INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 19, 739–764 (1994)

GRID ADAPTATION USING A DISTRIBUTION OF SOURCES APPLIED TO INVISCID COMPRESSIBLE FLOW SIMULATIONS

N. P. WEATHERILL AND M. J. MARCHANT

Department of Civil Engineering, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, U.K.

O. HASSAN

Computational Dynamics Research, University of Wales, Swansea, Singleton Park, Swansea, SA2 8PP, U.K.

AND

D. L. MARCUM

NSF Engineering Research Center, Mississippi State University, U.S.A.

SUMMARY

Unstructured tetrahedral grids are generated using a new, very efficient procedure based upon the Delaunay triangulation. The generation procedure is extremely fast, having the capability to generate large grids in minutes on workstations. To maximize this computational performance, a new form of adaptivity has been developed involving the use of sources placed within regions of the domain which require further grid point resolution. A source has a position and a specified grid point density. An error indicator is used to find the elements within the grid which require refinement. Within such elements sources are placed with specified grid point densities which are proportional to the amount of refinement required. The grid generation procedure is then invoked and a grid generated whose grid point density is controlled by the sources. The resulting grid is thus refined in the regions identified by the error indicator as requiring greater resolution. The paper discusses the generation process and emphasizes the new solution adaptation capability. Several examples of the approach are given, including aerospace compressible flow simulations over realistic configurations.

KEY WORDS Grid generation Solution adaptation Inviscid compressible flow

1. INTRODUCTION

Two major advantages of unstructured grids over structured grids are their applicability to complex geometries and the ease with which grid adaptation can be incorporated.[1–4] A new algorithm[5,6] has been developed which, it is believed, has taken a major step towards reducing the grid generation time for realistic configurations, to the order of minutes on workstations of modest capabilities, and thus further strengthening the applicability of the unstructured grid approach. The performance achieved with this new method can be enhanced by the use of grid adaptation, the computational times for which have also been substantially reduced. This paper

CCC 0271-2091/94/210739-26

Received October 1993
Revised February 1994

will outline the new approach for grid generation and will give details of the approach taken to grid adaptation. Examples of the method will be given for the simulation of inviscid compressible flow over two- and three-dimensional aerospace geometries.

2. TETRAHEDRAL GRID GENERATION

The unstructured tetrahedral grid generation is based upon the Delaunay triangulation.[1,7–10] Details of the implementation of the new efficient approach have been given elsewhere.[5,6] Only the main features which are relevant to the adaptation procedure will be presented here.

2.1. Point connectivity

The Delaunay triangulation[11] is the criterion used to connect points to form tetrahedra. This geometrical construction is dual to the Voronoi diagram.[12] Each point is assigned to a territory which is closer to that point than to any other point. Such regions for each point are called Voronoi neighbourhoods, and if points with common boundaries of Voronoi neighbourhoods are connected, then the Delaunay triangulation is obtained. Each tetrahedron has associated with it a vertex of the Voronoi diagram, which is located at the circumcentre of the sphere through the four points which form the tetrahedron. No other point can lie within this sphere and as such this property is called the in-circle criterion. The algorithm used to generate this construction is based upon a modified Bowyer approach.[13]

2.2. Point creation

The Delaunay algorithm gives no indication as to how points should be generated. Hence a procedure must be constructed to perform this task. The point generation for arbitrary three-dimensional domains is achieved using the following algorithm.

Algorithm 1

1. Compute the point distribution function for each boundary point $\mathbf{r}_O = (x, y, z)$, i.e. for point o

$$\mathrm{d}p_O = \frac{1}{M}\sum_{i=1}^{M} |\mathbf{r}_i - \mathbf{r}_O|,$$

   where $|\ |$ is the Euclidean distance and it is assumed that point o is surrounded by M points, $i = 1, \ldots, M$.
2. Generate the Delaunay triangulation of the boundary points.
3. Initialize the number of interior field points created, $N = 0$.
4. For all tetrahedra within the domain:
   (a) Define a prospective point Q to be the centroid of the tetrahedron.
   (b) Derive the point distribution $\mathrm{d}p_Q$ for the point Q by interpolating the point distribution function from the nodes of the tetrahedron, $\mathrm{d}p_m$, $m = 1, \ldots, 4$.
   (c) Compute the distances d_m, $m = 1, \ldots, 4$, from the prospective point Q to each of the four points of the tetrahedron. If $\{d_m < \alpha \mathrm{d}p_Q\}$ for any $m = 1, \ldots, 4$, then reject the point; return to the beginning of step 4. Otherwise compute the distance $s_j, j = 1, \ldots, N$, from the prospective point Q to other points to be inserted, namely P_j, $j = 1, \ldots, N$. If $\{s_j < \beta\, \mathrm{d}p_Q\}$ for any $j = 1, \ldots, N$, then reject the point; return to the beginning of step 4. Otherwise accept the point Q for insertion by the Delaunay triangulation algorithm and include Q in the list P_j, $j = 1, \ldots, N$.

(d) Assign the interpolated value of the point distribution function dp_Q to the new node P_N.
(e) Next tetrahedra.

5. If $N = 0$, go to step 7.
6. Perform the Delaunay triangulation of the derived points P_j, $j = 1, \ldots, N$. Return to step 3.
7. Smooth the mesh.

To clarify this procedure, Figure 1 shows several steps of the above algorithm applied in two dimensions. Figure 1(a) shows a rectangular domain discretized by a set of points. The grid generated within the domain should reflect the initial boundary point spacing. Hence it is necessary to derive some characteristic lengths associated with the boundary points. In the initial phase the only length scales present are the lengths associated with each edge. Hence, to derive length scales at nodes, these edge lengths are transferred to nodes using the equation given in step 1. The boundary nodes are then connected together using the Delaunay triangulation. The resulting triangulation within the domain is shown in Figure 1(b). Given this triangulation, it is necessary to determine whether the shape of each triangle satisfies the length scale associated

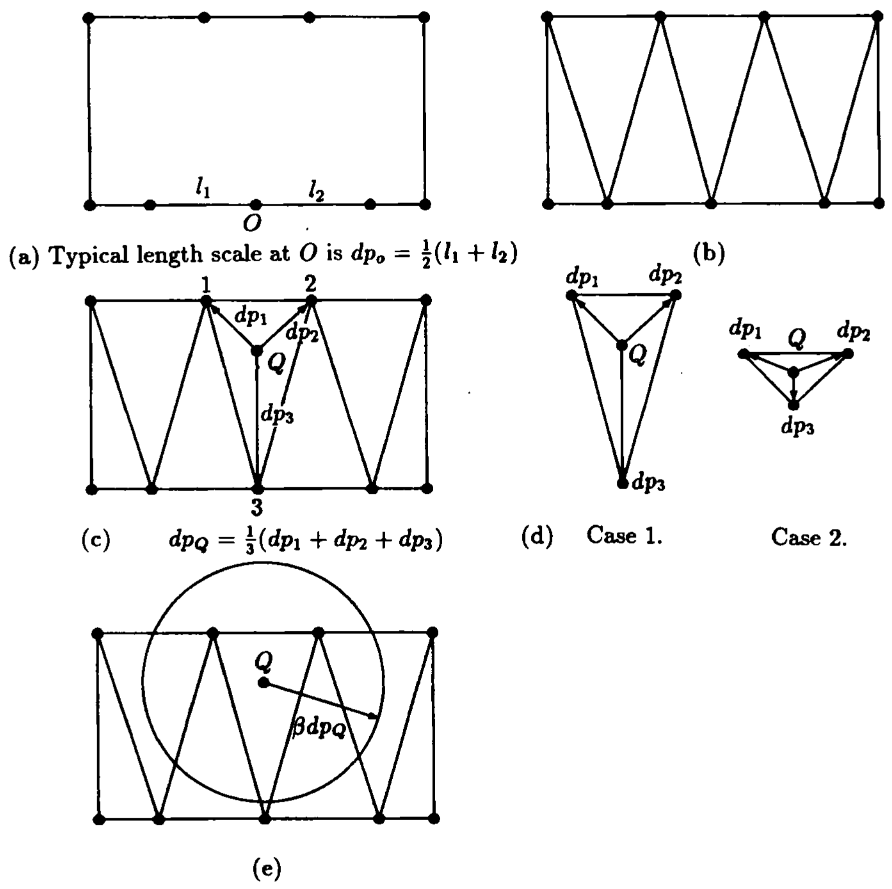


Figure 1. Explanation of the automatic point creation

with each of the nodes. This assessment is performed by determining whether the edge lengths of the formed triangles are consistent with the characteristic length scale of each of the nodes. This consistency check for a triangle is performed by creating a point at the centroid position and interpolating from the three vertices of the triangle the characteristic length associated with that position within the grid; see Figure 1(c). The length of each edge formed by connecting the centroid node to the three vertices is then computed. If any of these edge lengths is less than the interpolated characteristic length, then the point at the centroid is rejected. Figure 1(d) shows two cases for different triangle shapes. This strategy is implemented to ensure refinement of the triangles until the lengths of edges of triangles are consistent with the length scales within the mesh, as interpolated from the initial boundary nodes. The checks so far performed are very easy and computationally efficient to implement. However, the test as to whether a point is accepted is performed with respect to the edges which are assumed to be created between the centroid point and the vertices of the triangle. In practice, if the point is accepted, the Delaunay triangulation will be used to connect the point within the existing triangulation. If points are placed close to each other, then there will be many connections between inserted centroid points and the assumptions made about consistency of interpolated characteristic lengths and the lengths of edges will not be valid. Hence, once a centroid point has been created, an exclusion zone is imposed such that no other centroid point can be created within this region (Figure 1(e)) on the same cycle within Algorithm 1. The exclusion zone is proportional to the interpolated length scale at the centroid. After testing each triangle, the centroid points created are then inserted via the Delaunay triangulation. This process is repeated until no centroid points are created. The point insertion procedure has then converged, resulting in the final grid. A realistic application of the above approach is shown in Figure 2.

