A91-40717

AIAA-91-1542-CP

An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations

Darren De Zeeuw [*]

Kenneth G. Powell [†]

The University of Michigan

Department of Aerospace Engineering

Ann Arbor, MI 48109-2140

April, 1991

Abstract

A method for adaptive refinement of a Cartesian mesh for the solution of the steady Euler equations is presented. The algorithm creates an initial uniform mesh and cuts the body out of that mesh. The mesh is then refined based on body curvature. Next, the solution is converged to a steady state using a linear reconstruction and Roe's approximate Riemann solver. Solution-adaptive refinement of the mesh is then applied to resolve high-gradient regions of the flow. The numerical results presented show the flexibility of this approach and the accuracy attainable by solution-based refinement.

1 Introduction

In the past several years, unstructured mesh methods for fluid dynamics have become more and more prevalent, as a way of surmounting some of the difficulties of generating body-fitted meshes about arbitrary bodies[1, 2, 3, 4]. Flow-fields with multiple bodies are particularly difficult to generate structured meshes for, and multi-block methods [5] or patched-mesh methods [6] must be implemented in these cases if structured meshes are to be used.

Even unstructured mesh generation is not simple for complex configurations, however. Advancing front methods [2] must be carefully implemeted to avoid high aspect-ratio or highly skewed cells; Delaunay methods [3, 4] typically require the generation of a cloud of points to triangulate, and special steps must be taken to avoid the breaking of boundary faces.

*Doctoral Candidate, Member AIAA

†Assistant Professor, Member AIAA

A very simple mesh generation technique is to "cut" the bodies in the flow out of a Cartesian mesh. Indeed, this technique is commonly used for potential flow calculations [7]. Advantages of using Cartesian meshes for Euler solvers, besides increased ease of mesh generation, include simpler flux formulations and simplifications in the data structure. In addition, Cartesian cells lead to fortuitous cancellation of truncation errors not occuring on less regular meshes. Two difficulties that arise in developing an Euler solver for a Cartesian mesh are:

1. poor resolution of leading and trailing edges, and
2. the introduction of cut cells that are a small fraction of the size of uncut cells.

Euler solvers based on central-differencing on Cartesian meshes have been developed by Clarke et al [8] and Epstein et al [9]. In the work of Clarke et al, resolution of leading and trailing edges was achieved simply by clustering the mesh lines near the points of interest; cut cells were handled by merging them with neighboring uncut cells. In the work of Epstein et al, resolution was achieved by local mesh refinement; cut cells were handled by a non-conservative extrapolation procedure. An upwind-differencing method on an adaptively refined Cartesian mesh has been developed by Berger and LeVeque [10], for unsteady flows. In their work, the large time-step method of Leveque [11] was used to help remove the numerical stiffness due to small cut cells.

In the work presented in this paper, resolution is achieved by use of adaptive refinement; cut cells are handled by a reconstruction method for general unstructured meshes coupled with local time-stepping. The work consists of a mesh generation based on a Cartesian mesh

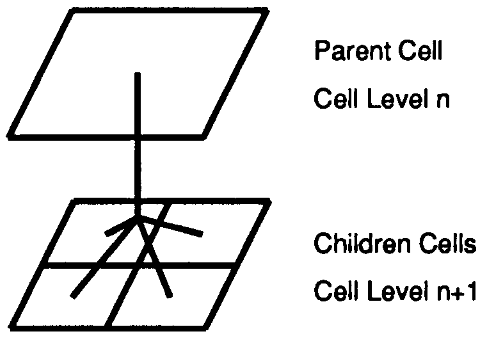

Figure 1: Parent/Children Relationship

with geometry-based refinement [12], and a flow solver based on the MUSCL concept [13], with a linear reconstruction technique [14] and Roe's approximate Riemann solver [15]. In addition, solution-adaptive refinement of the mesh is used to gain resolution in high-gradient regions of the flow [16, 17]. Each element of the mesh generation technique and the flow solver is described in the following text, along with the data structure used. Results for a collection of test cases are presented and discussed.

2 Generation of the Mesh and Data Structure

2.1 Data Structure

The basic data structure used is a hierarchical cell-based quadtree structure — "parent" cells are refined by division into four "children" cells. This concept is illustrated in Figure 1. Each cell has a pointer to its parent cell (if one exists) and to its four children cells (if they exist). The cells farthest down the hierarchy, that is, the ones with no children, are the cells on which the calculation takes place.

In addition to this basic structure, a list of cell-faces is made, and a structure containing the cell-to-face connectivity, and the face-to-cell connectivity, is stored. Similarly, a list of nodes and the cell-to-node connectivity of the mesh is stored. The face structure is not strictly necessary; however it does reduce the time spent in traversing the quadtree structure. The node structure is necessary only for the post-processing step; it is not utilized in the solution process.

Other data structure issues include the need to mark cells, nodes and faces as to whether they are inside the flow domain, inside the body, or crossing the body surface. Also included in the mesh data structure are the coordinates of the nodes, the face midpoints, and the cell centroids. Again, these need not necessarily be stored; they could be computed relatively inexpensively from the basic quadtree structure.

With this data structure in place, refining a cell (i.e. spawning four children cells) reduces primarily to a trivial change of the cell-based quadtree structure. Resulting changes in the list of nodes and faces, and the pointers, can be derived from the quadtree structure.

2.2 Generation of an Initial Mesh

To generate the initial mesh, a coarse Cartesian mesh of equal-sized cells is generated that covers the entire area of interest. The body is then "cut" out of this mesh. The procedure for computing the intersections of the body with the mesh depends upon how the body has been defined.

For a body defined by a piecewise analytic function $y(x)$, with the inverse function $x(y)$ also known, a node of the Cartesian mesh can be classified as inside or outside the body immediately, by determining whether the $x =$ constant and $y =$ constant lines through the node interesect the body an even or odd number of times between the node and the outer boundary. For a body defined by a piecewise analytic function $y(x)$, with the inverse function not known, the intersections of the line $x =$ constant with the body are known immediately; a search procedure must be set up to determine the intersections in the $x-$direction. For a body defined simply by a set of data points, a search procedure must be carried out in both the x- and y-directions to find the intersections. For bodies defined in this manner, a spline (C^0 at trailing-edges, C^2 elsewhere) is fit through the points defining the body. The spline-fit of the body data points was found to be necessary to obtain smooth flow solutions.

Each node in the Cartesian mesh is tested and classified as to whether it is inside one of the bodies, or in the flow domain. Once each node has been thus classified, the faces of the Cartesian mesh are classified as inside the body (both nodes defining the face are classified as inside), outside the body (both nodes defining the face are classified as outside), or "cut" (one node is classified as being outside, the other, inside). For faces classified as cut, the location of the intersection of the face and the body (which is known from the procedure for classifying the nodes) is stored.

Cells are similarly classified, based on the number of nodes inside or outside the body or bodies. If only one node is outside the body, the cell is a cut cell, with three faces (two cut faces and one face on the body). If two (or three) nodes are outside the body, the cell is a cut cell with four (or five) faces. Uncut cells are those with all four nodes either outside or inside the body.

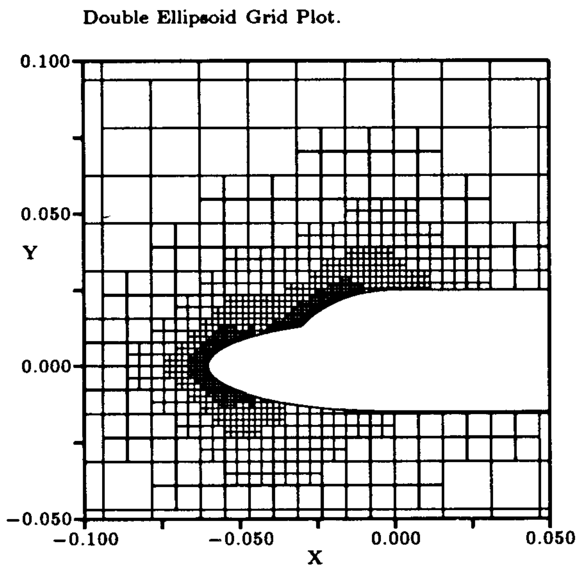

Figure 2: Example Mesh

2.3 Geometry-Based Mesh Refinement

Geometry-based mesh refinement [12] is the next step in creating a suitable mesh. Once the initial Cartesian mesh has been generated, and the cut locations determined, a combination of cut-cell refinement and curvature refinement is applied to the mesh.

