ADAPTIVE GRID EMBEDDING FOR THE TWO-DIMENSIONAL FLUX-SPLIT EULER EQUATIONS

by

Gary Patrick Warren

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Aerospace Engineering

Mississippi State, Mississippi

May 1990

ADAPTIVE GRID EMBEDDING FOR THE TWO-DIMENSIONAL FLUX-SPLIT EULER EQUATIONS

by

Gary Patrick Warren

APPROVED:

Professor of Aerospace Engineering (Major Professor)	Director of Graduate Instruction College of Engineering
Associate Professor, Graduate Coordinator, Department of Aerospace Engineering	Dean of the College of Engineering Aerospace Engineering
Professor and Head of the Department of Aerospace Engineering	Vice President for Graduate Studies and Research

May, 1990

ACKNOWLEDGMENTS

The author wishes to thank the NASA Langley Research Center for their financial support for this project. Also, thanks go to Jim Thomas for helpful discussions and Jerry South for his patience. Special thanks to W. Kyle Anderson for always having the time to be a teacher and a friend.

ABSTRACT

Gary Patrick Warren, Master of Science, 1990

Major: Engineering, Department of Aerospace Engineering

Title of Thesis: Adaptive Grid Embedding for the Two-Dimensional Flux-Split Euler Equations

Directed by: Joe F. Thompson

Pages in Thesis: 65 Words in Abstract: 185

ABSTRACT

A numerical algorithm is presented for solving the two-dimensional flux-split Euler equations using a multigrid method with adaptive grid embedding. The method uses an unstructured data set along with a system of pointers for communication on the irregularly shaped grid topologies. An explicit two-stage time advancement scheme is implemented. A multigrid algorithm is used to provide grid level communication and to accelerate the convergence of the solution to steady state. Results are presented for a subcritical airfoil and a transonic airfoil with 3 levels of adaptation. Comparisons are made with a structured upwind Euler code which uses the same flux integration techniques of the present algorithm. Good agreement is obtained with converged surface pressure coefficients. The lift coefficients of the adaptive code are within $2\frac{1}{2}\%$ of the structured code for the subcritical case and within $4\frac{1}{2}\%$ of the structured code for the transonic case using approximately one-third the number of grid points.

TABLE OF CONTENTS

NOMENCLATURE

$\mathbf{A}, \mathbf{B}$	flux Jacobians, $\frac{\partial \mathbf{F}_1}{\partial \mathbf{Q}}$ and $\frac{\partial \mathbf{F}_2}{\partial \mathbf{Q}}$
a	speed of sound
CFL	Courant-Friedrichs-Lewy number
C_p	pressure coefficient
c	chord length
c_d	drag coefficient
c_ℓ	lift coefficient
e	total energy
$\mathbf{F}$	fluxes of mass, momentum, and energy
G_i	grid level i
$\mathbf{I}_i^{i-1}$	collection operator used for transferring information on grid level i to grid level $i-1$
$\mathbf{I}_{i-1}^{i}$	interpolation operator used for transferring information on grid level $i-1$ to grid level i
$\hat{\mathbf{I}}_i^{i-1}$	collection operator for residual
$\hat{i}, \hat{j}$	unit vectors in x and y directions
k_n	denotes ξ or η
ℓ	length of a face
M	Mach number
M_ξ	Mach number in ξ direction

$\hat{n}$	unit normal
$\hat{n}_x, \hat{n}_y$	x and y components of a unit normal
p	pressure
$\mathbf{Q}$	conserved state vector, $\mathbf{Q} = [\rho, \rho u, \rho v, e]^T$
$\mathbf{q}_i$	most current approximation to $\mathbf{Q}$ on grid level i
q	velocity magnitude
$\mathbf{R}$	residual vector for mass, momentum, and energy
S	surface area
s	entropy, also parameter for flux limiter
t	time
U, V	contravariant velocities
u, v	cartesian velocities in x and y directions
$\mathbf{V}_i$	correction at grid level i
x, y	cartesian coordinates
Δt	time step
α	angle–of–attack
β	refinement parameter
γ	ratio of specific heats, taken as 1.4
∇	gradient operator
ϕ	accuracy parameter
$\tilde{\nabla}$	undivided gradient operator
ξ, η	general curvilinear coordinates

κ	parameter controlling spatial difference–scheme type
ρ	density
τ	relative truncation error

Subscripts:

c	information at coarse grid face
$f1, f2$	information at fine grid faces
∞	denotes conditions at infinity

Superscripts:

n	denotes time level
$+, -$	denotes direction from which information was obtained

List of Figures

List of Tables

CHAPTER I

INTRODUCTION

In spite of many recent developments in computational fluid dynamics algorithms and substantial increases in computer speed and memory, many flow situations still impose unsatisfactory requirements on computer time. These requirements are driven by the large number of grid points required to sufficiently resolve important flowfield features. Since most algorithms use structured grids without embedding, adequate resolution of important flowfield features requires either highly stretched grids or the inclusion of many unnecessary grid points away from the main areas of interest.

As a solution to this problem, several adaptive techniques have been developed in recent years which attempt to concentrate grid points only in regions of high gradients. These techniques can be grouped into one of two categories. In the first category, higher resolution of specific areas is accomplished by redistributing the grid points of an existing structured grid into the regions of interest. The primary advantage of this method is its relative ease of implementation into existing flow solvers because no modifications to the data structure is required. Unfortunately, this technique often results in highly stretched and skewed grid lines and can leave some regions of the flowfield with inadequate resolution. Another method sometimes used to enhance grid resolution is the embedding of grid points into specific areas of an existing grid based on some reasonable criteria such as high flow gra-

dients or high local errors. This allows improved resolution of important features with a relatively small increase in grid points and without adversely affecting other portions of the grid. Grid embedding techniques, however, are difficult to implement into existing computer codes because of the necessary data structure required for inter-grid transfer of information. The difficulties encountered in grid embedding, however, are more than compensated by increased capability.

There have been several recent investigations into grid embedding using central-difference algorithms[1,2,3]. Berger[1] developed a grid embedding technique for a structured grid algorithm using a central-difference method developed by Jameson. This algorithm adapted to an estimate of the truncation error. More recently, Dannenhoffer[2,3] developed an adaptive grid embedding method that was used in conjunction with a central differenced node based scheme. Dannenhoffer's method used an unstructured grid along with a multigrid algorithm.

In recent years, upwind differencing has gained in popularity because of increased accuracy and robustness over its central differencing counterparts. The finite volume technique, as described in references 4 and 5, has been applied to a variety of flow problems. This scheme uses the flux-vector splitting of van Leer[6] or the flux-difference-splitting of Roe[7] coupled with an implicit, multigrid algorithm. This multigrid scheme is a well proven and documented algorithm which has been used mainly on block-structured grids with limited use on embedded grids[8].

In the present study, the same spatial differencing used in reference 4 is applied to a data structure designed for the use of an unstructured adaptive grid embed-

ding algorithm. Because of the unstructured grids, an implicit time advancing method is not easily implemented and so the equations are advanced in time with an explicit, two-stage time marching scheme. Since explicit time marching often results in poor convergence rates, a multigrid algorithm is employed. Use of the multigrid algorithm not only improves the convergence rate, but as importantly, it also provides a mechanism for inter-level grid communication. This thesis will present algorithmic details and results of this method.

CHAPTER II

GOVERNING EQUATIONS

2.1 Euler Equations

The governing equations are the time dependent Euler equations, which express the conservation of mass, momentum, and energy for an inviscid gas. The equations are given by

$$\frac{\partial \tilde{\mathbf{Q}}}{\partial \tilde{t}} + \nabla \cdot \vec{\mathbf{F}} = 0 \tag{2.1}$$

where the state vector $\tilde{\mathbf{Q}}$ and the flux vectors $\vec{\mathbf{F}}$ are given as

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \tilde{\rho} \\ \tilde{\rho}\tilde{u} \\ \tilde{\rho}\tilde{v} \\ \tilde{e} \end{bmatrix} \tag{2.2}$$

$$\vec{\mathbf{F}} = \tilde{\mathbf{F}}_1 \hat{\mathbf{i}} + \tilde{\mathbf{F}}_2 \hat{\mathbf{j}} \tag{2.3}$$

$$\tilde{\mathbf{F}}_1 = \begin{bmatrix} \tilde{\rho}\tilde{u} \\ \tilde{\rho}\tilde{u}^2 + \tilde{p} \\ \tilde{\rho}\tilde{u}\tilde{v} \\ (\tilde{e} + \tilde{p})\tilde{u} \end{bmatrix} \qquad \tilde{\mathbf{F}}_2 = \begin{bmatrix} \tilde{\rho}\tilde{v} \\ \tilde{\rho}\tilde{u}\tilde{v} \\ \tilde{\rho}\tilde{v}^2 + \tilde{p} \\ (\tilde{e} + \tilde{p})\tilde{v} \end{bmatrix} \tag{2.4}$$

The equations are closed with the equation of state for a perfect gas

$$\tilde{p} = (\gamma - 1)[\tilde{e} - \tilde{\rho}(\tilde{u}^2 + \tilde{v}^2)/2] \tag{2.5}$$

The variables in equations 2.1-2.4 can be non-dimensionalized by introduction of the following definitions:

$$\rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty}, \qquad v = \frac{\tilde{v}}{\tilde{a}_\infty}, \qquad x = \frac{\tilde{x}}{\tilde{l}}, \qquad t = \frac{\tilde{t}\tilde{a}_\infty}{\tilde{l}}$$
$$u = \frac{\tilde{u}}{\tilde{a}_\infty}, \qquad p = \frac{\tilde{p}}{\tilde{\rho}_\infty \tilde{a}_\infty^2}, \qquad y = \frac{\tilde{y}}{\tilde{l}}, \qquad a = \frac{\tilde{a}}{\tilde{a}_\infty} \tag{2.6}$$

The non-dimensional equations are not repeated here since introduction of the previous variables into equations (2.1)–(2.4) results in an identical form of equations as given above but with the (˜) removed.

2.2 Integral Formulation

The Euler equations can be integrated over an arbitrary control volume, as shown in figure 1, with the resulting equations given by

$$\int\int_S \frac{\partial \mathbf{Q}}{\partial t} dS + \int\int_S (\nabla \cdot \vec{\mathbf{F}}) dS = 0 \tag{2.7}$$

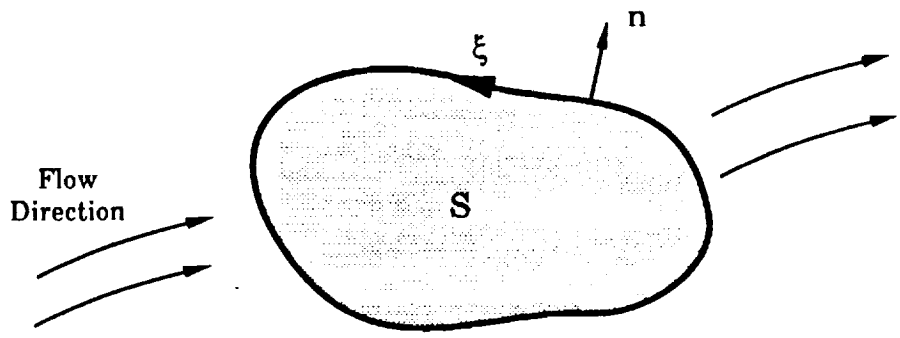


Figure 1. Arbitrary Control Volume

The first integral in equation (2.7) is then written as

$$\frac{\partial \hat{\mathbf{Q}}}{\partial t} = \int\int_S \frac{\partial \mathbf{Q}}{\partial t} dS \tag{2.8}$$

where $\hat{\mathbf{Q}}$ is given by

$$\hat{\mathbf{Q}} = S\bar{\mathbf{Q}} \tag{2.9}$$

and the average value of the state vector over the control volume is defined as

$$\bar{\mathbf{Q}} = \frac{1}{S}\int\int_S \mathbf{Q} dS \tag{2.10}$$

The second integral in equation (2.1) is converted to a line integral by use of the divergence theorem

$$\int\int_S (\nabla \cdot \vec{\mathbf{F}}) dS = \oint_\xi \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} d\xi \tag{2.11}$$

where $\hat{\mathbf{n}}$ is the outward pointing unit vector normal to the control volume.

$$\hat{\mathbf{n}} = \hat{\xi}_x \hat{\mathbf{i}} + \hat{\xi}_y \hat{\mathbf{j}} \tag{2.12}$$

In this form, the fluxes normal to the surface of the control volume are written as

$$\hat{\mathbf{F}} = \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} = \begin{bmatrix} \rho U \\ \rho U u + \hat{\xi}_x p \\ \rho U v + \hat{\xi}_y p \\ (e+p)U \end{bmatrix} \tag{2.13}$$

where U is the velocity in the direction of the outward pointing normal

$$U = \hat{\xi}_x u + \hat{\xi}_y v \tag{2.14}$$

2.3 Flux–Vector Splitting

The time dependent Euler equations form a hyperbolic system of equations and hence information is propagated along characteristics. Therefore, to correctly model the flow physics, some form of upwind differencing should be used in order to model the characteristic nature of the governing equations. The upwind method used in the current study is the flux-vector-splitting method in which the flux vectors are split into forward and backward running contributions and differenced accordingly.

The split flux-vectors used in the current study were developed by van Leer[6]. These have been generalized to curvilinear coordinates by Anderson[4] and have since been widely used for flow computations on structured grids[4,5,8]. They are continuously differentiable at eigenvalue sign changes and allow shocks to be captured with at most two interior zones. In practice, only one zone is usually observed[4]. After splitting, the flux vectors normal to the cell face given in equation (2.13) can be re-expressed as

$$\widehat{\mathbf{F}} = \widehat{\mathbf{F}}^{+} + \widehat{\mathbf{F}}^{-} \tag{2.15}$$

where $\widehat{\mathbf{F}}^{+}$ and $\widehat{\mathbf{F}}^{-}$ are given below for $| M_{\xi} | < 1$

$$\widehat{\mathbf{F}}^{\pm} = \begin{bmatrix} f^{\pm}{}_{mass} \\ f^{\pm}{}_{mass}[\hat{\xi}_x \frac{(-U \pm 2a)}{\gamma} + u] \\ f^{\pm}{}_{mass}[\hat{\xi}_y \frac{(-U \pm 2a)}{\gamma} + v] \\ f^{\pm}{}_{energy} \end{bmatrix} \tag{2.16}$$

where

$$f^{\pm}{}_{mass} = \pm\rho a(M_\xi \pm 1)^2/4 \tag{2.17}$$

$$\begin{aligned} f^{\pm}{}_{energy} = & f^{\pm}{}_{mass}[\{-(\gamma-1)U^2 \pm 2(\gamma-1)Ua \\ & + 2a^2\}/(\gamma^2-1) + (u^2+v^2)/2] \end{aligned} \tag{2.18}$$

Here, $M_\xi = \frac{U}{a}$ is the Mach number normal to the surface of the control volume. For supersonic flow (i.e.$\mid M_\xi \mid > 1$), the fluxes are given as

$$\widehat{\mathbf{F}}^+ = \widehat{\mathbf{F}} \qquad \widehat{\mathbf{F}}^- = 0 \qquad (M_\xi \geq +1) \tag{2.19a}$$

$$\widehat{\mathbf{F}}^- = \widehat{\mathbf{F}} \qquad \widehat{\mathbf{F}}^+ = 0 \qquad (M_\xi \leq -1) \tag{2.19b}$$

where $\widehat{\mathbf{F}}$ is given by equation (2.13).

CHAPTER III

NUMERICAL METHODS

3.1 Data Structures

The governing equations are solved on a discrete curvilinear grid which consists of cells, faces, and nodes. Generally, a structured grid consists of a collection of cells which are organized into a single or multiple block structure as shown in figure 2. In each block an implied data structure exists in which the relationships between the nodes, cells, and faces are determined by simple addition or subtraction of integer values from the current cell index (i.e. $i + 1$, $i - 1$, etc.). Unstructured grid algorithms, however, rely on a system of pointers to determine node, cell, and face relationships needed to transfer information within the grid. Every cell, face, and node is assigned an integer value, as shown in figure 3, and access to each is obtained through the use of various pointers. For example, to obtain information stored at the nodes which define a cell, an array called a cell-to-node pointer must be accessed. The number and type of pointers needed is highly dependent on the computational stencil for a particular scheme.

To efficiently employ adaptive grid embedding, it is desirable to remove the restrictions of the implied data structure required for single and multiple block grids. In the current work, the grids consist of a collection of cells which are aligned with the ξ and η coordinates but no block structure is required. In addition,

all information in the grid is obtained through a system of pointers as in an unstructured grid. The use of the ξ and η coordinate system makes it simple to implement the spatial differencing techniques of structured algorithms in an unstructured framework. This approach can be referred to as a semi–unstructured method since some structure is required for the cell alignment but not the cell collection as shown in figure 4.

There are two basic methods in which the embedded grid regions can be handled. The first method is to treat the entire grid as a single level and have special data structures to handle the hanging nodes as shown in figure 5. This can be somewhat tedious because some cells may have more than 4 faces. For example, cell a in figure 5 consists of 6 faces and 6 nodes. The second method, which is used in the current work, is to treat the embedded grids as separate levels as shown in figure 6. This allows every cell to have only 4 faces and nodes. This multi–level grid structure also facilitates the use of a multigrid algorithm to enhance the convergence rate of the flow solver. The use of upwind differencing, adaptive embedding, and multigrid, is accommodated with the following set of pointers:

1. Cell–to–Face pointer - relates a cell to its four faces.
2. Cell–to–Node pointer - relates a cell to its four nodes.
3. Face–to–Cell pointer - relates a face to the cells used in determining quantities on the face.
4. Face–to–Node pointer - relates a face to its two nodes.

5. Link pointer - relates a cell to the four cells below it and the four cells above it. The coarse cell which lies below four finer cells is referred to as a parent and the finer cells are referred to as children.

These pointers, and the numbering system for each, are demonstrated in figures 7a-f. For setting the boundary conditions, additional information is required which contains the face numbers of both the solid walls and the inflow/outflow boundaries. The indexing used in the algorithm descriptions will be based on the use of these pointers.