The coefficient α controls the grid point density by changing the allowable shape of formed tetrahedra, whilst β has an influence on the regularity of the triangulation by not allowing points within a specified distance of each other to be inserted in the same sweep of the tetrahedra within the field. It should be noted that this algorithm for point generation can be used for grid adaptation on surfaces, as will be described later.

It is clear from the description given that the grid point density distribution is controlled by step 4(b). Three different possibilities can be implemented within the algorithm. The one described

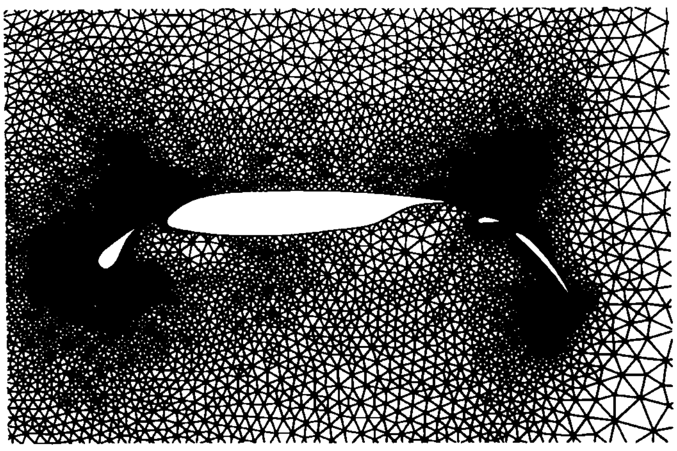


Figure 2. Example of the point creation procedure applied to a realistic geometry

in Algorithm 1 uses the boundary point distribution, details of which are given in Section 2.2.1. Two alternative methods are described in Sections 2.2.2 and 2.2.3.

2.2.1. Point distribution interpolated from the boundary nodes. The procedure outlined creates a distribution of points within the domain which is governed by the point distribution on the boundaries. This procedure can be thought of as equivalent to a boundary value problem and acts like the popular grid generation procedure for structured grids.[14]

In general, linear interpolation of the boundary point spacing is used, but when boundaries are close together and the spacing varies rapidly, it is effective to use a boundary weighting function of the form

$$\mathrm{d}p = 0{\cdot}25(\mathrm{d}p_{\mathrm{n1}} + \mathrm{d}p_{\mathrm{n2}} + \mathrm{d}p_{\mathrm{n3}} + \mathrm{d}p_{\mathrm{n4}}),$$

$$\mathrm{d}p_{\mathrm{a}} = \min\{\mathrm{d}p, \mathrm{d}p_i\}, \quad i = \mathrm{n1, n2, n3, n4},$$

$$\mathrm{d}p_{\mathrm{boundary}} = \mathrm{d}p - \omega(\mathrm{d}p - \mathrm{d}p_{\mathrm{a}}),$$

where n1, n2, n3 and n4 are the nodes of a tetrahedron with at least one node a boundary node and ω is a weighting parameter, typically 0·3.

To construct grids on surfaces, the generation is performed in the parametric co-ordinates of the surface. Figure 3 shows an application of this approach to the generation of a grid on the surface of a nacelle and pylon. The algorithm, working in the parametric surface co-ordinates, produces a grid of triangles whose sizes are governed by the boundary point spacing. It is clear that the procedure produces very regular grid points and triangles.

2.2.2. Point creation by the use of sources. In somewhat of an analogous way to point sources used as control functions with elliptic partial differential equations,[14] it is possible to define line and point sources to provide grid control for unstructured meshes. The local point spacing at position r can be defined as any appropriate function $\mathrm{d}p(r) = f(x, y, z)$. One convenient and flexible functional form is

$$\mathrm{d}p(r) = A_j \mathrm{e}^{B_j |R_j - r|}, \tag{1}$$

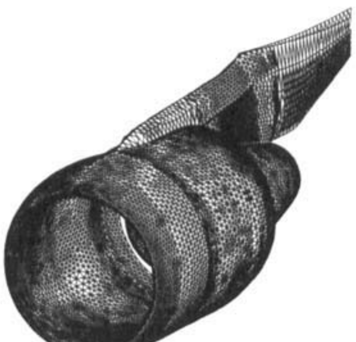


Figure 3. Grid on the surface of a nacelle generated using the Delaunay triangulation and the automatic point creation algorithm

where A_j and B_j are the user-specified amplification and decay parameters respectively of the source $j, j = 1, \ldots, M$, and R_j is the position of each point source. A_j can be interpreted as the required spacing at R_j, while B_j can be derived such that the spacing A_j decays at a specified rate away from R_j. Grid point creation is then performed as outlined in Algorithm 1, but in step 4(b) the appropriate point distribution function at the centroid of a tetrahedron is determined by equation (1).

This technique provides a mechanism for clustering points, but it can prove difficult to choose appropriate amplification and decay parameters so as to ensure adequate point clustering away from the sources. Hence a simple modification is to define the point spacing as

$$\mathrm{d}p(r) = \min\{\mathrm{d}p_{\text{boundary}}, A_j \mathrm{e}^{B_j|R_j - r|}\}. \tag{2}$$

In this case $\mathrm{d}p_{\text{boundary}}$ is the point distribution derived from the boundary spacing.

This approach can be readily extended to include line sources and planar sources. In general, the implementation of the point clustering due to sources, as described here, requires a search over the number of sources to determine the appropriate value of the point distribution function. For a small number of sources this is not a significant workload. However, an alternative technique, which is more computationally efficient for large numbers of sources, is to firstly generate a grid from the boundary point distribution ignoring any sources in the field. Then the tetrahedral elements are found which contain each of the sources. The amplification factor of each source is used to modify the point-spacing function at the nodes of the tetrahedral element. For example, if the amplification factor was set to 0·5, this could be interpreted to imply that the point spacings at the nodes of the elements which contain the sources are decreased by this factor. If all nodes of elements which contain the sources are appropriately modified, then on returning to the point insertion algorithm, some elements will no longer satisfy the spacing parameter and additional points must be added. The rate of decay of the influence of the point sources can be incorporated by moving out from the element which contains a source and appropriately modifying the nodes of the neighbour elements. This then provides the desired point clustering. Examples, which for clarity are presented for two-dimensional domains, of the use of the source approach are shown in Figure 4.

2.2.3. Point creation by the use of a background mesh. This option is included for completeness, since it is clear that the point creation algorithm readily admits this procedure. The method is now well known and is widely used.[2–4] It requires, in the first instance, the user to specify a mesh which covers the domain to be gridded. At each of the mesh points the grid point density is defined through interpolation from the background mesh and this information can then be passed via step 4(b) of Algorithm 1 through to the point creation algorithm.

2.3. *Boundary integrity*

Following the automatic point creation and connections by the Delaunay algorithm, the resulting triangulation is made boundary-conforming by a method which utilizes transformations on tetrahedra.[5,6] Edges of missing boundary faces are recovered by finding the tetrahedra which are intersected by the missing edge and then creating or modifying these tetrahedra to recover the missing edge. A similar approach is used to recover boundary faces.

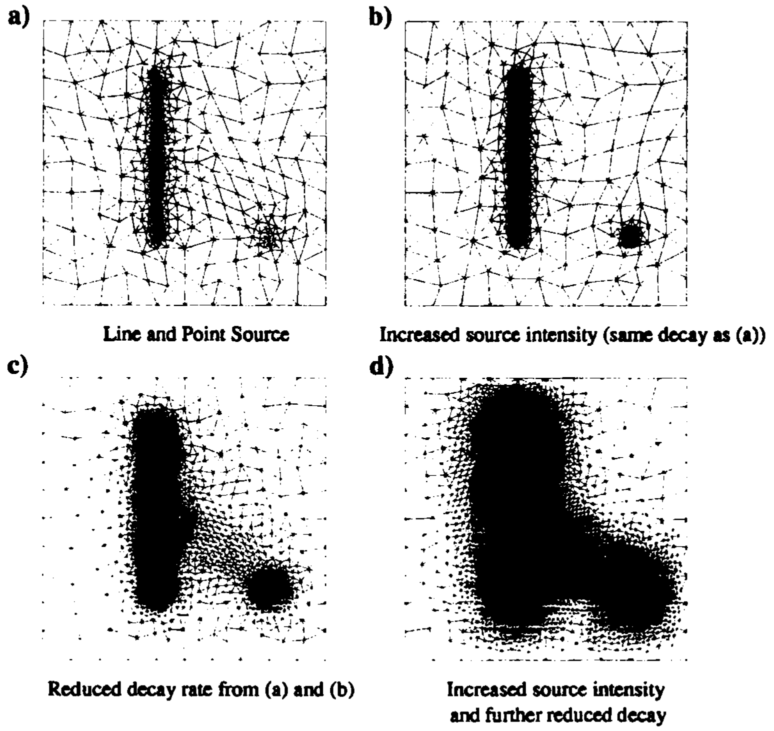

a)
Line and Point Source
b)
Increased source intensity (same decay as (a))
c)
Reduced decay rate from (a) and (b)
d)
Increased source intensity
and further reduced decay

Figure 4. Effect of amplification and decay parameters of sources

2.4. *Efficiency*

Table I lists some computational times, as measured on a variety of computers, for the generation of 10^6 tetrahedra. The results demonstrate that realistic grids for engineering computations can be readily obtained on workstations.

