JOURNAL OF COMPUTATIONAL PHYSICS 103, 269–285 (1992)

Adaptive Remeshing for Three-Dimensional Compressible Flow Computations

J. PERAIRE AND J. PEIRÓ

Department of Aeronautics, Imperial College of Science, Technology and Medicine, Prince Consort Road, London SW7 2BY, United Kingdom

AND

K. MORGAN

Department of Civil Engineering, University College, Swansea SA2 8PP, United Kingdom

Received November 12, 1990; revised October 11, 1991

An adaptive mesh procedure for computing steady state solutions of the compressible Euler equations in three dimensions is presented. The method is an extension of previous work in two dimensions. The approach requires the coupling of a surface triangulator, an automatic tetrahedral mesh generator, a finite element flow solver and an error estimation procedure. An example involving flow at high Mach number is included to demonstrate the numerical performance of the proposed approach. The example shows that the use of this form of adaptivity in three dimensions offers the potential of even greater computational savings than those attained in the corresponding two-dimensional implementation. © 1992 Academic Press, Inc.

1. INTRODUCTION

One of the most challenging problems currently facing computational fluid dynamicists is the accurate solution of industrial three-dimensional flows. These flows are characterised by solution domains of complicated shape and/or by the appearance of complex flow features. Assuming that the geometry of the computational domain can be adequately represented, accurate solutions can be produced by existing techniques, provided that the mesh employed is sufficient to resolve the solution. Approaches which attempt to achieve this by a priori generating the finest grid which will fit into the largest available computer are of limited usefulness. The natural alternative is to attempt to adapt the grid in some fashion, according to the computed solution, so that the best use is made of the available grid points by distributing them in an 'optimal' manner.

A popular approach for the analysis of two-dimensional compressible inviscid flows has been to attempt to achieve this adaptation of the mesh by local enrichment and successful demonstrations have been made on both quadrilateral [1, 2] and triangular grids [3–5]. However, as there is a significant increase in the problem size following each adaptation, experiences with this method in three dimensions [4, 6] have indicated that, currently, only one or two refinements can generally be afforded. An alternative adaptive approach, which has proved to be successful in three dimensions, is to move the mesh nodes while maintaining the original grid connectivity [7, 8]. With this approach, however, as no new nodes are added, the accuracy of the final computation is limited by the structure and resolution of the initial grid.

In an earlier paper [9], the authors addressed these problems in two dimensions within the context of solution algorithms implemented on unstructured triangular grids. The essential feature of the advocated approach was the introduction of a general triangular advancing front mesh generator, which was capable of automatically generating meshes according to a prescribed distribution of mesh spacing. In addition, directionality was included by allowing for the generation of stretched elements according to a prescribed distribution of stretching directions. When a computed solution was subjected to a directional error indication procedure, information about the 'optimal' mesh distribution was obtained and the mesh was adapted to the computed solution by complete regeneration. This adaptive method allowed for the creation of new nodes in the regions where higher resolution was deemed to be required and, at the same time, the mesh could be made coarser in smoother regions of the flow. Several examples were included which showed how the solution quality could be enhanced without a dramatic increase in the number of unknowns. The results showed that the advocated adaptive algorithm combined the main advantages of both the mesh enrichment and the mesh movement methods. This approach has subsequently been successfully followed by other workers in the same area [10–12] and has been extended to handle the simulation of transient compressible flows involving moving bodies [13–15]. By simply changing the error estimator, the method has also found application in the fields of solid mechanics [16–19] and heat transfer analysis [20].

It is therefore natural to attempt to further extend these ideas and investigate the possibilities of producing, in this way, a practical solution algorithm for steady three-dimensional Euler flows. The essential feature of the approach will again be an advancing front mesh generator, which in this case will be capable of generating tetrahedral elements of a prescribed size and with a prescribed direction of stretching. The boundary of the computational domain will be discretised into triangular elements and this collection of triangular faces will form the initial front for the tetrahedral generator. The boundary triangulation process requires that the surface of the computational domain be represented in a mathematically convenient form. An adaptive mesh method can then be proposed, exactly as in two dimensions, in which a computed solution is subjected to an error indicating procedure and a new grid is generated according to the predicted distributions of mesh spacing and stretching. In this paper, we describe this three-dimensional extension, involving the coupling of a surface triangulator, an automatic tetrahedral mesh generator, an unstructured grid flow solver, and an error estimating procedure. An efficient computational implementation of the proposed method is essential if the solution of realistic problems is to be attempted and a description of how this can be achieved in this context is given. The numerical performance of the adaptive remeshing procedure in three dimensions is demonstrated for a problem involving a shock interaction in high Mach number flow over a swept circular cylinder.

2. GEOMETRY MODELLING

Three-dimensional flow computations can be performed when the chosen computational domain has been suitably discretised. Here the domain to be discretized is viewed as a three-dimensional region which is bounded by surfaces which intersect along curves. The portions of these curves and surfaces needed to define the three-dimensional domain of interest are called curve and surface components, respectively. Figure 1 shows the decomposition of the boundary of a three-dimensional domain into its appropriate surface and curve components. For computational convenience a mathematical definition of these components is employed. This definition is achieved by using composite curves and surfaces [21–23] which are fitted through a user-prescribed set of points.

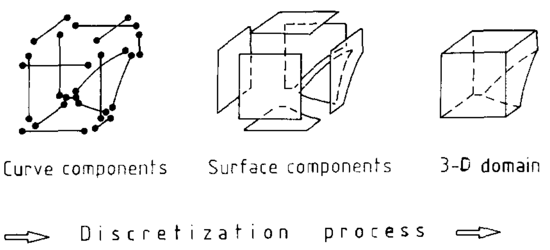


FIG. 1. Decomposition of the boundary of a three-dimensional computational domain into surface and curve components.

2.1. *Curve Representation*

Curves are represented in parametric form by constructing a piecewise interpolation of cubic polynomials through an ordered set of data points. In the Ferguson representation [22], each cubic polynomial is expressed, in terms of the position and tangent vectors at the two end points, as

$$\mathbf{r}(v)=\{1\ v\ v^2\ v^3\}\mathbf{C}\begin{bmatrix}\mathbf{r}^{(1)}\\ \mathbf{r}^{(2)}\\ \mathbf{t}^{(1)}\\ \mathbf{t}^{(2)}\end{bmatrix},\qquad 0\leqslant v\leqslant 1, \tag{1}$$

where $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$ are the coordinates of the end points of the segment, $\mathbf{t}^{(1)}$ and $\mathbf{t}^{(2)}$ are their respective tangents, and $\mathbf{C}$ is a constant matrix given by

$$\mathbf{C}=\begin{bmatrix}1&0&0&0\\0&0&1&0\\-3&3&-2&-1\\2&-2&1&1\end{bmatrix}. \tag{2}$$

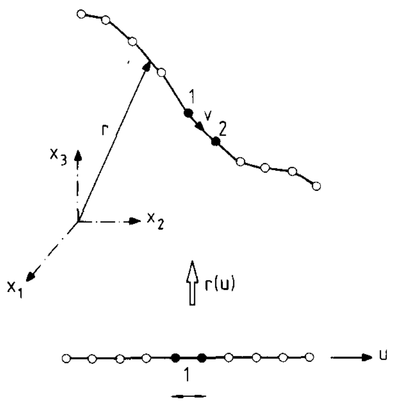


FIG. 2. Interpolation of a piecewise cubic spline through a set of points.