3.2 Flux Integration

The equations are solved using a finite volume formulation which entails solving for the average quantities in each cell. The flux integral in equation (2.11) is approximated by:

$$\oint_{\xi}(\hat{\mathbf{F}})d\xi = \sum_{n=1}^{4}(\hat{\mathbf{F}}_n)\ell_n \tag{3.1}$$

where the subscript n is the face number (see figure 7a) and ℓ_n is the length of each face. Note that the flux balance given by equation (3.1) is based on outward pointing normals whereas the normals for the semi-unstructured grid have been chosen, for convenience, to always point in the positive ξ and η direction. The resulting flux integration (with splitting) for a cell is written as

$$\begin{aligned}\oint_{\xi}(\hat{\mathbf{F}})d\xi = &- [\tilde{\mathbf{F}}_1^+(\mathbf{Q}_1^-) + \tilde{\mathbf{F}}_1^-(\mathbf{Q}_1^+)]\ell_1 + [\tilde{\mathbf{F}}_2^+(\mathbf{Q}_2^-) + \tilde{\mathbf{F}}_2^-(\mathbf{Q}_2^+)]\ell_2 \\ &+ [\tilde{\mathbf{F}}_3^+(\mathbf{Q}_3^-) + \tilde{\mathbf{F}}_3^-(\mathbf{Q}_3^+)]\ell_3 - [\tilde{\mathbf{F}}_4^+(\mathbf{Q}_4^-) + \tilde{\mathbf{F}}_4^-(\mathbf{Q}_4^+)]\ell_4\end{aligned} \tag{3.2}$$

The fluxes $\tilde{\mathbf{F}}^{\pm}$ are constructed using the same equations as $\hat{\mathbf{F}}^{\pm}$ but normals pointing in the positive coordinate directions are used in place of outward pointing normals. Here, ℓ_n is the physical length of face n and $\mathbf{Q}^{\pm}$ are the state variables interpolated to the faces from the positive and negative directions. The interpolation formulas, using the stencil shown in figure 7*c*, are given by

$$\mathbf{Q}^{-}_{face(n)} = \mathbf{Q}_{cell(2)} + \frac{1}{4}\phi[(1-\kappa)\Delta_{-} + (1+\kappa)\Delta_{+}]\mathbf{Q}_{cell(2)} \tag{3.3}$$

$$\mathbf{Q}^{+}_{face(n)} = \mathbf{Q}_{cell(3)} - \frac{1}{4}\phi[(1+\kappa)\Delta_{-} + (1-\kappa)\Delta_{+}]\mathbf{Q}_{cell(3)} \tag{3.4}$$

where Δ_+ and Δ_- are forward and backward difference operators defined as

$$\Delta_{+}\mathbf{Q}_{cell(n)} = \mathbf{Q}_{cell(n+1)} - \mathbf{Q}_{cell(n)} \tag{3.5}$$

$$\Delta_{-}\mathbf{Q}_{cell(n)} = \mathbf{Q}_{cell(n)} - \mathbf{Q}_{cell(n-1)} \tag{3.6}$$

For $\kappa = -1$, these interpolation formulas give a fully one-sided extrapolation of the dependent variables to the cell faces which results in a second–order accurate difference formula for equation (3.2) on a uniformly spaced grid[5]. Similarly, the use of $k = \frac{1}{3}$ results in a three–point interpolation formula which produces a third–order accurate difference approximation to equation (3.2) on a uniform grid.

When using a three–point interpolation formula, oscillations in the solution can occur in regions of strong gradients such as shocks. To eliminate these oscillations, flux–limited interpolations[5] are employed which are given by

$$\mathbf{Q}^{-}_{face(n)} = \mathbf{Q}_{cell(2)} + \frac{s}{4}\phi[(1-\kappa s)\Delta_{-} + (1+\kappa s)\Delta_{+}]\mathbf{Q}_{cell(2)} \tag{3.7}$$

$$\mathbf{Q}^{+}_{face(n)} = \mathbf{Q}_{cell(3)} - \frac{s}{4}\phi[(1+\kappa s)\Delta_{-} + (1-\kappa s)\Delta_{+}]\mathbf{Q}_{cell(3)} \tag{3.8}$$

where

$$s = \frac{2\Delta_{+}\Delta_{-} + \varepsilon}{(\Delta_{+})^2 + (\Delta_{-})^2 + \varepsilon} \tag{3.9}$$

and ε is a small number ($\varepsilon = 10^{-6}$) preventing division by zero in regions of null gradients. The variable ϕ is a user defined parameter which is equal to 1 but can be set to 0 for a one–sided first–order method.

3.3 Time Advancement

The solution of equation (3.2) is advanced in time using the modified Euler method[9], which is an explicit 2-stage scheme of the form

$$\mathbf{Q}^{*} = \mathbf{Q}^{n} + \Delta t \mathbf{R}^{n} \tag{3.10}$$

$$\mathbf{Q}^{n+1} = \frac{1}{2}(\mathbf{Q}^{n} + \mathbf{Q}^{*}) + \frac{1}{2}\Delta t \mathbf{R}^{*} \tag{3.11}$$

where the residual is given as

$$\begin{aligned}\mathbf{R} = -\{ &- [\tilde{\mathbf{F}}^{+}_{1}(\mathbf{Q}^{-}_{1}) + \tilde{\mathbf{F}}^{-}_{1}(\mathbf{Q}^{+}_{1})]\ell_1 + [\tilde{\mathbf{F}}^{+}_{2}(\mathbf{Q}^{-}_{2}) + \tilde{\mathbf{F}}^{-}_{2}(\mathbf{Q}^{+}_{2})]\ell_2 \\ &+ [\tilde{\mathbf{F}}^{+}_{3}(\mathbf{Q}^{-}_{3}) + \tilde{\mathbf{F}}^{-}_{3}(\mathbf{Q}^{+}_{3})]\ell_3 - [\tilde{\mathbf{F}}^{+}_{4}(\mathbf{Q}^{-}_{4}) + \tilde{\mathbf{F}}^{-}_{4}(\mathbf{Q}^{+}_{4})]\ell_4\}\end{aligned} \tag{3.12}$$

The superscripts for the residual designate the time level of the state variables used in the flux evaluation.

The solution for each cell is advanced at its own local time step corresponding to a CFL number given by:

$$CFL = \Delta t\{|\ U\ | + |\ V\ | + a[|\ grad(\xi)\ | + |\ grad(\eta)\ |]\} \tag{3.13}$$

The contravariant velocity in the ξ direction, U, is given in equation (2.14). The contravariant velocity, V, is obtained simply by replacing ξ with η.

3.4 Stability Analysis

The stability analysis follows from the work of reference 4. The stability characteristics are examined using the linearized two-dimensional Euler equations given by

$$\frac{\partial \mathbf{Q}}{\partial t} = -P(\mathbf{Q}) \tag{3.14}$$

where

$$P(\mathbf{Q}) = \mathbf{A}\frac{\partial \mathbf{Q}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{Q}}{\partial y} \tag{3.15}$$

The flux Jacobians, $\mathbf{A} = \frac{\partial \mathbf{F}_1}{\partial \mathbf{Q}}$ and $\mathbf{B} = \frac{\partial \mathbf{F}_2}{\partial \mathbf{Q}}$, are computed using the split flux vectors for the x and y directions respectively. The differentials are computed across the cell using the interpolation formulas given by equations (3.3) - (3.6). The Fourier mode for $\mathbf{Q}$ is assumed to be

$$\mathbf{Q} = \bar{\mathbf{Q}} e^{i\beta x} e^{i\Gamma y} \tag{3.16}$$

where $\bar{\mathbf{Q}}$ is an initial state vector. Substituting $-P(\mathbf{Q})$ for the residual in equations (3.10) and (3.11) and replacing $\mathbf{Q}$ with its Fourier mode, a system of equations is obtained of the form

$$\bar{\mathbf{Q}}^{n+1} = \mathbf{G}\bar{\mathbf{Q}}^{n} \tag{3.17}$$

The magnitude of the maximum eigenvalue of the amplification matrix $\mathbf{G}$ must be less than unity for stability. A computer program was used to cycle through a

fixed number of each of the spatial frequencies in the range

$$0 \leq \beta \Delta x, \Gamma \Delta y \leq 2\pi \tag{3.18}$$

for a series of CFL numbers between 0.1 and 1.2. The generalized eigenvalue problem is solved using an IMSL[10] routine. In addition, the high frequency error damping properties (smoothing) of the scheme can be studied by examining the maximum eigenvalue in the frequency range between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$.

Shown in figures 8 and 9 are the stability characteristics for $\kappa = 1/3$ and $\kappa = -1$ for a Mach number of 0.8 and $\alpha = 0.0$. Both schemes exhibit poor high frequency error damping properties. Because the $\kappa = 1/3$ scheme allows a higher CFL number along with higher spatial accuracy, this was the scheme chosen for the computations presented later.

3.5 Boundary Conditions

For setting the boundary conditions, there are three possible boundary types which must be considered. These include inflow/outflow, solid wall, and interfaces between embedded grid regions. All of the boundary conditions are implemented using "ghost" cells which lie outside of the computational domain. The solid wall and inflow/outflow boundary faces have only one ghost cell and hence the general four point stencil shown in figure 7c does not exist. The differencing on these faces is accomplished by setting $\mathbf{Q}^+$ or $\mathbf{Q}^-$ (depending on the boundary) equal to that of the ghost cell. For the solid wall and embedded interface boundary conditions,

the ghost cells are updated before each time step whereas, for the inflow/outflow boundaries, the ghost cells are set to freestream values.

3.5.1 Inflow/Outflow Boundary Conditions

The inflow/outflow boundary conditions are developed using characteristic theory[11] which is summarized here. A local coordinate system $(\bar{x}, \bar{y})$ is constructed orthogonal to the boundary as shown in figure 10. The characteristic equations are constructed assuming the tangential derivatives are negligible so that a one-dimensional analysis normal to the boundary can be employed. The resulting characteristics are sketched in figure 11. Assuming locally isentropic flow, the characteristic equations are given by

$$\frac{ds}{dt} = 0 \quad \text{along} \quad C^0 = u \tag{3.19a}$$

$$\frac{d\bar{v}}{dt} = 0 \quad \text{along} \quad C^0 = u \tag{3.19b}$$

$$\frac{d}{dt}(R^{\pm}) = 0 \quad \text{along} \quad C^{\pm} = \bar{u} \pm a \tag{3.19c}$$

where the Riemann invariants, $R^{\pm}$, are given by

$$R^{\pm} \equiv \bar{u} \pm \frac{2a}{(\gamma - 1)} \tag{3.20}$$

The variables at the new time level t^{n+1} are obtained from the Riemann invariants. Assuming that $\bar{x}$ points away from the computational domain, R^+ and R^- are evaluated using the state variables from outside and inside the computational domain respectively. The normal velocity and speed of sound on the inflow/outflow

boundary are obtained by adding and subtracting the Riemann invariants.

$$\bar{u}_{face} = \frac{1}{2}(R^+ + R^-) \tag{3.21}$$

$$a_{face} = \frac{(\gamma - 1)}{4}(R^+ - R^-) \tag{3.22}$$

Once the normal velocity and speed of sound are calculated, the Cartesian velocities are determined by decomposing the normal and tangential velocity vectors

$$u_{face} = u_{ref} + \hat{n}_x(\bar{u}_{face} - \bar{u}_{ref}) \tag{3.23}$$

$$v_{face} = v_{ref} + \hat{n}_y(\bar{u}_{face} - \bar{u}_{ref}) \tag{3.24}$$

where the subscript ref represents freestream values for inflow or from the cell inside the domain adjacent to the boundary for outflow. It should be noted that since the local coordinate $\bar{x}$ points away from the computational domain, inflow corresponds to $\bar{u} < 0$ and outflow corresponds to $\bar{u} > 0$. The variables $\hat{n}_x$ and $\hat{n}_y$ are the x and y components of the outward pointing unit normal for the face.

The entropy s is determined using the value from outside the domain for inflow and from inside the domain for outflow. Once the entropy on the boundary is known, the density on the face is calculated from the entropy and speed of sound on the face

$$\rho_{face} = \left[\frac{a^2_{face}}{\gamma s_{face}}\right]^{\frac{1}{\gamma - 1}} \tag{3.25}$$

The energy on the face is then calculated using the equation of state.

For supersonic conditions, the state variables are extrapolated to the boundary from the exterior for inflow boundaries and extrapolated from the interior for outflow boundaries.

3.5.2 Solid Wall Boundary Conditions

The variables on solid wall boundaries are computed by extrapolating from the interior of the computational domain. For the cases presented here, the pressure and density on the face are simply set to the values of the cell above it. The normal velocity is then set to zero so that the resulting Cartesian velocities on the face are given by

$$u_{face} = u_{cell} - \hat{n}_x(\bar{u}_{cell}) \tag{3.26}$$

$$v_{face} = v_{cell} - \hat{n}_y(\bar{u}_{cell}) \tag{3.27}$$

Here, $\bar{u}_{cell}$ is constructed with a normal facing in the positive ξ or η direction and $\hat{n}_x$ and $\hat{n}_y$ are the x and y components of this normal. The density, pressure, and Cartesian velocities are then extrapolated to the ghost cells using

$$\phi_{ghost} = 2\phi_{face} - \phi_{cell} \tag{3.28}$$

Note that only the Cartesian velocities need to be extrapolated since the pressure and density in the ghost cell are the same as in the computational cell above it.

3.5.3 Embedded Interface Boundary Conditions

A difficulty encountered when using embedded grids is in determining the fluxes on the interface between the coarse and fine grids inside the computational

domain so that the accuracy and conservation properties of the flow solver are maintained. The method for obtaining the flux on the coarse grid is the same as that for obtaining the other coarse grid faces since the four–point computational stencil is complete. Obtaining the flux on the two finer faces can be accomplished several ways. The method developed here is similar to that of Berger[1]. A set of ghost cells is constructed outside of the interface region as shown in figure 12. Note that with the addition of the ghost cells, the four–point stencil needed for the interpolations to obtain the state variables on the faces is preserved. In this region, the state variables are determined by bilinear interpolation from the next coarsest grid level using the most current solution on that level. This produces the interpolation stencil shown in figure 13 for the fine grid fluxes since all nine of the shaded coarse grid values are used to obtain the values in the ghost cells. The coarse grid flux stencil is shown in figure 14.

This method of calculating the fluxes, however, does not enforce conservation at the interface region since there is nothing to ensure the sum of the fluxes on the finer grid equals the flux on the coarse grid. To enforce conservation at the interface, the fluxes from the two finer faces are added and injected onto the coarse grid before the coarse grid update

$$\tilde{\mathbf{F}}_c \ell_c = \tilde{\mathbf{F}}_{f1} \ell_{f1} + \tilde{\mathbf{F}}_{f2} \ell_{f2} \tag{3.29}$$

3.6 Multigrid Algorithm

The algorithm presented thus far is still incomplete because no communication

exists between the grid levels. The increased accuracy of the fine grids is not passed to the coarser grids, so the solution on the coarse grid remains unchanged. This is unacceptable since the ghost cells for the finer grids are determined by interpolation from the coarse grid solution. A multigrid algorithm is a natural way to obtain grid–level communication. The truncation error between a fine and a coarse grid is passed to the coarse grid in such a way that the coarse grid is driven by the fine grid residual. Hence, the fine grid accuracy is maintained on the coarser grid.

The multigrid algorithm used in the current study is the Full-Approximation Scheme (FAS) which has been primarily used in solutions of the Euler equations as a method of convergence acceleration such as described in reference 4. Because of the embedded grids used in the current algorithm, slight modifications to the algorithm presented in reference 4 must be made since not all cells on the coarser grids have finer cells above them.

The application of multigrid algorithms to Euler solvers has primarily been due to the convergence acceleration which is attained. The improved convergence is attributed to the fact that the multigrid algorithm efficiently damps low-frequency errors on the finer grids by using a sequence of grid levels denoted by G_N, G_{N-1}, $\cdots$,G_1 where the coarsest grid level corresponds to G_1. Since the lower levels have greater spacing between grid lines, the low frequency errors on the higher levels appear as high frequency errors on the lower levels where they can be efficiently damped.

Explanation of the multigrid process is facilitated by first examining the Euler equations written in operator notation and discretized on a given grid, G_N, which is defined to be the finest level.

$$\mathbf{L}_N(\mathbf{Q}_N) = 0 \tag{3.30}$$

where $\mathbf{Q}_N$ is the exact solution to the discretized system and $\mathbf{L}_N$ is the discrete steady state operator given by

$$\begin{aligned}\mathbf{L}_N(\mathbf{Q}_N) = -\{ &- [\tilde{\mathbf{F}}_1^+(\mathbf{Q}_1^-) + \tilde{\mathbf{F}}_1^-(\mathbf{Q}_1^+)]\ell_1 + [\tilde{\mathbf{F}}_2^+(\mathbf{Q}_2^-) + \tilde{\mathbf{F}}_2^-(\mathbf{Q}_2^+)]\ell_2 \\ &+ [\tilde{\mathbf{F}}_3^+(\mathbf{Q}_3^-) + \tilde{\mathbf{F}}_3^-(\mathbf{Q}_3^+)]\ell_3 - [\tilde{\mathbf{F}}_4^+(\mathbf{Q}_4^-) + \tilde{\mathbf{F}}_4^-(\mathbf{Q}_4^+)]\ell_4\}\end{aligned} \tag{3.31}$$

When solving iteratively, equation (3.31) is solved approximately at each time step so that the right hand side of (3.30) is not identically zero. This can be represented by introduction of a residual defined implicitly by

$$\mathbf{L}_N(\mathbf{q}_N^c) = \mathbf{R}_N \tag{3.32}$$

where $\mathbf{q}_N^c$ is the most current approximation to $\mathbf{Q}_N$ and $\mathbf{R}_N$ is the residual. Note that the residual will be zero only when the approximate solution ($\mathbf{q}_N^c$) is equal to the exact solution ($\mathbf{Q}_N$).