Table I. Computational times for the generation of a grid with 10^6 elements. Times include I/O, consistency checks and post processing for the flow solvers

Computer	CPU time (min)
CRAY YMP	4·96
IBM Risc 6000 550	12·28
SGI PI	28·84
SGI Indigo Elan	25·28
SGI Indigo XS 24/4000	14·75
SGI Crimson	14·36
SUN Sparc II	26·98

3. GRID ADAPTATION

The fundamental idea behind adaptation is to modify the grid or solution algorithm to better resolve the features in the flow field. Here only techniques which modify the grid will be considered.

Grid adaptation techniques fall into the categories of (i) h-refinement whereby points are added or deleted, (ii) r-refinement where the number of nodes remains fixed but the nodes are relocated to better resolve the important features and (iii) remeshing methods where, given information on a grid, a completely new grid is derived from that information. All techniques are driven by an error or solution criterion.

Here we propose a new method which uses the automatic point creation outlined in Algorithm 1 and the ideas outlined for point clustering using sources. The new approach is a combination of h-refinement and remeshing and recovers both these procedures for given input parameters. The technique is equally applicable for steady and transient adaptation. The main steps are as follows.

Algorithm 2

1. Generate the initial mesh.
2. Obtain a flow solution.
3. Derive sources (a) on the surface and (b) in the field.
4. Generate the adapted surface grid.
5. Generate the adapted field grid.
6. Return to step 2.

Once a flow solution has been obtained, the sources are derived by detecting regions in the domain where solution or error activity is high.

3.1. Error indication and source creation

For the present solution-adaptive grid generation procedure an error indicator is required that detects and locates appropriate features in the flow field. Since the grid is usually de-refined to the original grid, the indicator should determine where all the appropriate features are located. In order to provide flexibility in isolating varying features, multiple error indicators are used. Each can isolate a particular type of feature. The error indicators are set to the negative and positive components of the gradient in the direction of the velocity vector, as given by

$$e_1 = -\min[V \cdot \nabla(\phi), 0], \qquad e_2 = \max[V \cdot \nabla(\phi), 0],$$

and the magnitude of all the gradients in all directions normal to the velocity vector, as given by

$$e_3 = \frac{|\nabla\phi - V(V \cdot \nabla\phi)|}{|V \cdot V|},$$

where V is the velocity vector and ϕ is any suitable flow property. Typically density is used as the basis for the error indicator. The first two error indicators represent expansions and compressions in the flow direction and the third represents gradients normal to the flow direction. The indicators can be scaled by the relative element size. Length scaling can improve the detection of weak features on a coarse grid with the present procedure. Each error indicator is treated independently, allowing particular features in the flow field to be isolated. For each error indicator an error is determined from

$$e_{\text{lim}} = e_{\text{m}} + c_{\text{lim}} e_{\text{s}},$$

where e_{lim} is the error limit, e_{m} is the mean of the error indicator, e_{s} is the standard deviation of the error indicator and c_{lim} is a constant. Typically a value near unity is used for the constant.

The error indicators are converted into grid generation sources which, as indicated, locally reduce the relative element size during grid generation. The grid generation sources are created at grid points where an error indicator is greater than the corresponding error limit E_{tol}. The strength of the source is proportional to the normalized magnitude of the error indicator and is scaled within a maximum and minimum strength, $\mathrm{d}p_{\text{max}}$ and $\mathrm{d}p_{\text{min}}$ respectively. Hence for any element with $e_{\text{lim}} > E_{\text{tol}}$ the source strength is determined from

$$\mathrm{d}p_{\text{source}} = \mathrm{d}p_{\text{max}} + (\mathrm{d}p_{\text{min}} - \mathrm{d}p_{\text{max}}) \frac{E_{\text{e}} - E_{\text{min}}}{|E_{\text{max}} - E_{\text{min}}|},$$

where E_{max} and E_{min} are the maximum and minimum computed error values respectively throughout the field and $\mathrm{d}p_{\text{max}}$ and $\mathrm{d}p_{\text{min}}$ are the maximum and minimum normalized point-spacing parameters. When the maximum error occurs, the minimum spacing is required. For example, a value of $\mathrm{d}p_{\text{min}} = 0{\cdot}5$ will result in an enriched mesh at the position of a source which has a point spacing half that of the initial mesh. In regions where the error indicator lies in the range $E_{\text{lim}} < E_{\text{tol}}$, no sources are created and the mesh is unaltered.

3.2. *Surface adaptation*

Grid adaptation on the configuration surface is perfomed as follows.

Algorithm 3

1. Input the previous surface mesh.
2. Input the surface sources.
   (i) Determine the surface triangles which contain the sources.
   (ii) Insert a point at the position of each source and connect to form triangles using a local Delaunay algorithm.
   (iii) Modify the values of the point distribution function at the nodes which form the elements which contain the sources.
3. Perform the automatic point creation routine to generate additional points, connecting the points with a local Delaunay algorithm.

The mechanics of the surface grid adaptation are as outlined in Algorithm 1, except that the point connection is performed by a direct connection between the new point and the three points which form the triangle which contains it, followed by several implementations of a 'local Delaunay in-circle criterion' diagonal-swapping routine. This latter approach is used since a two-dimensional Delaunay algorithm is not applicable on a three-dimensional surface.

In the surface adaptation procedure it is necessary to ensure that the added points are placed on the geometrical surface of the configuration. The method adopted here is to reconstruct the surface geometry using a transfinite, visually continuous, triangular interpolant.[15] It is viewed that this approach is more efficient and applicable than returning to the geometrical definition of the surface and will be of major benefit when adaptation is applied to transient, moving

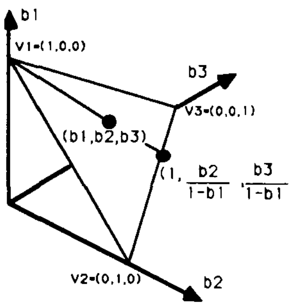


Figure 5. Schematic of the surface triangle

boundary problems. The interpolant utilizes outward surface normals, unlike such methods as the Ferguson patch which uses partial derivatives on boundaries. The resulting reconstructed surface is a G^1 representation in that the surface has a continuously varying outward normal vector.

Consider a surface segment which is a vector-valued function represented by

$$F(b_1, b_2, b_3) = \begin{cases} x(b_1, b_2, b_3) \\ y(b_1, b_2, b_3) \\ z(b_1, b_2, b_3) \end{cases} \tag{3}$$

with $(b_1, b_2, b_3) \in T$, where the domain T is defined as

$$T = \{(b_1, b_2, b_3)\colon 0 \leq b_i \leq 1,\ i = 1, 2, 3;\ b_1 + b_2 + b_3 = 1\} \tag{4}$$

and the vertices of T are denoted by

$$V_1 = (1, 0, 0), \qquad V_2 = (0, 1, 0), \qquad V_3 = (0, 0, 1). \tag{5}$$

Figure 5 shows such a triangle in schematic form. The edges are defined by

$$e_1 = \{(0, b_2, b_3) \in T\}, \qquad e_2 = \{(b_1, 0, b_3) \in T\}, \qquad e_3 = \{(b_1, b_2, 0) \in T\}, \tag{6}$$

as shown in Figure 6. It is convenient to introduce the set of indices

$$I = \{(1, 2, 3), (2, 3, 1), (3, 1, 2)\}.$$

The univariate cubic Hermite operator along edges (Figure 7) produces a cubic interpolant from end points V_0 and V_1 and tangent vectors V'_0 and V'_1 and can be written as

$$H[V_0, V_1, V'_0, V'_1](t) = h_0(t)V_0 + h_1(t)V_1 + H_0(t)V'_0 + H_1(t)V'_1, \tag{7}$$

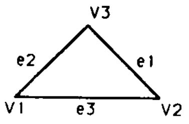


Figure 6. Edge vertex notation

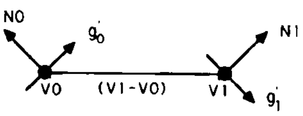


Figure 7. Schematic of the edge of a triangular patch

where the blending functions are

$$\begin{aligned} h_0(t) &= 1 - 3t^2 + 2t^3, & h_1(t) &= 3t^2 - 2t^3, \\ H_0(t) &= t - 2t^2 + t^3, & H_1(t) &= t^3 - t^2, \end{aligned} \tag{8}$$

$0 \leqslant t \leqslant 1$, and a prime denotes differentiation with respect to the parametric variable t.

For the present applications we need an operator of the form

$$g[V_0, V_1, N_0, N_1] = g(t) \tag{9}$$

with the properties

$$g(0) = V_0, \qquad g(1) = V_1, \qquad \langle g'(0), N_0 \rangle = 0, \qquad \langle g'(1), N_1 \rangle = 0, \tag{10}$$

where $\langle\ \rangle$ denotes a scalar product.

However, V'_0 and V'_1 are tangent vectors which are not uniquely determined by the requirements of (10) and are unknown in our procedure. We require that $g'(0)$ be in the plane containing N_0 and $V_1 - V_0$ and that $g'(1)$ be in the plane of N_1 and $V_1 - V_0$. Hence it is necessary to compute end tangents from the normal vectors in the following way:

$$\begin{aligned} g'(0) &= a[(V_1 - V_0) - (N_0, V_1 - V_0)N_0] = aN_0 \times (V_1 - V_0) \times N_0, \\ g'(1) &= b[(V_1 - V_0) - (N_1, V_1 - V_0)N_1] = bN_1 \times (V_1 - V_0) \times N_1. \end{aligned} \tag{11}$$

Given these conditions, we can define the Hermite interpolant

$$\begin{aligned} g[V_0, V_1, N_0, N_1](t) = g(t) = h_0(t)V_0 + h_1(t)V_1 + \alpha H_0(t) \frac{N_0 \times (V_1 - V_0) \times N_0}{|N_0 \times (V_1 - V_0) \times N_0|} \\ + \beta H_1(t) \frac{N_1 \times (V_1 - V_0) \times N_1}{|N_1 \times (V_1 - V_0) \times N_1|}, \end{aligned} \tag{12}$$

where α and β are arbitrary non-zero constants which can be used much like tension parameters to affect the shape of $g(t)$. For all cases considered these have been taken to be unity.