Here, the tangent to the curve is defined to be

$$\mathbf{t}(v) = \mathbf{r}'(v) = \frac{d\mathbf{r}(v)}{dv}. \tag{3}$$

The number of data points, and their spatial distribution, should be given in such a manner that the interpolated curve accurately approximates the intersection of the corresponding surface components. The interpolation problem, which is illustrated in Fig. 2, consists of fitting a parametric spline, defined in a piecewise manner, through a set of n points $\mathbf{r}_j, j = 1, ..., n$. At interior points, continuity of slopes is guaranteed for any choice of the tangent vectors, provided that a unique tangent vector is used for the definition of the two adjacent cubic segments. However, by employing a simple procedure [21], these vectors can be determined so that continuity of curvature is achieved throughout the interpolated curve. At the two end points, zero curvature is assumed. For convenience, a global parametric coordinate u is defined. For the cubic polynomial joining points j and $j+1$ this coordinate is related to the local coordinate v according to $u = v + j - 1$. In this way, a global mapping $\mathbf{r}(u)$ is established between the region in the parametric space defined by $0 \leqslant u \leqslant n-1$ and the cubic composite curve.

2.2. *Surface Representation*

The mathematical representation of a surface component is obtained by interpolating a composite surface, made up of

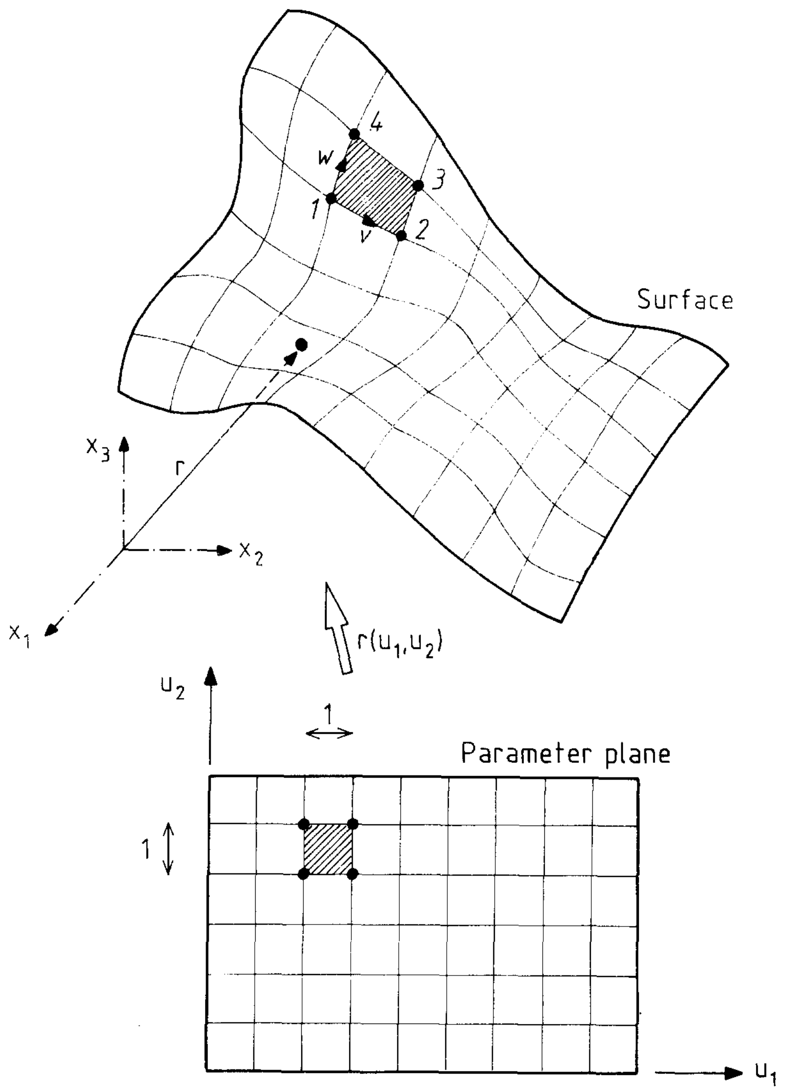


FIG. 3. Interpolation of a composite surface through a set of points.

quadrilateral patches, through a topologically rectangular set of data points $\mathbf{r}_{jk}, j = 1, ..., p; k = 1, ..., q$ (see Fig. 3). Two families of parametric lines are obtained by interpolating spline curves, first through the points of constant j and then through the points of constant k. The procedure used for interpolating each spline curve is that described in the previous section. The mathematical expression for every quadrilateral surface patch is given, in terms of the four cubic curves that form its boundary and the twist vector at the four corner points, as

$$\mathbf{r}(v, w) = (1\ v\ v^2\ v^3)\mathbf{C} \begin{bmatrix} \mathbf{r}^{(1)} & \mathbf{r}^{(4)} & \mathbf{r}^{(1)}_{,w} & \mathbf{r}^{(4)}_{,w} \\ \mathbf{r}^{(2)} & \mathbf{r}^{(3)} & \mathbf{r}^{(2)}_{,w} & \mathbf{r}^{(3)}_{,w} \\ \mathbf{r}^{(1)}_{,v} & \mathbf{r}^{(4)}_{,v} & \mathbf{r}^{(1)}_{,vw} & \mathbf{r}^{(4)}_{,vw} \\ \mathbf{r}^{(2)}_{,v} & \mathbf{r}^{(3)}_{,v} & \mathbf{r}^{(2)}_{,vw} & \mathbf{r}^{(3)}_{,vw} \end{bmatrix}$$

$$\times \mathbf{C}' \begin{bmatrix} 1 \\ w \\ w^2 \\ w^3 \end{bmatrix}, 0 \leqslant v \leqslant 1; 0 \leqslant w \leqslant 1 \qquad (4)$$

where $\mathbf{C}$ is the matrix previously defined in (2), $\mathbf{C}'$ is its transpose, and the comma denotes partial differentiation, i.e.,

$$\mathbf{r}_{,v} = \frac{\partial \mathbf{r}}{\partial v}, \qquad \mathbf{r}_{,w} = \frac{\partial \mathbf{r}}{\partial w}, \qquad \mathbf{r}_{,vw} = \frac{\partial^2 \mathbf{r}}{\partial v\, \partial w}. \qquad (5)$$

Here the corner points of the patch are given as

$$\mathbf{r}^{(1)} = \mathbf{r}(0, 0), \qquad \mathbf{r}^{(2)} = \mathbf{r}(1, 0),$$
$$\mathbf{r}^{(3)} = \mathbf{r}(1, 1), \qquad \mathbf{r}^{(4)} = \mathbf{r}(0, 1). \qquad (6)$$

This representation uses a Hermite interpolation between opposite boundaries of the patch [23]. The twist vectors $(\mathbf{r}_{,vw})_{jk}$ at the corner points are computed so that overall second-order continuity is achieved on the interpolated surface. The implementation details of this algorithm can be found in [24]. Global parametric coordinates u_1 and u_2 are defined, such that for the patch (j, k) these coordinates are related to the local patch coordinates v and w according to $u_1 = v + j - 1$ and $u_2 = w + k - 1$. Hence, a global mapping $\mathbf{r}(u_1, u_2)$ is established between the rectangular region in the parametric plane defined by $0 \leqslant u_1 \leqslant p - 1$, $0 \leqslant u_2 \leqslant q - 1$, and the tensor product surface.

