AIAA'89

AIAA-89-0366

THE DESIGN AND APPLICATION OF UPWIND SCHEMES ON UNSTRUCTURED MESHES

Timothy J. Barth and Dennis C. Jespersen
NASA Ames Research Center
Moffett Field, CA

27th Aerospace Sciences Meeting
January 9-12, 1989/Reno, Nevada

The Design and Application of Upwind Schemes on Unstructured Meshes

TIMOTHY J. BARTH[1] AND DENNIS C. JESPERSEN[1]
NASA Ames Research Center, Moffett Field, CA

Abstract. Solution and mesh generation algorithms for solving the Euler equations on unstructured meshes consisting of triangle and quadrilateral control volumes are presented. Cell-centered and mesh-vertex upwind finite-volume schemes are developed which utilize multi-dimensional monotone linear reconstruction procedures. These algorithms differ from existing algorithms (even on structured meshes). Numerical results in two dimensions are presented.

I. Introduction

Algorithms for solving the Euler equations on structured grids are well documented in the technical literature. The spatial derivative approximations used in these algorithms can usually be viewed as central space differencing with some added dissipation or some form of flux split upwind differencing. Even within the context of structured grids, accurate approximations are sometimes difficult to obtain due to solution discontinuities, mesh irregularities, boundary conditions, etc. Even so, reasonable results can be obtained with a modest effort. Recently attention has shifted to the solution of the Euler and Navier-Stokes equations on arbitrary unstructured meshes. This is largely due to impressive results shown by Jameson and Mavriplis [2,3,4] using cell-centered as well as mesh-vertex approximations with some added artificial viscosity. Desideri and Dervieux [5], Rostand and Stoufflet [21] have also shown results for upwind vertex approximations with Osher's flux formula. Unfortunately, these schemes require very careful construction to achieve high order spatial accuracy when the solution is smooth but the mesh is not. Obtaining accuracy estimates is not always easy and many open questions remain [7,18]. In some cases these estimates are not particularly sharp and conclusions can only be obtained from numerical experiment.

In the following sections mesh generation and solution algorithms for solving the Euler equations on unstructured meshes are discussed. In section II, a general mesh generation procedure is described which demonstrates the flexibility of the unstructured mesh technique in discretizing complex (multi-body) geometries. In section III, finite-volume upwind flow solvers are considered. Cell-center and mesh-vertex schemes are developed which utilize a multi-dimensional monotone reconstruction of cell averaged data and Roe's flux function. This differs from the upwind schemes of Refs. [5,21] for unstructured meshes which do not perform reconstructions of cell averaged data and which use the Osher flux function. Finally, numerical results are given in section IV.

II. Mesh Generation

The procedure we use for generating a mixed triangle-quadrilateral mesh about a given configuration is a synthesis of known techniques. There are three major steps: generation of standard body-conforming grids about each component; discarding unwanted points from each grid; and reconnecting the remaining points. In the following description we use as an example the problem of generating a grid about the three-component configuration depicted in Fig. 2.1a (data obtained from L. Wigton, Boeing Commercial Airplane Company).

We first generate body-conforming, logically rectangular meshes about each component of the given configuration. This can be done with any available mesh generation program. Much effort has been expended in developing efficient and robust mesh generation programs for single components, so the task of generating each component mesh is straightforward. See Fig. 2.1b for an example of a mesh about one component of the example configuration. These meshes were generated with a hyperbolic grid generation package [11].

The next step is interactive. Using a graphics workstation, we read each individual grid and plot them on top of one another (see Fig. 2.1c). Then we selectively eliminate points from the individual grids; for example, points from the grid about one component that lie inside another component would certainly be eliminated. We also strive to avoid having points from one component come too close to another component, i.e., we want the grid points near any component only to be those from the grid generated about that component. This will allow keeping some layers of quadrilaterals about each component. The result is shown in Fig. 2.1d. Keeping some layers of quadrilaterals also helps avoid the surface breakthrough problem described in [13].

The final step begins with the selection of a certain

[1]Research Scientist, Computational Fluid Dynamics Branch.

Fig. 2.1a) Three-component configuration.

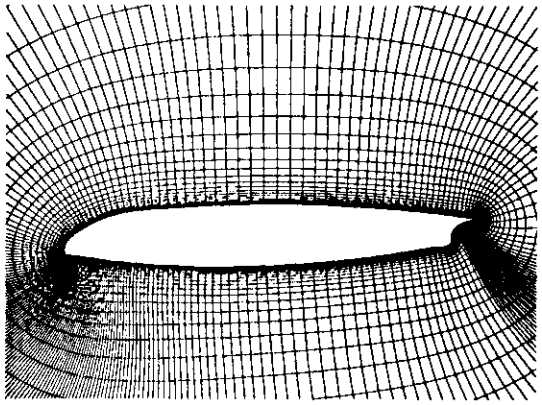

Fig. 2.1b) Grid about main element.

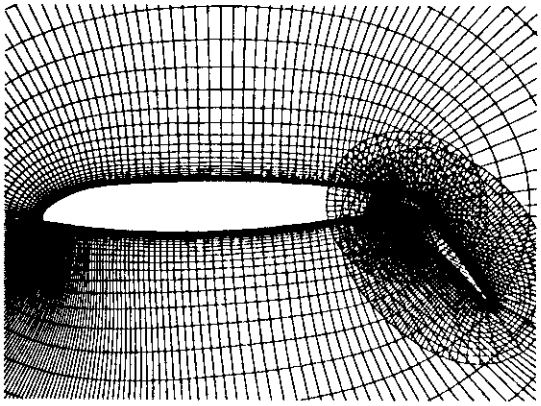

Fig. 2.1c) Grids plotted atop one another.

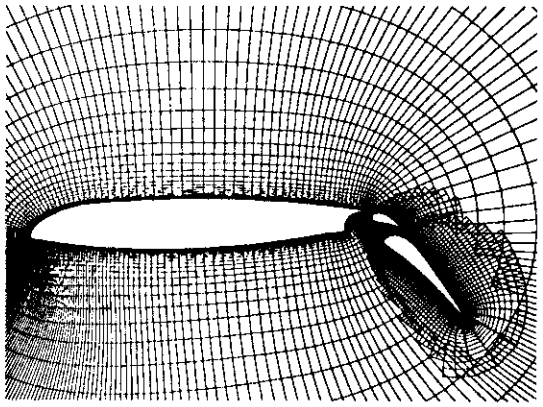

Fig. 2.1d) Grid after elimination of unwanted pts.