In cut-cell refinement, each cell cut by a body is refined, along with its three nearest neighbors. Refining neighbors of cut cells ensures a smoother transition to fine cells on the body from the coarser outer flow cells. Refinement of cut cells is applied successively until the mesh on the body is brought to a user-specified level.

Once the desired cut-cell refinement is completed, curvature refinement is applied to the mesh. The slopes of the body faces on two consecutive cut cells are compared. If the difference in slopes is above a threshhold value, both cells, and their nearest neighbors, are refined. The actual check used to flag cells for curvature refinement is given by

$$\left| \left(\frac{\Delta Y}{\Delta X}\right)_{Cell1} - \left(\frac{\Delta Y}{\Delta X}\right)_{Cell2} \right| \geq 0.05 \qquad (1)$$

with special care taken for faces with ΔX small.

An example mesh generated by the above procedures is shown in Figure 2.

2.4 Solution-Based Mesh Refinement

An adaptive mesh may be refined or coarsened based on the characteristics of the flow about the body. Refinement takes place only after a solution is sufficiently converged. At that point, cells are flagged for refinement, based on the difference (undivided) of the total velocity between cells. If the total velocity difference is above a user-specified fraction (typically 5%) of the maximum total velocity difference, then the two cells sharing that face are flagged for refinement. For each cell that is flagged, four children cells are added to the quadtree data structure, one level farther down the hierarchy than their parent cell.

2.5 Mesh "Smoothing"

The mesh resulting from the above procedure could have certain "undesirable features." Some are undesirable in that allowing them would complicate the data structure; others are undesirable in that computational experience shows that they may degrade the solution somewhat in their vicinity.

The eight features that are currently labelled "undesirable" are described below, and depicted in Figure 3.

1. Cell level differences greater than 1 between two neighboring cells
2. Cell level differences greater than 0 normal to body-cut cells
3. Cell level differences greater than 0 through outer flow boundaries
4. Cell level differences greater than 0 between three-sided cells and their neighbors
5. "Holes" in the mesh
6. More than two cuts on a cell
7. Cell level differences greater than 0 on trailing edge of body
8. Bodies too close, only two cells apart

When an undesirable feature is found, the mesh is "smoothed" to eliminate it, by refining appropriate cells until the feature no longer exists. This is a recursive procedure, that converges to a mesh with no undesirable features.

3 Flow Solver

The flow solver described here consists of three primary components: a linear reconstruction method, for obtaining accurate, limited values of the flow variables at face midpoints; an approximate Riemann solver, for computing the flux through cell-faces; and a multi-stage time-stepping scheme for advancing the solution to a steady state. The individual components of this procedure are described below.

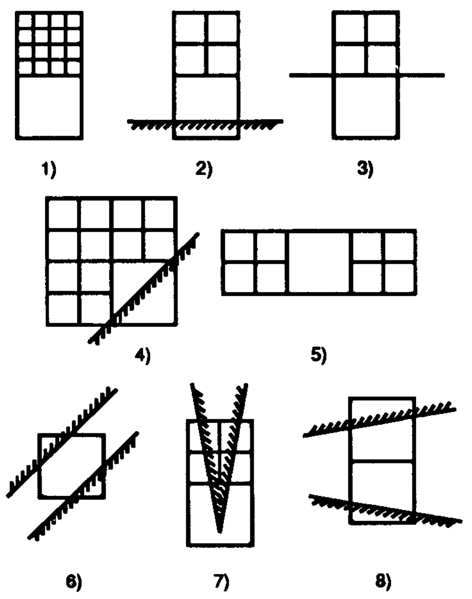

Figure 3: "Undesirable" Mesh Features

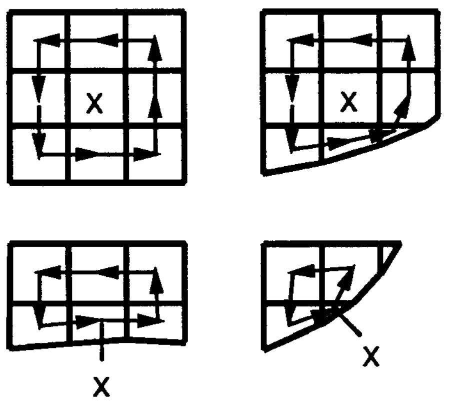

Figure 4: Normal And Special Paths

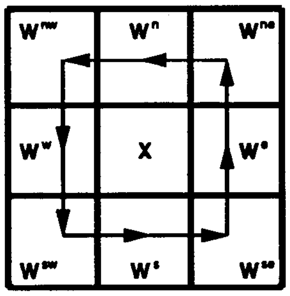

Figure 5: Integral Path

3.1 Reconstruction Procedure

In order to evaluate the flux through a face, flow quantities are required at both sides of the face. To achieve higher-order accuracy, solution-gradient information must be used. A linear reconstruction method [1] is used to determine a second-order approximation to the state at the face midpoint, based on the cells in the neighborhood of the face. It relies on a suitable path integral about the cell of interest, with the gradient of a quantity W_k in a cell being determined by

$$\nabla W_k = \frac{1}{A_\Omega} \int_{\partial\Omega} W_k \hat{\mathbf{n}} d\ell \,, \tag{2}$$

where A_Ω is the area enclosed by the path of integration, $\partial\Omega$. Here, W_k represents the quantity being reconstructed; in this work, the primitive variables $\mathbf{W} = (\rho, u, v, p)^T$ are reconstructed.

The path for the integration is constructed by connecting the centroids of neighboring cells. Away from cut cells or cell-level differences, the eight immediate neighbors of a cell are used. Near a body, as few as four cells are used to construct the path. For a linear reconstruction, a minimum of three non-colinear cell-centers are required to form a proper path. Some examples of the paths are shown in Figure 4 to the normal path, with the X denoting the cell for which the gradient is being calculated.

Once the cells in the path are determined, the path integral is carried out numerically. In general, the area inside the path is calculated by summing the areas of the triangles formed by connecting the centroids in the path to the centroid of the cell for which the gradient is being calculated. The x and y components of the gradient are normalized by the cell width and height, respectively.

In the regions of the mesh where a cell and its eight neighbors are uncut and the same size, the resulting integral path is a square twice the size of the cell. In this case, higher accuracy can be achieved by combining pairs of faces in a trapezoidal rule. When the gradient is normalized by the cell width or height, it becomes a function of only the cell values, not the geometry. These more accurate representations of the normalized gradient are

$$\begin{aligned} \frac{\partial W_k}{\partial x} &= \frac{1}{12}(W_k^{ne} + 4W_k^e + W_k^{se} \\ &\qquad - W_k^{nw} - 4W_k^w - W_k^{sw}) \end{aligned} \tag{3a}$$

$$\begin{aligned} \frac{\partial W_k}{\partial y} &= \frac{1}{12}(W_k^{nw} + 4W_k^n + W_k^{ne} \\ &\qquad - W_k^{sw} - 4W_k^s - W_k^{se}) \,. \end{aligned} \tag{3b}$$

Figure 5 shows the path for this case.