Since it is the errors that are damped on any given grid, it is desirable to cast the governing equations for a given grid in terms of the errors. This is achieved by subtraction of equation (3.32) from equation (3.31)

$$\mathbf{L}_N(\mathbf{Q}_N) - \mathbf{L}_N(\mathbf{q}_N^c) = -\mathbf{R}_N \tag{3.33}$$

If the time stepping method adequately eliminates the high-frequency errors on the current grid level G_N, the residual equation, (3.31), may be adequately approximated on the coarser parent cells G_{N-1} by

$$\mathbf{L}_{N-1}(\mathbf{Q}_{N-1}) = \hat{\mathbf{I}}_N^{N-1}(-\mathbf{R_N}) + \mathbf{L}_{N-1}(\mathbf{I}_N^{N-1}\mathbf{q}_N^c) \tag{3.34}$$

Here $\mathbf{I}_N^{N-1}$ is a volume-weighted collection operator that transfers dependent variables from the children to the parents so that conservation is maintained[4] and is given by

$$\mathbf{I}_N^{N-1}\mathbf{Q}_N = \frac{\sum V\mathbf{Q}_N}{\sum V} \tag{3.35}$$

Here, V is the volume of each child cell. Similarly $\hat{\mathbf{I}}_N^{N-1}$ is the collection operator for the residual which is defined as

$$\hat{\mathbf{I}}_N^{N-1}\mathbf{R}_N = \sum \mathbf{R}_N \tag{3.36}$$

For equations (3.35) and (3.36) the summations are over the four fine grid cells (children) which make up the coarser parent cell. In the current study, the dependent variables in the ghost cells for the finer grid are not restricted down to the coarser grid.

Note that equation (3.34) can be written as

$$\mathbf{L}_{N-1}(\mathbf{Q}_{N-1}) = \tau_{N-1} \tag{3.37}$$

where

$$\tau_{N-1} = \mathbf{L}_{N-1}(\mathbf{I}_N^{N-1}\mathbf{q}_N^c) - \hat{\mathbf{I}}_N^{N-1}(\mathbf{R}_N) \tag{3.38}$$

is the relative truncation error between grid levels G_N and G_{N-1}. For coarse grid cells with no finer cells above them, τ_{N-1} is set to zero. The inclusion of τ_{N-1} allows the solution in the parent cells to be driven by the finer grid so that the order of accuracy on the fine grid is maintained. The new residual which can be written in a form applicable to all grid levels, G_i is given by

$$\mathbf{R}_i = \mathbf{L}_i(\mathbf{q}_i^c) - \tau_i \tag{3.39}$$

This residual is now used for updating the solution on the current grid using the time advancing algorithm, equations (3.10) and (3.11), described earlier.

To proceed down to the next coarsest level, G_{N-2}, the analogous equation to equation (3.37) is given by

$$\mathbf{L}_{N-2}(\mathbf{Q}_{N-2}) = \tau_{N-2} \tag{3.40}$$

where

$$\tau_{N-2} = \mathbf{L}_{N-2}(\mathbf{I}_{N-1}^{N-2}\mathbf{q}_{N-1}^c) - \hat{\mathbf{I}}_{N-1}^{N-2}(\mathbf{R}_{N-1}) \tag{3.41}$$

Note that $\mathbf{R}_{N-1}$contains the relative truncation error between levels N and $N-1$ so that the relative truncation error on this grid is the sum of the truncation errors between levels N and $N-1$ as well as $N-1$ and $N-2$.

Thus far, the relative truncation errors between the parents and children have been succesively passed down to each of the parent cells so that the order of accuracy of the children is preserved. In this manner, the accuracy of the finer levels "drives" the accuracy of the coarser levels.

Since the lower levels damp out the low frequency errors of the higher levels, a correction from the coarse grid must be passed up to the higher levels. This correction $\mathbf{V}$ is formed on grid level $N-2$ as

$$\mathbf{V}_{N-2} = \mathbf{q}^c_{N-2} - \mathbf{I}^{N-2}_{N-1}\mathbf{q}^c_{N-1} \tag{3.42}$$

This correction is then added to the dependent variable in the children cells

$$\mathbf{q}^c_{N-1} \leftarrow \mathbf{q}^c_{N-1} + \mathbf{I}^{N-1}_{N-2}\mathbf{V}_{N-2} \tag{3.43}$$

where $\mathbf{I}^{N-1}_{N-2}$ is an interpolation operator for passing information from a lower level to a higher level as described in Appendix A. A correction is then formed on level $N-1$ and passed up to level N using equations (3.42) and (3.43) by replacing $N-1$ with N and $N-2$ with $N-1$. Note that no additional iterations are performed on grid $N-1$ before interpolation of corrections to grid level N.

The process described above is a three–level "V"-cycle algorithm and is depicted in figure 15. For further clarity, detailed steps for a three–level V-cycle are given below.

1. Start on the highest grid level and advance the solution in time using equation (3.39) and $\tau_N = 0$.
2. Calculate the residual with the most current values on the highest level from equation (3.39) and $\tau_N = 0$.
3. For cells on G_{N-1} with children, collect the dependent variables from G_N using equation (3.35).

4. For cells on G_{N-1} with children, collect the residual from G_N using equation (3.36) and calculate the relative truncation error between the grids using equation (3.38). For cells on G_{N-1} with no children, set $\tau_{N-1} = 0$.

5. Advance the solution in time on level G_{N-1} using equation (3.39) to calculate the residual. Since τ_{N-1} has been previously calculated, the residual is calculated by evaluating $\mathbf{L}_{N-1}(\mathbf{q}^c_{N-1})$ from the most current values of the dependent variables on this level and subtracting τ_{N-1}. Again, note that $\tau_{N-1} = 0$ for cells on G_{N-1} with no children.

6. Calculate the residual on this level using equation (3.39).

$$\mathbf{R}_{N-1} = \mathbf{L}_{N-1}(\mathbf{q}^c_{N-1}) - \tau_{N-1}$$

7. For cells on G_{N-2} with children, collect the dependent variables from G_{N-1} using equation (3.35).

8. For cells on G_{N-2} with children, collect the residual from G_{N-1} using equation (3.36) and calculate the relative truncation error using equation (3.41). For cells on G_{N-2} with no children, set $\tau_{N-2} = 0$.

9. Advance the solution in time on this level using equation (3.39) to calculate the residual on G_{N-2}. Since this is the lowest level used in the present example, several solution updates are performed to get an approximation to $\mathbf{Q}_{N-2}$. During each step, the residual is updated to use the most current values of the dependent variables in $\mathbf{L}_{N-2}(\mathbf{q}^c_{N-2})$. Note that τ_{N-2} will not change.

9. Calculate the correction on this level to give

$$\mathbf{V}_{N-2} = \mathbf{q}^c_{N-2} - \mathbf{I}^{N-2}_{N-1}\mathbf{q}^c_{N-1}$$

10. Pass the correction to the next finest mesh using bilinear interpolation and update the solution to give

$$\mathbf{q}_N^c \leftarrow \mathbf{q}_N^c + \mathbf{I}_{N-1}^N \mathbf{V}_{N-1}$$

12. Calculate the correction on the $N-1$ level with

$$\mathbf{V}_{N-1} = \mathbf{q}_{N-1}^c - \mathbf{I}_N^{N-1} \mathbf{q}_N^c$$

13. Pass this correction to the highest level and update the solution

$$\mathbf{q}_N^c \leftarrow \mathbf{q}_N^c + \mathbf{I}_{N-1}^N \mathbf{V}_{N-1}$$

14. Advance the solution in time to smooth the errors.

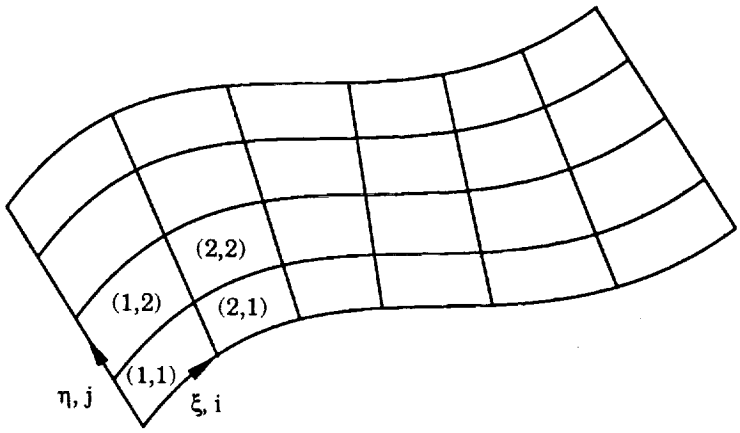


Figure 2. Structured Grid

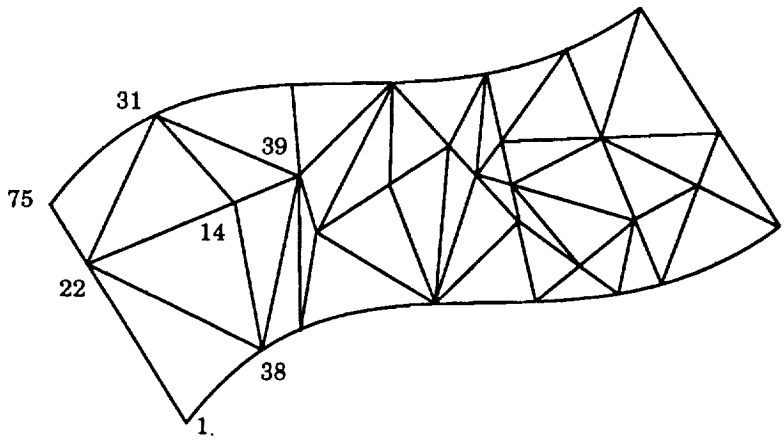


Figure 3. Unstructured Grid

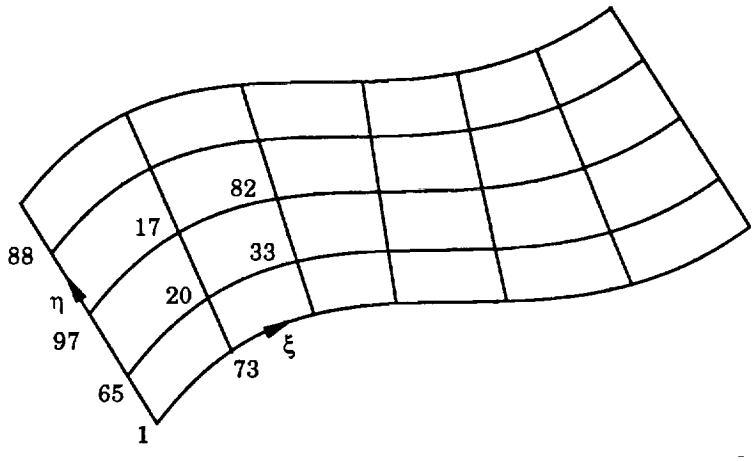


Figure 4. Semi-unstructured Grid

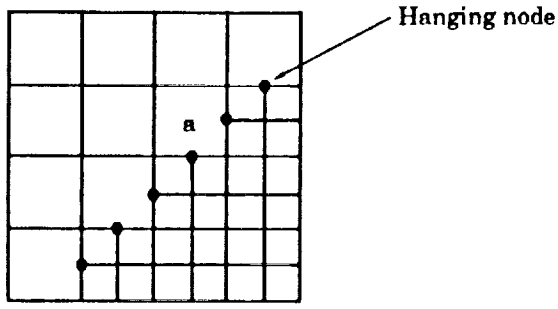


Figure 5. Single Grid Level Structure

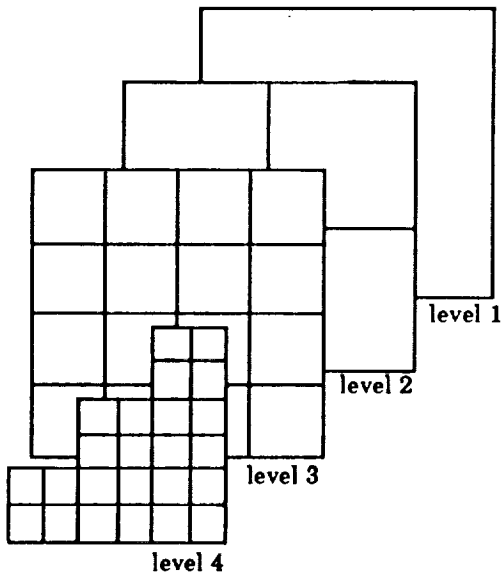
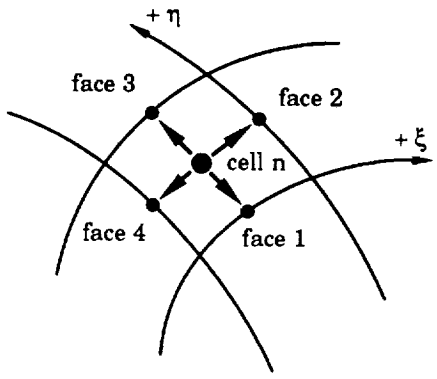
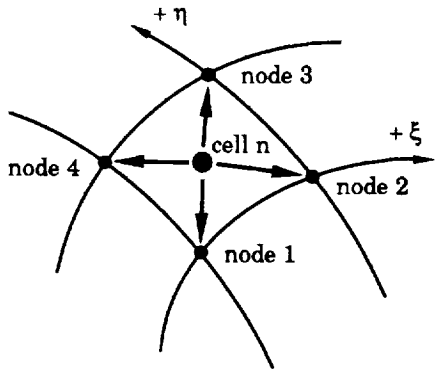


Figure 6. Multiple Grid Level Structure



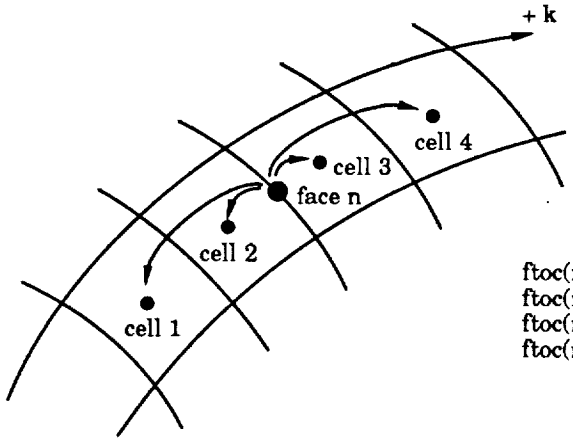
ctof(n,1) = face 1
ctof(n,2) = face 2
ctof(n,3) = face 3
ctof(n,4) = face 4