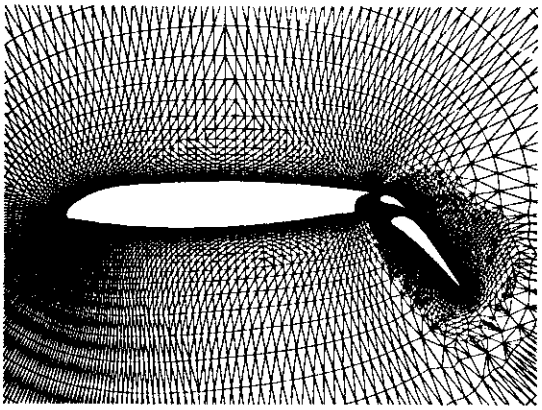

Fig. 2.1e) Grid after reconnection of points.

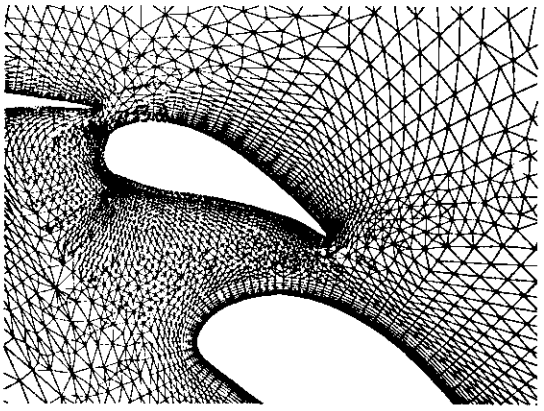

Fig. 2.1f) Grid after reconnection, detail.

Figure (2.1) Mesh generation synthesis of 3 element airfoil.

number of layers of quadrilateral cells to retain near the surface of each component. We are allowing the possibility of retaining some of the body-conforming quadrilaterals near each component surface, as we have in mind the computation of viscous flows with the thin-layer approximation. This also avoids the generation of triangles with extremely small ratio of altitude to base. The points remaining after the quadrilateral layers are reserved are considered as a cloud of points without any connectivity information. A program based on the Delaunay triangulation algorithm of Bowyer [12] then connects the points. The Delaunay triangulation of a set of points can be defined as the dual of the Dirichlet tessellation of the set. The Dirichlet tessellation of a set of points is the pattern of convex regions in the plane, each region being the portion of the plane closer to some given point P of the set of points than to any other point. The Delaunay triangulation is formed by connecting two points if and only if their Dirichlet regions have a common border segment. An alternative characterization of the Delaunay triangulation of a set of points is that three points are connected to form a triangle if and only if the circumcircle of the three points contains no other points of the set. After the Delaunay triangulation has been computed, a certain amount of smoothing may be necessary, replacing the coordinates of selected points by a weighted average of their coordinates and the average of the coordinates of their neighbors. We generally use one or two smoothing sweeps with underrelaxation parameter 0.5. See Figures 2.1e and 2.1f for the resulting triangular grid about the three-component configuration.

III. Upwind Solution Algorithms

Two upwind algorithms for solving the Euler equations will be considered. The algorithms are natural extensions of cell-center and mesh-vertex upwind schemes to unstructured grids. Both schemes build from the first order scheme by performing piecewise linear reconstructions of the cell averaged data. Before beginning the discussion of these algorithms, we briefly digress in order to review basic terminology relevant to unstructured grids.

Review of Mesh - Dual Mesh Geometries

Consider a mesh G of vertices, edges (sides), and faces (cells) denoted by v, e and f respectively. For a given mesh vertex, define the degree of a vertex, $d(v)$, as the number of incident edges to the vertex. Similarly, define the degree of a face, $d(f)$, as the number of edges bounding the face. Note that the number of vertices, edges, and faces $n(v)$, $n(e)$ and $n(f)$ are not independent and are related by Euler's formula

$$(3.1)\quad n(f) = n(e) - n(v) + 1 \qquad \text{(Euler's Formula)}$$

Furthermore, edges and faces are related by the formula

$$(3.2)\quad 2\, n(e)_{interior} + n(e)_{boundary} = \sum_{i=3}^{max(d(f))} i\, n(f)_i$$

where we have defined $n(f)_i$ as the number of faces of a particular face degree, $d(f) = i$. Edges are specified by the two vertices (end-points) that they connect. If the end-points of e are v_i and v_j then $e = (v_i, v_j)$ or $e = (v_j, v_i)$. Vertex pairs are termed adjacent when they describe edges which are members of the mesh edge set. Given the mesh G, we informally define a dual mesh G_{Dual} to be any mesh with the following properties: each vertex of G_{Dual} is associated with a face of G; each edge of G is associated with an edge of G_{Dual}; if an edge separates two faces f_i and f_j of G, then the associated dual edge connects two vertices of G_{Dual} associated with f_i and f_j. Note that edges of G are always assumed straight, but edges of G_{Dual} need not be. In Fig. 3.1, dual edges and faces about the central vertex are shown for duals formed from median segments, centroid segments, and Dirichlet tessellation. Both median and centroid segment faces have been used to construct numerical algorithms, [5,7].

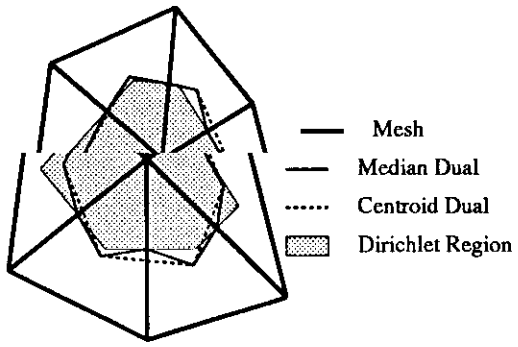

Figure (3.1) Mesh - Mesh Duals.

Note that in the Dirichlet tessellation of triangles, the vertices of the Dirichlet regions are the centers of the circumcircles for each triangle. These points need not lie within the triangle, making the Dirichlet regions a poor candidate for control volumes.

In the implementation of numerical schemes, all edges, faces, and vertices are independently ordered. For application in finite-volume schemes, it is natural to assign a direction to each edge. If $e = (v_i, v_j)$ then the edge is understood to be directed from the first vertex v_i to the second vertex v_j. Each edge of G shares exactly two neighboring faces; indices of these faces are

associated with vertices of G_{Dual} or equivalently the end-points of the corresponding dual edge. Each edge and dual edge pair is assumed cooriented. For example in Fig. 3.2 the edge and dual edge pair are directed such that the z-component of $(\overrightarrow{AB})_{Dual} \times \overrightarrow{AB} \geq 0$. The orientation of the edge and its dual also serve to define the orientation of the "exterior" normal for edges and dual edges. This leads to logically simple algorithms for the implementation of finite-volume schemes on arbitrary unstructured meshes. We will demonstrate this point by example in a subsequent section.

