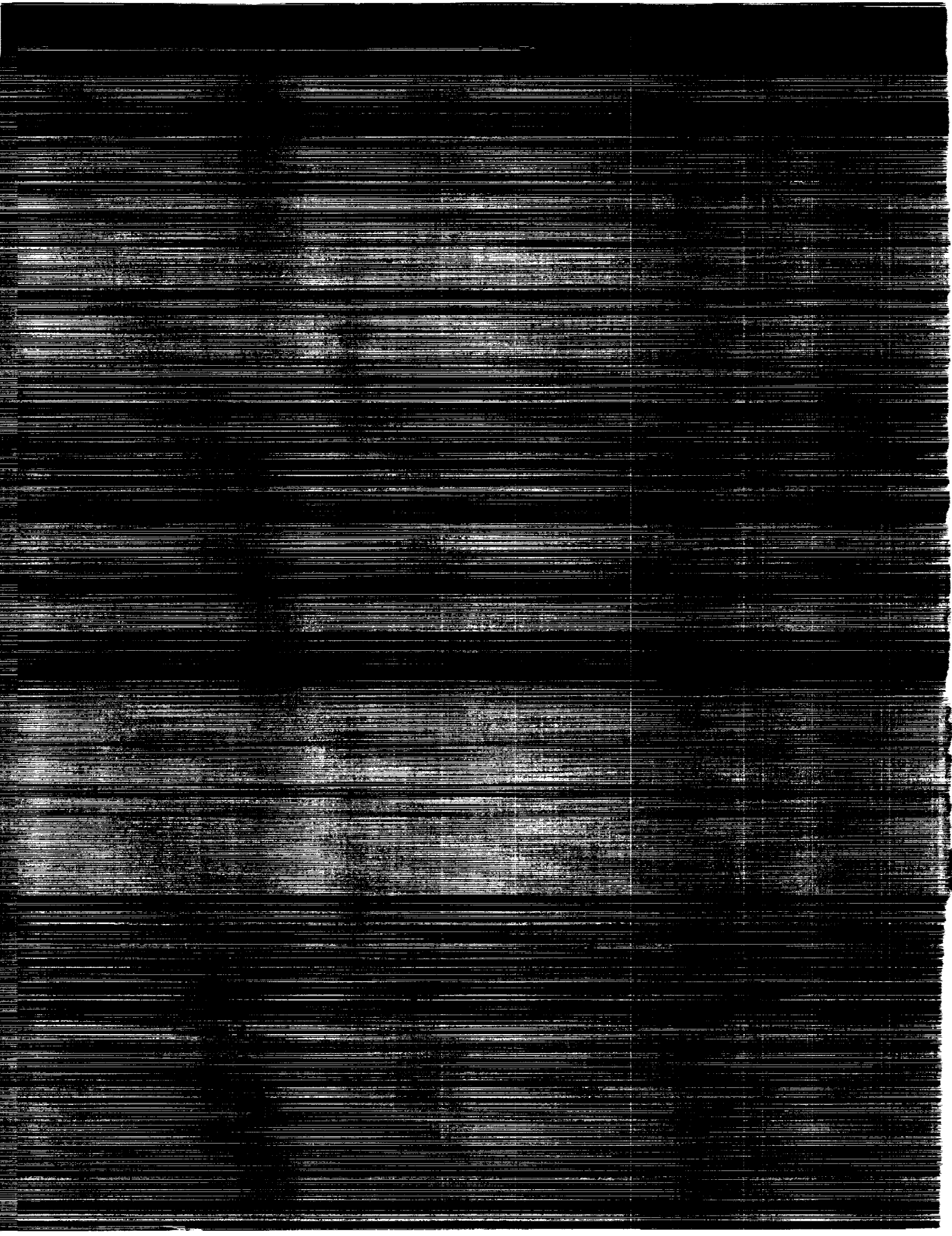
IN-02
84440
P-20

NASA Technical Memorandum 4295

Grid Generation and Flow Solution Method for Euler Equations on Unstructured Grids

W. Kyle Anderson

APRIL 1992



NASA Technical Memorandum 4295

Grid Generation and Flow Solution Method for Euler Equations on Unstructured Grids

W. Kyle Anderson
Langley Research Center
Hampton, Virginia

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1992

Abstract

A grid generation and flow solution algorithm for the Euler equations on unstructured grids is presented. The grid generation scheme, which utilizes Delaunay triangulation, generates the field points for the mesh based on cell aspect ratios and allows clustering of grid points near solid surfaces. The flow solution method is an implicit algorithm in which the linear set of equations arising at each time step is solved using a Gauss-Seidel procedure that is completely vectorizable. In addition, a study is conducted to examine the number of subiterations required for good convergence of the overall algorithm. Grid generation results are shown in two dimensions for an NACA 0012 airfoil as well as a two-element configuration. Flow solution results are shown for a two-dimensional flow over the NACA 0012 airfoil and for a two-element configuration in which the solution has been obtained through an adaptation procedure and compared with an exact solution. Preliminary three-dimensional results also are shown in which the subsonic flow over a business jet is computed.

Introduction

The use of unstructured grids for a solution of the Euler equations offers several advantages over the use of structured grids. These advantages include the ease with which adaptive methodology can be incorporated into the flow solvers and the relatively short time to generate grids about complex configurations. Although the overall time to generate grids about complex configurations is much shorter for unstructured grids compared with that of block-structured grids, the computer time required for the unstructured flow solvers historically has been much longer than that of structured grids. Although unstructured flow solvers will continue to require longer computer times than those of structured grids because of indirect addressing, recent advances (refs. 1 and 2) make three-dimensional computations on unstructured grids more competitive with those of structured grids.

As mentioned, the success of unstructured grids mainly is due to the relative ease at which grids can be obtained over complex configurations. Two dominant methods of generating unstructured grids currently exist. The first of these techniques is the advancing-front method in which the cells that make up the interior of the mesh are computed by marching away from the domain boundaries (refs. 3 and 4). This method has been used with success to generate grids about many complex configurations (ref. 5). Further details of this technique can be found in references 3 to 6 and the references contained therein.

The other method commonly used for generation of unstructured grids is Delaunay triangulation (refs. 7 and 8), which is emphasized in the current study. This approach triangulates a given set of points in a unique way so that the minimum angle of each triangle in the mesh is maximized. The advantage of this technique is that the resulting meshes are optimal for the given point distribution because they do not usually contain many extremely skewed cells.

The field points for generating grids using the Delaunay triangulation approach usually are specified a priori by generating points about individual components with structured grids (ref. 9), by subdividing existing quadrilateral cells using a quadtree encoding method (ref. 6), or by embedding the geometry into a Cartesian grid (ref. 10). A novel approach to the generation of field points is given by Holmes and Snyder (ref. 11); in this approach, the field points are generated as the triangulation proceeds based on the aspect ratio and cell area of current triangles. This technique generates grids that are not highly skewed because new points are introduced to continually reduce the cell aspect ratios. Unfortunately, grids generated in this manner generally are too coarse to be used for obtaining accurate flow-field solutions without adaptation.

In the present study, an approach similar to that of Holmes and Snyder is used, and an extension is incorporated which automatically adds new nodes to cluster points in the regions of interest. Using the new generator, grids that are suitable for computations are efficiently generated around complex, multibody configurations.

Many advances also have been made in flow solvers for obtaining flow-field solutions on unstructured grids. Impressive results, in which solutions are found for a wing configuration using a node-based, central-differencing scheme with multigrid to achieve rapid convergence, have been obtained by Mavriplis (ref. 2). In this reference, solutions on a three-dimensional grid consisting of more than 2 million cells are obtained in approximately 1 hour.

For upwind solvers, Frink et al. (ref. 1) have generated results for many steady-state applications using a cell-centered, multistage time-stepping scheme and Roe's approximate Riemann solver (ref. 12). For unsteady applications, Batina (ref. 13) has developed both explicit and implicit algorithms for obtaining aeroelastic applications, and Rausch et al. (ref. 14) have coupled some of these methods with adaptive mesh refinement.

In this study, an implicit algorithm for solving the Euler equations is described. This method, along with the work in references 13 and 15, is based on the backward-Euler time-differencing scheme, but it is formulated in a manner that permits full vectorization. In addition, the number of subiterations necessary to sufficiently solve the linear problem and to obtain the best convergence rate is examined. Results are shown for both two- and three-dimensional calculations.

The author acknowledges Daryl Bonhaus for generating the grid around the business jet.

Symbols

$\mathbf{A}$	matrix
A	area of cell
a	speed of sound
body	conditions on body
CFL	Courant-Friedrichs-Lewy number
C_p	pressure coefficient
c	chord length
$\mathbf{D}$	diagonal components of $\mathbf{A}$
d	distance to nearest surface node
E	total energy per unit volume
$\mathbf{F}$	fluxes of mass, momentum, and energy
$\widehat{\mathbf{F}}$	fluxes normal to cell face
$\widehat{\mathbf{F}}^{\pm}$	split fluxes
f	function used for clustering grid points
$\mathbf{I}$	identity matrix
l_i	length of cell face i
$\mathbf{M}$	components of $\mathbf{A}$ below diagonal
M_n	Mach number normal to cell face
M_∞	free-stream Mach number
$\mathbf{N}$	components of $\mathbf{A}$ above diagonal
N	total number of cells
$\hat{\mathbf{n}}$	unit normal
n	number of edges meeting at node
$\hat{n}_x, \hat{n}_y$	x and y components of unit normal
$\mathbf{O}$	all off-diagonal components of $\mathbf{A}$
p	pressure
$\mathbf{Q}$	conserved state vector, $\mathbf{Q} = [\rho \quad \rho u \quad \rho v \quad E]^T$
$\mathbf{q}$	primitive state vector, $\mathbf{q} = [\rho \quad u \quad v \quad p]^T$
$\mathbf{R}$	residual for cell
$R^{\pm}$	Riemann invariants
$\mathbf{r}$	vector from center of cell to center of edge
ref	reference condition
S	entropy
t	time
U	velocity normal to cell face
u, v	Cartesian velocities in x and y directions
x, y	Cartesian coordinates
α	angle of attack
β	parameter used for grid clustering
γ	ratio of specific heats, taken as 1.4
η	percent of spanwise location on wing
ρ	density
σ	standard deviation of ϕ
ϕ	function used for grid clustering
$\bar{\phi}$	average value of ϕ
Ω	boundary of cell
ω	relaxation factor

Two-Dimensional Grid Generation

Delaunay Triangulation

The foundation of the proposed grid generation procedure is the Delaunay triangulation method described in detail in reference 7. In this technique, a set of points is triangulated by inserting each point, one at a time, into a current triangulation so that no vertex from one triangle lies within the circumcircle of any other triangle. The procedure is initiated by first identifying all the cells that have a circumcircle enclosing the point to be inserted. An example is shown in figure 1; in this figure, the point to be inserted lies within the circumcircle of two triangles. The Delaunay cavity, shown in figure 2, then is formed from the union of all the triangles identified previously. At this stage, a new triangulation is made by simply connecting the new point to each of the nodes lying on the boundary of the Delaunay cavity, as depicted in figure 3.