Figure 7a. Cell-to-Face Pointer



cton(n,1) = node 1
cton(n,2) = node 2
cton(n,3) = node 3
cton(n,4) = node 4

Figure 7b. Cell-to-Node Pointer



ftoc(n,1) = cell 1
ftoc(n,2) = cell 2
ftoc(n,3) = cell 3
ftoc(n,4) = cell 4

Figure 7c. Face-to-Cell Pointer

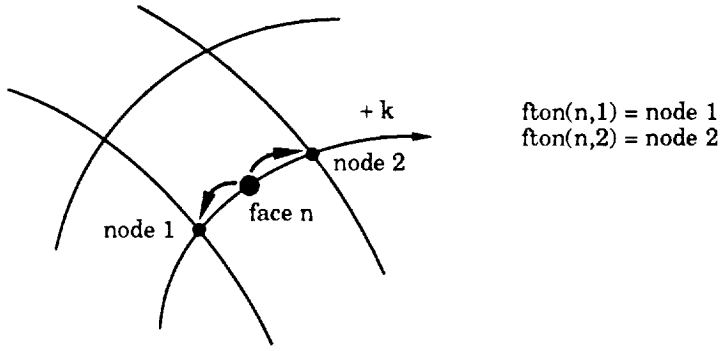


Figure 7d. Face-to-Node Pointer

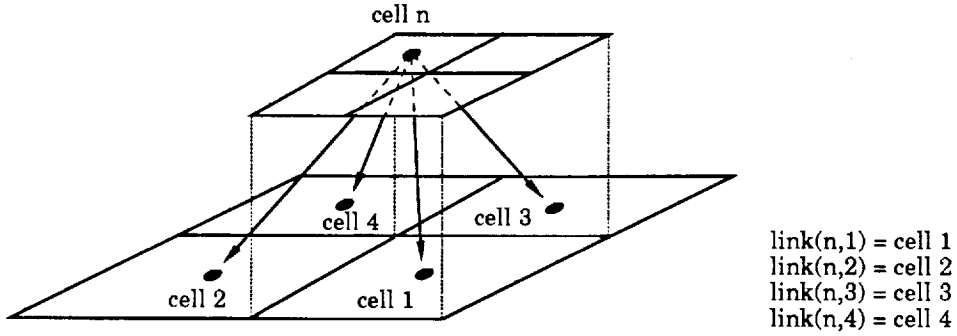


Figure 7e. Lower Link Pointer

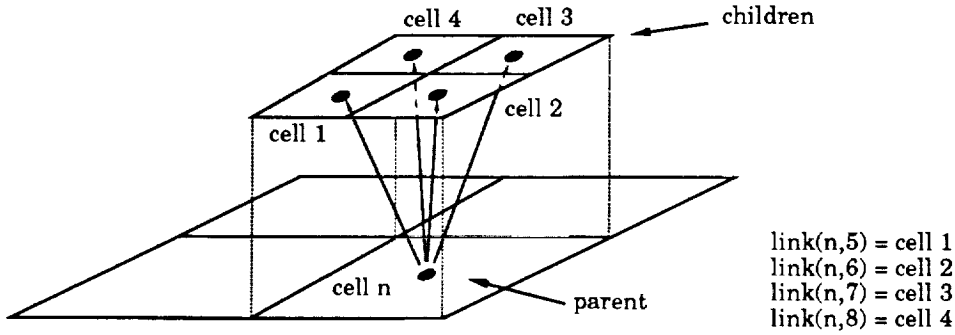


Figure 7f. Upper Link Pointer

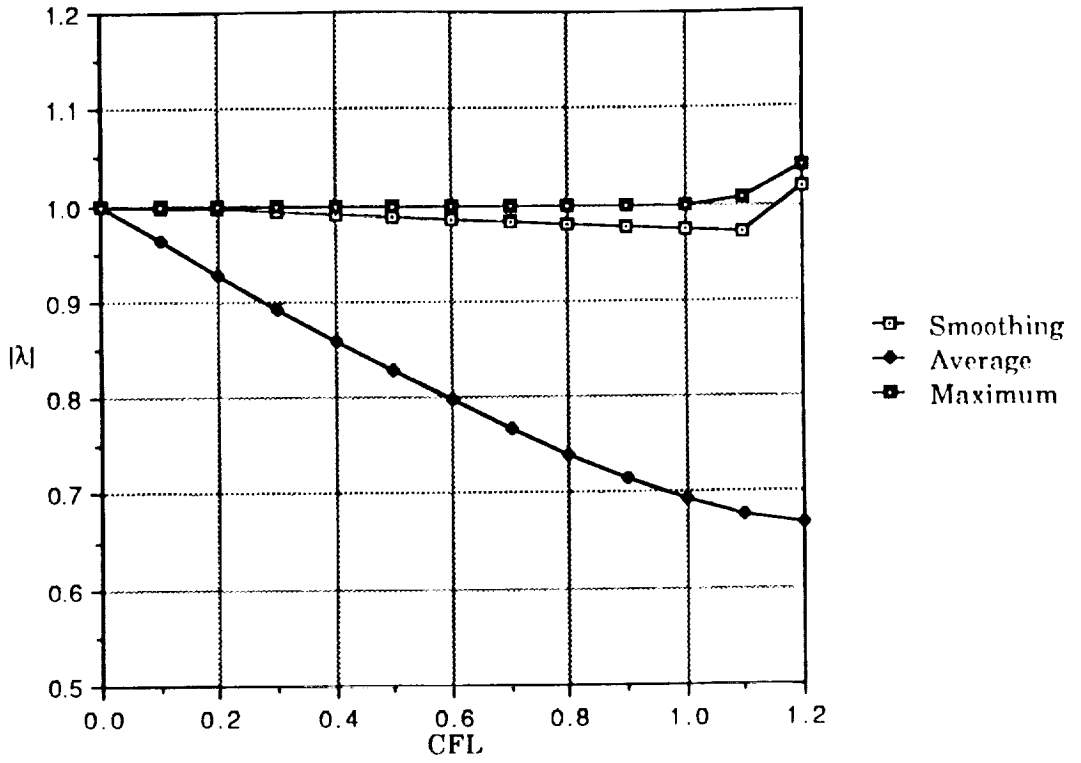


Figure 8. Stability Characteristics for k=1/3

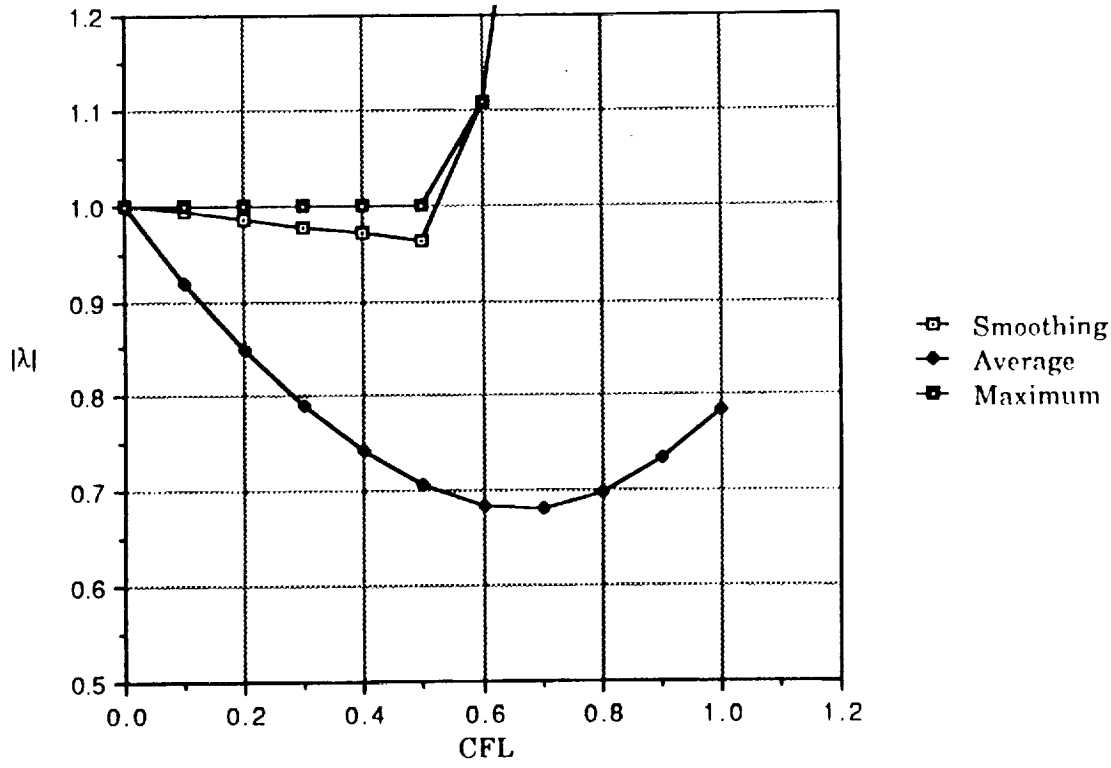


Figure 9. Stability Characteristics for k=-1

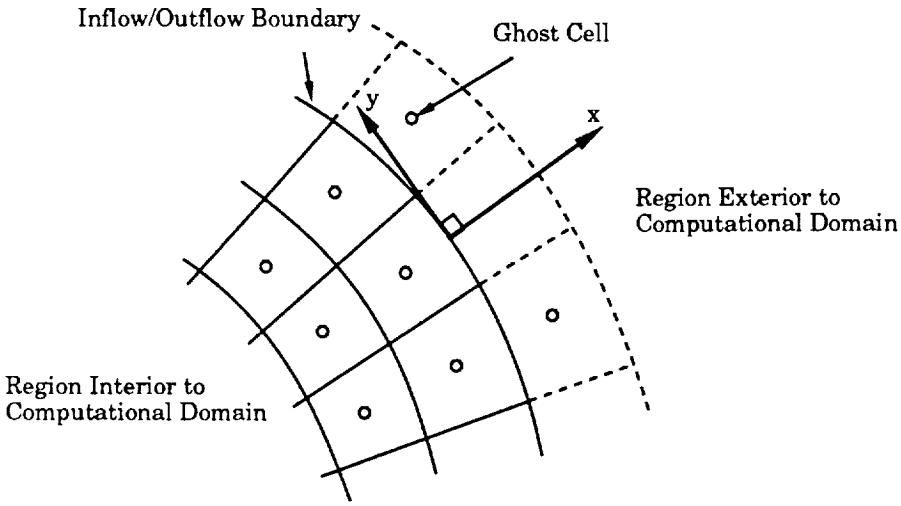


Figure 10. Inflow/Outflow Boundary

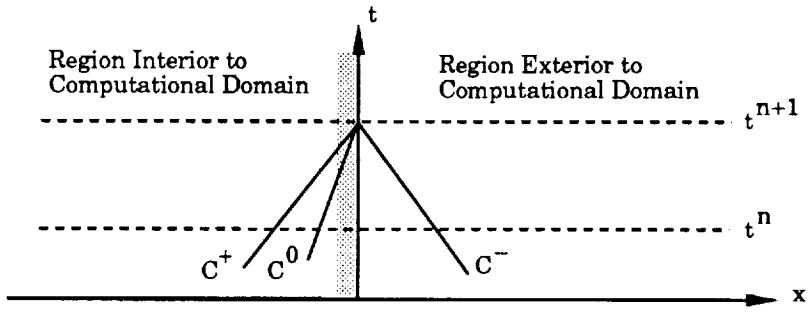


Figure 11. Characteristics at Inflow/Outflow Boundary

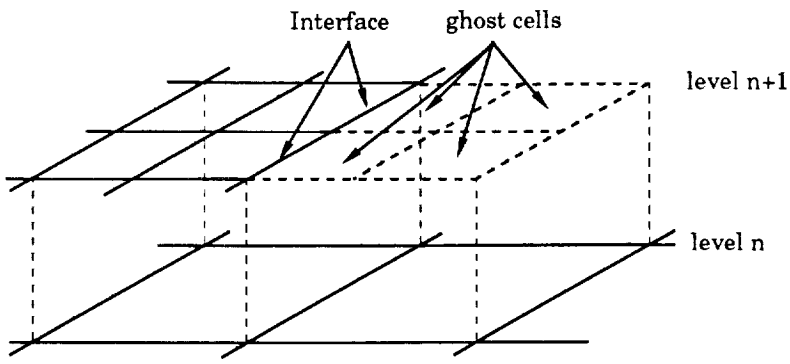


Figure 12. Embedded Ghost Cells

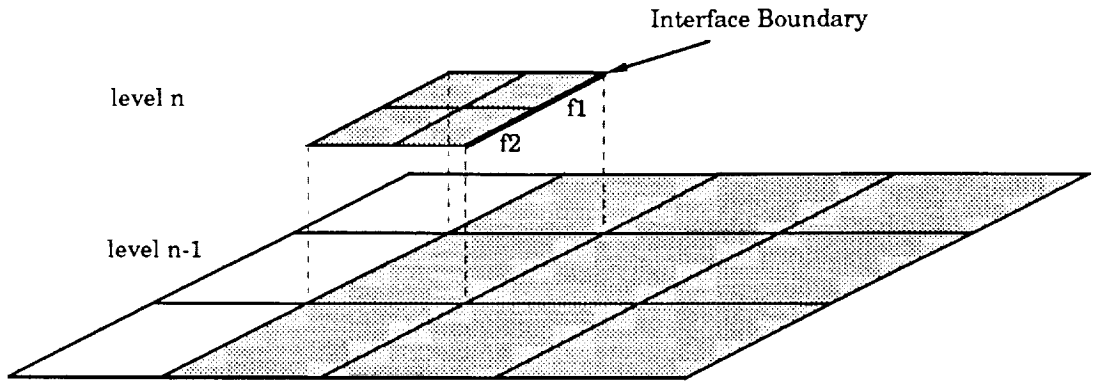


Figure 13. Computational Stencil for Fine Grid Fluxes

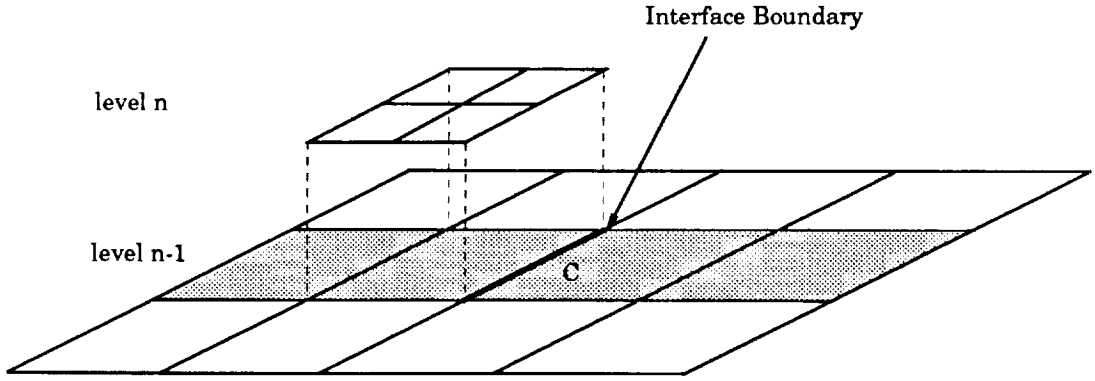


Figure 14. Computational Stencil for Coarse Grid Flux

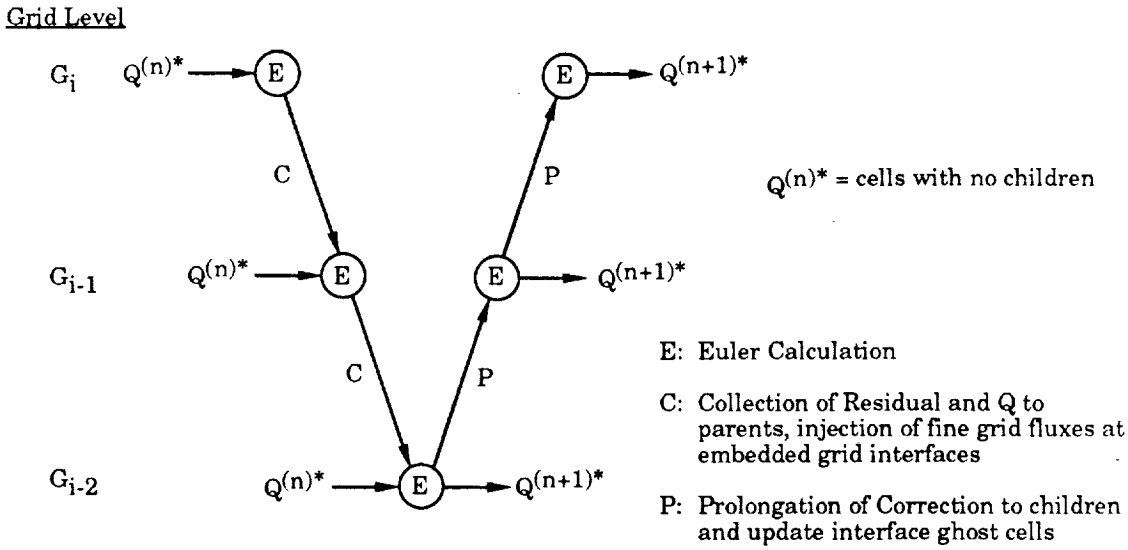


Figure 15. Multigrid V-cycle

CHAPTER IV

GRID EMBEDDING AND ADAPTATION

4.1 Grid and Pointer Generation

The data structure of this algorithm requires the generation of all the pointers described in chapter 3. This is accomplished by first obtaining an initial structured grid from which all the necessary pointers are then extracted. The initial grids used in this study are O-type grids, as shown in figure 16. These grids are generated using transfinite interpolation to obtain the coordinates of each of the nodes[12]. Global coarser grids, required for use in the multigrid algorithm, are generated simply by removing every other point from the next finest grid. On each of these grids, the pointers are obtained in the same manner as for the finest grid.

4.2 Adaptation

For grid adaptation, it is first necessary to determine which regions of the flowfield contain high gradients. For this, various refinement parameters can be used which have been investigated in recent years. An extensive investigation into the suitability of many parameters appropriate for inviscid transonic flow has been conducted by Dannenhoffer[3]. A summary of the refinement parameters and their effectiveness in detecting flow features is given in table 4.1. As seen in the table, the refinement parameters $|\tilde{\nabla}\rho|$ (ρ = denisity) and $|\tilde{\nabla}q|$ (q = velocity magnitude)

β	Feature type				
	shock wave	slip line	expansion fan	stagnation zone	shock wake
$\lvert\tilde{\rho}\rvert$	0	0	0	1	0
$\tilde{\nabla}\rho$	2	1	2	1	0
$\tilde{\nabla}^2\rho$	2	2	1	1	0
$\lvert p\rvert$	0	0	0	2	0
$\tilde{\nabla}p$	2	0	2	2	0
$\tilde{\nabla}^2 p$	2	2	1	2	0
$\lvert q\rvert$	0	0	0	0	0
$\tilde{\nabla}q$	2	2	2	2	0
$\tilde{\nabla}^2 q$	2	2	1	2	0
$\lvert p_o\rvert$	1	0	0	0	2
$\tilde{\nabla}p_o$	2	2	0	0	2
$\tilde{\nabla}^2 p_o$	2	2	0	0	2

Note: 0 $\Rightarrow$ the feature is not detected

1 $\Rightarrow$ the feature is somewhat detected

2 $\Rightarrow$ the feature is well detected

Table 4.1: Expected effectiveness of various refinement parameters for inviscid, transonic flows over airfoil-like bodies (from reference [3])

are both capable of detecting all of the features considered except the wake region behind a shock. As noted in reference 3, refinement parameters based on total pressure can be used to successfully refine this region but tend to over–refine the grid downstream of the shock with the result that the grid extending from the shock all the way to the downstream boundary is flagged for refinement.

The operator $\tilde{\nabla}$ used in defining the refinement parameter is the undivided gradient whose magnitude for an arbitrary scalar ϕ is given by

$$| \tilde{\nabla}\phi | = \sqrt{\left[\frac{\partial \phi}{\partial \xi}\right]^2 + \left[\frac{\partial \phi}{\partial \eta}\right]^2} \tag{4.1}$$

The partial derivatives in equation (4.1) are currently evaluated using undivided central differences. The refinement parameters, $\beta = |\tilde{\nabla}\rho|$ and $\beta = |\tilde{\nabla}q|$, are normalized to form a new parameter, $\bar{\beta}$

$$\bar{\beta} = \frac{\beta - \beta_{min}}{\beta_{max} - \beta_{min}} \tag{4.2}$$

which varies from 0.0 to 1.0.

After a solution is obtained on a current grid level, each childless cell is examined to determine whether $\bar{\beta}$ for that cell is less than a user input threshold value $\bar{\beta}^*$, in which case the cell is flagged for refinement. Note that the value of $\bar{\beta}^*$ is somewhat arbitrary. However, selecting a threshold value too low will embed too many cells and generally pick up "noise" in the solution. Conversely, a value too high will not embed enough cells and may result in critical flow features remaining undetected. For the current study, it has been found that specifying $\bar{\beta}^*$ to be approximately 0.1 generally gave acceptable results.

After flagging cells, voids in the new grid topology can exist as shown in figure 17. These voids occur in areas of the grid where the refinement parameter $\bar{\beta}$ is close to the threshold value $\bar{\beta}^*$ causing a scattering of the flagged cells over the parent grid. Voids in the adapted grid region can be avoided by eliminating undesirable cell arrangements using a mapping function along with the face-to-cell pointers to add new cells. This mapping function is illustrated in figure 18 in which five undesirable cell arrangements have been identified and a more desirable cell arrangement is suggested for each. This process of eliminating voids is referred to as grid amalgamation.

After the grid amalgamation process, the ghost cells for the embedded interfaces must be created. These cells are not constructed in the same manner as the computational embedded cells since they are updated by interpolation and not through the time advancement scheme. Only the nodes and link pointers are generated for these cells. The link pointers are needed for the bilinear interpolation. The link pointers for the parent cell do not point to the children who are embedded ghost cells since no information is passed from the children to the parents for these cells (such as the relative truncation error).

After the cells are flagged for adaptation, new nodes must be generated in order to refine the grid in these regions. The generation of new nodes, however, is somewhat difficult since it is necessary to maintain both the grid smoothness and stretching of the parent grid. This is particularly true for cells which lie on the surface of the geometry since nonsmooth grid lines can result in oscillations in

the solution. The most straightforward approach to generating new grid points is to simply average the coordinates of the neighboring nodes. This technique, however, does not resolve the curvature of the grid lines and will produce discontinous metrics. Another method for generating new points is to use an interpolation function, such as cubic splines. However, this approach presents two problems. The first problem arises if there is an inadequate number of defining points in the initial grid so that the resulting spline coefficients may not represent the true grid shape accurately. The second problem is caused by the pointer structure used for the unstructured grids which only provides local information on the connectivity of faces and does not provide the global connectivity required to implement an interpolation function such as cubic splines. This could be circumvented by creating the connectivity required but would add to the complexity of the data structure and would not eliminate the first problem of the initial coarse grid.