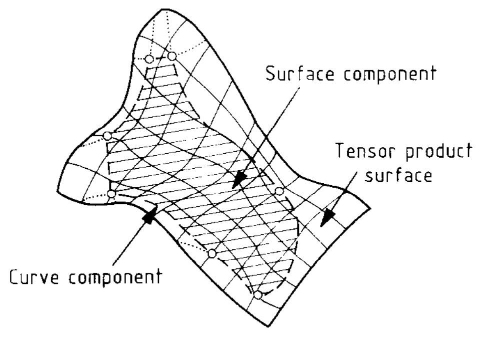


FIG. 4. Approximate geometry for a surface component.

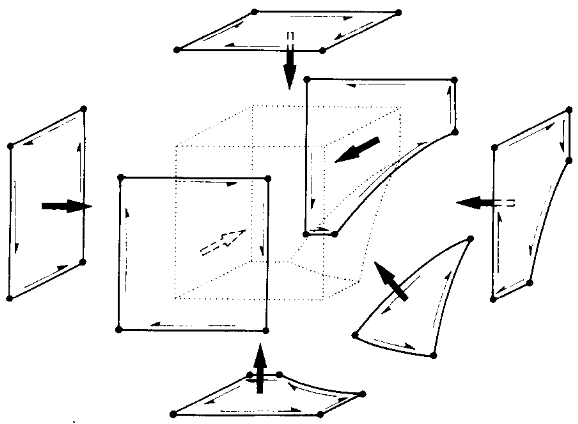


FIG. 5. Orientation of the curve components relative to the surface components.

An example of the approximated geometry for a surface component and its corresponding curve components is displayed in Fig. 4. It can be observed that the boundaries of the interpolated composite surface are not required to coincide with those of the surface component.

For each surface component the set of data points is defined in such a way that the vector product of $\mathbf{r}_{,v}$ and $\mathbf{r}_{,w}$ points into the computational domain. In addition, the curve components have an orientation relative to the surface component being considered. This is illustrated in Fig. 5.

3. MESH GENERATION

The algorithmic procedure adopted for the generation of tetrahedra is a three-dimensional extension of the triangulation method proposed in [9] and is based upon a generalization of the advancing front technique [25, 26]. The generation problem consists of subdividing an arbitrarily complex three-dimensional domain into a consistent assembly of tetrahedra. A distinctive feature of the approach is that tetrahedra and points are generated simultaneously. This enables the generation of elements of variable size and stretching and differs from the approach followed in tetrahedral generators which are based upon Delaunay concepts [27, 28], which generally connect grid points which have already been distributed in space.

The three-dimensional mesh is constructed by first discretising the boundary curve components, then subdividing the boundary surfaces into triangular planar faces, and finally discretising the three-dimensional domain into tetrahedra.

3.1. *Characterisation of the Mesh: Mesh Parameters*

The geometrical characteristics of a general mesh are locally defined in terms of certain mesh parameters [9]. In three dimensions the parameters used are a set of three mutually orthogonal directions $\boldsymbol{\alpha}_i$, $i = 1, 2, 3$, and three associated element sizes δ_i, $i = 1, 2, 3$ (see Fig. 6). Thus, at a certain point, if all three element sizes were equal, the mesh in the vicinity of that point would consist of approximately equilateral tetrahedra. For convenience a transformation **T** is introduced which is defined as a function of $\boldsymbol{\alpha}_i$ and δ_i. This transformation is represented by a symmetric 3×3 matrix and maps the physical space onto a space in which tetrahedra, in the neighbourhood of the point being considered, will be approximately equilateral with unit average size. This new space will be referred to as the normalised space. The transformation **T** is the result of superimposing three scaling operations with factors $1/\delta_i$ in each $\boldsymbol{\alpha}_i$ direction. Thus

$$\mathbf{T}(\boldsymbol{\alpha}_i, \delta_i) = \sum_{i=1}^{3} \frac{1}{\delta_i} \boldsymbol{\alpha}_i \otimes \boldsymbol{\alpha}_i, \tag{7}$$

where $\otimes$ denotes the tensor product of two vectors. The effect of this transformation in two dimensions is illustrated in Fig. 7 for a case in which the mesh parameters are constant throughout the domain.

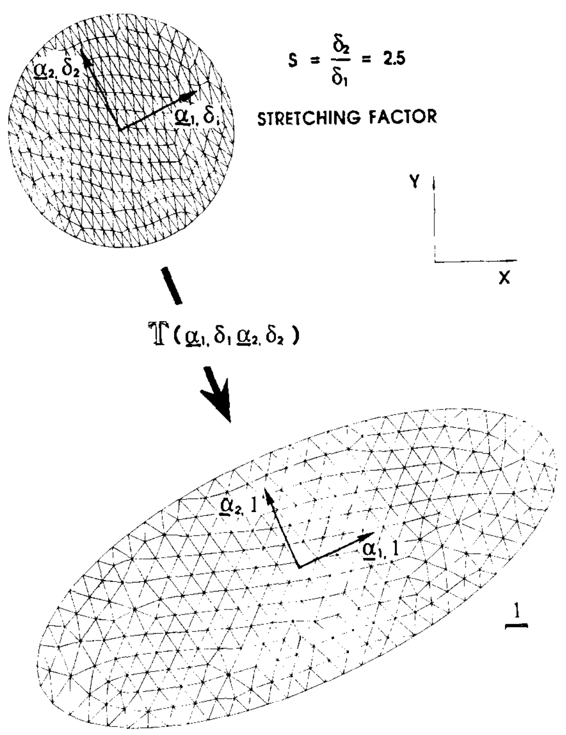

$\underline{\alpha}_2, \delta_2$
$\underline{\alpha}_1, \delta_1$
$S = \frac{\delta_2}{\delta_1} = 2.5$
STRETCHING FACTOR
Y
X
$\mathbb{T}(\underline{\alpha}_1, \delta_1 \underline{\alpha}_2, \delta_2)$
$\underline{\alpha}_2, 1$
$\underline{\alpha}_1, 1$
1

FIG. 7. The transformation **T** applied to a mesh with constant distribution of mesh parameters.

3.2. *Mesh Control: The Background Mesh*

The inclusion of adequate mesh control is a key ingredient in ensuring the generation of a mesh of the desired form. Control over the mesh characteristics is obtained by the specification of a spatial distribution of mesh parameters. This is achieved by the use of a background mesh. The background mesh is used for interpolation purposes only and is made up of tetrahedral elements. Values of $\boldsymbol{\alpha}_i$ and δ_i, and hence **T**, are defined at the nodes of the background mesh. At any point within an element of the background mesh, the transformation **T** is computed by linearly interpolating its components from the element nodal values. The background mesh employed must cover the region to be discretized. In the generation of an initial mesh for the analysis of a particular problem, the background mesh will usually consist of a small number of tetrahedra. The generation of the background mesh in this case can thus be accomplished without resorting to sophisticated procedures. For instance, a background mesh consisting of a single element can be used to impose the requirement of linear or constant spacing and stretching through the computational domain.