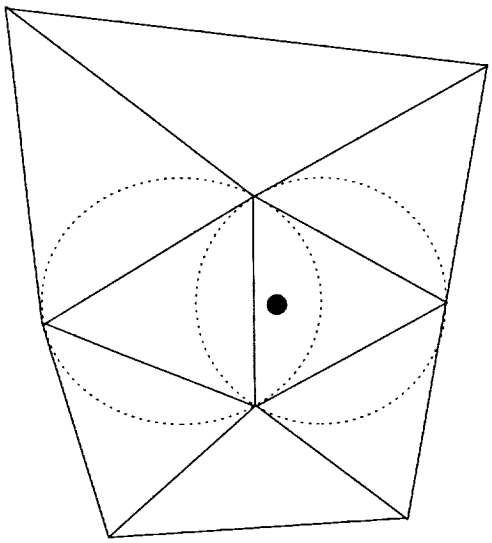


Figure 1. Identifying cells broken by introducing new point.

To generate grids about arbitrary two-dimensional configurations, an initial triangulation consisting of a square divided into two triangles is formed first. This square has four corner points located a sufficient distance from all solid surfaces. The points that define the solid surfaces then are inserted using Bowyer's algorithm (ref. 7), followed by a predetermined number of far-field points that are located in a circular pattern which is a specified radius from the center of the bodies. The cells that make up the interior of the body then are identified according to whether the center of each cell is located inside or outside one of the bodies.

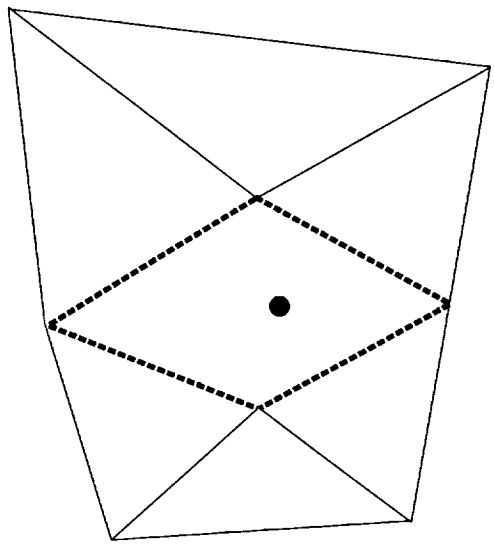


Figure 2. Delaunay cavity.

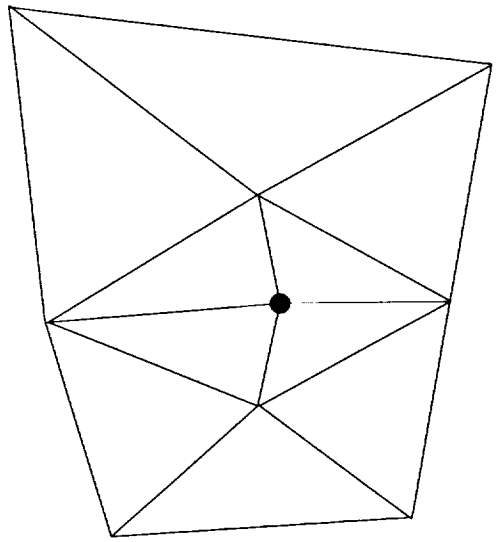


Figure 3. Reconnection of grid after inserting new point.

After this initial phase of the process, a loop is conducted over all the cells, and a new point is immediately introduced at the center of the circumcircle of any triangle that has an aspect ratio (defined as the ratio of the circumcircle radius to twice the in-circle radius) exceeding a predetermined tolerance of approximately 1.5. The surface integrity is maintained by rejecting any point that would result in breaking the cells that make up the airfoil interior (ref. 8). Note that when a cell aspect ratio is larger than the tolerance, the new point is immediately added into the existing triangulation. This addition prevents duplicate points from being added when two triangles have points that define the same circumcircle.

Immediately adding points in this manner eliminates the searching that is otherwise necessary to identify the first triangle, which is broken by the addition of the current point. The elimination is possible because the cell that has an aspect ratio greater than the tolerance will also correspond to one of the triangles which forms the Delaunay cavity. Because no searching is required, the computer time necessary to generate the field points in this manner is small. This addition of new points is similar to that of the method used in reference 11 in which new points are introduced based on both the cell area and the aspect ratio.

An example of this process is shown in figures 4 through 9 for a sample grid around an NACA 0012 airfoil. The airfoil surface is defined with 200 points along the surface and 32 points placed around the outer boundary. Note that the outer boundary is placed close to the airfoil for illustrative purposes, thus allowing the entire grid to be seen. Figure 4 shows the initial triangulation in which only the surface points and the outer boundary points have been included; this triangulation has cells with aspect ratios as high as 160.

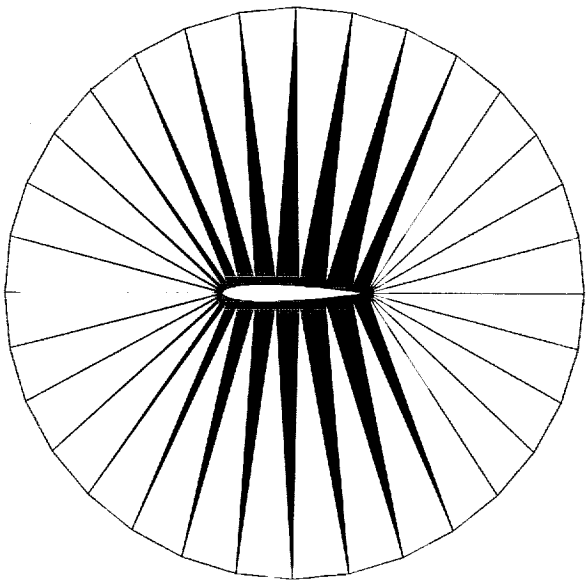


Figure 4. Initial sample grid around NACA 0012.

New points now are introduced at the center of the circumcircle of any cell that has an aspect ratio exceeding 1.5. Figures 5 to 8 show a few of the intermediate triangulations after inserting the first, second, third, and fourth points, respectively.

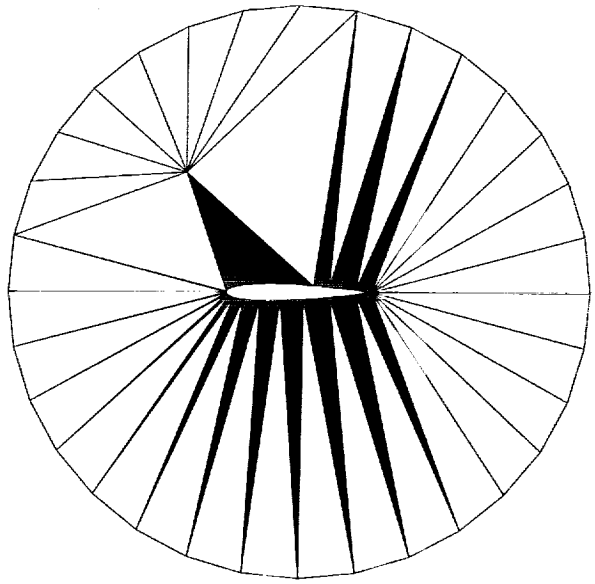


Figure 5. Sample grid around NACA 0012 after inserting one point.

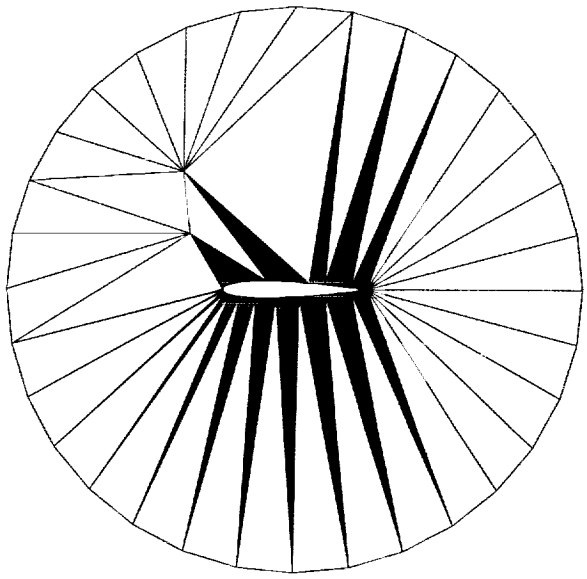


Figure 6. Sample grid around NACA 0012 after inserting two points.

The final grid obtained by adding field points in this manner is shown in figure 9. This grid, which consists of 1328 nodes, 3748 faces, and 2420 cells, has a maximum aspect ratio of 1.495. Although all the resulting cells are nearly equilateral, the grids generated with this technique are coarse a short distance from the airfoil and are not sufficient for accurate computations. Therefore, increasing the grid density in the vicinity of the airfoil is necessary.

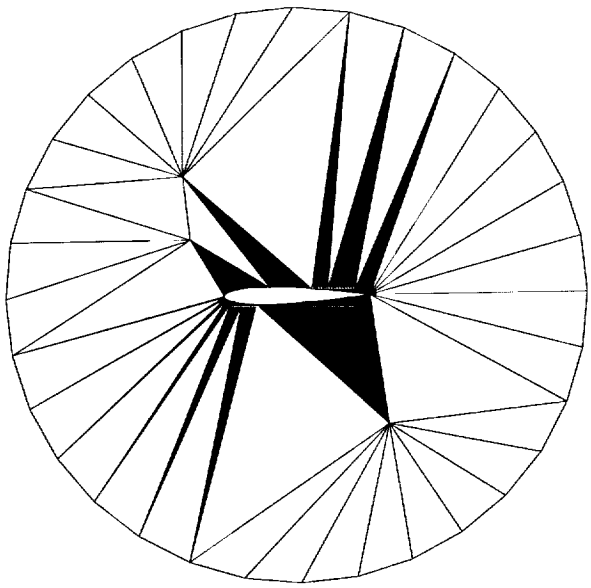


Figure 7. Sample grid around NACA 0012 after inserting three points.