The current approach used to construct the embedded regions is to create the finest level grid with a structured grid generation package and extract the nodes from this fine grid data file as needed. This approach, although costly, ensures that the new grids maintain the same stretching and smoothness.

After the embedded regions are generated, the solution from the previous grid is interpolated to the new cells using the same bilinear interpolation operators used by the multi-grid algorithm. The solution is then advanced in time using the multigrid method of section 3.6. The overall adaptation process for a single grid level is illustrated in figure 19.

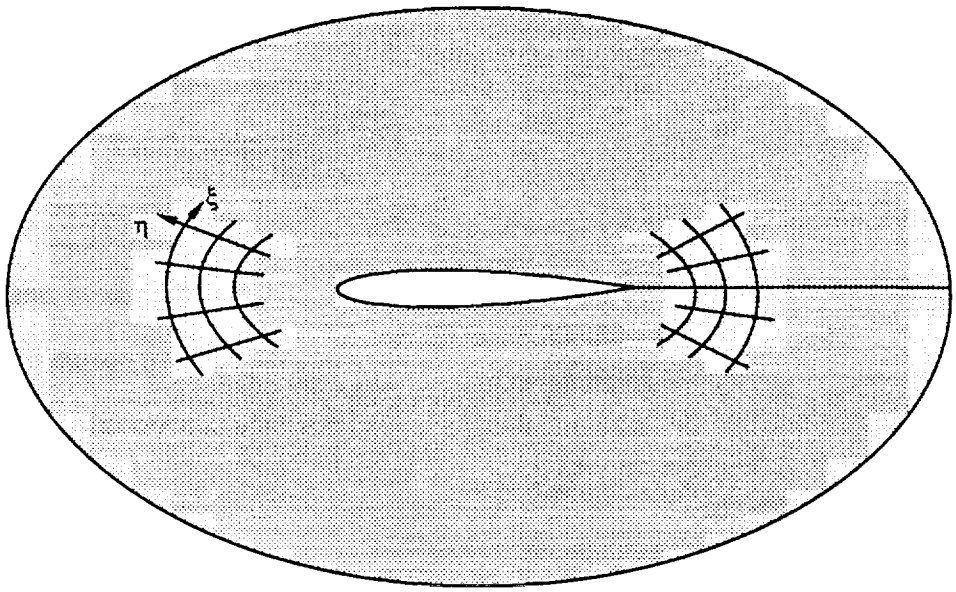


Figure 16. O-Grid Topology

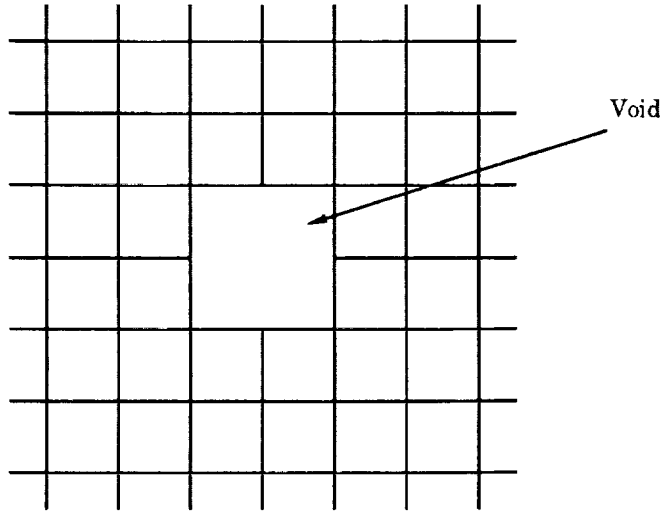


Figure 17. Voids in Grid

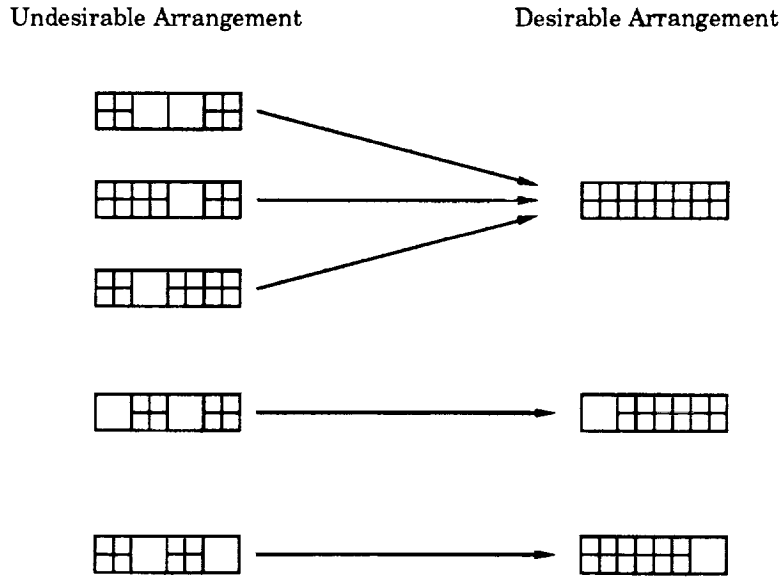


Figure 18. Grid Remapping Criteria

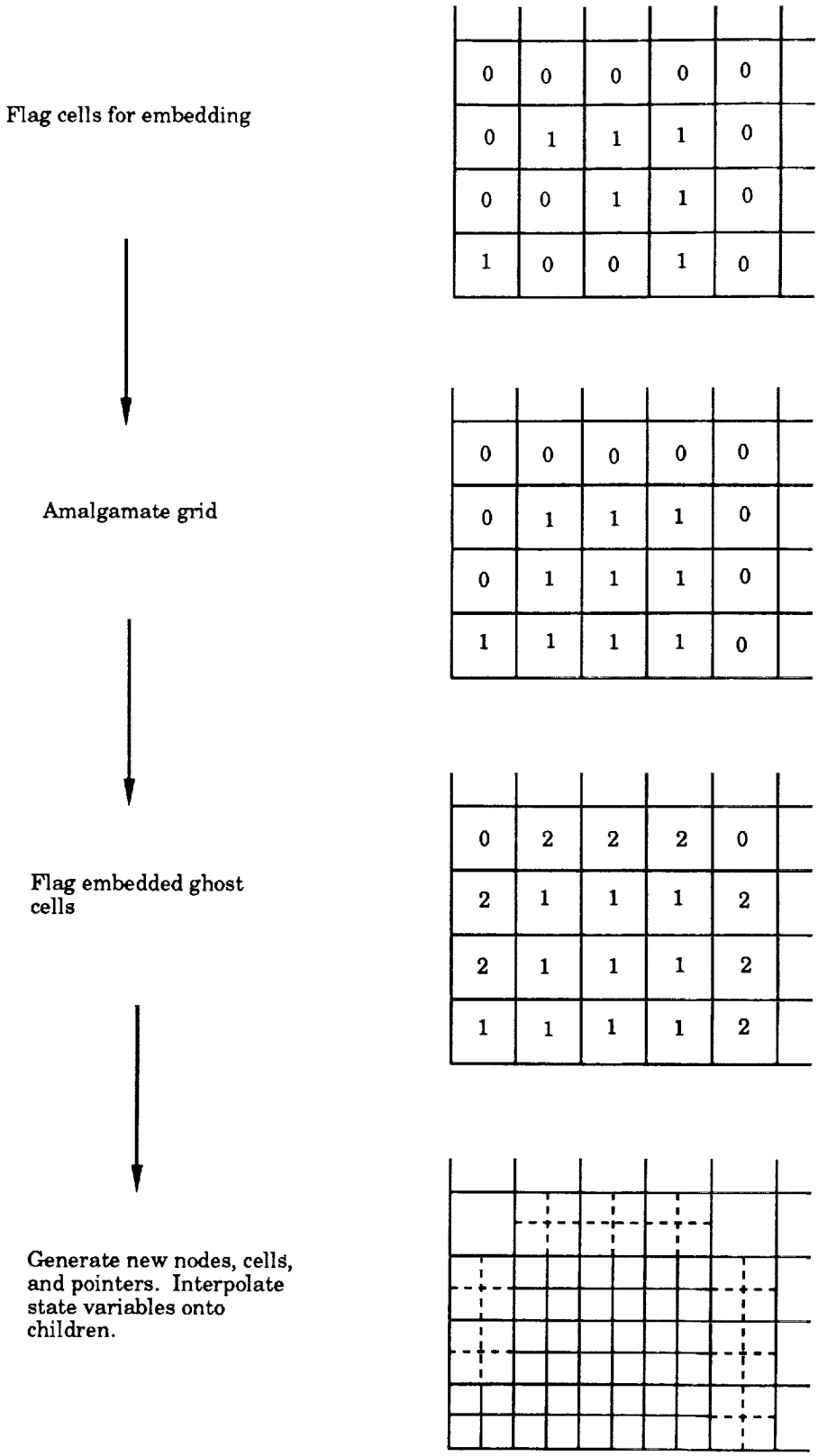


Figure 19. Grid Adaptation Steps

CHAPTER V

RESULTS

Results are now presented for the adaptive semi-unstructured Euler solver. The computer code for this algorithm used 105 storage locations for each computational cell. The iterative scheme of the Euler solver is completely vectorizable and has a computational rate of 14 microseconds per cell per iteration for a single grid level on the Cray–2 supercomputer at NASA Langley Research Center. Implementation of the multigrid algorithm with five grid levels requires approximately 30 microseconds per cell per iteration but this is dependent on the number of embedded ghost cells in the computational domain. All of the cases presented here were run using $\kappa = \frac{1}{3}$ for equations 3.7 and 3.8.

Comparisons are made with the two–dimensional Euler code of reference 4 referred to as CFL2D. As mentioned previously, the flux integration techniques for this code are the same as the present adaptive code. The version of CFL2D used for comparisons requires 139 storage locations per cell and has a computational rate of approximately 45 microseconds per cell per iteration. This Euler code uses an implicit time advancing scheme and has better smoothing characteristics for the multigrid algorithm than the explicit scheme used in the adaptive code. Hence, on a given structured grid, CFL2D will converge in less cpu time than the adaptive code.

The grid used with CFL2D was a 257x97 O-grid with an outer boundary 40 chords away from the airfoil. This was the same fine grid used for extracting points for the adaptive Euler code. The solutions presented here were considered converged after the rms of the average residual had been reduced by at least 7 orders of magnitude which generally required 2000 iterations for the adaptive code (1 V-cycle = 1 iteration).

5.1 NACA 0012, Mach = $0.63, \alpha = 2.0°$

The NACA 0012 airfoil in a freestream Mach number of 0.63 and at 2.0 degrees angle–of–attack is presented here. This is an isentropic case and should produce a drag coefficient of zero. Figure 20 shows the initial grid for the adaptive Euler code which is a 33x13 structured O-grid. The surface pressure coefficients are shown in figure 21 and the Mach number contours are shown in figures 22 and 23. The coarseness of the grid produces large entropy values near the leading edge of the airfoil and degrades the solution downstream which is evident from the poor resolution of the trailing edge stagnation point.

The first adaptive embedding was performed using the undivided gradient of the density magnitude $| \bar{\nabla}\rho |$ with the threshold value $\bar{\beta}^*$ set to 0.1. The solution was converged on each level before the next level of embedded grids was created. Figures 24 through 29 show each successive level of refinement with the corresponding surface pressures. As shown, the embedding significantly improves the overall solution quality without a large increase in the number of cells. Figures

36 and 37 show the Mach contours on the final adapted grid. These contours show the smooth transition of the solution at the embedded interfaces.

The second adaptive embedding case was performed using the undivided gradient of the velocity magnitude $|\tilde{\nabla} q|$ with $\bar{\beta}^*$ again set to 0.1. The computations were carried out in the same manner as the previous case with the resulting grids and surface pressures shown in figures 30 through 35. As shown, this refinement parameter embedded more points than the density gradient for the same value of $\bar{\beta}^*$. Table 5.1 shows a comparison of the lift and drag values obtained for the two cases along with those obtained with CFL2D.

5.2 NACA 0012, Mach = $0.8, \alpha = 1.25°$

The NACA 0012 airfoil in a freestream Mach number of 0.80 and at 1.25 degrees angle of attack is presented here. Using $|\tilde{\nabla}\rho|$ or $|\tilde{\nabla} q|$ produced nearly identical grids for this case and so only the $|\tilde{\nabla} q|$ case is shown.

These freestream conditions produce a strong shock on the upper surface of the airfoil with a relatively weak shock on the lower surface. The Mach contours and pressure distribution for the initial grid are shown in figures 38 and 39. The final adapted grid, pressure distribution, and Mach contours are shown in figures 40 through 42. As shown, the final grid produces a sharp shock on both the upper and lower surfaces with no oscillations. Table 5.2 shows the comparison with CFL2D. The final c_ℓ has about a $4\frac{1}{2}$ % difference from that of CFL2D.

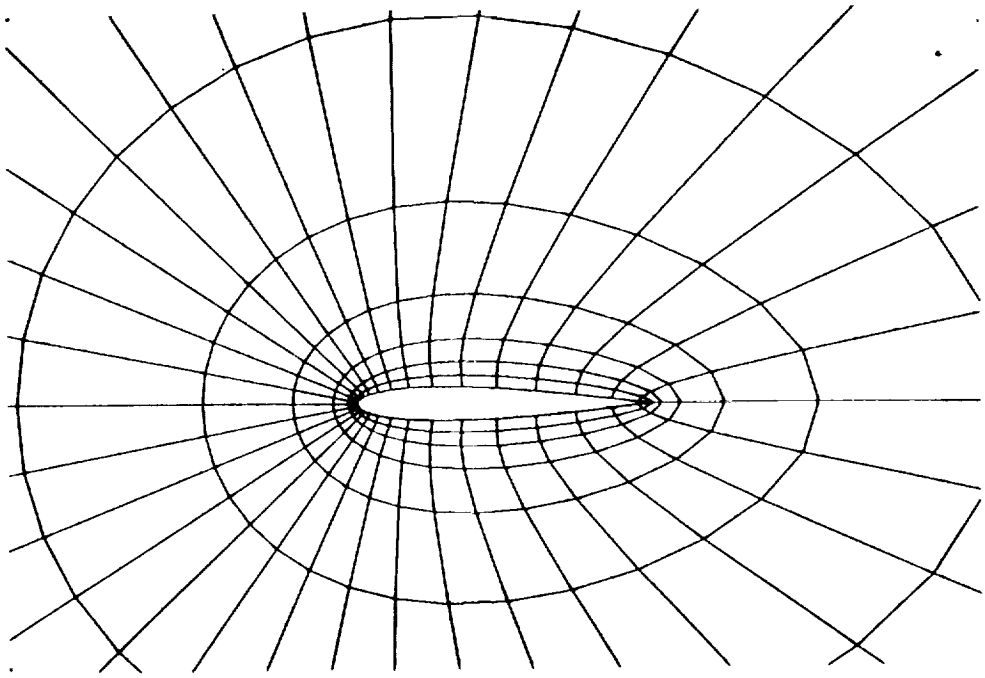


Figure 20. Initial Grid

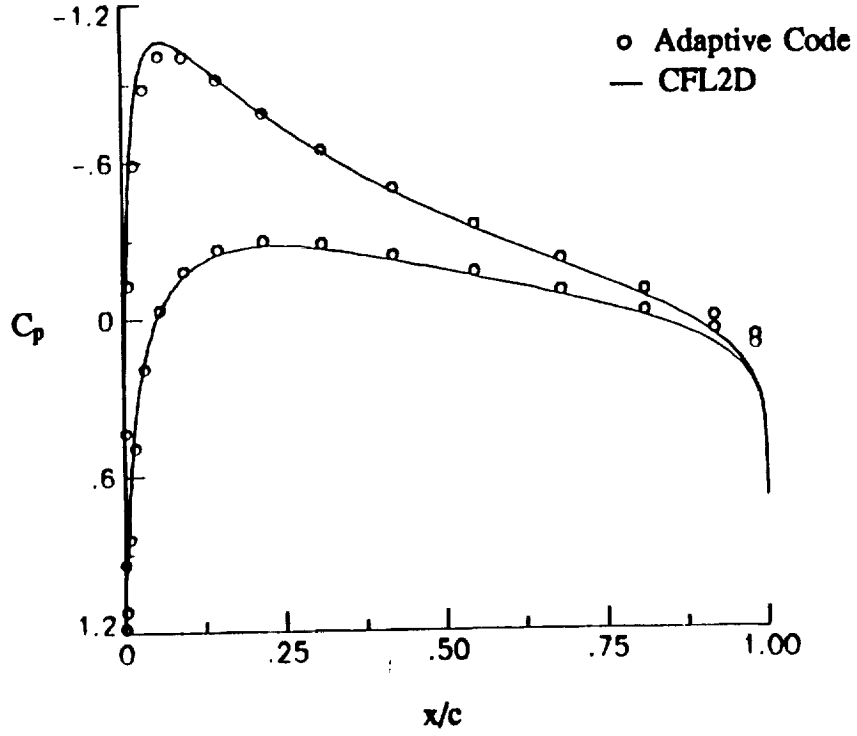


Figure 21. Initial Grid Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

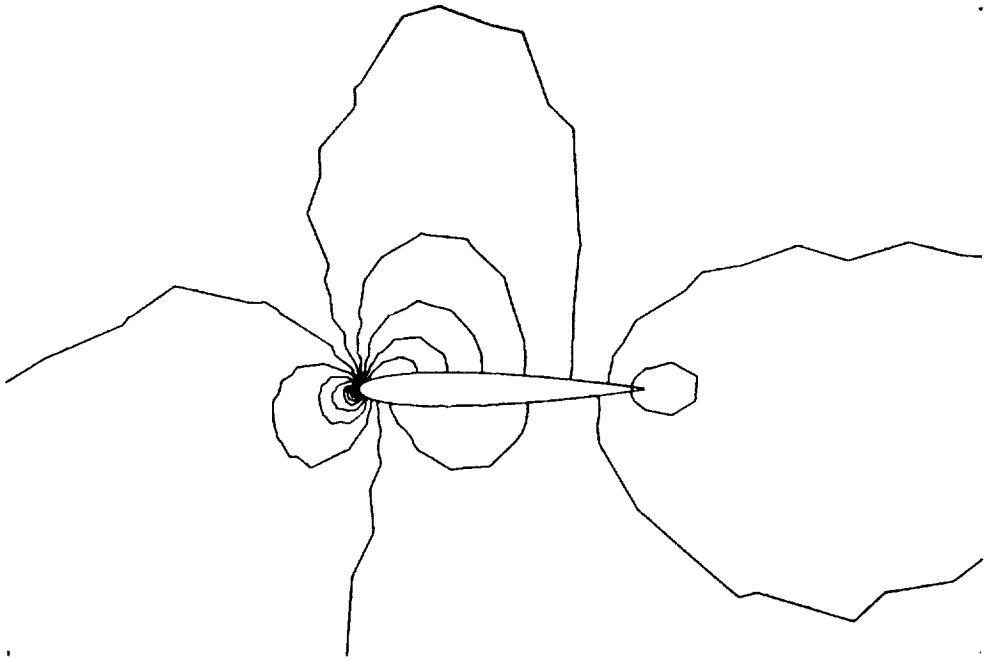


Figure 22. Initial Grid Mach Contours, $M_\infty = 0.63, \alpha = 2.0°, \Delta M = 0.05$