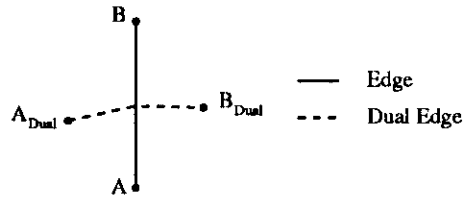

Figure (3.2) Edge - Edge Dual.

Finite-Volume Upwind Discretization

The Euler equations in integral form for a control volume Ω with boundary $\partial\Omega$ are written

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{u}\, dx\, dy + \oint_{\partial\Omega} \bar{\mathbf{f}}(\mathbf{n})\, dl = 0 \tag{3.3}$$

Here $\mathbf{u}$ is the vector of conserved variables and $\bar{\mathbf{f}}$ is the oriented flux along the unit exterior normal $\mathbf{n}$, $\bar{\mathbf{f}}(\mathbf{n}) = n_x\mathbf{f} + n_y\mathbf{g}$ where $\mathbf{f}$ and $\mathbf{g}$ are the Cartesian flux vectors. Numerical approximations of Eq. (3.3) about polygonal faces can be written in the following form (shown here for Euler explicit timestepping, $u^n = u(t_0 + n\ \Delta t)$):

$$\mathbf{u}_j^{n+1} = \mathbf{u}_j^n - \frac{\Delta t}{area_j}\sum_{i=1}^{d(j_j)} \Delta l_i\, \mathbf{h}(\overline{\mathbf{u}}_i^+, \overline{\mathbf{u}}_i^-; \mathbf{n}_i), \quad j = 1, n(f) \tag{3.4}$$

In this equation $\mathbf{h}(\overline{\mathbf{u}}_i^+, \overline{\mathbf{u}}_i^-; \mathbf{n}_i)$ represents the numerical flux (oriented along $\mathbf{n}$) for the i^{th} edge with scalar length Δl_i. This particular numerical flux is assumed to be a function of two states, $\overline{\mathbf{u}}^+$ and $\overline{\mathbf{u}}^-$ which are interpreted as states on either side of the given edge. The Roe flux function [15] is used in all cases and takes the following form:

$$\begin{aligned}\mathbf{h}(\mathbf{u}^+, \mathbf{u}^-; \mathbf{n}) =& \frac{1}{2}(\bar{\mathbf{f}}(\mathbf{u}^+; \mathbf{n}) + \bar{\mathbf{f}}(\mathbf{u}^-; \mathbf{n})) \\ &- \frac{1}{2}|A(\mathbf{u}^+, \mathbf{u}^-; \mathbf{n})|(\mathbf{u}^+ - \mathbf{u}^-)\end{aligned} \tag{3.5}$$

where $|A|$ is a positive definite matrix formed from the flux Jacobian $\partial\bar{\mathbf{f}}/\partial\mathbf{u}$, see [15]. The role of Roe's flux function is to distinguish ingoing and outgoing wave components for each wave family when projected onto the edge normal. We are now left with the specification of the states $\overline{\mathbf{u}}_i^+$ and $\overline{\mathbf{u}}_i^-$. This will distinguish various schemes.

1st Order Upwind Scheme

The first order upwind scheme plays an essential role in the development of higher order nonoscillatory schemes. At solution extrema, the high order schemes described below will revert back to this first order scheme and rely on its monotonicity properties in preventing solution oscillations. In appendix A, a simple derivation and verification of the monotonicity of the first order upwind scheme for scalar advection on arbitrary polygonal control volumes is outlined.

In constructing a first order upwind "cell-center" scheme on the mesh G, the geometry is defined at the vertices of G and the states $\overline{\mathbf{u}}^+$ and $\overline{\mathbf{u}}^-$ are simply the solution values at the vertices of G_{Dual} (this follows from the assumption of a piecewise constant cell distribution). Note that the first order upwind "mesh-vertex" scheme is obtained by taking the dual of this cell-center scheme, i.e. geometry is obtained from vertices of G_{Dual} and the solution variables are accessed at the vertices of G. These simple schemes have many possible implementations on vector supercomputers. For the vertex scheme, we use the following strategy for interior edges: compute numerical fluxes for all dual edges (oriented) and simultaneously distribute to compute the solution update at the vertices of G. This strategy requires one loop through all interior edges. Weak variational boundary conditions can then be applied to all boundary edges, see [2,3,4,5,22] and later discussions. The following pseudo-code details the process for interior edges for one time step $(\Delta u_j = u_j^{n+1} - u_j^n)$

First Order Mesh-Vertex Scheme (interior edges):

For $i = 1, n(e)_{interior}$ *! Loop through interior edges*

$j_1 = e^{-1}(i, 1)$ *! Pointer to 1st end-point of ith edge*

$j_2 = e^{-1}(i, 2)$ *! Pointer to 2nd end-point of ith edge*

h_i=$h(u(j_2), u(j_1); \mathbf{n}_{i_{Dual}})$ *! Oriented numerical flux*

$\Delta u(j_1) = \Delta u(j_1) - \frac{\Delta t\ \Delta l_i\ h_i}{area(j_1)}$ *! j_1 vertex update*

$\Delta u(j_2) = \Delta u(j_2) + \frac{\Delta t\ \Delta l_i\ h_i}{area(j_2)}$ *! j_2 vertex update*

Endfor

This strategy has the distinct advantage that faces of arbitrary degree and in any combination are automatically treated by this single loop. Unfortunately, this algorithm has one major weakness: this loop fails to

vectorize because of the data dependency in accumulating flux contributions at vertices, i.e. two dual edges can simultaneously distribute their portion of the flux to the same vertex location. One way to avoid this difficulty is to order edges into disjoint sets, "edge coloring", such that no two edges of the same "color" share a common vertex. This eliminates the data dependency and loops over a single color vectorize. All implementations of the various schemes exploit data coloring whenever needed.

Higher Order Upwind Schemes

In constructing higher order schemes the assumption of piecewise constant cell distributions is replaced by a linear distribution. Starting with cell averaged data, the solution is piecewise linearly reconstructed such that new extrema are not generated. The fluxes are then evaluated and the solution evolved one time step and averaged over the cell. The process can then be repeated. We note that the use of linear reconstructions does not always imply second or higher order accuracy even when the solution is smooth. This point is addressed in [7,18] for Galerkin-like schemes but has not been addressed for schemes of our type. Also note the solution process requires a multi-dimensional reconstruction. This turns out to be very simple for linear distributions and is discussed in the following section.

