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) + \\ w\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) + \sigma\frac{\partial T}{\partial y} \end{Bmatrix}$$

(2.4.4)

$$\tilde{H} = \begin{Bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (E+p)w \end{Bmatrix}, \quad \tilde{H}_v = \frac{M_{ref}}{R_{ref}}(\mu + \mu_{\mathbf{T}}) \begin{Bmatrix} 0. \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{2}{3}\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \\ u\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) + v\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) + \\ w\frac{2}{3}\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) + \sigma\frac{\partial T}{\partial z} \end{Bmatrix}$$

(2.4.5)

where

$$\sigma = \frac{1}{\beta(\mu + \mu_{\mathbf{T}})}\left(\frac{\mu}{P_r} + \frac{\mu_{\mathbf{T}}}{P_{r\mathbf{T}}}\right) \tag{2.4.6}$$

In comparing with the thin-layer Navier-Stokes equations in nondimensional Cartesian form, the only difference is between the viscous fluxes, $\tilde{F}_v$, $\tilde{G}_v$, and $\tilde{H}_v$, where the viscoscity, μ, is relaced by the sum $\mu+\mu_{\mathbf{T}}$, and $\frac{1}{\beta P_r}$ is replaced with $\frac{1}{\beta(\mu+\mu_{\mathbf{T}})}\left(\frac{\mu}{P_r} + \frac{\mu_{\mathbf{T}}}{P_{r\mathbf{T}}}\right)$. Therefore, to obtain the turbulent thin-layer Navier-Stokes equations in body-fitted nondimensional form, these two simple changes need to be made to the laminar, thin-layer Navier-Stokes body-fitted nondimensional equations.

2.5 Baldwin-Lomax Algebraic Turbulence Model

The Baldwin-Lomax algebraic turbulence model [72] is a two layer eddy viscosity model. The inner layer eddy viscosity model is the Prandtl-van Driest formulation defined as:

$$\mu_{\mathbf{T}inner} = \rho l^2 |\omega| \frac{R_{ref}}{M_{ref}} \tag{2.5.1}$$

where

$$l = k_1\, y\left[1 - exp\left\{-\frac{y^+}{A^+}\right\}\right] \tag{2.5.2}$$

$|\omega|$ = the magnitude of the vorticity,

$$|\omega| = \sqrt{\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right)^2} \tag{2.5.3}$$

and

$$y^+ = y\sqrt{\frac{\rho_w \omega_{max}}{\mu_w}}\ \sqrt{\frac{R_{ref}}{M_{ref}}} \tag{2.5.4}$$

where, y is the distance normal to the wall, k_1 is von Karman's constant (0.4), A^+ is taken as a constant (26.0), and ω_{max} is the maximum vorticity along the coordinate direction normal to the wall. For a particular wall location, ρ_w and μ_w are the values of density and molecular viscosity at the wall, respectively. The original Baldwin-Lomax

algebraic turbulence model did not use ω_{max}, but instead suggested using the shear stress at the wall, τ_w. This can produce erroneous values for the turbulence model if the flow is separated; therefore using ω_{max} has been found to be more reliable, and if there isn't any flow separation on the wall, ω_{max} can be shown to approximately equal τ_w. The second part or outer layer of the Baldwin-Lomax algebraic turbulence model is the Clauser Formulation, given as;

$$\mu_{\mathbf{T}outer} = K_2 C_{cp}\, \rho\, F_{wake} F_{KLEB} \frac{R_{ref}}{M_{ref}} \tag{2.5.5}$$

where

$$F_{wake} = \left\{ \begin{array}{c} y_{max} F_{max} \\ \\ \text{or} \\ \\ \frac{C_{wk}\, y_{max} V_{dif}^2}{F_{max}} \end{array} \right\}, \quad \begin{array}{l} \text{the smallest of} \\ \text{the two values} \end{array} \tag{2.5.6}$$

with the Clauser constant K_2 = 0.0168, C_{cp} = 1.6, C_{wk} = 1.0 for transonic flow, and

$$V_{dif} = \left(\sqrt{u^2+v^2+w^2}\right)_{|max} - \left(\sqrt{u^2+v^2+w^2}\right)_{|min} \tag{2.5.7}$$

along the coordinate perpendicular to the surface at a particular wall location. For example the difference is along a constant x-location, if x is the streamwise direction. The value y_{max} corresponds to when $F(y) = F(y)_{max}$, where

$$F(y) = y|\omega|\left[1 - exp\left\{-\frac{y^+}{A^+}\right\}\right] \tag{2.5.8}$$

In the wake region the exponential term for the previous equation is set equal to zero. The Klebanoff intermittency factor is given as;

$$F_{KLEB} = \left[1 + 5.5\left[\frac{y\, C_{KLEB}}{y_{max}}\right]^6\right]^{-1} \tag{2.5.9}$$

where C_{KLEB} = 0.3.

To obtain the eddy viscosity, both the outer and the inner formulations are computed for that particular streamwise coordinate location. Then a comparison is made between the two viscosities to determine the location, starting from the wall or slip line, where the inner eddy viscosity value becomes larger than the outer eddy viscosity value. It is at this location that one switches from using the inner eddy viscosity model's values to using the values from the outer eddy viscosity model. The final eddy viscosity values are used in conjunction with the dynamic viscosity when the viscous flux derivatives are computed.

2.6 Euler Equations

The Euler equations are obtained by the simple elimination of the viscous terms, F_v, G_v, and H_v, from the nondimensional, body-fitted, thin-layer Navier-Stokes equations. This is easily accommodated in the computer code by having the viscous fluxes evaluated in a separate subroutine.

Chapter 3 NUMERICAL SCHEMES

Upwind solvers were chosen to determine the inviscid fluxes. They gather information from both sides of an interface and then, based on the characteristic directions, a blending of the gathered information is performed. In this process if there is supersonic flow passing through a cell face then information from only the upstream side of the cell face is used. If the flow is subsonic, information from both upstream and downstream of the cell face is used. This process is based on a one-dimensional analysis, and thus assumes that information passes normal to the cell face. There have been studies to use true multi-dimensional characteristic directions, but they have met with only limited success [76]. The general approach is to perform one-dimensional analyses in the three coordinate directions independently, and then sum the results from the three directions. This was the method adopted for the present work.

The viscous fluxes at the cell interfaces are evaluated using central-difference operators, because these operators are better suited for these terms, especially evaluating the second order derivatives.

3.1 van Leer's Flux-Vector Splitting

Flux-vector splitting is similar to the method of characteristics because it attempts to establish zones of influence and dependence in the flow field. Each flux-vector is split into a forward flux-vector and a backward flux-vector, allowing upwind differencing to be used for the spatial derivatives of the split fluxes. In 1982, van Leer [23] introduced a flux-vector splitting scheme that was designed to meet seven requirements. These

requirements were to provide, among other qualities, continuous split flux-vectors, so there would be smooth transitions when eigenvalues changed signs, and stationary shock structures within no more than two cells. These are qualities that the previous flux-vector splitting techniques lacked [20].

Following Ref. [23], the flux vectors F, G, and H can each be split into two vectors, a forward flux-vector based on non-negative eigenvalues, and a backward flux-vector based on non-positive eigenvalues.

$$F = F^+ + F^-,\ G = G^+ + G^-,\ H = H^+ + H^- \tag{3.1.1}$$

For local supersonic Mach numbers:

$$\begin{aligned} M_l \geq \ \ 1.0,\ \ F_l^+ = F_l,\quad F_l^- = 0 \\ M_l \leq -1.0,\ \ F_l^+ = 0,\quad\ \ F_l^- = F_l \end{aligned} \tag{3.1.2}$$

where $l = \xi, \eta$, and ζ to indicate the three coordinate directions.

For subsonic local Mach numbers, $|M_l| < 1.0$ (in general notation for body-fitted coordinates [77]), a local scaled contravariant velocity component, $\bar{u}_l$, is defined as

$$\bar{u}_l = \frac{l_x u + l_y v + l_z w}{\sqrt{l_x^2 + l_y^2 + l_z^2}} \qquad l = \xi, \eta, \zeta \tag{3.1.3}$$

where the local Mach number is given as

$$M_l = \frac{\bar{u}_l}{a} \tag{3.1.4}$$

and a is the local speed of sound. The fluxes are:

$$F_l^{\pm} = \frac{1}{J} f_{mass}^{\pm} \begin{Bmatrix} 1 \\ \hat{l}_x(-\bar{u}_l \pm 2a)/\gamma + u \\ \hat{l}_y(-\bar{u}_l \pm 2a)/\gamma + v \\ \hat{l}_z(-\bar{u}_l \pm 2a)/\gamma + w \\ f_{energy}^{\pm} \end{Bmatrix} \tag{3.1.5}$$

where,

$$\hat{l}_n = \frac{l_n}{\sqrt{l_x^2 + l_y^2 + l_z^2}}, \qquad n = x,\ y,\ z \tag{3.1.6}$$

$$f^{\pm}_{mass} = \pm \rho a \frac{1}{4}(M_l \pm 1)^2 \tag{3.1.7}$$

and

$$f^{\pm}_{energy} = \left[\frac{-\beta \bar{u}_l^2 \pm 2\beta \bar{u}_l a + 2a^2}{\gamma^2 - 1} + \frac{u^2 + v^2 + w^2}{2} \right] \tag{3.1.8}$$

Here,

$$F_\xi = F \qquad F_\eta = G \qquad F_\zeta = H \tag{3.1.9}$$

$$u_\xi = u \qquad u_\eta = v \qquad u_\zeta = w \tag{3.1.10}$$

and

$$\beta = \gamma - 1 \tag{3.1.11}$$

The "+" indicates the forward flux and the "–" indicates the backward flux.

Carrying out a central difference on each flux vector at the cell center gives:

$$\begin{aligned} RHS = & \\ -\Delta t[& F^+_{i+\frac{1}{2}} - F^+_{i-\frac{1}{2}} + F^-_{i+\frac{1}{2}} - F^-_{i-\frac{1}{2}} \\ & + G^+_{j+\frac{1}{2}} - G^+_{j-\frac{1}{2}} + G^-_{j+\frac{1}{2}} - G^-_{j-\frac{1}{2}} \\ & + H^+_{k+\frac{1}{2}} - H^+_{k-\frac{1}{2}} + H^-_{k+\frac{1}{2}} - H^-_{k-\frac{1}{2}} \ ] \end{aligned} \tag{3.1.12}$$

The present formulation, when applied to transonic and low supersonic flows, does not require the use of flux limiters for essentially oscillation free shocks. This was noticed by Anderson, Thomas, and van Leer [78], von Lavante and Haertl [79], Melson and von Lavante [80], and Cannizzaro, von Lavante, and Melson [81] and was explained in more detail by van Leer [23].

3.2 Roe's Flux-Difference Splitting

Roe's flux-difference splitting is an upwind scheme that approximates the Riemann problem at an interface between two cells by Roe's averaging procedure [82]. The spatial derivatives across a cell interface (for example, the ξ-direction) can be written conservatively as a flux balance across the cell in the form:

$$\frac{\partial F_i}{\partial \xi} = F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \tag{3.2.1}$$

where F is the numerical flux-vector at the corresponding cell interface. Following Ref. [83], the cell interface flux is evaluated as

$$F_{i+\frac{1}{2}} = \frac{1}{2}[F_R\{Q_R\} + F_L\{Q_L\} - \left|\tilde{A}\right|(\Delta Q)] \tag{3.2.2}$$

F_L and F_R are the flux vectors computed from the left and right states, and $\tilde{A}$ is the Roe averaged flux Jacobian matrix

$$\tilde{A} = A\left[\tilde{Q}\right] \tag{3.2.3}$$

where

$$A = \frac{\partial F}{\partial Q} \tag{3.2.4}$$

and

$$\left|\tilde{A}\right| = S_\xi |\Lambda| S_\xi^{-1} \tag{3.2.5}$$

The Δ refers to the difference between the state variables on the left and right sides of the cell interface (For example, $\Delta Q = Q_R - Q_L$). S_ξ and S_ξ^{-1} are the left and right eigenvector matrices, respectively, for the "ξ" direction, and Λ is the diagonal eigenvalue matrix. The present notation was adopted from Ref. [83], because it provides a simple programming strategy of these expressions. The ˜ 's refer to Roe averages computed as

$$\tilde{u} = \frac{u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{3.2.6}$$

The last term in equation (3.2.2), $\left|\tilde{A}\right|(Q_R - Q_L)$ is a damping term due to the upwind character of the scheme and is given in detail in Ref. [82] as

$$\left|\tilde{A}\right|(Q_R - Q_L) = \begin{bmatrix} \alpha_4 \\ \tilde{u}\alpha_4 + l_x\alpha_5 + \alpha_6 \\ \tilde{v}\alpha_4 + l_y\alpha_5 + \alpha_7 \\ \tilde{w}\alpha_4 + l_z\alpha_5 + \alpha_8 \\ \tilde{a}_5 \end{bmatrix} \tag{3.2.7}$$

where,

$$\tilde{a}_5 = \tilde{h}\alpha_4 + \tilde{\bar{u}}_l\alpha_5 + \tilde{u}\alpha_6 + \tilde{v}\alpha_7 + \tilde{w}\alpha_8 - \frac{\alpha_1\tilde{a}^2}{\gamma - 1} \tag{3.2.8}$$

and

$$\begin{aligned}
\alpha_1 &= \left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l\right|\left(\Delta\rho - \frac{\Delta p}{\tilde{a}^2}\right) \\
\alpha_2 &= \frac{1}{2\tilde{a}^2}\left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l + \tilde{a}\right|(\Delta p + \rho\tilde{a}\Delta\bar{u}_l) \\
\alpha_3 &= \frac{1}{2\tilde{a}^2}\left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l - \tilde{a}\right|(\Delta p - \rho\tilde{a}\Delta\bar{u}_l) \\
\alpha_4 &= \alpha_1 + \alpha_2 + \alpha_3 \\
\alpha_5 &= \tilde{a}(\alpha_2 - \alpha_3) \\
\alpha_6 &= \left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l\right|\tilde{\rho}(\Delta u - l_x\Delta\bar{u}_l) \\
\alpha_7 &= \left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l\right|\tilde{\rho}(\Delta v - l_y\Delta\bar{u}_l) \\
\alpha_8 &= \left|\frac{\sqrt{\varphi}}{J}\right|\left|\tilde{\bar{u}}_l\right|\tilde{\rho}(\Delta w - l_z\Delta\bar{u}_l)
\end{aligned} \tag{3.2.9}$$

with $\sqrt{\varphi} = \sqrt{l_x^2 + l_y^2 + l_z^2}$, for $l = \xi,\ \eta,$ or ζ, and $\tilde{h}$ representing the Roe averaged enthalpy.

3.3 MUSCL Type Differencing

Rather than determine the values of the inviscid fluxes at the cell centers and then extrapolate them to the cell interfaces, Monotone Upstream-centered Schemes for Conservative Laws (MUSCL) type differencing was used to determine the flux values at the cell interfaces. The dependent variables are extrapolated to the cell interfaces and

from these extrapolated values the fluxes at the cell interfaces are computed. This can be seen in Fig. 3.3.1. Thus,

$$\begin{aligned} F^+_{i+\frac{1}{2}} &= F^+\left(Q^+_{i+\frac{1}{2}}\right) & F^-_{i+\frac{1}{2}} &= F^-\left(Q^-_{i+\frac{1}{2}}\right) \\ G^+_{j+\frac{1}{2}} &= G^+\left(Q^+_{j+\frac{1}{2}}\right) & G^-_{j+\frac{1}{2}} &= G^-\left(Q^-_{j+\frac{1}{2}}\right) \\ H^+_{k+\frac{1}{2}} &= H^+\left(Q^+_{k+\frac{1}{2}}\right) & H^-_{k+\frac{1}{2}} &= H^-\left(Q^-_{k+\frac{1}{2}}\right) \end{aligned} \tag{3.3.1}$$

Where the "+" and "-" are used to distinguish between the two directions of extrapolation. This approach provides a more accurate blending of the fluxes, because the blending will be based on the flow values at the interface rather than on some type of weighted averaging of the flow values from the separate cell centers.

In many cases the primitive, or non-conservative, variables are extrapolated in the MUSCL approach rather than the conservative variables.

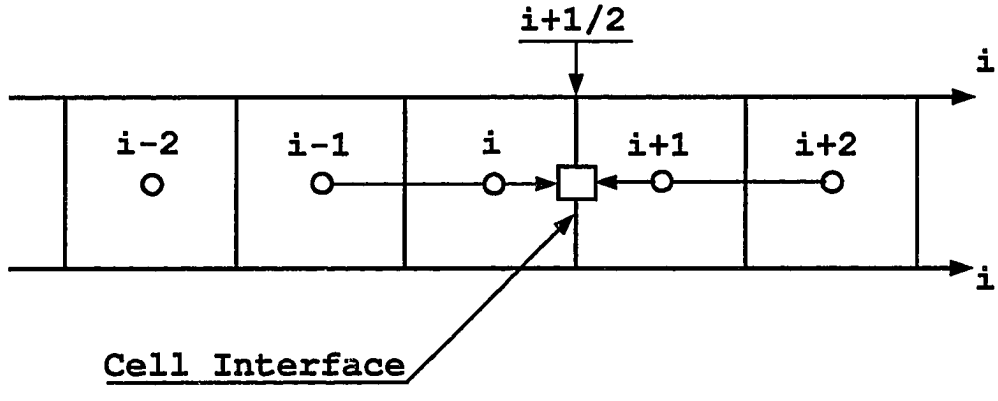


Figure 3.3.1 Schematic of MUSCL Type Differencing

The extrapolation procedure was written in the κ-scheme formulation, where

$$q_{i+\frac{1}{2}} = q_i + \frac{swtch}{4}[(1-\kappa)\nabla_i + (1+\kappa)\Delta_i] \tag{3.3.2}$$

with $\nabla_i = q_i - q_{i-1}$ and $\Delta_i = q_{i+1} - q_i$. The value of κ determines the spatial accuracy of the extrapolation; κ = -1 is pure second order accurate upwind, κ = 0 is Fromm's (1968) scheme [84], which is second order accurate upwind biased, κ = 1/3 is third

order accurate, upwind biased (it is less than third order accurate for multidimensional computations), and $\kappa = 1$ is the second order accurate central difference. If "$swtch$" is set equal to zero, then the extrapolation is first order.

3.4 Time Integration Method

There are two types of time integration schemes used to advance numerical calculations to steady state solutions. One is implicit and the other is explicit time integration. Implicit time integration schemes [11–18] require more computational work per time step, but they allow the time steps to be large and aid in the propagation of boundary information because of their elliptic nature. Many implicit schemes involve the solution of either scalar or block tri-diagonal matrices, or they approximate these matrices with bi-diagonal or even diagonal matrices [85]. Explicit methods usually require at least two computation stages, such as MacCormack's [8] predictor-corrector method, or they can have many stages, such as a Runge-Kutta multistage scheme. Modern implementation of Runge-Kutta methods can be found in the work of Jameson, Schmidt, and Turkel, [32], Jameson and Baker [33], and Jameson [34]. More stages generally permit a higher CFL number, and better error smoothing properties. With each stage, the flux evaluations are computed. After completing all of the stages, the numerical solution is advanced one time step. Most upwind methods use an implicit time integration technique because it provides better smoothing and the fluxes are evaluated only twice per time step (once for time level "n", and once for time level "n+1"). Thus, if a multistage method is employed, more stages may be required to provide comparable smoothing, causing more flux evaluations, which could lead to a higher computational effort than the implicit time integration approach. The explicit time integration schemes are generally central-difference schemes because the flux evaluations cost much less than the upwind

approach; therefore more stages can be executed at a cheaper computational cost than using an implicit time integration.

The current work was directed at having an upwind explicit computer program. Upwind methods were chosen for their accuracy, and explicit time integration was chosen in an attempt to capitalize on what the central-difference explicit schemes have accomplished, in terms of computational speed. Also, in considering a multi-block computer program, there will be many flow configurations that will not allow the implicit sweeps to be conducted across all of the blocks simultaneously. Instead, each implicit sweep would have to be performed one block at a time, reducing the effectiveness of propagating boundary information across the entire domain. Although this will change the convergence rate, it has not been reported to cause a divergence of the numerical solution.

The basic characteristics that are desired for the explicit time integration are good high frequency error damping, a minimum amount of computational effort, and robustness. The damping qualities are required for two reasons. One obviously is because the computer program will converge properly and expediently. Second, if good high frequency error damping is not achieved, multigrid acceleration will not perform properly. The amount of computational effort required is important, because if too many stages are required, it would cost less to use implicit time integration. If the scheme is not robust, it will not be generally applicable to various flow configurations, which would defeat the main purpose of developing this computer program.

To obtain these characteristics it was determined that the ability to adjust the stability range and amplification factor would provide avenues by which the explicit technique could be tuned to satisfy the necessary damping qualities. The multistage Runge-Kutta method has been modified for many of the explicit central difference codes [32, 86], and therefore was chosen as the time integration technique for the proposed computer code.

The classical fourth order Runge-Kutta time integration scheme was used by Jameson, Schmidt, and Turkel [32] for solving a finite-volume formulation of the Euler equations (It provides fourth order temporal accuracy for linear equations and second order accuracy for non-linear equations). This approach requires storing the solution at each stage as follows;

$$\begin{aligned} Q^1 &= Q^n - \frac{\Delta t}{2} RHS\{Q^n\} \\ Q^2 &= Q^n - \frac{\Delta t}{2} RHS\{Q^1\} \\ Q^3 &= Q^n - \Delta t\, RHS\{Q^2\} \\ Q^{n+1} &= Q^n - \frac{\Delta t}{6} \left[RHS\{Q^n\} + 2RHS\{Q^1\} + 2RHS\{Q^2\} + RHS\{Q^3\}\right] \end{aligned} \tag{3.4.1}$$

This would be very memory intensive for a three-dimensional computer code (where RHS represents the flux evaluations). In 1983, Jameson and Baker [33] presented the following four stage Runge-Kutta scheme;

$$\begin{aligned} Q^1 &= Q^n - \alpha_1\, \Delta t\, RHS\{Q^n\} \\ Q^2 &= Q^n - \alpha_2\, \Delta t\, RHS\{Q^1\} \\ Q^3 &= Q^n - \alpha_3\, \Delta t\, RHS\{Q^2\} \\ Q^{n+1} &= Q^n - \alpha_4\, \Delta t\, RHS\{Q^3\} \end{aligned} \tag{3.4.2}$$

The coefficients α_1, α_2, and α_3 could be independently varied to obtain a range of stability and amplification properties. The coefficient for the last stage, α_4, must equal one for consistency, and also to provide at least first order accuracy in time. This new approach was also adopted by others [87]. Initially, researchers still used the standard four stage scheme coefficients of $\alpha_1 = \frac{1}{4}$, $\alpha_2 = \frac{1}{3}$, $\alpha_3 = \frac{1}{2}$, and $\alpha_4 = 1.0$, which can be obtained from a Taylor series expansion. Attempts to obtain more desirable stability and amplification characteristics were pursued by varying the coefficients and the number of stages [42, 88, 89]. These approaches were soon investigated for upwind schemes by the present author [35] and others [36, 37]. The coefficients developed by coworkers [35] for

this computer program are presented for various extrapolations in the following tables 3.1 3.2. The linear wave equation was used as the model equation. A full explanation of the approach and detailed results can be found in the Ph.D. dissertation of Alaa Elmiligui [90], who was a coworker in the development of the computer program.

Table 3.1 Multistage Coefficients for First and Pure Second Order Schemes.

	First Order			Pure Second Order		
	Number of Stages			Number of Stages		
Multistage Coefficients	2	3	4	2	3	4
α_1	0.220	0.105	0.056	0.220	0.150	0.091
α_2	1.000	0.325	0.152	1.000	0.400	0.240
α_3		1.000	0.340		1.000	0.420
α_4			1.000			1.000

Table 3.2 Multistage Coefficients for Fromm and $\kappa = 1/3$ Schemes.

	Fromm			$\kappa = 1/3$		
	Number of Stages			Number of Stages		
Multistage Coefficients	2	3	4	2	3	4
α_1	0.420	0.210	0.110	0.460	0.220	0.135
α_2	1.000	0.440	0.255	1.000	0.480	0.260
α_3		1.000	0.46		1.000	0.440
α_4			1.000			1.000

3.5 Local Time Stepping

Local time stepping allows each cell to advance in time at its own or local stability limit. This approach provides for faster signal propagation, which produces faster convergence to a steady state solution. The local time step Δt is based on the Courant-Friedrichs-Lewy (CFL) stability limit. It is calculated as follows;

$$\frac{1}{\Delta t} \geq \frac{1}{\Delta t_\xi} + \frac{1}{\Delta t_\eta} + \frac{1}{\Delta t_\zeta} + \frac{1}{\Delta t_\xi^v} + \frac{1}{\Delta t_\eta^v} + \frac{1}{\Delta t_\zeta^v} \tag{3.5.1}$$

where

$$\frac{CFL}{\Delta t_l} \geq \lambda_l = |U_l| + a\sqrt{\phi_l^2} \tag{3.5.2}$$

with $U_l = ul_x + vl_y + wl_z$ being the contravariant velocity for the "l" direction, a the local speed of sound, λ_l the eigenvalue, and $\phi_l = \sqrt{l_x^2 + l_y^2 + l_z^2}$. The viscous contributions to the time step are:

$$\frac{CFL}{\Delta t_l^v} \geq \lambda_l^v = \frac{\mu}{\rho}\left[\phi_l\left\{max\left(\frac{4}{3}, \frac{\gamma}{Pr}\right)\right\} + \frac{1}{3}(|l_x l_y| + |l_x l_z| + |l_y l_z|)\right] \tag{3.5.3}$$

The first three terms on the right hand side of equation (3.5.1) are due to the stability limitation on the inviscid flux, while the last three terms are due to viscous flux stability limitations. The viscous time step limitation terms, $\frac{1}{\Delta t_l^v}$, make the scheme more robust on fine viscous grids for boundary-layer type flows [91, 92].

3.6 Implicit Variable Coefficient Residual Smoothing

The purpose of residual smoothing is to reduce the magnitude of any spikes in the residuals. The residuals are generated by executing the flux differentiations, and are used to adjust the values of the dependent variables. These adjustments are necessary for the dependent variables to obtain their correct values for the given flow conditions. Residual smoothing can reduce high frequency errors, which is beneficial for multigrid acceleration techniques. Reducing these errors allows the restriction process in multigrid to be implemented without causing aliasing (Aliasing is further explained in the chapter containing multigrid). Implicit residual smoothing can increase the stability region and enhance the damping properties of a multistage time-stepping scheme. The formulation for three-dimensional problems is usually applied in the form,

$$(I - \beta_\xi \nabla_\xi \Delta_\xi)(I - \beta_\eta \nabla_\eta \Delta_\eta)(I - \beta_\zeta \nabla_\zeta \Delta_\zeta) R^* = R \tag{3.6.1}$$

where R is the residual, and ∇ and Δ are the standard backward and forward difference operators relative to the ξ, η, and ζ directions. The coefficients β_ξ, β_η, and β_ζ can be

constant, which applies the same magnitude of damping on every residual value. That approach is useful but not as general as having variable coefficients. Variable coefficients self-adjust to produce damping only where needed; therefore variable-coefficient, residual-smoothing can provide better convergence than the constant coefficients without tuning. The two-dimensional variable coefficient method presented by Swanson, Turkel and White [93] was extended to three-dimensions as follows:

$$\beta_l = \left\{ \frac{1}{8} \left[\left(\frac{CFL}{CFL^*} \frac{\lambda_l}{\lambda_\xi + \lambda_\eta + \lambda_\zeta} \right) \right], \; 0. \right\} \tag{3.6.2}$$

where $l = \xi,\ \eta$, and ζ. This formulation makes β_l a function of the grid aspect ratio and the spectral radii, λ_l. The ratio of $\frac{CFL}{CFL^*}$ is the CFL of the smoothed scheme to that of the basic explicit scheme CFL^*. The optimum ratio was found to be $\frac{CFL}{CFL^*} \approx 2.0$ [93]. This operator was applied before each Runge-Kutta stage.

3.7 Implicit Corrector Smoothing

Corrector smoothing was applied using constant coefficients. This technique was to provide a better multigrid correction value from the coarser meshes, by eliminating any erroneous spikes in the correction data.

Chapter 4 MULTIGRID MULTI-BLOCK AND THEIR INTERACTION

4.1 Multigrid

Multigrid is a technique used to accelerate the rate of convergence. It has been adapted to solve the ordinary and partial differential equations found in fluid mechanics. Acceleration is achieved by attacking the low frequency errors, which generally are not well damped by the equation solver. Most equation solver techniques can damp the high frequency errors, but require more iterations to damp out the low frequency errors, and as the number of mesh or grid points increases, it takes the original equation solver many more iterations to reduce the low frequency errors. The number of iterations increases non-linearly with the number of cells. True multigrid performance does not diminish with an increase in the number of grid points. Hence, multigrid can provide a converged solution in the same number of cycles as a grid that contained only every other point.

The multigrid technique reduces the low frequency errors by solving a set of governing equations on successively coarser grids. Thus, what was a low frequency on a fine grid becomes a higher frequency on a coarser grid. In multigrid a fine grid that has had every other point eliminated in all directions is defined as a coarser grid. Thus, the equation solver is again working on high frequency error, for which it is best suited. Information gained from the coarser grids is used to reduce the low frequency errors on the finer grids. Generally three or four grid levels are used for a calculation. Also, running calculations on the coarser grids requires less computational work because of reduction in the number of points.

Multigrid acceleration techniques were originally applied to linear elliptic equations. They have since been modified to handle non-linear hyperbolic equations. Excellent developments of multigrid techniques can be found in references [94–96]. Multigrid performs a certain amount of averaging of flow variables in restricting the values from a fine grid to a coarser grid, and in prolongating the correction to the flow variables from a coarse grid to a finer grid. Thus, it is better suited for elliptic flows than for parabolic or hyperbolic flows, due to the way information is physically propagated in the flow field. At the present, multigrid techniques have had their biggest impact in transonic flows, but modifications are being introduced to handle other demanding flow cases, such as hypersonic flows.

A brief explanation of multigrid will be presented in two sections. The first section will explain multigrid methods for linear equations. This will provide the basis for the second section, which will explain multigrid methods with non-linear equations.

4.1.1 Linear Equations

Consider the problem

$$L^h U^h = f^h \tag{4.1.1}$$

where L^h is a linear, finite-difference operator on a grid, g^h, and h is the cell spacing. The forcing function, f^h, is known and U^h is the solution to the problem on the grid with spacing h. If we take u^h as an approximation to U^h with an error of V^h, i.e.

$$V^h = U^h - u^h \tag{4.1.2}$$

then equation (4.1.1) can be written as

$$L^h\left(u^h + V^h\right) = f^h \tag{4.1.3}$$

Since L^h is a linear operator, this can be written as

$$L^h\left(u^h\right) + L^h\left(V^h\right) = f^h \tag{4.1.4}$$

If V^h is a smooth function, meaning it does not have any high frequency errors, it can be represented on a coarser grid, g^{2h}, with spacing $2h$, which has twice the spacing between points as the grid with spacing h. The grid g^{2h}, is formed by removing every other point in grid g^h. Therefore, $g^{2h} \in g^h$. Points are eliminated from g^{2h} to form g^{4h} and so forth to form g^{8h}, g^{16h}, etc. Each subsequent grid is a subset of the previous grid. (If a function is not smooth, aliasing will occur during the transformation of information from the finer to the coarser grids, thus preventing an accurate representation of data from the finer grid on the coarser grid.)

It is possible to solve for an approximation to V^h on grid g^{2h}, using the equation

$$L^{2h}\left(I_h^{2h}V^h\right) = I_h^{2h}\left(f^h - L^h u^h\right) \tag{4.1.5}$$

where I_h^{2h} is the restriction operator which transfers the values of a function from the fine grid to the coarse grid (An explanation of the restriction operator and how it is implemented can be found in Appendix B). If the coarse grid forcing function is defined as

$$f^{2h} \equiv I_h^{2h}\left(f^h - L^h u^h\right) \tag{4.1.6}$$

and the coarse grid error is taken to be

$$V^{2h} = I_h^{2h}V^h \tag{4.1.7}$$

then

$$L^{2h}V^{2h} = f^{2h} \tag{4.1.8}$$

Since equation (4.1.8) is for a grid that is coarser than equation (4.1.1), the numerical evaluation of V^{2h} is much cheaper than the evaluation of V^h on the fine grid. Once V^{2h}

is obtained, it is used to correct the fine grid iterative solution, u^h, using

$$\left(u^h\right)_{new} = \left(u^h\right)_{old} + I_{2h}^h V^{2h} \tag{4.1.9}$$

The coarse grid to fine grid transfer operator, I_{2h}^h, is the prolongation operator (An explanation of the prolongation operator and how it is implemented can be found in Appendix B).

Since the form of equation (4.1.8) is the same as equation (4.1.1), it is obvious that a grid with spacing $4h$ can be used to find corrections to the "solution" of the problem on the grid with spacing $2h$. Successively coarser grids may be used until a grid is reached which is so coarse that a direct solution may be used (or a nearly exact solution with only a few relaxation sweeps). The correction from the coarsest grid is then used to correct the correction on the next finer grid; and this is continued through successively finer grids until the finest level is reached and the approximate solution is updated.

The usefulness of corrections obtained on a coarser grid is dependent on the smoothness of the fine grid error passed to the coarse grid. Hence, it is absolutely necessary that the high-frequency components of the error on the fine grid are reduced, if not completely eliminated. It is the responsibility of the smoother (usually a relaxation algorithm) to damp the high frequency components of the error. The removal of the low-frequency components of the error is unimportant for all but the coarsest grid since these frequencies can be resolved on the coarser grids where they become high-frequencies. If the high-frequencies are not damped, then the restriction operator will pass aliased information to the coarser grid and the entire multigrid scheme will cease to converge [97]. Obviously, the choice of smoother is critical to multigrid functioning properly.

4.1.2 Non-linear Equations

The previous development of the multigrid scheme was for linear operators. Unfortunately, many problems in engineering are described by non-linear equations or sets of equations. This is particularly true in the field of Computational Fluid Dynamics (CFD). Because of the non-linear nature of the equations, the Full Approximation Storage (FAS) multigrid scheme [40] must be used (FAS is applicable to both linear and non-linear problems). A brief description of FAS follows and relies heavily on the description of multigrid for linear problems given in the previous section.

In the development of multigrid for linear problems, the linearity of the operator was used to split the error out from the approximate solution as shown in the step from equation (4.1.3) to equation (4.1.4) above. If the operator is non-linear, this splitting is not valid. Instead, the derivation proceeds as follows, starting with the non-linear problem:

$$L^h U^h = f^h \tag{4.1.10}$$

Again, the substitution for the exact solution is made to give:

$$L^h\left(u^h + V^h\right) = f^h \tag{4.1.11}$$

Now, $L^h u^h$ is subtracted from both sides of equation (4.1.11) to give:

$$L^h\left(u^h + V^h\right) - L^h\left(u^h\right) = f^h - L^h\left(u^h\right) \tag{4.1.12}$$

On the coarse grid, equation (4.1.12) becomes:

$$L^{2h}\left(I_h^{2h} u^h + V^{2h}\right) - L^{2h}\left(I_h^{2h} u^h\right) = I_h^{2h}\left(f^h - L^h u^h\right) \tag{4.1.13}$$

As in the linear case, it is assumed that the error, V^h, can be represented accurately on the next coarser grid as V^{2h}. If the second term on the left-hand side is moved to the

right-hand side, equation (4.1.13) can be written as:

$$L^{2h}\left(u^{2h}\right) = f^{2h} \tag{4.1.14}$$

where

$$f^{2h} = I_h^{2h}\left(f^h - L^h u^h\right) + L^{2h}\left(I_h^{2h} u^h\right) \tag{4.1.15}$$

and

$$u^{2h} = I_h^{2h} u^h + V^{2h} \tag{4.1.16}$$

The values of u^{2h} are obtained on the coarse grid and used to update the fine grid solution using the following equation:

$$\left(u^h\right)_{new} = \left(u^h\right)_{old} + I_{2h}^h\left[u^{2h} - I_h^{2h}\left(u^h\right)_{old}\right] \tag{4.1.17}$$

Note that the prolongation term on the right-hand side of equation (4.1.17) is the correction to be applied to the fine grid solution. Examination of this term shows that the solution on the coarse grid is actually a solution to the originally posed problem and not just a correction to the fine grid solution as in the linear case. This is an important difference because it allows the use of the fine grid boundary conditions on all the coarse grids as well. As with the linear problem, the non-linear FAS scheme uses the same operator on all the grids. This of course simplifies the programming of the multigrid scheme.

The following section describes the data structure and programming approach chosen in an attempt to efficiently code the multigrid acceleration technique. The structure was then utilized to allow the inclusion of multi-block flexibility with a minimum amount of changes to the existing computer program. Setting up the data structure in this manner allows considerable flexibility in how the subroutines are coded, as will be explained in the following sections.

4.1.3 Fortran Data Structure

In explaining how the data structure was chosen, an understanding of how the memory is set up in a computer is needed. Despite the number of dimensions in an array, all data are stored in a one-dimensional array. During compilation, pointers are generated that indicate the register locations where different data can be acquired. This process is invisible to the user, but knowing how the memory process works allows one to exploit it to conserve memory space and write a general computer program that is easy to read. Knowledge of how the computer memory is arranged provides flexibility in how subroutines can be written, and be generic in terms of the different types of mesh topologies it can handle.

It is a general practice in Fortran to use common blocks to dimension arrays, and transfer those arrays from the main program to the subroutines and from subroutine to subroutine. When using common blocks, the array dimensions are set in the main program and in all the subroutines at compilation, thus fixing the size of the arrays in the subroutines. This becomes a disadvantage for a multigrid computer code, because of the numerous grid levels (generally a minimum of three). Each successively coarser grid level, for a three-dimensional computer code, has one-eighth as many grid points as the next finer grid level; therefore requiring much less memory space than for the finer grid level. The arrays could be made one-dimensional, and include all grid levels. Unfortunately this approach requires special counters and pointers to allow calculations to be preformed on the different grid levels. Another approach would be to have a separate three-dimensional array for each grid level, but it would be very cumbersome to manage. Furthermore, separate subroutines would be required to handle the different grid levels, which would obviously not be efficient. The most straightforward approach is to account

for all of the points that an array will need, including all grid levels, and set this array up as a one-dimensional array in a common block in the main program. However, instead of using common blocks in the subroutines, the main program should pass the information required by each subroutine through its argument list. When an array is passed through a subroutine argument, what is actually passed is the starting location, or address, where that array data can be found. Therefore, by passing to the subroutine the starting location and the dimensions of the array, for that particular grid level, the array can then be dimensioned in the subroutine during execution time, allowing the size of the array to change depending on the dimensions passed to the subroutine. The subroutine then deals with only that section of the array that the prescribed dimensions allow it to access.

In the main program, the arrays used for multigrid are stored as one-dimensional arrays. An integer is then created whose values are the starting addresses for the multigrid data stored in the array. Generally, the fine grid data are stored first, then the intermediate grids, and then finally the coarse grid data. A schematic drawing of this storage sequence is shown in Fig. 4.1.3.1.