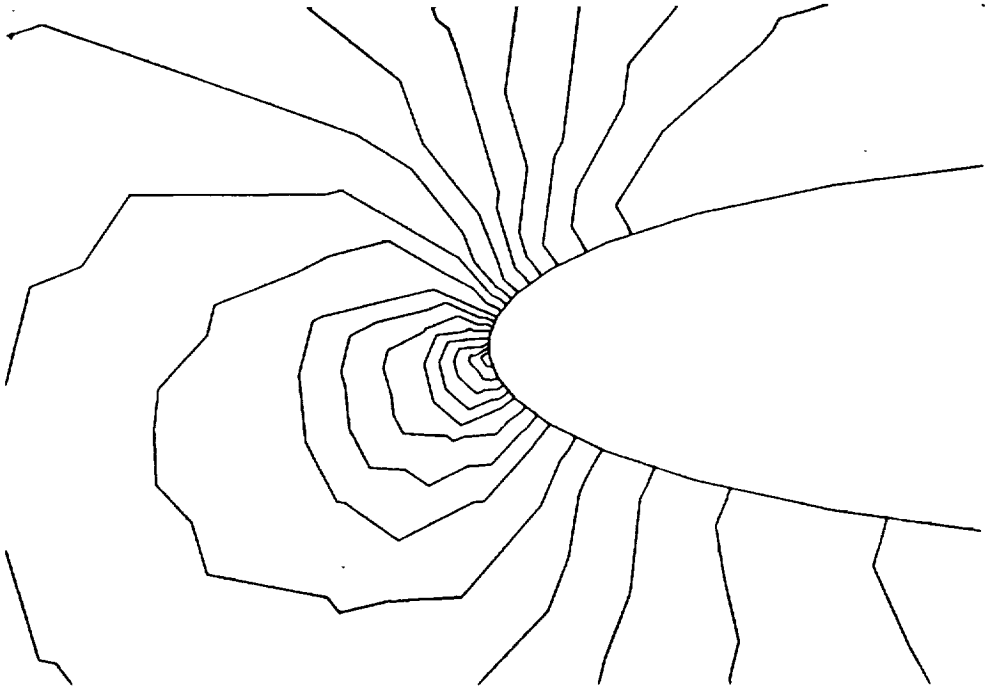


Figure 23. Leading Edge Initial Grid Mach Contours, $M_\infty = 0.63, \alpha = 2.0°$

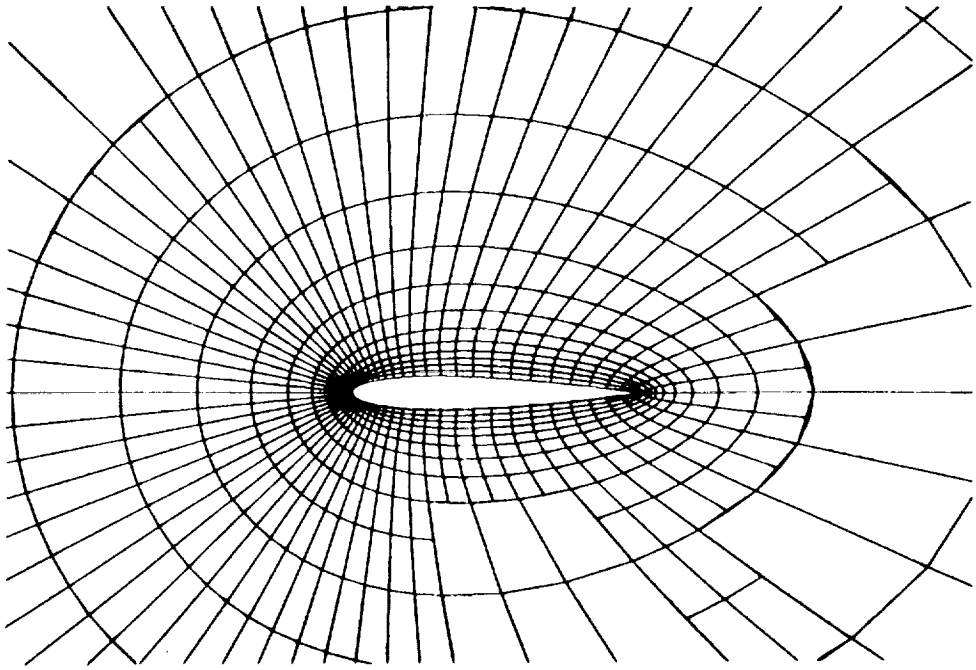


Figure 24. 2 Level Adapted Grid ($\bar{\nabla}\rho$), $M_\infty = 0.63, \alpha = 2.0°$

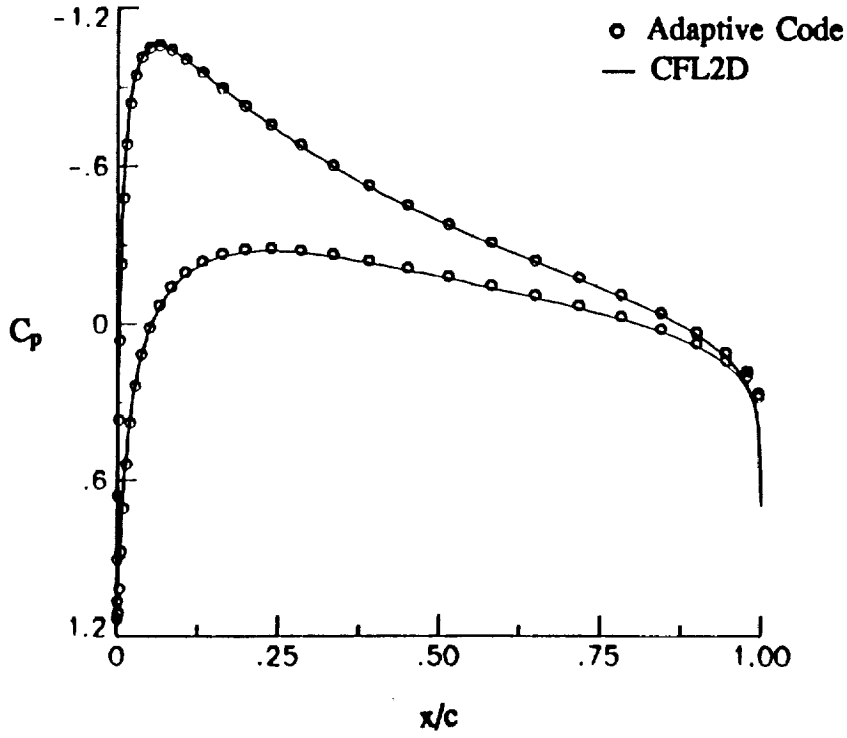


Figure 25. 2 Level ($\bar{\nabla}\rho$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

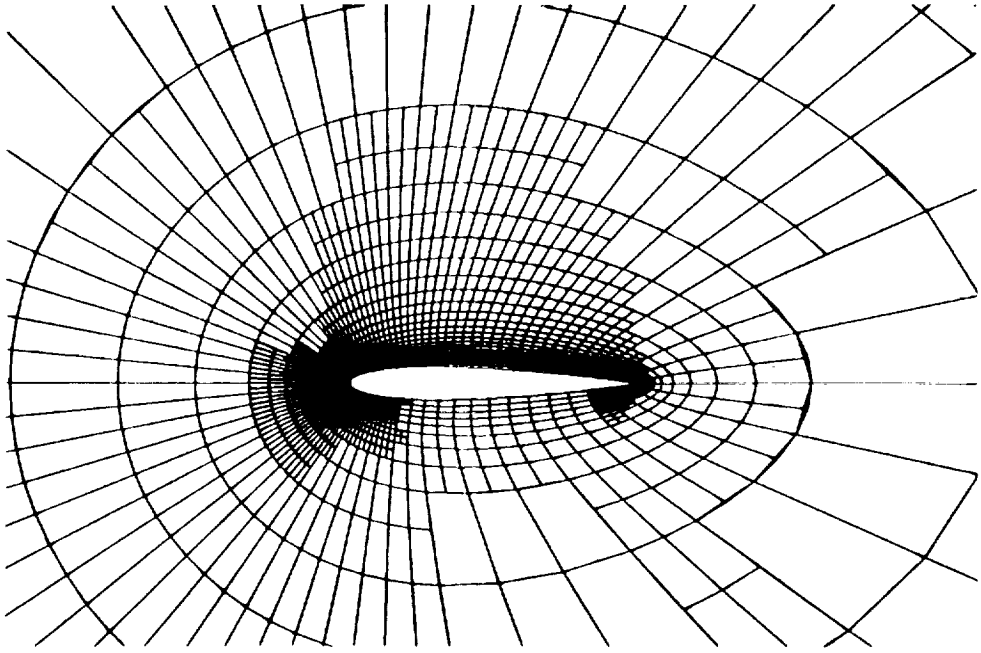


Figure 26. 3 Level Adapted Grid ($\tilde{\nabla}\rho$), $M_\infty = 0.63, \alpha = 2.0°$

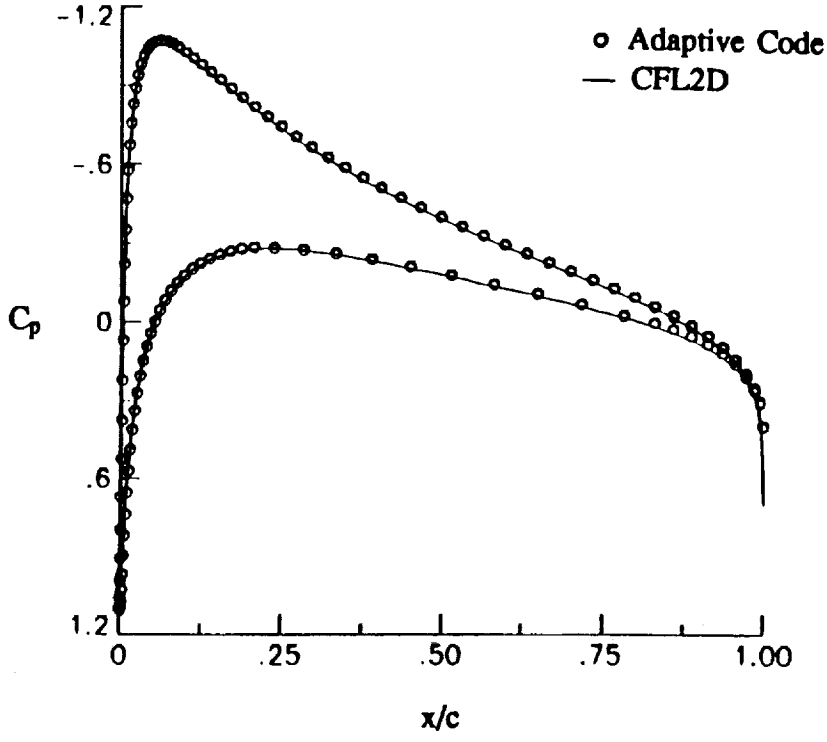


Figure 27. 3 Level ($\tilde{\nabla}\rho$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

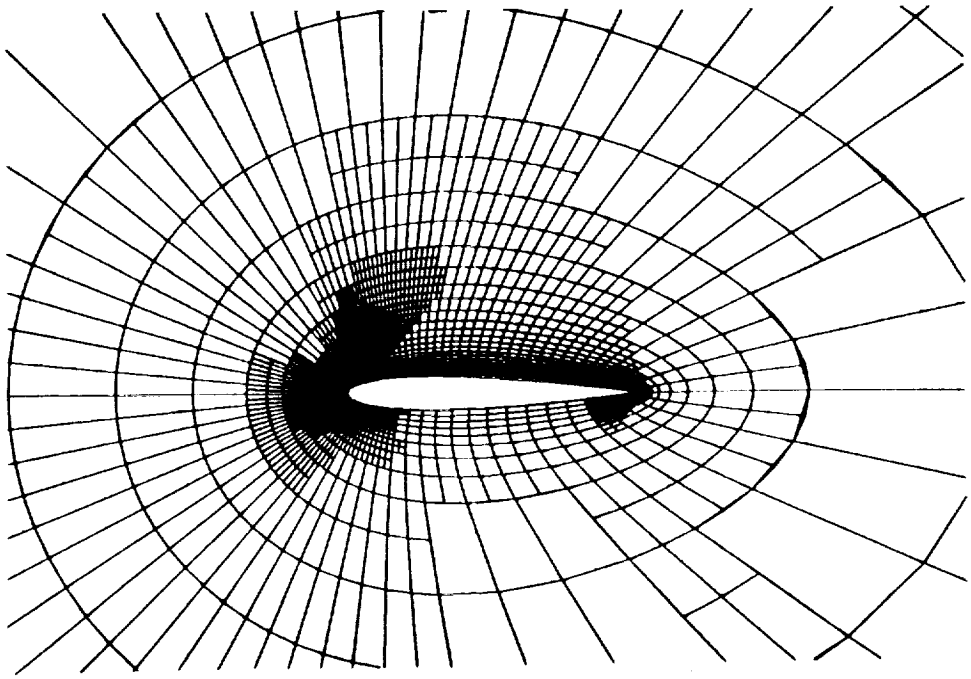


Figure 28. 4 Level Adapted Grid ($\tilde{\nabla}\rho$), $M_\infty = 0.63, \alpha = 2.0°$

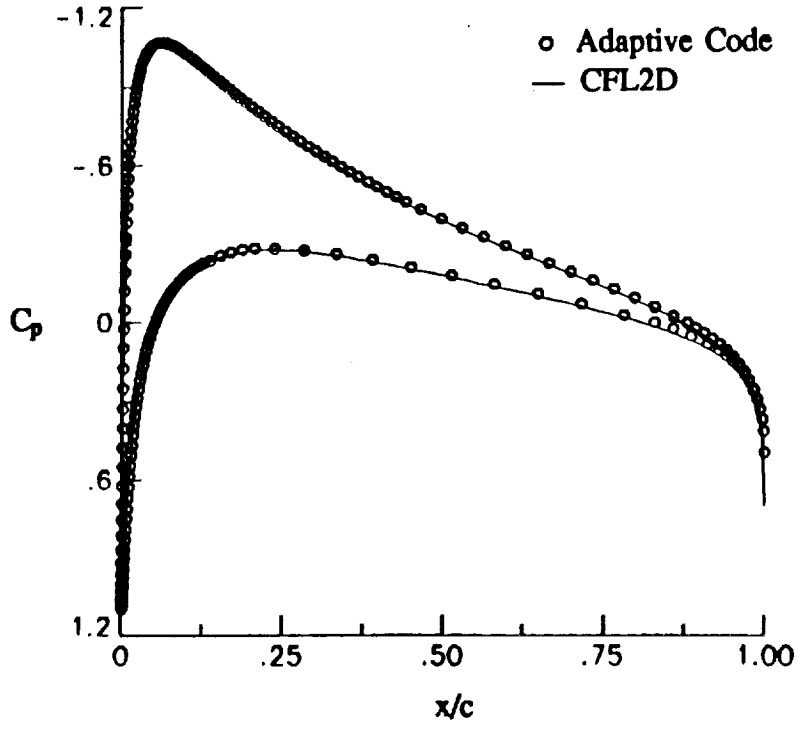


Figure 29. 4 Level ($\tilde{\nabla}\rho$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

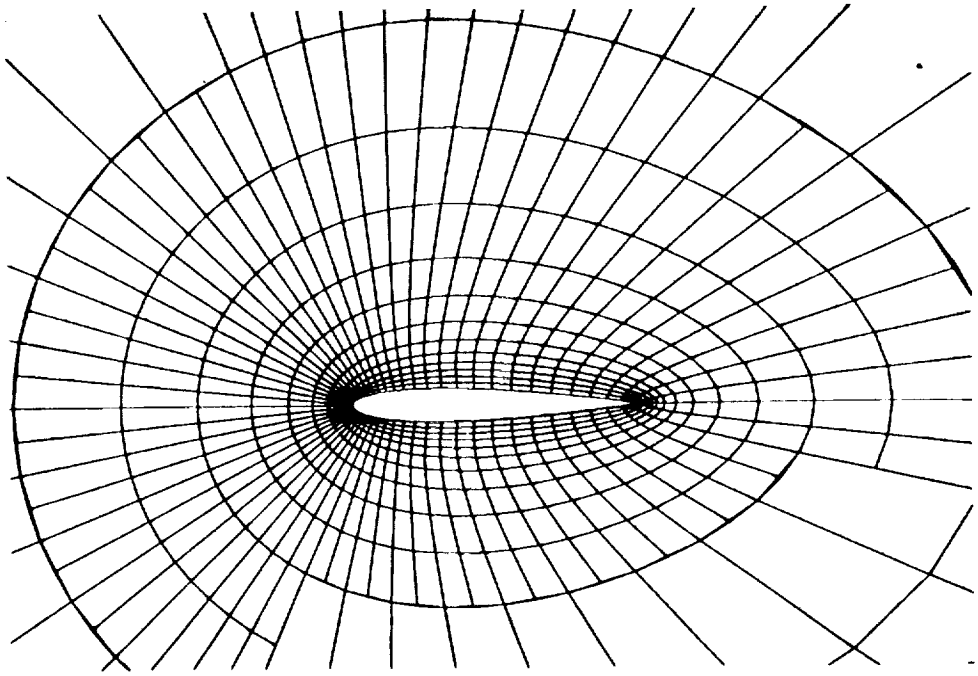


Figure 30. 2 Level Adapted Grid ($\tilde{\nabla} q$), $M_\infty = 0.63, \alpha = 2.0°$