Multi-Dimensional Linear Reconstructions of Cell Averaged Data

In the following scheme we attempt to obtain higher order accuracy by performing piecewise linear reconstructions of the cell averaged solution data while insuring that new extrema are not created in the reconstruction process. Although we demonstrate the technique for triangular faces and a cell-center scheme, the technique extends to arbitrary polygonal faces and dual schemes as well. Figure 3.3 depicts the situation for a typical triangle A with (immediate) neighboring triangles B, C, and D.

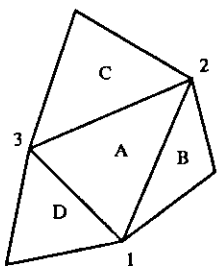

Figure (3.3) Typical Triangular Mesh Configuration

In triangle A, we wish to obtain a linear representation of the state variables about some as yet arbitrary origin (x_0, y_0):

(3.6)
$$\begin{aligned} u(x,y) &= u(x_0,y_0) + u_x \cdot (x - x_0) + u_y \cdot (y - y_0) \\ &= u(x_0,y_0) + \nabla u \cdot \Delta r \end{aligned}$$

We require that this be a redistribution of the cell averaged data, i.e. that

$$u(x_0,y_0) = \frac{1}{area_A} \int_A u(x,y)da \tag{3.7}$$

This is satisfied for any constant gradient vector when (x_0, y_0) is the centroid of triangle A. The vector ∇u represents the best estimate of the solution gradient in the cell computed from surrounding centroid data. (In one dimension this would correspond to the central difference gradient.) The exact form of this gradient is given in the next section for the cell-center and mesh-vertex schemes. Now consider a "limited" form of the reconstructed function about the centroid of triangle A

(3.8)
$$u(x,y)_A = u(x_0,y_0)_A + \Phi_A \ \nabla u_A \cdot \Delta r_A, \quad \Phi \in [0,1]$$

The idea is to find the largest admissible Φ_A while invoking a monotonicity principle that values of the linearly reconstructed function must not exceed the maximum and minimum of neighboring centroid values (including the centroid value in triangle A). Note that this definition differs from that of Harten, Hyman, and Lax [19], but coincides with the monotonicity definition used recently by Spekreijse [20] for structured meshes in multi-dimensions. First compute $u_A^{min} = \min(u_A, u_{neighbors})$ and $u_A^{max} = \max(u_A, u_{neighbors})$ then require that

$$u_A^{min} \le u(x,y)_A \le u_A^{max} \tag{3.9}$$

For linear reconstructions, extrema in $u(x,y)_A$ occur at the vertices of the face and sufficient conditions for (3.8) can be easily obtained. For each vertex of the face compute $u_i = u(x_i, y_i), i = 1, 2, 3, ..., d(f)$ to determine the limited value, Φ_A, which satisfies (3.9):

$$\overline{\Phi}_{A_i} = \begin{cases} \min(1, \frac{u_A^{max} - u_A}{u_i - u_A}), & \text{if } u_i - u_A > 0 \\ \min(1, \frac{u_A^{min} - u_A}{u_i - u_A}), & \text{if } u_i - u_A < 0 \\ 1 & \text{if } u_i - u_A = 0 \end{cases} \tag{3.10}$$

with $\Phi_A = \min(\overline{\Phi}_{A_1}, \overline{\Phi}_{A_2}, \overline{\Phi}_{A_3}, ..., \overline{\Phi}_{A_{d(f)}})$. The states $\overline{u}^+$ and $\overline{u}^-$ for use in (3.5) are then computed as (Δr is a vector from centroid A to a particular edge location)

$$\overline{u} = u_A + \Phi_A \nabla u_A \cdot \Delta r \tag{3.11}$$

Following this procedure guarantees that the linearly reconstructed state variables satisfy the monotonicity principle when evaluated anywhere within a face. For the flux function (3.5), the state variables are only needed at isolated points for each edge. A less restrictive procedure is to guarantee that these points satisfy the monotonicity principle stated above. This amounts to a minor alteration of (3.10), (just evaluate u_i at these points). Numerical evidence shows that both methods work well. Note that in one dimension, the reconstruction process described above reduces to the limited form of Fromm's scheme with maximum compression parameter, see [9,10], i.e., the ratio of adjacent slopes (with the same sign) is allowed to reach 3 before limiting occurs.

Gradient Estimation for Higher Order Schemes

We now consider the construction of the solution gradient ∇u for each face. In Fig. 3.4 we sketch a triangular face A with immediate centroid neighbors B, C, and D. Gradients are computed from approximate forms of the exact relation:

$$\int_\Omega \nabla u \, da = \oint_{\partial\Omega} u \, \mathbf{n} \, ds \tag{3.12a}$$

The solution gradient for face A is estimated by computing the boundary integral of Eq. 3.12a for some path, $\partial\Omega_f$, surrounding face A:

$$\nabla u_A = \frac{1}{a_\Omega} \oint u \, \mathbf{n} \, ds \tag{3.12b}$$

In choosing the path and numerical quadrature for (3.12b), we require that the following two constraints be satisfied:

(A) Exact calculation of ∇u_A when u has linear variation.

(B) ∇u_A must be defined for arbitrary meshes.

Constraint (A) is necessary in order to fulfill the more general design principle that the spatial scheme integrate linear multi-dimensional advection exactly whenever the solution varies linearly in space. Constraint (B) is necessary for algorithm robustness and generality. These constraints are not met by some obvious paths and quadratures. For example in Fig. 3.4 the solution gradient in face A might be calculated using one of the following three strategies:

(1) Choose the perimeter of A for the path $\nabla\Omega_f$ and integrate (3.12b) by approximate midpoint rule, i.e., for the edge $e(v_1, v_2)$ evaluate u at its center by taking the arithmetic average of u_A and u_B. This is similar to the flux quadrature for the first order upwind scheme.

(2) Choose the centroid-centroid path connecting the immediate neighbor set B, C, D and compute the gradient using the trapezoidal rule.

(3) Choose the centroid-centroid path tracing the convex hull of the neighbor set of all centroids that share a common vertex with face A and compute the gradient via the trapezoidal rule (outer dashed segments in Fig. 3.4).