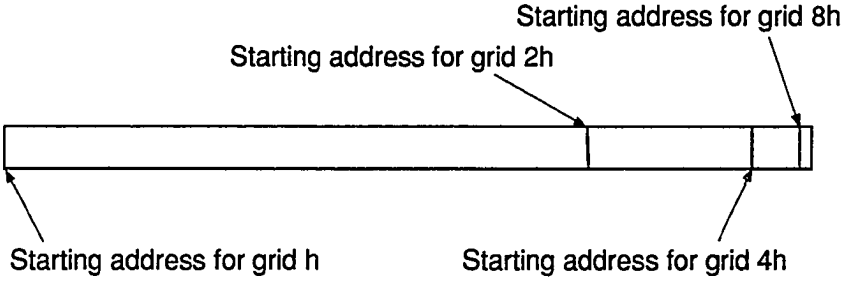


Figure 4.1.3.1 Multigrid Storage Arrangement for Arrays.

The starting locations and dimensions of the data arrays for each of the multiple grids are pre-calculated and stored in integer arrays as a function of the grid. A Fortran example is given below for the grid arrays containing the x, y, and z coordinates of the points in a three-dimensional computational domain. Several notational conventions are

used throughout the paper. The variable *ngrid* always refers to the maximum number of multigrid levels and grid level *1* refers to the finest grid.

```
      program multigrid
      parameter (ngrid=4)
      parameter (i1=97,      j1=33,      k1=17     )
      parameter (i2=(i1+1)/2, j2=(j1+1)/2, k2=(k1+1)/2)
      parameter (i3=(i2+1)/2, j3=(j2+1)/2, k3=(k2+1)/2)
      parameter (i4=(i3+1)/2, j4=(j3+1)/2, k4=(k3+1)/2)
      parameter (ijkmax=i1*j1*k1 + i2*j2*k2 +
     .                  i3*j3*k3 + i4*j4*k4)
                     .
                     .
                     .
      common /mg/ istart(ngrid),idim(ngrid)
      common /coord/ x(ijkmax)y(ijkmax),z(ijkmax)
                     .
                     .
                     .
      imax(1) = i1
      jmax(1) = j1
      kmax(1) = k1
      istart(1) = 1

      do 100 igrid=2,ngrid
      imax(igrid) = (imax(igrid-1) + 1) / 2
      jmax(igrid) = (jmax(igrid-1) + 1) / 2
      kmax(igrid) = (kmax(igrid-1) + 1) / 2
      istart(igrid) = istart(igrid-1) +
     .                imax(igrid-1)*jmax(igrid-1)*
     .                kmax(igrid-1)
  100 continue
                     .
                     .
                     .
      do 200 igrid=1,ngrid
      call metrics(x(istart(igrid)),y(istart(igrid)),
     .             z(istart(igrid)),...
     .             imax(igrid),jmax(igrid),
     .             kmax(igrid),... )
  200 continue
                     .
                     .
```

```
                .
      subroutine metrics(x,y,z,...,imax,jmax,kmax,...)
      dimension  x(imax,jmax,kmax),y(imax,jmax,kmax),
     .           z(imax,jmax,kmax)
                .
                .
```

4.1.4 V- and W-Cycles

One multigrid cycle is started by performing work, or iterations, on the finest grid level, restricting that information to the next coarser grid, performing iterations on that grid level and then continuing on to a coarser grid level. Once iterations have been performed on the coarsest grid level, the correction information is prolonged to the next finer grid level. This continues until the last iterations are performed on the finest grid level, thus completing one multigrid cycle. The cycles are repeated until sufficient convergence is obtained on the finest grid. In this section, fixed cycles known as V- and W-Cycles will be described.

In the present work, Fortran IF statements were avoided as much as possible. This led to a method of coding the multigrid cycles that relied heavily on DO loops. Basically, a standard V-Cycle can be broken into halves. The first half is the restriction part of the cycle going from the fine grid through the coarser grids down to the coarsest grid. The second half is the prolongation part of the cycle going from the coarsest grid up to the finest grid. An example is shown in Fig. 4.1.4.2 for a four level multigrid. The circles indicate when iterations are performed on the given grid level, and the lines between grid levels indicate either a restriction or prolongation operation. Notice that the circle for the fine grid at the beginning of the cycle is omitted since the iterations on the fine grid are performed at the end of the prolongation section. This ensures that the last operations

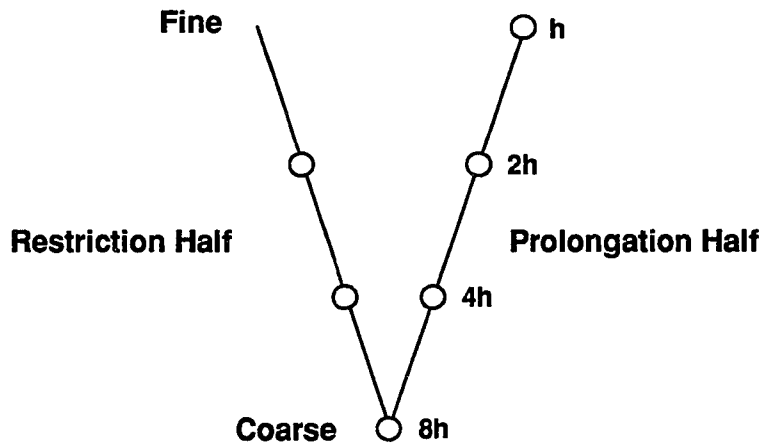


Figure 4.1.4.2 Schematic of Computation Sequence for a Four-Grid V-Cycle.

in a multigrid cycle include updates on the fine grid. The control of the grid level is handled by DO loops as shown in the following, where *ncycle* is the total number of multigrid cycles to be performed. The fine grid is *1*, the coarsest grid is *ngrid*; *iterate* is the iterative solver, *restrict* performs the restriction operation, and *prolong* performs the prolongation operation.

```
      .
      .
      .

igrid=1
call iterate(...,igrid,...)
      .
      .
      .


do 5000 icycle=1,ncycle



do 1000 igrid=2,ngrid,1
      .
      .
      .
```

```
      call restrict(...,igrid-1,igrid,....)
      call iterate(...,igrid,...)
           .
           .
           .

1000  continue
           .
           .
           .
      do 2000 igrid=ngrid-1,1,-1
           .
           .
           .
      call prolong(...,igrid+1,igrid,...)
      call iterate(...,igrid,...)
           .
           .
           .

2000  continue
5000  continue
```

It is often necessary to perform more than one iteration on a given grid level to get the required smoothness in the error for multigrid to perform correctly. For simplicity, this iteration loop has been left out of the section of code shown above.

A W-Cycle can be thought of as consisting of several components which are similar to V-Cycles but with varying 'coarsest' and 'finest' grid levels. This idea is shown in Fig. 4.1.4.3, where a W-Cycle is expanded graphically to show its 'legs'.This requires a simple coding modification to the V-Cycle program to allow W-Cycles. Another DO

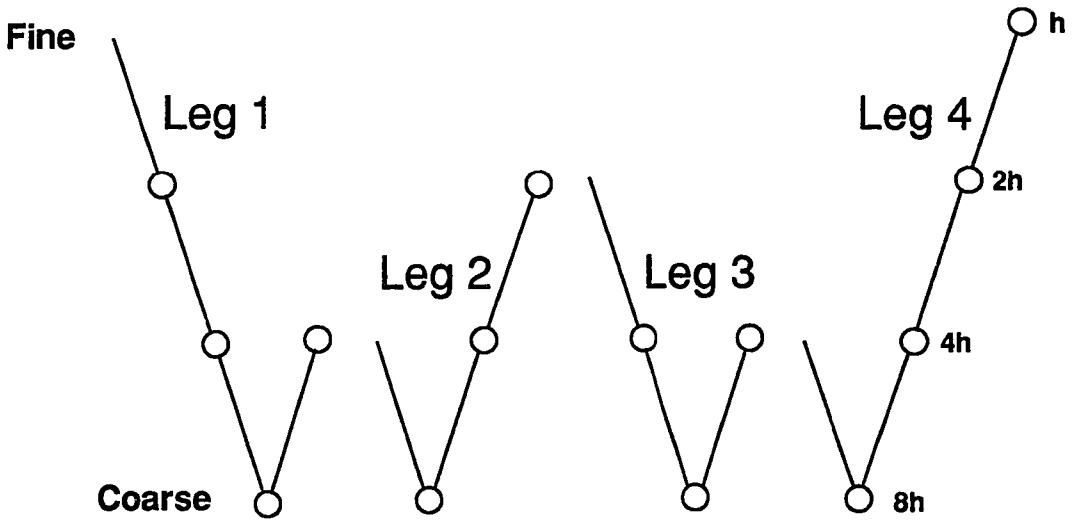


Figure 4.1.4.3 Schematic of Computation Sequence for a Four Grid W-Cycle.

loop is added to perform the various legs of the W-Cycle and then the beginning and ending grid levels of the legs are varied to create the W-Cycle. For the four-grid-level W-Cycle shown in Fig. 4.1.4.3, the fine and coarse grids, as a function of the W-Cycle 'leg', are shown in the following table. This W-Cycle has only four legs; the fifth leg

Table 4.1. – Legs for Four Level W-Cycle.

LEG ileg	FINE GRID ifine(ileg)	COARSE GRID icoarse(ileg)
1	1	4
2	3	4
3	2	4
4	3	4
(5)	(1)	

is actually the first leg of the next W-Cycle, but it is shown here for completeness. A representative listing of the Fortran coding for the W-Cycle is shown below. The variable *nleg* is the number of legs in the cycle; for example, *nleg* = 4 for the W-Cycle shown in

Fig. 4.1.4.3. The variable *ileg* specifies the current leg of the cycle and varies from *1* to *nleg*. The bold characters show the changes from the V-Cycle code to the W-Cycle code.

```
      .
      .
      .
igrid=ifine(1)
call iterate(...,igrid,...)
      .
      .
      .


do 5000 icycle=1,ncycle
do 3000 ileg=1,nleg
      .
      .
      .
do 1000 igrid=ifine(ileg)+1,icoarse(ileg),1
      .
      .
      .
call restrict(...,igrid-1,igrid,....)
call iterate(...,igrid,...)
      .
      .
      .
1000 continue
      .
      .
      .
do 2000 igrid=icoarse(ileg)-1,ifine(ileg+1),-1
      .
      .
      .
call prolong(...,igrid+1,igrid,...)
call iterate(...,igrid,...)
      .
```

```
          .
          .
2000 continue
3000 continue
5000 continue
```

One method often used to increase the rate of convergence when utilizing the multigrid process is to execute what is called full multigrid (FMG). If the grid configuration contained enough cells to support a four level V-Cycle, FMG would start by just cycling through the coarse grids; therefore the initial V-Cycle may be only a two level V-Cycle, where the two grids used are the two coarsest grids of the possible four grids. Of the two coarsest grids, the finer is treated as a solution grid; therefore its multigrid forcing function is zero. After a sufficient drop in the residual the next finer grid will be included into the process, making either a three level V-Cycle or the coarsest grid could be dropped from the cycle so that only a two level V-Cycle will still be used. It is the computer code operator's decision whether a two or three level V-Cycle is used. This process is continued until the finest grid is incorporated into the V-Cycle. The objective is that convergence will be enhanced because on just the coarser grids the solution will set up faster and also require less CPU time; therefore, once the fine grid is included into the V-Cycle, the large flow field characteristics should be developed. The FMG for a V-Cycle is shown in Fig. 4.1.4.4 and for a W-Cycle in Fig. 4.1.4.5.

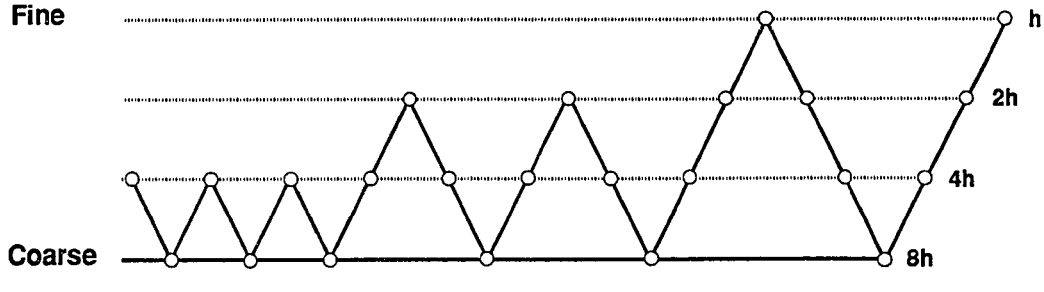


Figure 4.1.4.4 Schematic of Full Multigrid Four Grid Level V-Cycle

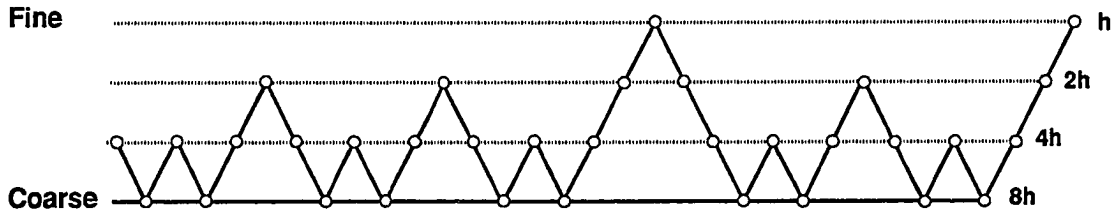


Figure 4.1.4.5 Schematic of Full Multigrid Four Grid Level W-Cycle

4.2 Multi-block Structure

The multi-block concept is best explained by considering complex grid configurations. If one considers the case of an engine mounted on a wing/body configuration, or an internal/external afterbody, or any other designs containing multiple solid surfaces, it becomes apparent that these designs cannot be handled with a simple single-block configuration. But, if each element of the configuration can be accommodated separately, the grid generation task becomes much easier. Plus, in some cases more than one type of grid topology may be required. One task is the grid generation of these configurations, and deciding what grid topology to employ. One body part may best be fit with an H-H topology, while another may be best fit with an H-O topology. A multi-block code which can handle multiple solid boundaries for various topologies and configurations, such as those mentioned in the first chapter, without requiring changes to the source code, allows one to grid a given configuration in a manner that provides an optimum topology for the different components of the grid.

A multi-block computer code allows a complex configuration to be divided into sections or blocks. Each block can be defined as a six sided volume, and can be of a different grid topology. Each side or face of the block can have different boundary conditions than the other faces. Plus, each face can be divided into multiple segments or patches, with each patch having a different boundary condition. The blocks communicate

with each other through interface conditions. Interface conditions can also be used to allow a block to communicate with itself, such as across a wake cut for a wing calculation. The simplest interface condition is for the grid lines to have C^1 continuity at the interfaces, meaning the grid lines have to match one for one (be homogeneous) across the interface, and that the grid be continuous across the interface. More complex interface conditions would allow more grid points on one block face of the interface than on the other block face of the interface. The type of interface conditions that can be allowed depends on the sophistication of the interface routine. Note that the only way to maintain higher order accuracy and flux conservation across an interface is by having a continuous grid across the block interfaces. Further discussion on interfaces will be provided in the boundary condition chapter.

The present multi-block code was designed on the premise that if it can accommodate multiple blocks, and each block can have various boundary conditions, then the source code can handle any geometrical configuration that can be represented with six-sided blocks. The block and boundary information is provided to the executable through an input file. This allows the source code to accommodate the different geometrical configurations without being altered.

Once the code was developed to handle the multigrid format, the amount of work needed to include multi-block flexibility was greatly reduced; since the subroutines were grid level independent for the multigrid structure, they will be block independent as well. The only constraints on some of the subroutines will be the boundary condition ranges, which will be passed through the argument list. The last decision before making changes which would provide the multi-block capabilities, is whether to execute all multigrid levels in each block and then proceed to the next block (this method is called executing multigrid inside of multi-block), or execute the flux evaluations for all the blocks at a

particular multigrid level then proceed to the next multigrid level for all blocks (this method is called executing multi-block inside of multigrid). It was decided that better communication would be delivered if multi-block inside of multigrid was used. This will be explained further in the multi-block multigrid interaction section.

4.2.1 Multi-Block Storage and Programing Strategy

Basically, each of the subroutines in the multigrid computer code was designed to be independent of the multigrid level. This same characteristic makes each of the subroutines independent of the block passed to it in a multi-block environment; therefore, the memory allocation scheme described in figure 4.1.3.1 is expanded to include multiple blocks as shown in figure 4.2.1.1.

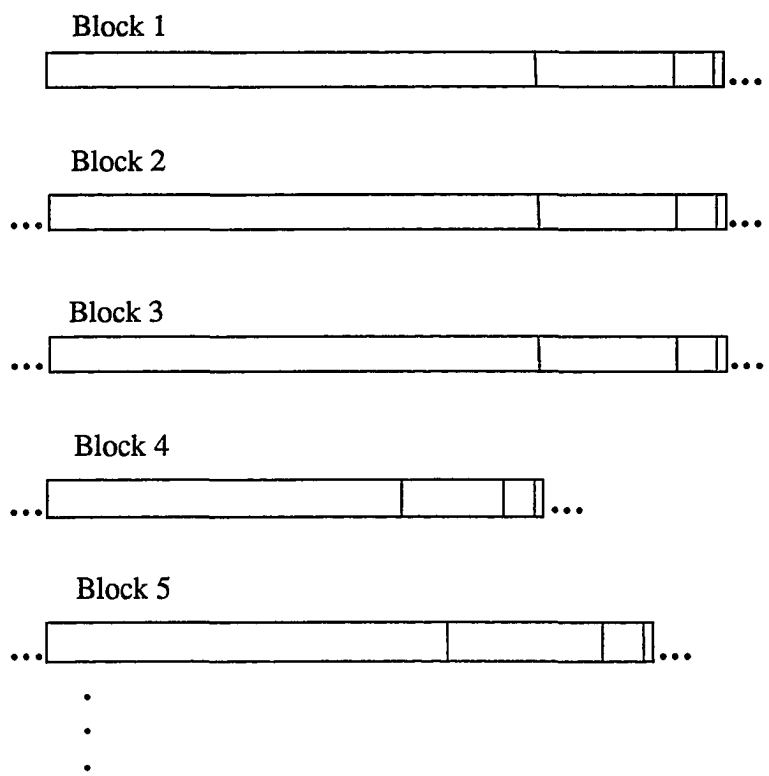


Figure 4.2.1.1 Multigrid Multi-block Storage Arrangement for Arrays.

To accommodate the extra memory needed for the multi-block strategy, an extra index is added to the pointers and parameters used for finding a specific location in an array and defining the size of the arrays, respectively. The changes to the previously shown multigrid code structure are indicated in bold print.

```
      program multigrid
      parameter (ngrid=4)
      parameter (nblock=2)
c
c     BLOCK 1
c
      parameter (i11=97,         j11=33,         k11=17        )
      parameter (i21=(i11+1)/2, j21=(j11+1)/2, k21=(k11+1)/2)
      parameter (i31=(i21+1)/2, j31=(j21+1)/2, k31=(k21+1)/2)
      parameter (i41=(i31+1)/2, j41=(j31+1)/2, k41=(k31+1)/2)
      parameter (ijkmax1=(i11*j11*k11 + i21*j21*k21 +
     .                  i31*j31*k31 + i41*j41*k41))
c
c     BLOCK 2
c
      parameter (i12=65,      j12=49,      k12=33     )
      parameter (i22=(i12+1)/2, j22=(j12+1)/2, k22=(k12+1)/2)
      parameter (i32=(i22+1)/2, j32=(j22+1)/2, k32=(k22+1)/2)
      parameter (i42=(i32+1)/2, j42=(j32+1)/2, k42=(k32+1)/2)
      parameter (ijkmax2=(i12*j12*k12 + i22*j22*k22 +
     .                  i32*j32*k32 + i42*j42*k42))
c
c     TOTALS
c
      parameter (ijkmax = ijkmax1+ijkmax2)

                  .
                  .
                  .
      common /mg/ istart(ngrid,nbloc),idim(ngrid,nbloc)
      common /coord/ x(ijkmax),y(ijkmax),z(ijkmax)
                  .
                  .
                  .
      imax(1,1) = i11
```

```
      jmax(1,1) = j11
      kmax(1,1) = k11
      istart(1,1) = 1


      do 100 igrid=2,ngrid
      imax(igrid,1) = (imax(igrid-1,1) + 1) / 2
      jmax(igrid,1) = (jmax(igrid-1,1) + 1) / 2
      kmax(igrid,1) = (kmax(igrid-1,1) + 1) / 2
      istart(igrid,1) = istart(igrid-1,1) +
     .                  imax(igrid-1,1)*jmax(igrid-1,1)*
     .                  kmax(igrid-1,1)

100   continue

      imax(1,2) = i12
      jmax(1,2) = j12
      kmax(1,2) = k12
      istart(1,2) = istart(ngrid,1) + imax(ngrid,1)*jmax(ngrid,1)*
     .                                kmax(ngrid,1)


      do 110 igrid=2,ngrid
      imax(igrid,2) = (imax(igrid-1,2) + 1) / 2
      jmax(igrid,2) = (jmax(igrid-1,2) + 1) / 2
      kmax(igrid,2) = (kmax(igrid-1,2) + 1) / 2
      istart(igrid,2) = istart(igrid-1,2) +
     .                  imax(igrid-1,2)*jmax(igrid-1,2)*
     .                  kmax(igrid-1,2)

110   continue

                .
                .
                .
      do 200 igrid=1,ngrid
      do 200 iblock=1,nblock
```

```
      call metrics(x(istart(igrid,iblock)),y(istart(igrid,iblock)),
     .             z(istart(igrid,iblock)),...
     .             imax(igrid,iblock),jmax(igrid,iblock),
     .             kmax(igrid,iblock),... )
  200 continue
                   .
                   .
                   .
      subroutine metrics(x,y,z,...,imax,jmax,kmax,...)
      dimension x(imax,jmax,kmax),y(imax,jmax,kmax),
     .          z(imax,jmax,kmax)
                   .
                   .
                   .
```

It is important to notice that in this example no changes were made to the subroutine *metrics*. Only changes to the main program were required and these involved the pointers. The important working subroutines in the computer code are the same for the multi-block program as for the multigrid computer code. New communication routines must be written to pass data between the blocks, but these are a separate issue and will be discussed in the Boundary Conditions chapter.

Once the pointers are set up to accommodate a multi-block grid, then the DO loop structure must be modified to include loops to visit all of the blocks. The example for a W-Cycle has been modified to include multi-block flexibility and is shown by the bold characters in the following example:

```
          .
          .
          .

      igrid=ifine(1)
      do 900 iblock=1,nblock
      call iterate(...,igrid,iblock,...)
```

```
      .
      .
      .


      do 5000 icycle=1,ncycle
      do 3000 ileg=1,nleg
            .
            .
            .

      do 1000 igrid=ifine(ileg)+1,icoarse(ileg),1
      do 1000 iblock=1,nblock
            .
            .
            .

      call restrict(...,igrid-1,igrid,iblock,....)
      call iterate(...,igrid,iblock,...)
            .
            .
            .
1000  continue
            .
            .
            .
      do 2000 igrid=icoarse(ileg)-1,ifine(ileg+1),-1
      do 2000 iblock=1,nblock
            .
            .
            .

      call prolong(...,igrid+1,igrid,iblock,...)
      call iterate(...,igrid,iblock,...)
            .
            .
            .
2000  continue
3000  continue
5000  continue
```

At this point the advantages of grid topology independent subroutines has become apparent. The same subroutine can be used for all levels of multigrid for each block with no special logic required in the subroutine. Having only one subroutine to handle a specified set of tasks, regardless of the multigrid level on a block, also reduces the chances of coding errors, and the overall size of the source code. Also, a grid topology independent multi-block formulation allows the same executable to calculate the flows for a variety of geometric configurations.

4.3 Multigrid Multi-Block Arrangements

As stated in the multi-block structure section, there are two strategies that can be used in programming the multi-block – multigrid interaction.

- Multigrid Inside of Multi-Block
- Multi-Block Inside of Multigrid

The first strategy is to have all multigrid levels run on a particular block before continuing on to the next block of the computational domain. This is referred to as the multigrid inside of multi-block method. This method lends itself to allowing an efficient use of memory space by computing one block at a time, writing that block out, reading in another block and operating on it before continuing to the next block. This is a very appealing way to handle computational problems that have large memory requirements. The disadvantage of this approach is that there is no communication between the coarse grid levels of the different blocks. At best this would only reduce the rate of convergence, because the lower frequencies would not be properly damped, if damped at all. At worst, the program will diverge, which becomes the case more times than not [68, 69]. Therefore, this approach was deemed unacceptable, because of its lack of robustness.

The second strategy, multi-block inside of multigrid, was chosen for this work. This approach operates with all of the blocks at the same multigrid level. Once the calculations for a particular multigrid level are completed, all of the blocks are processed to the next multigrid level, where operations are continued. A schematic of this approach is shown in Fig. 4.3.2. A problem with this approach is that if the memory requirements dictate that only one block be in use at a time, the input/output operation count becomes exceedingly high. The positive side of this approach is that this strategy mimics the same rate of exchange of information between cells as a single-block calculation. Thus, exactly the same convergence can be produced, and the original numerical efficiency of the single-block computer program is maintained, except for the time lost during the exchange of information across the block interfaces. The last statement is true only if the interface boundary conditions are of C^1 continuity, and there are no convergence acceleration techniques employed that are dependent on the placement of boundary locations.

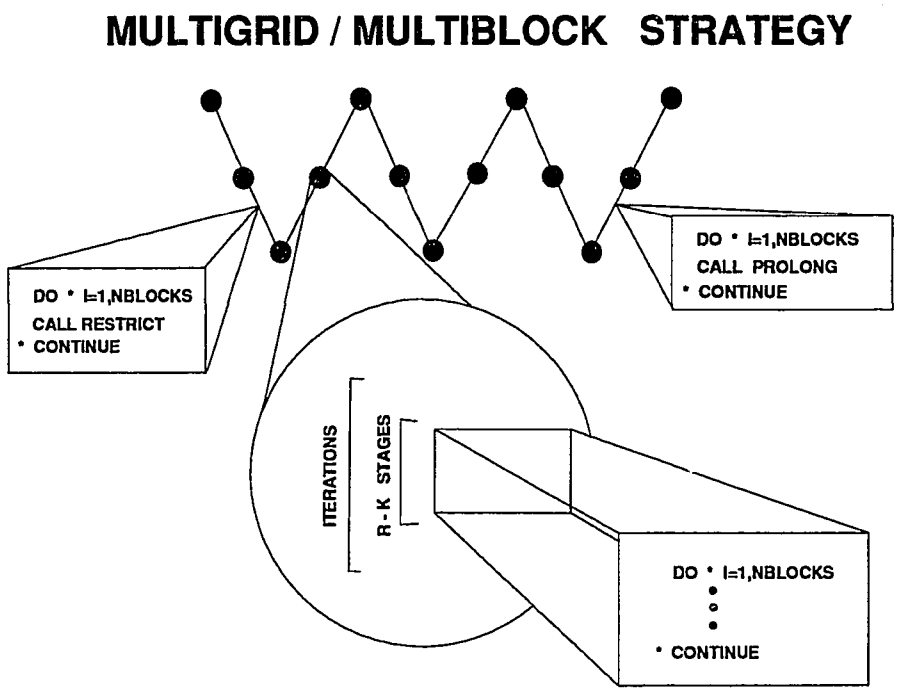


Figure 4.3.2 Multigrid Multi-block Interaction Schematic.

4.3.1 Time Integration Strategy with Multigrid

The calculation of a multigrid, four-grid, V-Cycle on a multi-block configuration, using an n-stage Runge-Kutta time integration is outlined below. As previously mentioned, the type of multigrid multi-block interaction that was employed in the present work was multi-block inside of multigrid. The governing equation is

$$N\left[\Delta u^i\right] = -\Delta t\left[L^i\left(u^i\right) - f^i\right] = RHS^i \tag{4.3.1}$$

where N is the n-stage modified Runge-Kutta time integration scheme used to smooth out the high frequency errors, and i represents h, $2h$, $4h$, $8h$, etc. to indicate the different grid spacings. The numerical derivatives of the flux-vectors are symbolized by $L^i\left(u^i\right)$, where L^i is the operator providing the Euler or thin-layer Navier-Stokes flux derivatives, and u^i represents the dependent variables. For the finest grid level, g^h, f^h is zero, because there is no multigrid forcing function for the finest grid level, only for the coarser grid levels. Starting the V-Cycle on the finest grid, g^h, iterations are performed to smooth out the high frequency errors. One iteration involves computing all stages of the modified Runge-Kutta time integration method. The flux derivatives, which are called residuals, are computed for all blocks before continuing on to the next stage of the modified Runge-Kutta method. After each stage the dependent variables are updated, as well as the boundary and interface conditions. Also, if there is to be any residual smoothing, it is incorporated before the dependent variables are updated. After completing the necessary number of iterations to remove the high frequency errors, the dependent variables and the residuals are restricted to the next grid level, g^{2h}. Equation (4.1.15) provides the relation used to obtain f^{2h}. In this relation are two types of restriction processes. One is the restriction of the dependent variables $I_h^{2h}\left(u^h\right)$, and the other is the restriction of the residuals $I_h^{2h}\left[L^h u^h\right]$. The initial values of u^{2h} are obtained on the coarse grid by the

following volume weighted averaging,

$$u^{2h} \approx I_h^{2h}\left(u^h\right) = \frac{\sum_{i=1}^{8} Vol_i^h \, u_i^h}{\sum_{i=1}^{8} Vol_i^h} \tag{4.3.2}$$

The approximation sign is used because the error term, V^{2h}, shown in equation (4.1.16), is unknown and unaccountable (This initial value of u^{2h} will be used later as part of the correction for u^h in the prolongation process from g^{2h} to g^h). The restriction of the residuals is performed by a simple summation of the residuals from the eight fine grid cells that compose a coarse grid cell, as shown in Fig. (B.1.4). The dependent variable residual restriction is given by

$$L^{2h}\left(u^{2h}\right) = I_h^{2h}\left[L^h u^h\right] = \sum_{i=1}^{8} \left[L^h u^h\right]_i \tag{4.3.3}$$

With the initial u^{2h} values, the boundary and interface conditions are computed. The iterative process is performed again, eliminating the high frequency errors so that the $2h$ grid data can be successfully restricted to the $4h$ grid. The same equations that provided the relations for the transformation from g^h to g^{2h} can be used to restrict data from g^{2h} to g^{4h}. After the high frequency errors are eliminated on this grid level, the data can be restricted to the g^{8h} level. Finishing the necessary number of iterations on g^{8h}, the prolongation operation is then performed from g^{8h} to g^{4h}. The prolongation process provides a correction to the already existing u^{4h} values. This is done by calculating the difference between u_{old}^{8h}, which was computed by the restriction process, and the u^{8h} that has been updated by the iteration process. This is accomplished by the relation given in equation (4.1.17), which is rewritten here as follows,

$$\left(u^{4h}\right)_{new} = \left(u^{4h}\right)_{old} + I_{8h}^{4h}\left[u^{8h} - I_{4h}^{8h}\left(u^{4h}\right)_{old}\right] \tag{4.3.4}$$

Then with the corrected u^{4h} more iterations are computed and then the correction data is prolonged to the next finer grid, g^{2h}. After iterating on this grid level the correction data

is finally prolonged back to the finest grid, g^h. This completes one multigrid V-Cycle. The computation of a W-Cycle is similar.

Chapter 5 BOUNDARY CONDITIONS

In general most CFD computer codes have what can be called standard inflow/outflow far field boundaries, inviscid wall, viscous wall, and symmetry plane boundary conditions. With a multi-block code there is an added interface boundary condition, which is used to exchange information between adjoining blocks. Each block is defined as a six sided volume, where each side is considered to be a face. For the present work, each face has two layers of ghost cell layers which contain boundary information. The ghost cells are used when the fluxes are being evaluated. Having two ghost cells allows higher order operations to be performed at block interfaces without any loss of accuracy or modifications. For solid wall boundaries only the ghost cell closest to the interior is used. It enforces no flow through the wall, and either parallel flow along an inviscid wall or no slip on a viscous wall. The only contribution a solid wall provides to the flux is the pressure term in the momentum equations. For the symmetry plane, inflow/outflow far field boundaries, and interfaces, the fluxes are computed at the boundary in the same manner as for an interior point, because both layers of ghost cells on the block face are used.

Boundary conditions normally drive the flow field. Ghost cells are used as an aid in enforcing boundary conditions accurately. These cells can be used in different ways. One way is to treat them as storage space to maintain dependent variable values on a wall. They are also used to maintain the far field boundary information, which controls the type of flow field the geometrical configuration will encounter.

The present control volume approach uses ghost cells as an extension of the interior cells of the computational domain. This allows the ghost cells to have assigned volume and area vectors. For external far fields, the ghost cell values are determined based on the prescribed flow conditions and interior cell values of the computational domain. The intent is to have ghost cell values that represent what the correct flow values would be in those specific x, y, and z locations for an identical physical flow field.

In treating the wall boundaries, the ghost cells again have volumes and area vectors, but rather than store flow information on the wall, they maintain values that when combined with the first interior cell, normal to the wall, provide the correct wall flow quantities. This approach is useful in evaluating the fluxes, because it allows the gathering of cell information for the first interior cell, normal to the wall, to be conducted with the same approach as any other interior cell in the domain.

5.1 Far Field Inflow/Outflow Boundaries

For the far field inflow/outflow boundaries the Mach number normal to the cell face is computed to determine if the flow is subsonic or supersonic at the inflow/outflow boundary. If the flow is supersonic then the direction of the flow is determined and a direct transfer of the reference conditions to the ghost cells is used for the inflow case. For outflow, a linear extrapolation of the interior values to the ghost cells is used. If the flow is subsonic then the one-dimensional characteristic equations are used to determine the ghost cell values. The two Riemann invariants, R^+, and R^-, are computed, and their average taken to give the velocity normal to the cell face, q_n, which is used to indicate whether the case is an inflow or outflow boundary condition. From the one-dimensional

characteristic equations, the following formulations are obtained;

$$
\begin{aligned}
q_{ref} &= u_{ref}\hat{l}_x + v_{ref}\hat{l}_y + w_{ref}\hat{l}_z \\
q_{int} &= u_{int}\hat{l}_x + v_{int}\hat{l}_y + w_{int}\hat{l}_z \\
R^+ &= q_{int} + \frac{2}{\gamma - 1}a_{int} \\
R^- &= q_{ref} - \frac{2}{\gamma - 1}a_{ref} \\
q_n &= \frac{1}{2}(R^+ + R^-) \\
\hat{l}_n &= \frac{l_n}{\sqrt{l_x^2 + l_y^2 + l_z^2}}, \quad \text{where } l = \xi,\ \eta,\ \&\ \zeta, \text{ and } n = x,\ y,\ \&\ z
\end{aligned}
\tag{5.1.1}
$$

with a being the speed of sound, and subscripts ref and int indicating reference and interior, respectively. If the normal velocity, q_n, is negative, then the boundary condition is subsonic inflow. The ghost-cell, Cartesian velocities are computed based on the reference Cartesian velocities, u_{ref}, v_{ref}, and w_{ref}, and the difference between the average Riemann normal velocity, q_n, and the reference normal velocity, q_{ref} as follows;

$$
\begin{aligned}
s^* &= \rho_{ref}^{\gamma} \,/\, p_{ref} \\
u_g &= u_{ref} + (q_n - q_{ref})\hat{l}_x \\
v_g &= v_{ref} + (q_n - q_{ref})\hat{l}_y \\
w_g &= w_{ref} + (q_n - q_{ref})\hat{l}_z
\end{aligned}
\tag{5.1.2}
$$

where s^* is an isentropically derived entropy value, and g denotes the ghost cell values. This formulation stipulates that the entropy will not change across the far field boundary. For the subsonic outflow case;

$$
\begin{aligned}
s^* &= \rho_{int}^{\gamma} \,/\, p_{int} \\
u_g &= u_{int} + (q_n - q_{int})\hat{l}_x \\
v_g &= v_{int} + (q_n - q_{int})\hat{l}_y \\
w_g &= w_{int} + (q_n - q_{int})\hat{l}_z
\end{aligned}
\tag{5.1.3}
$$

The density and total energy per unit volume are then computed as follows;

$$
\begin{aligned}
a &= \frac{1}{4}(\gamma - 1)(R^{+} - R^{-}) \\
\rho_g &= \left(\frac{a^2 s^*}{\gamma}\right)^{\frac{1}{\gamma - 1}} \\
p_g &= \frac{a^2 \rho_g}{\gamma} \\
E_g &= \frac{p_g}{\gamma - 1} + \frac{1}{2}\rho_g (u_g^2 + v_g^2 + w_g^2)
\end{aligned}
\tag{5.1.4}
$$

5.2 Symmetry Plane and Solid Wall Conditions

For the symmetry plane boundary condition, the velocity vectors of the two interior cells adjacent to the symmetry plane are reflected into the ghost cells, and the densities and total energies per unit volume are transferred directly. The inviscid wall conditions reflect the velocity vector of the first interior cell across the wall surface, maintaining the tangential velocity, just as with the symmetry condition. The pressure and density are obtained from the interior cells by either a direct transfer or a linear extrapolation. The total energy per unit volume is then computed using the ghost cell values. For the viscous wall the same procedures are used with the exception of having a no slip condition on the wall.

5.3 Block Interface Conditions

The interface conditions require that the blocks meet with C^1 continuity, meaning that the grid lines are continuous and that continuous grid metrics are maintained across block interfaces. This approach allows ghost cells of one block to receive information by direct transfer from the corresponding cells in the adjoining block; therefore the process is a one to one transfer of data, with no averaging or approximation. Having the two layers of ghost cells on each block face allows operations at the interface to be

performed in the same manner as any operation would be performed in the interior. There is no issue concerning the conservation of mass or a flux balance across the interface due to the continuity requirement across the interface. Plus, this approach provides the same order of accuracy at the interface cells that is present in the interior cells. All other interface conditions cannot provide both higher order accuracy and maintain a complete conservation of the fluxes across an interface. One advantage of this type of interface condition is that it allows the computer code to obtain exactly the same convergence for a case that could be run as a single block. This was very helpful as a debugging tool during the initial development of the computer program. The computer code was developed to allow any face of one block to interface with any face of another block. This required that the indices of one block be allowed to adjoin different index families of the adjacent block. Also the interface routine permits the indices to increase in the same direction or in opposite directions across a block interface. All that is required is that both blocks adhere to the right hand rule. More sophisticated interface conditions can be implemented by changing the interface routine to one that best suits a particular configuration's requirements. The rest of the computer code will remain the same, because at no other time is it necessary for the blocks to interact with each other. That is why the two layers of ghost cells on each block face are used. Once these cells have been updated the blocks are not required to communicate with each other until the next iteration. Each block is self-contained, allowing it to go through the flux evaluator and the prolongation and restriction multigrid routines without requiring any additional cell information from the other blocks.

Each face of a block can be divided into multiple segments or patches, and each patch can have a different boundary condition. The only constraint on these patches is that they need to be of multigridable indices, so that the physical x, y, and z locations of

the patch will not change when the solver is on a different multigrid level. This is very important, especially if one of the multiple patches is interfacing with another block.

5.4 Definition of Multigridable Index

A multigridable index is an integer value based on 2^n+1. An example of a multigridable index is the number 129, which is 2^7+1. It is multigridable because if this were the number of points in a particular coordinate direction, and every other point was eliminated, the total number of points would become 65, which is 2^6+1, and the 65^{th} point would be in the same x, y, and z location as the 129^{th} point of the finer grid. Also, if every other point were eliminated again, the total number of points would become 33, which is 2^5+1. The 33^{rd} point would also be in the exact x, y, and z location as the 65^{th} and the 129^{th} points of the finer grids. This would not be the case if the initial number of points on the fine grid was 112. If every other point is eliminated, starting from the first point, the total number of points would be 61, and the 61^{st} point would not be in the same x, y, and z location as the 112^{th} point. This is very important when multiple boundaries are used, because when patches are involved, the starting and ending index needs to be multigridable; otherwise error could be introduced into the calculations due to moving boundary locations or the exchange of incorrect information across interfaces. When the restriction process to remove every other point is executed on a 65–point grid, every value less than 65 that is a multigridable index will maintain the same x, y, and z location.