Given these data, the transfinite interpolant can be defined. It is based upon the ideas of the side vertex method for the interpolant defined by g. The arguments of g are a vertex and a point on the opposite facing edge where a ray emanating from the vertex and passing through the point (b_1, b_2, b_3) intersects. These boundary points can be expressed as

$$c = \frac{b_j V_j + b_k V_k}{1 - b_i}. \tag{13}$$

The interpolant is given by

$$G_i[F](b_1, b_2, b_3) = g[F(V_i), F(c), N[F](V_i), N[F](c)](1 - b_i), \tag{14}$$

where $N[F](V_i)$ is the outward surface normal vector at the vertex V_i, which leads to

$$G[F] = W_1 G_1[F] + W_2 G_2[F] + W_3 G_3[F] \tag{15}$$

with

$$W_i = \frac{b_j^2 b_k^2}{b_1^2 b_2^2 + b_2^2 b_3^2 + b_3^2 b_1^2}. \tag{16}$$

Nielson[15] proves that (14)–(16) satisfy the interpolation conditions

$$G(F)(\partial T) = F(\partial T), \qquad N[G(F)](\partial T) = N[F](\partial T),$$

where ∂T refers to the boundaries of the triangle t. The interpolation defined by (15) represents a transfinite continuous patch. In the case of triangular interpolants for function data a common approach to discretization is to take the function values along an edge as cubic Hermite interpolants and the cross-boundary normal derivatives as linear interpolants. In this way the boundary data and consequently the entire triangular approximation depends only upon the nine values at the vertices, namely the value of the function and its two first-order partial derivatives at each of the vertices of the triangular domain. In a similar manner we now wish to obtain a six-parameter, G^1 interpolant. This will depend solely upon the position and outward surface normal at the three vertices of T.

Given the transfinite formulation, the discretized form for each of the three radial projectors is given by

$$g_i(b_1, b_2, b_3) = g\left\{F_i, g[F_j, F_k, N_j, N_k]\left(\frac{b_j V_j + b_k V_k}{1 - b_i}\right), N_i, N_i\left(\frac{b_j V_j + b_k V_k}{1 - b_i}\right)\right\}(1 - b_i). \tag{17}$$

Hence the interpolant involves three vertex-to-side interpolations. This approach ensures that the added points are consistent with a G^1 representation of the configuration surface.

The accuracy of the technique has been examined for an unstructured surface grid representing part of the surface of a sphere with unit radius. The initial surface triangulation is shown in Figure 8(a) and consists of just 21 grid points connected by 28 triangles. The surface has been reflected along one side to enhance the definition of the surface shape on the 'far' horizon of the part sphere.

The adaptation of the surface was performed by specifying that the required point spacing for each grid point not on the boundary or reflection plane be reduced by a factor of 30. Adapting the surface by placing points at a position derived as the averaged co-ordinates of the nodes of a triangle into which the point is inserted is the crudest form of surface definition and is used as the benchmark. The surface grid generated in this way is shown in Figure 8(b) and the average error of the additional 7199 points is 2·54%, with a maximum error of 4·64%. This compares with an average error for the points placed using the triangular interpolant, or so-called G^1 patch, of 0·494%, with a maximum value of 1·09%. The surface generated using the G^1 patch is shown in Figure 8(c). An analysis of the errors obtained for each sweep of point addition during the Delaunay triangulation procedure is presented in Figures 9(a) and 9(b), which show the average and maximum displacement errors for the number of points inserted on each sweep of the Delaunay triangulation procedure respectively.

It is apparent that the use of the G^1 patch to calculate the position of points being inserted reduces the displacement error by a factor in excess of four for both the average and maximum displacement values. A study of Figure 8 clearly shows the benefit of using the G^1 patch when

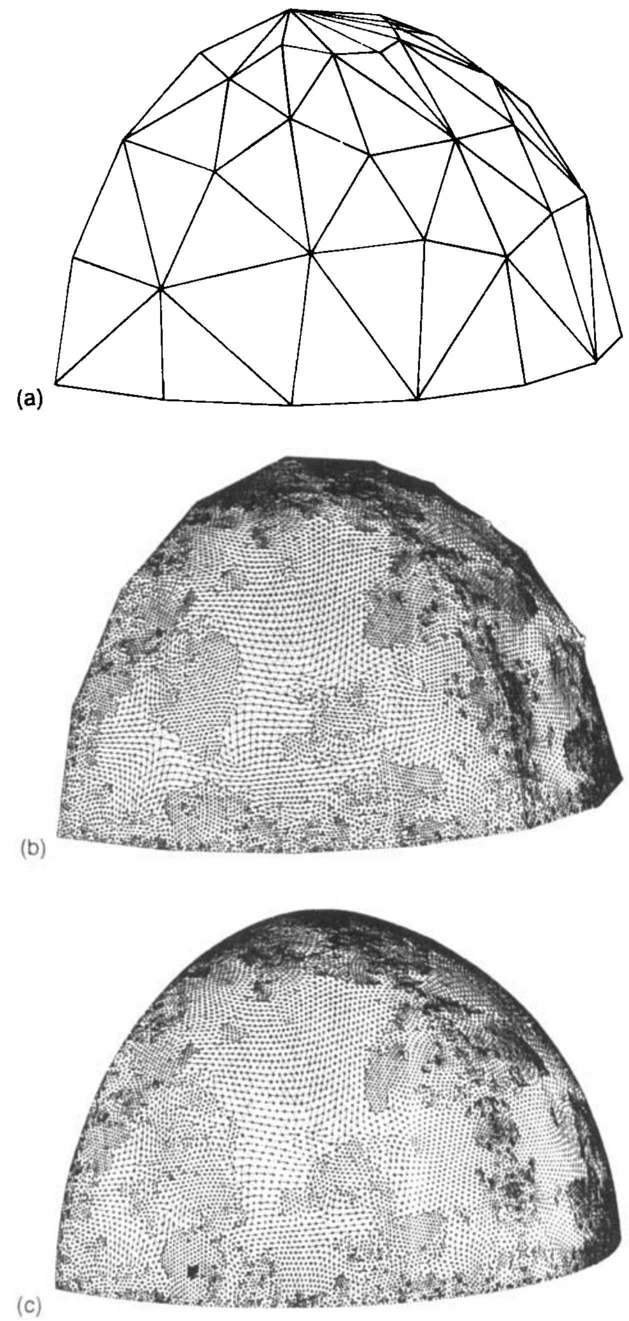


Figure 8. Surface reconstruction: (a) initial surface triangulation; (b) surface triangulation using linear interpolation for point position; (c) surface triangulation using G^1 interpolation for point position

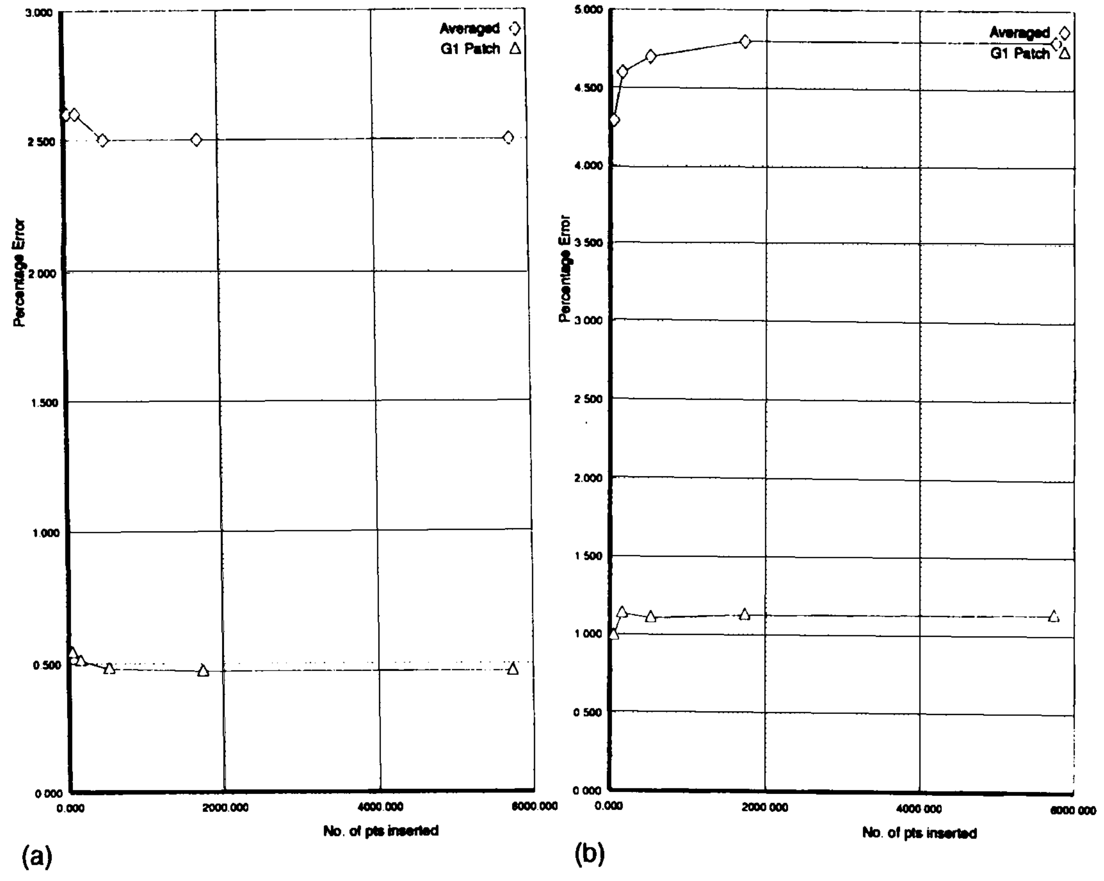


Figure 9. Error plots for the surface reconstruction: (a) average displacement error for each sweep of point addition; (b) maximum displacement error for each sweep of point addition

adapting a surface grid, since the circular shape of the far horizon of the sphere has been greatly improved and the peaks apparent in the surface derived from linear interpolation of the co-ordinates do not appear in the surface generated using the G^1 patch.