Strategy (1) fails constraint (A) unless the segment connecting adjacent edge centroids bisects the edge. Strategy (2) satisfies constraint (A) but fails (B) and is undefined for distorted geometries when B, C, and D become collinear. Strategy (3) satisfies (A) and (B) for arbitrary geometries and will be used in all the cell-center results presented in Section IV.

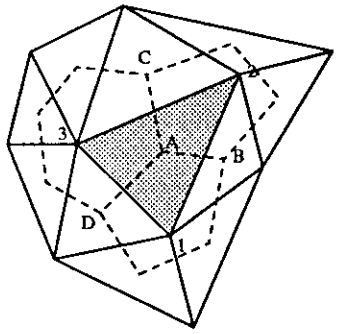

Figure (3.4) Typical triangle with neighbors.

Note that in the vertex scheme the choice of neighbor set is clear and satisfies constraints (A) and (B); gradients are calculated at the vertices of the mesh and the neighbor set is simply all vertices adjacent to a given vertex.

Boundary Conditions

Perhaps the only area in which the cell-center and mesh-vertex schemes differ significantly is the specification of boundary conditions. The implementation of boundary conditions for the cell-center schemes is essentially the same as that employed in structured mesh algorithms. This is especially true when implementing the simpler types of boundary conditions based on "image cells" and reflection principles. Surface boundary condition procedures that solve normal momentum-like equations are only slightly more complicated. Vertex schemes require special attention in implementing boundary conditions. Boundary schemes for structured meshes based on image cells and reflections do not easily extend to unstructured meshes. Since the flow variables are computed at the walls and far-field boundaries, boundary conditions are typically only weakly enforced, i.e., for control volumes containing solid surface boundary edges, boundary fluxes only contain momentum terms arising from

pressure forces. This does not guarantee that the velocity field will be tangent to the solid wall surface. Mavriplis [22] suggests rotating the surface velocity vectors onto a surface tangent vector. Unfortunately the choice of surface tangent vector is not well defined and may not be consistent with the choice of boundary flux. For the results presented below, we employ the boundary procedures suggested by Mavriplis.

IV. Numerical Results

Inviscid Euler flow about single and multi-element airfoil configurations have been numerically calculated using the cell-center and mesh-vertex algorithms described in Section III. Note that in presenting graphical results for the cell-center scheme, we use the reconstruction algorithm to obtain values of the solution at the vertices of the mesh. Because the reconstruction process does not produce a global interpolant, merely a piecewise one, vertex values are obtained by simultaneous interpolation to a vertex from all neighboring cells and arithmetic averaging. Transonic flow calculations ($M_\infty = .8, \alpha = 1.25°$) about a NACA 0012 airfoil geometry have been performed on two meshes of triangles which exhibit high degrees of mesh stretching and irregularity. The first mesh is constructed by subdividing a body-conforming quadrilateral mesh (193x33) as shown in Fig. 4.1a. The stretching near the airfoil surface results in very high cell aspect ratios, a critical test for numerical schemes. Mach number contours and pressure coefficient distributions are plotted in Figs. 4.1b-e for the cell-center and vertex schemes with piecewise constant and linear cell distributions. The poor accuracy of the piecewise constant scheme is clearly seen. In this case the lower surface shock is absent and boundary-layer-like viscous losses are visible in the Mach number contours. The schemes with linear reconstructions easily capture both the upper and lower surface shocks. The pressure coefficient plot shows excellent agreement with a conventional structured grid solution using a TVD algorithm described in [9,10]. Close inspection of the vertex scheme in Fig. 4.1c reveals a small anomaly in near wall Mach number contours downstream of the upper surface shock and trailing edge. These problems appear to be related to the weak enforcement of boundary conditions in the vertex scheme, especially near strong shocks and surface discontinuities. Note that the lower surface solution contours appear normal (even with a weak shock present). Proper boundary condition enforcement for vertex schemes appears to be an important area for future investigation, see also Mavriplis [22] for a discussion of these issues.

The second mesh (see Fig. 4.2a) was constructed by generating weighted pairs of pseudo-random numbers and Delaunay triangulation. Both meshes contain an approximately equivalent number of vertices and similar airfoil surface point distributions. The use of pseudo-random numbers results in a highly irregular mesh although the cell aspect ratio remains relatively close to unity. This type of mesh provides a severe test for numerical schemes. Figures 4.2b-e plot Mach number contours and pressure distributions for the various schemes. As expected, the piecewise constant reconstruction scheme produces poor results. The lower surface shock and trailing edge slip line are absent. Mach number contours appear very irregular, even in regions were the true solution is smooth. The performance of the linear reconstruction algorithms is dramatic. Both upper and lower surface shocks are captured and solution contours away from shocks appear very smooth. Note that subtle differences in the cell-center and vertex schemes can be observed. The shock transition width for the cell-center is slightly narrower (in some places one cell wide) than the vertex scheme, evidently due to the smaller control volumes in the cell-center schemes. This does lead to a slightly more "ragged" shock shape as a result of the irregular mesh. Small contour irregularities can be seen in the cell-center scheme near the airfoil surface although this may be due to interpolating cell-averaged quantities onto the vertices of the mesh for contouring.

As a final example, Euler flow ($M_\infty = .2, \alpha = 0.0°$) is computed over the 3-element airfoil geometry using the vertex scheme with linear reconstructions on the mesh illustrated in Section II. Recall that the mesh is predominantly triangles with a few layers of quadrilateral cells near the surface of the flap elements. Figures 4.3a-d plot pressure coefficient distributions and solution contours. Surface pressures supplied by L. Wigton using a structured mesh and full potential equation solver are also plotted in Fig. 4.3a for comparison. The pressure coefficient distribution compares favorably with the full potential solution although the unstructured grid solution gives a slightly lower pressure coefficient on the upper surface of all three elements. Unfortunately the solution is sensitive to placement of the far-field boundaries and/or the far-field boundary conditions owing to the high lift (circulation) generated by the geometry. The solution is also sensitive to proper resolution of the leading edge suction peak. Both issues need further investigation. The solution contours of Mach number and pressure appear very smooth even in regions were the mesh becomes somewhat irregular. Note that the slip line downstream of the aft flap is virtually nonexistent due to coarseness of the mesh in that area. This is an excellent example where grid adaptation and local mesh refinement would greatly improve the solution quality.