It should be noted however, that the convergence rate for a problem can change if a single-block domain is divide into multiple blocks when residual and/or corrector smoothing is being employed. The smoothers employed in the present computer code only operate on one block at a time, and if a domain has been divided into smaller segments, then the influence of boundaries at one end of the entire computational domain will take

longer to reach flow field data at the other end of the domain. For subsonic flows this could definitely slow the rate of convergence, whereas it may be beneficial to supersonic flows, based on the physical directions of information propagation. Therefore, dividing a single block into multiple blocks may provide better or worse convergence, depending on the physics of the flow when various domain boundary-dependent acceleration techniques are employed.

Chapter 6 CASES STUDIED

The flow cases presented here were chosen to validate the computer code, and display its flexability in accommodating different types of geometrical configurations. Known flow cases were chosen to validate the incorporated flow solvers. This was done to insure that the employed acceleration techniques of residual and corrector smoothing, and multigrid acceleration were not introducing error into the final numerical results. The flexibility was examined by testing a variety of geometric flow configurations. First a corner flow case was computed using a single block, then using a multi-block configuration. Next the multiple inflow boundary condition was tested, followed by a multiple boundary condition including a block interfacing with itself. The final case tested required true multi-block capabilities, coupled with the ability to accommodate an interface between blocks with different mesh topologies. All of the flow configurations were computed with the same computer code, only changes to the input file were required; therefore proving the computer code's flexibility.

6.1 Inviscid Corner Flow

The first case studied was an inviscid corner flow. The main reason this case was chosen was to demonstrate that the present multi-block method allowed the interior cells at the interfaces of the blocks to be treated as any other interior cell. This approach allowed the multi-block configuration to produce the exact same results as the single-block configuration. This was demonstrated by computing the flow field using a single-block grid configuration, and then dividing that single block into eight blocks and computing

the flow field again. By doing so, this case showed what is required of the interface routine for similar flow configurations. Finally, the three-dimensional flow effects of this case required that there be no coordinate direction biasing of the flow solution by the computer code.

A compression corner was generated by placing a compression ramp on the bottom and back walls of a rectangular duct. A schematic of the corner generated from the connection of the back and bottom walls is shown in Fig. 6.1.1. Indicated in the figure are P_b, which was taken as a pressure reference point, and Y_o, which indicates the y location at the intersection of the compression ramps. These values were used in results comparisons. The supersonic inlet flow was $M_{inlet} = 3.0$, and the ramp angles were $\alpha = 9.5°$. Thus, the grid and therefore the flow was symmetric about the intersection of the back and bottom walls, forming the compression corner. For supersonic corner

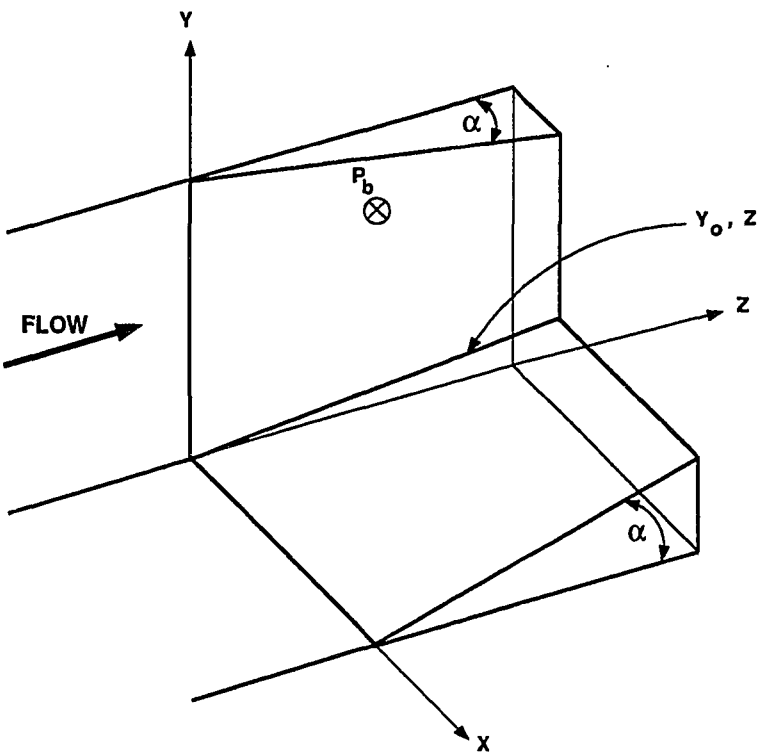


Figure 6.1.1 Schematic of Compression Corner Duct.

flow, three shock structures are produced. Two of the shock structures are wedge flow shocks, and their flow properties can be verified using two-dimensional analysis based on the Mach number normal to the leading edge of the wedge. The third structure is produced when the two wedge shocks coalesce to form a three-dimensional flow region shaped like a cone, with its apex located at the intersection of the leading edges of the compression ramps. The base of the cone is coincident with the exit plane of the duct. These characteristics can be seen in Fig. 6.1.2, where Mach line contours are projected on the back and bottom walls, and on the exit plane of a 49x49x49 channel or duct grid. This case was performed using FMG, no flux limiter, and modified Runge-Kutta time integration. The position of the wedge shocks is shown on the back and bottom walls by the region of highly concentrated Mach lines perpendicular to the inflow direction. The edges of the cone shaped shock surface can also be seen on these two walls. Four flow regions are present on the exit plane. In the upper right corner is free-stream flow, which is one-dimensional. From the middle of the plane to the lower left corner, the flow is three-dimensional, and the bottom of the cone surface, a partial disc, can be seen. The wedge shock planes, which are two-dimensional flows, can be seen in the upper left corner and the lower right corner of the exit plane. Note that since the geometry of the channel is symmetric about the compression corner (the one joining the back and bottom walls to the exit plane of the channel), the flow field should be and is symmetric about this corner. Corner flow shock structures are identified by having triple points, where the three-dimensional flow region meets the wedge shock flow and the one-dimensional free-stream flow region. The results shown here have two triple points. Shown in Fig. 6.1.3 is a schematic of how a plane parallel with the exit plane of the channel would appear. It shows the flow structure that results from this type of corner flow (figure based on information obtained from Ref. [98]). The present flow results are in good agreement

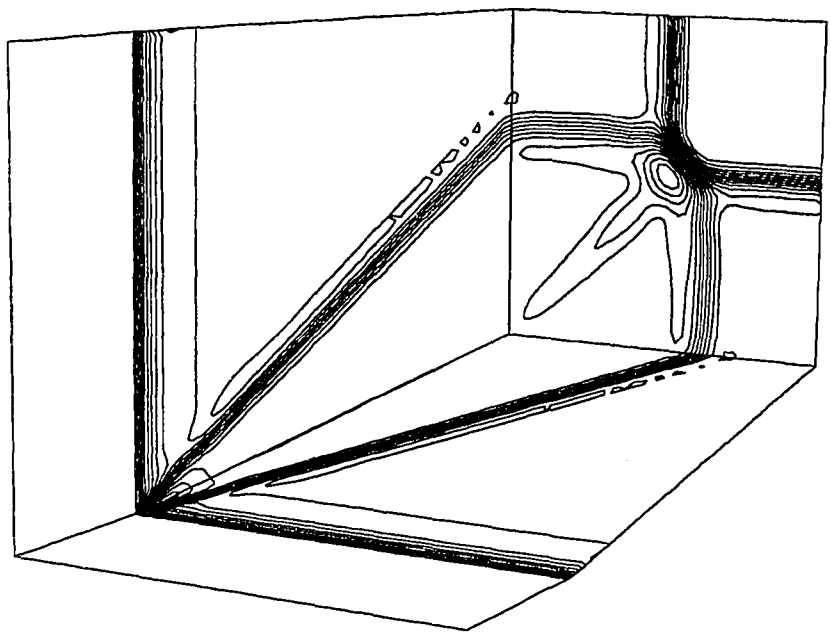


Figure 6.1.2 Mach Line Contours, $M_{inlet} = 3.0$ and $\alpha = 9.5°$.

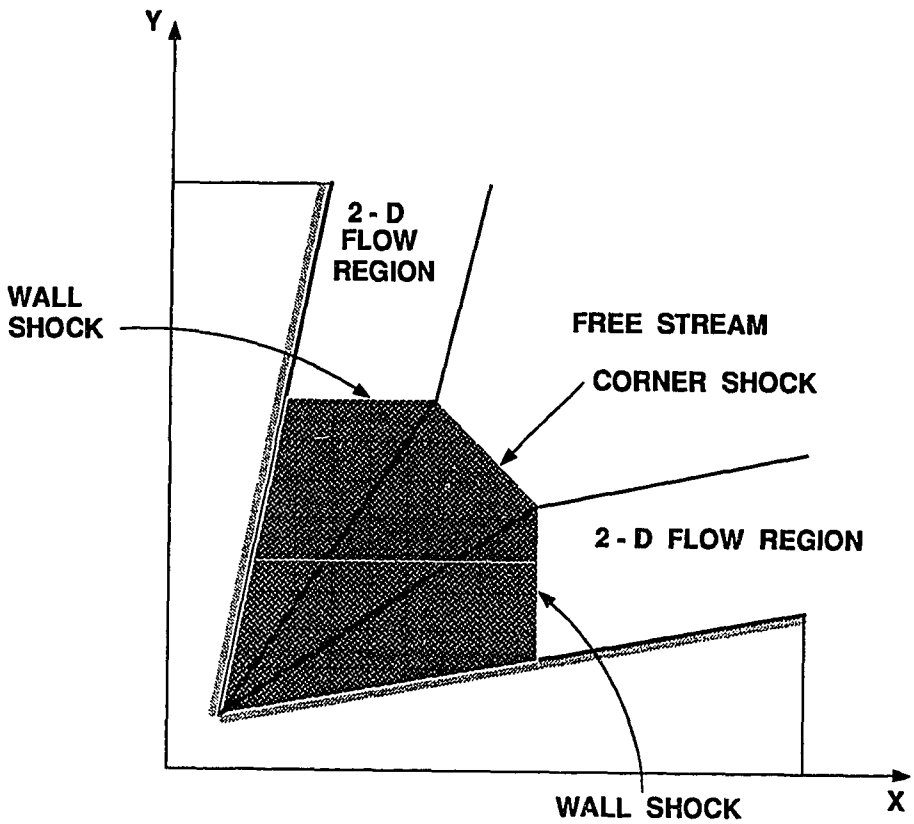


Figure 6.1.3 Compression Corner Shock.

with the numerical results of Marconi [98] and Kutler [99]. Shown in Fig. 6.1.4 is a comparison of the present work with the shock fitting numerical results of Marconi [98], the finite difference results of Kutler [99], and the experimental results of West and Korkegi [100]. It shows the relative pressure distribution on the surface of the back wall along a coordinate line that is parallel to the exit plane of the channel (Note that since the geometry of the channel is symmetric about the compression corner, either the back or bottom wall could have been used). The results of Marconi and Kutler were plotted from data presented in Ref. [98], where they solved for the corner flow in a two-dimensional plane, similar to what is shown in Fig. 6.1.3. The range of the pressure distribution for the present work is very similar to Kutler's predictions, but the shock is smeared in the present study because its structure is skewed relative to the computational grid, and the grid spacing in the streamwise direction is rather coarse.

To better capture the shock, the same test case was performed on a 65x65x65 grid. A comparison between the 49x49x49, the previously mentioned grid and a 33x33x33 grid is shown in Fig. 6.1.5, where it can be seen that the shock is less smeared on the finer grids. The packing of points in the streamwise direction has a very significant influence on this type of results comparison. Convergence histories are shown in Fig.6.1.6 for this corner flow case using different extrapolation techniques. As expected, the first-order extrapolation converged the quickest, followed by Fromm's method and κ=1/3. The residual for the pure second-order upwind method became hung after converging nine orders.

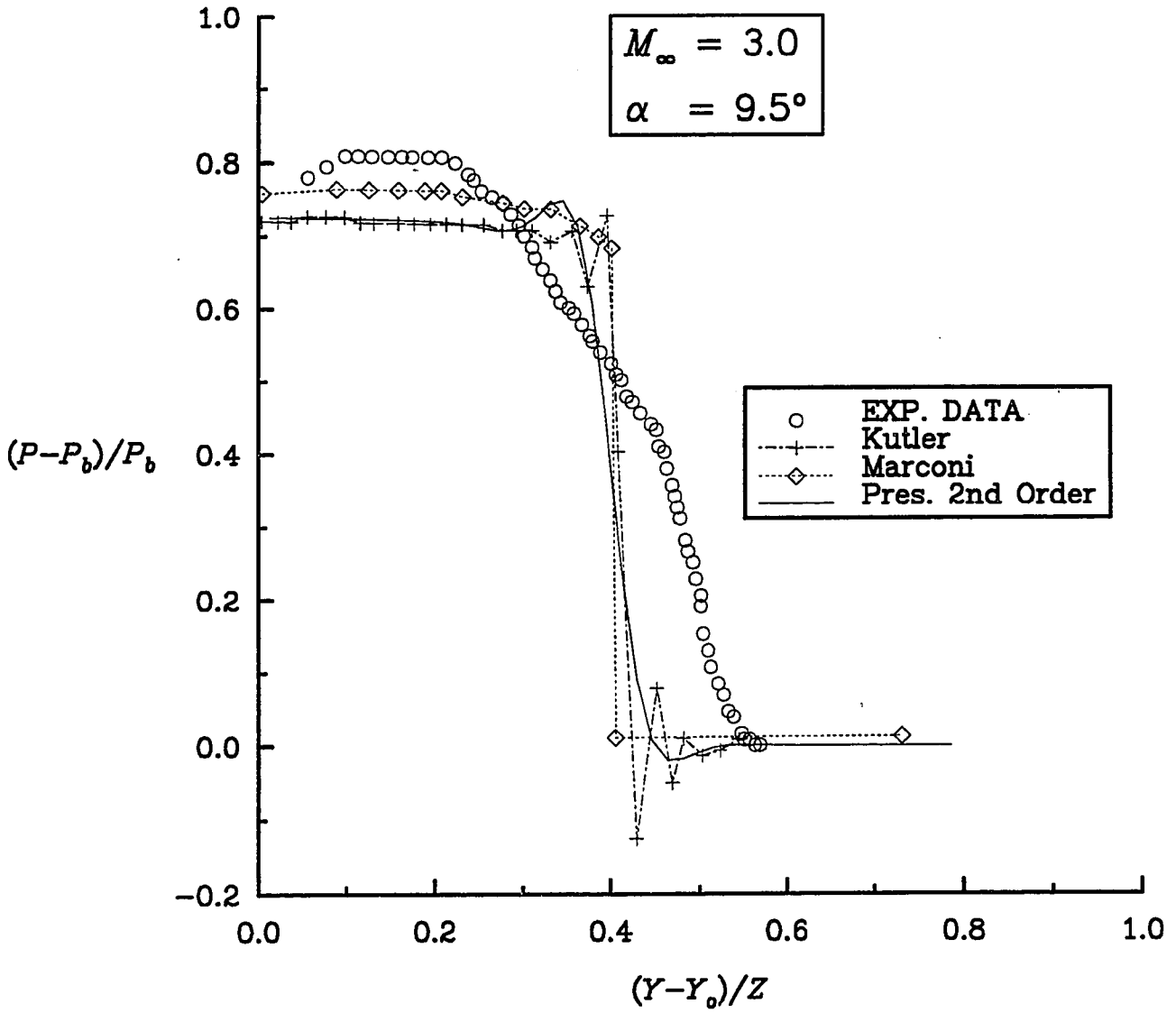


Figure 6.1.4 Comparison of Relative Pressure Distributions on Wall of Corner Flow.

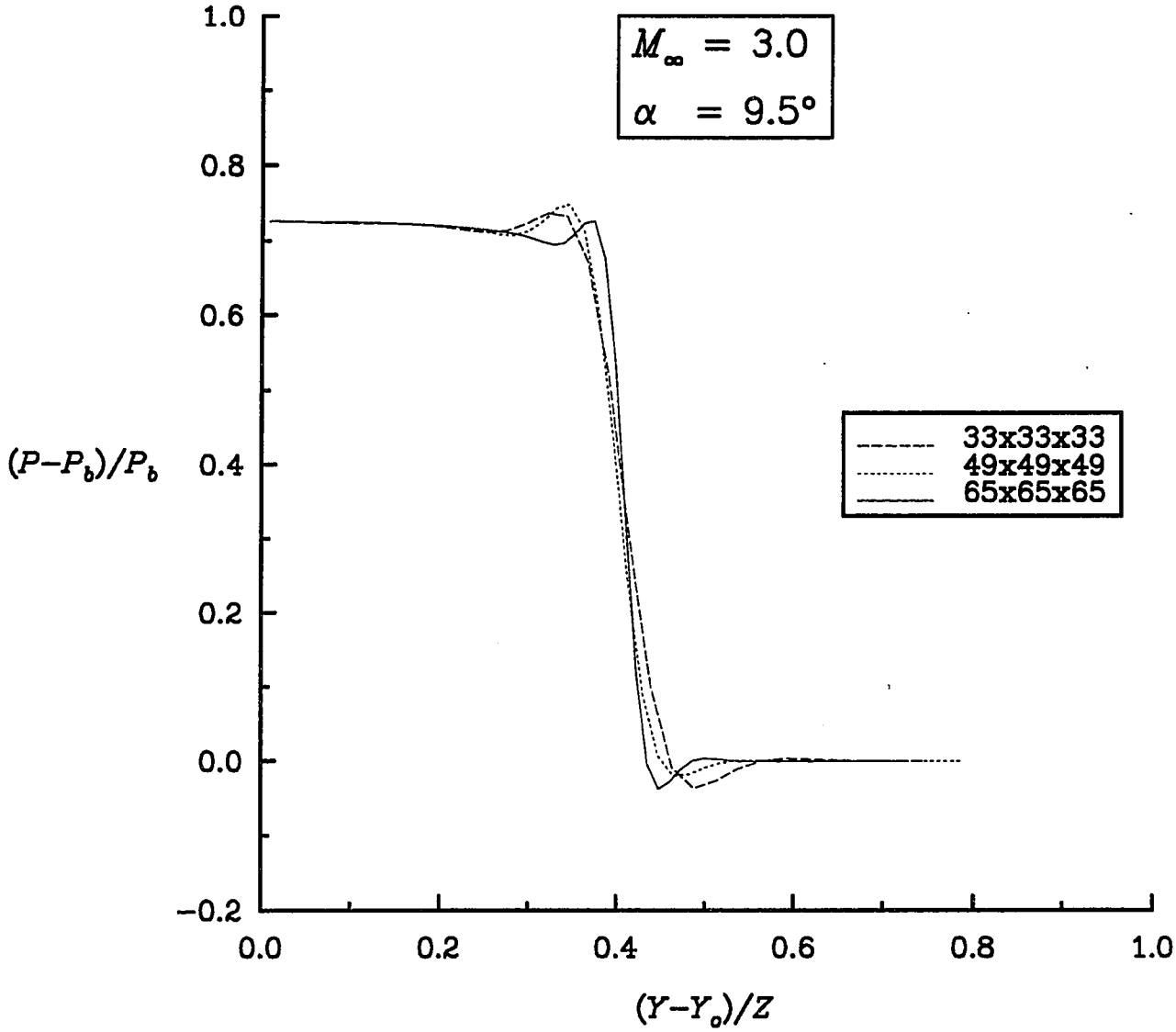


Figure 6.1.5 Grid Refinement Study on Corner Flow.

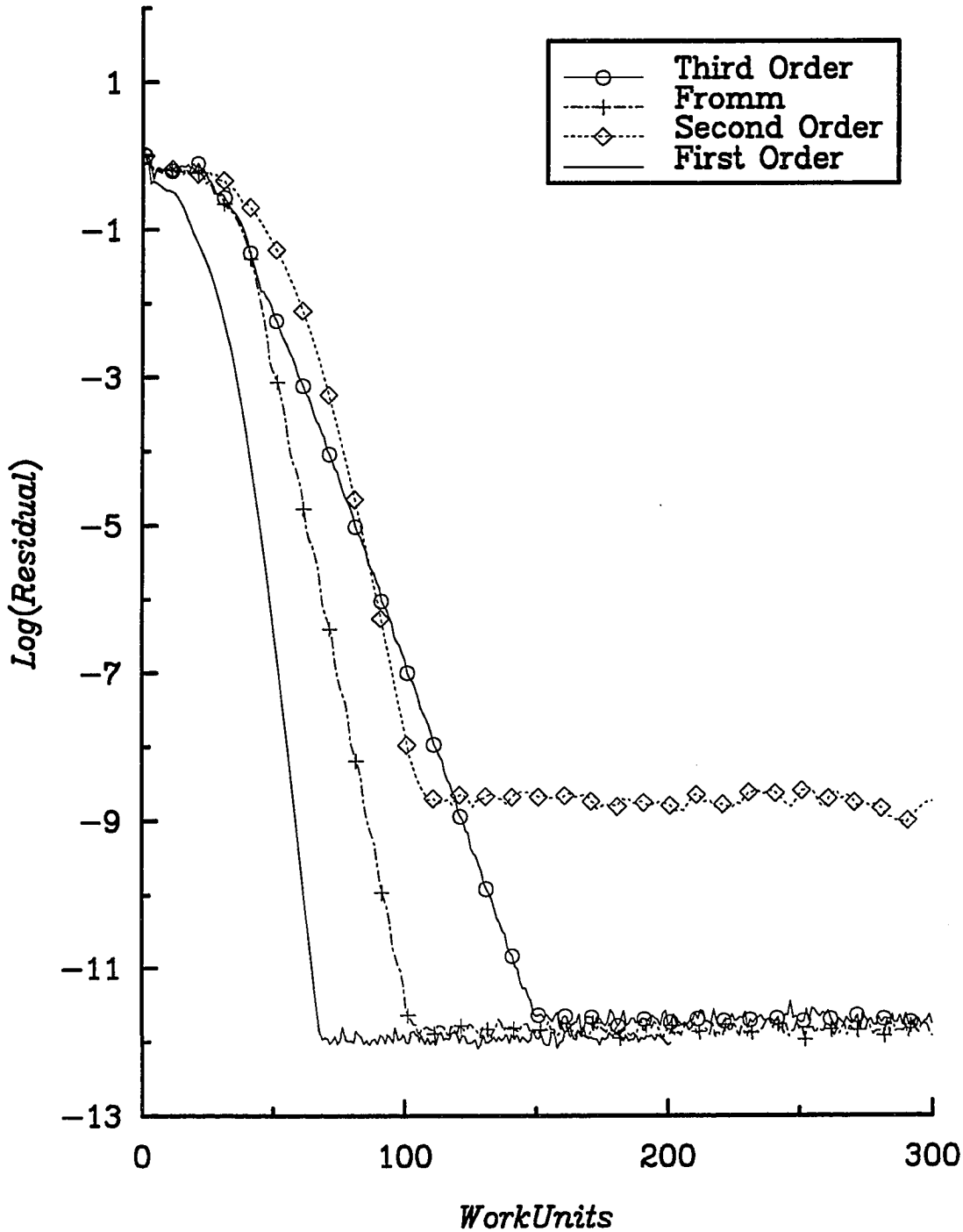


Figure 6.1.6 Convergence Histories for the Corner Flow Using Different Extrapolation Techniques.

6.1.1 Multiple Interface Requirements

To test the multi–block capabilities, the 65x65x65 corner flow grid was divided into eight blocks by bisecting each coordinate direction. A schematic of the block domains is shown in Fig. 6.1.1.7, where the shaded surfaces indicate the interfaces between the blocks. For computing the fluxes, the exchange of information across an interface needs

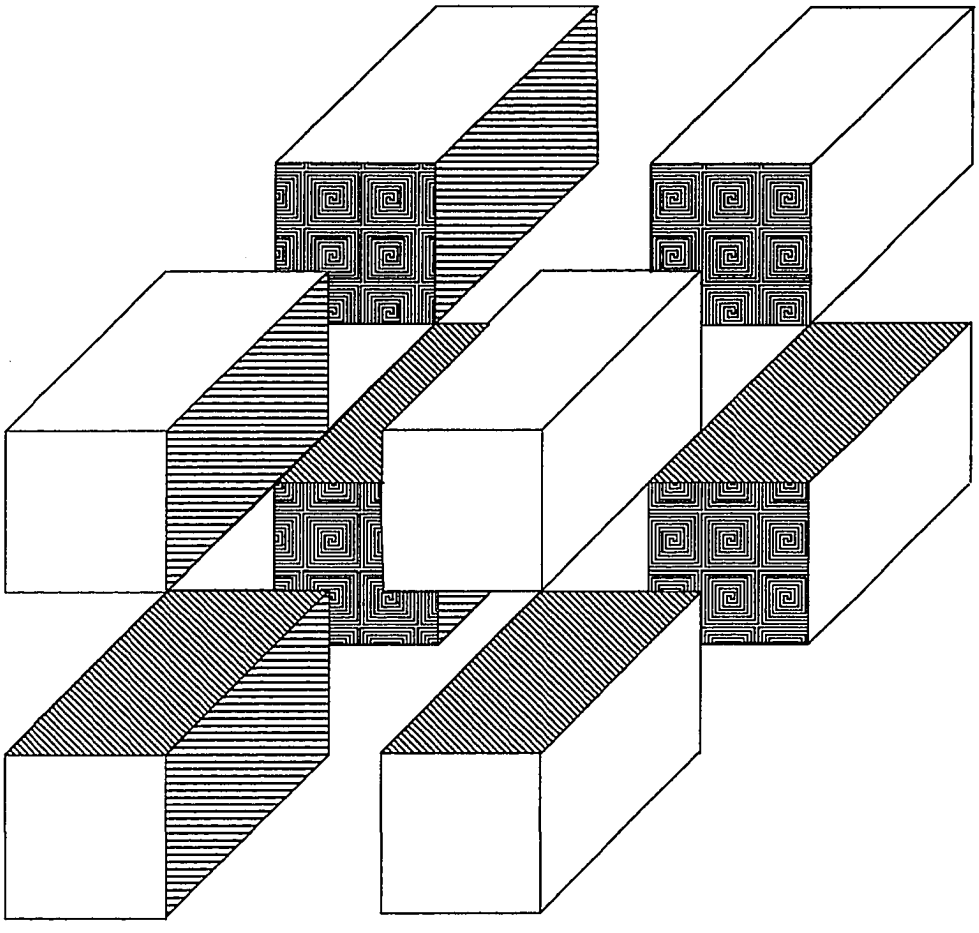


Figure 6.1.1.7 Eight-Block Configuration for Compression Corner.

to include only the ranges of the interior cells on that interface. However, when using multigrid techniques, consideration must be given to the restriction and prolongation operations. Whether or not there are interfaces has no influence on the present restriction process, but it can have an influence on the prolongation process. By dividing the

domain into blocks there are block boundaries where there would have normally been standard interior cells. This can affect the values produced by the trilinear interpolation prolongation process. The prolongation process is explained in Appendix B. The standard procedure for the present prolongation process at a boundary is to set the corrections to zero. That is because the boundary ghost cells are not solved directly; they are updated based on the flow values of the interior cells. This indicates that the correction at a block interface should be treated differently than that of a solid surface. Interface corrections should allow the prolongation process to compute the same results for a divided region as it would for an undivided region. To properly execute the prolongation process at an interface requires computing the correction for the ghost cell values and using them in the same manner as the correction values of the interior cells. The problem arises at the edges of the interface. It is these locations that require information from the ghost cells normal to the block interface, and also from the cells diagonal to it. This can be seen in Fig. 6.1.1.8, where a plane from four interfacing blocks is shown. The thick lines represent the interior regions of the blocks, and the thin lines represent the ghost cells. If the four blocks were joined as one, the prolongation for the upper left corner of block 1 would be influenced by information in the other three blocks; therefore the range of information transferred should fill the ghost cells of block 1 with the first two bottom rows of "x's" from block 2, the four lower right corner "o's" from block 3 and the first two right-hand-side columns of "z's" from block 4. Transferring these values then allows the prolongation for the upper left corner of block 1 to produce the same values as would be produced if the four blocks were combined as one. It is important to note that block 1 interfaces with only blocks 2 and 4; therefore, the range of information gathered across an interface needs to include more than just the values from the interior cells; it needs to obtain two extra cells from each end of the interface. A range which includes two extra

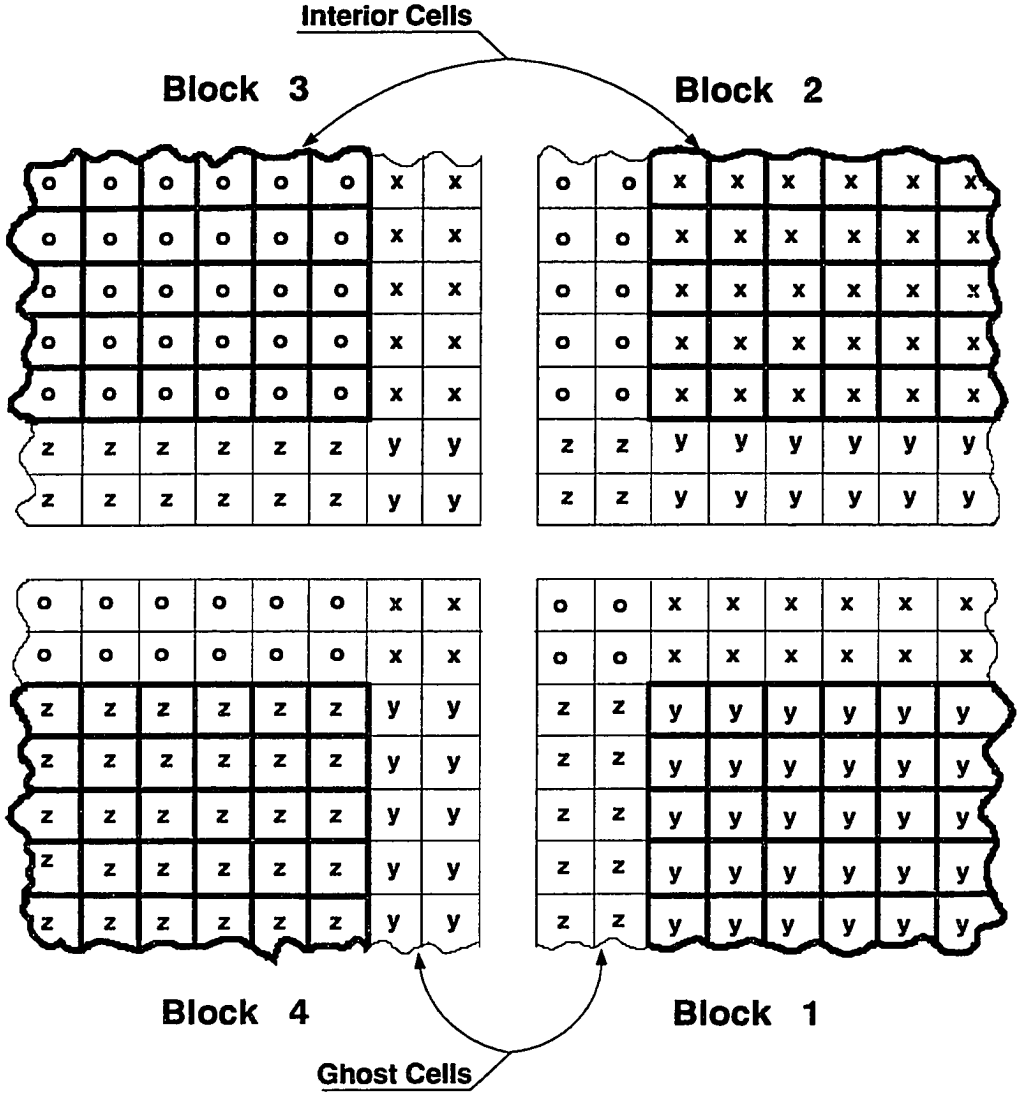


Figure 6.1.1.8 Four-Block Interface.

cells at each end of the interface provides the "o"s from block 3 to block 1, and allows the ghost cells of each block to have all of the information necessary to correctly complete the prolongation process. Since information is exchanged between at most two blocks at a time, two passes through the interface routine are required so that all of the interface ghost cells (beyond the interior index range of the interface) will be updated with the current interior cell values from the different blocks. This is necessary since the interface

ghost cells are updated in a sequential manner, generally starting with block 1 interfaces, then proceeding to block 2 and so on. As one can see from Fig. 6.1.1.8, the interface ghost cell values required by block 1 from block 3 will lag because block 1 gets its final block 3 information from block 4. Therefore, until block 4 has had its interfaces updated with current block 3 information, the information block 4 passes to block 1, on the first pass through the interface routine, will be from the previous time step. This problem is eliminated by executing the interface routine twice. Also, this method requires that block 1 only know the blocks that are normal to its interfaces, and not what blocks are diagonal to its corners, which in this test case is block 3. This simplifies the input information about the boundary conditions. Also, this interface approach satisfies the requirements of a block having three connected surfaces as interfaces, which occurs in this eight-block corner flow configuration. Notice that every block in this configuration interfaces with three other blocks, as can be ascertained from Fig. 6.1.1.7.

In Fig. 6.1.1.9 and Fig. 6.1.1.10 the Mach line contours from the single-block and eight-block calculations are shown, respectively, on the back and bottom walls, and the exit plane of the channel. For the eight-block configuration, the interfaces between the blocks are shown as thick solid lines on the back and bottom walls, and exit plane. As can be seen, the Mach lines pass through the interfaces without deviation, proving that the interfaces are not introducing any error. Similar results are displayed in Figs. 6.1.1.11 and 6.1.1.12, where the pressure contours are shown for both the single- and eight-block configurations, respectively. Again, there is no difference in the contours between the single-block and the eight-block calculations. As shown in Fig. 6.1.1.13, the convergence histories for the single-block and the eight-block calculations are the same.

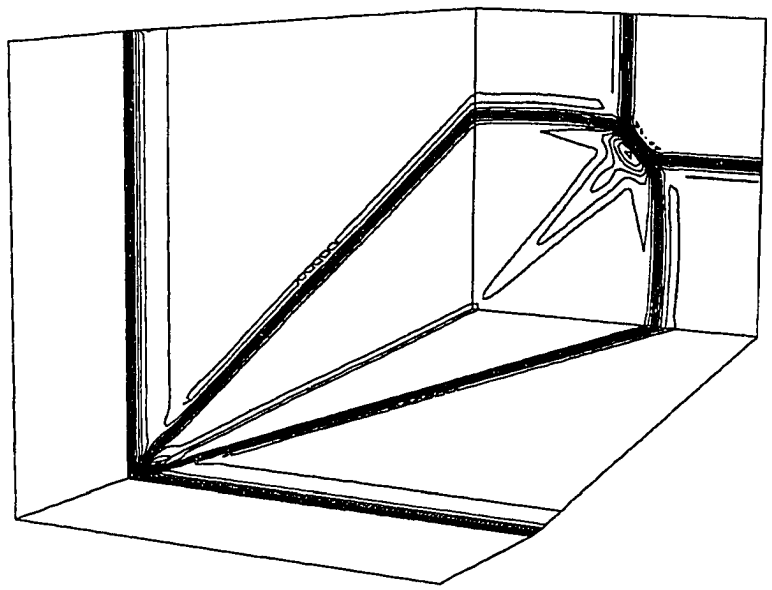


Figure 6.1.1.9 Mach Line Contours for the
Single-Block Calculation, $M_{inlet} = 3.0$ and $\alpha = 9.5°$.

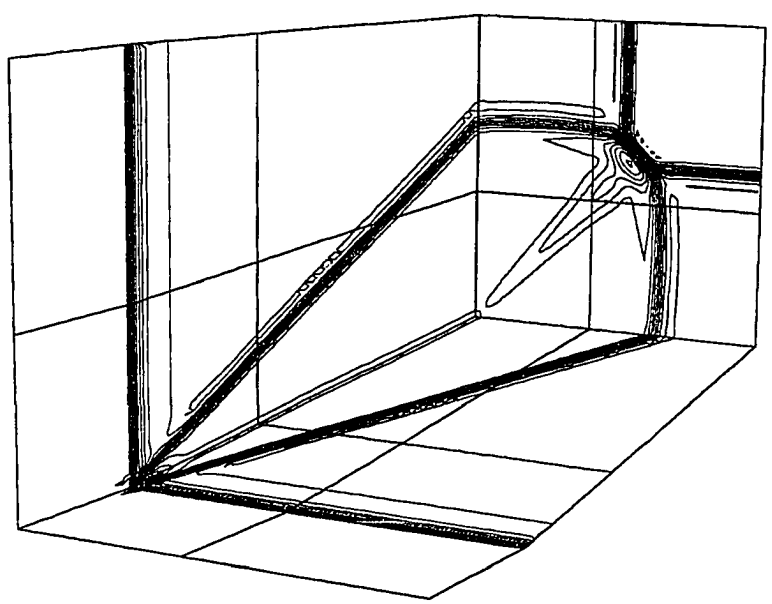


Figure 6.1.1.10 Mach Line Contours for the
Eight-Block Calculation, $M_{inlet} = 3.0$ and $\alpha = 9.5°$.

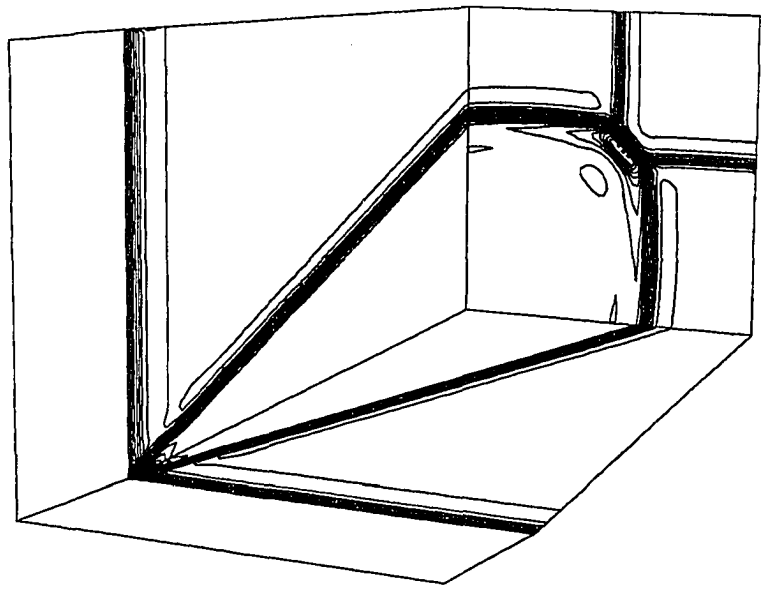


Figure 6.1.1.11 Pressure Line Contours for the Single-Block Calculation, $M_{inlet} = 3.0$ and $\alpha = 9.5°$.