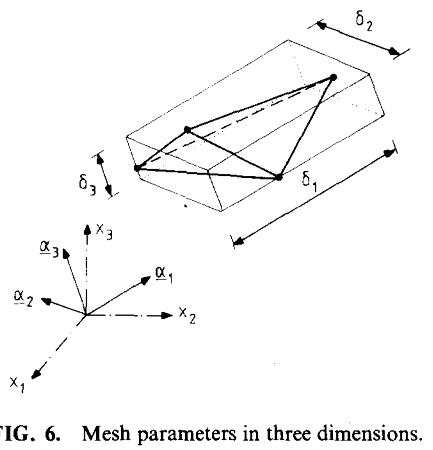


FIG. 6. Mesh parameters in three dimensions.

3.3. *Curve Discretization*

The discretization of the boundary curve components is performed by positioning nodes along the curves according to the spacing dictated by the local value of the mesh parameters. In order to determine the position and number of nodes to be created on each curve component, the following steps are followed:

(i) Each cubic segment is subdivided into smaller segments whose length is less than the minimum mesh spacing specified in the background mesh. When subdividing a cubic segment, the position and tangent vectors corresponding to the new data points are found directly from the original definition of the segment.

(ii) The points used to define the curve and those created to satisfy the maximum length criterion in step (i) above are denoted by $\mathbf{r}_j$, $j = 1, ..., n_s$. For each such point $\mathbf{r}_j$ the coefficients of the transformation $\mathbf{T}_j$ are interpolated from the background mesh and the position and tangent vectors are transformed according to $\hat{\mathbf{r}}_j = \mathbf{T}_j \mathbf{r}_j$ and $\hat{\mathbf{t}}_j = \mathbf{T}_j \mathbf{t}_j$. The vectors $\hat{\mathbf{r}}_j$, $\hat{\mathbf{t}}_j$, $j = 1, ..., n_s$, define a spline curve which can be interpreted as the image of the original curve component in the normalised space.

(iii) The length of the curve component is computed in the normalised space and subdivided into segments of approximately unit length. For each newly created point, its parametric coordinates and the cubic segment in which it is contained are calculated. This information is used to determine the coordinates of the new nodes in the physical space, using the curve component definition.

Consecutive points are then joined by straight lines and these lines will form element sides in the final mesh.

3.4. *Surface Discretization*

The next stage in the process consists of subdividing the boundary surfaces into triangular planar faces [24]. For each surface component, the straight lines produced when discretising its boundary curves are assembled into the so-called initial front. The relative orientation of the curve components with respect to the surface must be taken into account in order to give the correct orientation to the sides in this front. The method followed for the triangulation of the surface components is then an extension of the two-dimensional advancing front mesh generation procedure [9]. In this approach, the front is a dynamic data structure which changes continuously during the generation process. At any given time, the front contains the set of all the sides which are currently available to form a triangular face. A side is selected from the front and a triangular element is generated. This may involve the creation of a new node or simply connecting to an existing one. After a triangle has been generated, the front is updated and the generation proceeds until the front is empty. Figure 8 illustrates the idea of the advancing front technique applied to a circular planar domain and shows the initial front and the form of the mesh and the front at various stages during the generation process.

In the present context, the discretization of each surface component is accomplished by generating a two-dimensional mesh of triangles in the parametric plane $u_1 u_2$ and then using the mapping $\mathbf{r}(u_1, u_2)$ defined in Section 2.2. This mapping establishes a one to one correspondence between the boundary surface component and a region on the parametric plane $u_1 u_2$ (Fig. 9). Thus, a consistent triangular mesh in the parametric plane will be transformed, by the mapping $\mathbf{r}(u_1, u_2)$, into a valid triangulation of the surface component. The construction of the triangular mesh in the parameter plane $u_1 u_2$, using the two-dimensional mesh generator, requires the determination of an appropriate spatial distribution of two-dimensional mesh parameters. These consist of a set of two mutually orthogonal directions $\boldsymbol{\alpha}_i^*$, $i = 1, 2$, and two associated element sizes δ_i^*, $i = 1, 2$.

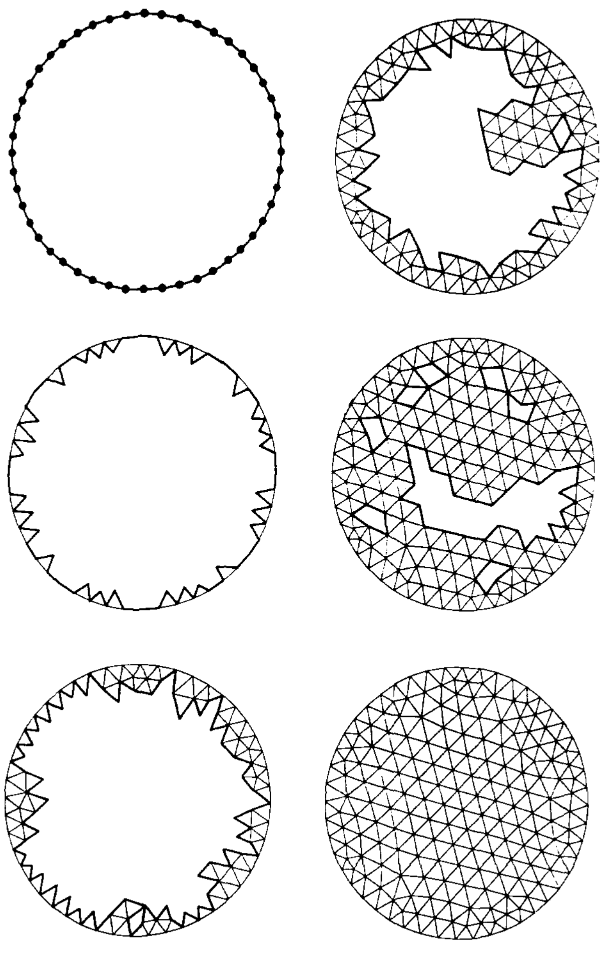
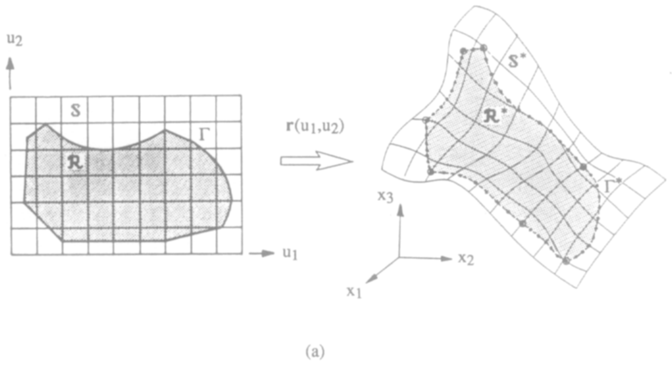


FIG. 8. The advancing front technique applied to a circular planar region. Different stages during the triangulation process.