Once the gradient of W_k is known in each cell, the value of W_k can be found anywhere in the cell from

$$W_k(x,y) = W_k^c + \nabla W_k \cdot \mathbf{dr} \tag{4}$$

where W_k^c is the value of W_k at the cell centroid, and $\mathbf{dr}$ is defined as

$$dr_x = \frac{(x - x^c)}{\Delta x_{cell}} \tag{5a}$$

$$dr_y = \frac{(y - y^c)}{\Delta y_{cell}} . \tag{5b}$$

For example, the values at the face midpoints of an uncut cell are simply

$$W_k^{top} = W_k^c + \frac{1}{2}\nabla_y W_k \tag{6a}$$

$$W_k^{right} = W_k^c + \frac{1}{2}\nabla_x W_k \tag{6b}$$

$$W_k^{bottom} = W_k^c - \frac{1}{2}\nabla_y W_k \tag{6c}$$

$$W_k^{left} = W_k^c - \frac{1}{2}\nabla_x W_k . \tag{6d}$$

3.2 Limiting

If the full gradient were used in reconstructing the values at face midpoints, the computed values could fall outside the bounds of the data used in the path integral. To avoid this, the computed gradients are *limited*; that is the primitive variables $\mathbf{W} = (\rho, u, v, p)^T$ are reconstructed via

$$\mathbf{W}(x,y) = \mathbf{W}^c + \phi \nabla \mathbf{W} \cdot \mathbf{dr} \tag{7}$$

where ϕ is a limiter, with a value between zero and one. In regions where $\phi = 1$, a linear reconstruction is being used; in regions where $\phi = 0$, a piecewise constant reconstruction is being used. The limiter ϕ is defined as

$$\phi = \min \left\{ \begin{array}{l} 1 \\ \min_k \left(\frac{\max_{path}|W_k - W_k^c|}{\max_{cell}|W_k - W_k^c|} \right) \\ \min_k \left(\frac{\min_{path}|W_k - W_k^c|}{\min_{cell}|W_k - W_k^c|} \right) \end{array} \right. . \tag{8}$$

The minimum and maximum over the path are found by examining the values of W_k used in the path integration; the minimum and maximum over the cell are found by using the gradient to reconstruct W_k at the corners of the cell. Thus, the limiter acts to ensure that the values of W_k at the nodes of the cell for which the gradient is being calculated are bounded by the values of W_k that are used in calculating the gradient. Using a single limiter for the gradient of the vector $\mathbf{W}$ was found to give superior results. In standard MUSCL-type schemes [13], a separate limiter is typically used for each variable, and for each mesh direction, resulting in eight limiters for each cell.

Unfortunately, limiting can seriously hamper the convergence to a steady state, with the nonlinearity of the scheme resulting in limit cycles. To combat this problem, the limiter values are "frozen" after a certain point in the convergence, using previously stored values of the limiter rather than recomputing them at each time step. Freezing the limiters allows the residuals to converge to machine zero.

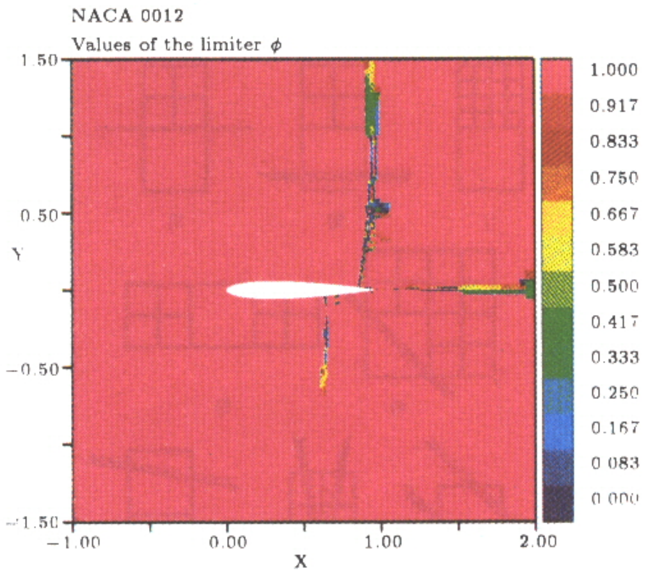

Figure 6: Limiter Values — Transonic Airfoil

Typical values of the limiter are shown in Figure 6, in which the $\phi(x,y)$ is plotted for a transonic airfoil case. As can be seen, a linear reconstruction ($\phi = 1$) is used nearly everwhere. In the immediate vicinity of the shocks, and the wake, the limiter reduces the order of accuracy of the scheme. The percentage of cells in which the limiter is less than one is extremely small, however. A representative convergence history is shown in Figure 7.

3.3 Approximate Riemann Solver

The finite-volume form of the Euler equations can be written as

$$\frac{d\mathbf{U}}{dt} = -\frac{1}{A}\sum_{faces}(\mathbf{F}\Delta y - \mathbf{G}\Delta x) \tag{9}$$

where A is the area of that cell, Δx and Δy are the changes of x and y along a face (defined so that the integral is carried out in a counter-clockwise sense), and $\mathbf{U}$, $\mathbf{F}$ and $\mathbf{G}$ are defined as

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \tag{10a}$$

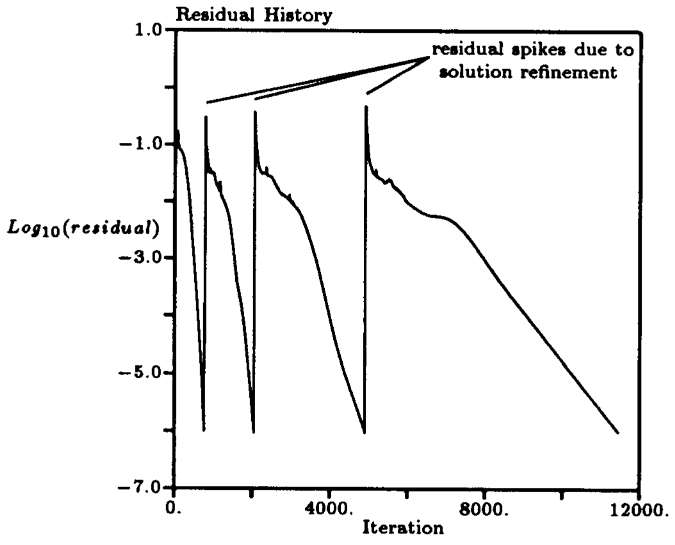

Figure 7: Representative Convergence History

$$\mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix} \qquad (10b)$$

$$\mathbf{G} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{pmatrix} . \qquad (10c)$$

Defining the face length and the normal and tangential velocities as

$$\Delta s = \sqrt{(\Delta x)^2 + (\Delta y)^2} \qquad (11)$$

$$u_n = \frac{(u\Delta y - v\Delta x)}{\Delta s} \qquad (12a)$$

$$u_t = \frac{(u\Delta x + v\Delta y)}{\Delta s}, \qquad (12b)$$

the flux through a face may be written as

$$(\mathbf{F}\Delta y - \mathbf{G}\Delta x) = \begin{pmatrix} \rho u_n \\ \rho u_n u + p\frac{\Delta y}{\Delta s} \\ \rho u_n v - p\frac{\Delta x}{\Delta s} \\ \rho u_n H \end{pmatrix} \Delta s \equiv \boldsymbol{\Phi}\Delta s . \quad (13)$$

The flux through a face is a function of the values at the face midpoint, given by the reconstruction in the cells to the "left" and "right" of the face. Using Roe's approximate Riemann solver, this flux function is

$$\begin{aligned} \boldsymbol{\Phi}(\mathbf{U}_L, \mathbf{U}_R) &= \frac{1}{2}[\boldsymbol{\Phi}(\mathbf{U}_L) + \boldsymbol{\Phi}(\mathbf{U}_L)] \\ &\quad - \frac{1}{2}\sum_{k=1}^{4} |\hat{a}_k|^* \Delta V_k \hat{\mathbf{R}}_k \end{aligned} \qquad (14)$$

with

$$\hat{\mathbf{a}} = \begin{pmatrix} \hat{u}_n - \hat{c} \\ \hat{u}_n \\ \hat{u}_n \\ \hat{u}_n + \hat{c} \end{pmatrix} \qquad (15a)$$

$$\mathbf{\Delta V} = \begin{pmatrix} \frac{\Delta p - \hat{\rho}\hat{c}\Delta u_n}{2\hat{c}^2} \\ \frac{\hat{\rho}\Delta u_t}{\hat{c}} \\ \Delta\rho - \frac{\Delta p}{\hat{c}} \\ \frac{\Delta p + \hat{\rho}\hat{c}\Delta u_n}{2\hat{c}^2} \end{pmatrix} \qquad (15b)$$

$$\hat{\mathbf{R}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ \hat{u} - \hat{c}\frac{\Delta y}{\Delta s} & \hat{c}\frac{\Delta x}{\Delta s} & \hat{u} & \hat{u} + \hat{c}\frac{\Delta y}{\Delta s} \\ \hat{v} + \hat{c}\frac{\Delta x}{\Delta s} & \hat{c}\frac{\Delta y}{\Delta s} & \hat{v} & \hat{v} - \hat{c}\frac{\Delta x}{\Delta s} \\ H - \hat{u}_n\hat{c} & \hat{u}_t\hat{c} & \frac{\hat{u}^2+\hat{v}^2}{2} & H + \hat{u}_n\hat{c} \end{pmatrix} \qquad (15c)$$

and

$$\hat{\rho} = \sqrt{\rho_L \rho_R} \qquad (16a)$$

$$\hat{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \qquad (16b)$$

$$\hat{v} = \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \qquad (16c)$$

$$\hat{H} = \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \qquad (16d)$$

where $\hat{c}$, $\hat{u}_n$, and $\hat{u}_t$ are calculated directly from $\hat{\rho}$, $\hat{u}$, $\hat{v}$, and $\hat{H}$.