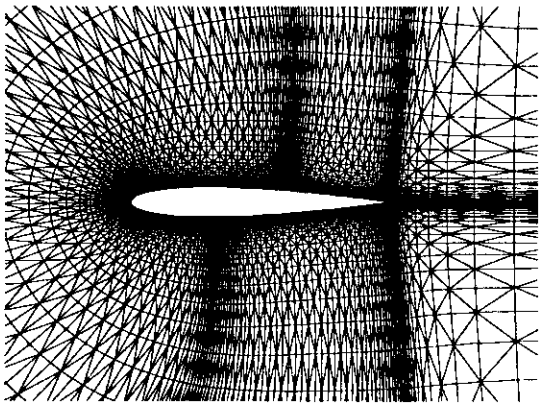

Fig. 4.1a) Closeup of mesh.

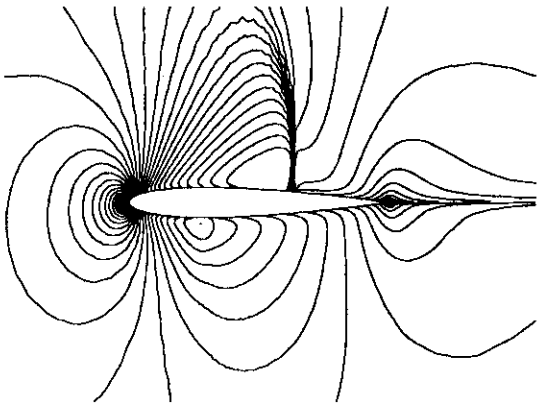

Fig. 4.1b) 1st order vertex scheme.

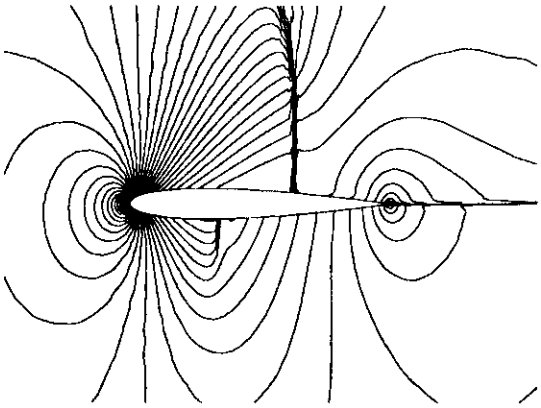

Fig. 4.1c) Higher order vertex scheme.

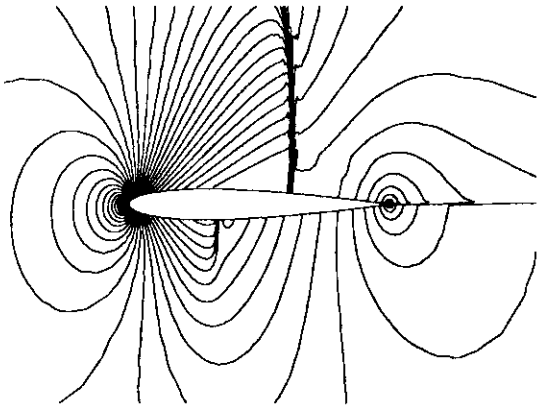

Fig. 4.1d) Higher order cell-center scheme.

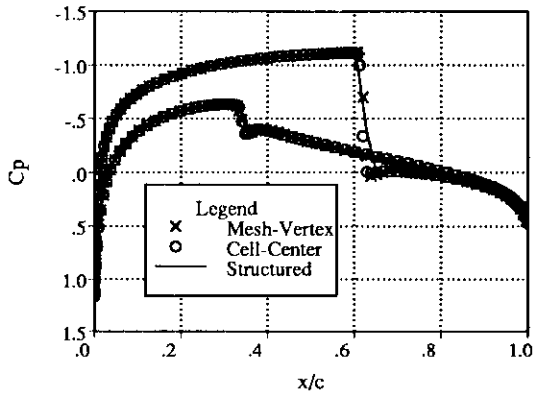

Fig. 4.1e) Cp distribution on airfoil.

Figure (4.1) Grid and Mach Contours for solution on regular mesh.

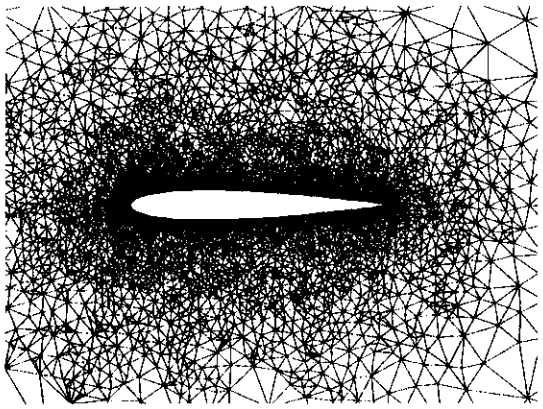

Fig. 4.2a) Closeup of mesh.

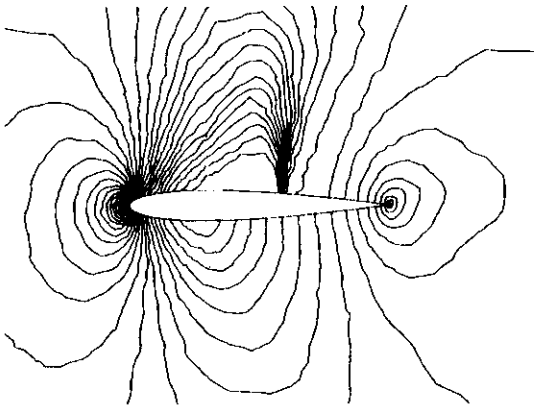

Fig. 4.2b) 1st order vertex scheme.

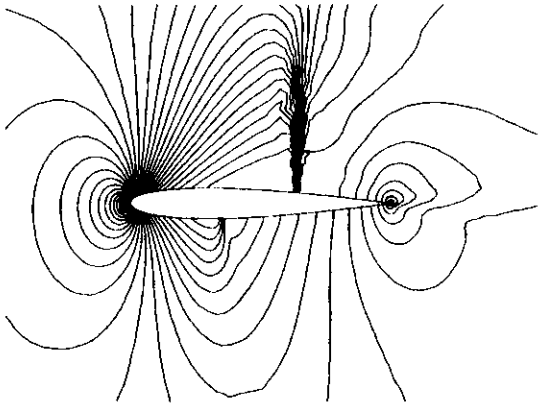

Fig. 4.2c) Higher order vertex scheme.