The two-dimensional mesh parameters in the $u_1 u_2$ plane can be evaluated from the spatial distribution of the three-dimensional mesh parameters and the distortion and stretching introduced by the mapping. To illustrate this process, consider the evaluation of the mesh parameters δ_i^*, $\boldsymbol{\alpha}_i^*$, $i = 1, 2$, at a point P^*, with coordinates (u_{1P}, u_{2P}) in the parametric plane. The image of P^* on the surface will be the point P with position vector $\mathbf{r}(u_{1P}, u_{2P})$. The transformation $\mathbf{T}_P$ between the physical space and the normalised space at this point can be obtained by direct interpolation from the background mesh. In the neighbourhood of P, a new



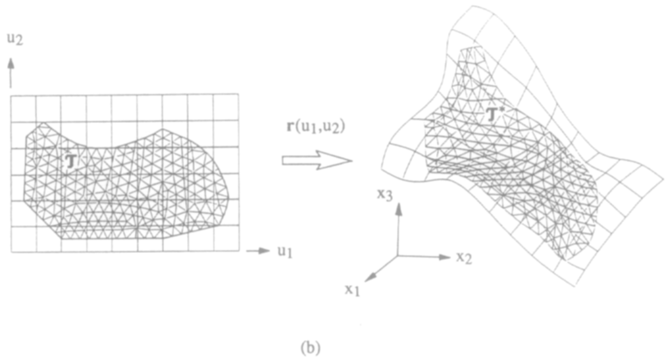


FIG. 9. Inducing a triangulation on a surface component by mapping: (a) the mapping $\mathbf{r}(u_1, u_2)$; (b) transformation of a triangular mesh in the parameter plane into a surface triangulation.

mapping can now be defined between the parametric plane $u_1 u_2$ and the normalised space as

$$\mathbf{R}(u_1, u_2) = \mathbf{T}_P \mathbf{r}(u_1, u_2). \tag{8}$$

A curve in the parametric plane passing through point P^* and with unit tangent vector $\boldsymbol{\beta} = (\beta_1, \beta_2)$ at this point, is transformed by the above mapping into a curve in the normalised space passing through the point $\mathbf{T}_P \mathbf{r}(u_{1P}, u_{2P})$. The arc length parameters ds and $d\zeta$, along the original and transformed curves, respectively, are related by the expression [29]

$$(d\zeta)^2 = \left\{ \sum_{i;j=1}^{2} \frac{\partial \mathbf{R}}{\partial u_i} \frac{\partial \mathbf{R}}{\partial u_j} \beta_i \beta_j \right\} (ds)^2. \tag{9}$$

This relation between the arc length parameters is assumed to hold also for the spacings, so that the spacing δ_β along the direction $\boldsymbol{\beta}$ in the parameter plane may be computed as

$$\delta_\beta = \sqrt{\sum_{i;j=1}^{2} \frac{\partial \mathbf{R}}{\partial u_i} \frac{\partial \mathbf{R}}{\partial u_j} \beta_i \beta_j}. \tag{10}$$

The two-dimensional mesh parameters $\boldsymbol{\alpha}_i^*$, δ_i^*, $i = 1, 2$, are determined from the directions in which δ_β attains an extremum. This reduces to finding the eigenvalues and eigenvectors of a symmetric 2×2 matrix.

By assembling the discretised boundary surface components, an initial front for the domain discretisation procedure is constructed from the set of oriented triangles which constitute the discretized boundary of the domain. The order in which the nodes of these triangles are given defines the orientation, which is the same as that of the corresponding boundary surface component. To form the front, the (u_1, u_2) coordinates of the nodes already generated on the boundary curve components have to be computed. The coordinates (u_1, u_2) of such points are found numerically by using a direct iteration procedure [24].

3.5. *Generation of Tetrahedra*

The final stage in the process is the discretisation of the three-dimensional domain into tetrahedra. Although the generation of tetrahedra proceeds along similar lines to the algorithm described in [9] for the generation of triangles, in the three-dimensional case the range of possible options at each stage is much wider and the number of geometrical operations involved increases considerably. Thus, the ability of the method to produce a mesh and the efficiency of its implementation rely heavily upon the type of strategy selected. The approach adopted here for the generation of a generic tetrahedral element involves the following steps (Fig. 10):

(i) A triangular face ABC is selected from the front. This will be used as a base for the tetrahedron to be generated. In principle, any face from the front could be chosen but we have found it to be advantageous in practice to consider the smallest faces first. It has been found that this choice ensures that the generated mesh follows more correctly the specified distribution of mesh parameters, especially in regions of rapid variation. For this purpose, the size of the face is defined to be the size of its shortest height.

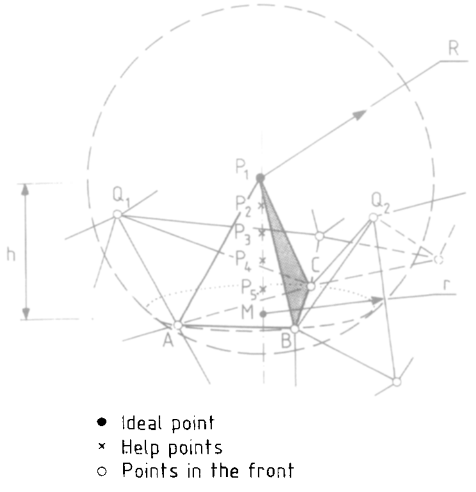


FIG. 10. Generation of a generic tetrahedral element.

(ii) The transformation **T**, at the centroid of the face M, is interpolated from the background grid and applied to the nodes in the front which are relevant to the triangulation. In our implementation we define the relevant points to be those which lie inside the sphere of center M and radius three times the value of the maximum dimension of the face being considered.

(iii) The ideal position P_1 for the vertex of the tetrahedral element in the transformed space is determined. The point P_1 is placed, at a distance δ_1 from the point M, on the line which passes through M and which is perpendicular to the face. The orientation of the face uniquely determines the location of the point P_1. The value δ_1 is computed so that the average length of the three newly created sides which join point P_1 with points A, B, and C is unity.

(iv) Other possible candidates for the vertex are selected and ordered in a list. Two types of points are considered viz.: (a) the nodes Q_1, Q_2, ... in the current generation front which are, in the normalised space, interior to a sphere with centre P_1 and radius $r = 1.5$ and which lie on the same side of the plane ABC as the point P_1; and (b) the set of points $P_1, ..., P_5$ generated along the height $P_1 M$. For each point Q_i, a sphere is constructed with center O_i on the line defined by points P_1 and M. The radius of the sphere is chosen such that it passes through the points Q_i and D, where D is the member of the set A, B, C which is furthest away from M. The points O_i are placed in an ordered list according to their distance from the point P_1. The points $P_1, ..., P_5$ are added at the end of this list.

(v) The best connecting point is selected. This is the first point in the list which gives a consistent tetrahedron. Consistency is guaranteed by ensuring that none of the newly created sides intersects with any of the existing faces in the front and that none of the existing sides in the front intersect with any of the newly created faces.

(vi) Finally, if a new node is created, its coordinates in the physical space are obtained by using the inverse transformation $\mathbf{T}^{-1}$.

3.6. *Data Structures*

From the previous section it is apparent that a successful implementation of the presented algorithm will require the use of data structures which enable certain sorting and searching operations to be performed efficiently. For instance, the generation front will require a data structure which allows for the efficient insertion/deletion of items, or faces, and which also enables the operations of geometric searching and intersection to be performed efficiently; e.g., we frequently require the identification of the items inside a prescribed region or the items which intersect in space with a given geometrical object. In our implementation we use the alternate digital tree [30]. This is basically a generalisation of the binary tree structure [31] that has the ability to handle geometric objects such as nodes, faces, or tetrahedra as single items. When this data structure is used for the generation front and for the background mesh, the computational cost involved in generating three-dimensional meshes scales typically as $N^* \log(N)$, where N is the number of elements generated. This data structure results in a scalar process, with intensive indirect addressing operations.