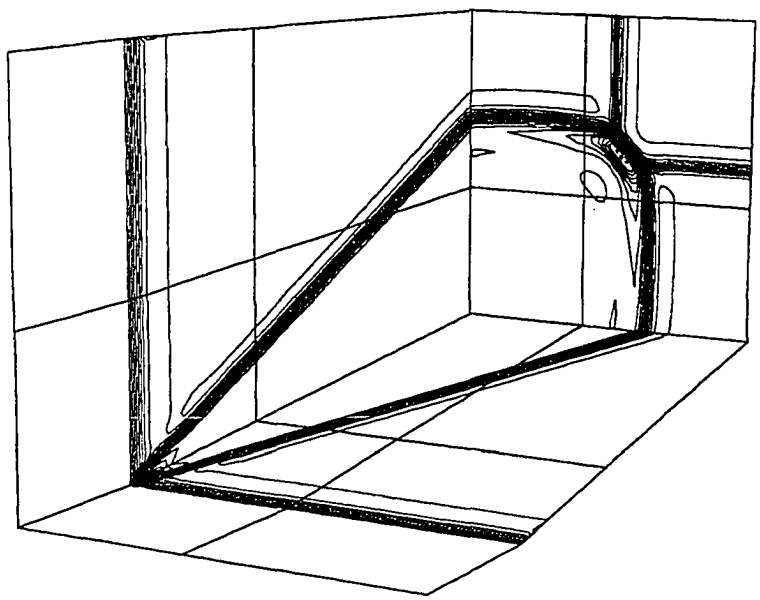


Figure 6.1.1.12 Pressure Line Contours for the Eight-Block Calculation, $M_{inlet} = 3.0$ and $\alpha = 9.5°$.

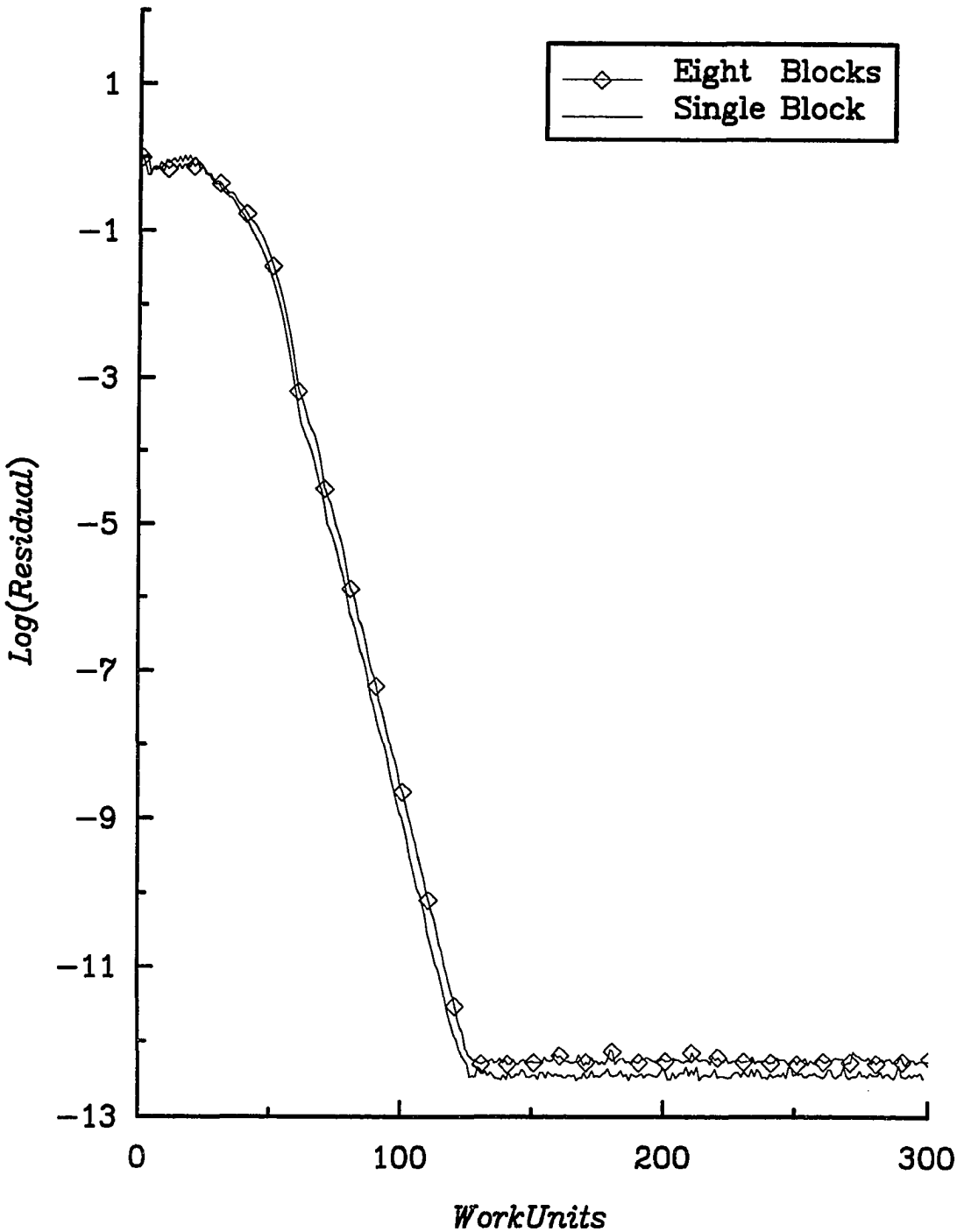


Figure 6.1.1.13 Comparison of Convergence Histories Between the Single-Block and Eight-Block Calculations for the Corner Flow.

6.2 Pseudo Two-Dimensional Jet Exhaust Plume

The second case studied was an inviscid pseudo two-dimensional jet exhaust plume. This case tested the multiple boundary input format of the computer code as well as provided a basis for comparing the two different upwind solvers, in terms of their ability to predict both slip lines and shocks. It also served as a forum for the examination of the different extrapolation techniques of pure second order upwind, Fromm's scheme, and κ = 1/3. This case is considered a pseudo two-dimensional flow because the cross flow is negligible.

The inflow surface of the computational domain was divided into two sections. One section had as its inflow conditions the exhaust from a jet of height "h", where M_{jet} = 1.5. The second section had a free-stream inflow of M_{∞} = 2.5. The static pressure and static temperature ratios between the two flows were p_{jet}/p_{∞} = 3.5 and T_{jet}/T_{∞} 3.0, respecitvely. A schematic of this flow field is provided in Fig. 6.2.14. This figure identifies the slip line, expansion fan, and shock that is present in this type of flow. The "wall" in this figure was actually treated as a symmetry plane. The grid generated for the pseudo two-dimensional plume flow is shown in Fig. 6.2.15. This case was first tested incorporating Roe's flux-differencing using: (1) pure second order upwind, (2) Fromm's scheme, and (3) κ = 1/3 extrapolation of the primitive variables, without the use of a flux limiter. These results were compared with those from a validated shock fitting code developed by Salas [101]. This case was evaluated using FMG, variable coefficient residual smoothing and modified Runge-Kutta time integration. At the first station, x/h $\approx$ 1.0, shown in Fig. 6.2.16a, the second order extrapolation produced a larger undershoot at the slip line than the other higher order methods. It is comparable everywhere else, produces a smaller overshoot at the shock, and has no oscillations. The κ = 1/3 continually produces the largest overshoot at the shock and is accompanied by

a small oscillation (Figs. 6.2.16d - 6.2.16e), up until station x/h $\approx$ 5.0 (Fig. 6.2.16e). At this station and station x/h = 10., shown in Fig. 6.2.17f, the overshoots at the shock are greatly diminished for all extrapolation methods. The two vertical columns along the right side of the figures indicate the packing density of points in the y/h direction. The shock fitting computer code generates its own grid, and its packing density is shown as the first column starting on the left. The Roe scheme employed a different grid, which had a vertical packing density shown as the right column. The columns are shown in the order of the flow solver legend, with the far left column representing the grid spacing for the last flow solver in the legend. Based on these results, and the desire not to employ a limiter (because of the convergence difficulties they produce), the pure second order upwind extrapolation method was chosen for the next case. Convergence history comparisons between the different extrapolation techniques for the pseudo two-dimensional jet exhaust plume are shown in Fig. 6.2.17.

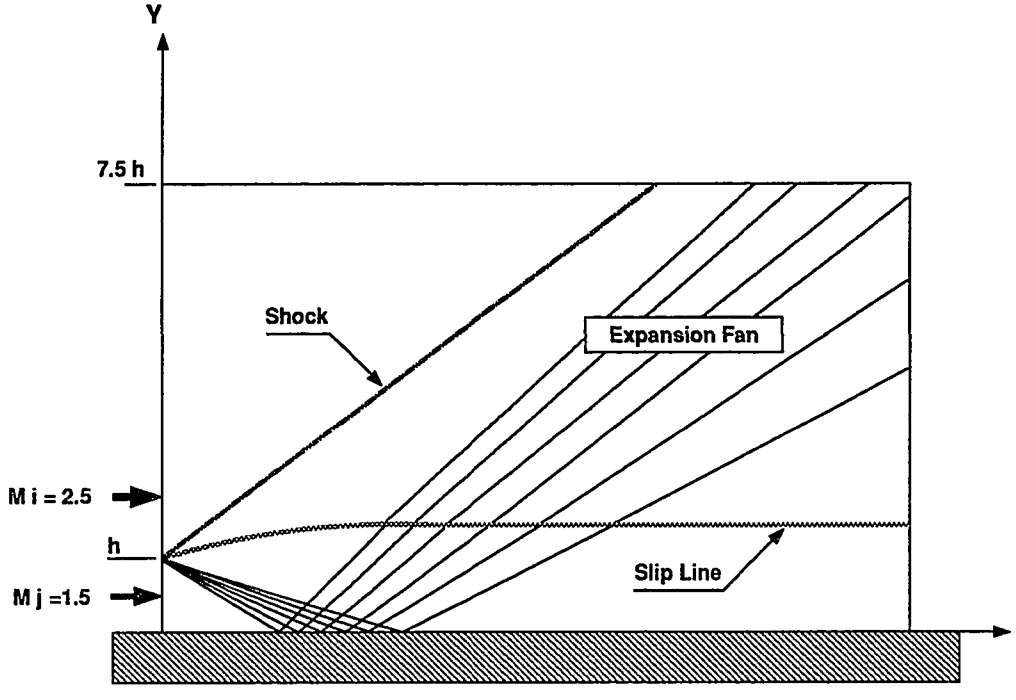


Figure 6.2.14 Schematic of Pseudo Two-Dimensional Jet Exhaust Plume.

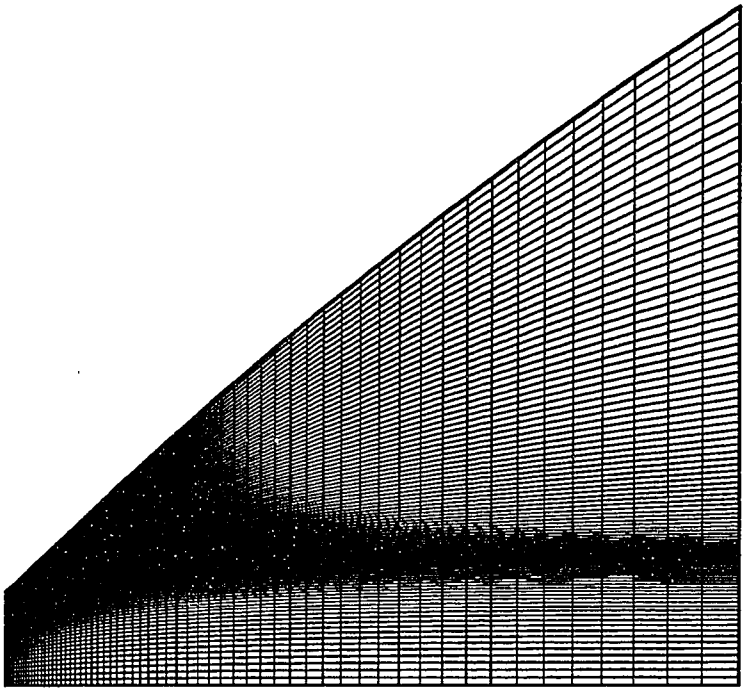
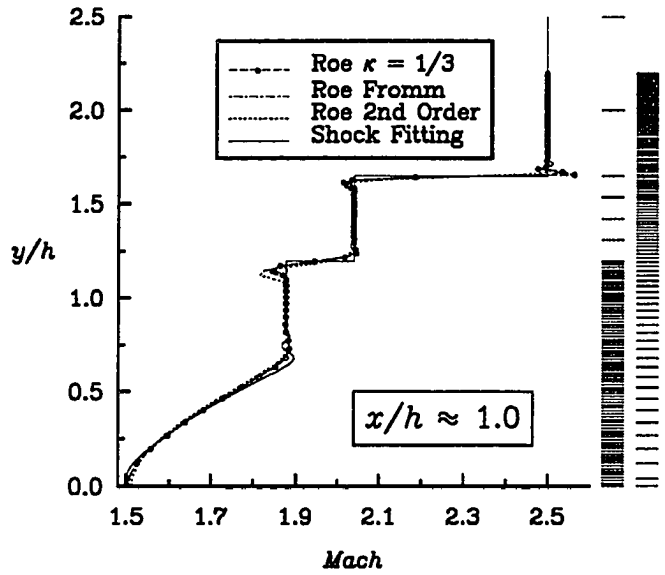


Figure 6.2.15 Grid Used for the Present Pseudo Two-Dimensional Jet Exhaust Plume Calculations.

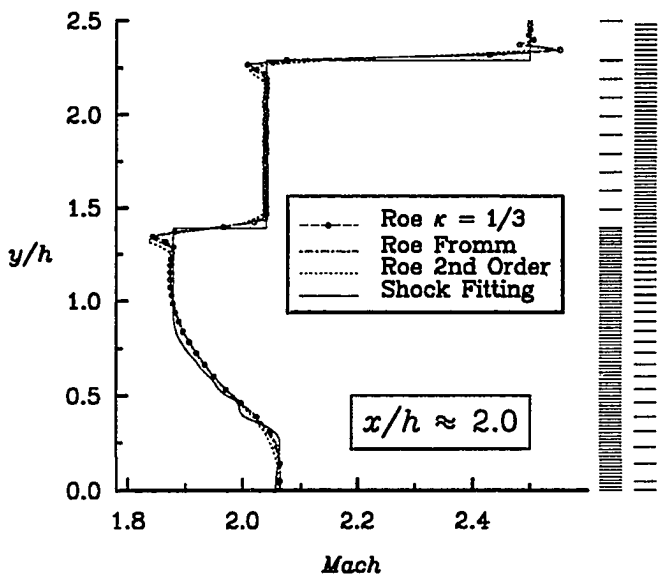
The next comparison for this case was between Roe's flux-difference splitting and van Leer's flux-vector splitting. Figures 6.2.18 and 6.2.19 show the Mach line contours for Roe's flux-difference splitting method, and van Leer's flux-vector splitting method, respectively. In Figs. 6.2.20 and 6.2.21 are the pressure contours for Roe's scheme and van Leer's scheme, respectively. Both methods employed the same Runge-Kutta time integration scheme, using pure second order upwind extrapolation, without flux limiters, and they employed variable-coefficient residual-smoothing. Conservative variables were extrapolated for van Leer's scheme, because the use of primitive variables is known to cause oscillations. Primitive variables were extrapolated for Roe's scheme. The Mach line contours appear to provide the same solution, with van Leer's method spreading the slip line a little wider than Roe's method. This is to be expected, because flux-vector splitting does not have a mechanism for resolving the slip line, which flux-difference

splitting does because it is based on an approximate Riemann solver. Overall the differences in the two sets of contours are negligible. Further comparisons of these results are shown in Figs. 6.2.22a - 6.2.22f, with the validated shock fitting code of Salas [101]. The first figure (Fig. 6.2.22a) is for station x/h $\approx$ 1.0, the only disparity between the flux-vector splitting and the flux-difference splitting methods comes just after the constant Mach number region and just before the slip line. In that region, the flux-difference splitting method has a larger undershoot. Both upwind methods have the same magnitude of undershoot and overshoot at the shock, and neither had an overshoot at the slip line. The shock fitting method tends to have oscillations in the transition region between the expansion fan and the constant Mach number region, but provides crisp results throughout the remainder of the flow for this x/h location. For x/h $\approx$ 2.0, shown in Fig. 6.2.22b, one can again see that the only disparity between the van Leer and Roe methods is the undershoot by Roe's scheme at the slip line. Starting at this x/h location, the overshoot at the shock has essentially disappeared. This is due to the shock becoming aligned with the grid. The next location, x/h $\approx$ 2.5 (Fig. 6.2.22c), shows that the undershoot from Roe's scheme at the slip line has diminished. At the x/h $\approx$ 3.0 location, shown in Fig. 6.2.22d, the Roe and van Leer methods are providing the same magnitude of undershoot at the slip line, but at x/h $\approx$ 5.0 (Fig. 6.2.22e) and x/h = 10.0 (Fig. 6.2.22f) the two upwind methods are providing essentially the same error in the overshoots and undershoots; however, Roe's scheme again gives a slightly larger undershoot at the slip line.

Overall the agreement between the two upwind methods was very good and their results compare well with those of the shock fitting computer code of Salas [101]. Both Roe's and van Leer's upwind methods predict overshoots and undershoots at slip lines and shocks on a grid that is not properly aligned with the flow physics.

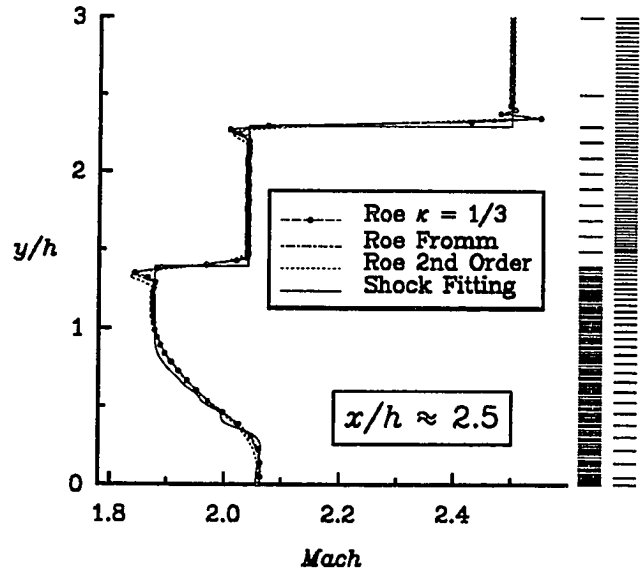


(a)

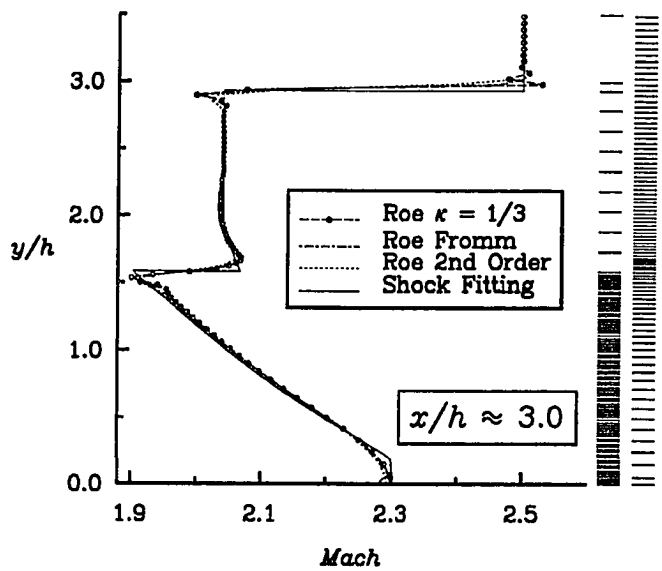


(b)

Figure 6.2.16 Comparisons Between Different Extrapolations of Roe's Scheme and a Shock Fitting Code for a Pseudo Two-Dimensional Exhaust Plume. (Continued . . .)

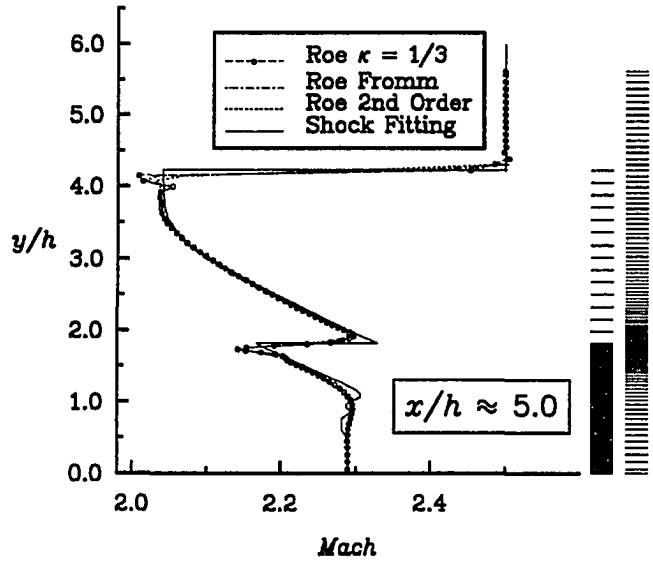


(c)

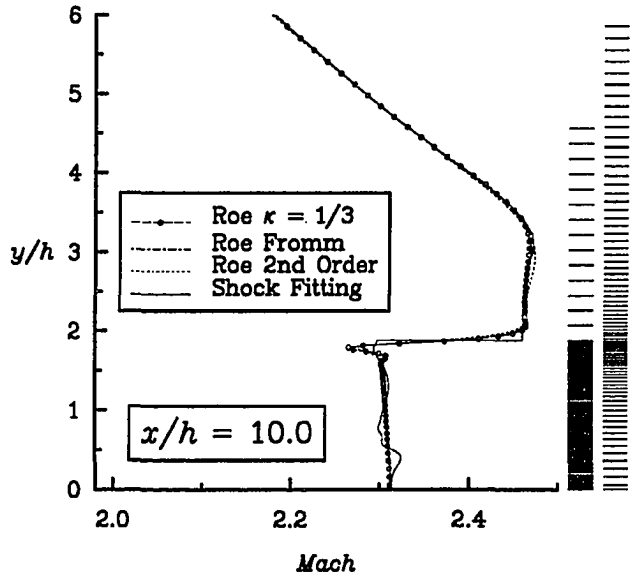


(d)

Figure 6.2.16 Comparisons Between Different Extrapolations of Roe's Scheme and a Shock Fitting Code for a Pseudo Two-Dimensional Exhaust Plume. (Continued . . .)



(e)



(f)

Figure 6.2.16 Comparisons Between Different Extrapolations of Roe's Scheme and a Shock Fitting Code for a Pseudo Two-Dimensional Exhaust Plume.

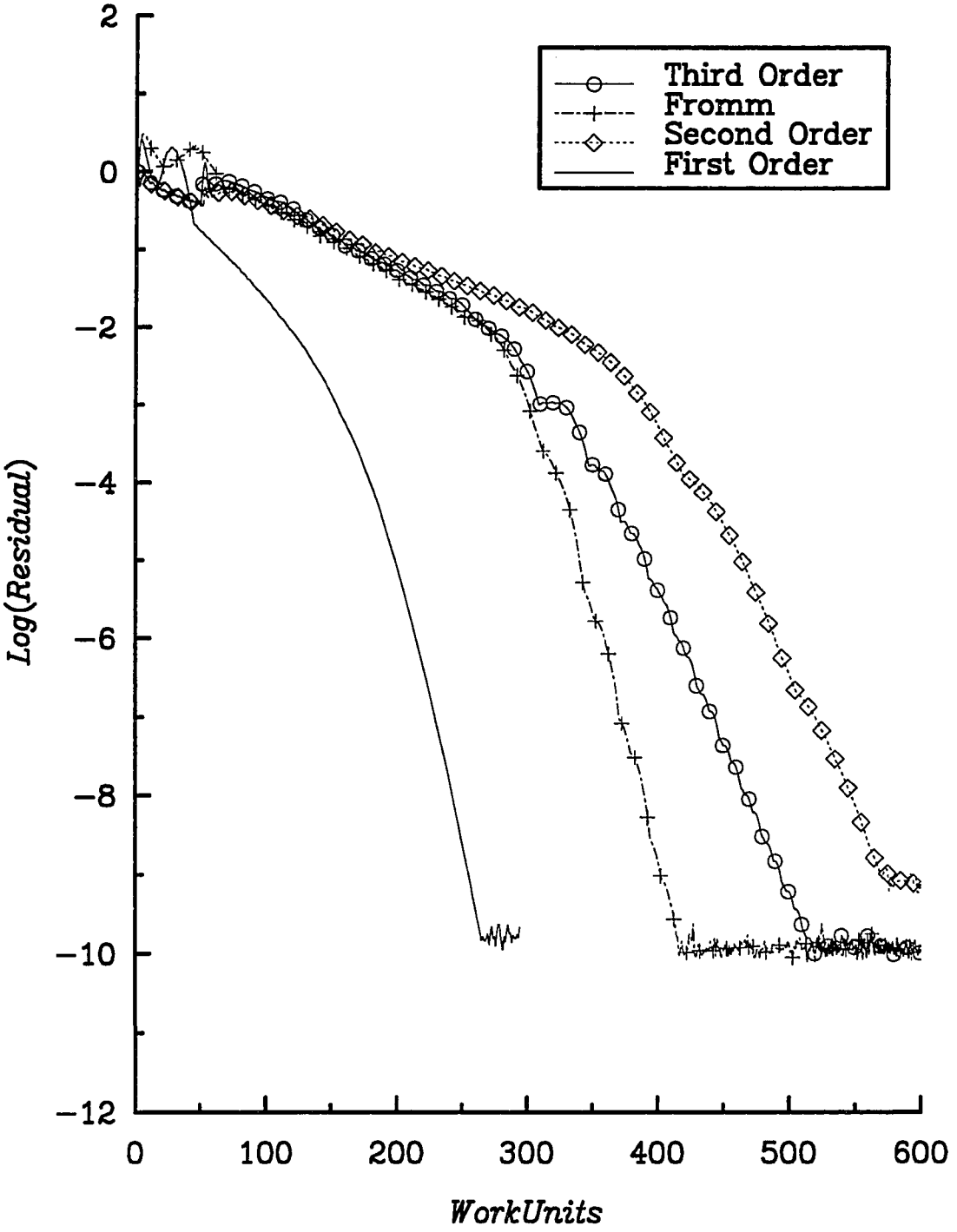


Figure 6.2.17 Convergence History Comparisons Between Different Extrapolation Techniques for the Pseudo Two-Dimensional Jet Exhaust Plume.

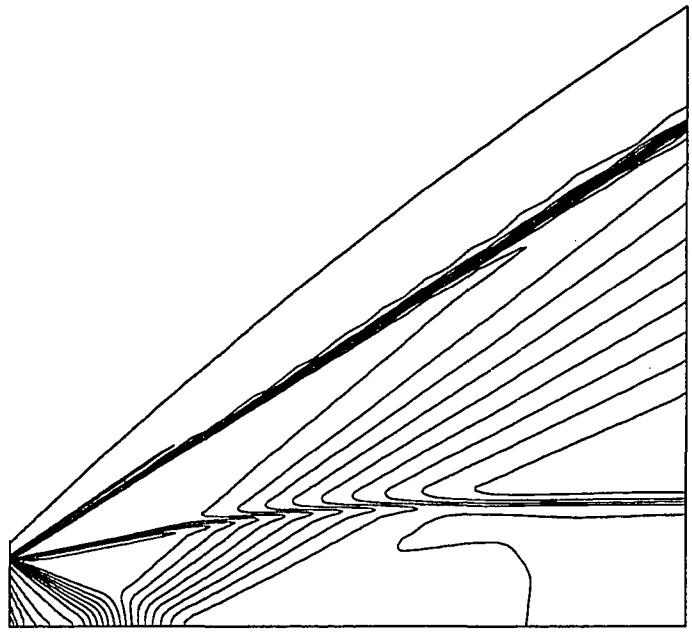


Figure 6.2.18 Mach Line Contours for Roe's Scheme,
$M_\infty = 2.5$, $M_{jet} = 1.5$, $P_{jet}/P_\infty = 3.5$, and $T_{jet}/T_\infty = 3.0$.

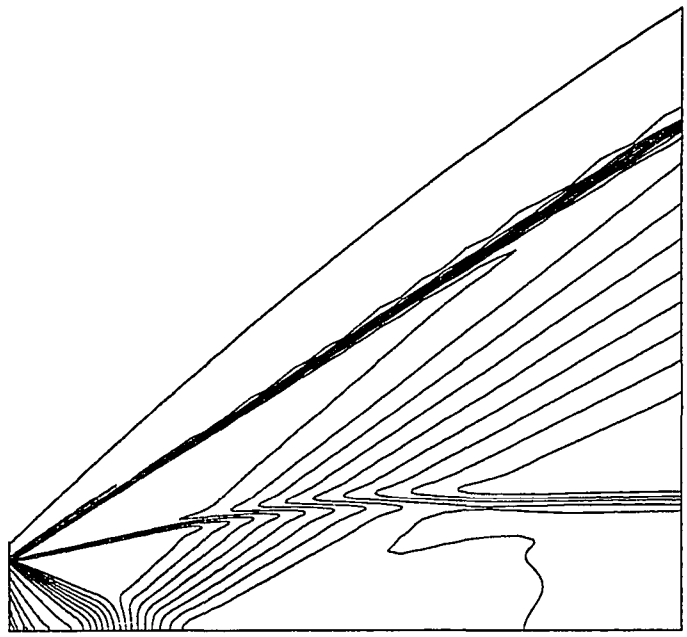


Figure 6.2.19 Mach Line Contours for van Leer's Scheme,
$M_\infty = 2.5$, $M_{jet} = 1.5$, $P_{jet}/P_\infty = 3.5$, and $T_{jet}/T_\infty = 3.0$.

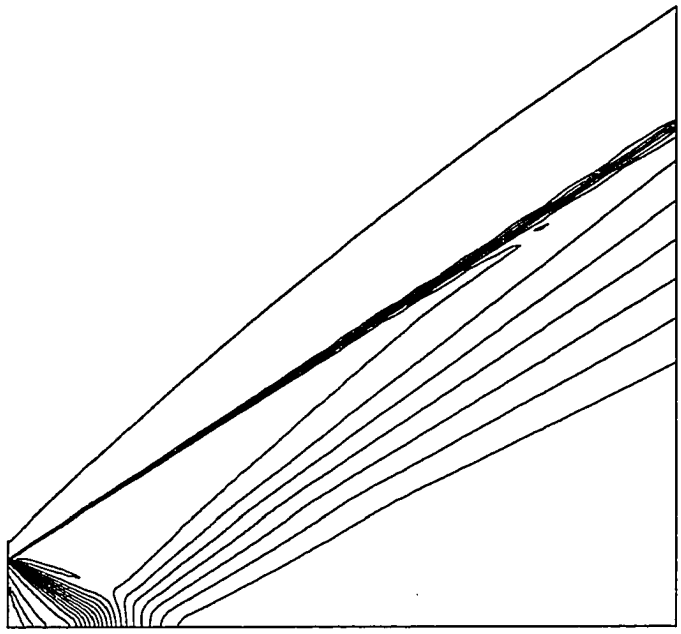


Figure 6.2.20 Pressure Line Contours for Roe's Scheme,
$M_\infty = 2.5$, $M_{jet} = 1.5$, $P_{jet}/P_\infty = 3.5$, and $T_{jet}/T_\infty = 3.0$.

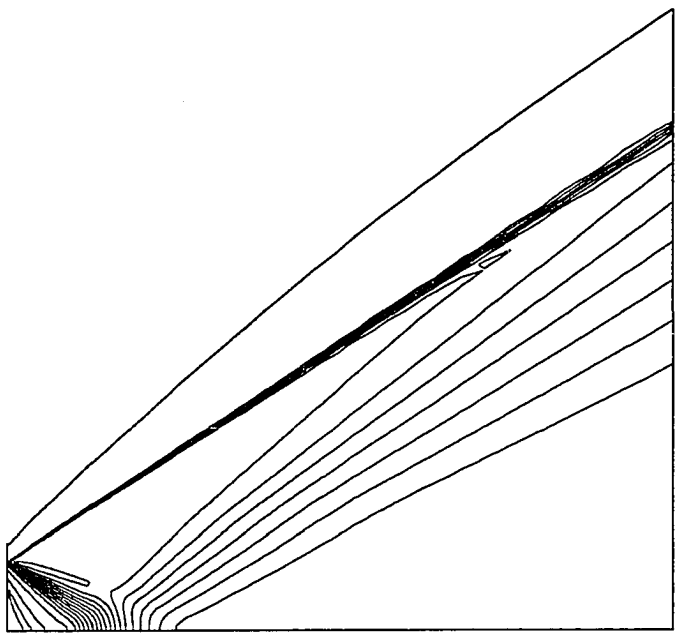
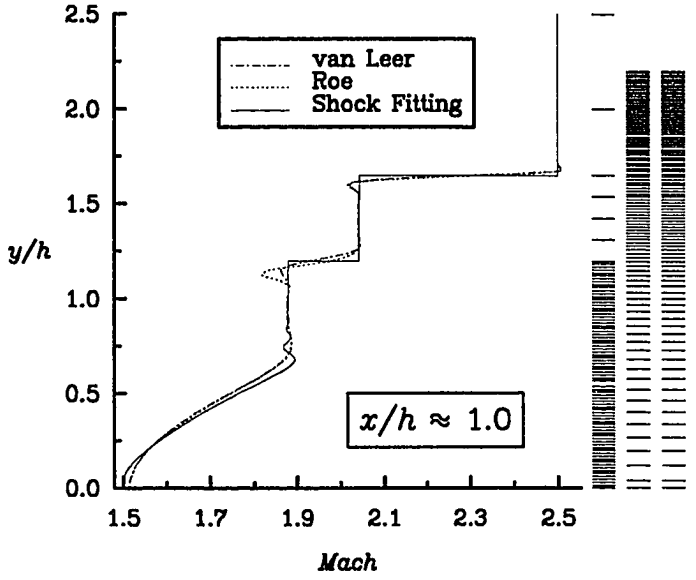
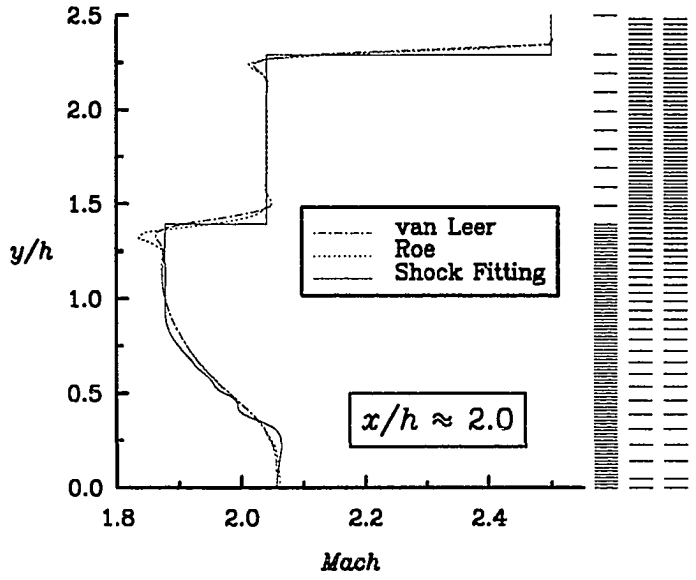


Figure 6.2.21 Pressure Line Contours for van Leer's Scheme,
$M_\infty = 2.5$, $M_{jet} = 1.5$, $P_{jet}/P_\infty = 3.5$, and $T_{jet}/T_\infty = 3.0$.

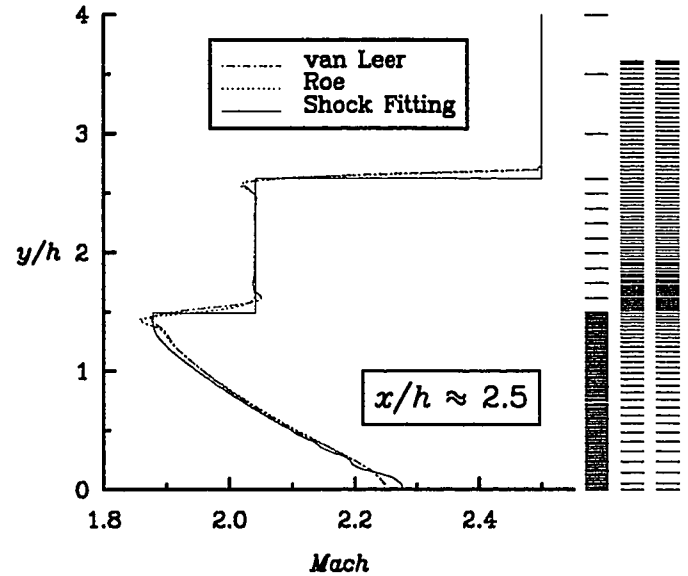


(a)

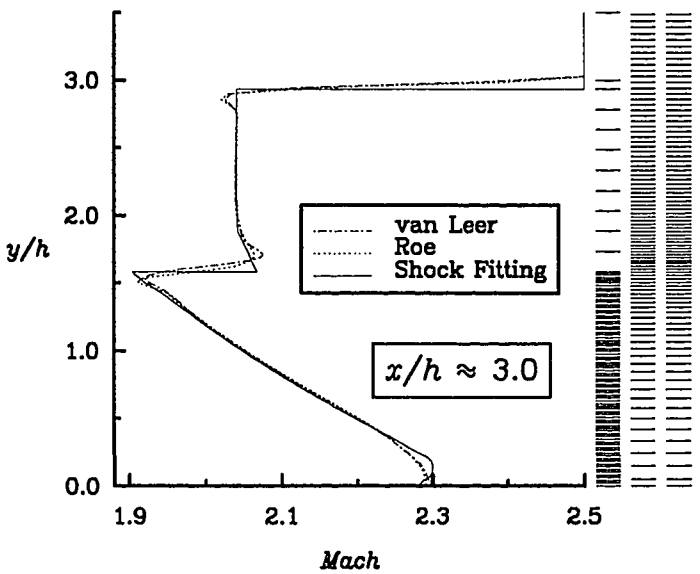


(b)

Figure 6.2.22 Comparison Between Roe's Flux-Differencing and van Leer's Flux-Vector Splitting for a Pseudo Two-Dimensional Exhaust Plume. (Continued . . .)

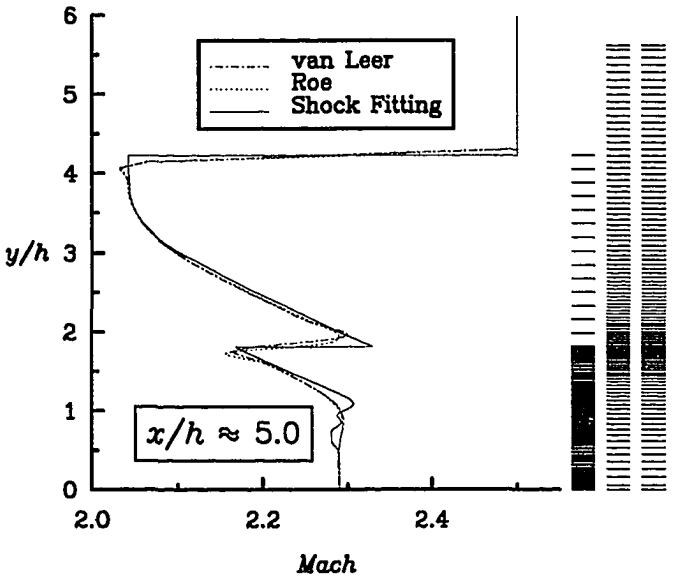


(c)

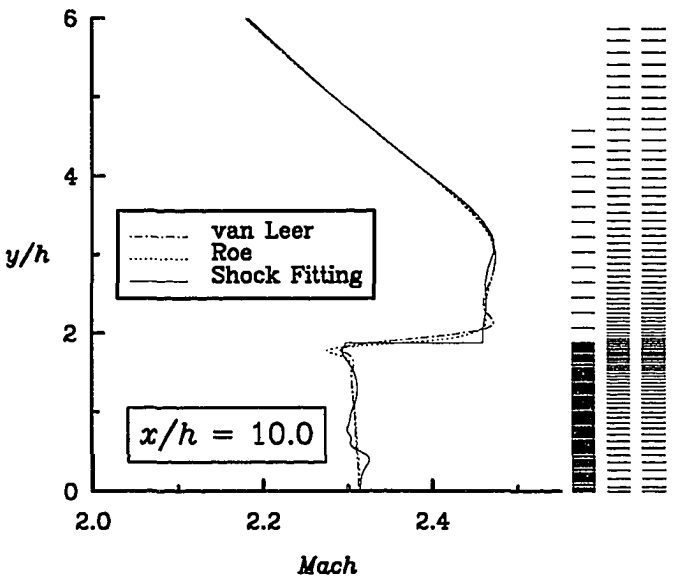


(d)

Figure 6.2.22 Comparison Between Roe's Flux-Differencing and van Leer's Flux-Vector Splitting for a Pseudo Two-Dimensional Exhaust Plume. (Continued . . .)