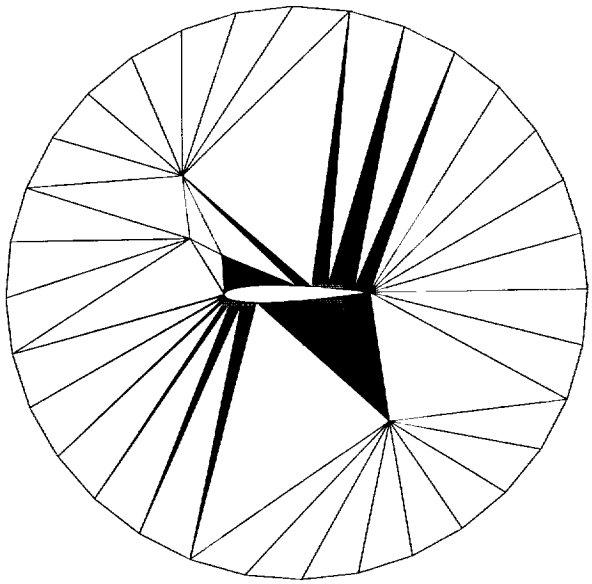


Figure 8. Sample grid around NACA 0012 after inserting four points.

Extensions for Clustering Mesh Points

To add new points in the vicinity of the airfoil, a value is first assigned to each existing cell; this value is the product of the cell area and a weighting function that decreases as the distance from the cell center to a solid surface increases:

$$\phi(A, d) = A \times f(d) \tag{1}$$

In this equation, d is the distance from the cell center to the nearest node that lies on a solid boundary.

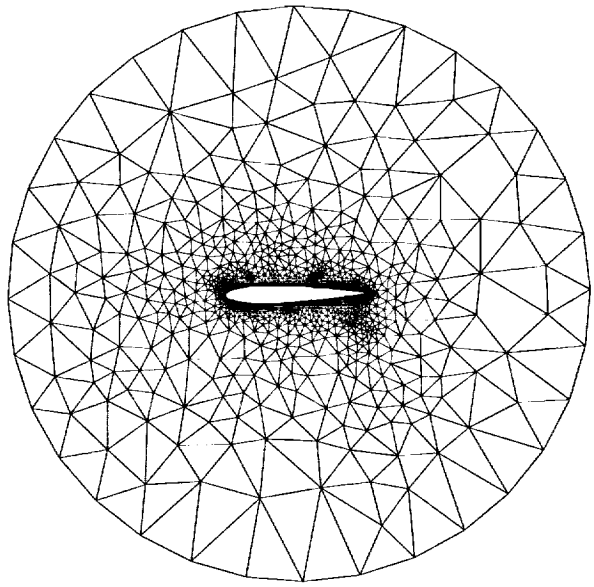


Figure 9. Final sample grid around NACA 0012 with all aspect ratios <1.5.

This variable will be used to add subsequent points in cells in which the deviation of ϕ from the average is larger than the standard deviation. For this reason, the average and standard deviations of this variable are first computed:

$$\bar{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi_i \tag{2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (\bar{\phi} - \phi_i)^2}{N}} \tag{3}$$

A list of new points that will be inserted into the existing grid then is constructed from the cell centers of all triangles in which the local value of $\phi(A, d)$ exceeds that of the average plus the standard deviation, i.e., whenever $\phi_i \geq \bar{\phi} + \sigma$. This list of new points then is introduced as before, using Bowyer's algorithm. By adding new points in this manner, the function ϕ tends to be evenly distributed over the grid, and new points are added first at larger cells near the body. Few, if any, new points are introduced far from the solid surfaces.

The weighting function used in the current study is given by

$$f(d) = \frac{1}{1 + e^{\beta(d - d_0)}} \tag{4}$$

In this equation, d_0 is a distance that is measured from the airfoil surface; clustering will occur

predominantly in regions where the distance to the airfoil surface is less than d_0. A plot of this function is shown in figure 10 for several values of β and $d_0 = 0.5$. As seen, the transition of this function at $d = 0.5$ steepens as β increases, and the value decreases as the distance from the airfoil increases. Thus, the transition between clustered and nonclustered regions can be made smoothly, and the distance away from the airfoil in which clustering occurs is also controlled. Note that because this procedure only adds one point at the center of each triangle, the amount of clustering for the final grid (i.e., how many new points are introduced) is increased by repeating this procedure several times. In practice, three or four repetitions in which β is gradually increased lead to grids with good clustering near the surface of the airfoils, and a reasonably smooth transition region between the clustered and nonclustered areas is obtained. Further enhancements to this procedure may be achieved by varying the weighting function.

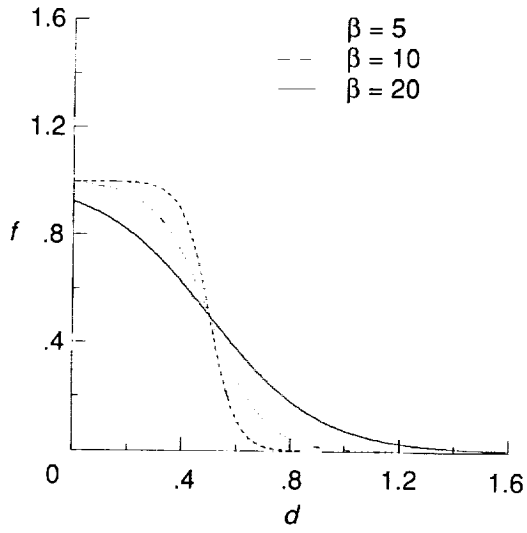


Figure 10. Weighting function for several values of β and $d_0 = 0.5$.

The final step is to smooth the grid with a simple Laplacian-type procedure as given in reference 16. This process is achieved by repositioning the mesh points according to

$$\left.\begin{aligned} x_i^{n+1} &= x_i^n + \frac{\omega}{n}\sum_{k=1}^{n}(x_k - x_i) \\ y_i^{n+1} &= y_i^n + \frac{\omega}{n}\sum_{k=1}^{n}(y_k - y_i) \end{aligned}\right\} \qquad (5)$$

where ω is a relaxation factor and the sum is obtained over all edges meeting at node i. For the current study, a relaxation factor of 0.2 is typically used, and 100 to 200 iterations of smoothing are performed.

The final sample grid for the NACA 0012 airfoil is shown in figure 11. This grid, which demonstrates the success of the clustering procedure, is a clear improvement to the grid previously shown in figure 9.

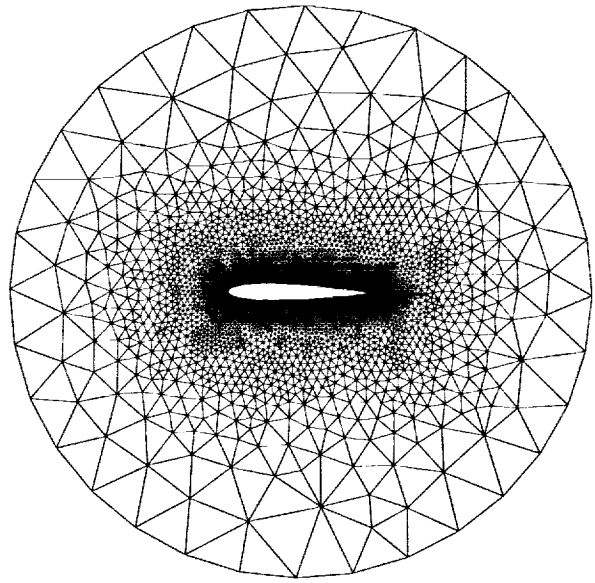


Figure 11. Final sample grid around NACA 0012.

Although the techniques just outlined currently have not been implemented in three dimensions, no readily apparent implementation obstacle exists.

Euler Solver

The Euler flow solver is an implicit, cell-centered, upwind-differencing code in which the fluxes on the cell faces are obtained using the Van Leer flux-vector splitting technique (ref. 17). The solution at each time step is updated using an implicit algorithm that uses the linearized, backward-Euler, time-differencing scheme. At each time step, the linear system of equations is solved with a subiterative procedure in which the mesh cells are divided into groups (or colors) so that no two cells in a given group share a common edge. For each subiteration, the solution is obtained by solving all the unknowns in a given color before proceeding to the next color. Because the solution of the unknowns in each group depends on those from previous groups, a Gauss-Seidel-type procedure that is completely vectorizable is obtained.

Governing 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

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \oint_{\Omega} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \, d\Omega = 0 \tag{6}$$

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

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \tag{7}$$

$$\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} = \widehat{\mathbf{F}} = \begin{bmatrix} \rho U \\ \rho U u + \hat{n}_x p \\ \rho U v + \hat{n}_y p \\ (E + p) U \end{bmatrix} \tag{8}$$

and U is the velocity in the direction of the outward pointing unit normal to a cell face

$$U = \hat{n}_x u + \hat{n}_y v \tag{9}$$

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

$$p = (\gamma - 1) \left[E - \rho \left(u^2 + v^2 \right) / 2 \right] \tag{10}$$

Flux-Vector and Residual Calculation

For the computations shown in this report, the flux vectors in equation (8) are upwind differenced using the flux-vector splitting technique of Van Leer (ref. 17). These flux vectors are given in terms of the Mach number normal to the cell face, defined as $M_n = U/a$. For supersonic flow in the direction of a face normal ($M_n \geq 1$)

$$\widehat{\mathbf{F}}^{+} = \left(\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \right)^{+} = \mathbf{F} \qquad \widehat{\mathbf{F}}^{-} = \left(\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \right)^{-} = 0 \tag{11}$$

whereas for supersonic flow in the opposite direction of the face normal ($M_n \leq -1$)

$$\widehat{\mathbf{F}}^{-} = \left(\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \right)^{-} = \mathbf{F} \qquad \widehat{\mathbf{F}}^{+} = \left(\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \right)^{+} = 0 \tag{12}$$