4. SOLUTION OF THE COMPRESSIBLE EULER EQUATIONS

The equations governing three-dimensional compressible flow are considered in the conservative vector form as

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{j=1}^{3} \frac{\partial \mathbf{F}_i}{\partial x_i} = \mathbf{0}, \tag{11}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ \rho \varepsilon \end{bmatrix}, \qquad \mathbf{F}_i = \begin{bmatrix} \rho v_i \\ \rho v_1 v_i + \delta_{1i} p \\ \rho v_2 v_i + \delta_{2i} p \\ \rho v_3 v_i + \delta_{3i} p \\ v_i(\rho\varepsilon + p) \end{bmatrix} \tag{12}$$

with ρ, p, and ε denoting the density, pressure, and specific total energy of the fluid, respectively, and v_i the component of the fluid velocity in the x_i direction. The pressure is determined from the state equation for a perfect gas

$$p = (\gamma - 1)\,\rho(\varepsilon - \tfrac{1}{2}(v_1^2 + v_2^2 + v_3^2)), \tag{13}$$

where γ is the ratio of specific heats.

Here we are concerned with computing the solutions of steady compressible flows, which will be obtained by advancing equation (11) in time, starting from a given initial condition, until a steady state is reached. In the example to be reported later, the equations are solved by using an explicit time marching scheme with the explicit addition of artificial viscosity for the capturing of discontinuities. The vector of unknowns **U** is approximated using piecewise linear continuous finite element shape functions. This scheme has been widely reported for two- and three-dimensional simulations [6, 32–35] and therefore will not be considered further here. The type of solver employed is not essential for the adaptive strategy to be presented below; e.g., for two-dimensional simulations, the same adaptive procedure has also been employed in conjunction with implicit upwind solvers [36, 37].

5. ADAPTIVE REMESHING

The procedures described above allow for the computation of an initial approximation to the steady state solution of a given problem. This approximation can generally be improved by adapting the mesh in some manner. Here, following the approach presented in [9], the proposal is that the computed solution be used to predict the desired characteristics (i.e., element size and shape) for a new, adapted mesh. The ultimate aim of the adaptation procedure is to predict the characteristics of the optimal mesh. This can be defined as the mesh in which the number of degrees of freedom required to achieve a specified level of accuracy is a minimum. Alternatively, it can be interpreted as the mesh in which a given number of degrees of freedom are distributed in such a manner that the accuracy of the solution obtained is the highest possible. In practical situations, however, there are several factors which make the achievement of such optimal meshes extremely difficult. Some of these factors are:

(i) The concept of optimality is intimately linked to that of accuracy, which is not uniquely defined. Hence optimality of a mesh needs to be defined with respect to a given norm or measure of the error. An additional inconvenience related to the measure of accuracy, in the present context, arises from the fact that we are attempting to solve a coupled set of nonlinear partial differential equations and, therefore, a rigorous measure of the error should involve all the relevant variables.

(ii) For linear elliptic operators, finite element algorithms are readily derived which guarantee that the approximation obtained is the most accurate amongst all the possible approximations within the space of trial functions [38]. The accuracy is defined with respect to a norm implied by the operator itself (the energy norm). For the Euler equations, however, such an energy norm does not exist and no numerical schemes are known which possess an equivalent best approximation property.

(iii) This best approximation property means that the error of the computed solution, measured in the energy norm, is bounded above by that of the exact interpolant, i.e., the approximation in the space of current trial functions which has exact nodal values. Using results of interpolation theory [38], it is then possible to produce rigorous bounds on the error of the numerical approximation. These results are based on certain regularity assumptions on the solution, which for the Euler equations will be invalid in the vicinity of discontinuities in the flow.

(iv) Finally, the error estimates produced are based on the computed solution. As this is only an approximate solution, the error estimates will only be as good as the computed solution. This means that, even in the best situation, the optimal mesh will only be achieved in the asymptotic limit, i.e., when the solution is so good that the computed error becomes very reliable.

In view of these observations and limitations, we have developed a heuristic adaptive strategy. This strategy uses error estimates which are based upon interpolation theory concepts and is a direct extension of the approach proposed in [9]. The possible presence of discontinuities in the solution is taken into account and, in addition, the procedure provides information about any directionality which may be present in the solution. The advantages of using directional error indicators become apparent when we consider the nature of the solutions which are to be computed, involving flows with shocks, contact discontinuities, etc. Such features can be economically represented on meshes which are stretched in appropriate directions. Although, these error estimates have no associated mathematical rigour, considerable success has been achieved with their use in practical situations.

The computed error, estimated from the current solution, is transformed into a spatial distribution of the mesh parameters, interpolated using the current mesh. This mesh parameter distribution describes the characteristics of the new mesh, so that the original background mesh has now been discarded and its role has been taken by the current mesh. The new mesh is generated using the mesh generator described above. The resulting mesh is employed to produce a new solution. This procedure can be repeated several times until the user is satisfied with the quality of the solution.

5.1. *Error indicator in* 1*D*

The development of a method for error indication is considerably simplified if we restrict consideration to problems involving a single scalar variable. For this reason, when solving the Euler equations, a key variable is identified and then the mesh adaptation is based on the error analysis for that variable. The choice of the best key variable remains an open problem, but the density has been adopted for the computations reported in this paper.

We consider first the one-dimensional situation in which the exact values of the key variable σ are interpolated by a piecewise linear function σ^*. The error E is then defined as

$$E = \sigma(x_1) - \sigma^*(x_1). \tag{14}$$

We note here that if σ is a linear function of x_1 then the error will vanish. This is because our interpolation has been constructed using piecewise linear finite element shape functions. Moreover, if σ is not linear, but is smooth, then it can be represented, to any order of precision, using polynomial shape functions.

To a first order of approximation, the error E can be estimated as the difference between a quadratic finite element interpolation $\hat{\sigma}$ and the linear approximation. Assuming that the nodal values of the quadratic and linear approximations coincide, i.e., the nodal values of E are zero, a quadratic approximation can be constructed on each element, once the value of the second derivative is known. Thus the variation of the error E within an element e can be expressed as

$$E_e = \frac{1}{2} \zeta (h_e - \zeta) \left. \frac{d^2 \hat{\sigma}}{dx_1^2} \right|_e, \tag{15}$$

where ζ denotes a local element coordinate and h_e denotes the element length. The root mean square value E_e^{RMS} of this error over the element is computed as

$$E_e^{\mathrm{RMS}} = \left\{ \int_0^{h_e} \frac{E_e^2}{h_e} \, d\zeta \right\}^{1/2} = \frac{1}{\sqrt{120}} h_e^2 \left| \frac{d^2 \hat{\sigma}}{dx_1^2} \right|_e, \tag{16}$$

where $|\cdot|$ stands for absolute value.

We define the "optimal" mesh, for a given degree of accuracy, as the mesh in which this root mean square error is equal over each element. In the present context, this requirement may be regarded as being somewhat arbitrary. However, the requirement of equidistribution of an appropriate error leads to optimal results when applied to elliptic problems. This requirement, therefore, leads to the expression

$$h_e^2 \left| \frac{d^2 \hat{\sigma}}{dx_1^2} \right| = C, \tag{17}$$

where C denotes a positive constant.

5.2. *Recovery of the Second Derivatives*

The first derivative of the computed solution is piecewise constant and discontinuous across element boundaries. Therefore, straightforward differentiation of σ^* leads to a second derivative which is zero inside each element and is not defined at the element boundaries. However, by using a recovery process, based upon a variational or weighted residual statement [40], it is possible to compute approximate nodal values of the second derivatives from element values of the first derivatives of σ^*.