(e)



(f)

Figure 6.2.22 Comparison Between Roe's Flux-Differencing and van Leer's Flux-Vector Splitting for a Pseudo Two-Dimensional Exhaust Plume.

6.3 Laminar and Turbulent Flows Over a Flat Plate

The third test case was laminar and turbulent flow over a flat plate. The purpose of this case was to demonstrate that the viscous terms had been implemented correctly and that the Baldwin-Lomax algebraic turbulence model had been employed correctly for attached turbulent flows.

6.3.1 Laminar Flow

For laminar flow over a flat plate, the Mach and Reynolds numbers were taken to be $M_\infty = 0.5$ and $R = 1,000/(\text{unit length})$, respectively. The normalized height of the first cell normal to the plate was $1\text{x}10^{-4}$ units, with one unit being the reference length of the plate. Good agreement was obtained between the present results and the Blasius solution. Comparisons with the skin friction coefficient and velocity profile are shown in Fig. 6.3.2.23. The grid was 65x65x5, with 64 cells in the streamwise direction, 64 cells normal to the plate, and 4 cells in the span-wise direction, to allow for three levels of multigrid. Computations were performed incorporating FMG, variable coefficient residual smoothing, and modified Runge-Kutta time integration. There was a minimum of 34 grid cells in the fully developed boundary layer (where fully developed is defined as having self similar velocity profiles). Convergence histories for the laminar flat plate flow comparing different acceleration techniques are shown in Fig. 6.3.2.24. As one can see, multigrid acceleration coupled with residual smoothing provided the best convergence rate.

6.3.2 Turbulent Flow

The turbulent flat plate flow conditions were $M_\infty = 0.5$, $R = 1,000,000/(\text{unit length})$, and the normalized height of the first cell normal to the plate was $1\text{x}10^{-5}$ units, with

one unit being the reference length of the plate. Good agreement was obtained for the skin friction, velocity profile, and law of the wall plot, as shown in Fig. 6.3.2.25. The grid was 65x95x5, with 64 cells in the streamwise direction, 94 cells normal to the plate, and 4 cells in the span-wise direction, to allow for three levels of multigrid. Computations were performed incorporating FMG, variable coefficient residual smoothing, and modified Runge-Kutta time integration. A minimum of 42 grid cells were in the fully developed boundary layer (fully developed meaning self similar velocity profile).

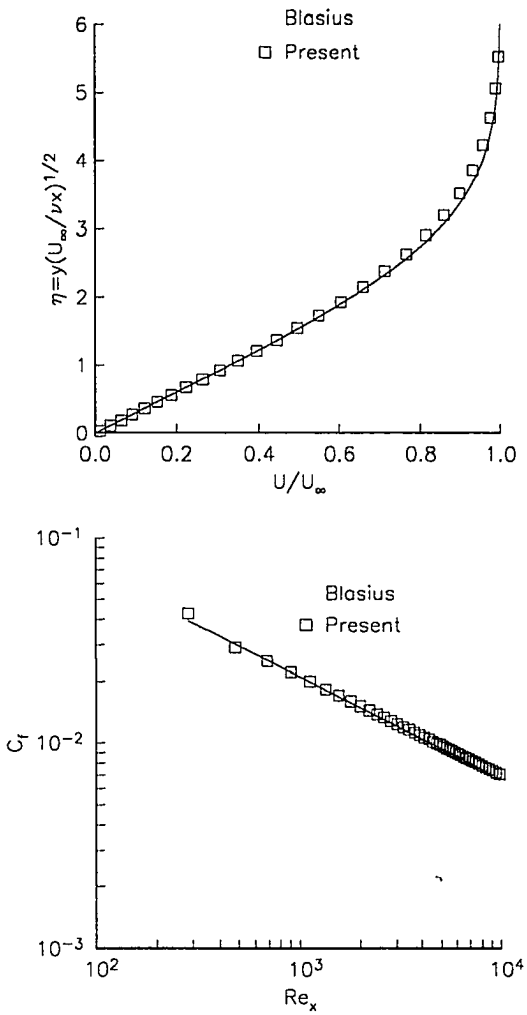


Figure 6.3.2.23 Laminar Flat Plate Comparisons with Analytical Calculations

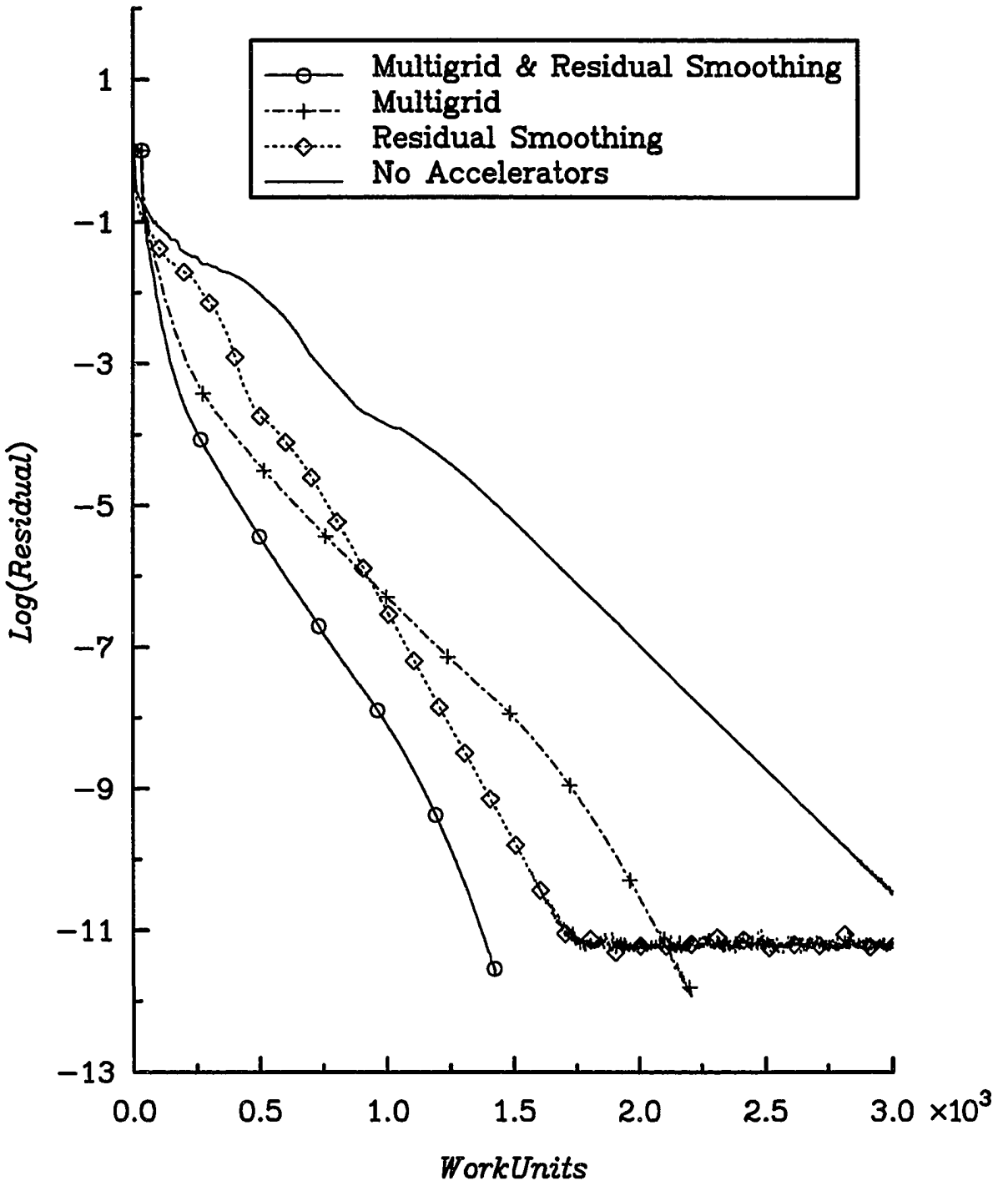
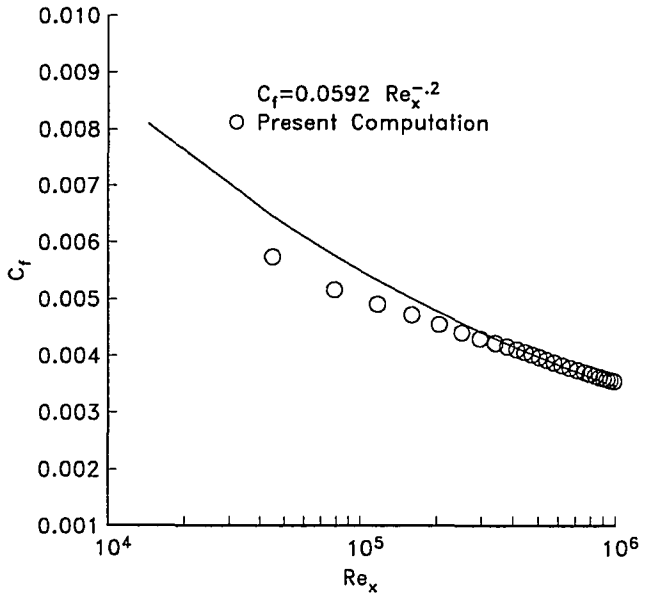


Figure 6.3.2.24 Convergence Histories for the Laminar Flat Plate Flow Comparing Different Acceleration Techniques.



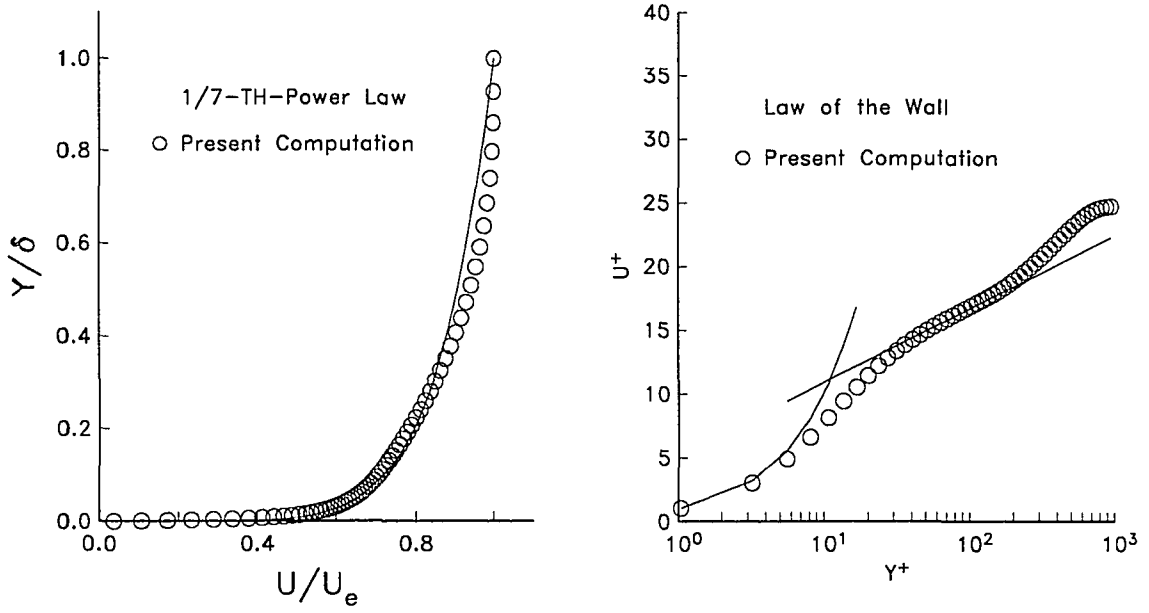


Figure 6.3.2.25 Turbulent Flat Plate Comparisons with Analytical Calculations

6.4 Turbulent Flow Over an ONERA M6 Wing

The fourth case studied was turbulent flow over the ONERA M6 wing [102] with 193x49x33 grid points in a C-O mesh topology (schematic shown in Fig. 6.4.26). This case was chosen for two reasons. First, this case has true three-dimensional turbulent flow. Second, this flow configuration places special requirements on the block interface routine, which will be explained in the next section. The first test case was $M_\infty = 0.699$, $\alpha = 3.06°$, and $R = 11.7\text{x}10^6/(\text{unit length})$. The wing was normalized to a semi-span of a unit length. This was a subcritical case shown to valid the computer code, which is in good agreement with the experimental data [102], as indicated by the C_p plots in Figs. 6.27a - 6.27f, where η is the dimension distance from the wing root.

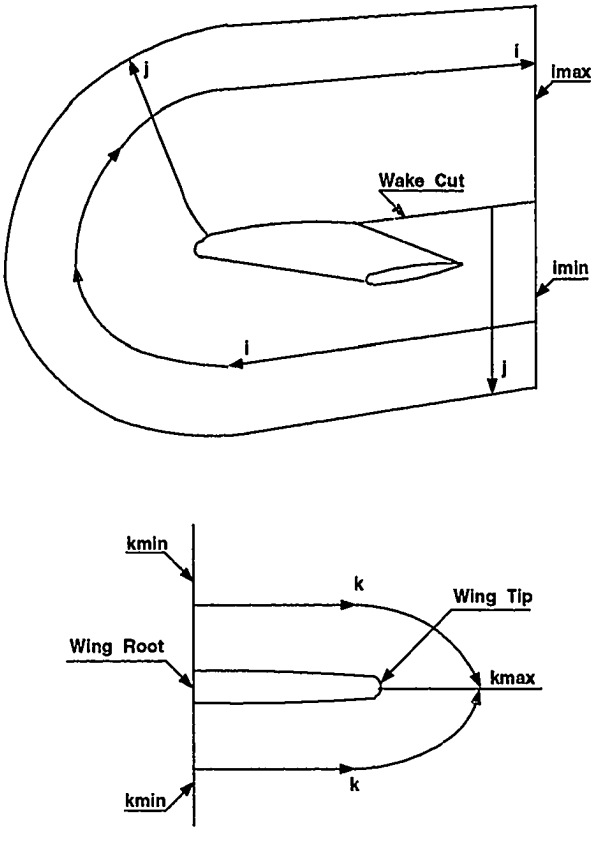
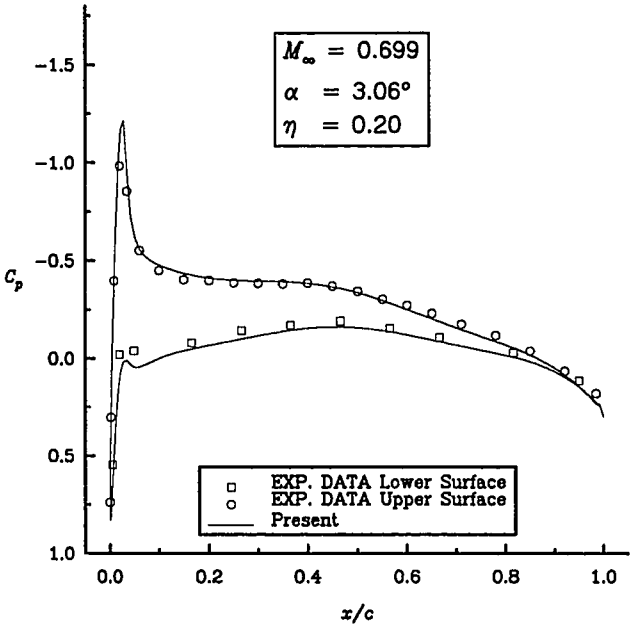
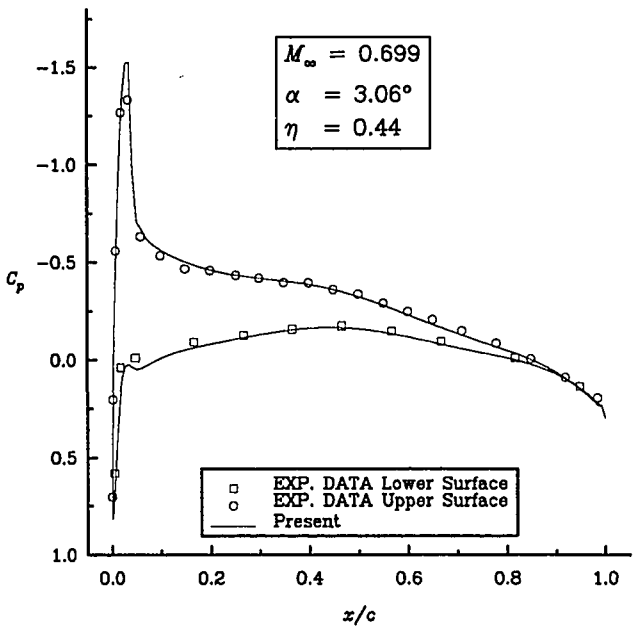


Figure 6.4.26 Schematic of C-O Mesh Topology for ONERA M6 Wing.

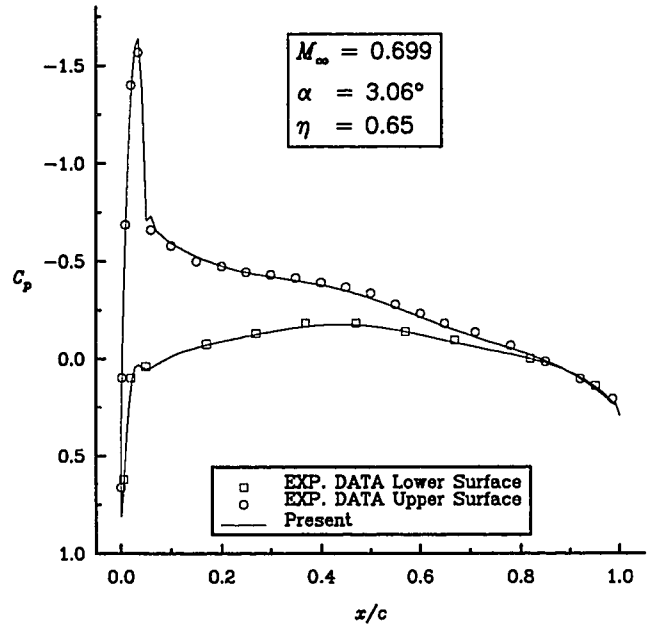


(a)

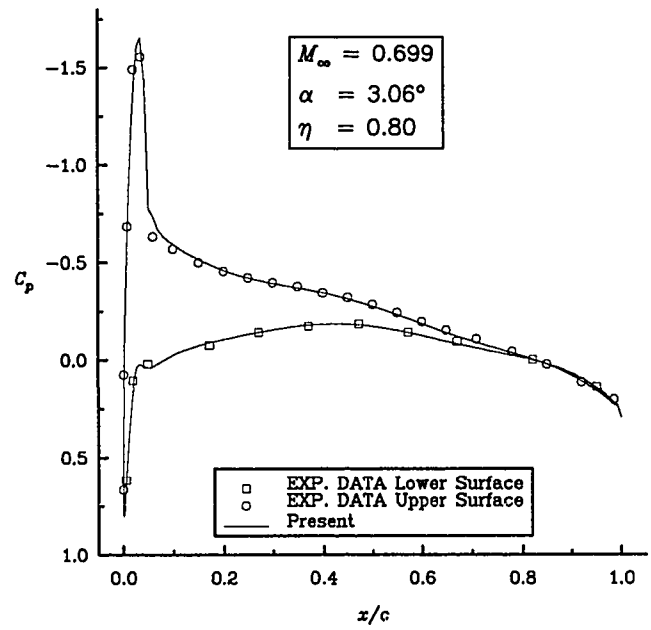


(b)

Figure 6.27 Comparison of Numerical Results with Experimental Data for ONERA M6 Wing. (Continued . . .)

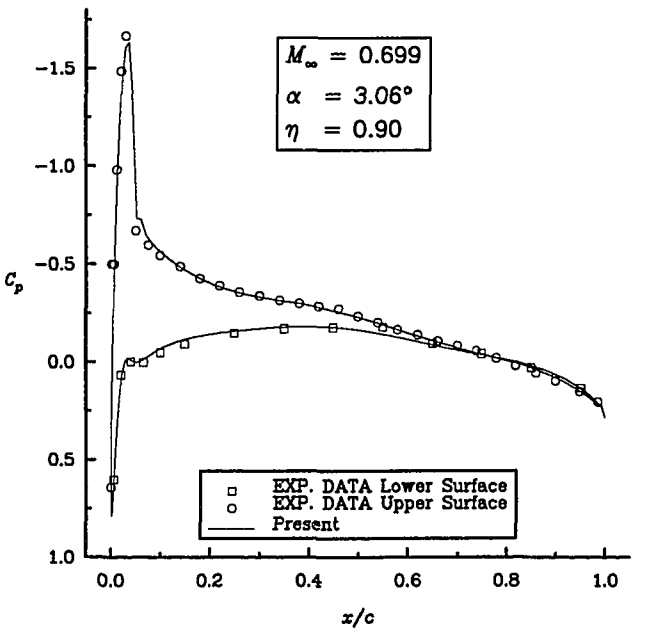


(c)

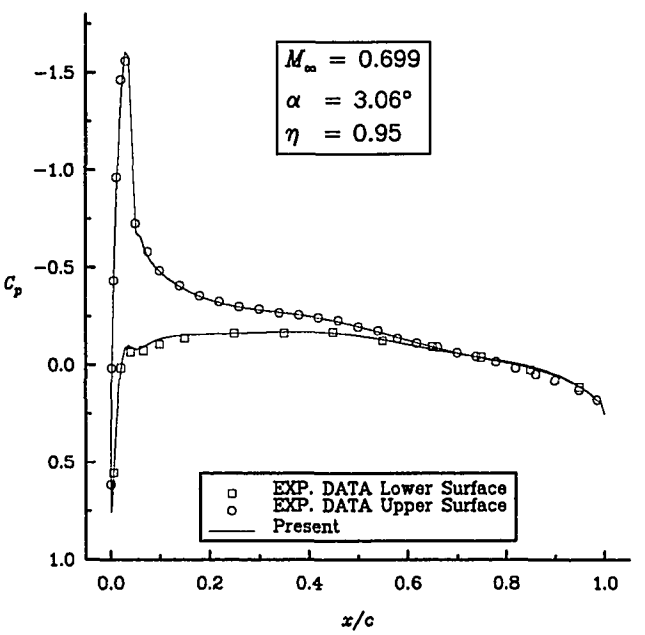


(d)

Figure 6.27 Comparison of Numerical Results with Experimental Data for ONERA M6 Wing. (Continued . . .)



(e)

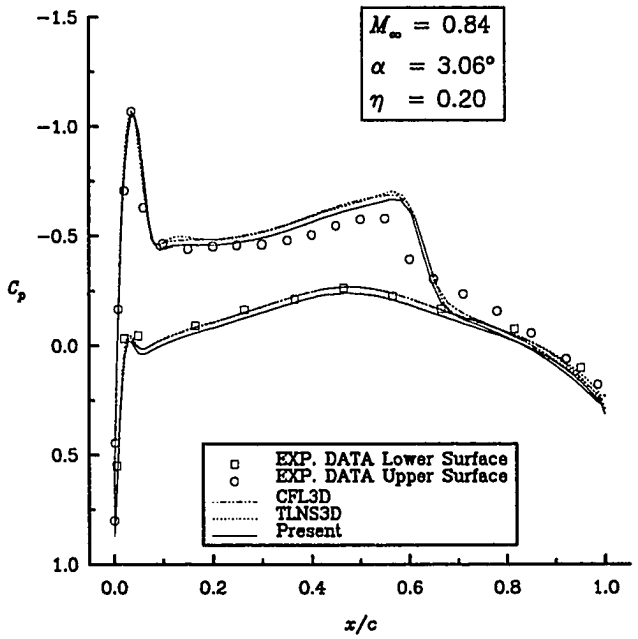


(f)

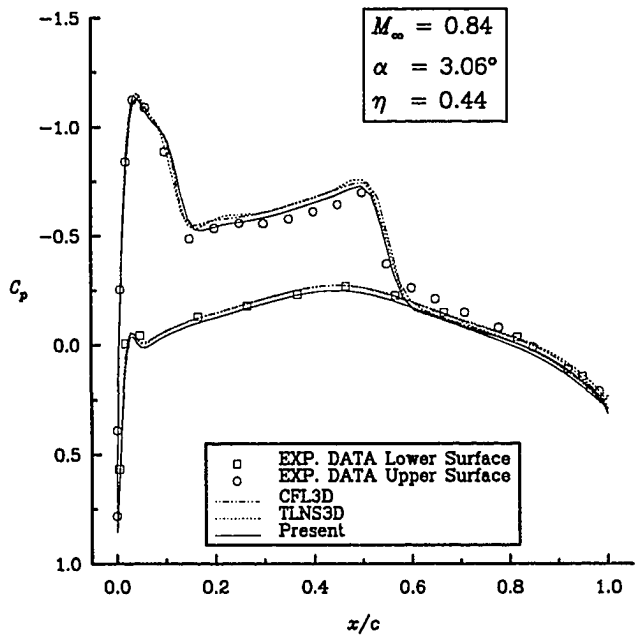
Figure 6.27 Comparison of Numerical Results with Experimental Data for ONERA M6 Wing.

The second test case has a lambda shock structure on the upper surface, and is much more demanding numerically, with $M_\infty = 0.84$, $\alpha = 3.06°$, and $R = 11.7\text{x}10^6/(\text{unit length})$. Since the Reynolds number is the same as the previous case, the same grid was used. Note that a significantly higher Reynolds number would require smaller vertical spacing between the points in the boundary-layer regions. Comparisons of C_p plots with experimental data and other numerical results are shown in Figs. 6.4.28a - 6.4.28f. TLNS3D is a Jameson type, thin-layer, central difference, modified Runge-Kutta computer code, developed by Vatsa and Wedan [86], which employed the Baldwin-Lomax algebraic turbulence model [72]. CFL3D is an implicit approximate factorization, thin-layer, upwind computer code using Roe's flux-difference splitting, and it also utilized the Baldwin-Lomax algebraic turbulence model [72]. This computer code was developed by Thomas, Krist, and Anderson [103]. The present results used Roe's flux-difference splitting, with pure second order upwind extrapolation, which did not require a flux limiter. As can be seen in these figures, the present results are in good agreement with the experimental data [102], except at the second shock location at the $\eta = 0.80$ station, shown in Fig. 6.4.28d. Here all of the numerical results disagreed with the experimental data.

The convergence history for this case can be seen in Fig. 6.4.29. Although the residual, which is the L_2 norm of the density, only converges three and a half orders, the coefficients of lift and drag and the number of normalized supersonic points are converged in less than two hundred work units.

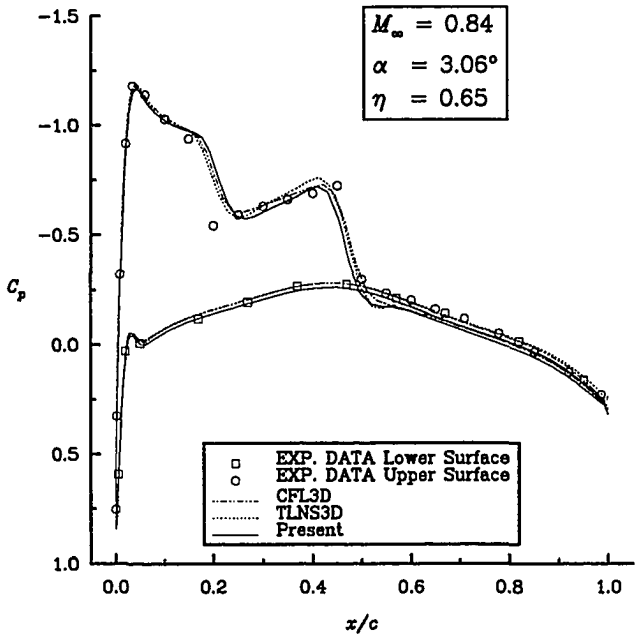


(a)

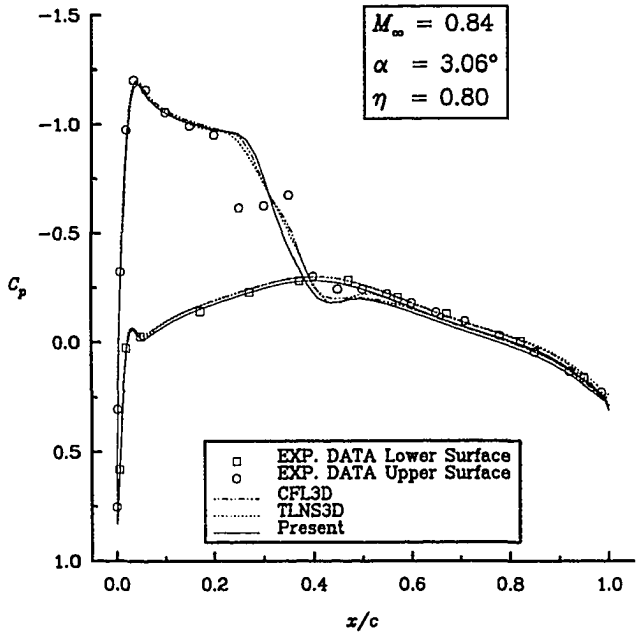


(b)

Figure 6.4.28 Comparison Between the Present Results and Other Numerical Results for the ONERA M6 Wing. (Continued . . .)

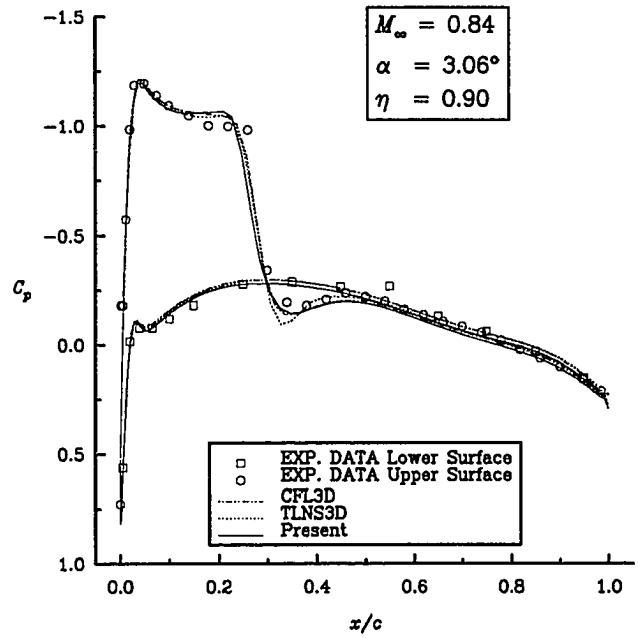


(c)

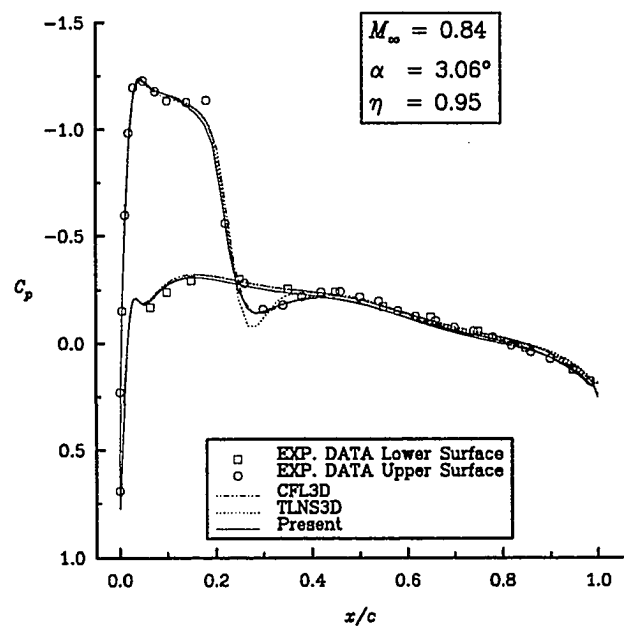


(d)

Figure 6.4.28 Comparison Between the Present Results and Other Numerical Results for the ONERA M6 Wing. (Continued . . .)



(e)



(f)

Figure 6.4.28 Comparison Between the Present Results and Other Numerical Results for the ONERA M6 Wing.

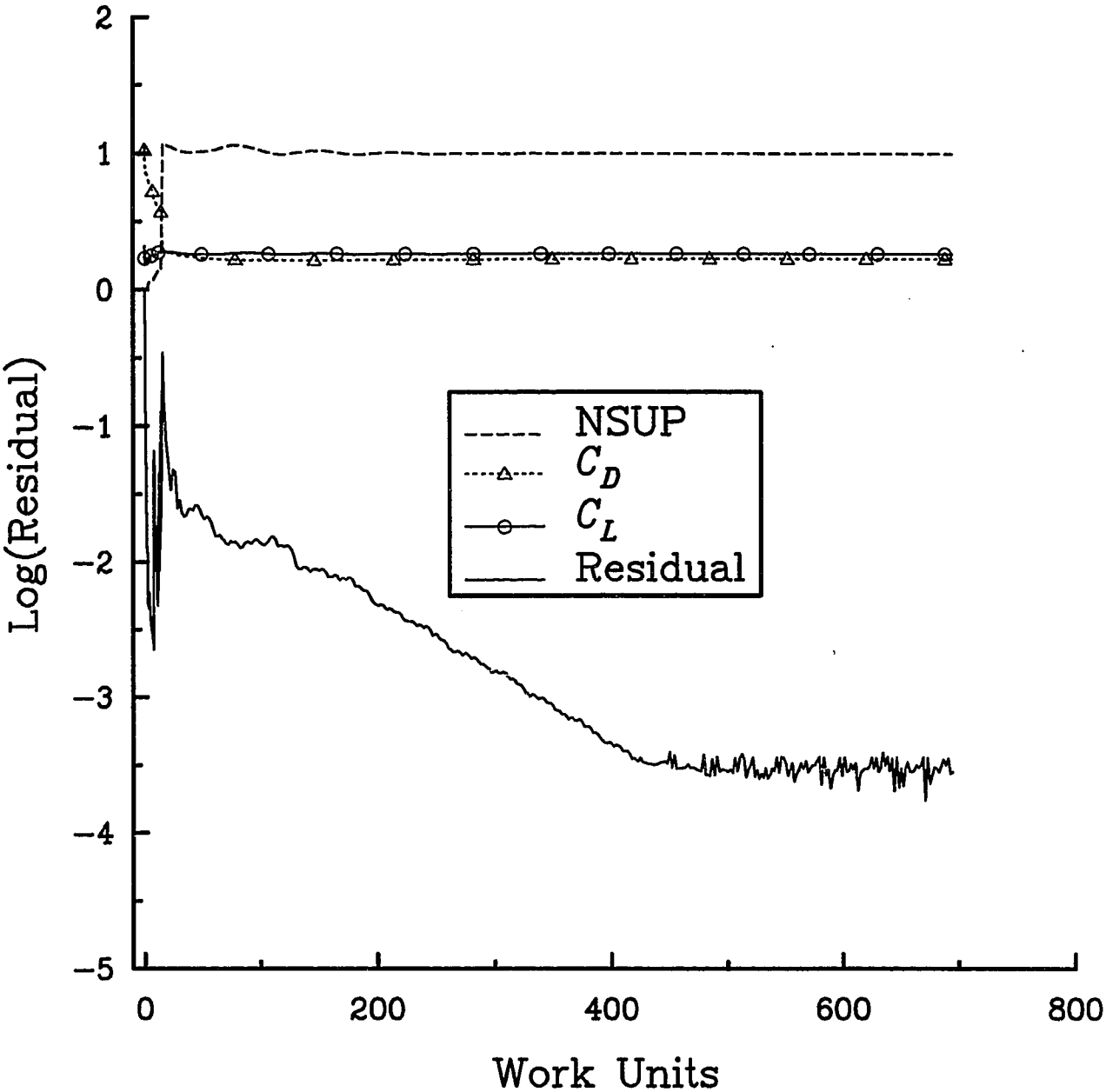


Figure 6.4.29 Convergence History for ONERA M6
Wing at $M_\infty = 0.84$, $\alpha = 3.06^o$, $R = 11.7 \times 10^6$/unit.