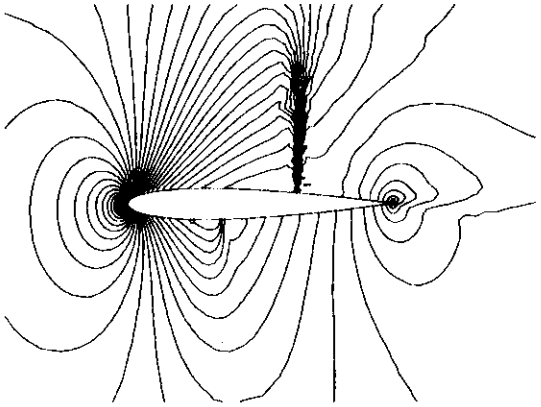

Fig. 4.2d) Higher order cell-center scheme.

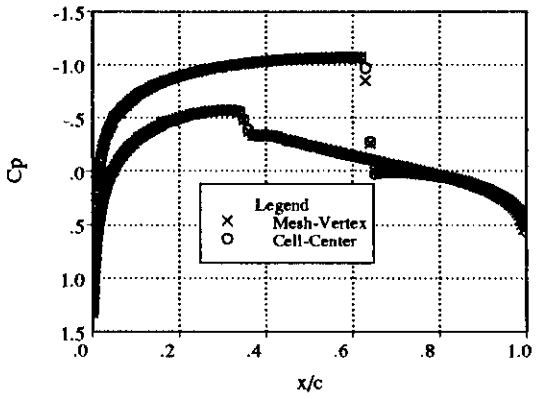

Fig. 4.2e) Cp distribution on airfoil.

Figure (4.2) Grid and Mach Contours for solution on irregular mesh.

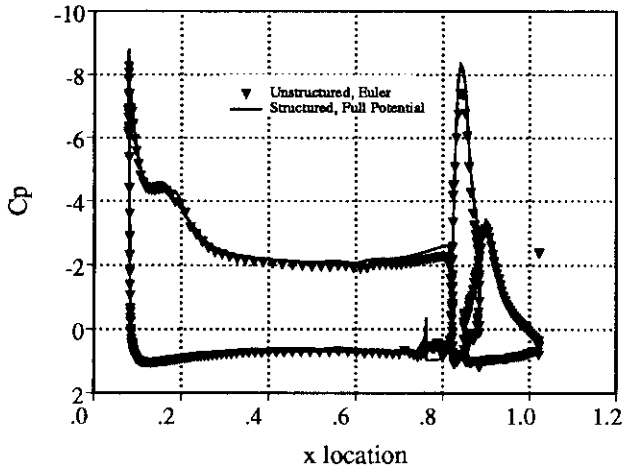

Fig. 4.3a) Cp comparison with full potential solution.

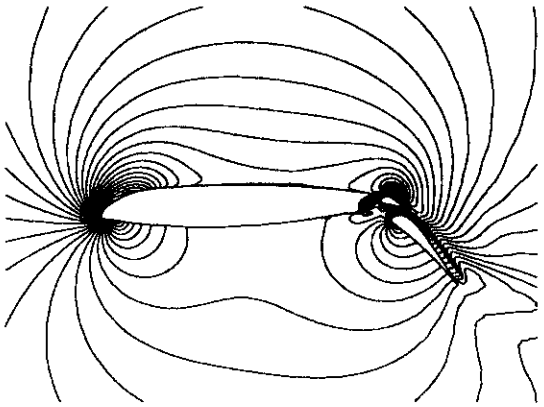

Fig. 4.3b) Mach Contours.

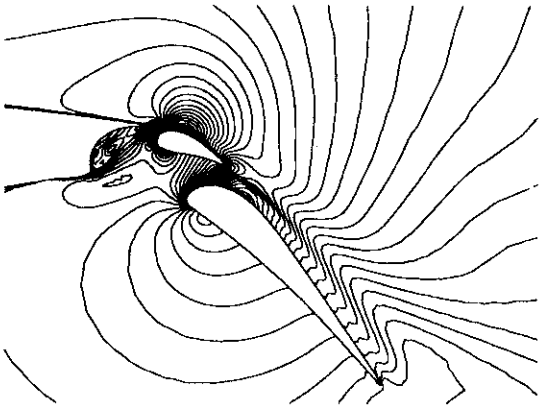

Fig. 4.3c) Mach contours, detail.

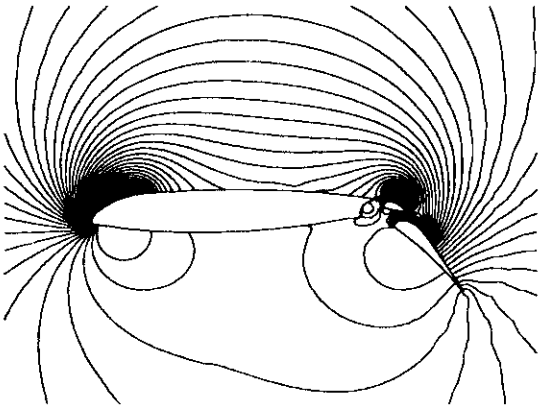

Fig. 4.3d) Pressure Contours.

Figure (4.3) C_p distribution and Solution Contours for 3 element airfoil.

V. Remarks

One obvious challenge in unstructured mesh algorithms is achieving true second order spatial accuracy. For "smoothly" varying meshes this appears to be possible. For meshes with abrupt changes in cell size and orientation the problem is much more difficult. Although we have not shown second order accuracy, the schemes do show drastic improvement over the first order scheme and should provide suitable accuracy for many engineering problems of interest.

VI. Acknowlegements

The authors wish to thank Dr. V. Venkatakrishnan of NASA Langley for numerous helpful conversations and Dr. D.J. Mavriplis at ICASE for supplying the graphics code used to generated the contour plots as well as helpful discussions.

Appendix A. Monotone First Order Upwinding on Arbitrary Control Volumes

Monotonicity of the first order upwind scheme for arbitrary control volumes will be demonstrated for the following multi-dimensional advection equation:

$$u_t + \vec{\lambda} \cdot \nabla u = 0 \tag{A.1}$$

or in integral form

$$\frac{\partial}{\partial t} \int_\Omega u \, da + \oint_{\partial\Omega} (\vec{\lambda} \cdot \mathrm{n}) \, u \, ds = 0 \tag{A.2}$$

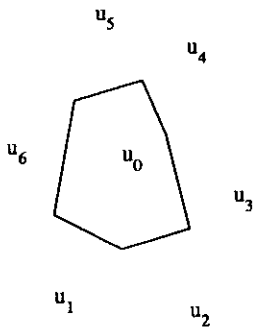

Figure (A.1) Arbitrary Control Volume, $d(f) = 6$

After applying the midpoint approximation and finite sum, the first order upwind scheme is written:

$$\frac{\partial}{\partial t} \int_\Omega u \, da + \sum_{j=1}^{d(f)} (\vec{\lambda} \cdot \mathrm{n}_j) \Delta s_j \, \overline{u}_j = 0 \tag{A.3}$$

with

$$\overline{u}_j = \frac{1}{2}(u_0 + u_j) - \frac{1}{2}\mathrm{sign}(\vec{\lambda} \cdot \mathrm{n}_j)(u_j - u_0) \tag{A.4}$$