To illustrate this process, consider a one-dimensional domain, $0 < x_1 < L$, which has been discretised into $(n-1)$ linear two-noded finite elements. The piecewise linear distribution of the computed solution σ^* is expressed as

$$\sigma^* = \sum_{i=1}^{n} N_i \sigma_i^*, \tag{18}$$

where N_i is the standard linear finite element shape function [40] associated with node i. Similarly, a piecewise linear approximation to the distribution of the second derivative, which we seek to determine, can be written as

$$\frac{d^2 \sigma^*}{dx_1^2} = \sum_{i=1}^{n} N_i \left. \frac{d^2 \sigma^*}{dx_1^2} \right|_i. \tag{19}$$

Then, the nodal values of the second derivative can be computed from the approximate variational requirement that

$$\begin{aligned} \sum_{i=1}^{n} \left(\int_\Omega N_i N_k \, d\Omega \right) \left. \frac{d^2 \sigma^*}{dx_1^2} \right|_i &= - \int_\Omega \left(\sum_{i=1}^{n} \frac{dN_i}{dx_1} \sigma_i^* \right) \frac{dN_k}{dx_1} \, d\Omega \\ &\quad - \left(\frac{d\sigma^*}{dx_1} N_k \right)_{x_1 = 0} + \left(\frac{d\sigma^*}{dx_1} N_k \right)_{x_1 = L}, \\ k &= 1, ..., n. \end{aligned} \tag{20}$$

The values of the derivatives at the two end points can be inserted, if known, or can be taken to be equal to the constant value of the derivative in the adjacent elements. The resulting set of algebraic equations may be solved by using a few iterations of a Jacobi procedure [41].

5.3. *Computation of the Mesh Parameters*

The computed solution is to be used to produce a spatial distribution of the mesh parameters which will be used by the mesh generator described in Section 2 to generate a new adapted mesh for the problem under investigation. The requirement of Eq. (17) suggests that the desired spacing δ on the new mesh should be computed according to

$$\delta^2 \left| \frac{d^2 \sigma^*}{dx_1^2} \right| = C. \tag{21}$$

Following the process described in the previous section, nodal values of the second derivative can be obtained from the approximate solution on the current mesh. The use of expression (21) then yields directly a nodal value of the local spacing for the new mesh.

For the three-dimensional case, the approach adopted is to compute initially the components m_{ij} of the 3×3 symmetric matrix $\mathbf{m}$ of second derivatives of the computed solution, i.e.,

$$m_{ij} = \frac{\partial^2 \sigma^*}{\partial x_i \, \partial x_j}. \tag{22}$$

This matrix can be decomposed in the form

$$\mathbf{m} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1}, \tag{23}$$

where $\mathbf{\Lambda}$ is a diagonal matrix. Now, expression (21) can be directly extended to the three-dimensional case by writing the quadratic form

$$\delta\beta^2 \left(\sum_{i,j=1}^{3} |m|_{ij} \beta_i \beta_j \right) = C, \tag{24}$$

where $\boldsymbol{\beta}$ is an arbitrary unit vector, $\delta\beta$ is the spacing along the direction of $\boldsymbol{\beta}$, and $|m|_{ij}$ are the components of a 3×3 symmetric matrix, $|\mathbf{m}|$, defined as

$$|\mathbf{m}| = \mathbf{X}\,|\mathbf{\Lambda}|\,\mathbf{X}^{-1}. \tag{25}$$

It can be observed that this generalisation of the one-dimensional analysis amounts to using Eq. (21) independently along each of the principal directions of the Hessian matrix. These derivatives are computed, at each node of the current mesh, by using the three-dimensional equivalent of the procedure presented in the previous section. Figure 11 shows how Eq. (24) is to be interpreted, with the value of the spacing in the $\boldsymbol{\beta}$ direction obtained as the distance from the origin to the point of intersection of the vector $\boldsymbol{\beta}$ with the surface of an ellipsoid.

In a three-dimensional flow simulation, "optimal" values for the mesh parameters can now be obtained at each node of the current mesh. The $\boldsymbol{\alpha}_i$, $i = 1, 2, 3$, directions are taken to be the principal directions of the matrix $\mathbf{m}$. The corresponding spacings are computed from the eigenvalues e_i, $i = 1, 2, 3$, of the matrix $\mathbf{m}$ as

$$\delta_i = \sqrt{C/e_i} \qquad \text{for} \quad i = 1, 2, 3. \tag{26}$$

The spatial distribution of the mesh parameters will be completely defined when a value for the constant C has been specified. Clearly, the total number of elements in the adapted mesh is controlled by the value of C. For smooth regions of the flow, this constant is a measure of the predicted root mean square error in the new mesh. Therefore, this constant should be decreased each time a new mesh adaption is performed. On the other hand, solutions of the Euler equations are known to exhibit discontinuities, where the error estimate will always remain large. Therefore a different strategy is needed in the vicinity of such features. In the practical implementation of the present method, two threshold values for the computed spatial distribution of spacing are used: a minimum spacing $\delta_{\min}$ and a maximum spacing $\delta_{\max}$, so that

$$\delta_{\min} \leqslant \delta_i \leqslant \delta_{\max} \qquad \text{for} \quad i = 1, 2, 3. \tag{27}$$

The maximum value $\delta_{\max}$ is introduced to account for the possibility of a vanishing eigenvalue in (26) which would render that expression meaningless. The value of $\delta_{\max}$ is chosen to be the spacing size which will be used in the regions where the flow is uniform. On the other hand, maximum values of the second derivatives occur near discontinuities (if any) in the flow, where the error indicator will then indicate that smaller elements are required. By imposing a minimum value $\delta_{\min}$ for the mesh size, we attempt to avoid an excessive concentration of elements near discontinuities. The flow algorithm is known to spread discontinuities over a fixed number of elements (i.e., two or three) and $\delta_{\min}$ is therefore set to a value that is considered appropriate to represent discontinuities to a required resolution. This treatment also improves the performance of the method in the presence of shocks of different strength. There, the numerical values of the second derivative are different and the use of Eq. (26) alone would result in the assignment of different spacings (e.g., larger spacings to weaker shocks).

The total number of elements to be generated in the new mesh will now depend on the values selected for C, $\delta_{\max}$, and $\delta_{\min}$. The values given to these constants are somewhat arbitrary. The criterion employed here is to select a value that produces a computationally affordable number of elements while reducing the estimated error.

Application of the above procedure in one dimension is illustrated in Fig. 12. The distribution of the key variable used for adaptation is assumed to be given by a certain function $\bar{\sigma}(x)$. Figure 12a shows one such function together with its second derivative. Application of the one-dimensional version of Eq. (24) can then be used in order to determine the distribution of spacing for the adapted mesh. This is shown in Fig. 12b as a dashed line. Figure 12b also shows how this distribution is modified in practice using Eq. (27) when the parameters $\delta_{\min}$ and $\delta_{\max}$ are introduced.