6.5 Requirements for Non-Interface with Interface Multiple Boundary Conditions

The block interface difficulty comes at the intersection of the trailing edge of the wing and the wake region. When using one block to solve for the wing flow, the wake region is treated as an interface. In this case the "j = jmin" face of the block, which is a constant ζ surface, has three boundary conditions. Starting at "i = imin", which is the lower half of the exit plane, and continuing to the trailing edge of the wing, the "jmin" surface boundary condition is an interface across the wake cut. Around the wing, the boundary condition is a turbulent solid wall. Continuing from the upper wing surface trailing edge to "imax", which is the upper half of the exit plane boundary, the "jmin" surface boundary condition is an interface across the wake cut to the first "jmin" interface boundary condition; therefore across the wake the "jmin" face of the block exchanges information with itself. The interface conditions used for the corner flow test case, where the grid was divided into eight blocks, will cause problems at the trailing edge of the wing. The process of gathering variable information from the extra ghost cells across the interface becomes fatal in this case because it replaces the solid wall boundary conditions with interior flow cell values. This occurs at the first two pairs of cells, at the trailing edge of the wing on both the upper and lower surfaces. This can be seen in Fig. 6.5.30, where the affected cells are indicated by the octagons and circles. The thick lined region represents the interior cells, and the thin lined region represents the ghost cells. The surface between the two regions is part solid wall, which is the wing indicated by the hash marks, and the dotted lines indicate the wake cut. If the interface conditions were not adjusted for this particular case, the values for the ghost cells on the lower side of the wing would be the "y's" in the octagons rather than the "sL" values, which are the solid wall boundary conditions for the lower surface. The same problem exists for the circled

ghost cells on the upper side of the wing. Obviously these ghost cells need to be set to their proper values before they are used. This is a problem for this particular situation

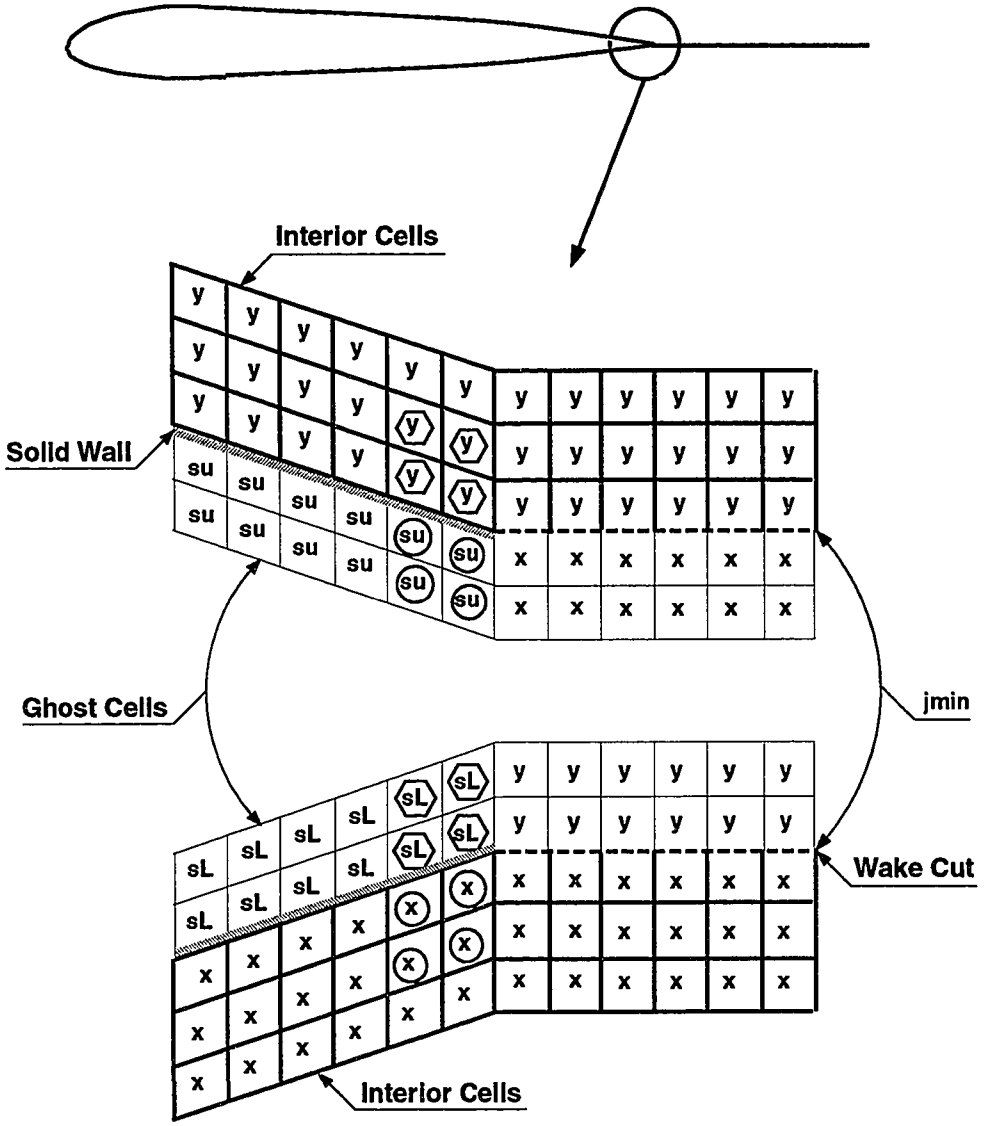


Figure 6.5.30 Interface Condition at Trailing Edge of Wing.

only because a non-interface and an interface boundary condition are both used on the same block face. This problem can be eliminated by enforcing the solid wall boundary condition again, after the interface routine has been executed. To identify this problem requires an evaluation of the types of multiple boundaries prescribed for a block face. Note that at the "kmax" face, which is at the wing tip and is where the C-O mesh folds

or closes onto itself, the interface condition is used again to allow the "kmax" surface to exchange information with itself. The bottom half of the C mesh, which starts at "imin" and continues to the leading edge of the wing, exchanges information with the top half of the wing which starts at the leading edge and continues to "imax".

6.6 Afterbody with Internal Nozzle

The final test case required a multi-block code to handle internal and external flow for an afterbody configuration, shown in Figs. 6.6.31 and 6.6.32. This afterbody was examined in a wind tunnel at various free-stream Mach numbers by Putnam and Mercer [104], and Compton, Thomas, Abeyounis, and Mason [105]. For certain cases C_p data were obtained from the center of the top and side of the boat-tail. The external geometry was relatively easy to grid, except for the boat-tail, which required the use of a ninth order super elliptic equation, with various offsets to produce the correct curvature for the boat-tail corner edges. The formulations and geometry of this afterbody are described in reference [104]. In the present work a polar grid was used to generate the outer surface geometry of the afterbody. This can be seen in Fig. 6.6.33. A closer view of the afterbody is shown in Fig. 6.6.34, displaying some its grid structure, where only everyother point is shown on the afterbody for visual clarity. Using a polar grid allowed for the direction normal to the external surface to always be in the η direction. This in turn provided the necessary direction for the length scale that was needed for the algebraic turbulence model. The more demanding part of the grid generation process was the interior nozzle. Air was supplied subsonically at a specified temperature and pressure into a circular cross sectional area (section FS 40.95 Fig. 6.6.31). The flow passed through a baffle plate and then continued toward a settling chamber, with the cross sectional area changing smoothly from circular to rectangular. The width of the internal geometry

remains constant, as the upper and lower surfaces converge to produce a sonic throat region. The nozzle re-expands producing a supersonic exit flow of $M_{jet\text{EXIT}} = 1.6$. The nozzle exit is rectangular, which is the type of exit a rectangular thrust vectoring/thrust reversing nozzle would have.

The grid for the internal nozzle was very demanding. First, the initial cross section was circular, which was best suited by a polar grid; however, once the cross sectional area changed to rectangular, a polar grid was no longer appropriate. In fitting a polar grid to a rectangular cross section, it becomes extremely difficult to force the radial lines, extending from the polar axis, to be normal to the sides of the rectangular cross section. Also, many computer codes have difficulty accommodating singular grid lines, such as the polar axis in this case. The problem generally stems from the fact that at singular grid lines, one face of the control volume cell has a zero area. This problem is accentuated when the grid lines are packed in the circumferential direction, creating highly skewed cells. One way to alleviate both problems is to use an H-H grid topology for the internal geometry. This eliminates the singular line problem and the grid lines are naturally normal to the solid surfaces. To capture the turbulence effects at the walls requires a dense packing of points, which with the H-H topology results in an extremely large number of points in the corners of the rectangular cross section. Also, at the exit of the nozzle, which is at the end of the afterbody, the grid lines from the internal grid are required to match the external grid lines with C^1 continuity. This is an obvious problem. For the external grid to have such a large cluster of points at the corner edges of the afterbody would have required a tremendous number of grid lines, which is infeasible with the current memory restrictions on today's super computers. Furthermore, such a large number of points would require a large amount of CPU time to achieve converged flow solutions. The best approach for griding the interior nozzle is to use a polar grid at the solid wall

surfaces, pack the necessary number of points for the turbulent boundary layer, and then have the polar grid interface with an H-H grid, which then fills in the remainder of the interior of the grid and eliminates the singular line. Three x–locations were chosen to show the interfacing of the H-H topology and the polar topology in Fig. 6.6.35. In this figure, every third point is shown for the polar topology for visual clarity. This griding approach provides the best compromise. A schematic of a cross section of the internal nozzle is shown in Fig. 6.6.36. The band of polar grid properly meets the external grid with C^1 continuity at the exit of the afterbody, as well as provides the normal distance from the solid interior walls for the length scale necessary for the algebraic turbulence model. Plus, having the polar grid interface with the H-H grid reduces the difficulty in maintaining the polar lines normal to the solid surfaces. The polar grid meshed with the H-H grid quite easily, with the caveat that the cell sizes across the interface had to be within 20% of each other. This is the same rate of change that is allowed between cells in the absence of an interface. If this were not enforced there would be metric discontinuities across the interface, which would contaminate the solution.

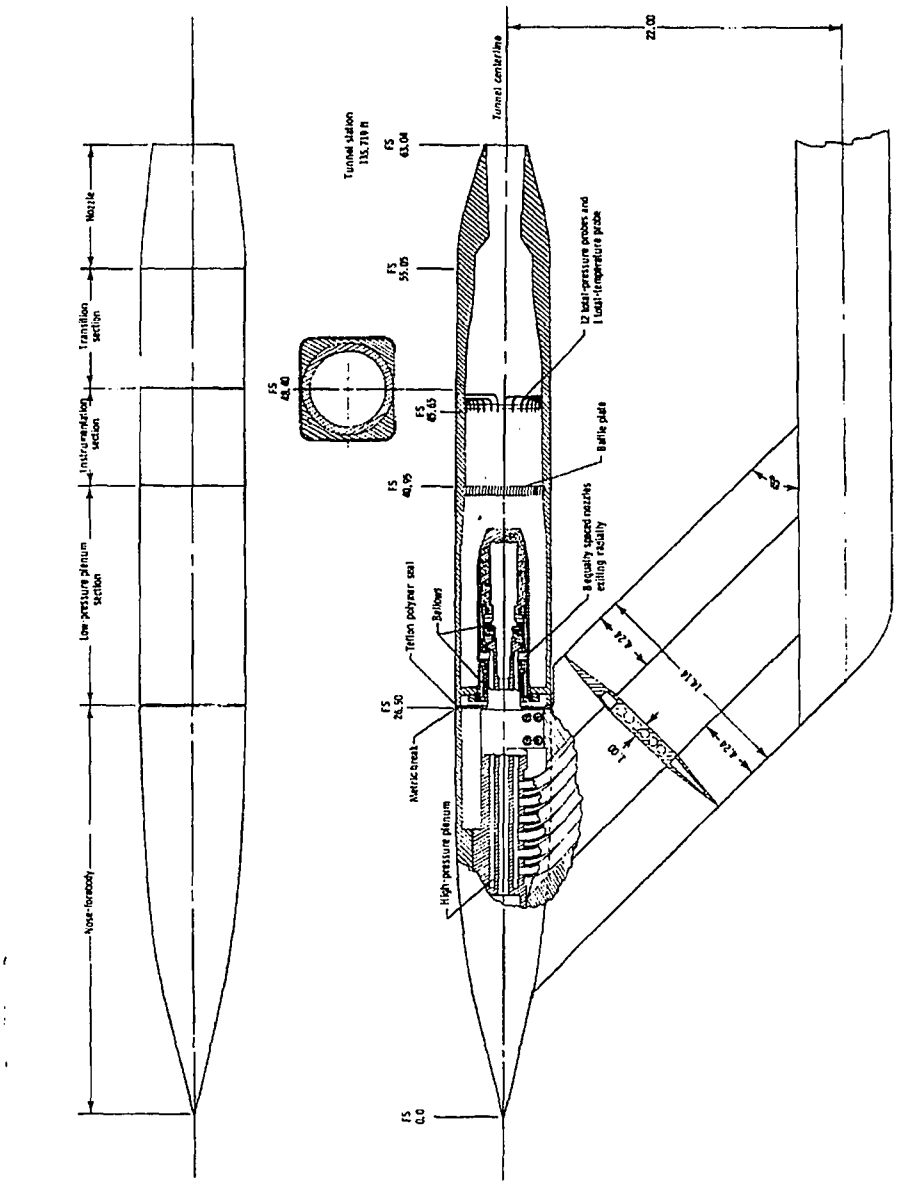


Figure 6.6.31 Sketch Of Afterbody Model Showing Internal Details. All Dimesions are in Inches Unless Otherwise Noted [104].

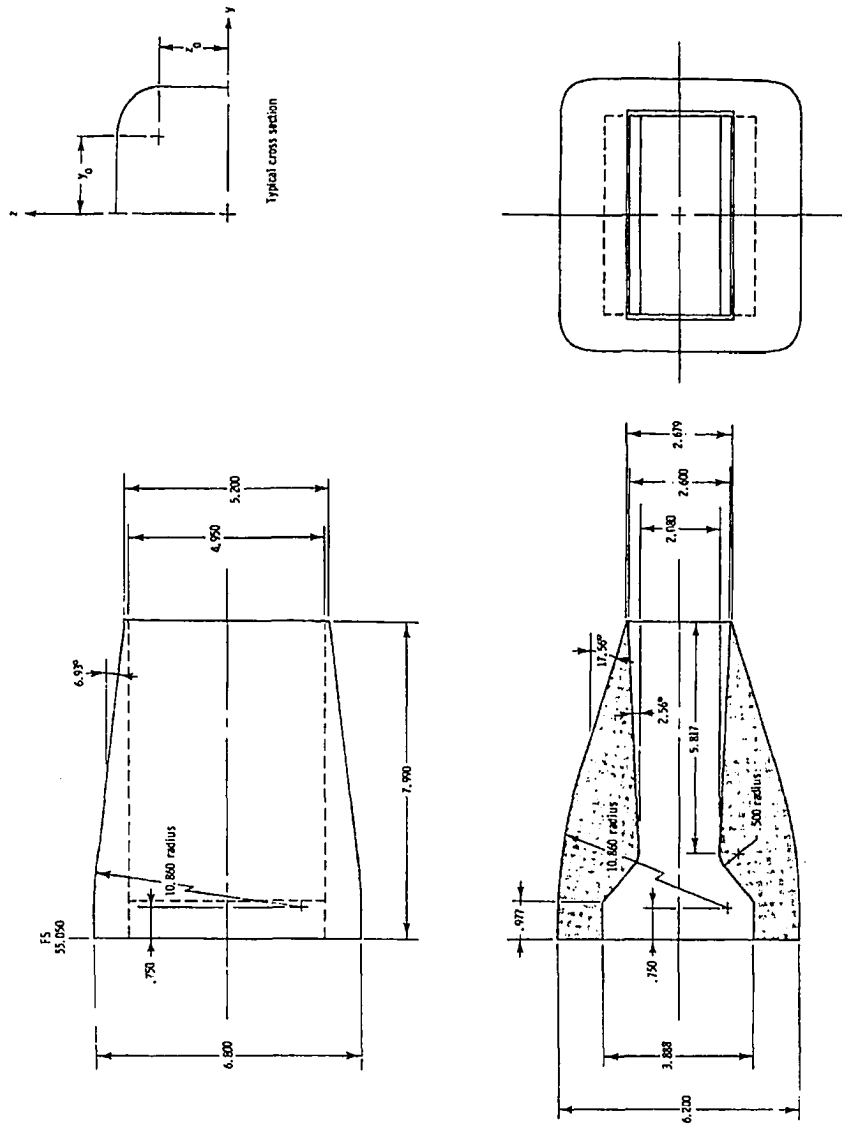


Figure 6.6.32 Details of the Nozzle. Linear Dimensions Are in Inches [104].

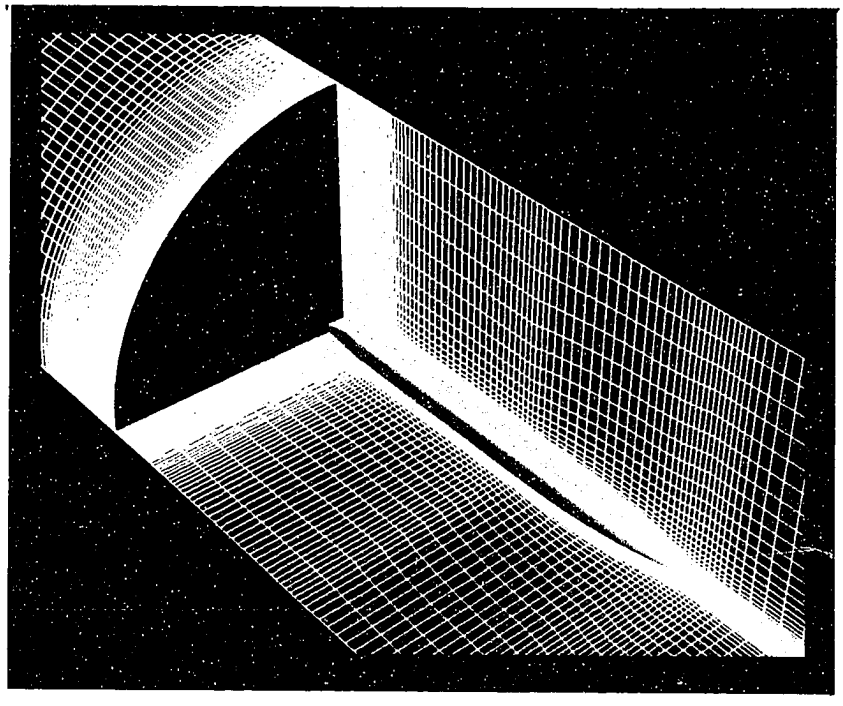


Figure 6.6.33 Afterbody Surface and Exterior Polar Grid Configuration.

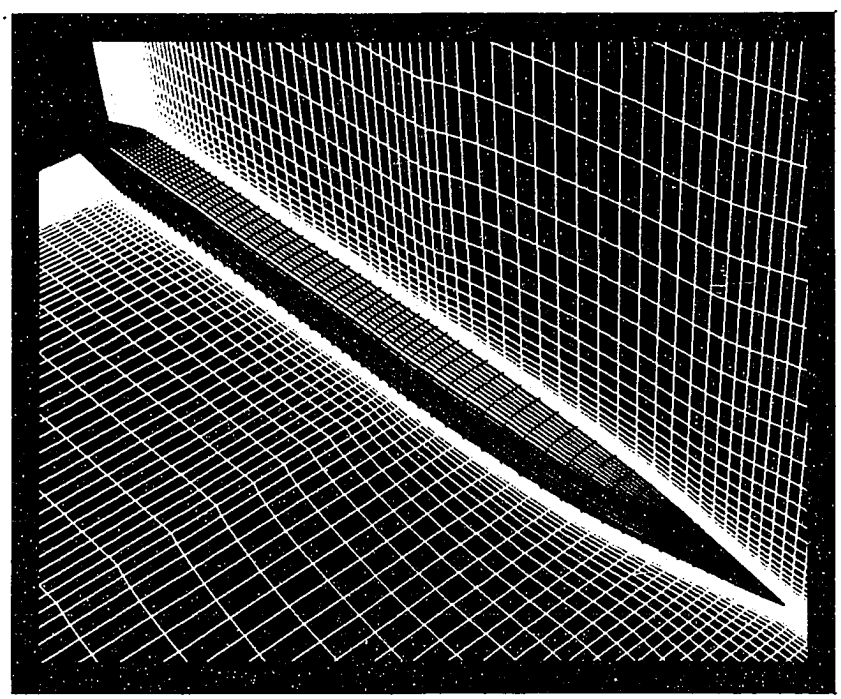


Figure 6.6.34 Afterbody Grid Geometry.

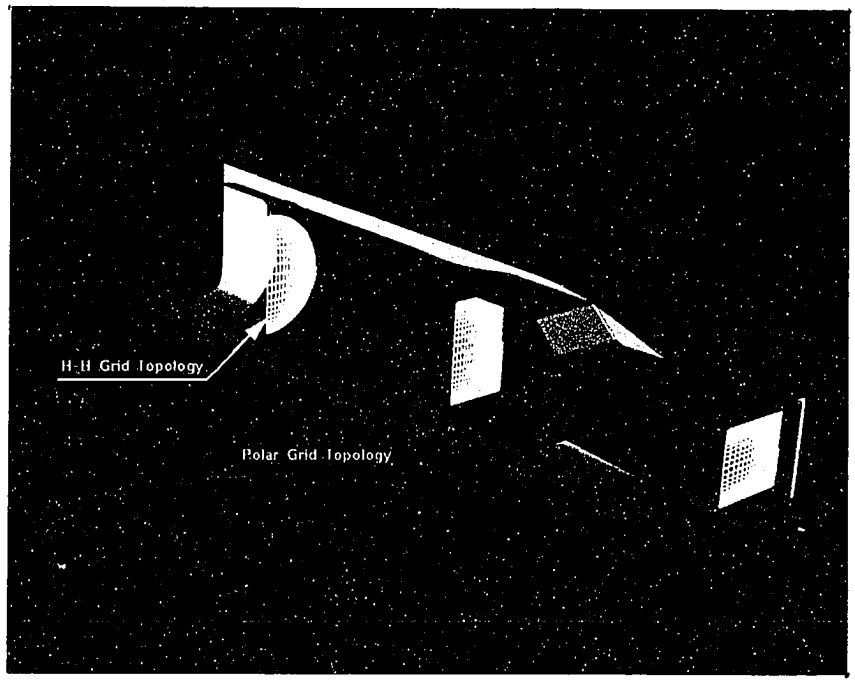


Figure 6.6.35 Internal Nozzle with Combined H-H and Polar Grid Topologies.

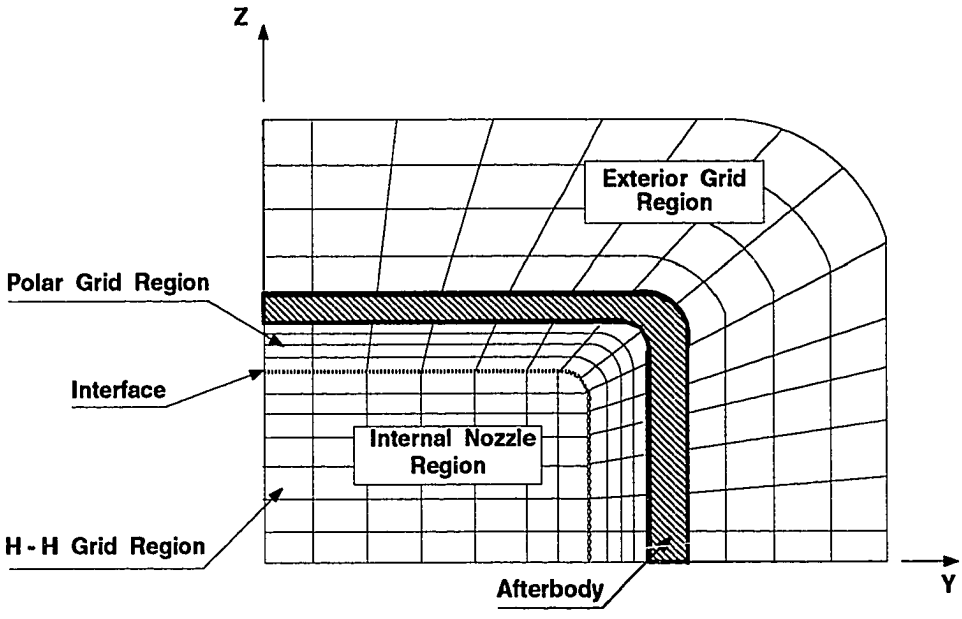


Figure 6.6.36 Schematic of H-H Grid and Polar Grid Topologies Interfacing.

The first case tested was for $M_\infty = 0.60$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.97$, and $R = 273,000/(\text{unit length})$, with the afterbody being 63.04 units long. The grid used in Ref [105] was obtained, which contained only the external geometry. This grid was used as a preliminary grid only, because it is not well suited for multigrid applications since locations of geometrical change were not at multigridable indices. This causes the physical locations to change for different multigrid levels, which in turn can introduce error. Calculations were performed on this grid, with the assumption that the plume emanating from the nozzle would remain the same size as the cross sectional area of the nozzle. This grid configuration assumed the end of the afterbody had a sharp trailing edge, rather than incorporate the flat surface which separates the external surface from the internal nozzle. The difference is shown schematically in Fig. 6.6.37. The grid had 129x33x65 points, with 129 points streamwise, 33 points distributed circumferentially, and 65 points normal to the exterior surface. The cell spacing normal to the surface was $1\text{x}10^{-4}$ units. Only half of the body was grided,

because the angle of attack and yaw were zero; therefore the flow was symmetric about the longitudinal axis. The results for this case are shown in Fig. 6.6.38, for the C_p data on the top of the boat-tail, and Fig. 6.6.39 for the C_p data on the side of the boat-tail. Good agreement was obtained for both locations in comparison with the experimental data [105], with the exception of the end of the boat-tail. These results are comparable to the numerical results of Compton, Thomas, Abeyounis, and Mason [105], for the same case where they used a Baldwin-Lomax algebraic turbulence model. The same grid was used for a case at $M_\infty = 0.80$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.99$, and $R = 309,000/(\text{unit length})$. The results for the top and side wall C_p data are shown in Figs. 6.6.40 and 6.6.41, respectively. Again, the agreement with the experimental data, was good [105], and comparable to the numerical results of Compton et al [105].

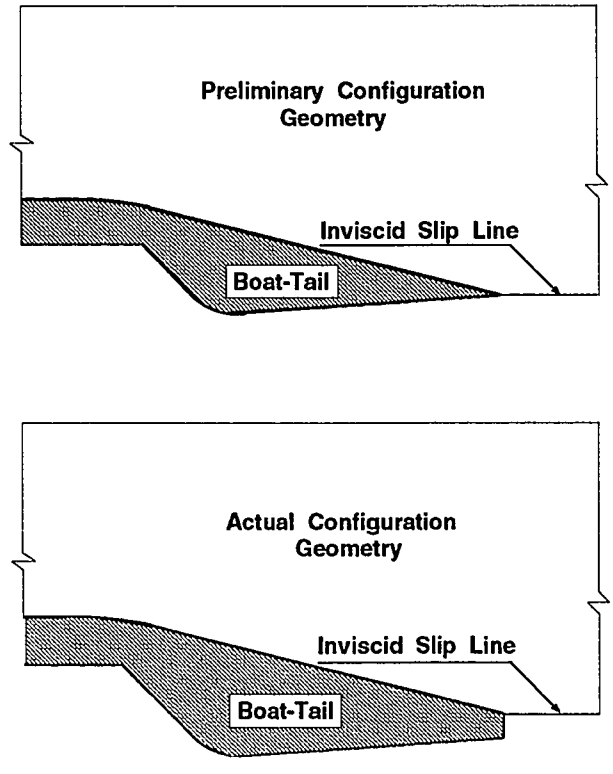


Figure 6.6.37 Comparison of Preliminary Configuration Geometry and Actual Configuration Geometry.

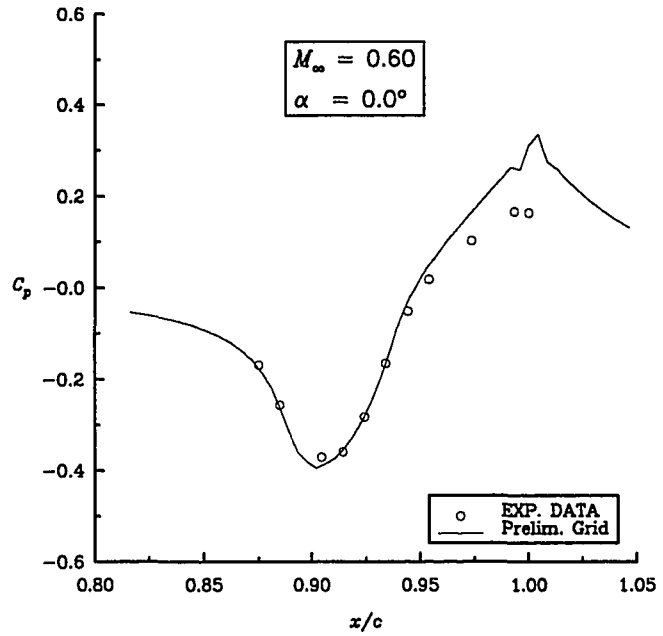


Figure 6.6.38 Preliminary Afterbody Nozzle Top Wall Pressure Coefficient.

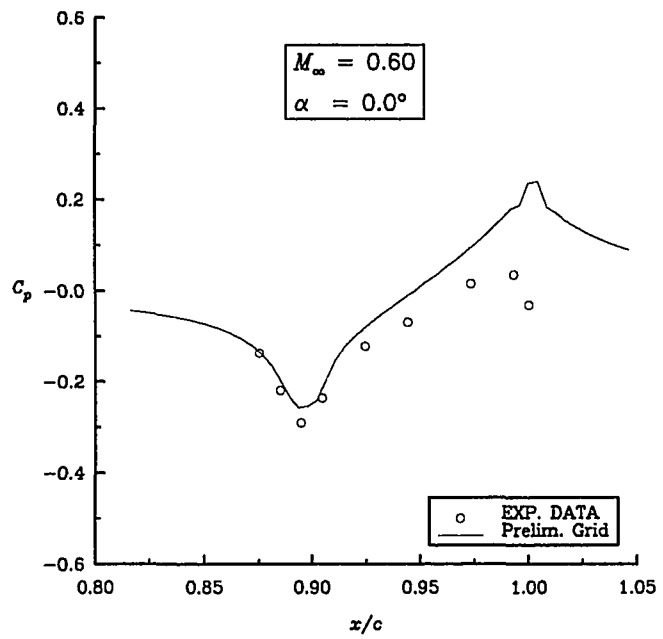


Figure 6.6.39 Preliminary Afterbody Nozzle Side Wall Pressure Coefficient.

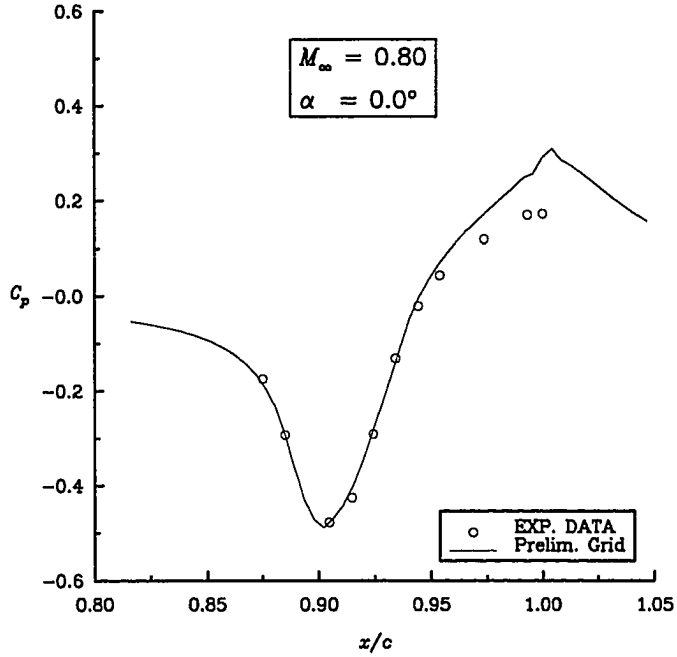


Figure 6.6.40 Preliminary Afterbody Nozzle Top Wall Pressure Coefficient.

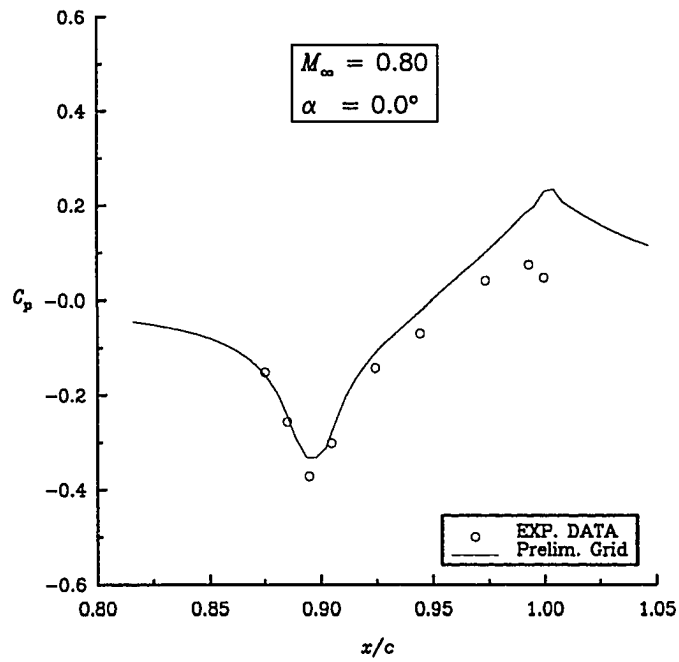


Figure 6.6.41 Preliminary Afterbody Nozzle Top Wall Pressure Coefficient.

Since the interior grid needed to be computed, along with the concerns about the external grid being ill suited for multigrid, a new multigrid compatible grid was generated. This grid had 177 points in the streamwise direction, with 97 points on the body. Since the flow was to be symmetric about the longitudinal axis, only the quarter plane was generated, with 33 points in the circumferential direction, and 49 points in the direction normal to the body. The cell spacing normal to the body on the boat-tail was $5\text{x}10^{-5}$ units.

The first task after generating the new grid was to make a comparison between it and the preliminary grid, using only the external grid configurations. The results for the top and side wall C_p data for the case of $M_\infty = 0.60$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.97$, and $R = 273{,}000/(\text{unit length})$, are shown in Figs. 6.6.42 and 6.6.43, respectively. As can be seen, the comparison between the two different grids is good. The new grid was then tested for the case of $M_\infty = 0.80$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.99$,and $R = 309{,}000/(\text{unit length})$. Again the agreement was good, as can be seen by comparisons of the C_p data for the top and side walls in Figs. 6.6.44 and 6.6.45, respectively. For these cases the Baldwin-Lomax algebraic turbulence model was active only over the solid wall surfaces, and the shear line, which should have been between the plume and the free-steam flow was treated as an inviscid surface.

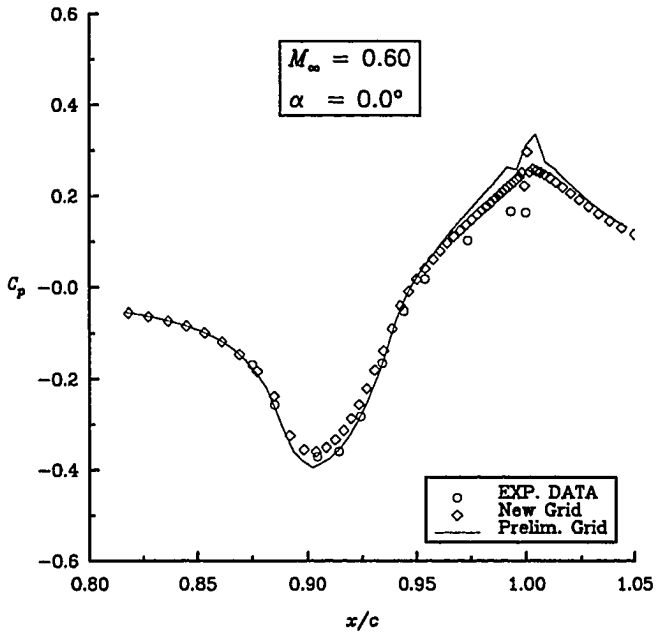


Figure 6.6.42 Comparison of Single-Block Afterbody Grids for Top Wall Pressure Coefficient.

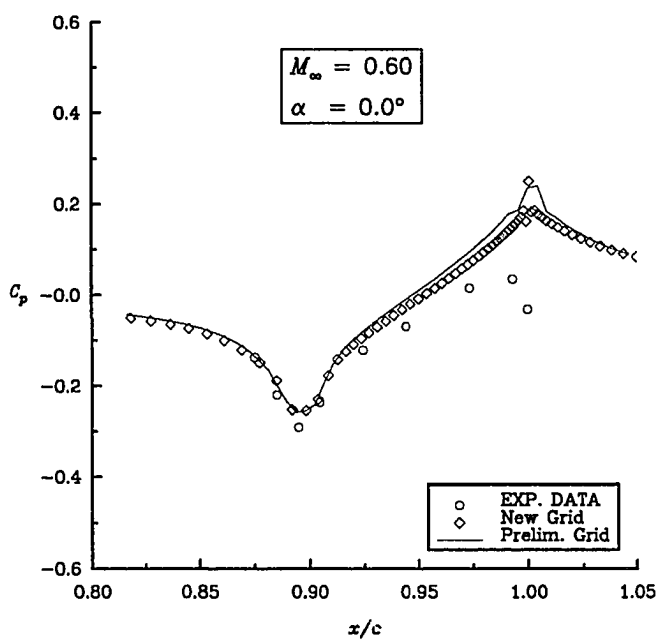


Figure 6.6.43 Comparison of Single-Block Afterbody Grids for Side Wall Pressure Coefficient.

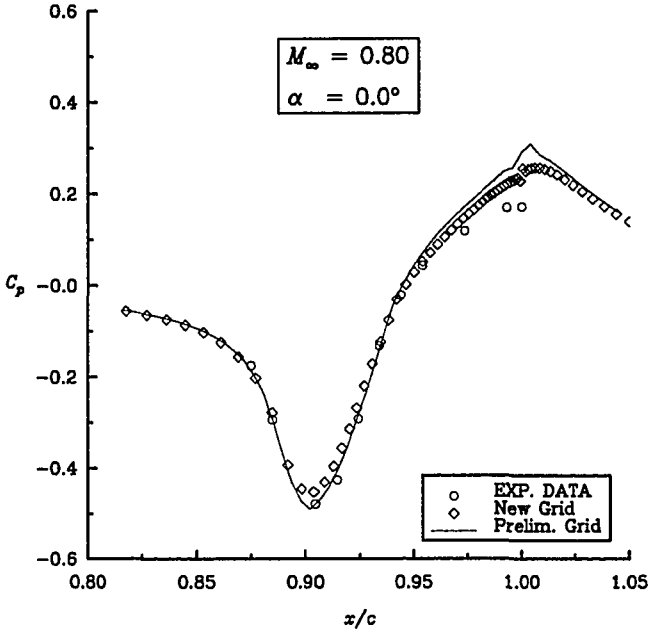


Figure 6.6.44 Comparison of Single-Block Afterbody Grids for Top Wall Pressure Coefficient.

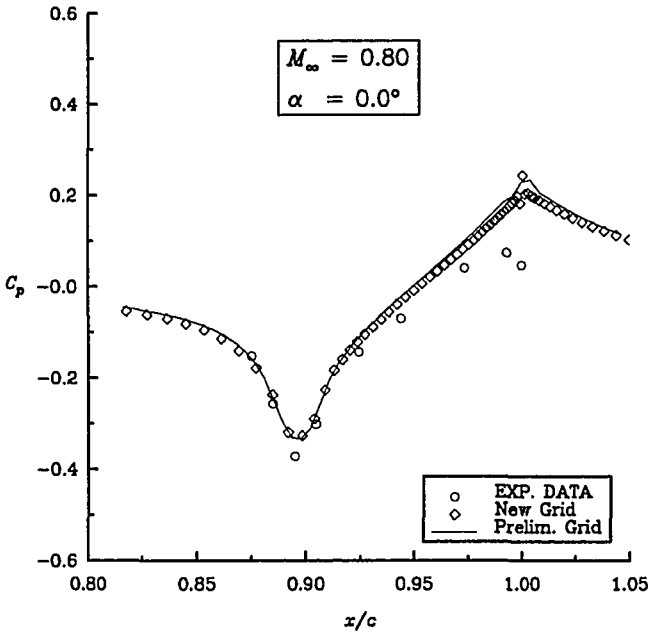


Figure 6.6.45 Comparison of Single-Block Afterbody Grids for Side Wall Pressure Coefficient

After verifying that the newly generated grid performed adequately, the test case of $M_\infty = 0.60$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.97$, and $R = 273,000/(\text{unit length})$ was tested again, utilizing the grid that included the thickness between the exterior surface and the internal grid at the end of the afterbody, as shown in Fig. 6.6.46. As can be seen by the comparison of C_p predictions for the top and side walls in Fig. 6.6.47 and 6.6.48, the agreement with the experimental data [105] was very similar to that of the single-block configuration. However, the blunt geometry was somewhat better in predicting the aft portion of the sidewall pressure coefficient.