3.3. Field adaptation

Grid adaptation in the field is performed as follows.

Algorithm 4

1. Generate a mesh from the non-adapted surface mesh (if appropriate, a different concentration factor α can be used) or input the previous mesh.

2. Input the additional surface points which are included in the adapted surface grid and connect with the Delaunay algorithm.
3. Input the field sources.
   (i) Determine the elements which contain the sources.
   (ii) Insert a point at the position of the source and connect with the Delaunay algorithm.
   (iii) Modify the values of the point distribution function at the nodes in the element.
4. Perform the automatic point creation to generate the adapted field mesh. A different concentration factor from that used to generate the initial grid can be used in this step of the Delaunay grid generation. To distinguish this, the concentration factor in this step will be denoted by α_a.

Steps 1 and 2 are straightforward to apply. Step 3 requires a searching process to find the elements which contain the sources. This type of search is similar to the one used in the Delaunay algorithm to find all spheres which contain a point. Hence in the implementation of step 3(i) the Delaunay algorithm search routine is used with the addition of a routine to determine the element rather than the sphere which contains the source. The important issue in the search is that the tree data structure, which is essential for an efficient implementation of the Delaunay algorithm, is used.

If the parameter α_a is small, typically in the range 0·8–1·4, then points will in general be added by the automatic point creation procedure until the point distribution satisfies that which was specified with the sources. If, however, α_a is large, say of the order of 10^3, then after the insertion of a point corresponding to the position of the source the automatic point creation procedure will not add points. In this way, with the appropriate values of α_a, the proposed adaptation procedure degenerates to standard h-refinement. This was also the case for the surface grid as considered in Section 3.2. It is clear, therefore, that the method proposed generalizes h-refinement so that an arbitrary number of points can be added. Furthermore, since in the adaptive cycle α can be modified from its initial grid value in step 1 of Algorithm 4, it is possible to regenerate a mesh prior to the inclusion of sources, so that once features in the flow field have been detected and sources defined, the initial mesh can be coarsened. Hence, the proposed method has a remeshing capability to ensure that with successive adaptation the number of grid points does not always increase. As with remeshing, the proposed procedure can result in a final adapted mesh having fewer points than the initial mesh.

4. FLOW ALGORITHM

4.1. Governing equations

The Euler equations for the time-dependent, three-dimensional, compressible, inviscid, non-heat-conducting flow of a simple system in thermodynamic equilibrium in the absence of body forces can be expressed in conservation form as

$$\frac{\partial U}{\partial t} + \frac{\partial F^i}{\partial x^i} = 0, \quad i = 1, 2, 3, \tag{18}$$

where U is the solution vector and F^i is the advection flux vector in the i-direction. The solution and advective flux vectors are given respectively by

$$U = \begin{bmatrix} \rho \\ \rho V^1 \\ \rho V^2 \\ \rho V^3 \\ \rho E \end{bmatrix}, \qquad F^i = \begin{bmatrix} \rho V^i \\ \rho V^1 V^i + \delta^{1i} p \\ \rho V^2 V^i + \delta^{2i} p \\ \rho V^3 V^i + \delta^{3i} p \\ (\rho E + p) V^i \end{bmatrix}, \tag{19}$$

where ρ is the density, V^i is the velocity component in the i-direction, E is the total energy and p is the static pressure.

For the present investigation a thermally and calorically perfect gas is assumed and the equation of state for pressure is given by

$$p = (\gamma - 1)[\rho E - \tfrac{1}{2}\rho(V^{1^2} + V^{2^2} + V^{3^2})], \tag{20}$$

where γ is the specific heat ratio, which is taken for this work to be 1·4.

4.2. *Numerical solution procedure*

The governing equations (18) are solved using an explicit multistage Runge–Kutta scheme.[1,6] The governing equations are discretized in space using a Galerkin weighted residual approximation with the solution domain subdivided into tetrahedral finite elements. Time discretization is achieved using an explicit multistage Runge–Kutta procedure. A full description of the method of characteristics used to implement the boundary conditions for the Euler solver is given in detail elsewhere.[16]

5. RESULTS

5.1. *NACA 0012 test case*

Since the method proposed contains some new aspects, it is important to investigate how well the procedure performs when compared with more established techniques. Hence the first example is that for the inviscid flow over an NACA 0012 aerofoil at a freestream Mach number of 0·75 and an angle of incidence of 2° computed using a fine structured grid, adapted structured and adapted unstructured grids with standard point enrichment and the new proposed method utilizing the source approach. The relevant grids and flow field contours are shown in Figures 10(a)–10(d). The number of points used in each computation and the computed lift are given in Table II.

The flow field contours clearly demonstrate the benefits of grid adaptation, even over a fine grid, in resolving the shock wave. The conventionally refined grids required four levels of adaptation, but the grid using the sources was achieved using only two applications of the sources. The number of points used in the adaptation with sources is less than those used by the other techniques, yet the results compare well.

5.2. *Engine inlet*

The second example to be presented is a three-dimensional Mach 3 inviscid flow within an engine inlet. Figure 11(a) shows a schematic of the geometry. The surface grid was generated using a two-dimensional Delaunay triangulation algorithm which has been applied to each of the planar faces of the inlet. Figures 11(b) and 11(c) show two views of the inlet. In each case

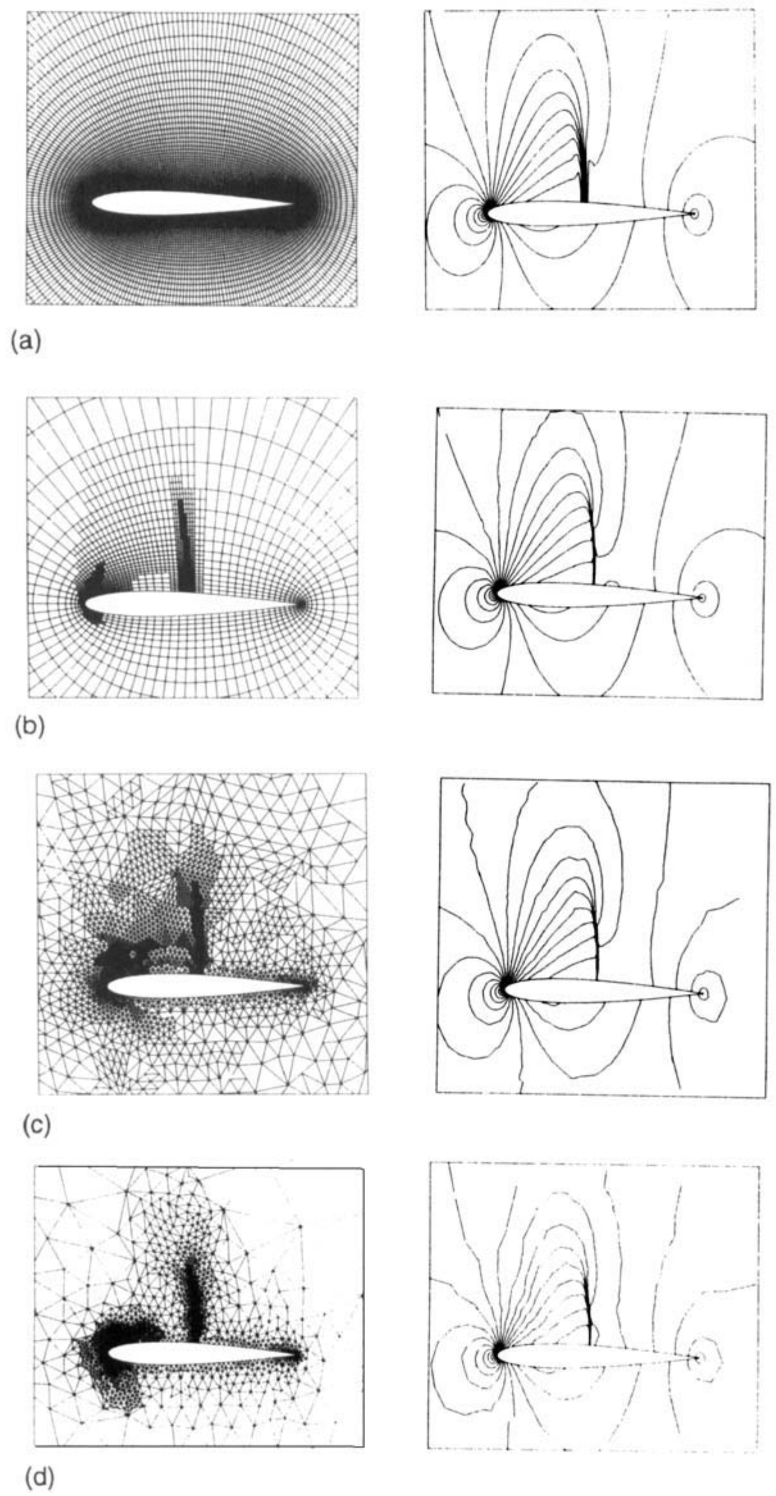


Figure 10. Grids (left) and flow contours (right) for a NACA 0012 aerofoil: (a) fine structured grid; (b) structured grid with h-refinement; (c) unstructured grid with h-refinement; (d) unstructured grid with refinement by sources

Table II. Details of the grids and computed lift coefficient for an NACA 0012 aerofoil at freestream Mach number of 0·75 and angle of incidence of 2°

Grid	Points	Lift coefficient
Fine structured	16640	0·4269
Adapted structured	8274	0·4334
Adapted unstructured	5206	0·4296
Adapted unstructured (sources)	3676	0·4298

the initial, first adapted and second adapted surface grids and contours of pressure are presented. The number of points used in each grid is given in Table III.