For subsonic flow ($|M_n| < 1$), the fluxes are split into two contributions, $\widehat{\mathbf{F}}^{+}$ and $\widehat{\mathbf{F}}^{-}$, such that the Jacobian matrix of $\widehat{\mathbf{F}}^{+}$ has positive eigenvalues and the Jacobian matrix of $\widehat{\mathbf{F}}^{-}$ has negative eigenvalues. The split fluxes are given by

$$\widehat{\mathbf{F}}^{\pm} = \left(\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \right)^{\pm} = \begin{Bmatrix} f^{\pm}_{\text{mass}} \\ f^{\pm}_{\text{mass}} \left\{ \left[\hat{n}_x \left(-U \pm 2a \right) / \gamma \right] + u \right\} \\ f^{\pm}_{\text{mass}} \left\{ \left[\hat{n}_y \left(-U \pm 2a \right) / \gamma \right] + v \right\} \\ f^{\pm}_{\text{energy}} \end{Bmatrix} \tag{13}$$

where

$$f^{\pm}_{\text{mass}} = \pm \rho a \left(M_n \pm 1 \right)^2 / 4 \tag{14}$$

and

$$f^{\pm}_{\text{energy}} = f^{\pm}_{\text{mass}} \left[\frac{(1 - \gamma) U^2 \pm 2 (\gamma - 1) U a + 2a^2}{\gamma^2 - 1} + \frac{u^2 + v^2}{2} \right] \tag{15}$$

The steady-state residual, given by

$$\mathbf{R} = - \oint_{\Omega} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \, d\Omega \tag{16}$$

is calculated using a trapezoidal integration by summing the fluxes over each of the faces that make up the control volume. For example, the residual in a triangular cell is calculated as

$$\mathbf{R} = - \oint_{\Omega} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \, d\Omega = - \sum_{i=1}^{i=3} \left[\widehat{\mathbf{F}}^{+} \left(\mathbf{Q}_i^{-} \right) + \widehat{\mathbf{F}}^{-} \left(\mathbf{Q}_i^{+} \right) \right] l_i \tag{17}$$

Here $\widehat{\mathbf{F}}^{\pm}(\mathbf{Q}^{\mp})$ represents the split fluxes on the cell faces formed from an upwind interpolation of the data to each face. For first-order accurate differencing, the data on the face are obtained from the data in the cells that lie on each side of the cell face. For higher order differencing, the primitive variables are extrapolated to the cell faces using a Taylor series expansion about the center of the cell so that the data on the face are given by

$$\mathbf{q}_{\text{face}} = \mathbf{q}_{\text{center}} + \nabla \mathbf{q} \cdot \mathbf{r} \tag{18}$$

where $\mathbf{r}$ is the vector extending from the center of the cell to the center of the cell face.

For evaluating the gradient $\nabla \mathbf{q}$, the data first are interpolated to the nodes using inverse distance weighting, and the gradient then is evaluated using Green's theorem. This interpolation method is further discussed in reference 18. Note that obtaining the data at the nodes also has been accomplished using a linear least-squares fit of the data in the surrounding cells with no apparent differences observed in the solutions obtained with either method.

Boundary Conditions

The boundary conditions on the body are set according to characteristic-type boundary conditions similar to those in reference 19. The density, pressure, and velocity components on the body are determined according to

$$p_{\text{body}} = p_{\text{ref}} + \rho_{\text{ref}} a_{\text{ref}} \left(\hat{n}_x u + \hat{n}_y v\right) \tag{19}$$

$$\rho_{\text{body}} = \rho_{\text{ref}} + \left(p_{\text{body}} - p_{\text{ref}}\right) / a^2_{\text{ref}} \tag{20}$$

$$u_{\text{body}} = u_{\text{ref}} - \hat{n}_x \left(\hat{n}_x u + \hat{n}_y v\right)_{\text{ref}} \tag{21}$$

$$v_{\text{body}} = v_{\text{ref}} - \hat{n}_y \left(\hat{n}_x u + \hat{n}_y v\right)_{\text{ref}} \tag{22}$$

from which the energy is set using the equation of state given in equation (10). The reference conditions for equations (19) through (22) are taken from the first cell in the grid interior.

Because an implicit scheme is used in this study, implicit boundary conditions are implemented by assuming that

$$\Delta\rho_{\text{body}} = \Delta\rho_{\text{ref}} \tag{23}$$

$$\Delta\left(\rho u\right)_{\text{body}} = \Delta\left(\rho u\right)_{\text{ref}} - \hat{n}_x \left(\hat{n}_x \Delta\left(\rho u\right) + \hat{n}_y \Delta\left(\rho v\right)\right)_{\text{ref}} \tag{24}$$

$$\Delta\left(\rho v\right)_{\text{body}} = \Delta\left(\rho v\right)_{\text{ref}} - \hat{n}_y \left(\hat{n}_x \Delta\left(\rho u\right) + \hat{n}_y \Delta\left(\rho v\right)\right)_{\text{ref}} \tag{25}$$

$$\Delta E_{\text{body}} = \Delta E_{\text{ref}} \tag{26}$$

In this manner, the matrix entries that correspond to cells lying adjacent to a solid surface can be easily modified to include the boundary influence.

For the far field, explicit boundary conditions are used in which the velocity and speed of sound are obtained from two locally one-dimensional Riemann invariants given by

$$R^{\pm} = U \pm \frac{2a}{\gamma - 1} \tag{27}$$

These invariants are considered constant along characteristics that are defined normal to the outer boundary. For subsonic conditions at the boundary, R^- can be evaluated locally from free-stream conditions outside the computational domain, and R^+ is evaluated locally from the interior of the domain. The local normal velocity and speed of sound on the boundary are calculated using the Riemann invariants as

$$U_{\text{boundary}} = \frac{1}{2}\left(R^+ + R^-\right) \tag{28}$$

$$a_{\text{boundary}} = \frac{\gamma - 1}{4}\left(R^+ - R^-\right) \tag{29}$$

The Cartesian velocities are determined on the outer boundary by decomposing the normal and tangential velocity vectors into components that yield

$$\left.\begin{aligned} u_{\text{boundary}} &= u_{\text{ref}} + \hat{n}_x \left(U_{\text{boundary}} - U_{\text{ref}}\right) \\ v_{\text{boundary}} &= v_{\text{ref}} + \hat{n}_y \left(U_{\text{boundary}} - U_{\text{ref}}\right) \end{aligned}\right\} \tag{30}$$

where the subscript ref represents values obtained from one point outside the domain for inflow and from one point inside the domain for outflow.

The entropy is determined by using the value from either outside or inside the domain, depending on whether the boundary is an inflow or outflow boundary. Once the entropy is known, the density on the far-field boundary is calculated from the entropy and speed of sound as

$$\rho_{\text{boundary}} = \left(\frac{a^2_{\text{boundary}}}{\gamma S_{\text{boundary}}}\right)^{\frac{1}{\gamma - 1}} \tag{31}$$

The energy then is calculated from the equation of state.

Time Advancement Schemes

Implicit algorithms. The starting point for the time advancement algorithm is the linearized, backward-Euler, time-differencing scheme that yields a system of linear equations for the solution at each step given by

$$[\mathbf{A}]^n \{\Delta\mathbf{Q}\}^n = \{\mathbf{R}\}^n \tag{32}$$

where

$$[\mathbf{A}]^n = \frac{A}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}} \tag{33}$$

The solution of equation (32) can, in principle, be obtained by a direct inversion of $[\mathbf{A}]^n$; this solution has the advantage of a resulting scheme that becomes a Newton iteration in the limit as the time step approaches infinity if the exact linearization of $\mathbf{R}^n$ is used in forming $[\mathbf{A}]^n$. Although this technique is quite successful in two dimensions (ref. 20), the solution at each time step requires a great deal of memory to store the components of $[\mathbf{A}]^n$ as well as extensive computer time to perform the matrix inversions. This approach, therefore, is currently not very feasible for practical calculations in three dimensions.

Because the number of operations required to invert a matrix depends on the matrix bandwidth,

first-order accurate approximations on the left-hand side of equation (32) are often utilized to reduce both required storage and computer time. With this simplification, the consistency between the left- and right-hand sides of equation (32) requires that first-order approximations also be used on the right-hand side to achieve quadratic convergence. However, with first-order approximations on the left-hand (implicit) side and second-order approximations on the right-hand side, this scheme remains stable for large time steps. First-order differencing of the left-hand side with higher order differencing on the right-hand side, therefore, is considered in the present study.

A sample configuration of triangles in which the cells are randomly ordered is shown in figure 12. The corresponding form of the matrix $[\mathbf{A}]^n$ is shown in figure 13; in this figure a circle represents the nonzero entries.

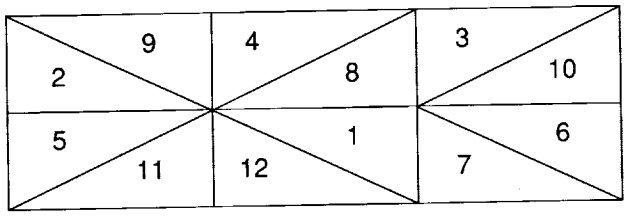


Figure 12. Sample cell configuration.

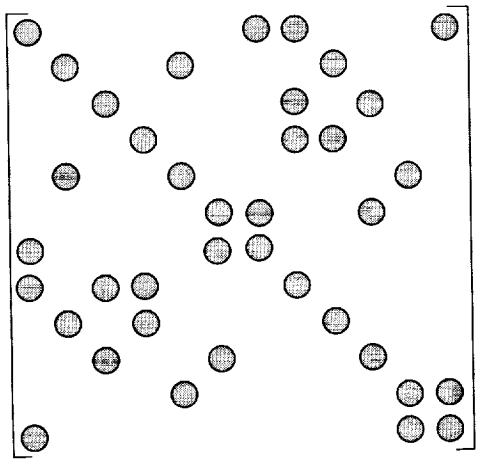


Figure 13. Form of matrix for cells in figure 12.

Although the solution of the system of equations may be obtained through a direct inversion of $[\mathbf{A}]^n$, as previously mentioned, the need for large memory can be circumvented through the use of a variety of relaxation schemes. In these schemes, the solution of equation (32) is obtained through a sequence of iterations in which an approximation of $\Delta\mathbf{Q}$ is continually refined.