To prevent expansion shocks, an entropy fix is imposed [18]. A smoothed value, $\left|\hat{a}^{(k)}\right|^*$, is defined to replace $\left|\hat{a}^{(k)}\right|$ for the two acoustic waves ($k = 1$, $k = 4$). For those two waves,

$$\left|\hat{a}^{(k)}\right|^* = \begin{cases} \left|\hat{a}^{(k)}\right| & \left|\hat{a}^{(k)}\right| \geq \frac{1}{2}\delta a^{(k)} \\ \frac{\left(\hat{a}^{(k)}\right)^2}{\delta a^{(k)}} + \frac{1}{4}\delta a^{(k)} & \left|\hat{a}^{(k)}\right| \leq \frac{1}{2}\delta a^{(k)} \end{cases}$$

$$\delta a^{(k)} = max\left(4\Delta a^{(k)}, 0\right), \Delta a^(k) = a_R^{(k)} - a_L^{(k)} . \qquad (17)$$

For each cell, the face fluxes, calculated as above, are summed to give the residual for the cell,

$$\mathbf{Res}\,(\mathbf{U}) = -\frac{1}{A}\sum_{faces} \boldsymbol{\Phi}\;\Delta s\,. \tag{18}$$

These residuals are then integrated in time, as described below.

3.4 Time-Stepping Scheme

The time-stepping scheme used is one of the optimally-smoothing multi-stage schemes developed by Tai [19, 20]. The general m-stage scheme is defined as

$$\begin{aligned}\mathbf{U}^{(0)} &= \mathbf{U}^{n}\\ \mathbf{U}^{(k)} &= \mathbf{U}^{(0)} + \alpha_k \Delta t \mathbf{Res}\left(\mathbf{U}^{(k-1)}\right), k = 1, m\\ \mathbf{U}^{n+1} &= \mathbf{U}^{(m)}\,. \end{aligned} \tag{19}$$

The five-stage scheme which gives optimal damping of Fourier modes in the range $\pi/4 \leq k\Delta x \leq \pi$ has multi-stage coefficients

$$\begin{aligned}\alpha_1 &= 0.0695\\ \alpha_2 &= 0.1602\\ \alpha_3 &= 0.2898\\ \alpha_4 &= 0.5060\\ \alpha_5 &= 1.0000\end{aligned} \tag{20}$$

and CFL number 1.1508.

Local time-stepping is used, and indeed is necessary. The meshes generated have extremely large differences in area from cell to cell, due to the cut cells.

4 Post-Processing

Post-processing requires transfering the known cell-centered values to nodal values for plotting as accurately as possible. To do this, the limited cell gradients are used to extrapolate to each cell's nodes as shown in Figure 8

The function as reconstructed at the nodes is multi-valued; there is one value resulting from the representation in each cell sharing that node. These multiple values at each node are averaged to yield a single accurate, bounded value there.

5 Current Results

The method described above has been tested on several internal and external flows. The cases were chosen so as to show the flexibility of the Cartesian mesh approach, and the fidelity of results available by use of solution-based refinement.

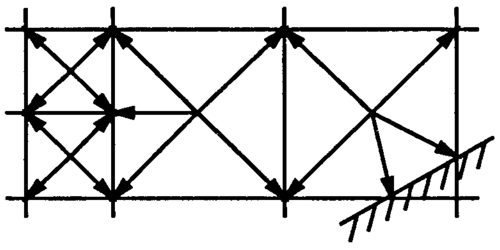

Figure 8: Obtaining Post-Processed Nodal Values

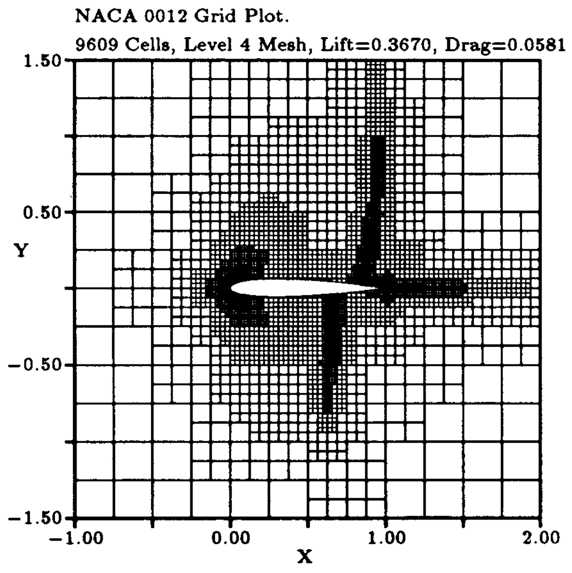

Figure 9: NACA 0012, Mesh

5.1 Transonic NACA Airfoil

The steady-state flow is computed about a NACA 0012 airfoil at $M_\infty = 0.85$ and $\alpha = 1°$. For these conditions, shocks exist on the upper and lower surfaces of the airfoil.

The outer boundary of the mesh was set at a 2048-chord radius. This virtually eliminated the need for a vortex boundary condition; simply specifying the free-stream state at the outer boundary faces proved adequate. The initial mesh was refined based on curvature, to resolve the leading and trailing edge. Three levels of solution-based refinement were done, as well.

As can be seen in Figures 9 and 10, both shocks, as well as the wake, are well resolved. The pressure coefficient, plotted in Figure 11, results in a lift of 0.3670 and a drag of 0.0581.

To show the effect of the outer boundary conditions on the solution, this case was run for outer boundaries ranging from a 4-chord to a 2048-chord radius. Geometry-based refinement was carried out so as to ensure an equiva-