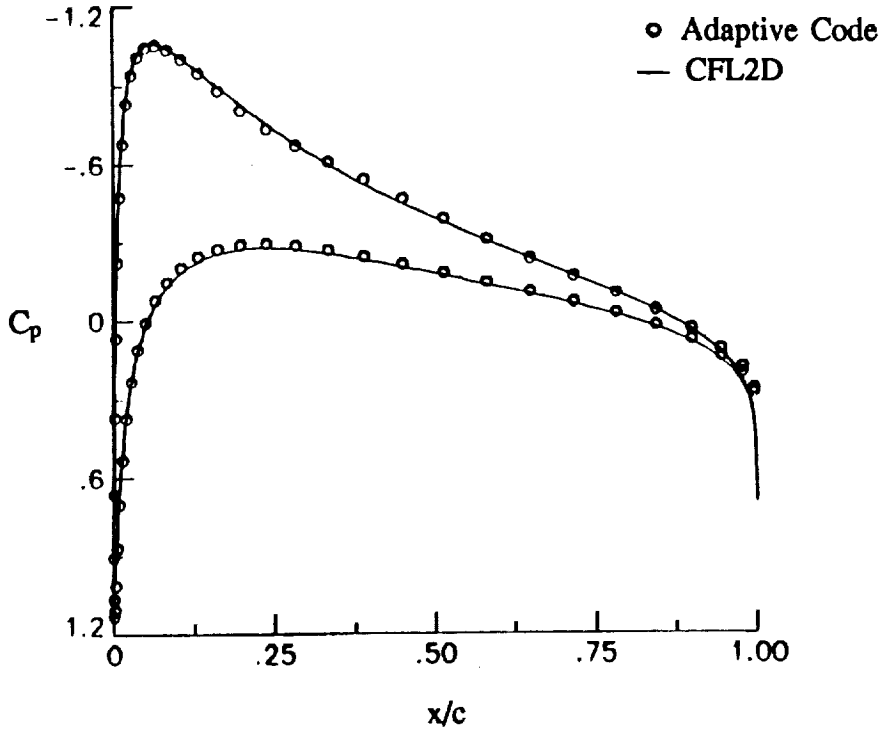


Figure 31. 2 Level ($\tilde{\nabla} q$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

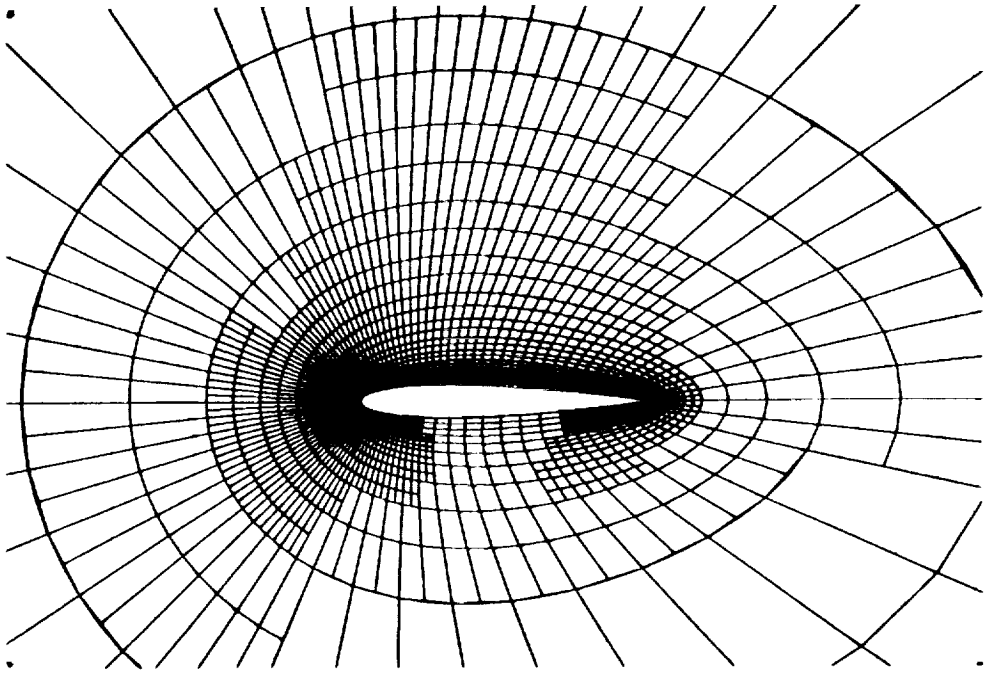


Figure 32. 3 Level Adapted Grid ($\bar{\nabla} q$), $M_\infty = 0.63, \alpha = 2.0°$

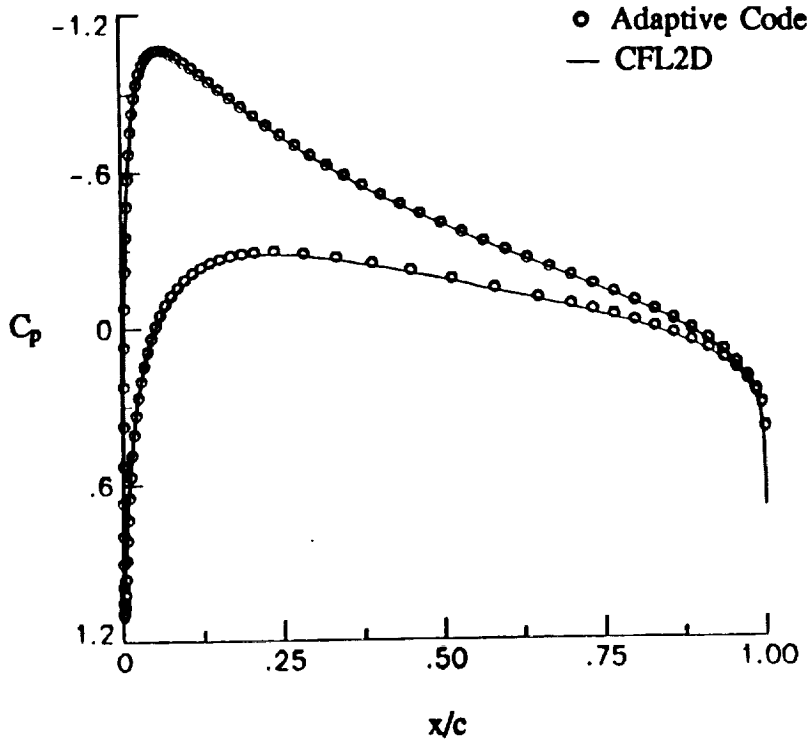


Figure 33. 3 Level ($\bar{\nabla} q$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

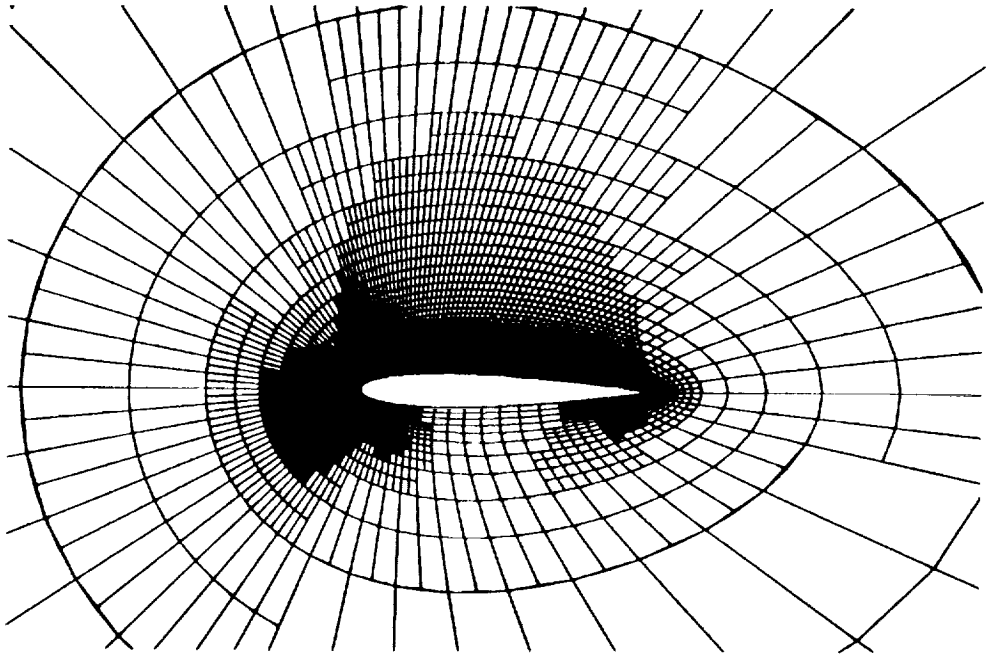


Figure 34. 4 Level Adapted Grid ($\tilde{\nabla} q$), $M_\infty = 0.63, \alpha = 2.0°$

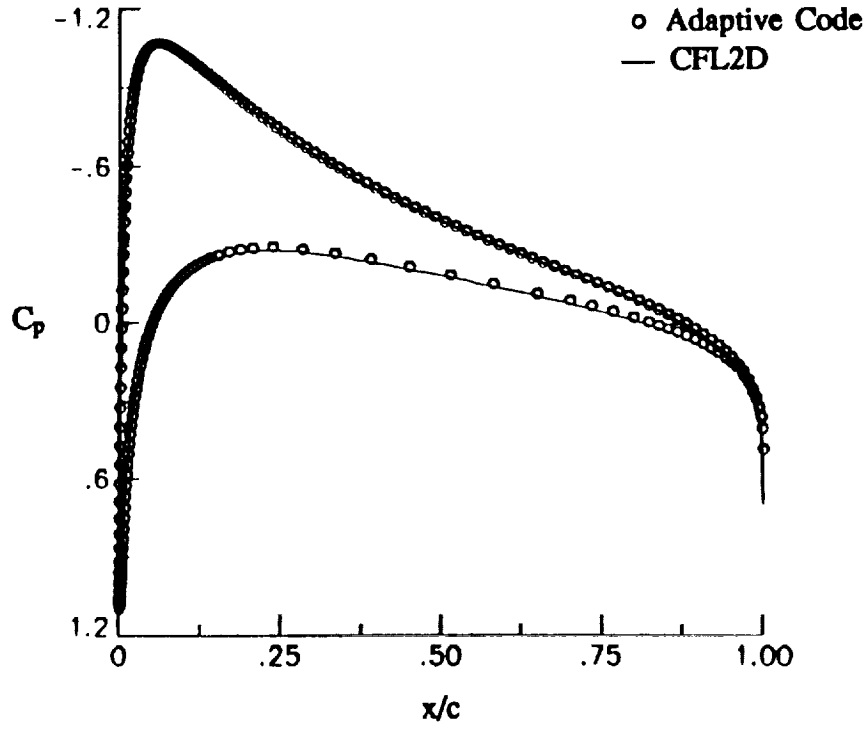


Figure 35. 4 Level ($\tilde{\nabla} q$) Pressure Distribution, $M_\infty = 0.63, \alpha = 2.0°$

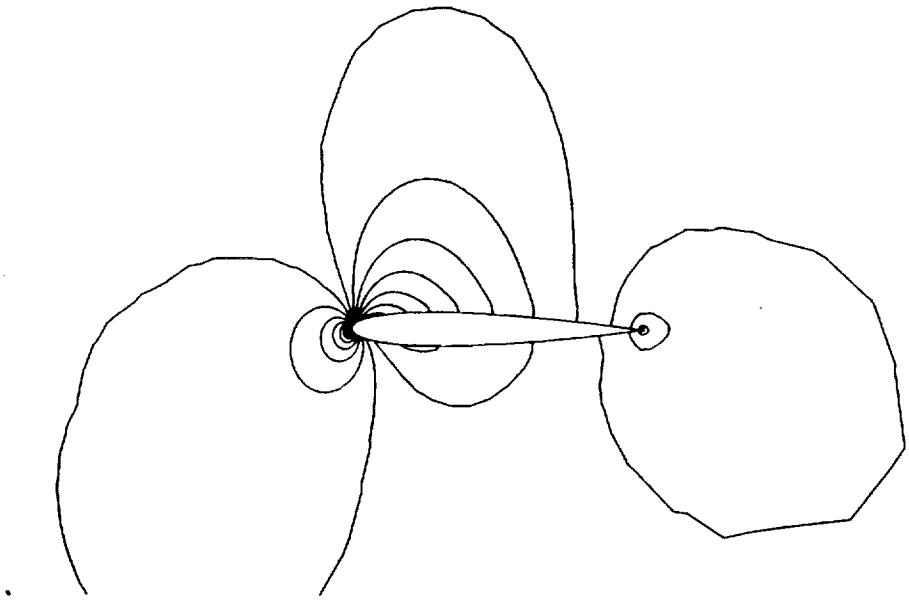


Figure 36. 4 Level ($\tilde{\nabla}\rho$) Mach Contours, $M_\infty = 0.63, \alpha = 2.0°, \Delta M = 0.05$

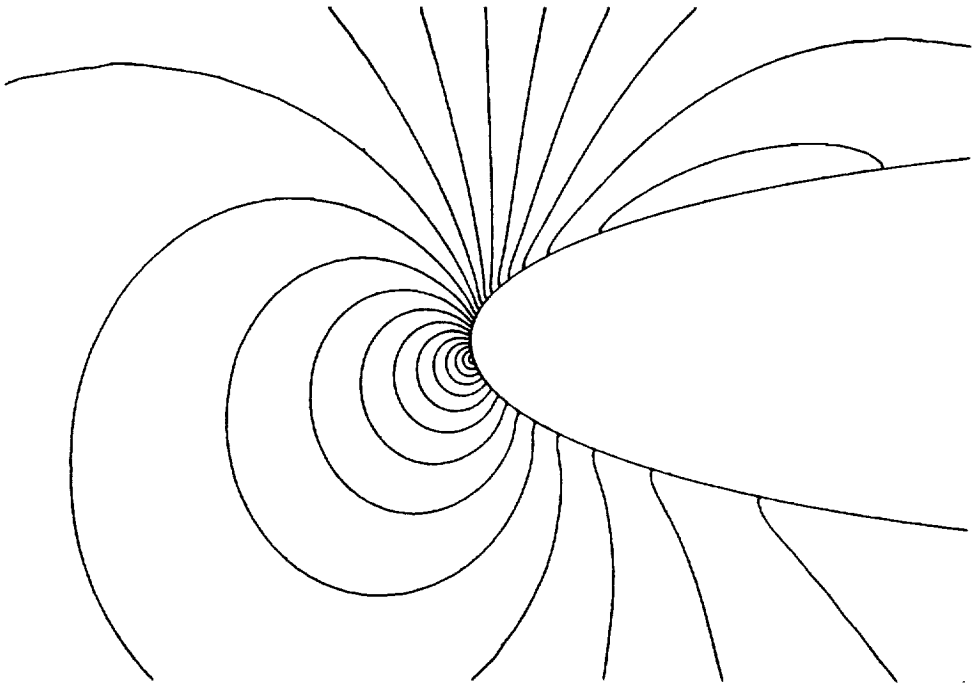


Figure 37. Leading Edge 4 Level ($\tilde{\nabla}\rho$) Grid Mach Contours, $M_\infty = 0.63, \alpha = 2.0°$

Adaptation Criteria	Number of Cells	c_l	c_d
None (CFL2D)	32,298	.328	0.270×10^{-3}
∇q	11,132	.320	0.249×10^{-3}
$\nabla \rho$	7,648	.323	0.769×10^{-3}