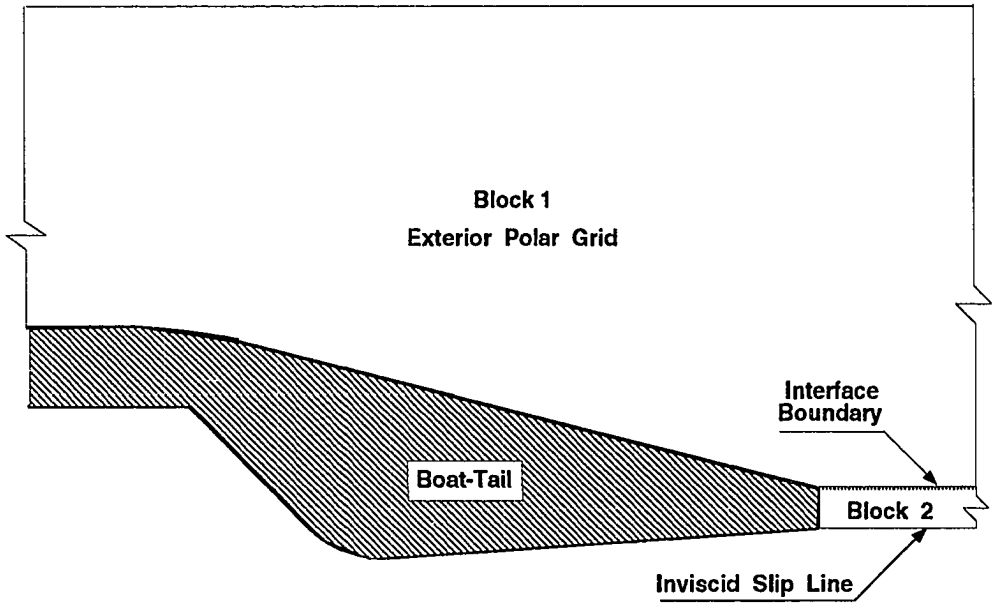


Figure 6.6.46 Schematic of Afterbody for Two Block External Configuration.

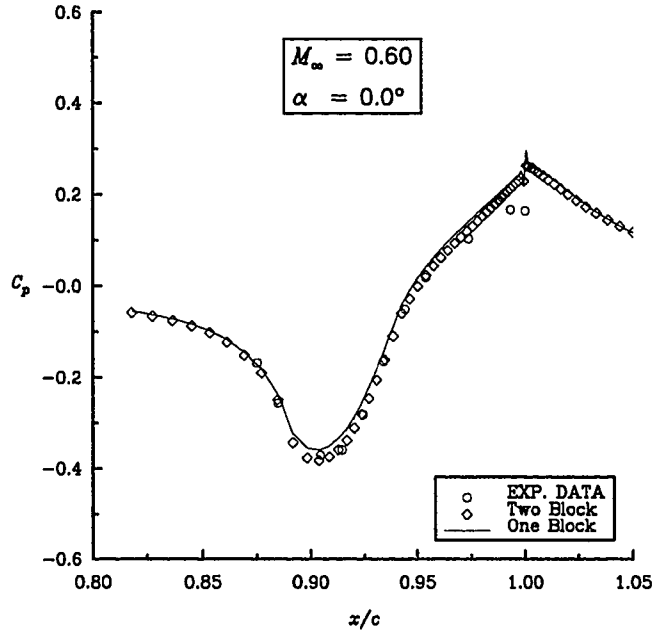


Figure 6.6.47 Comparison of Single-Block and Two-Block Afterbody Nozzle Top Wall Pressure Coefficient.

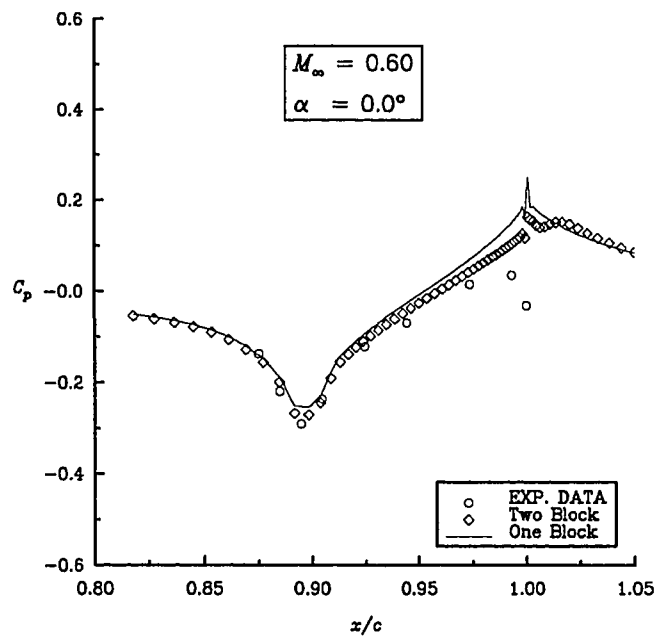


Figure 6.6.48 Comparison of Single-Block and Two-Block Afterbody Nozzle Side Wall Pressure Coefficient.

The same case was examined again employing both the plume grids and the internal nozzle grids, as shown in Fig. 6.6.49. This gave a total of four blocks. The external block extended only to the end of the afterbody, as did the interior polar block. The H-H grid block began in the circular region of the internal nozzle, as did the interior polar grid, and extended all the way to the exit plane of the numerical domain, which was approximately 110 units downstream from the end of the afterbody. The fourth block

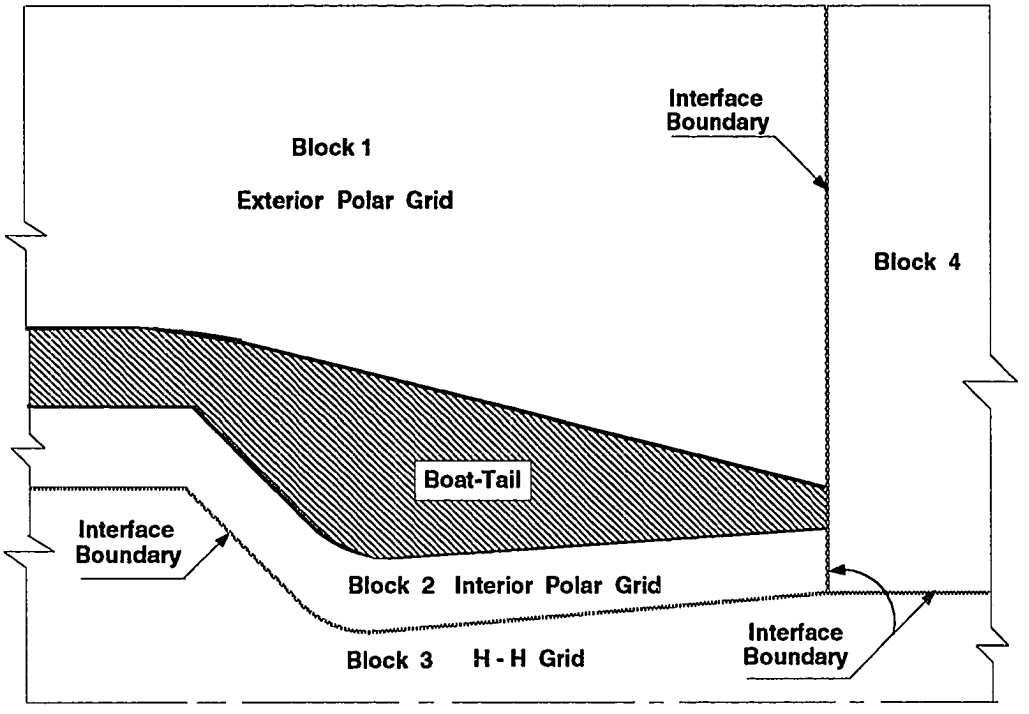


Figure 6.6.49 Schematic of Four-Block Internal and External Afterbody Configuration

began at the end of the afterbody, where that face of the block incorporated three boundary conditions. One was to interface with the polar grid block from the interior nozzle. The second boundary condition was to represent the solid wall thickness between the internal and external surfaces of the afterbody, and the final condition was to interface with the external block of the afterbody. Again, the algebraic turbulence model was employed only on the solid wall surfaces of the afterbody, including both the interior and exterior surfaces. The results of this case are shown as comparisons of C_p data for the top and

side walls in Fig. 6.6.50 and 6.6.51, where there is good agreement with the experimental data [105] and the numerical results of the two-block case, except at the trailing edge of the boat-tail. One reason the four-block configuration predicted higher pressures on the surface of the boat-tail maybe because as the flow exits the nozzle, it has a velocity that is not parallel to the horizontal axis, as was the assumption for the one- and two-block configurations. There is a w velocity component due to the divergence of the nozzle, which forces the shear layer to turn upward, and generates a higher pressure upstream, on the boat-tail. Also, the flow coming out of the nozzle is supersonic, so it is going to expand if the surrounding pressure regions allow it. This expansion will also cause the shear line to turn outward. Flow from on the boat-tail definitely expands into the thickness region at the end of the afterbody, and when subsonic flow expands its velocity drops and its pressure increases. This higher pressure in the thickness region can influence the pressures upstream on the boat-tail, and there is no direct pressure correction term in the Baldwin-Lomax algebraic turbulence model; therefore the higher pressures at the end of the afterbody, in the thickness region, may erroneously influence the pressures on the boat-tail. Also in this region, there appears to be a re-circulation bubble which can add to the higher upstream pressures.

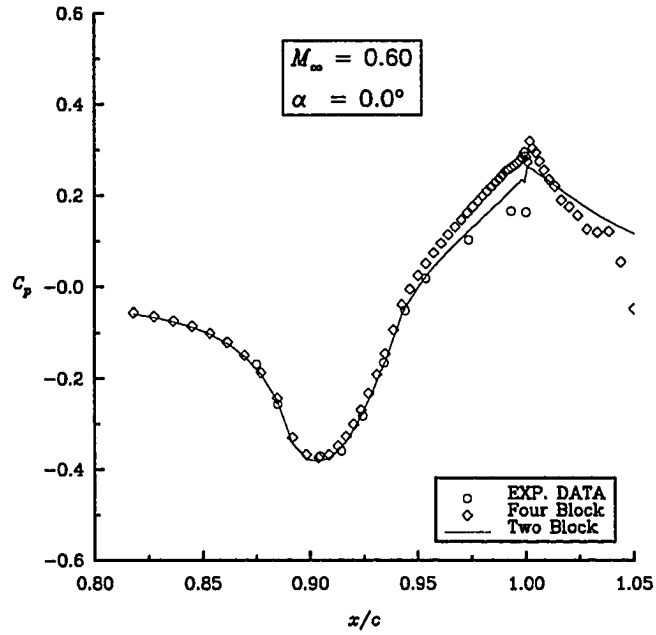


Figure 6.6.50 Comparison of Two-Block and Four-Block Afterbody Nozzle Top Wall Pressure Coefficient.

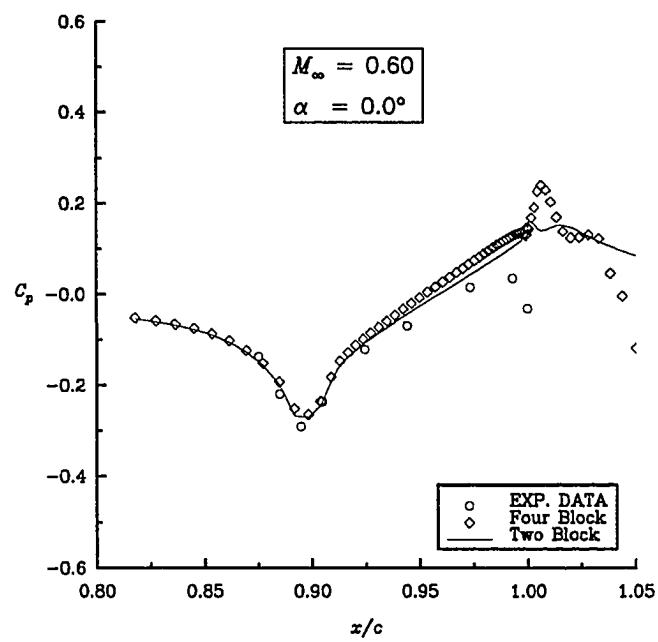


Figure 6.6.51 Comparison of Two-Block and Four-Block Afterbody Nozzle Side Wall Pressure Coefficient.

The same four-block configuration was tested for the $M_\infty = 0.80$, $\alpha = 0.0$, $M_{jet} = 0.33$, $p_{jet}/p_\infty = 3.71$, $T_{jet}/T_\infty = 0.99$, and $R = 309,000/(\text{unit length})$ case. Again, the pressures on the boat-tail were higher than what the single-block configuration predicted, as is displayed by the C_p data for the top and side walls in Figs. 6.6.52 and 6.6.53.

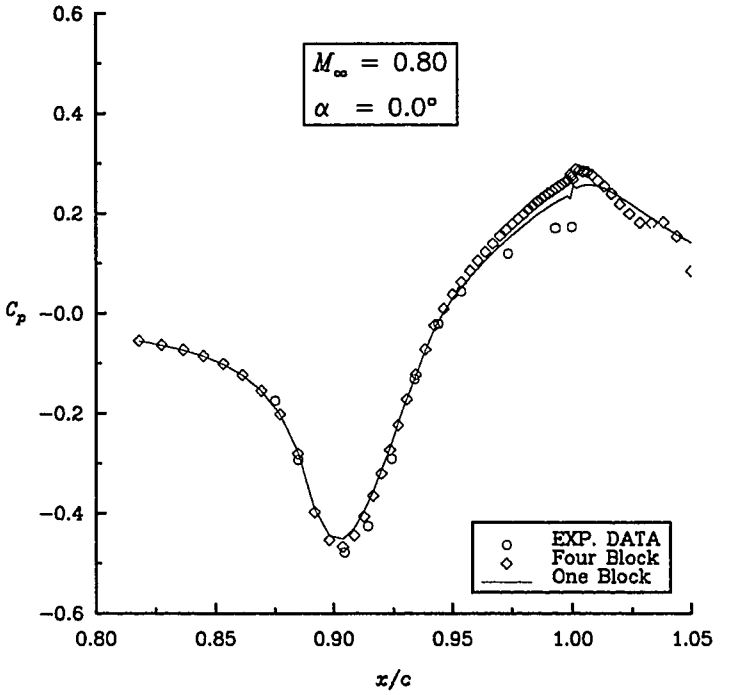


Figure 6.6.52 Comparison of Single-Block and Four-Block Afterbody Nozzle Top Wall Pressure Coefficient.

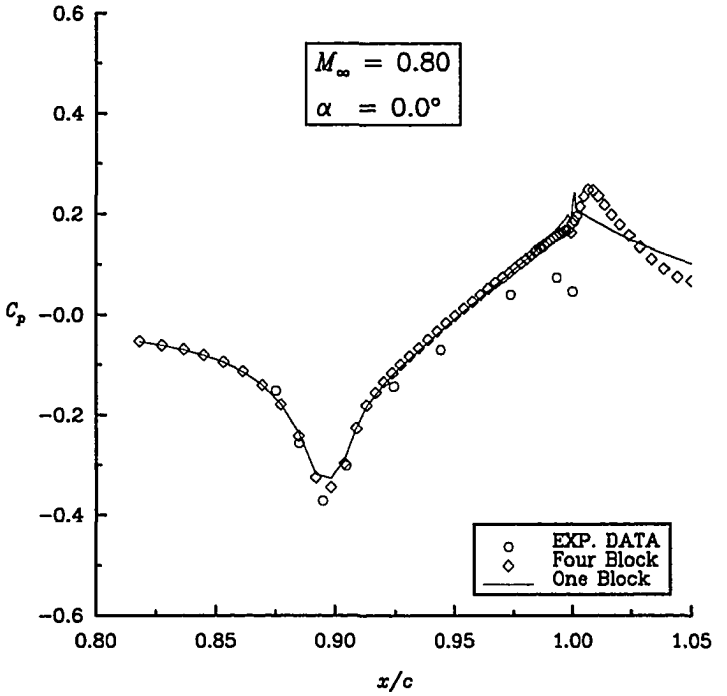


Figure 6.6.53 Comparison of Single-Block and Four-Block Afterbody Nozzle Side Wall Pressure Coefficient.

Chapter 7 CONCLUSIONS

This work was aimed at developing a state of the art computer code, capable of evaluating arbitrary, complex geometric configurations. The computer code developed utilizes upwind solvers, van Leer's flux-vector splitting and Roe's flux-difference splitting, coupled with an explicit modified Runge-Kutta time integration scheme. To accelerate the rate of convergence the techniques of FAS multigrid, variable coefficient residual smoothing and constant coefficient corrector smoothing were incorporated. Development of the computer code was an attempt to capitalize on the benefits of both the upwind implicit time integration and the central-difference explicit Runge-Kutta time integration methods. Although the upwind methods require more CPU time for the flux evaluations, they provide better shock capturing and less smearing of contact discontinuities than the central-difference methods. However, the multistage central difference methods are generally more capable of smoothing out the high frequency errors, allowing the multigrid acceleration techniques to restrict information without causing aliasing. For the upwind computer codes to provide a sufficient amount of smoothing an implicit time integration is usually used, which again requires more CPU time. The present computer code incorporated a modified Runge-Kutta time integration technique, very similar to what the central-difference codes employ, in an effort to attain the speed of the central-difference codes, coupled with the accuracy of the upwind codes. The thrust of this work was to aid in the development of the computer code and provide the flexibility of multi-block capabilities. It was desired that this computer code be capable of accommodating any configuration that could be defined by six-sided blocks. To

incorporate multigrid techniques efficiently requires an understanding of how computers store data, and operate. The same can be said for incorporating multi-block capabilities. Many of the qualities a computer program requires for multigrid techniques are the same for multi-block capabilities, which are mainly grid level, block, and topology independent subroutines. These attributes allow for having only one subroutine to execute a particular task, regardless of what grid level, block, or mesh topology is being evaluated. This leads to having a generic computer program, which allows it will be applicable to any configuration.

The test cases were chosen based on the criteria of validating the computer code and displaying its versatility. The corner flow required three-dimensional flow capabilities. Also, when it was divided into eight blocks, it demanded special interface requirements allowing the multigrid prolongation calculations to be independent of the interface locations. This type of interface condition would be common at the back end of an aircraft engine, where internal and external flows meet, or where flows join from around different solid wall body parts. Good agreement with other numerical results [98, 99] were obtained. This case was performed using multigrid acceleration, with and without variable coefficient residual smoothing. Since this case was supersonic flow, and the grids were not highly stretched, the residual smoothing did not aid the rate of convergence. This acceleration technique is best suited for stretched grids, such as viscous grids, and obviously more compatible with subsonic and transonic flows, than for supersonic flows.

The pseudo two-dimensional jet plume required multiple inlet boundary conditions, and provided a testing ground for comparing how well different solvers would compute the slip line and shock surface. Favorable comparisons were made with the validated shock fitting code of Salas [101]. For the upwind methods, Roe's flux-difference splitting and van Leer's flux-vector splitting, their ability to accurately predict shocks and slip lines

was dependent on how these structures were aligned with the computational grid. The flux-difference splitting method should provide better slip line resolution than flux-vector splitting, because it is an approximate Riemann solver, which by design is capable of resolving contact surfaces. The flux-vector splitting method resolved the slip line just as accurately as the flux-difference splitting method, partly because the slip line was not directly aligned with the grid. If it was better aligned, the flux-difference splitting method should have provided better resolution. Based on this case, it was determined that both upwind methods would provide acceptable results for the rest of the cases studied.

The laminar and turbulent flat plate cases were computed with great success. The laminar results compared well with the Blasius solution, indicating that the viscous terms had been properly incorporated into the governing equations. The turbulent results compared well with the analytical velocity profile and skin friction, plus the law of the wall plot. Thus indicating that the Baldwin-Lomax algebraic turbulence model [72] was functioning properly for attached turbulent flows. Full multigrid acceleration was used, with the cell weighted residual smoothing. These two techniques made a large improvement in the convergence rate for these two cases.

The ONERA M6 wing cases were true three-dimensional attached turbulent flows. Results for both cases compared well with the experimental data [102] and other numerical results [86, 103]. These cases were computed using FMG acceleration and variable coefficient residual smoothing. Again, these acceleration techniques provided improved convergence rates. This configuration also placed special requirements on the interface conditions, which are needed when a block face has an interface and a wall boundary condition next to each other.

The final case studied was the afterbody configuration. This configuration required a multi-block code in order to accommodate the internal and external geometries. The

results for this case are preliminary due to the quality of the grid. The resolution on the solid surfaces was sufficient for the formulation of the boundary layers, but at locations of large geometrical change many cells became abnormally skewed. Much could be done to improve the grid, with the proper experience and software. The afterbody grid went through four revisions before a final selection was made. Three main areas required improvements: the nose of the afterbody, the sharp compression in the nozzle geometry (which started just upstream of the throat) and the thickness region at the end of the afterbody. The nose created numerical difficulty because of the polar axis coupled with viscous compatible cell spacing. Relaxing the cell spacing normal to the nose and along the polar axis greatly relieved some of the numerical difficulty, allowing this region to numerically converge. The thickness region, at the end of the afterbody, required a smooth continuous variation of cell spacing from the interior nozzle to the exterior surface. This was a difficult procedure, especially at the corners, because the grid lines from the interior nozzle had to match the grid lines of the external surface. Also, the thickness of this region, at the end of the afterbody, changed at the corners of the boat-tail. The top and bottom thicknesses were 0.0395 units, and the sides were 0.125 units. This made having smooth variations of cell sizes from the interior nozzle to the exterior surface more difficult. The start of the compression region for the interior nozzle requires true three-dimensional grid surfaces to reduce the cell skewness. This region is where many of the numerical difficulties remained. Much more sophisticated grid generation techniques would be required to effectively approach this problem.

This case required a multi-block computer program to simultaneously accommodate the internal and external domains. The computer program performed well based on the pressure coefficient results obtained for the two flow cases. Further improvements in the grid quality may not effect these results, but employing a different type of turbulence

model definitely could, especially if it retains the turbulent kinetic energy term, which is eliminated in algebraic turbulence models. Note that if the turbulent kinetic energy term is actively accounted for, then there will be a direct link with the static pressure through the equation of state [106].

Overall, the computer code performed well. Its design enabled it to handle many different configurations without requiring any alteration of the source code. This flexibility makes it a very versatile tool in examining many types of flow configurations. Also, it provided competitive solutions when compared with other numerical results [86, 98, 99, 101, 103] and experimental data [102]. The multigrid acceleration decreased the amount of work units required to obtain a solution for all cases, except supersonic Euler flow on an unstretched grid. Since the multigrid process executes averaging that resembles elliptic information propagation, it sent information upstream in the supersonic flow cases, which can obviously reduce the convergence rate. A typical example of this was the supersonic corner flow. As for subsonic and transonic flows, and flows on viscous grids, the multigrid acceleration was a great aid to accelerate the rate of convergence. The variable coefficient residual smoothing had the best impact on the viscous grids as well. This is due primarily to the fact that the residual smoothing coefficients are generated relative to a specified value of cell aspect ratios and spectral radii. This smoothing technique did aid in the rate of convergence for viscous flow cases. The corrector smoothing, which used constant coefficients, never accelerated the rate of convergence for the cases tested. It also was intended to aid the viscous flow cases, but for the range of Mach numbers and grids employed in the work presented, it actually reduced the rate of convergence for test performances, and therefore was not used in the final analysis of the flow configurations.

If the numerical efficiency of the computer program was measured based on the amount of work units necessary to obtain global flow values, the computer code did

well. Generally, it required approximately 150 to 200 work units of FMG, using the pure second order upwind extrapolation, four stage modified Runge-Kutta time integration, coupled with variable coefficient residual smoothing, for the afterbody cases. The global residual, which was based on the change of density, did not converge very well. Is was case dependent as well as being very grid dependent, which is why four different grids were generated for the afterbody. The wing cases converged about three orders under the same execution conditions. Again, the global quantities were obtained with just a couple hundred work units of FMG. The flat plate cases converged better, based on the amount the residual dropped, but it did require more work units until the entire boundary layer was completely developed, because of the high number of cells in the boundary layer. All of the inviscid cases converged well. Unfortunately the more numerically demanding cases, such as the viscous cases, require more frequency damping than is currently being provided by the modified Runge-Kutta time integration scheme. Much more attention needs to be directed toward modeling the frequencies that are being produced with this type of numerical approach, so that a more accurate set of modified Runge-Kutta coefficients can be determined. This problem has been investigated by a number of scientist with some success, but much still needs to be done. One issue is the incorporation of such acceleration techniques as multigrid, residual and corrector smoothing. An other issue is to actually incorporate the van Leer's flux-vector splitting and Roe's flux-difference splitting techniques in the model equations.

True multigrid performance is rarely seen in any of the complicated flow configurations, for any method, but especially for upwind methods. Having a computer program that is completely independent of cell aspect ratios is still a goal that has not been achieved by anyone in the CFD community.

BIBLIOGRAPHY

[1] Hess, J. L. and Smith, A. M. O., "Calculation of Non-Lifting Potential Flow About Arbitrary Three-Dimensional Bodies," Technical Report, Douglas Aircraft Report ES40622, 1962.

[2] Rubbert, P. E. and Saaris, G. R., "A General Three-Dimensional Potential Flow Method Applied to V/STOL Aerodynamics," SAE Paper 680304, 1968.

[3] Woodward, F. A., "An Improved Method for the Aerodynamic Analysis of Wing-Body-Tail Configurations in Subsonic and Supersonic Flow: Part 1 Theory and Application," Technical Report, NASA CR 2228 Part 1, May 1973.

[4] Kandil, O. A. and Yates Jr., E. C., "Transonic Vortex Flows Past Delta Wings: Integral Equation Approach," *AIAA Journal*, 24(11):1729–1736, 1986.

[5] Kandil, O. A. and Hong, H., "Full-Potential Integral Solution for Transonic Flows with and without Embedded Euler Domains," AIAA Paper 87–1461, 1987.

[6] Murman, E. M. and Cole, J. D., "Calculation of Plane Steady Transonic Flows, "*AIAA Journal*, 9(1):114–121, 1970.

[7] Magnus, R. and Yoshihara, H., "Inviscid Transonic Flow Over Airfoils," *AIAA Journal*, 8:2157–2162, 1970.

[8] MacCormack, R. W., "The Effect of Viscosity in Hypervelocity Impact Cratering," AIAA Paper 69-352, 1969.

[9] MacCormack, R. W. and Paullay, A. J., "Computational Efficiency Achieved by Time Splitting of Finite Difference Operators," AIAA Paper 72-154, 1972.

[10] Beam, R. M. and Warming, R. F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form,"*Journal of Computational Physics*, 22:87–110, 1976.

[11] Douglas, J., "On the Numerical Integration of $u_{xx} + u_{yy} = u_t$ By Implicit Methods," *Journal of the Society of Industrial and Applied Mathematics*, 3:42–65, 1955.

[12] Peaceman, D. W. and Rachford, H. H., "The Solution of Parabolic and Elliptic Differential Equations," *Journal of the Society of Industrial and Applied Mathematics*, 3:28–41.

[13] Douglas, J. and Gunn, J. E., "A General Formulation of Alternating Direction Methods, Part 1. Parabolic and Hyperbolic Problems,"*Numerische Mathematik*, 6:428–453.

[14] Gourlay, A. R. and Mitchell, A. R., "A Stable Implicit Difference Method for Hyperbolic Systems in Two Space Variables," *Numerical Mathematics*, 8:367–375, 1966.

[15] Briley, W. R. and McDonald, H., "Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique," *Proceedings of the 4th International Conference on Numerical Methods in Fluid Dynamics*, 1974.

[16] Steger, J. L., "Implicit Finite Difference Simulation of Flow About Arbitrary Two-Dimensional Geometries," *Journal of Computational Physics*, 16:679–686, 1978.

[17] Pulliam, T. H. and Steger, J. L., "On Implicit Finite-Difference Simulations of Three-Dimensional Flow," *Proceedings of the 16th Aerospace Sciences Meeting*, January 16–18, 1978.

[18] Beam, R. M. and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, 16:393–402, April 1978.

[19] Steger, J. L., "Coefficient Matrices for Implicit Finite Difference Solution of the Inviscid Fluid Conservation Law Equations," *Computer Methods In Applied Mechanics and Engineering*, 13:175–188, 1978.

[20] Steger, J. L. and Warming, R. F., "Flux-Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite-Difference Methods," *Journal of Computational Physics*, 40:263–293, 1981.

[21] Moretti, G., "The λ-Scheme," *Computers and Fluids*, 7:191–205, 1979.

[22] Chakravarthy, S. R. and Anderson, D. A. and Salas, M. D., "Split-Coefficient Matrix Method for Hyperbolic Systems of Gas Dynamic Equations," AIAA Paper 80-0268, January 1980.

[23] van Leer, B., "Flux-Vector Splitting for the Euler Equations," Technical Report, ICASE Report 82-30, 1982.

[24] Godunov, S. K., "A Difference Method for the Numerical Calculation of Discontinuous Solutions of Hydrodynamic Equations," *Mat. Sbornik*, pp. 271–306, 1959. Translated as JPRS 7225 by U.S. Dept. of Commerce, 1960.

[25] Roe, P. L., "The Use of the Riemann Problem in Finite Difference Schemes," *Proceedings of the 7th International Conference on Numerical Methods in Fluid Dynamics*, W. C. Reynolds and R. W. MacCormack, eds., pp. 354–359. Springer, 1981. Stanford, 1980.

[26] Roe, P. L., "Numerical Modeling of Shock Waves and Other Discontinuities," *Proceedings of the IMA Conference on Numerical Methods in Aeronautical Fluid Dynamics*, pp. 211–243. Academic Press, New York, 1982. Reading, 1981.

[27] Lombard, G. K. and Oliger, J. and Yang, J. Y., "A Natural Conservative Flux Difference Splitting for the Hyperbolic Equations of Gas Dynamics," AIAA Paper 82–0976, 1982.

[28] Engquist, B. and Osher, S., "One-Sided Difference Approximations for Nonlinear Conservation Laws," *Mathematics of Computation*, 36:321–351, 1981.

[29] Lax, P. D., "Weak Solutions of Nonlinear Hyperbolic Equations and Their Numerical Computation," *Communications of Pure and Applied Mathematics*, 7:159–193, 1954.

[30] Lax, P. D. and Richtmyer, R. E., "Survey of the Stability of Linear Finite Difference Equations," *Communications of Pure and Applied Mathematics*, 9:267–293, 1956.

[31] Lax, P. D. and Wendroff, B., "Systems of Conservation Laws," *Communications of Pure and Applied Mathematics*, 13:217–237, 1960.

[32] Jameson, A. and Schmidt, W. and Turkel, E., "Numerical Solutions of the Euler Equations by Finite-Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81–1259, June 1981.

[33] Jameson, A. and Baker, T. J., "Solution of the Euler Equations for Complex Configurations," AIAA Paper 83–1929, 1983.

[34] Jameson, A., "Multigrid Algorithms for Compressible Flow Calculations," Technical Report MAE Report 1743, Princeton University, Department of Mechanical and Aerospace Engineering, October, 4, 1985. Text of lecture given at 2nd European Conference on Multigrid Methods, Cologne.

[35] von Lavante, E. and Elmiligui, A. and Cannizzaro, F. E. and Warda, H., "Simple Explicit Upwind Schemes for Solving Compressible Flows," *Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics, NNFM*, volume 29, pp. 291–301, 1989.

[36] van Leer, B. and Tai, C. H. and Powell, K., "Design of Optimally Smoothing Multi-Stage Schemes for the Euler Equations," AIAA Paper 89–1933–CP, 1989.

[37] Blazek, J. and Kroll, J. and Radespiel, R. and Rossow, C. C., "Upwind Implicit Residual Smoothing Method for Multi-Stage Schemes," AIAA Paper 91–1533–CP, 1991.

[38] Fedorenko, R. P., "The Speed of Convergence of One Iteration Process," *USSR Comp. Math. and Math. Physics*, 4:227–235, 1964. (In Russian.)

[39] Brandt, A., "Multi-Level Adaptive Solutions to Boundary Value Problems," *Mathematics of Computations*, 31(138):333–390, 1977.

[40] South, Jr., J. C. and Brandt, A., "Application of a Multilevel Grid Method to Transonic Flow Calculations," *Transonic Flow Calculations in Turbomachinery*, T. C. Adamson and M. C. Platzer, eds., Hemisphere Publications, 1977.

[41] Ni, R. H., "A Multiple Grid Scheme for Solving the Euler Equations," *AIAA Journal*, 20:1565–1571, 1982.

[42] Jameson, A., "Solution of the Euler Equations by a Multigrid Method," *Applied Mathematics and Computation*, 13:327–356, 1983.

[43] NASA Conference Publication 2166. *Numerical Grid Generation Technique*, October 1980. Workshop held at NASA Langley Research Center, Hampton, VA.

[44] Thompson, J. F. and Warsi, Z. U. A., "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations: A Review," *Journal of Computational Physics*, 47(1):1–108, 1982.

[45] Thompson, J. F., ed. *Numerical Grid Generation.* North-Holland, New York, NY, 1982. Proceedings of a Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Their Use in the Numerical Solution of Partial Differential Equations, Apr. 1982.

[46] Ghia, K. N. and Ghia, U., eds. *Advances in Grid Generation*, Vol. 5. Applied Mechanics, Bioengineering and Fluids Engineering Conference, sponsored by the Fluids Engineering Division of ASME, June 1983.

[47] Boppe, J. "Calculation of Transonic Wing Flows by Grid Embedding," AIAA Paper 77–207, 1977.

[48] Hedman, S. G., "An Application of the Embedded Grid Technique to the Calculation of Transonic Flow Past Wings," Technical Report, The Aeronautical Research Institute of Sweden, Technical Note AU-1600, Stockholm, 1980.

[49] Thompson, D. S., "A Mesh Embedding Approach for Prediction of Transonic Wing/Body/Store Flow Fields," *Proceedings of the Symposium on Numerical Boundary Condition Procedures*, pp. 15–39. Supplementary Report to NASA Conference Publication 2201, October 1981.

[50] Atta, E. H. and Vadyak, J., "A Grid Interfacing Zonal Algorithm for Three-Dimensional Transonic Flows About Aircraft Configurations," AIAA Paper 82–1017, June 1982.

[51] Atta, E., "Component-Adaptive Grid Interfacing," AIAA Paper 81–0382, Jan. 1981.

[52] Benek, J. A., Steger, J. L., and Dougherty, F. C., "A Flexible Grid Embedding Technique with Application to the Euler Equations," AIAA Paper 83–1944, July 1983.

[53] Benek, J. A., Buning, P. G., and Steger, J. L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85–1523–CP, July 1985.

[54] Eberhardt, S. and Baganoff, D., "Overset Grids in Compressible Flow," AIAA Paper 85–1524, 1985.

[55] Cambier, L., Ghazzi, W., Veuillot, J. P., and Viviand, H., "Une approche par domaines pur le calcul d'ecoulements compressibles," Cinquieme Colloque International sur les Methodes de Calcul Scientifique et Technique, INRIA, Versailles, France, Dec. 14–18, 1981.

[56] Rai, M. M., Chakravarthy, S. R., and Hessenius, K. A., "Zonal Grid Calculations Using the Osher Scheme," *International Journal of Computers and Fluids*, 12(3):161–173, 1984.

[57] Rai, M. M., "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations," AIAA Paper 84–0164, Jan. 1984.

[58] Mastin, C. W. and McConnaughey, H. V., "Computational Problems on Composite Grids," AIAA Paper 84–1611, June 25–27, 1984.

[59] Rai, M. M., "A Conservative Treatment of Zonal Boundaries for Euler Equations Calculations," *Journal of Computational Physics*, 62:472–503, Feb. 1986. 2

[60] Walters, R. W., Thomas, J. L., and Switzer, G. F., "Aspects and Applications of Patched Grid Calculations," AIAA Paper 86–1063, May 12–14, 1986.

[61] Kathong, M., Smith, R. E., and Tiwari, S. N., "A Conservative Approach for Flow Field Calculation on Multiple Grids," AIAA Paper 88–0224, Jan. 11–14, 1988.

[62] Kathong, M., Smith, R. E., and Tiwari, S. N., "Application of Multiple Grids Topology to Supersonic Internal/External Flow Interactions," AIAA Paper 88–0224, July 25–28, 1988.

[63] Thomas, J. L., Rudy, D. H., Chakravarthy, S. R., and Walters, R. W., "Patched-Grid Computations of High-Speed Inlet Flows," Symposium on Advances and Applications in Computational Fluid Dynamics, Winter Annual Meeting of ASME, Chicago, IL, Nov. 28–Dec. 2, 1988.

[64] Thomas, J. L., Walters, R. W., Reu, T., and Ghaffari, F. A. "Patched-Grid Algorithm for Complex Configurations Directed Towards the F/A-18 Aircraft," AIAA Paper 89–0121, Jan. 9–12, 1989.

[65] Biedron, R. T. and Thomas, J. L., "A Generalized Patched-Grid Algorithm with Application to the F-18 Forebody with Actuated Control Strake," *Computing Systems in Engineering*, 1(2–4):563–576, 1990.

[66] Kassies, A and Tognaccini, R., "Boundary Conditions for Euler Equations at Internal Block Faces of Multi-Block Domains Using Local Grid Refinement," AIAA Paper 90–1590, June 18–20, 1990.

[67] Cannizzaro, F. E., Elmiligui, A. Melson, N. D., and von Lavante, E. A., "Multiblock Multigrid Method for the Solution of the Three-Dimensional Euler Equations," AIAA Paper 90–0105, 1990.

[68] Yadlin, Y. and Caughey, D. A., "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow," *AIAA Journal*, 29(5):712–719, Mar. 1990.

[69] Rossow, C. C., "Efficient Computation of Inviscid Flow Fields Around Complex Configurations Using a Multi-Block Multigrid Method," Fifth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, Mar. 31–Apr. 5, 1991.

[70] Melson, N. D., Cannizzaro, F. E., and Elmiligui, A., "A Multigrid Multiblock Programming Strategy," Fifth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, Mar. 31–Apr. 5, 1991.

[71] Bonhaus, D. L. and Wornom, S. F., "Comparison of Two Navier-Stokes Codes for Attached Transonic Wing Flows," *Journal of Aircraft*, 29(1):101–107, Jan.-Feb. 1992.

[72] Baldwin, B. S. and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78–257, Jan. 16–18, 1978.

[73] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2. John Wiley & Sons, 1990. Computational Methods for Inviscid and Viscous Flows.

[74] Hinze, J. O., *"Turbulence.* McGraw-Hill Book Company, Inc., 1987.

[75] Anderson, D. A., Tannehill, J. C., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer.* McGraw-Hill Book Company, Inc., 1984.

[76] Rumsey, C. L., *Development of a Grid-Independent Approximate Riemann Solver.* Ph.D. thesis, University of Michigan, 1991.

[77] Anderson, W. K., *Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations.* Ph.D. thesis, Mississippi State, 1986.

[78] Anderson, W. K., Thomas, J. L, and van Leer, B., "A Comparison of Finite Volume Flux Vector Splittings for the Euler Equations," AIAA Paper 85–0122, Jan. 14–17, 1985.

[79] von Lavante, E. and Haertl, A., "Numerical Solutions of Euler Equations Using Simplified Flux Vector Splitting," AIAA Paper 85–1333, 1985.

[80] Melson, N. D. and von Lavante, E., "Multigrid Acceleration of the Isenthalpic Form of the Compressible Flow Equations," Third Copper Mountain Conference on Multigrid Methods, Apr. 6–10, 1987.

[81] Cannizzaro, F. E., von Lavante, E., and Melson, N. D., "Calculations of Three-Dimensional Flows Using the Isenthalpic Euler Equations with Implicit Flux-Vector Splitting,"*Proceedings of the AIAA 6th Applied Aerodynamics Conference*, pp. 593–614. AIAA Paper 88–2516, 1988.

[82] Roe, R. L., "Characteristic Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, 18:337–365, 1986.

[83] Vatsa, V. N., Thomas, J. L., and Wedan, B. W., "Navier-Stokes Computations of a Prolate Spheroid at Angle of Attack," *Journal of Aircraft*, 26(11):986–993, 1989.