It is clear from the flow field contours that the adaptation enhances the resolution of the shock waves.

5.3. *Wing/fuselage/pylon/nacelle configuration*

The third example is the application of the proposed method to the transonic flow over a wing/fuselage/pylon/nacelle configuration. Two non-adapted solutions have been obtained. The first is for a mesh containing 592,380 tetrahedra, a solution which can be used to gauge the accuracy of the other results. The second is for a mesh containing 337,696 tetrahedra and thus allowed sufficient scope for adaptation to be applied and its benefits to be assessed. Two adaptations were performed on the smaller initial mesh, resulting in a final second adapted mesh containing a similar number of points/tetrahedra as the first larger non-adapted mesh. Two views of the surface meshes and contours of pressure are shown for the initial mesh of 337,696 tetrahedra and the second adapted mesh in Figure 12(a). The increased resolution of the shock

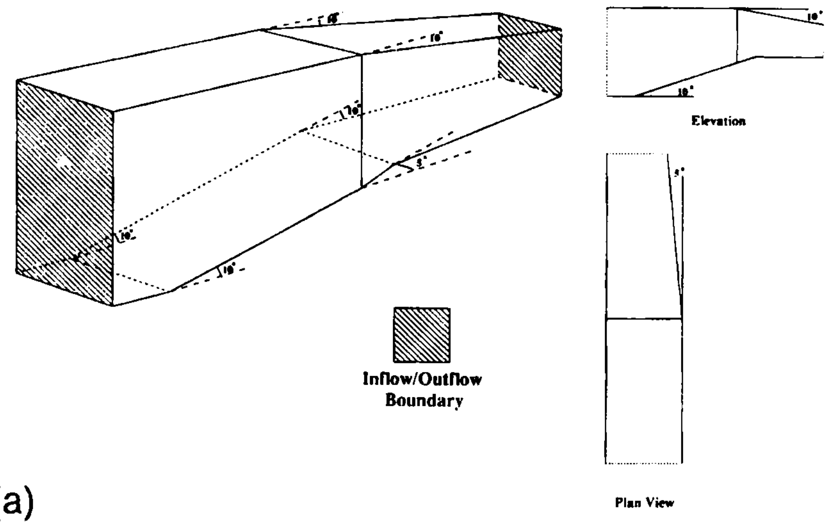
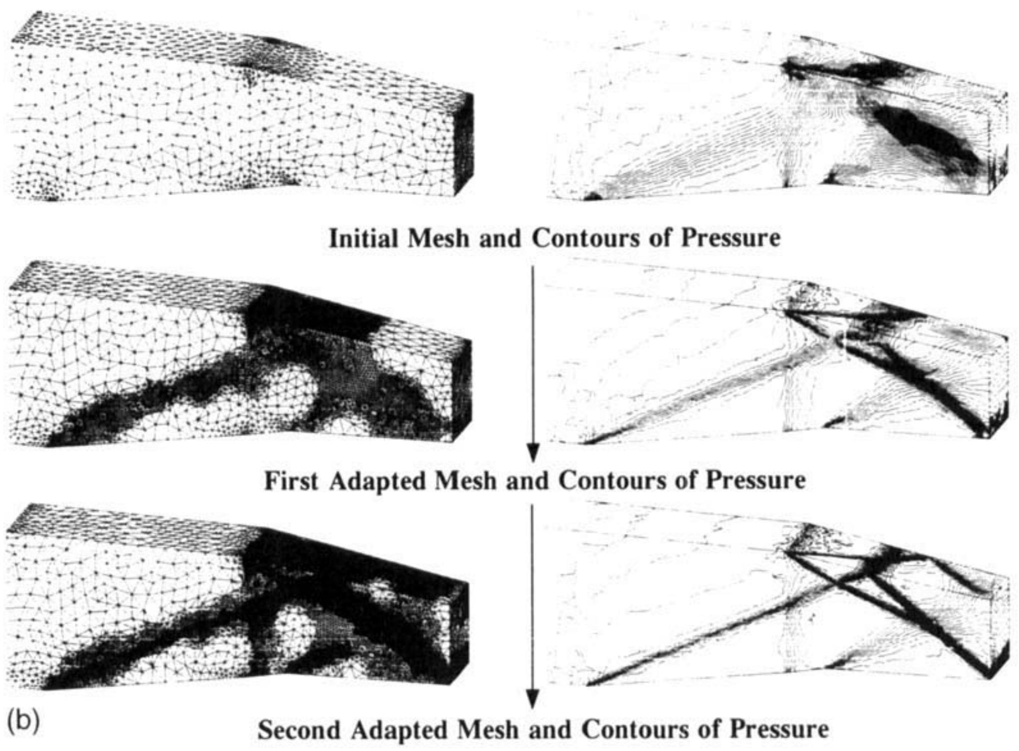
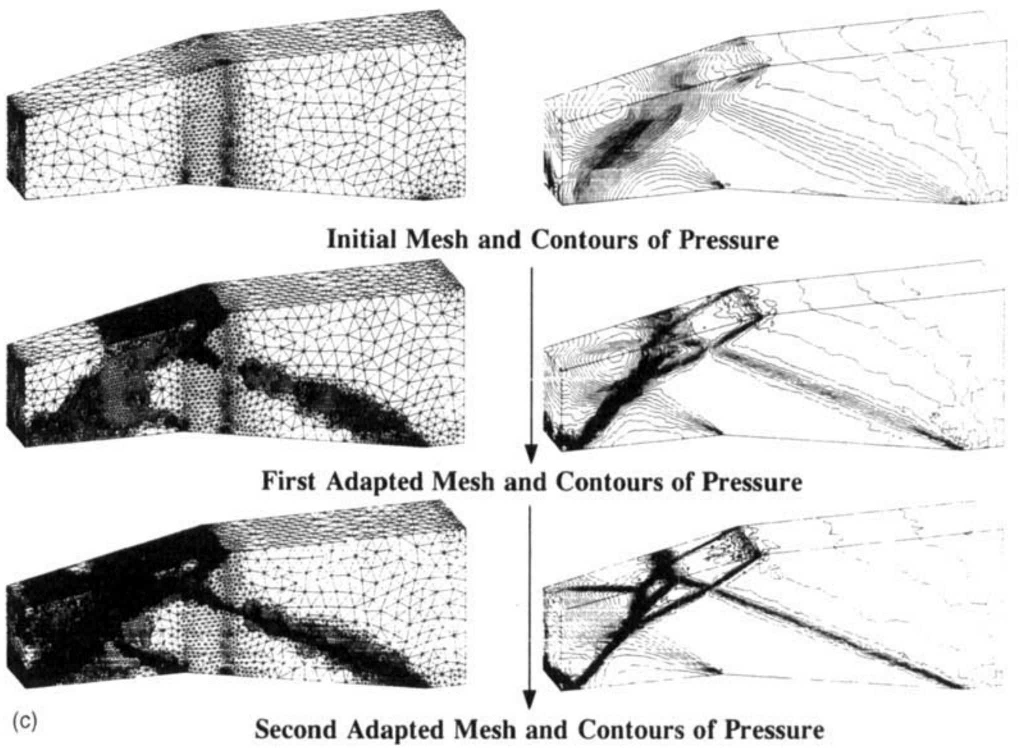


Figure 11. Supersonic engine inlet simulation: (a) engine intake geometry; (b, c) initial, first adapted and second adapted surface meshes and contours of pressure


Initial Mesh and Contours of Pressure
First Adapted Mesh and Contours of Pressure
Second Adapted Mesh and Contours of Pressure

(b)


Initial Mesh and Contours of Pressure
First Adapted Mesh and Contours of Pressure
Second Adapted Mesh and Contours of Pressure

(c)

Figure 11. *(continued)*

Table III. Details of the adapted grids for the engine inlet

	Triangles	Points
Initial surface mesh	5486	2745
First adapted surface mesh	22936	11465
Second adapted surface mesh	46124	23064
	Tetrahedra	**Points**
Initial field mesh	39328	7566
First adapted field mesh	85881	19771
Second adapted field mesh	221257	50488

Table IV. Details of the adapted grids for the wing/body/pylon/nacelle configuration

	Triangles	Points
Initial surface mesh	32102	15960
First adapted surface mesh	43884	21874
Second surface mesh	64164	31958
	Tetrahedra	**Points**
Initial mesh	337696	60605
First adapted mesh	545214	95406
Second adapted mesh	768267	135030

waves on both the upper surface of the wing and on the nacelle is apparent. Details of the mesh statistics for the adaptation procedure are given in Table IV.