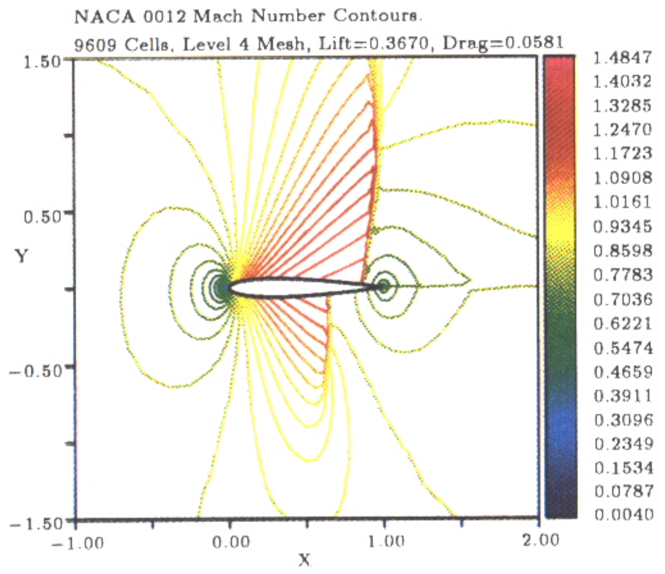

Figure 10: NACA 0012, Mach Number Contours

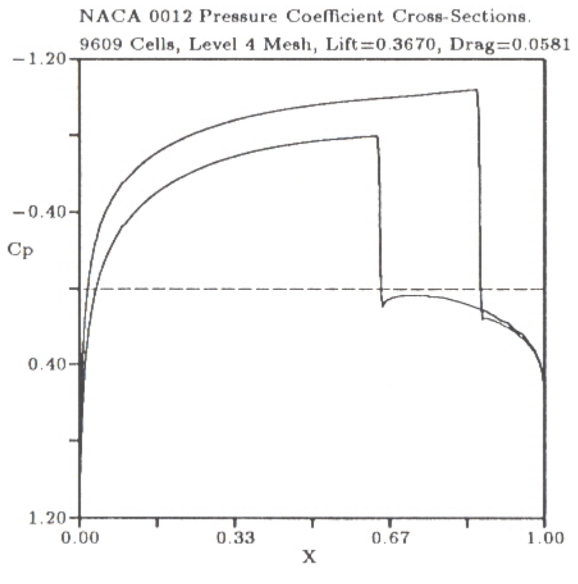

Figure 11: NACA 0012, Cp on the Body

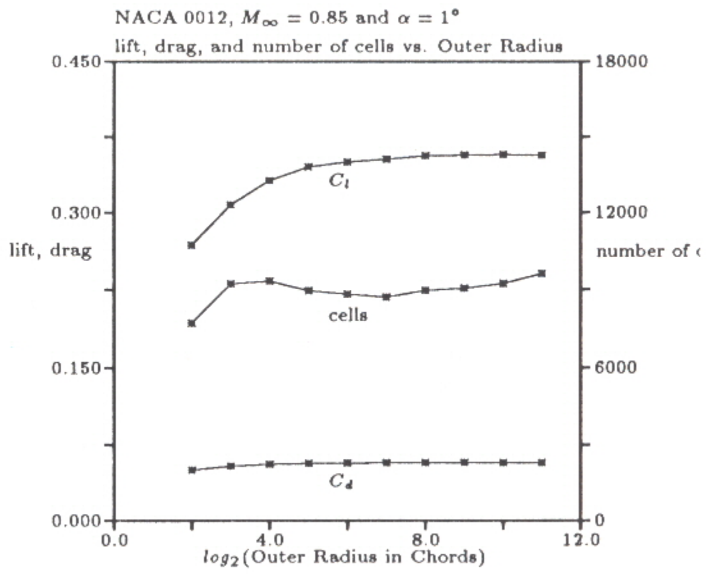

Figure 12: NACA 0012, Lift and Drag vs. Boundary Radius

lent mesh in the vicinity of the airfoil for each case. Three levels of solution-based refinement were also done for each case. As can be seen in Figure 12, this wide range of outer boundary radius resulted in only a small change in the total number of computational cells. Even though the crudest possible outer boundary condition (free-stream flow) was enforced, the lift and drag are converged by the 512 chord case. Thus, by use of mesh refinement, simple outer boundary conditions may be enforced, at a radius far from the body, at very little computational cost.

5.2 "Subsonic" Three-Element Airfoil

For this case, $M_\infty = 0.2$ and $\alpha = 0°$. Portions of the mesh are shown in Figures 13, 14, and 15. As can be seen, the geometry-adaptive refinement gives sufficient resolution of leading and trailing edges. Contours of Mach number are shown in Figures 16, 17, and 18. Due to the high effective camber caused by the flaps, the flow actually expands to supersonic speed about the leading edge, and a weak shock is present. The Mach number and pressure coefficient distributions on the surface are shown in Figures 19 and 20. Despite large variations in cell size on the body (the smallest cut cell is a factor 10^6 smaller than its uncut neighbor) the solution is smooth. The total pressure loss, $1 - p_0/p_{0_\infty}$, is shown in Figure 21. There is some total pressure loss on the upper surface of the main element ($\approx 0.5\%$), due to the shock at the leading-edge. On the lower surface of the main element, and on the flaps, the loss is less than 0.1%. The lift for the system is $C_\ell = 3.7842$; the drag is $C_d = 0.0159$.

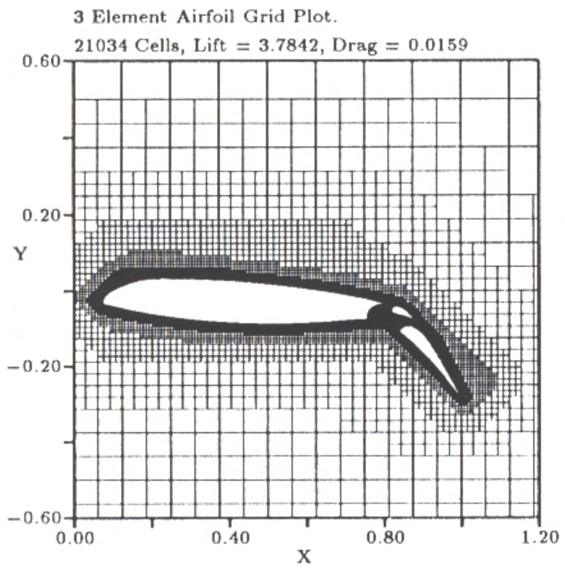
3 Element Airfoil Grid Plot.
21034 Cells, Lift = 3.7842, Drag = 0.0159
0.60
0.20
Y
-0.20
-0.60
0.00
0.40
0.80
1.20
X

Figure 13: Three Element Airfoil, Mesh

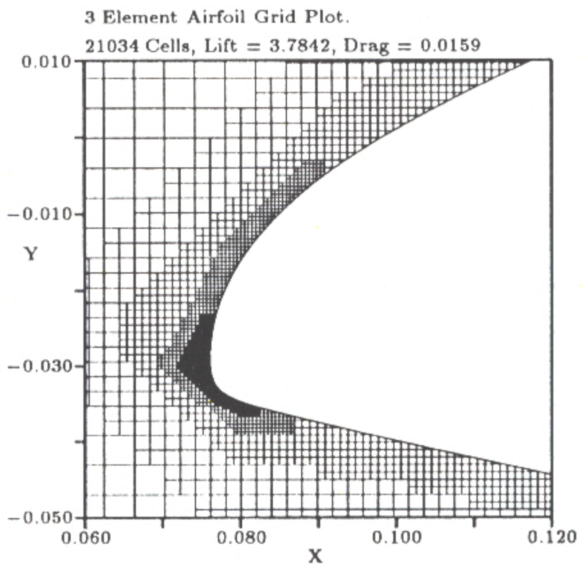
3 Element Airfoil Grid Plot.
21034 Cells, Lift = 3.7842, Drag = 0.0159
0.010
-0.010
Y
-0.030
-0.050
0.060
0.080
0.100
0.120
X

Figure 15: Three Element Airfoil, Mesh (Detail)

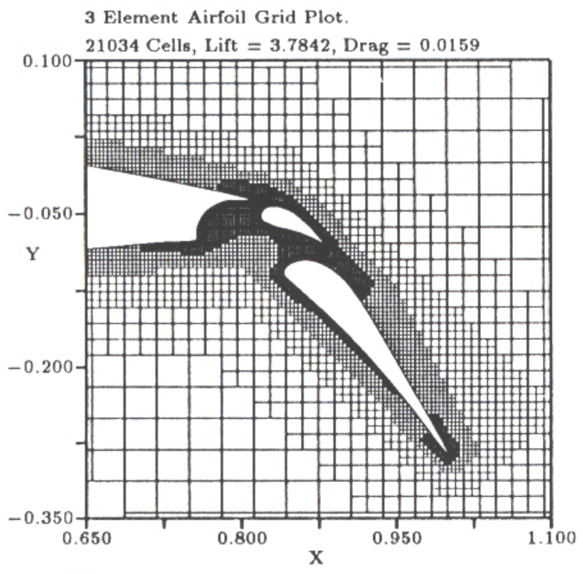
3 Element Airfoil Grid Plot.
21034 Cells, Lift = 3.7842, Drag = 0.0159
0.100
-0.050
Y
-0.200
-0.350
0.650
0.800
0.950
1.100
X

Figure 14: Three Element Airfoil, Mesh (Detail)