To facilitate the derivation of these schemes, $[\mathbf{A}]^n$ is first written as a linear combination of three matrices representing the diagonal, subdiagonal, and superdiagonal terms, that is

$$[\mathbf{A}]^n = [\mathbf{D}]^n + [\mathbf{M}]^n + [\mathbf{N}]^n \tag{34}$$

The simplest iterative scheme for obtaining a solution to the linear system of equations is a Jacobi-type method in which all the off-diagonal terms of $[\mathbf{A}]^n\{\Delta\mathbf{Q}\}$ (i.e., $[\mathbf{M}]^n\{\Delta\mathbf{Q}\} + [\mathbf{N}]^n\{\Delta\mathbf{Q}\}$) are taken to the right-hand side of equation (32) and are evaluated using the values of $\{\Delta\mathbf{Q}\}^i$ from the previous subiteration level i. This scheme can be represented as

$$\begin{aligned}[\mathbf{D}]^n\{\Delta\mathbf{Q}\}^{i+1} &= \left[\{\mathbf{R}\}^n - [\mathbf{M}+\mathbf{N}]^n\{\Delta\mathbf{Q}\}^i\right] \\ &= \left[\{\mathbf{R}\}^n - [\mathbf{O}]^n\{\Delta\mathbf{Q}\}^i\right]\end{aligned} \tag{35}$$

The disadvantage of this scheme is that the sequence of Jacobi iterations may converge slowly. To accelerate the convergence, a Gauss-Seidel procedure may be employed in which values of $\{\Delta\mathbf{Q}\}$ are used on the right-hand side of equation (35) as soon as they are available. An example of this scheme can be written as

$$[\mathbf{D}]\{\Delta\mathbf{Q}\}^{i+1} = \left[\{\mathbf{R}\}^n - [\mathbf{M}]^n\{\Delta\mathbf{Q}\}^{i+1} - [\mathbf{N}]^n\{\Delta\mathbf{Q}\}^i\right] \tag{36}$$

where the latest values of $\{\Delta\mathbf{Q}\}$ from the subdiagonal terms are immediately used on the right-hand side of the iteration equation. A slight modification to this algorithm in which the latest values of $\{\Delta\mathbf{Q}\}$ from the superdiagonal terms are used results in a similar scheme that is given by

$$[\mathbf{D}]\{\Delta\mathbf{Q}\}^{i+1} = \left[\{\mathbf{R}\}^n - [\mathbf{M}]^n\{\Delta\mathbf{Q}\}^i - [\mathbf{N}]^n\{\Delta\mathbf{Q}\}^{i+1}\right] \tag{37}$$

Another variation of this algorithm can be obtained by alternating the use of equation (36) with equation (37) so that a symmetric Gauss-Seidel-type procedure is obtained.

Note that the algorithms given by equations (36) and (37) can both be implemented by sweeping sequentially through each mesh cell and simply using the latest values of $\{\Delta\mathbf{Q}\}$ for all the off-diagonal terms that have been taken to the right-hand side. This procedure can be represented as

$$[\mathbf{D}]\{\Delta\mathbf{Q}\}^{i+1} = \left[\{\mathbf{R}\}^n - [\mathbf{O}]^n\{\Delta\mathbf{Q}\}^{\substack{i+1\\ i}}\right] \tag{38}$$

where $\mathbf{Q}^{\substack{i+1\\ i}}$ is the most recent value of $\mathbf{Q}$; this term will be at the subiteration level $i+1$ for the cells that

have been previously updated and at the level i for the cells that remain to be updated. The distinction between the algorithms in equations (36) and (37) comes from sweeping forward through the cells (eq. (36)) or backward through the cells (eq. (37)).

Two disadvantages of this scheme exist. The first disadvantage is that this process is not easily vectorized because the solution of each point must be obtained before proceeding to the next point. The second disadvantage is that although the off-diagonal terms may be updated and immediately used on the right-hand side, the solution of the next unknown may or may not depend on previously determined quantities. For example, as can be seen from figure 12, when solving for the second unknown using equation (36), the updated value of the solution at point 1 is not used; therefore, the solution for point 2 remains a Jacobi step.

Note that for structured grids in which the cells are ordered in a natural manner (e.g., left to right and top to bottom), the latest information will immediately be used for the calculation of the next unknown. This occurs because the ordering of the cells produces a banded matrix with terms grouped along the diagonal. The fact that the latest-obtained data are not necessarily used for updating information in unstructured grids is because of the random ordering of the cells.

An improvement to the scheme just described, therefore, can be obtained by simply renumbering the cells to group terms along the diagonal of the matrix. In this manner, the solution of each point will tend to ensure that previously updated information from the surrounding cells is used as soon as it is available. An example of this is shown in figure 14 where the same cells used in figure 12 are simply renumbered from bottom to top and left to right. The resulting form of the matrix, shown in figure 15, shows that the grouping along the diagonal is greatly improved. The ordering of the cells in this way should result in faster convergence of the linear problem than a random ordering of cells. Although the ordering of the cells in this example groups unknowns along the diagonal, other procedures such as the Cuthill-McKee method described in reference 21 are more effective for general configurations. Again, note that several variations of this scheme can be obtained by using various combinations of equations (36) and (37). An important disadvantage of this scheme, however, is that the contribution of the off-diagonal terms to the right-hand side of equation (38) still cannot be vectorized.

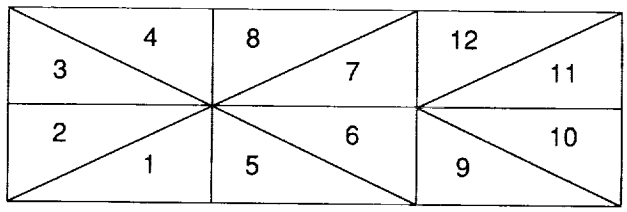


Figure 14. Sample cells.

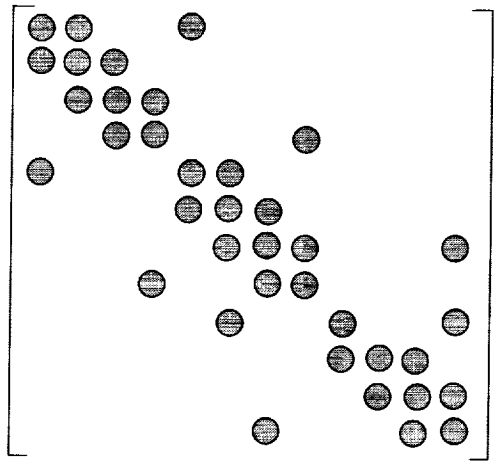


Figure 15. Form of matrix for cells in figure 14.

The Jacobi, Gauss-Seidel, and symmetric Gauss-Seidel schemes just described have all been used in practice by various researchers. An example of research that used these schemes to solve the Euler equations for transonic flow over a circular arc in a channel is given in reference 15. In this reference, the symmetric Gauss-Seidel scheme was shown to exhibit the fastest convergence rate of the three schemes. The successful use of a symmetric Gauss-Seidel algorithm for transonic flow over airfoils is described in reference 22; in this work, grouping the unknowns along the diagonal is enhanced by sorting them according to the x-coordinate direction.

Vectorization of Gauss-Seidel. The numbering of cells used in the current study is shown in figure 16. The ordering is obtained by grouping cells so that no two cells in a given group share a common edge. The resulting matrix form for $[\mathbf{A}]$ is given in figure 17. Note that for this example, only two groups are formed; in practice, at most, four groups will be formed for two-dimensional calculations and five groups are formed for three-dimensional calculations. The first group for the present example consists of the cells numbered 1 through 6, and the second group contains cells numbered 7 through 12.

The solution scheme, which can be written as before using equation (38), is implemented by solving

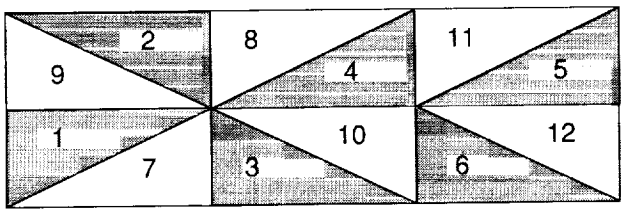


Figure 16. Sample cells.

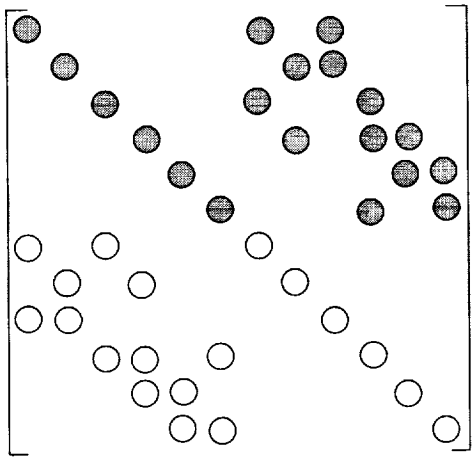


Figure 17. Form of matrix for cells in figure 16.

for all the unknowns in a group at a time. The cells in the first group are solved using a Jacobi-type iteration, and the cells in all the subsequent groups are obtained by using the most recently updated values of $\{\Delta\mathbf{Q}\}$ from the off-diagonal contributions. In this way, a Gauss-Seidel-type scheme is obtained which is easily implemented and is fully vectorized. Note that a symmetric Gauss-Seidel-type procedure is not necessary and is not used; stability is achieved as long as the matrix $[\mathbf{A}]$ maintains block diagonal dominance that occurs when first-order differencing is used on the implicit side of the equation (ref. 23).

In these discussions, the exact number of subiterations required to sufficiently converge the linear problem (eq. (32)) has not been specified. The number of subiterations used for each global time step has been determined through numerical experiments that are presented in the results.

Time Step Calculation

To enhance the convergence to a steady state, local time stepping is used. The time step calculation for each cell is given by

$$\Delta t = \mathrm{CFL}\frac{L}{\sqrt{u^2+v^2}+a} \tag{39}$$

where L is a length scale for the cell. This length scale is defined as the area of the cell divided by the perimeter.

Results

Flow-field calculations for several demonstration cases are presented. The first case is for an NACA 0012 airfoil at a free-stream Mach number of 0.8 and an angle of attack of 1.25°. The grid has an outer boundary placed approximately 50 chord lengths away from the body; this grid consists of 3624 nodes, 7012 cells, and 10 636 faces. A near-field view of the grid is shown in figure 18.