Figure 12(b) shows the stations along the wing at which the results obtained have been compared with experimental data, while Figure 12(c) shows the stations for the nacelle. Comparisons of the pressure coefficient obtained by experiment with those from the initial unadapted mesh and the second adapted mesh are presented in Figure 12(d). The plots show that the second adapted mesh solution is a great improvement over the initial solution, with a marked increase in the resolution of the shock wave on the upper surface.

Further comparisons with experimental data are possible on the engine nacelle and Figure 12(e) presents plots of the pressure coefficient.

The results presented thus far have compared the solutions at various stages of the adaptation procedure. An additional comparison can now be given for the second adapted mesh and the unadapted mesh consisting of 592,380 tetrahedra. Comparisons of the pressure coefficient at the four stations on the wing detailed in Figure 12(b) are shown in Figure 12(f). Again there appears to be a closer agreement between the adapted solution and the experimental data, particularly in the resolution of the shock wave on the upper surface.

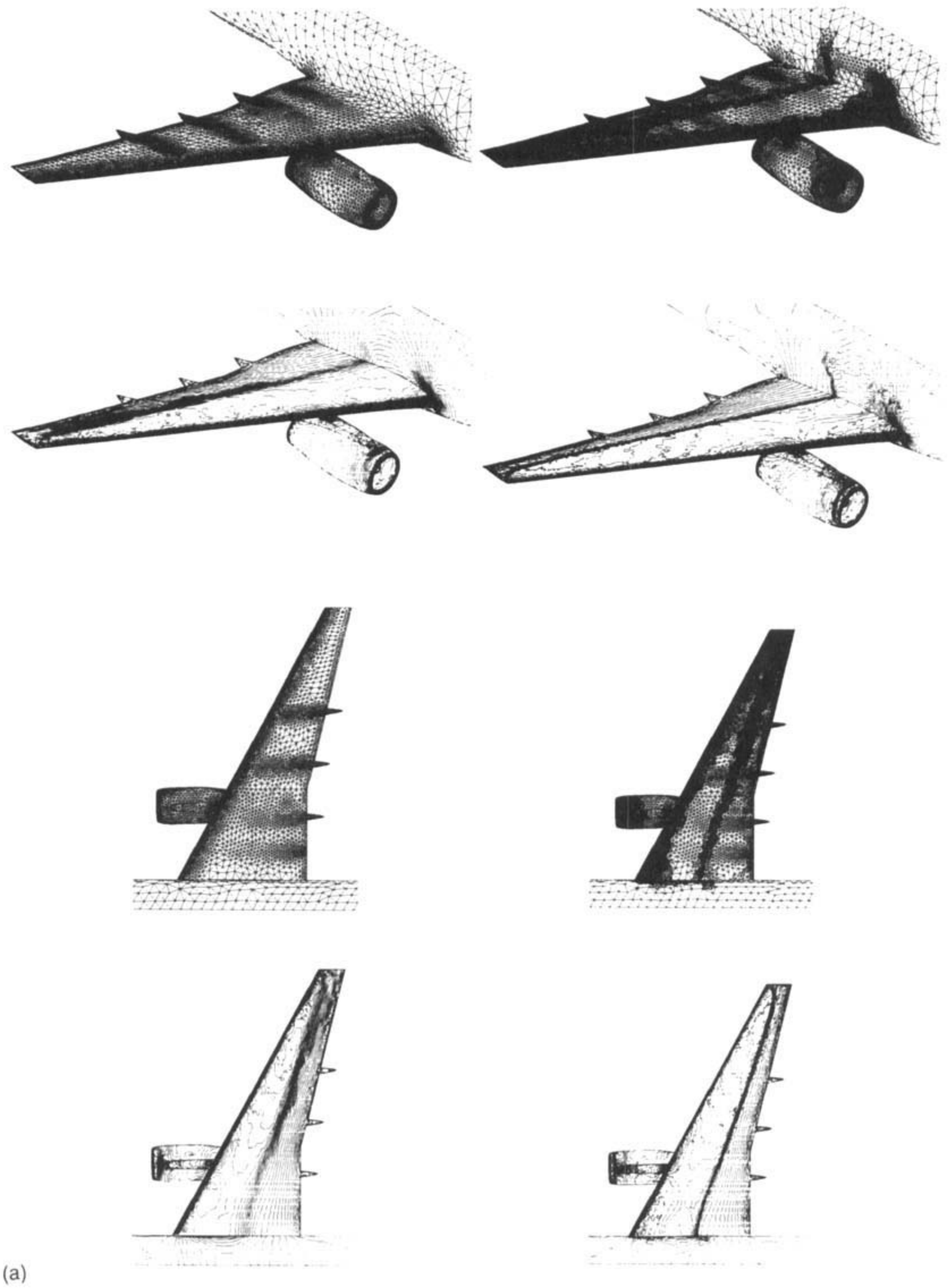
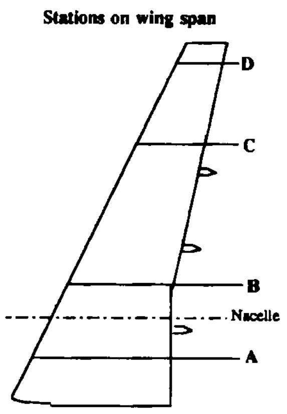
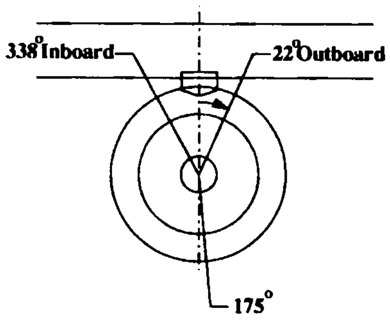

(a)

Figure 12. Transonic flow over wing/body/pylon/nacelle configuration: (a) B60 surface mesh and pressure contours; (b) C_p stations along the aircraft wing; (c) C_p stations around the nacelle; (d) C_p comparisons along the aircraft wing for the adapted meshes; (e) C_p comparisons around the nacelle for the adapted meshes; (f) C_p comparisons along the aircraft wing—adapted and fine non-adapted grids



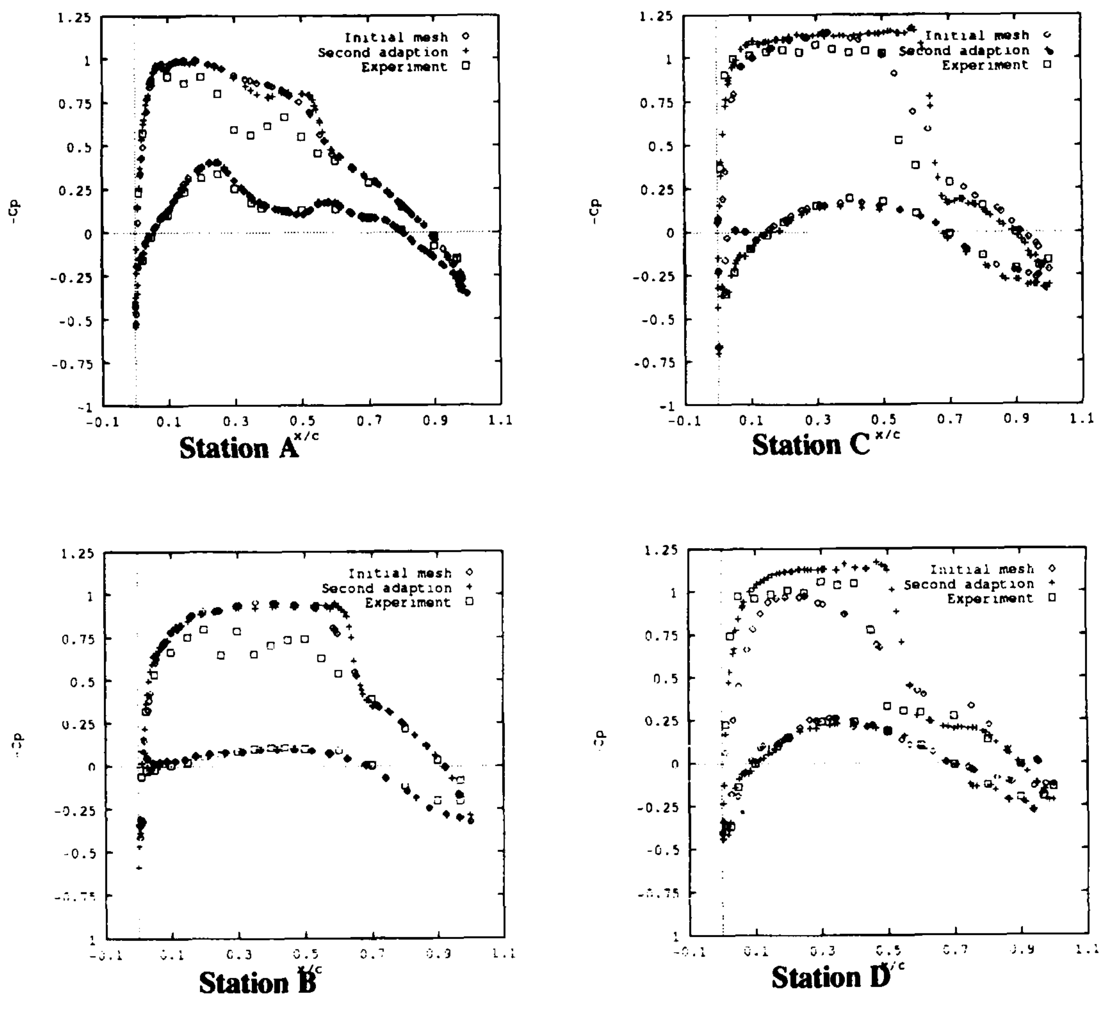
Station	% Span
A	14.4
B	33.4
C	69.6
D	91.9