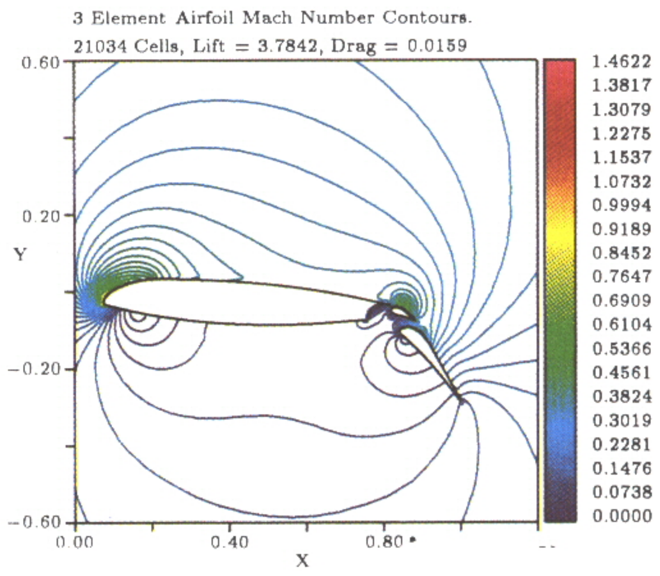

Figure 16: Three Element Airfoil, Mach Number Contours

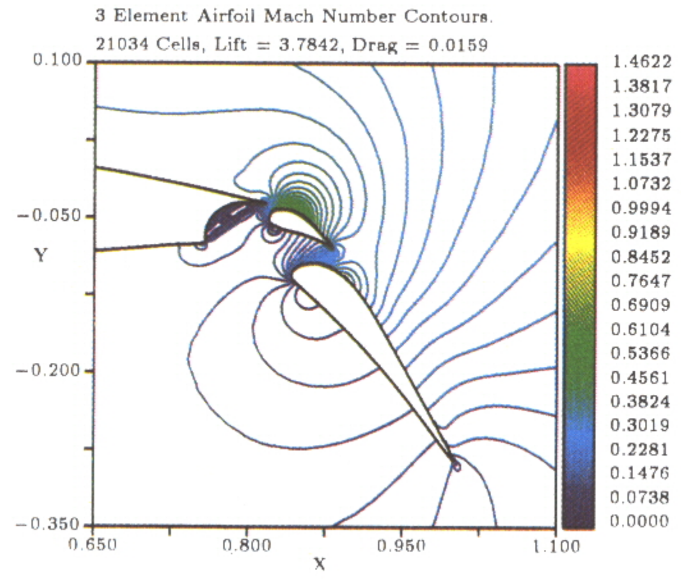

Figure 17: Three Element Airfoil, Mach Number Contours (Detail)

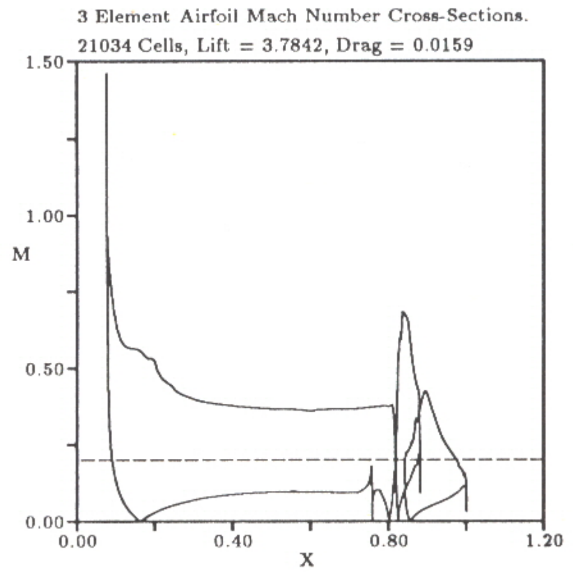

Figure 19: Three Element Airfoil, Mach Number on Surface

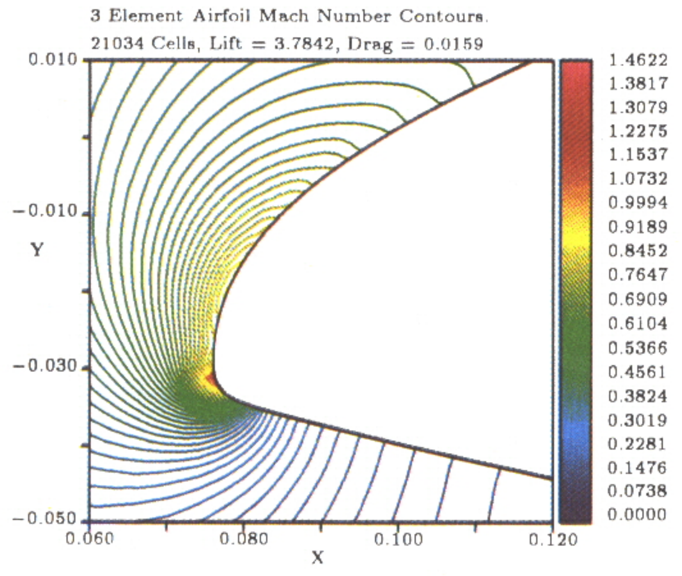

Figure 18: Three Element Airfoil, Mach Number Contours (Detail)

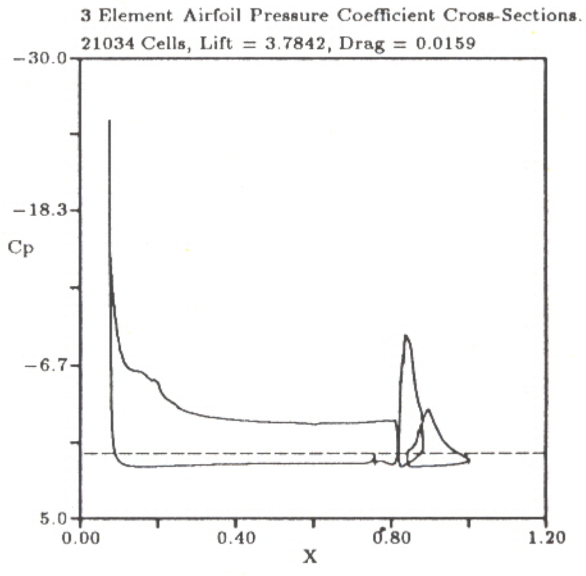

Figure 20: Three Element Airfoil, C_p on Surface

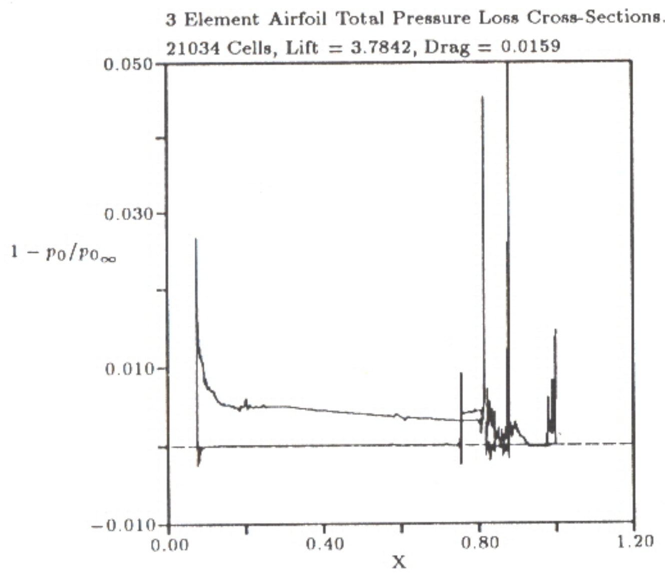

Figure 21: Three Element Airfoil, $1 - p_0/p_{0_\infty}$ on Surface

5.3 Supersonic Double Ellipsoid

For this case, $M_\infty = 8.15$ and $\alpha = 30°$. The mesh was refined four times based on the solution; the resulting mesh is shown in Figure 22. The refined regions correspond to the nose, the bow shock and the canopy shock. The Mach number and pressure contours are shown in Figures 23 and 24 respectively. The shocks and the expansion about the nose are all well resolved. Finally, the pressure coefficient on the body is plotted in Figure 25. Despite the presence of small cut cells on the body, the pressure distribution is very smooth, and the canopy shock is captured cleanly.

5.4 Supersonic Channel Flow

In this, the steady-state flow is computed in a channel with a 15° compression corner, followed by a 15° expansion corner. The free-stream Mach number is $M_\infty = 2.0$. There is an attached shock at the compression corner, which reflects from the top wall, forming a small Mach stem. The shock reflects from the bottom wall as well, before exiting the channel. The expansion corner acts to weaken the reflected shock. There is also a slip line, emanating from the triple point near the upper wall.