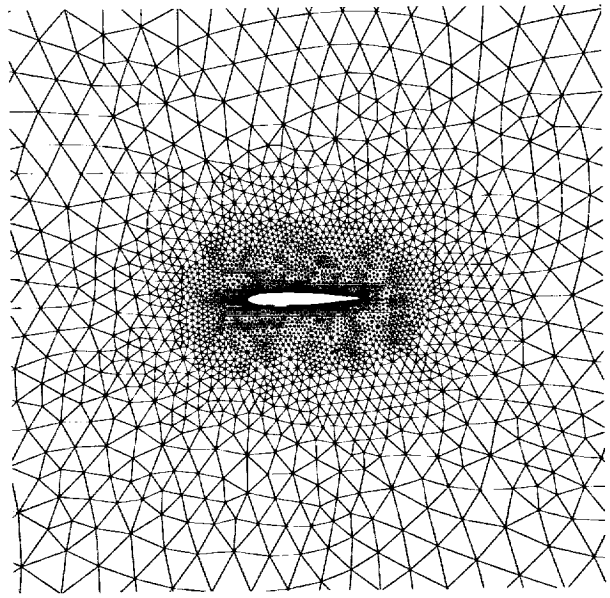


Figure 18. Near-field view of grid around NACA 0012 airfoil.

The pressure coefficient distribution along the airfoil surface is shown in figure 19. As seen, a moderately strong shock is captured on the upper surface of the airfoil, and a weaker shock is captured on the lower surface. Also, note that because a flux limiter has not been used for the present calculation, an "overshoot" is evident ahead of the upper-surface shock. The corresponding Mach number contours for this case are shown in figure 20.

For this calculation, the residual of the continuity equation has been reduced to "machine zero" in approximately 400 global iterations, as seen in figure 21. The CFL number began at 50 and was linearly ramped to 200 throughout 100 iterations. The CFL numbers used for the current calculation may not be optimal for the present case, but they give reasonably good convergence for a wide range

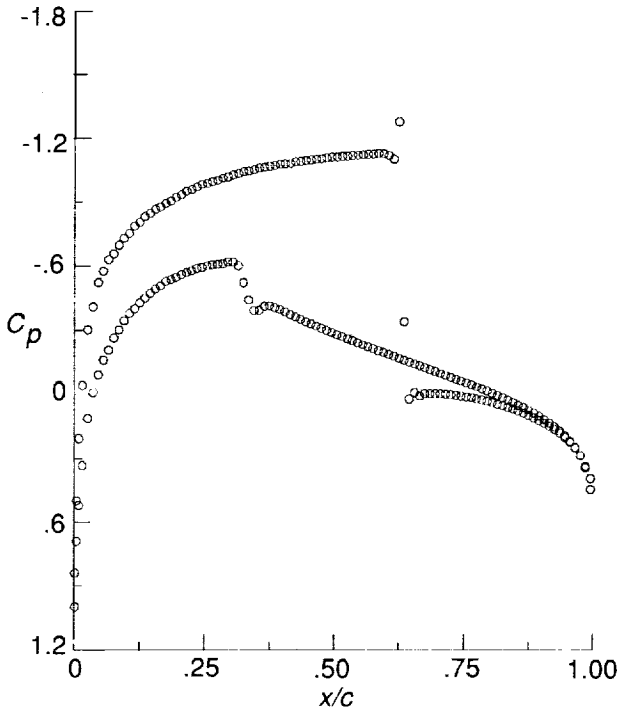


Figure 19. Pressure distribution for NACA 0012 airfoil. $M_\infty = 0.8; \alpha = 1.25°$.

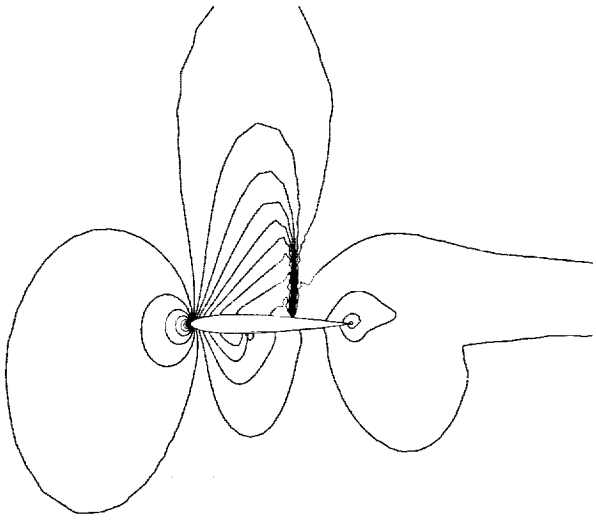


Figure 20. Mach number contours for NACA 0012 airfoil. $M_\infty = 0.8; \alpha = 1.25°$.

of test problems and grid densities. The memory required corresponds to approximately 180 words per cell. For each global iteration, 20 subiterations have been used to solve the linear system each time. This process results in a computational rate of approximately 60 μsec per cell per global time step on a CRAY YMP using a single processor. This computational rate, however, depends on the number of subiterations performed. For the current research, this number is based on results of a numerical study in which the number of subiterations has been varied for a wide variety of CFL numbers. A typical plot of the computer time required to obtain a four-order-of-magnitude reduction in the residual is shown in figure 22. This plot clearly indicates that 15 to 20 subiterations for each global iteration produce the fastest convergence rate. A similar study has been conducted on other grids and other cases with similar results. For this reason, between 15 to 20 sub- iterations are used for all cases shown in this report. Although this number of subiterations has proved adequate for the current work, further work in optimizing this parameter may prove beneficial.

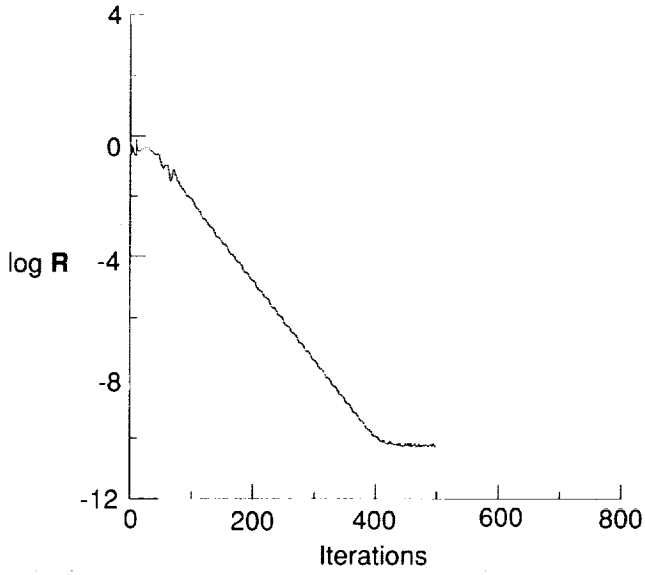


Figure 21. Convergence history for NACA 0012 airfoil. $M_\infty = 0.8; \alpha = 1.25°$.

A comparison of convergence rates obtained with both implicit and explicit boundary conditions on the airfoil surface is shown in figure 23. As seen, the use of implicit boundary conditions improves the rate of convergence through the first several orders of magnitude. In addition, the use of explicit boundary conditions impedes the convergence past approximately seven orders of magnitude. This behavior has been observed for a variety of other cases that are not presented. Note that the use of explicit boundary conditions seems to lead to a more robust code because ramping of the CFL number has not been necessary when explicit boundary conditions have been used. For most calculations in this study, implicit boundary conditions have been used after five iterations, and the CFL is ramped from 50 to 200.

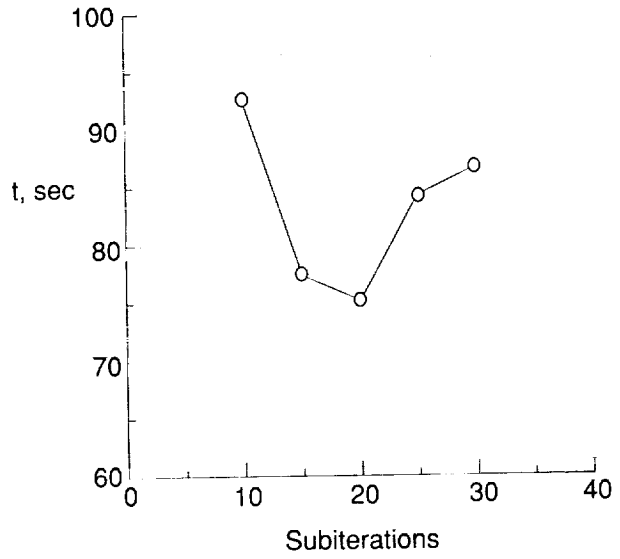


Figure 22. Computer time for four-order-of-magnitude reduction in residual for NACA 0012 airfoil. $M_\infty = 0.8$; $\alpha = 1.25°$.

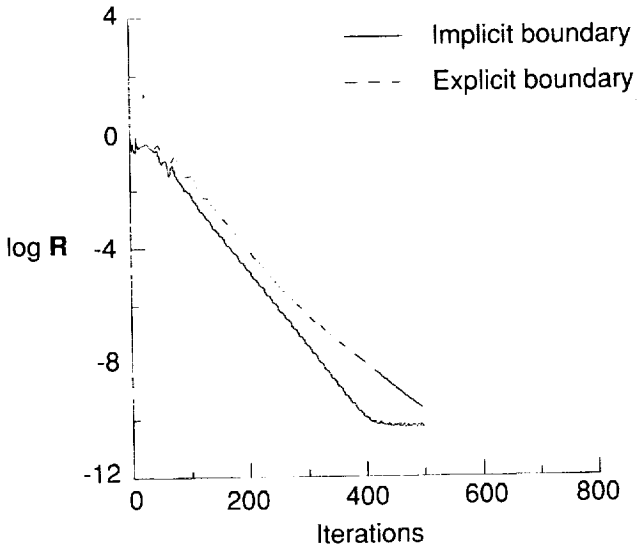


Figure 23. Comparison of convergence rates for implicit and explicit boundary conditions for NACA 0012 airfoil. $M_\infty = 0.8$; $\alpha = 1.25°$.

The next case presented is for a two-element airfoil in which an exact incompressible solution exists (ref. 24). The initial grid used for this calculation is shown in figure 24; this grid consists of 1556 points, 2882 cells, and 4439 faces. The main element and flap have 100 points each along the surface. Note that for this calculation, no clustering of cells has been performed near the surfaces because the final solution is obtained through an adaptation procedure described in reference 25. The pressure distribution calculated at a free-stream Mach number of 0.2 using this initial grid is shown in figure 25. As seen, the coarse grid yields results that agree poorly with the exact solution.