(b)



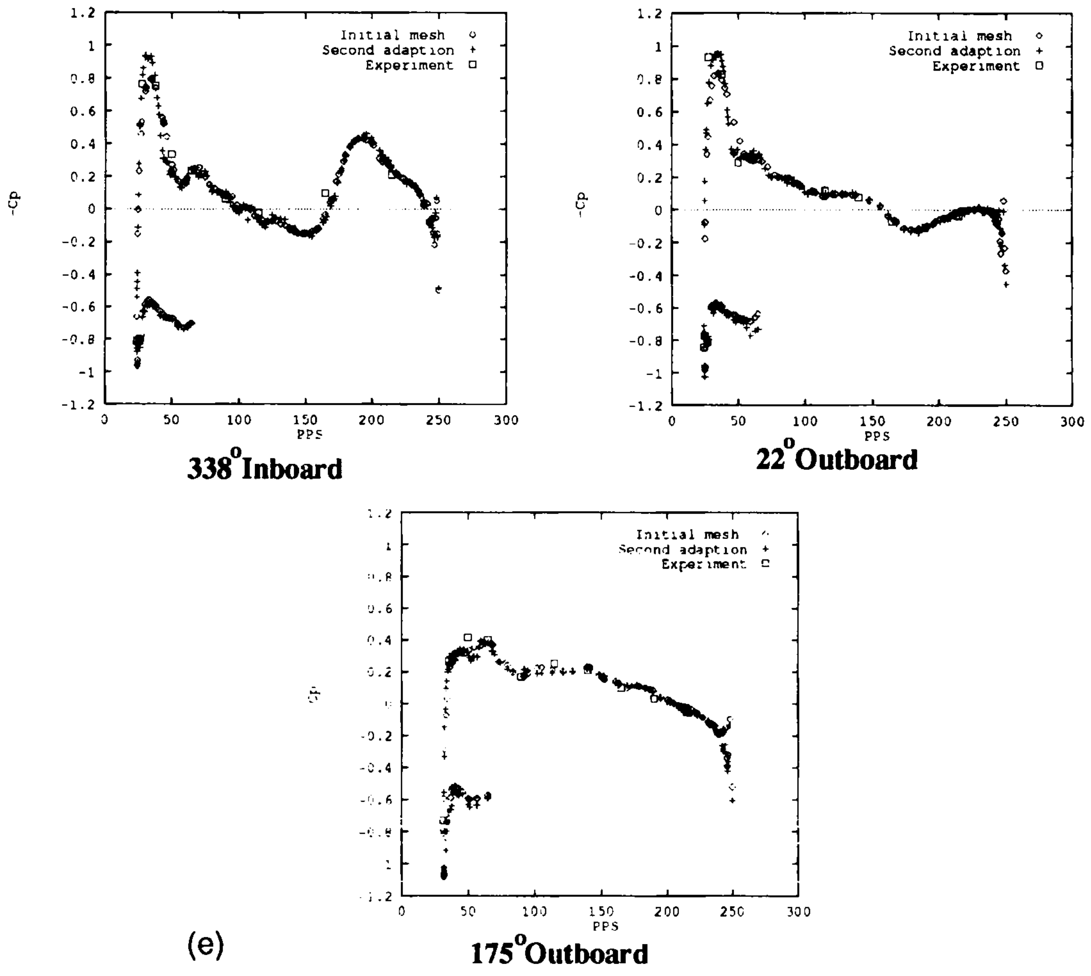
(c)

Figure 12. (*continued*)



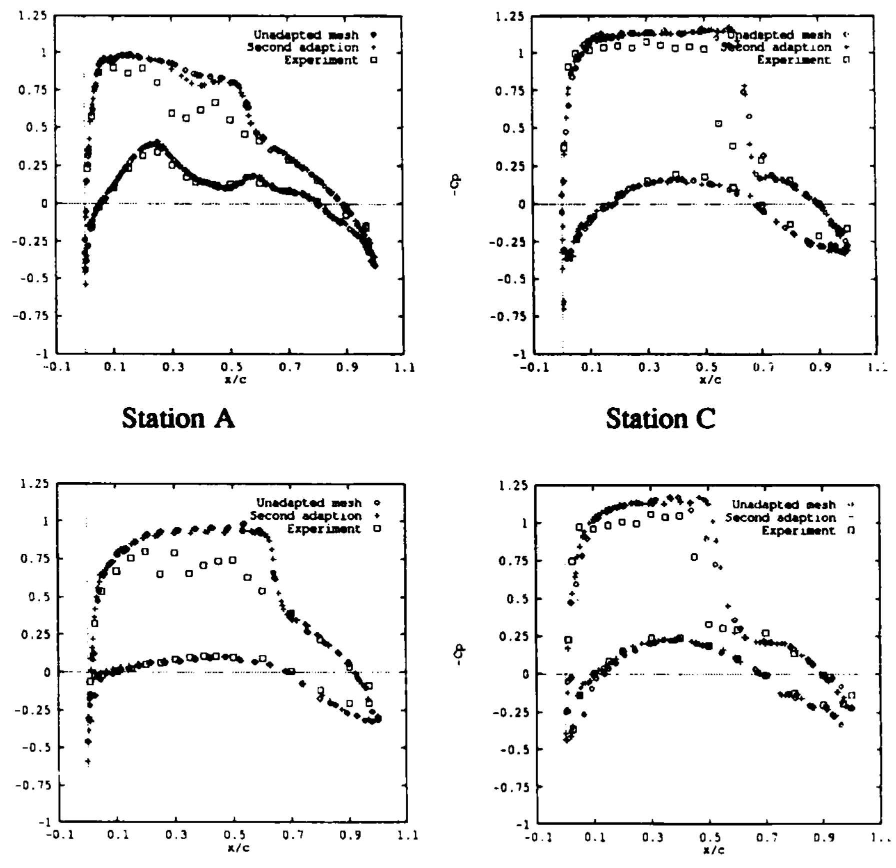
(d)

Figure 12. (*continued*)



(e)

Figure 12. (*continued*)



(f)

Figure 12. (*continued*)

6. CONCLUSIONS

The paper has outlined a new method of grid adaptation using a Delaunay triangulation grid generation algorithm coupled with sources placed on surfaces and in the field to achieve grid refinement. An example has been given which enables the results of the new approach to be compared with more conventional grid adaptation methods. The results obtained demonstrate that the new method obtains comparable results but with fewer levels of refinement. Two examples have been presented for the application of the approach to three-dimensional geometries. Good results have been obtained in both cases using just two levels of refinement. From the results obtained, the procedure for adaptation, integrated with the very efficient Delaunay generator, is cost-effective and potentially more flexible than conventional h-refinement. Future work will extend the range of applications and develop the method further for adaptation in transient problems.

ACKNOWLEDGEMENTS

Computing resources were provided in part by the Science and Engineering Research Council in the U.K. The authors wish to thank Rolls-Royce, in particular Dr. P. Stow, for permission to show the results for the B60 aircraft configuration, and DRA Farnborough for permission to publish the corresponding experimental data.

REFERENCES

1. A. Jameson, T. J. Baker and N. P. Weatherill, 'Calculation of inviscid transonic flow over a complete aircraft', *AIAA Paper 86-0103*, 1986.
2. J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, 'Finite element Euler computations in three dimensions', *Int. j. numer. methods eng.*, **26**, 2135–2159 (1988).
3. R. Lohner and P. Parikh. 'Three-dimensional grid generation by the advancing front method', *Int. j. numer, methods, fluids*, **8**, 1135–1149 (1988).
4. K. Morgan, J. Peraire, J. Peiro and O. Hassan', The computation of three-dimensional flows using unstructured grids', *Comput. Methods Appl. Mech. Eng.*, **87**, 335–352 (1991).
5. N. P. Weatherill and O. Hassan, 'Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints', *Int. j. numer. methods eng.*, **37**, 2005–2040 (1994).
6. N. P. Weatherill, O. Hassan and D. L. Marcum, 'Calculation of steady compressible flowfields with the finite element method', *AIAA Paper 93-0341*, 1993.
7. N. P. Weatherill, 'The generation of unstructured grids using Dirichlet tessellations', *Princeton University, MAE Rep. 1715*, 1985.
8. N. P. Weatherill, 'A method for generating irregular computational grids in multiply connected planar domains', *Int. j. numer. methods fluids*, **8**, 181–197 (1988).
9. N. P. Weatherill, 'Delaunay triangulation in computational fluid dynamics', *Comput. Math. Appl.*, **24**, (5–6), 129–150 (1992).
10. T. J. Baker, 'Three-dimensional mesh generation by triangulation of arbitrary point sets', *Proc. AIAA 8th CFD Conf.*, Honolulu, HI, June 1987.
11. B. Delaunay, 'Sur la sphere vide', *Bull. Acad. Sci. URSS, Class, Sci. Nat.*, 793–800 (1934).
12. G. Voronoi, 'Nouvelles applications des parametres continus a la theorie des formes quadratiques. Recherches sur les parallelloedres primitifs', *J. R. Angew. Math.*, **134**, (1908).
13. A. Bowyer, 'Computing Dirichlet tessellations', *Comput. J.*, **24**, 162–166, (1981).
14. J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North-Holland, Amsterdam, 1985.
15. G. M. Nielson, 'The side-vertex method for interpolation in triangles', *J Approx. Theory*, **25**, 318–336 (1979).
16. D. L. Marcum and R. K. Agarwal, 'A three-dimensional finite element Navier-Stokes solver with k–e turbulence model for unstructured grids', *AIAA Paper 90-1652*, 1990.