The Mach number contours and mesh are shown in Figure 26. A blowup of the Mach stem is shown in Figures 27 and 28. The shear that emanates from the triple point of the Mach stem is carried cleanly out through the flow, although somewhat weakened by the expansion. The pressure contours plotted in Figure 29 pass smoothly through the shear, as they should. The pressure on the wall is shown in Figure 30. Note that the pressure is constant on

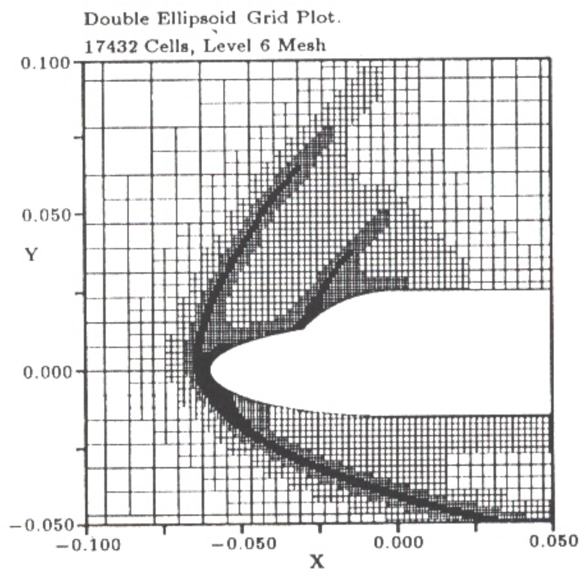

Figure 22: Double Ellipsoid, Mesh

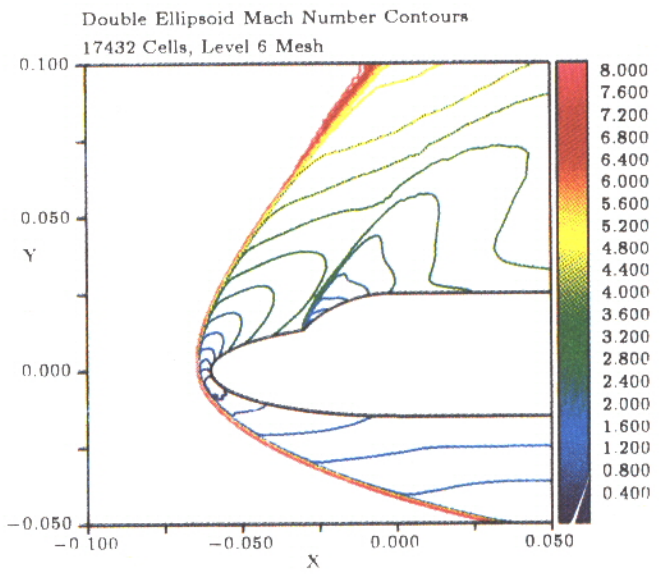

Figure 23: Double Ellipsoid, Mach Number Contours

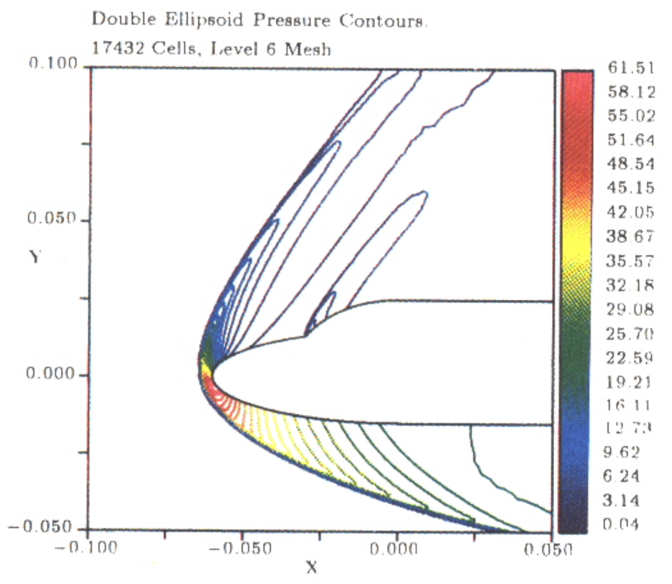

Figure 24: Double Ellipsoid, Pressure Contours

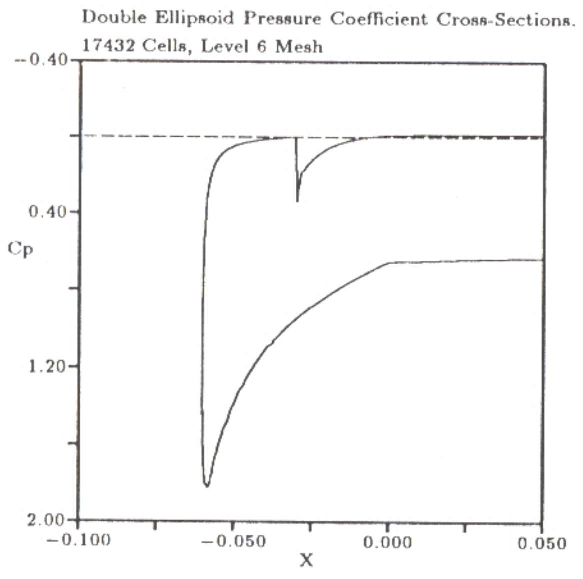

Figure 25: Double Ellipsoid, Pressure Coefficient on Body

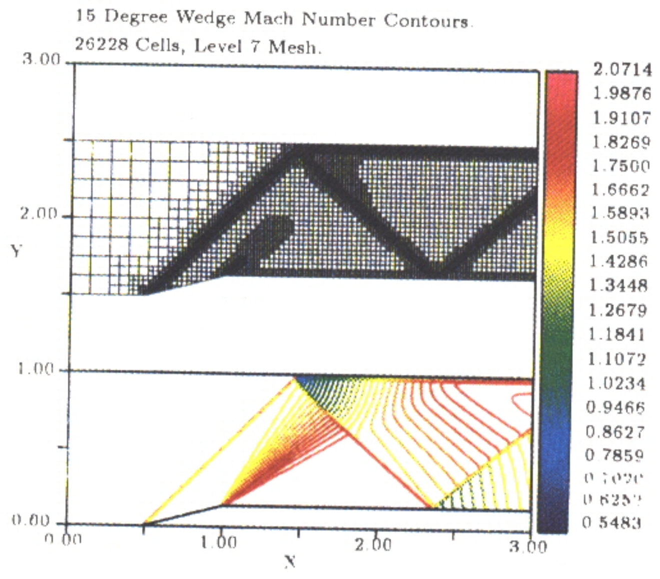

Figure 26: 15° Wedge, Mach Number Contours and Mesh

the incline of the ramp. Cell areas on the incline vary as much as six orders of magnitude from cell to cell without having a detrimental affect on the solution.

5.5 Axisymmetric Under-Expanded Jet Flow

In this case, an axisymmetric jet with Mach number $M_{jet} = 1.25$, exhausts into a stream with Mach number $M_{stream} = 1.25$. The total pressure ratio of jet to stream is fifteen; the total temperature ratio is unity. Two levels of solution-based refinement were done; the final mesh and Mach number contours are shown in Figure 31. The outer stream passes through a shock slightly upstream of the lip of the nozzle. A curved shear and a curved shock emanate from the lip of the nozzle; the shock reflects from the axis of symmetry, and interacts with the shear.

6 Summary

An adaptive Cartesian mesh algorithm has been developed, and has been used successfully to obtain steady-state solutions of the Euler equations for a variety of internal and external flows. Test cases included a transonic airfoil, a three-element airfoil, a double ellipsoid, a channel flow, and an axisymmetric jet flow. For airfoil cases, the outer boundary can be placed arbitrarily far from the airfoil without significantly increasing the number of cells, thus eliminating the need for sophisticated outer boundary conditions. Both geometry- and solution-based refinement effectively enhance resolution of regions of high

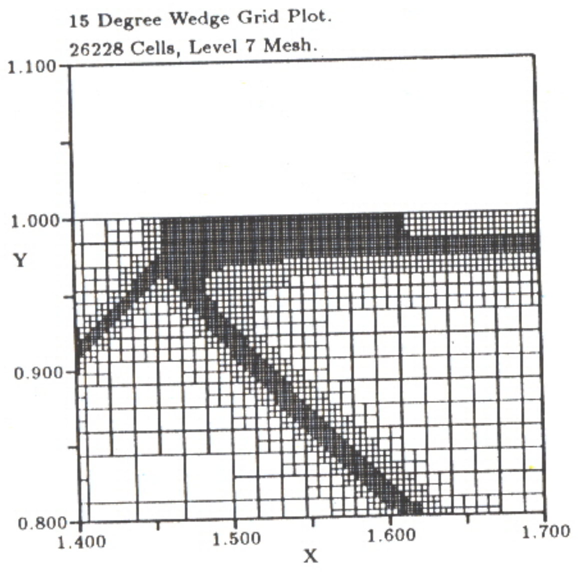

Figure 27: 15° Wedge, Mesh (Detail)