As previously mentioned, a solution also has been obtained by adapting the grid to the solution. Adaptation is achieved by first identifying a list of cells that require refinement. New points, which are located at the center of each of these cells, then are introduced into the existing triangulation using Bowyer's algorithm for the Delaunay triangulation, and the solution is interpolated to the new grid for use in restarting the solution. Because the present flow field does not contain discontinuities, the list of new cells is identified by flagging all the cells in which the undivided velocity gradient exceeds that of the average plus the standard deviation of all the cells in the grid (refs. 25 and 26).

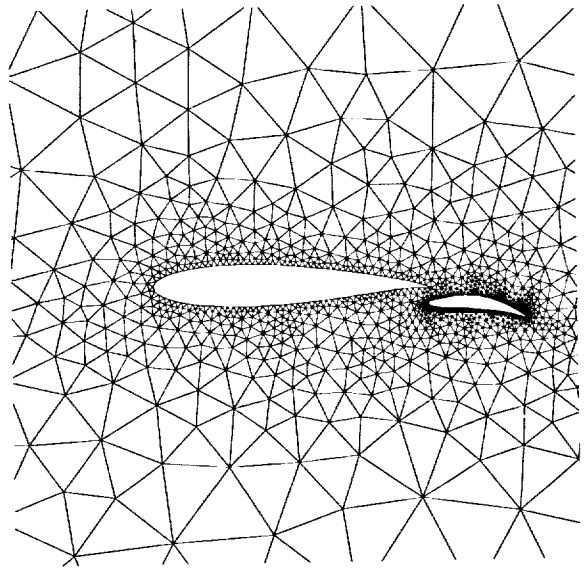


Figure 24. Grid around two-element configuration.

The final grid, shown in figure 26, consists of 3165 nodes, 6332 cells, and 9190 faces, with 148 nodes on the surface of the main element and 128 nodes on the flap. The pressure distribution obtained on this grid is shown in figure 27. The agreement with the exact solution is improved over that in figure 25. In addition, the calculated lift of 2.026 compares well with the exact value of 2.0281 given in reference 24.

The present algorithm also has been implemented in three dimensions with the preliminary results subsequently shown. The case shown is for a business jet at $M_\infty = 0.2$ and $\alpha = 3°$. The grid used for the computations has been generated using the advancing

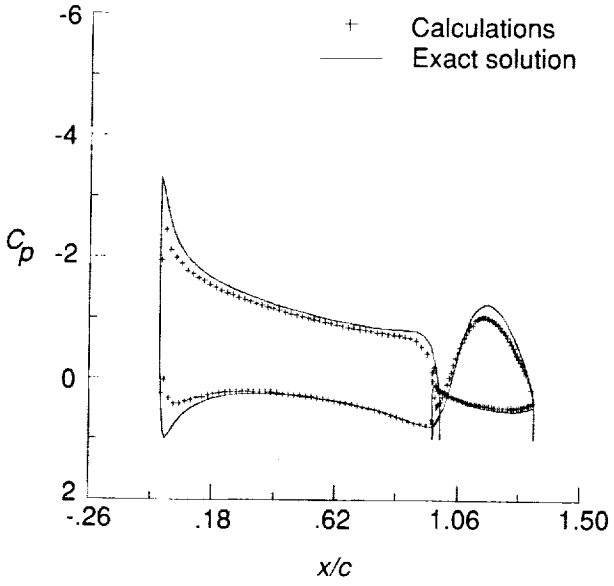


Figure 25. Pressure distribution over two-element configuration using initial grid.

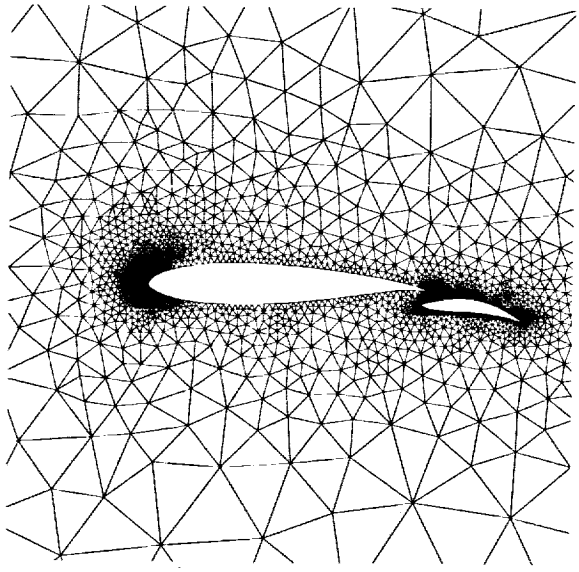


Figure 26. Grid obtained for two-element airfoil after adaptation.

front-type grid generation described in reference 27; this grid consists of 27 191 nodes, 144 100 cells, and 294 109 faces. The surface grid for this computation, which consists of 11 582 triangles, is shown in figure 28.

This case has been run at a constant CFL number of 300 with 15 subiterations; the convergence history is shown in figure 29. As seen, the residual is reduced between two and three orders of magnitude in 100 global iterations, at which point the convergence rate degrades. After 200 iterations, a residual reduction of slightly greater than three orders of magnitude is obtained. This "tailing off" behavior has not been observed in two dimensions when implicit boundary conditions are used, but it may be caused by the low free-stream Mach number or the close proximity of the outer boundary that extends approximately 10 body lengths ahead of and behind the airplane (but only about two body lengths above and below). Note that the "tailing off" of the residual may also indicate that the high-frequency errors in the scheme have been effectively reduced and that the low-frequency errors have begun to dominate. The use of a multigrid to rapidly eliminate these low-frequency errors should enhance the convergence.

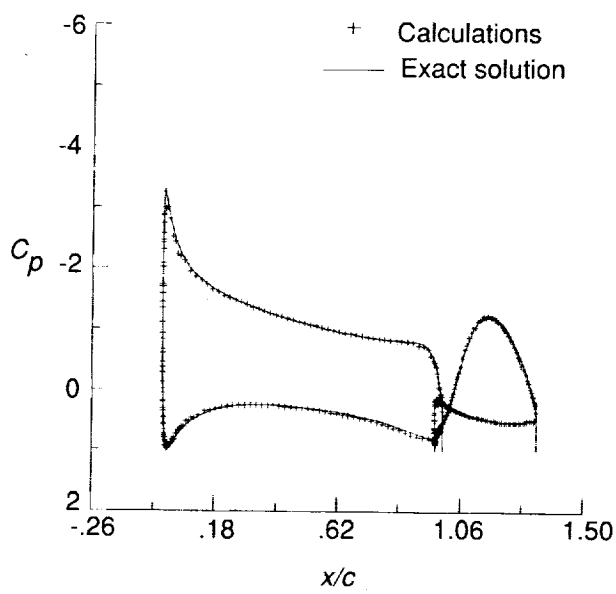


Figure 27. Solution obtained for two-element airfoil after adaptation.

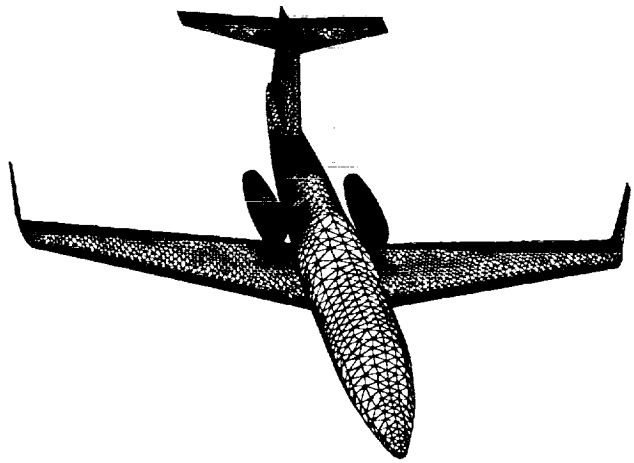


Figure 28. Surface grid for business jet.

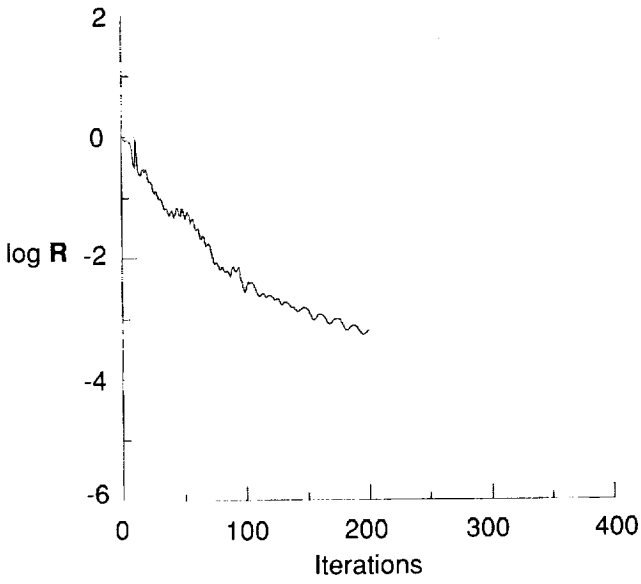


Figure 29. Convergence history for Lear jet. $M_\infty = 0.2$; $\alpha = 3.0°$.

A pressure distribution comparison at the $\eta = 0.44$ span station is made in figure 30 with the method described in reference 1. In figure 30, the results referred to as FUN3D are those of the present study and USM3D refers to those obtained using the computer code of reference 1. (This computer code is an upwind finite-volume code that uses multistage, time-stepping and flux-difference splitting.) As seen, the comparison between the two codes is reasonably close, and the main discrepancies occur at the leading edge. These differences are due to slight differences in the computation of the boundary fluxes and because the computations with USM3D use Roe's flux-difference splitting (ref. 12) instead of flux-vector splitting.

The current implementation of this code in three dimensions requires approximately 87 words of main memory per cell and about 50 words per face of a solid-state device (SSD). The computational rate on a CRAY YMP is approximately 140 μsec per cell per iteration, based on 15 subiterations per global time step. Note that this timing includes both user time and system time; without the use of SSD, the computational rate improves to approximately 92 μsec per cell per iteration because of a significant decrease in system time.