After some manipulation we obtain

$$\frac{\partial}{\partial t} \int_\Omega u \, da + \sum_{j=1}^{d(f)} \Delta s_j \left((\vec{\lambda} \cdot \mathrm{n}_j)^+ u_0 + (\vec{\lambda} \cdot \mathrm{n}_j)^- u_j \right) = 0 \tag{A.5}$$

where $(\cdot)^\pm$ are the positive and negative parts. For clarity, we rewrite (A.5) as

$$\frac{\partial}{\partial t} \int_\Omega u da + \beta \; u_0 + \sum_{j=1}^{d(f)} \alpha_i \; u_i = 0 \tag{A.6}$$

with

$$\beta = \sum_{j=1}^{d(f)} \Delta s_j (\vec{\lambda} \cdot \mathrm{n}_j)^+ \; > 0, \quad \alpha_j = \Delta s_j (\vec{\lambda} \cdot \mathrm{n}_j)^- \; < 0$$

For a closed control volume we have $\beta + \sum_{j=1}^{d(f)} \alpha_j = 0$. At steady state we have a maximum principle since the value u_0 equals a positive weighted average of all neighbors:

$$u_0 = \frac{-\sum_{j=1}^{d(f)} \alpha_j \; u_j}{-\sum_{j=1}^{d(f)} \alpha_j} = \frac{\sum_{j=1}^{d(f)} |\alpha_j| \; u_j}{\sum_{j=1}^{d(f)} |\alpha_j|} \tag{A.7}$$

For explicit time stepping a CFL-like condition is obtained for monotonicity. For Euler explicit time stepping we have the time approximation, $\frac{\partial}{\partial t} \frac{1}{area_f} \int_\Omega u_0 da \approx \frac{u_0^{n+1} - u_0^n}{\Delta t}$. We now need to again show positivity of coefficients (in this case coefficients sum to unity)

$$u_0^{n+1} = (1 - \frac{\Delta t}{area_f} \; \beta) \; u_0^n + \frac{\Delta t}{area_f} \sum_{j=1}^{d(f)} |\alpha_j| \; u_j^n \tag{A.8}$$

At the new timestep we have u_o^{n+1} as a positive weighted average of immediate neighbors and itself from the previous time step (a contraction mapping) under the constraint that

$$\Delta t \leq \frac{area_f}{\beta} \tag{A.9}$$

We note in passing that in one dimension this number coincides with the conventional CFL number constraint for first order upwind differencing. In multi-dimensions this constraint is sufficient but not necessary for stability. In two dimensions and depending on the particular control volumes, the typical value of Δt for stability can exceed this value by a factor of 1.5 or more.

VII. References

1. Ni, R.-H.,"A Multiple-Grid Scheme for Solving the Euler Equations", AIAA J., Vol. 20, No. 11, 1982, pp. 1565—1571.
2. Jameson, A. and Mavriplis, D., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh", AIAA paper 85-0435, January 1985.
3. Mavriplis, D., "Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes", ICASE Report No. 87-53.
4. Mavriplis, D. and Jameson, A., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes', AIAA paper 87-0353, January 1987.
5. Desideri, J. A., and Dervieux, A., "Compressible Flow Solvers Using Unstructured Grids", VKI Lecture Series 1988-05, March 7-11, 1988, pp. 1—115.
6. Allmaras, S. R., and Giles, M. B., " A Second Order Flux Split Scheme for the Unsteady 2-D Euler Equations on Arbitrary Meshes", AIAA paper 87-1119, January 1987.
7. Roe, P. L., "Error Estimates for Cell-Vertex Solutions of the Compressible Euler Equations", ICASE Report No. 87-6, January 1987.
8. Roe, P. L., "A Basis for Upwind Differencing of the Two-Dimensional Unsteady Euler Equations", Oxford U. Press, 1986, pp. 55—80.
9. Van Leer, B., " Upwind-Difference Methods for Aerodynamic Problems Governed by the Euler Equations", Lecture Notes in Applied Mathematics, Vol. 22, 1985.
10. Thomas, J. L., van Leer, B., and Walters, R. W., "Implicit Flux-Split Schemes for the Euler Equations", AIAA paper 85-1680, July 1985.
11. Cordova, J. Q., and Barth, T. J., "Grid Generation for General 2-D Regions using Hyperbolic Equations", AIAA paper 88-0520, January 1988.
12. Bowyer, A., "Computing Dirichlet Tessellations", The Computer Journal, Vol. 24, No. 2, 1981, pp. 162—166.
13. Jameson, A., and Baker, T. J., "Improvements to the Aircraft Euler Method", AIAA paper 87-0452, January 1987.
14. Venkatakrisnan, V., and Barth, T. J., "Application of a Direct Solver to Unstructured Meshes for the Euler and Navier-Stokes Equations", Paper submitted to this conference.
15. Roe, P.L.,"Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", J. Comput. Phys., Vol 43, 1981.
16. Barth, T.J., "Implicit Linearization Procedures for Upwind Algorithms", **Computational Mechanics '88**, Springer-Verlag, Vol. 2, pp. 51.i.1-8, 1988.
17. Roe, P. L., "Error Estimates for Cell-Vertex Solutions of the Compressible Euler Equations", ICASE Report No. 87-6, January 1987.
18. Giles, M., "Accuracy of Node-Based Solution on Irregular Meshes", 11th International Conference on Numerical Methods in Fluid Dynamics, Oxford, England, March 1988.
19. Harten, A., Hyman, J., Lax, P, "On Finite-Difference Approximations and Entropy Conditions for Shocks", Comm. Pure Appl. Math, Vol 29, 1976.
20. Spekreijse, S.P., "Multigrid Solution of the Steady Euler Equations", Doctoral Thesis, Deft University of Technology, 1987.
21. Rostand, P. and Stoufflet, B., "A Numerical Scheme for Computing Hypersonic Viscous Flows on Unstructured Meshes", Second International Conference on Hyperbolic Problems, Aachen, W. Germany, 1988.
22. Mavriplis, D. J., "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes", AIAA/ASME/SIAM/APS First National Fluid Dynamics Congress, July 25-28,1988.