[84] Fromm, J. E., "A Method for Reducing Dispersion in Convective Difference Schemes," *Journal of Computational Physics*, 3:176–198, 1968.

[85] Pulliam, T. H. and Chaussee, D. S., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," *Journal of Computational Physics*, 39:347–363, 1981.

[86] Vatsa, V. N. and Wedan, B. W., "Development of a Multigrid Code for 3–D Navier-Stokes Equations and its Application to a Grid-Refinement Study," *Computers & Fluids*, 18(4):391–403, 1990.

[87] Swanson, R. C. and Turkel, E., "A Multistage Time-Stepping Scheme for the Navier-Stokes Equations," Technical Report, ICASE Report 84–62, Feb. 1985.

[88] Swanson, R. C. and Turkel, E., "Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations, "AIAA Paper 87–1107–CP, June 9–11, 1987.

[89] Swanson, R. C. and Radespiel, R., "Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations," *AIAA Journal*, 29(5):697–703, May 1991.

[90] Elmiligui, A., Ph.D. thesis, Old Dominion University, 1992.

[91] Turkel, E., Swanson, R. C., Vatsa, V. N., and White, J. A., "Multigrid for Hypersonic Viscous Two- and Three-Dimensional Flows," AIAA Paper 91–1572, June 24–26, 1991.

[92] Vatsa, V. N., Turkel, E., and Abolhassani, J. S., "Extension of Multigrid Methodology to Supersonic/Hypersonic 3–D Viscous Flows," Fifth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, Mar. 31–Apr. 5, 1991.

[93] Swanson, R. C., Turkel, E., and White, J. A., "An Effective Multigrid Method for High-Speed Flows," Technical Report, ICASE Report 91–56, July 1991.

[94] Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method," AIAA Paper 79–1458, July 1979.

[95] Sankar, N. L., "A Multigrid Strongly Implicit Procedure for Two-Dimensional Transonic Potential Flow Problems," AIAA Paper 82–0931, 1982.

[96] Thames, F. C., "Multigrid Applications to Three-Dimensional Elliptic Coordinate Generation," *Applied Mathematics and Computation*, 15(4):325–342, 1984.

[97] Melson, N. D., "Vectorizable multigrid algorithms for transonic flow calculations," Master's thesis, The George Washington University, 1985.

[98] Marconi, F., "Internal Corner Flow Fields," AIAA Paper 79–0014, 1979.

[99] Kutler, P., "Numerical Solution for the Inviscid Supersonic Flow in the Corner Formed by Two Intersecting Wedges," AIAA Paper 73–675, July 16–18, 1973.

[100] West, J. E. and Korkegi, R. H., "Supersonic Interaction in the Corner of Intersecting Wedges at High Reynolds Numbers," *AIAA Journal*, 10(5): 652–656, May 1972.

[101] Salas, M. D., "The Numerical Calculation of Inviscid Plume Flow Fields," AIAA Paper 74–523, June 17–19, 1974.

[102] Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," No. 138, pp. B2–1—B2–61. AGARD, May 1979.

[103] Thomas, J. L., Krist, S. T., and Anderson, W. K., "Navier-Stokes Computations of Vortical Flows Over Low-Aspect-Ratio Wings," *AIAA Journal*, 28(2):205–212, February 1990.

[104] Putnam, L. E. and Mercer, C. E., "Pitot-Pressure Measurements in Flow Fields Behind a Rectangular Nozzle with Exhaust Jet for Free-Stream Mach Numbers of 0.00, 0.60, and 1.20," Technical Report, NASA, Nov. 1986. NASA Technical Memorandum 88990.

[105] Compton, III, William B., Thomas, J. L., Abeyounis, W. K., and Mason, M. L., "Transonic Navier-Stokes Solutions of Three-Dimensional Afterbody Flows," Technical Report Technical Memorandum 4111, NASA, July 1989.

[106] Morrison, J., "Flux Difference Split Scheme for Turbulent Transport Equations," AIAA Paper 90–5251, Oct. 29–31, 1990.

Appendix A Full Navier-Stokes Equations in Body-Fitted Coordinates

For the ξ–direction

$$F_v(1) = [0.0] \tag{A.1}$$

$$F_v(2) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\left(\frac{4}{3}\xi_x^2 + \xi_y^2 + \xi_z^2\right) + u_\eta\left(\frac{4}{3}\eta_x\xi_x + \eta_y\xi_y + \eta_z\xi_z\right) + \\ u_\zeta\left(\frac{4}{3}\zeta_x\xi_x + \zeta_y\xi_y + \zeta_z\xi_z\right) + \\ v_\xi\frac{1}{3}\xi_y\xi_x + v_\eta\left(\eta_x\xi_y - \frac{2}{3}\eta_y\xi_x\right) + v_\zeta\left(\zeta_x\xi_y - \frac{2}{3}\zeta_y\xi_x\right) + \\ w_\xi\frac{1}{3}\xi_z\xi_x + w_\eta\left(\eta_x\xi_z - \frac{2}{3}\eta_z\xi_x\right) + w_\zeta\left(\zeta_x\xi_z - \frac{2}{3}\zeta_z\xi_x\right) \end{bmatrix} \tag{A.2}$$

$$F_v(3) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\frac{1}{3}\xi_x\xi_y + u_\eta\left(\eta_y\xi_x - \frac{2}{3}\eta_x\xi_y\right) + u_\zeta\left(\zeta_y\xi_x - \frac{2}{3}\zeta_x\xi_y\right) + \\ v_\xi\left(\xi_x^2 + \frac{4}{3}\xi_y^2 + \xi_z^2\right) + v_\eta\left(\eta_x\xi_x + \frac{4}{3}\eta_y\xi_y + \eta_z\xi_z\right) + \\ v_\zeta\left(\zeta_x\xi_x + \frac{4}{3}\zeta_y\xi_y + \zeta_z\xi_z\right) + \\ w_\xi\frac{1}{3}\xi_z\xi_y + w_\eta\left(\eta_y\xi_z - \frac{2}{3}\eta_z\xi_y\right) + w_\zeta\left(\zeta_y\xi_z - \frac{2}{3}\zeta_z\xi_y\right) \end{bmatrix} \tag{A.3}$$

$$F_v(4) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\frac{1}{3}\xi_x\xi_z + u_\eta\left(\eta_z\xi_x - \frac{2}{3}\eta_x\xi_z\right) + u_\zeta\left(\zeta_z\xi_x - \frac{2}{3}\zeta_x\xi_z\right) + \\ v_\xi\frac{1}{3}\xi_y\xi_z + v_\eta\left(\eta_z\xi_y - \frac{2}{3}\eta_y\xi_z\right) + v_\zeta\left(\zeta_z\xi_y - \frac{2}{3}\zeta_y\xi_z\right) + \\ w_\xi\left(\xi_x^2 + \xi_y^2 + \frac{4}{3}\xi_z^2\right) + w_\eta\left(\eta_x\xi_x + \eta_y\xi_y + \frac{4}{3}\eta_z\xi_z\right) + \\ w_\zeta\left(\zeta_x\xi_x + \zeta_y\xi_y + \frac{4}{3}\zeta_z\xi_z\right) \end{bmatrix} \tag{A.4}$$

$$F_v(5) = \frac{M_{ref}\mu}{R_{ref}} \left[\begin{array}{l} u_\xi\left[u\left(\frac{1}{3}\xi_x^2 + \phi^2\right) + \frac{1}{3}v\xi_y\xi_x + \frac{1}{3}w\xi_z\xi_x\right] + \\ u_\eta\left[\begin{array}{r} u\left(\frac{4}{3}\eta_x\xi_x + \eta_y\xi_y + \eta_z\xi_z\right) + v\left(\eta_y\xi_x - \frac{2}{3}\eta_x\xi_y\right) + \\ w\left(\eta_z\xi_x - \frac{2}{3}\eta_x\xi_z\right) \end{array} \right] + \\ u_\zeta\left[\begin{array}{r} u\left(\frac{4}{3}\zeta_x\xi_x + \zeta_y\xi_y + \zeta_z\xi_z\right) + v\left(\zeta_y\xi_x - \frac{2}{3}\zeta_x\xi_y\right) + \\ w\left(\zeta_z\xi_x - \frac{2}{3}\zeta_x\xi_z\right) \end{array} \right] + \\ v_\xi\left[\frac{1}{3}u\xi_x\xi_y + v\left(\frac{1}{3}\xi_y^2 + \phi^2\right) + \frac{1}{3}w\xi_z\xi_y\right] + \\ v_\eta\left[\begin{array}{r} u\left(\eta_x\xi_y - \frac{2}{3}\eta_y\xi_x\right) + v\left(\eta_x\xi_x + \frac{4}{3}\eta_y\xi_y + \eta_z\xi_z\right) + \\ w\left(\eta_z\xi_y - \frac{2}{3}\eta_y\xi_z\right) \end{array} \right] + \\ v_\zeta\left[\begin{array}{r} u\left(\zeta_x\xi_y - \frac{2}{3}\zeta_y\xi_x\right) + v\left(\zeta_x\xi_x + \frac{4}{3}\zeta_y\xi_y + \zeta_z\xi_z\right) + \\ w\left(\zeta_z\xi_y - \frac{2}{3}\zeta_y\xi_z\right) \end{array} \right] + \\ w_\xi\left[\frac{1}{3}u\xi_x\xi_z + \frac{1}{3}v\xi_y\xi_z + w\left(\frac{1}{3}\xi_z^2 + \phi^2\right)\right] + \\ w_\eta\left[\begin{array}{r} u\left(\eta_x\xi_z - \frac{2}{3}\eta_z\xi_x\right) + v\left(\eta_y\xi_z - \frac{2}{3}\eta_z\xi_y\right) + \\ w\left(\eta_x\xi_x + \eta_y\xi_y + \frac{4}{3}\eta_z\xi_z\right) \end{array} \right] + \\ w_\zeta\left[\begin{array}{r} u\left(\zeta_x\xi_z - \frac{2}{3}\zeta_z\xi_x\right) + v\left(\zeta_y\xi_z - \frac{2}{3}\zeta_z\xi_y\right) + \\ w\left(\zeta_x\xi_x + \zeta_y\xi_y + \frac{4}{3}\zeta_z\xi_z\right) \end{array} \right] + \\ \sigma[T_\xi\phi^2 + T_\eta(\eta_x\xi_x + \eta_y\xi_y + \eta_z\xi_z) + \\ T_\zeta(\zeta_x\xi_x + \zeta_y\xi_y + \zeta_z\xi_z)] \end{array} \right] \tag{A.5}$$

For the η–direction

$$G_v(1) = [0.0] \tag{A.6}$$

$$G_v(2) = \frac{M_{ref}\mu}{R_{ref}} \left[\begin{array}{r} u_\xi\left(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z\right) + u_\eta\left(\frac{4}{3}\eta_x^2 + \eta_y^2 + \eta_z^2\right) + \\ u_\zeta\left(\frac{4}{3}\zeta_x\eta_x + \zeta_y\eta_y + \zeta_z\eta_z\right) + \\ v_\xi\left(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x\right) + v_\eta\frac{1}{3}\eta_x\eta_y + v_\zeta\left(\zeta_x\eta_y - \frac{2}{3}\zeta_y\eta_x\right) + \\ w_\xi\left(\xi_x\eta_z - \frac{2}{3}\xi_z\eta_x\right) + w_\eta\frac{1}{3}\eta_x\eta_z + w_\zeta\left(\zeta_x\eta_z - \frac{2}{3}\zeta_z\eta_x\right) \end{array} \right] \tag{A.7}$$

$$G_v(3) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y) + u_\eta\frac{1}{3}\eta_y\eta_x + u_\zeta(\zeta_y\eta_x - \frac{2}{3}\zeta_x\eta_y) + \\ v_\xi(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y + \xi_z\eta_z) + v_\eta(\eta_x^2 + \frac{4}{3}\eta_y^2 + \eta_z^2) + \\ v_\zeta(\zeta_x\eta_x + \frac{4}{3}\zeta_y\eta_y + \zeta_z\eta_z) + \\ w_\xi(\xi_y\eta_z - \frac{2}{3}\xi_z\eta_y) + w_\eta\frac{1}{3}\eta_y\eta_z + w_\zeta(\zeta_y\eta_z - \frac{2}{3}\zeta_z\eta_y) \end{bmatrix} \tag{A.8}$$

$$G_v(4) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi(\xi_z\eta_x - \frac{2}{3}\xi_x\eta_z) + u_\eta\frac{1}{3}\eta_z\eta_x + u_\zeta(\zeta_z\eta_x - \frac{2}{3}\zeta_x\eta_z) + \\ v_\xi(\xi_z\eta_y - \frac{2}{3}\xi_y\eta_z) + v_\eta\frac{1}{3}\eta_z\eta_y + v_\zeta(\zeta_z\eta_y - \frac{2}{3}\zeta_y\eta_z) + \\ w_\xi(\xi_x\eta_x + \xi_y\eta_y + \frac{4}{3}\xi_z\eta_z) + w_\eta(\eta_x^2 + \eta_y^2 + \frac{4}{3}\eta_z^2) + \\ w_\zeta(\zeta_x\eta_x + \zeta_y\eta_y + \frac{4}{3}\zeta_z\eta_z) \end{bmatrix} \tag{A.9}$$

$$G_v(5) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\begin{bmatrix} u(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z) + v(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y) + \\ w(\xi_z\eta_x - \frac{2}{3}\xi_x\eta_z) \end{bmatrix} + \\ u_\eta\left[u(\frac{1}{3}\eta_x^2 + \theta^2) + \frac{1}{3}v\eta_y\eta_x + \frac{1}{3}w\eta_z\eta_x\right] + \\ u_\zeta\begin{bmatrix} u(\frac{4}{3}\zeta_x\eta_x + \zeta_y\eta_y + \zeta_z\eta_z) + v(\zeta_y\eta_x - \frac{2}{3}\zeta_x\eta_y) + \\ w(\zeta_z\eta_x - \frac{2}{3}\zeta_x\eta_z) \end{bmatrix} + \\ v_\xi\begin{bmatrix} u(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x) + v(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y + \xi_z\eta_z) + \\ w(\xi_z\eta_y - \frac{2}{3}\xi_y\eta_z) \end{bmatrix} + \\ v_\eta\left[\frac{1}{3}u\eta_y\eta_x + v(\frac{1}{3}\eta_y^2 + \theta^2) + \frac{1}{3}w\eta_y\eta_z\right] + \\ v_\zeta\begin{bmatrix} u(\zeta_x\eta_y - \frac{2}{3}\zeta_y\eta_x) + v(\zeta_x\eta_x + \frac{4}{3}\zeta_y\eta_y + \zeta_z\eta_z) + \\ w(\zeta_z\eta_y - \frac{2}{3}\zeta_y\eta_z) \end{bmatrix} + \\ w_\xi\begin{bmatrix} u(\xi_x\eta_z - \frac{2}{3}\xi_z\eta_x) + v(\xi_y\eta_z - \frac{2}{3}\xi_z\eta_y) + \\ w(\xi_x\eta_x + \xi_y\eta_y + \frac{4}{3}\xi_z\eta_z) \end{bmatrix} + \\ w_\eta\left[\frac{1}{3}u\eta_z\eta_x + \frac{1}{3}v\eta_z\eta_y + w(\frac{1}{3}\eta_z^2 + \theta^2)\right] + \\ w_\zeta\begin{bmatrix} u(\zeta_x\eta_z - \frac{2}{3}\zeta_z\eta_x) + v(\zeta_y\eta_z - \frac{2}{3}\zeta_z\eta_y) + \\ w(\zeta_x\eta_x + \zeta_y\eta_y + \frac{4}{3}\zeta_z\eta_z) \end{bmatrix} + \\ \sigma[T_\xi(\xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z) + T_\eta\theta^2 + \\ T_\zeta(\zeta_x\eta_x + \zeta_y\eta_y + \zeta_z\eta_z)] \end{bmatrix} \tag{A.10}$$

For the ζ–direction

$$H_v(1) = [0.0] \tag{A.11}$$

$$H_v(2) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\left(\frac{4}{3}\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z\right) + u_\eta\left(\frac{4}{3}\eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z\right) + \\ u_\zeta\left(\frac{4}{3}\zeta_x^2 + \zeta_y^2 + \zeta_z^2\right) + \\ v_\xi\left(\xi_x\zeta_y - \frac{2}{3}\xi_y\zeta_x\right) + v_\eta\left(\eta_x\zeta_y - \frac{2}{3}\eta_y\zeta_x\right) + v_\zeta\frac{1}{3}\zeta_y\zeta_x + \\ w_\xi\left(\xi_x\zeta_z - \frac{2}{3}\xi_z\zeta_x\right) + w_\eta\left(\eta_x\zeta_z - \frac{2}{3}\eta_z\zeta_x\right) + w_\zeta\frac{1}{3}\zeta_z\zeta_x \end{bmatrix} \tag{A.12}$$

$$H_v(3) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\left(\xi_y\zeta_x - \frac{2}{3}\xi_x\zeta_y\right) + u_\eta\left(\eta_y\zeta_x - \frac{2}{3}\eta_x\zeta_y\right) + u_\zeta\frac{1}{3}\zeta_x\zeta_y + \\ v_\xi\left(\xi_x\zeta_x + \frac{4}{3}\xi_y\zeta_y + \xi_z\zeta_z\right) + v_\eta\left(\eta_x\zeta_x + \frac{4}{3}\eta_y\zeta_y + \eta_z\zeta_z\right) + \\ v_\zeta\left(\zeta_x^2 + \frac{4}{3}\zeta_y^2 + \zeta_z^2\right) + \\ w_\xi\left(\xi_y\zeta_z - \frac{2}{3}\xi_z\zeta_x\right) + w_\eta\left(\eta_y\zeta_z - \frac{2}{3}\eta_z\zeta_y\right) + w_\zeta\frac{1}{3}\zeta_z\zeta_y \end{bmatrix} \tag{A.13}$$

$$H_v(4) = \frac{M_{ref}\mu}{R_{ref}} \begin{bmatrix} u_\xi\left(\xi_z\zeta_x - \frac{2}{3}\xi_x\zeta_z\right) + u_\eta\left(\eta_z\zeta_x - \frac{2}{3}\eta_x\zeta_z\right) + u_\zeta\frac{1}{3}\zeta_z\zeta_x + \\ v_\xi\left(\xi_z\zeta_y - \frac{2}{3}\xi_y\zeta_z\right) + v_\eta\left(\eta_z\zeta_y - \frac{2}{3}\eta_y\zeta_z\right) + v_\zeta\frac{1}{3}\zeta_z\zeta_y + \\ w_\xi\left(\xi_x\zeta_x + \xi_y\zeta_y + \frac{4}{3}\xi_z\zeta_z\right) + w_\eta\left(\eta_x\zeta_x + \eta_y\zeta_y + \frac{4}{3}\eta_z\zeta_z\right) + \\ w_\zeta\left(\zeta_x^2 + \zeta_y^2 + \frac{4}{3}\zeta_z^2\right) \end{bmatrix} \tag{A.14}$$

$$H_v(5) = \frac{M_{ref}\mu}{R_{ref}} \left[\begin{array}{l} u_\xi \left[\begin{array}{r} u\left(\frac{4}{3}\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z\right) + v\left(\xi_y\zeta_x - \frac{2}{3}\xi_x\zeta_y\right) + \\ w\left(\xi_z\zeta_x - \frac{2}{3}\xi_x\zeta_z\right) \end{array} \right] + \\ u_\eta \left[\begin{array}{r} u\left(\frac{4}{3}\eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z\right) + v\left(\eta_y\zeta_x - \frac{2}{3}\eta_x\zeta_y\right) + \\ w\left(\eta_z\zeta_x - \frac{2}{3}\eta_x\zeta_z\right) \end{array} \right] + \\ u_\zeta \left[u\left(\frac{1}{3}\zeta_x^2 + \varphi^2\right) + \frac{1}{3}v\zeta_y\zeta_x + \frac{1}{3}w\zeta_z\zeta_x \right] + \\ v_\xi \left[\begin{array}{r} u\left(\xi_x\zeta_y - \frac{2}{3}\xi_y\zeta_x\right) + v\left(\xi_x\zeta_x + \frac{4}{3}\xi_y\zeta_y + \xi_z\zeta_z\right) + \\ w\left(\xi_z\zeta_y - \frac{2}{3}\xi_y\zeta_z\right) \end{array} \right] + \\ v_\eta \left[\begin{array}{r} u\left(\eta_x\zeta_y - \frac{2}{3}\eta_y\zeta_x\right) + v\left(\eta_x\zeta_x + \frac{4}{3}\eta_y\zeta_y + \eta_z\zeta_z\right) + \\ w\left(\eta_z\zeta_y - \frac{2}{3}\eta_y\zeta_z\right) \end{array} \right] + \\ v_\zeta \left[\frac{1}{3}u\zeta_y\zeta_x + v\left(\frac{1}{3}\zeta_y^2 + \varphi^2\right) + \frac{1}{3}w\zeta_z\zeta_y \right] + \\ w_\xi \left[\begin{array}{r} u\left(\xi_x\zeta_z - \frac{2}{3}\xi_z\zeta_x\right) + v\left(\xi_y\zeta_z - \frac{2}{3}\xi_z\zeta_y\right) + \\ w\left(\xi_x\zeta_x + \xi_y\zeta_y + \frac{4}{3}\xi_z\zeta_z\right) \end{array} \right] + \\ w_\eta \left[\begin{array}{r} u\left(\eta_x\zeta_z - \frac{2}{3}\eta_z\zeta_x\right) + v\left(\eta_y\zeta_z - \frac{2}{3}\eta_z\zeta_x\right) + \\ w\left(\eta_x\zeta_x + \eta_y\zeta_y + \frac{4}{3}\eta_z\zeta_z\right) \end{array} \right] + \\ w_\zeta \left[\frac{1}{3}u\zeta_z\zeta_x + \frac{1}{3}v\zeta_z\zeta_y + w\left(\frac{1}{3}\zeta_z^2 + \varphi^2\right) \right] + \\ \\ \sigma[T_\xi(\xi_x\zeta_x + \xi_y\zeta_y + \xi_z\zeta_z) + T_\eta(\eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z) \\ \\ + T_\zeta\varphi^2] \end{array} \right] \tag{A.15}$$

Appendix B Multigrid Restriction and Prolongation Operations

B.1 Restriction Operation

The best way to explain the restriction procedure is by showing the operation for a two-dimensional case. Starting with the fine grid, which is comprised of the small cells whose average values are designated by the empty circles, the coarse grid is generated by removing every other fine grid line (see Fig. B.1). Thus, producing the larger cells which are designated by the thicker borders. The shaded octagons, centered in the larger cells, represent the coarser grid cells' centered values. As shown in Fig. B.1, a volume weighted averaging of the flow values from the fine grid cells is used to provide coarse grid values, which are to represent the solution of the fine grid on the coarser grid.

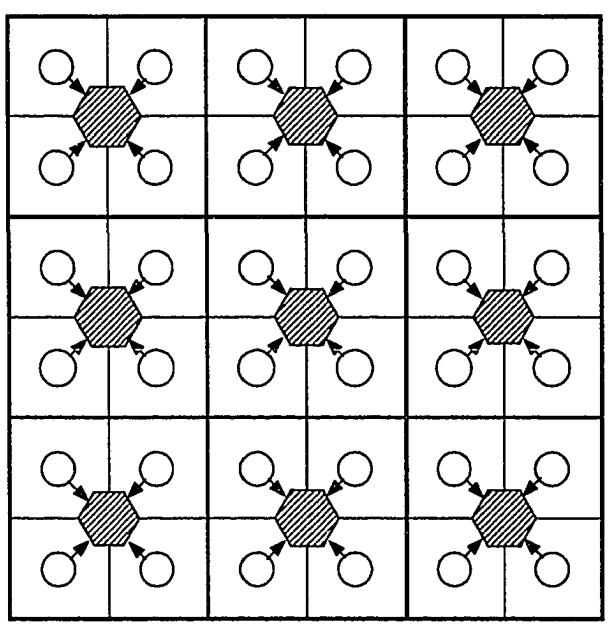


Figure B.1 Two-Dimensional Restriction Operation.

In three-dimensional space, the same type of averaging of the fine grid values is performed on the two fine grid planes that surround each coarse grid plane. To show this, it is necessary to indicate where the coarse grid and fine grid cell centers are located geometrically. This can be seen in Fig. B.2, which represents a three-dimensional volume. The fine grid is indicated by the thin lines, and the coarse grid by the thicker lines. The fine grid cell centers are at the centers of all the small cells, and the coarse grid cell centers are at the centers of the larger cells, indicated by the thicker lines. Cutting some of the cells away and putting in four planes, two representing fine grid cell centers and two representing coarse grid cell centers, shows the spacial relation of the coarse and fine grid cell centers, as can be seen in Fig. B.1.3. This figure identifies the various planes that are needed for both restriction and prolongation operations. Each coarse grid cell centered plane has a fine grid cell centered plane on both sides of it. Focusing just on these three planes, as shown in Fig. B.1.4, this averaging will involve eight fine grid cells to produce one coarse grid cell value. The volume weighted averaging uses the actual physical cell's volumes, and not those of the computational domain, which are unit volumes.

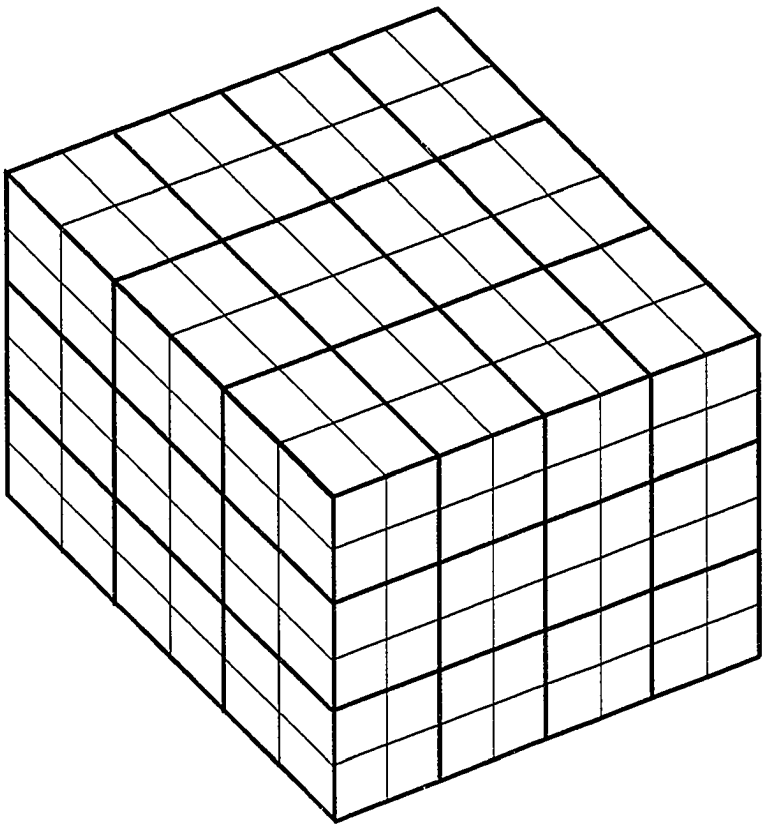


Figure B.2 Three-Dimensional Fine and Coarse Grid.

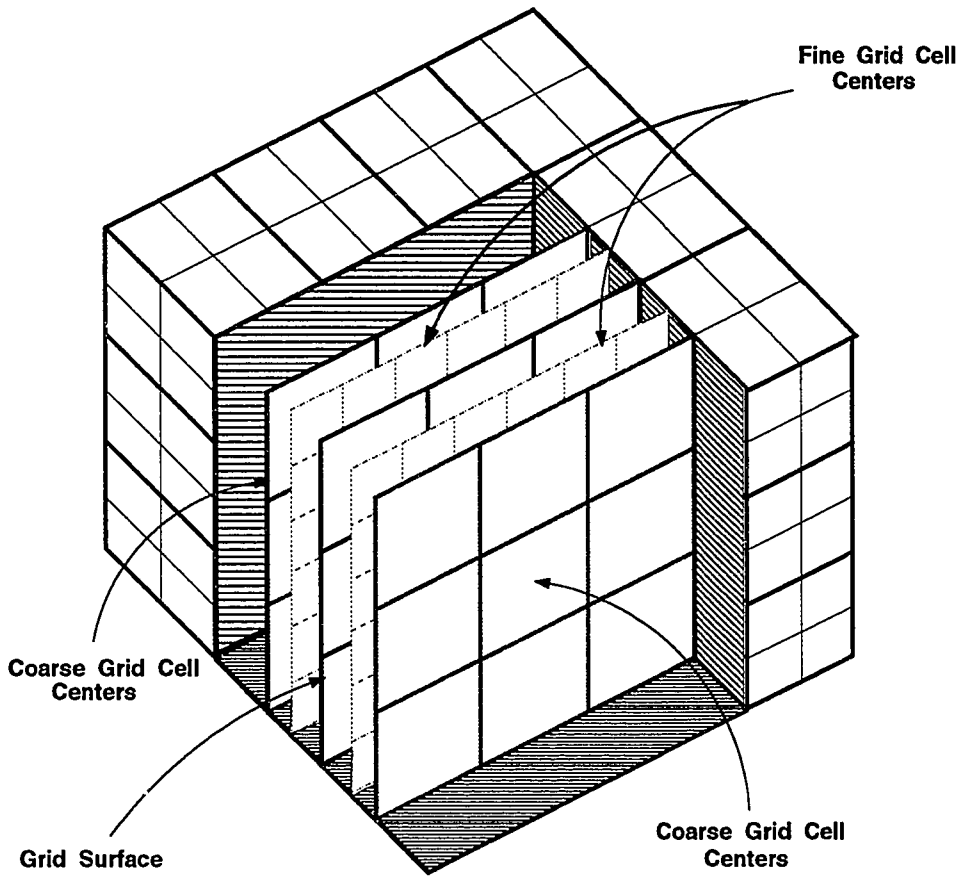


Figure B.1.3 Three-Dimensional Fine and Coarse Grid Cell Centers

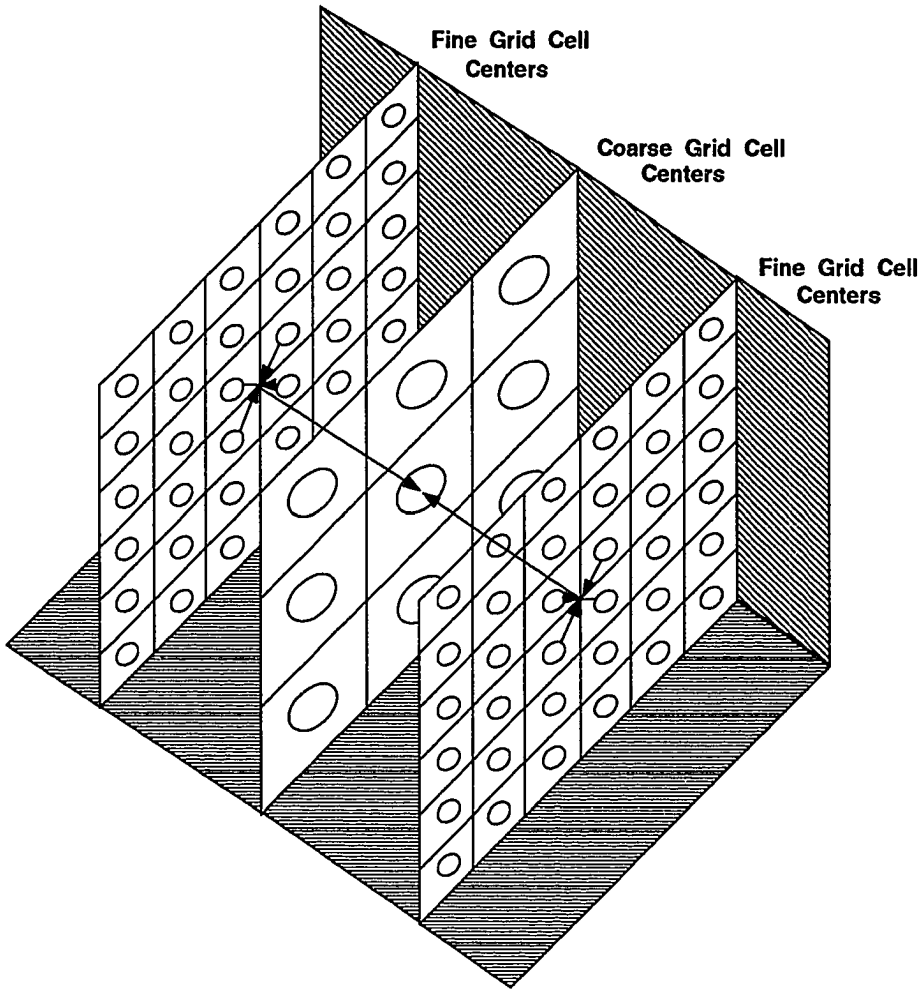


Figure B.1.4 Three-Dimensional Restriction Operation

B.2 Prolongation Operation

The prolongation operation is done in computational space. For the two-dimensional case a bilinear interpolation is performed among the coarse grid cells, which are indicated by the shaded octagons, shown in Fig. B.2.5. Each set of four shaded octagons has four empty circles within the octagons' perimeter. These circles represent the fine grid cells. The octagon that is closest to the circle has the largest weight factor, which gives it the most influence on the circle's value. How the prolongation values, for each circle in the perimeter, are obtained is indicated in the four different patterns shown in Fig. B.2.5.

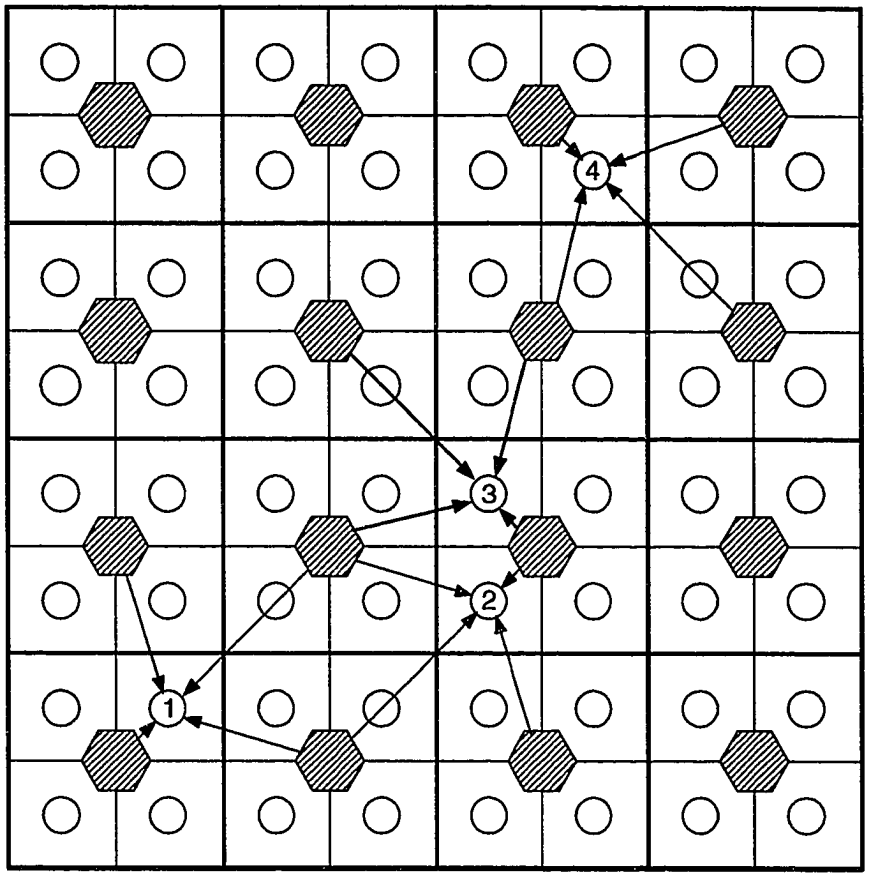


Figure B.2.5 Two-Dimensional Prolongation Operation.

In three-dimensional space, the prolongation operation is again done in the computational domain using a tri-linear interpolation. Referring back to Fig. B.1.3, the five planes are re-drawn in Fig. B.2.6, with circles added to indicate the cell centers. First a bilinear interpolation is performed on each coarse grid cell center plane, as shown in Fig. B.2.6. Then a linear interpolation between the two coarse grid prolongations is performed to give the value for the specified fine grid cell. As can be seen in Fig. B.2.6, there is a set of eight coarse grid cells that provide information to produce eight fine grid cells.

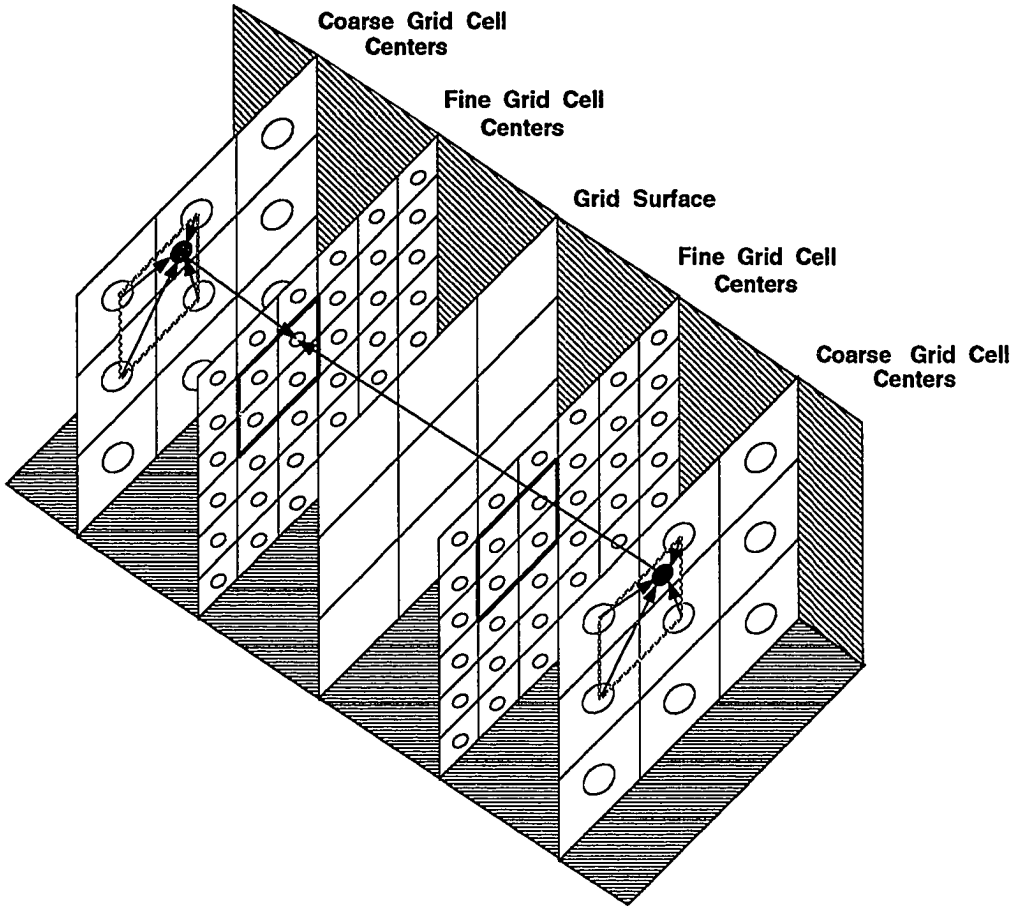


Figure B.2.6 Three-Dimensional Prolongation Operation.