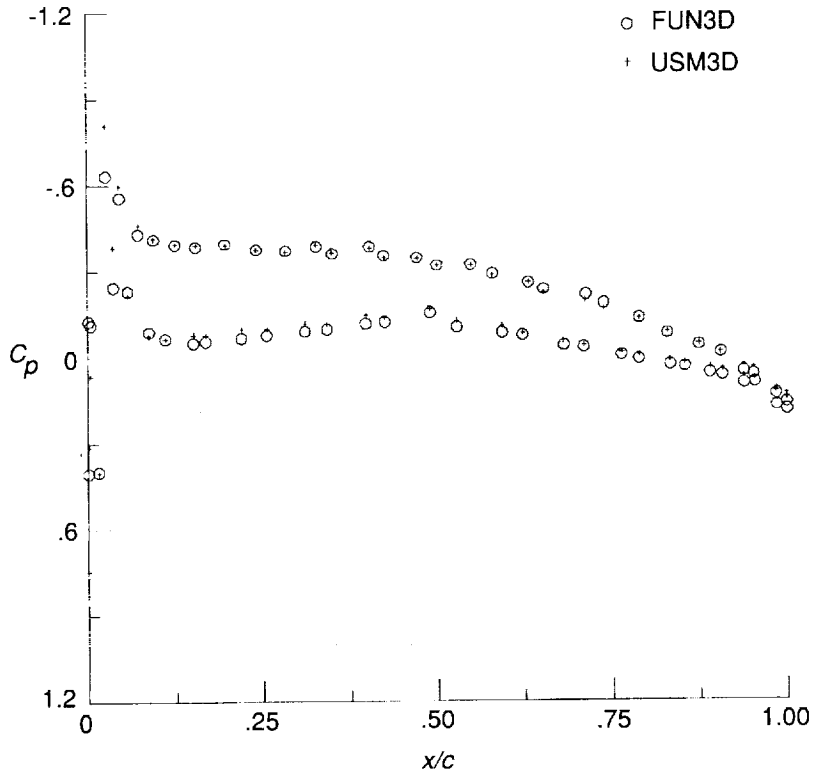


Figure 30. Comparison of surface pressure distribution. $M_\infty = 0.2$; $\alpha = 3.0°$; $\eta = 0.44$.

Concluding Remarks

A two-dimensional grid generation procedure has been devised which combines automatic point placement with Delaunay triangulation to efficiently produce good-quality unstructured meshes. This method, which uses the Delaunay triangulation algorithm of Bowyer, is based on the work of Holmes. The present algorithm improves the previously cited work by allowing the automatic generation of new mesh points so that the clustering of points near surfaces is achieved.

A flow solver that is implicit and can be completely vectorized is also developed in both two and three dimensions. This scheme is based on backward-Euler time differencing; the linear problem arising at each step is solved by using several iterations of a Gauss-Seidel-type procedure. In this method, the unknowns are divided into groups so that no cells in a given group share an edge; therefore, all the cells in a group are independent of each other so that their solutions can be obtained simultaneously.

The effect of the number of subiterations on the convergence rate (based on computer time) is also examined. Between 15 to 20 subiterations per global time step produce the best results. In addition, the use of implicit boundary conditions improves the convergence rate of the current algorithm.

Results are shown for a two-dimensional flow over an NACA 0012 airfoil and a two-element airfoil in which the solution is obtained with adaptation. For the two-element configuration, comparisons are made with the exact solution, and excellent results are obtained by adaptation. For three dimensions, the calculation of subsonic flow over a business jet is demonstrated.

NASA Langley Research Center
Hampton, VA 23665-5225
February 20, 1992

References

1. Frink, Neal T.; Parikh, Paresh; and Pirzadeh, Shahyar: A Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes. AIAA-91-0102, Jan. 1991.

2. Mavriplis, D. J.: Three Dimensional Unstructured Multigrid for the Euler Equations. *A Collection of Technical Papers AIAA 10th Computational Fluid Dynamics Conference,* June 1991, pp. 239 247. (Available as AIAA-91-1549-CP.)

3. Peraire, J.; Vahdati, M.; Morgan, K.; and Zienkiewicz, O. C.: Adaptive Remeshing for Compressible Flow Computations. *J. Comput. Phys.*, vol. 72, no. 2, Oct. 1987, pp. 449 466.

4. Löhner, Rainald; and Parikh, Paresh: Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method. AIAA-88-0515, Jan. 1988.

5. Parikh, Paresh; Löhner, Rainald; Gumbert, Clyde; and Pirzadeh, Shahyar: Numerical Solutions on a Pathfinder and Other Configurations Using Unstructured Grids and a Finite Element Solver. AIAA-89-0362, Jan. 1989.

6. Spragle, Gregory S.; McGrory, William R.; and Fang, Jiunn: Comparison of 2D Unstructured Grid Generation Techniques. AIAA-91-0726, Jan. 1991.

7. Bowyer, A.: Computing Dirichlet Tessellations. *Comput. J.*, vol. 24, no. 2, 1981, pp. 162–166.

8. Baker, Timothy J.: Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets. *A Collection of Technical Papers AIAA 8th Computational Fluid Dynamics Conference,* June 1987, pp. 255–270. (Available as AIAA-87-1124.)

9. Barth, Timothy J.; and Jespersen, Dennis C.: The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA-89-0366, Jan. 1989.

10. Vassberg, John C.; and Dailey, Kathleen B.: AIRPLANE: Experiences, Benchmarks & Improvements. *A Collection of Technical Papers, Part 1 AIAA 8th Applied Aerodynamics Conference,* American Inst. of Aeronautics and Astronautics, Aug. 1990, pp. 21 35. (Available as AIAA-90-2998-CP.)

11. Holmes, D. Graham; and Snyder, Derek D.: The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation. *Numerical Grid Generation in Computation Fluid Mechanics* '88, S. Sengupta, J. Häuser, P. R. Eiseman, and J. F. Thompson, eds., Pineridge Press, 1988, pp. 643 652.

12. Roe, P. L.: Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *J. Comput. Phys.*, vol. 43, no. 2, Oct. 1981, pp. 357 372.

13. Batina, John T.: Three-Dimensional Flux-Split Euler Schemes Involving Unstructured Dynamic Meshes. AIAA-90-1649, June 1990.

14. Rausch, Russ D.; Batina, John T.; and Yang, Henry T. Y.: Spatial Adaption Procedures on Unstructured Meshes for Accurate Unsteady Aerodynamic Flow Computation. *A Collection of Technical Papers, Part 3—AIAA/ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics, and Materials Conference,* American Inst. of Aeronautics and Astronautics, Apr. 1991, pp. 1904 1918. (Available as AIAA-91-1106-CP.)

15. Whitaker, D. L.; Slack, David C.; and Walters, Robert W.: Solution Algorithms for the Two-Dimensional Euler Equations on Unstructured Meshes. AIAA-90-0697, Jan. 1990.

16. Mavriplis, Dimitri; and Jameson, Antony: *Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes.* NASA CR-178346, ICASE Rep. No. 87-53, 1987.

17. Van Leer, Bram: Flux-Vector Splitting for the Euler Equations. *Eighth International Conference on Numerical Methods in Fluid Dynamics,* E. Krause, ed., *Volume 170 of Lecture Notes in Physics,* Springer-Verlag, 1982, pp. 507 512.

18. Frink, Neal T.: Upwind Scheme for Solving the Euler Equations on Unstructured Tetrahedral Meshes. *AIAA J.,* vol. 30, no. 1, Jan. 1992, pp. 70–77.

19. Janus, Jonathan Mark: The Development of a Three-Dimensional Split Flux Vector Euler Solver With Dynamic Grid Applications. M.S. Thesis, Mississippi State Univ., 1984.

20. Venkatakrishnan, V.; and Barth, Timothy J.: Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes. AIAA-89-0364, Jan. 1989.

21. Carey, Graham F.; and Oden, J. Tinsley: *Finite Elements: Computational Aspects, Volume III.* Prentice-Hall, Inc., c.1984.

22. Batina, John T.: Implicit Flux-Split Euler Schemes for Unsteady Aerodynamic Analysis Involving Unstructured Dynamic Meshes. AIAA-90-0936, Apr. 1990.

23. Mulder, William Alexander: Dynamics of Gas in a Rotating Galaxy. Ph.D. Thesis, Delft Univ. of Technology, June 1985.

24. Williams, B. R.: *An Exact Test Case for the Plane Potential Flow About Two Adjacent Lifting Aerofoils.* R. & M. No. 3717, British Aeronautical Research Council, 1973.

25. Warren, Gary P.; Anderson, W. Kyle; Thomas, James L.; and Krist, Sherrie L.: Grid Convergence for Adaptive Methods. AIAA-91-1592, June 1991.

26. Kallinderis, Yannis G.; and Baron, Judson R.: Adaptation Methods for a New Navier-Stokes Algorithm. *AIAA J.,* vol. 27, no. 1, Jan. 1989, pp. 37–43.

27. Parikh, Paresh; Pirzadeh, Shahyar; and Löhner, Rainald: *A Package for 3-D Unstructured Grid Generation, Finite-Element Flow Solution and Flow Field Visualization.* NASA CR-182090, 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1992	Technical Memorandum

4. TITLE AND SUBTITLE
Grid Generation and Flow Solution Method for Euler Equations on Unstructured Grids

5. FUNDING NUMBERS
WU 505-59-53-01

6. AUTHOR(S)
W. Kyle Anderson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23665-5225

8. PERFORMING ORGANIZATION REPORT NUMBER
L-16986

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-4295

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified Unlimited

Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A grid generation and flow solution algorithm for the Euler equations on unstructured grids is presented. The grid generation scheme, which utilizes Delaunay triangulation, generates the field points for the mesh based on cell aspect ratios and allows clustering of grid points near solid surfaces. The flow solution method is an implicit algorithm in which the linear set of equations arising at each time step is solved using a Gauss-Seidel procedure that is completely vectorizable. In addition, a study is conducted to examine the number of subiterations required for good convergence of the overall algorithm. Grid generation results are shown in two dimensions for an NACA 0012 airfoil as well as a two-element configuration. Flow solution results are shown for a two-dimensional flow over the NACA 0012 airfoil and for a two-element configuration in which the solution has been obtained through an adaptation procedure and compared with an exact solution. Preliminary three-dimensional results also are shown in which the subsonic flow over a business jet is computed.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Unstructured grids; Grid generation; Implicit schemes	18
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

Standard Form 298(Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

NASA-Langley, 1992