Table 5.1. Results Comparison for $M_\infty = 0.63, \alpha = 2.0°$

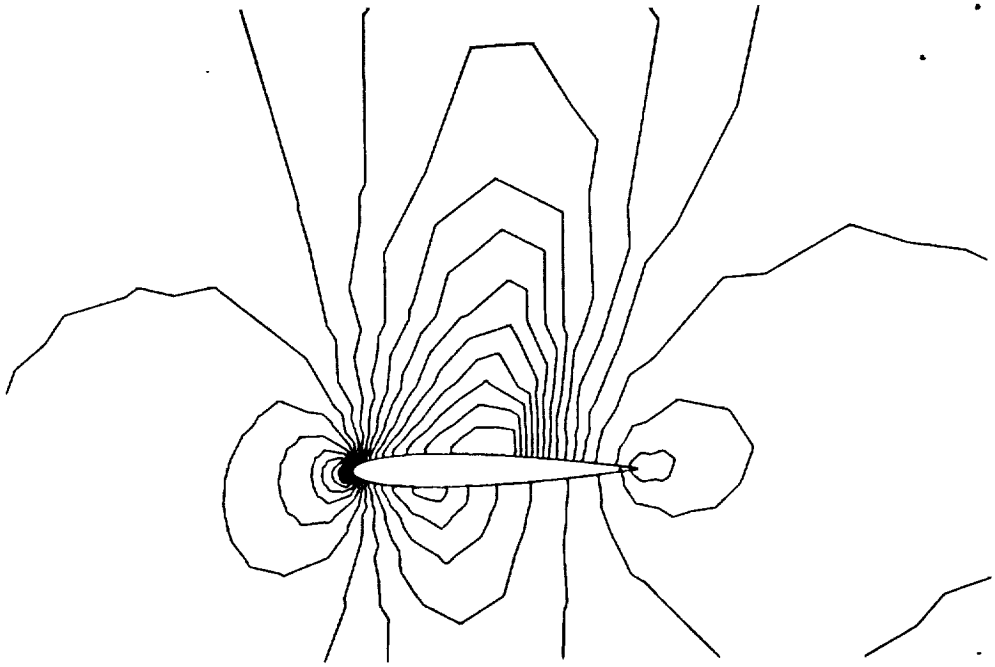


Figure 38. Initial Grid Mach Contours, $M_\infty = 0.80, \alpha = 1.25°$

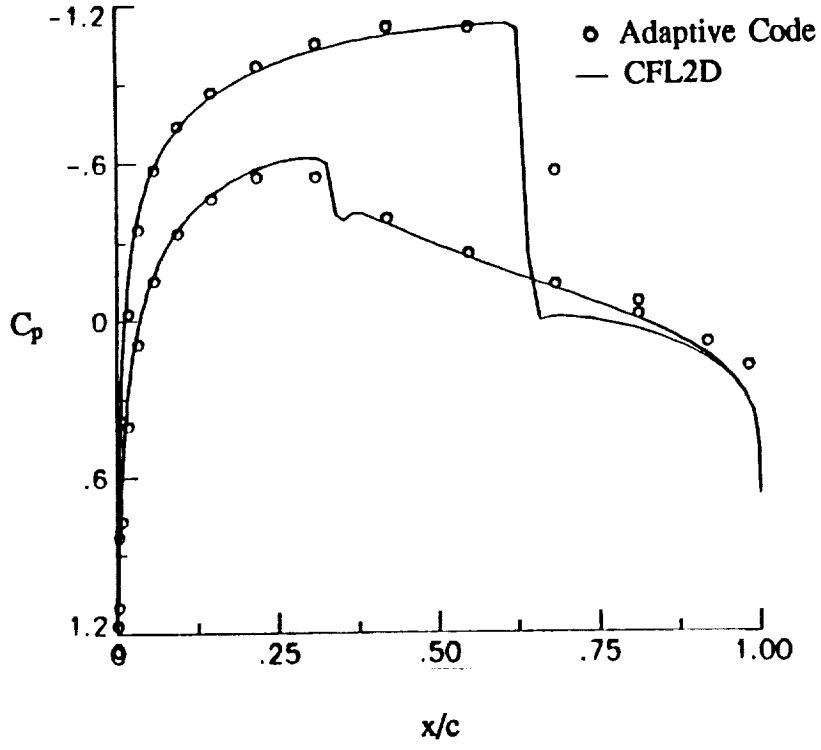


Figure 39. Initial Grid Pressure Distribution, $M_\infty = 0.80, \alpha = 1.25°$

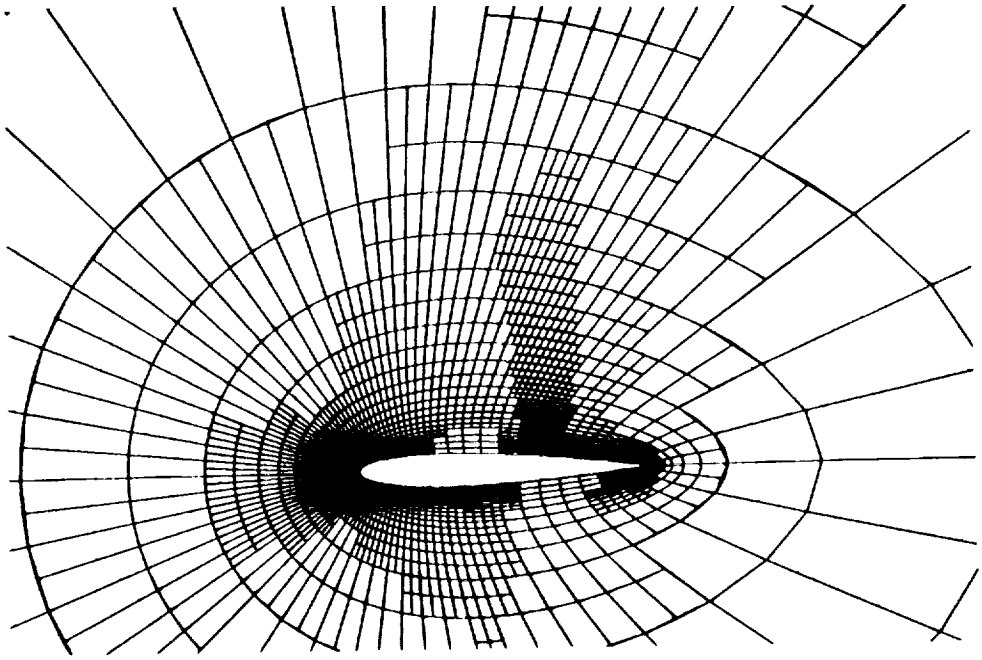


Figure 40. 4 Level Adapted Grid ($\tilde{\nabla} q$), $M_\infty = 0.80, \alpha = 1.25°$

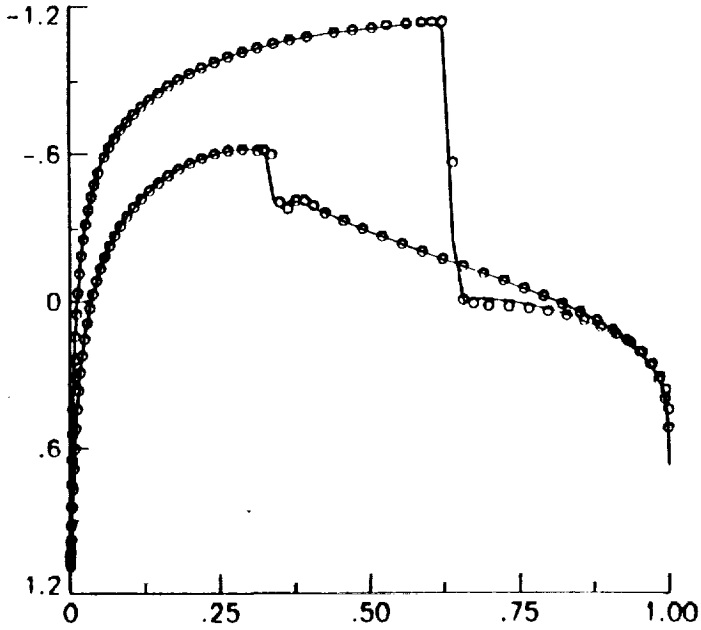


Figure 41. 4 Level ($\tilde{\nabla} q$) Pressure Distribution, $M_\infty = 0.80, \alpha = 1.25°$

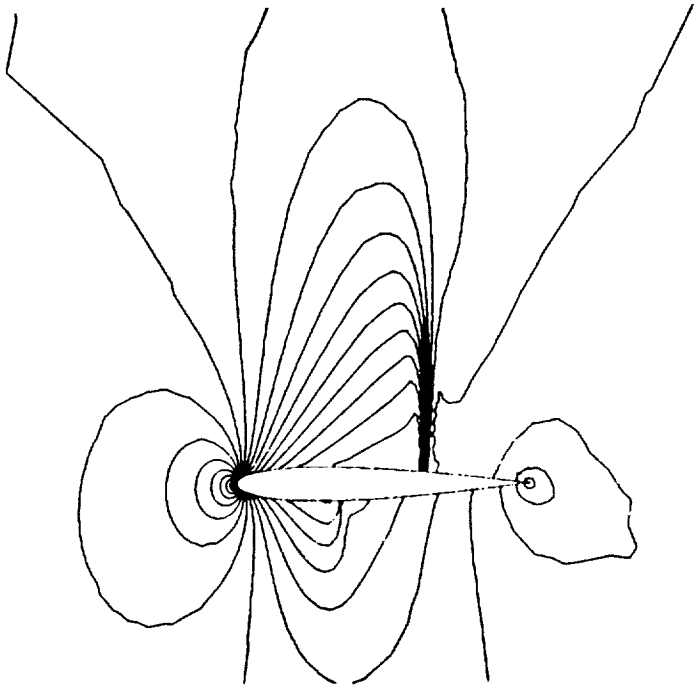


Figure 42. 4 Level ($\tilde{\nabla} q$) Mach Contours, $M_\infty = 0.80, \alpha = 1.25°$

Adaptation Criteria	Number of Cells	c_l	c_d
None (CFL2D)	32,298	.352	0.211×10^{-1}
∇q	11,132	.337	0.251×10^{-1}

Table 5.2. Results Comparison for $M_\infty = 0.80, \alpha = 1.25°$

CHAPTER VI

CONCLUDING REMARKS

A semi-unstructured algorithm to solve the two-dimensional Euler equations with adaptive grid embedding has been presented. A multigrid algorithm has been implemented to provide the required grid level communication and to accelerate the convergence.

Results have been shown for a subcritical airfoil and comparisons have been made with a structured algorithm using the same flux integration techniques. The final lift coefficients are within $2\frac{1}{2}\%$ of those obtained with CFL2D. Results have also been shown for a transonic airfoil where the increased resolution of the shocks is very substantial without the large increase in grid points. The final lift coefficients for this case are around $4\frac{1}{2}\%$ of CFL2D.

One problem with the present method is the poor error-damping characteristics of the two-stage time advancing scheme. Further research is needed in this area to produce an acceptable scheme. The memory requirements on a per cell basis for the present method are approximately the same as that for an implicit scheme. Since the present method requires less grid points to resolve the flow, the overall memory requirements are less. The advantages of using this scheme would probably be greater in three-dimensions.

APPENDIX A

Interpolation Operators

Interpolation can be performed in a two-dimensional field, shown in figure 43, by assuming a bilinear function of the form:

$$f(x,y) = a + bx + cy + dxy \tag{A.1}$$

The constants a, b, c, and d, can be evaluated given four boundary values:

$$f(x_1,y_1) = a + bx_1 + cy_1 + dx_1y_1 \tag{A.2a}$$

$$f(x_2,y_2) = a + bx_2 + cy_2 + dx_2y_2 \tag{A.2b}$$

$$f(x_3,y_3) = a + bx_3 + cy_3 + dx_3y_3 \tag{A.2c}$$

$$f(x_4,y_4) = a + bx_4 + cy_4 + dx_4y_4 \tag{A.2d}$$

This can be written in matrix form as:

$$\begin{pmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} f(x_1,y_1) \\ f(x_2,y_2) \\ f(x_3,y_3) \\ f(x_4,y_4) \end{pmatrix} \tag{A.3}$$

This matrix can be simplified if (x_1, y_1) is defined to be $(0,0)$. Equation $(A.2a)$ is reduced to:

$$f(x_1,y_1) = a \tag{A.4}$$

This reduces $(A.3)$ to a 3x3 matrix.

$$\begin{pmatrix} x_2 & y_2 & x_2y_2 \\ x_3 & y_3 & x_3y_3 \\ x_4 & y_4 & x_4y_4 \end{pmatrix} \begin{pmatrix} b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \Delta f_2 \\ \Delta f_3 \\ \Delta f_4 \end{pmatrix} \tag{A.5}$$

where:

$$\Delta f_n = f(x_n, y_n) - f(x_1, y_1)$$

The system of equations given in $(A.5)$ can now easily be solved by the use of Kramer's rule. The determinant of the coefficient matrix in $(A.5)$ is given as:

$$A = \begin{vmatrix} x_2 & y_2 & x_2y_2 \\ x_3 & y_3 & x_3y_3 \\ x_4 & y_4 & x_4y_4 \end{vmatrix} = x_2x_4y_3(y_4 - y_2) + x_2x_3y_4(y_2 - y_3) + x_3x_4y_2(y_3 - y_4) \quad (A.6)$$

The coefficients can then be solved for:

$$b = \frac{\begin{vmatrix} \Delta f_2 & y_2 & x_2y_2 \\ \Delta f_3 & y_3 & x_3y_3 \\ \Delta f_4 & y_4 & x_4y_4 \end{vmatrix}}{A}$$

$$= \frac{y_3y_4(x_4 - x_3)\Delta f_2 + y_2y_4(x_2 - x_4)\Delta f_3 + y_2y_3(x_3 - x_2)\Delta f_4}{A} \quad (A.7a)$$

$$c = \frac{\begin{vmatrix} x_2 & \Delta f_2 & x_2y_2 \\ x_3 & \Delta f_3 & x_3y_3 \\ x_4 & \Delta f_4 & x_4y_4 \end{vmatrix}}{A}$$

$$= \frac{x_3x_4(y_3 - y_4)\Delta f_2 + x_2x_4(y_4 - y_2)\Delta f_3 + x_2x_3(y_2 - y_3)\Delta f_4}{A} \quad (A.7b)$$

$$d = \frac{\begin{vmatrix} x_2 & y_2 & \Delta f_2 \\ x_3 & y_3 & \Delta f_3 \\ x_4 & y_4 & \Delta f_4 \end{vmatrix}}{A}$$

$$= \frac{(x_3y_4 - x_4y_3)\Delta f_2 + (x_4y_2 - x_2y_4)\Delta f_3 + (x_2y_3 - x_3y_2)\Delta f_4}{A} \quad (A.7c)$$

Assuming a uniform Cartesian grid of unit width and heighth, the nodes would be:

$$\begin{aligned} (x_1, y_1) &= (0, 0) \\ (x_2, y_2) &= (1, 0) \\ (x_3, y_3) &= (1, 1) \\ (x_4, y_4) &= (0, 1) \end{aligned} \quad (A.8)$$

Substituting ($A.8$) into ($A.7$) yields the following solution:

$$\begin{aligned} a &= f(x_1, y_1) \\ b &= \Delta f_2 \\ c &= \Delta f_4 \\ d &= -\Delta f_2 + \Delta f_3 - \Delta f_4 \end{aligned} \tag{A.9}$$

Substituting ($A.9$) into ($A.1$) yields:

$$\begin{aligned} f(x,y) =& f(x_1,y_1) + (f(x_2,y_2) - f(x_1,y_1))x + (f(x_4,y_4) - f(x_1,y_1))y \\ &+ (-(f(x_2,y_2) - f(x_1,y_1)) \\ &+ (f(x_3,y_3) - f(x_1,y_1)) - (f(x_4,y_4) - f(x_1,y_1)))xy \end{aligned} \tag{A.10}$$

Simplifying this gives:

$$\begin{aligned} f(x,y) = (1 - x - y + xy)f(x_1,y_1) + (x - xy)f(x_2,y_2) \\ + (xy)f(x_3,y_3) + (y - xy)f(x_4,y_4) \end{aligned} \tag{A.11}$$

For a Cartesian grid as shown in figure 44, $(x, y) = (0.25, 0.25)$, so that

$$\begin{aligned} f(0.25,0.25) = (0.5625)f(x_1,y_1) + (0.1875)f(x_2,y_2) \\ + (0.0625)f(x_3,y_3) + (0.1875)f(x_4,y_4) \end{aligned} \tag{A.12}$$

Although the grids are generally not Cartesian, this is a close approximation to the actual weightings for grids that are not highly stretched.

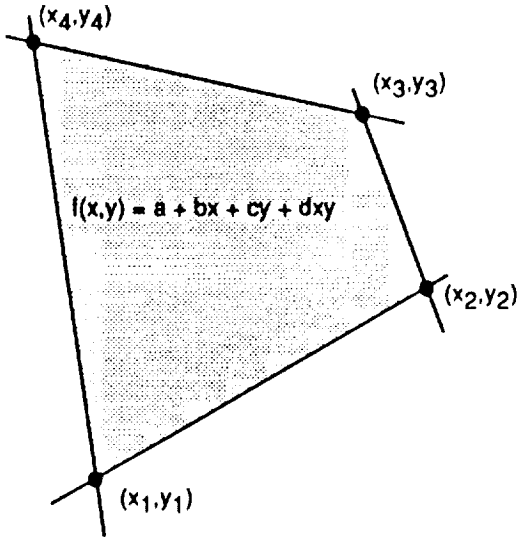


Figure 43. Typical Quadrilateral

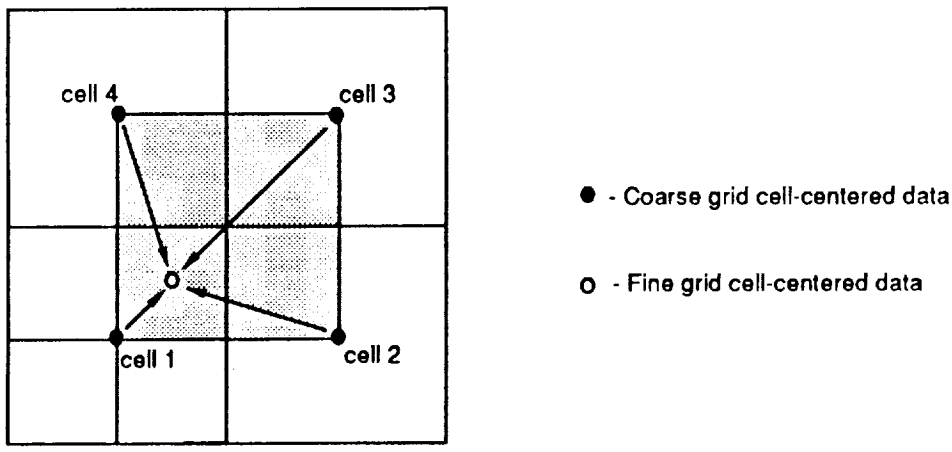


Figure 44. Quadrilateral in a Uniform Cartesian Grid

BIBLIOGRAPHY

1. Berger, M. J., Jameson, A., "Automatic Adaptive Grid Refinement for the Euler Equations," AIAA Journal, Vol. 24, No. 4, April 1985, pp. 561-567.

2. Dannenhoffer, J. F., and Baron, J. R., "Grid Adaptation for the 2-D Euler Equations," AIAA 85-0484, January 1985.

3. Dannenhoffer, J. F., "Grid Adaptation for Complex Two–Dimensional Transonic Flows." Sc.D. Thesis, Massachusetts Institute of Technology, August 1987.

4. Anderson, W. K., Thomas, J. L., and Whitfield, D. L.: *Three-Dimensional Multigrid Algorithms for the Flux-Split Euler Equations.* NASA TP-2829, 1988.

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

6. Van Leer, B.: Flux Vector Splitting for Euler Equations. Lecture Notes in Physics, Vol. 170, 1982, pp. 501-512.

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

8. Krist, S. L., Thomas, J. L., Sellers, W. L., Kjelgaard, S. O., "An Embedded Grid Formulation Applied to a Delta Wing," AIAA 90-0429, January 1990.

9. Shu, C. W., Osher, S., "Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes," ICASE Report No. 87-33, 1987.

10. International Mathematical and Statistical Library, Version 9, Houston, Texas.

11. Thomas, J. L., Salas, M. D., "Far-Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations," AIAA 85-1111, January 1985.

12. Thompson, J. F., Warsi, Z. U. A., Mastin, C. W.: *Numerical Grid Generation: Foundations and Applications.* North–Holland, New York, 1985.

13. Allmaras, S. R., Baron, J. R., "Embedded Mesh Solutions of the 2–D Euler Equations: Evaluation of Interface Formulations," AIAA 86-0509, January 1986.

14. Rai, M. M., "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations," AIAA 84-0164 January 1984.