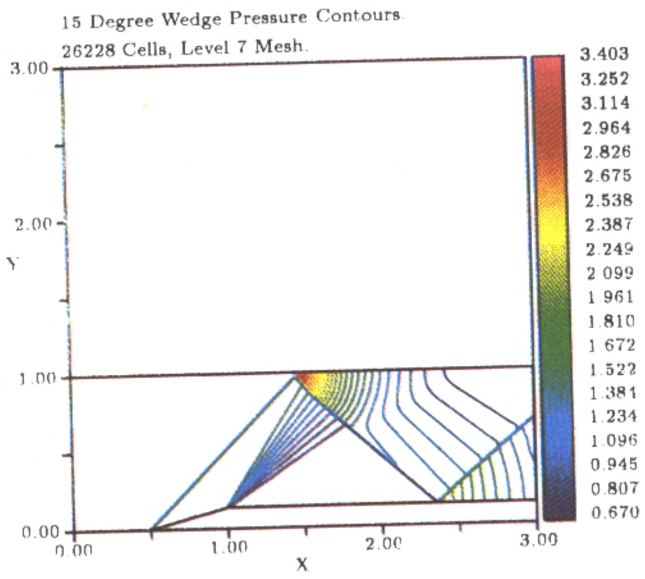

Figure 29: 15° Wedge, Pressure Contours

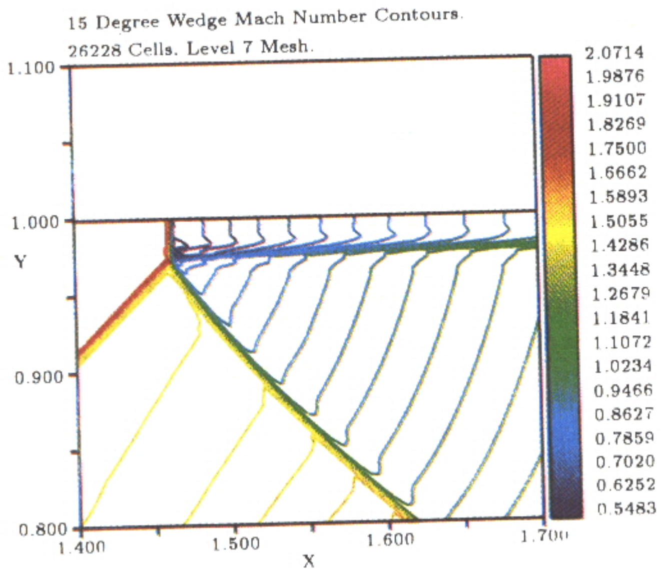

Figure 28: 15° Wedge, Mach Number Contours (Detail)

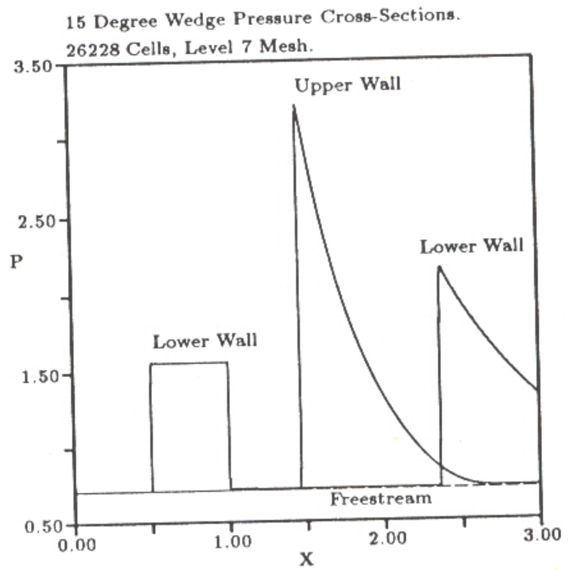

Figure 30: 15° Wedge, Pressure on Upper and Lower Walls

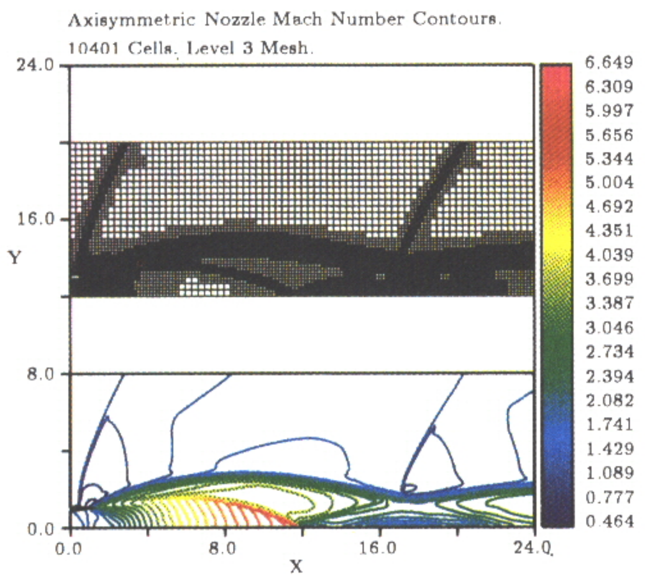

Figure 31: Axisymmetric Under-Expanded Jet, Mach Number Contours and Mesh

body curvature and large flow gradients. The reconstruction method, coupled with local time stepping, eliminate the problems typically associated with the small cut cells caused by cutting the body from the Cartesian mesh.

Acknowledgments

This work presented herein has been funded by the National Science Foundation (Grant EET-885-700, monitored by Dr. George Lea) and NASA Langley Research Center (Grant NAG-1-869, monitored by Dr. James Thomas). Calculations were carried out on IBM RS/6000 workstations which were donated by IBM. The authors gratefully acknowledge their support. The authors also thank Dr. Klaus-Peter Beier of the University of Michigan, for his help with the parametric-spline algorithm.

References

[1] T. J. Barth, "On unstructured grids and solvers," in *Computational Fluid Dynamics*, Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[2] R. Löhner and P. Parikh, "Generation of three-dimensional unstructured grids by the advancing-front method," AIAA Paper 88-0515, 1988.

[3] D. J. Mavriplis, "Adaptive mesh generation for viscous flows using Delaunay triangulation," *Journal of Computational Physics*, vol. 90, 1990.

[4] A. Jameson, T. J. Baker, and N. P. Weatherill, "Calculation of inviscid transonic flow over a complete aircraft," AIAA Paper 86-0103, 1986.

[5] J. F. Dannenhoffer III, "Computer-aided block structuring through the use of optimization and expert system techniques," in *AIAA 10th Computational Fluid Dynamics Conference*, 1991.

[6] M. M. Rai, "Unsteady three-dimensional Navier-Stokes simulations of turbine rotor-stator interaction," AIAA Paper 87-2058, 1987.

[7] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, and S. S. Samant, "A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics," *Journal of Computational Physics*, vol. 62, pp. 1–66, 1991.

[8] D. K. Clarke, M. D. Salas, and H. A. Hassan, "Euler calculations for muti-element airfoils using Cartesian grids," *AIAA Journal*, vol. 24, 1986.

[9] B. Epstein, A. L. Luntz, and A. Nachshon, "Multigrid Euler solver about arbitrary aircraft configurations with Cartesian grids and local refinement," in *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[10] M. J. Berger and R. J. LeVeque, "An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries," in *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[11] R. J. LeVeque, "High resolution finite volume methods on arbitrary grids via wave propagation." ICASE Report 87-68, 1987.

[12] K. Nakahashi, "An automatic grid generator for the unstructured upwind method," in *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[13] B. van Leer, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, vol. 32, 1979.

[14] T. J. Barth and P. O. Frederickson, "Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction," AIAA Paper 90-0013, 1990.

[15] P. L. Roe, "Approximate Riemann solvers, parameter vectors and difference schemes," *Journal of Computational Physics*, vol. 43, 1981.

[16] J. G. Kallinderis and J. R. Baron, "Adaptation methods for a new Navier-Stokes algorithm," in *AIAA 8th Computational Fluid Dynamics Conference*, 1987.

[17] M. J. Aftosmis and N. Kroll, "A quadrilateral based second-order TVD method for unstructured adaptive meshes," AIAA Paper 91-0124, 1991.

[18] B. van Leer, W. T. Lee, and K. G. Powell, "Sonic-point capturing," in *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[19] C.-H. Tai, *Acceleration Techniques for Explicit Euler Codes.* PhD thesis, University of Michigan, 1990.

[20] B. van Leer, C. H. Tai, and K. G. Powell, "Design of optimally-smoothing multi-stage schemes for the Euler equations," in *AIAA 9th Computational Fluid Dynamics Conference*, 1989.