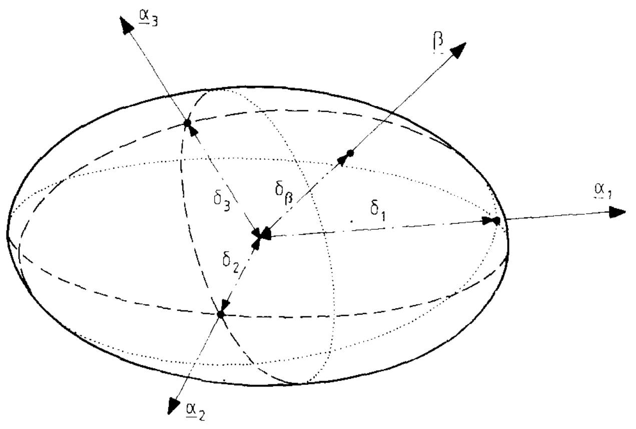


FIG. 11. Definition of the spacing $\delta\beta$ along the direction $\boldsymbol{\beta}$.

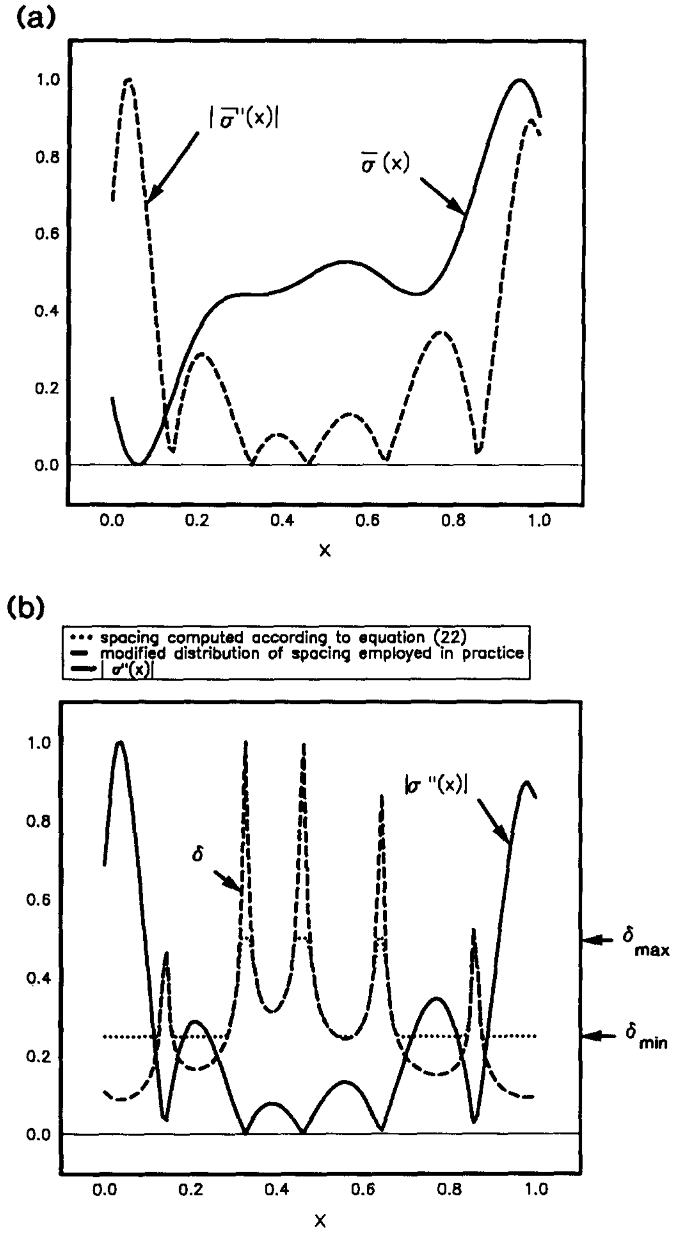


FIG. 12. Computation of the curve of distribution of mesh spacings for a one dimensional case: (a) the solution $\bar{\sigma}(x)$ and the absolute value of its second derivative $\bar{\sigma}''(x)$; (b) distribution of mesh spacing $\delta(x)$ versus $|\bar{\sigma}''(x)|$.

5.4. *Estimating the Number of Elements to Be Generated*

When regenerating a new mesh, the background mesh represents accurately the geometry of the computational domain. An estimate of the number of elements to be generated, denoted by N_e, can then be obtained as follows. For each element of the background mesh, the value of the transformation **T** is computed at the centroid. Using this transformation the volume V_e of this element in the normalised space is computed. The number of elements N_e will be directly proportional to the total volume of the domain to be meshed in the normalised space; i.e.,

$$N_e \approx \chi \sum_{e=1}^{N_b} V_e, \tag{28}$$

where N_b is the number of elements in the background mesh and χ is a constant. The value of χ is set equal to 10, which has been found to give reasonable estimates for several generated meshes. The value of N_e is then typically predicted with an error of less than 15%, which is accurate enough for most practical purposes. At this stage the value of the constant C can be adjusted until the predicted number of elements is judged to be acceptable. This determination of the value of C is accomplished using a simple bisection procedure and normally only three or four evaluations of Eq. (28) are required. The adapted mesh is now generated with the values of the mesh parameters determined by this choice of C.

6. EXAMPLE

The prediction of the flow produced in shock interactions on cylindrical leading edges is of great interest to the designers of hypersonic vehicles. In two dimensions the so-called type IV interactions [42] can lead to highly localised and intense pressures and heat transfer rates [43], which result in stress levels which could form a significant hazard to load-carrying structures. Interest is now being directed towards a study of a similar problem in three dimensions, where the cylindrical leading edge is swept, in an attempt of determine whether or not the same behaviour can be expected. Methods developed for the two-dimensional problem have been shown to be rather successful [44, 45], but no serious three-dimensional studies have been undertaken to date. Our initial attempts at performing an inviscid

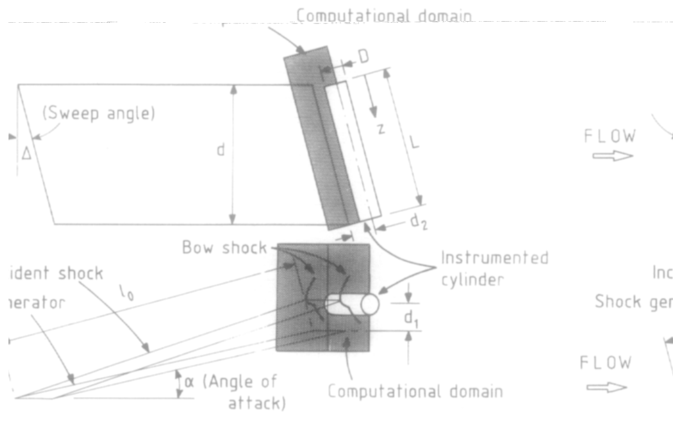


FIG. 13. Computational domain and experimental configuration for a shock interaction problem on a cylindrical leading edge at Mach 8.04.

simulation of a problem of this type are described here to illustrate the application of the adaptive remeshing process. The experimental configuration [43] and domain chosen for the computational simulation are shown diagramatically in Fig. 13. The numerical results have been produced for a sweep angle $\Delta = 15°$ a cylinder of diameter $D = 6$ and length $L = 18$. The free stream Mach number is 8.04. The fluid which has been turned by the shock generator enters the computational domain with a Mach number of 5.26. The angle of attack α of the shock generator was taken to be 12.5°. A first Euler solution was obtained on a uniform mesh of unit element size. The mesh employed and density contours of the solution are shown in Fig. 14. This solution was used in the adaptive remeshing procedure to produce a second mesh and solution which are shown in Fig. 15. Again, the adaptation process was repeated to produce the

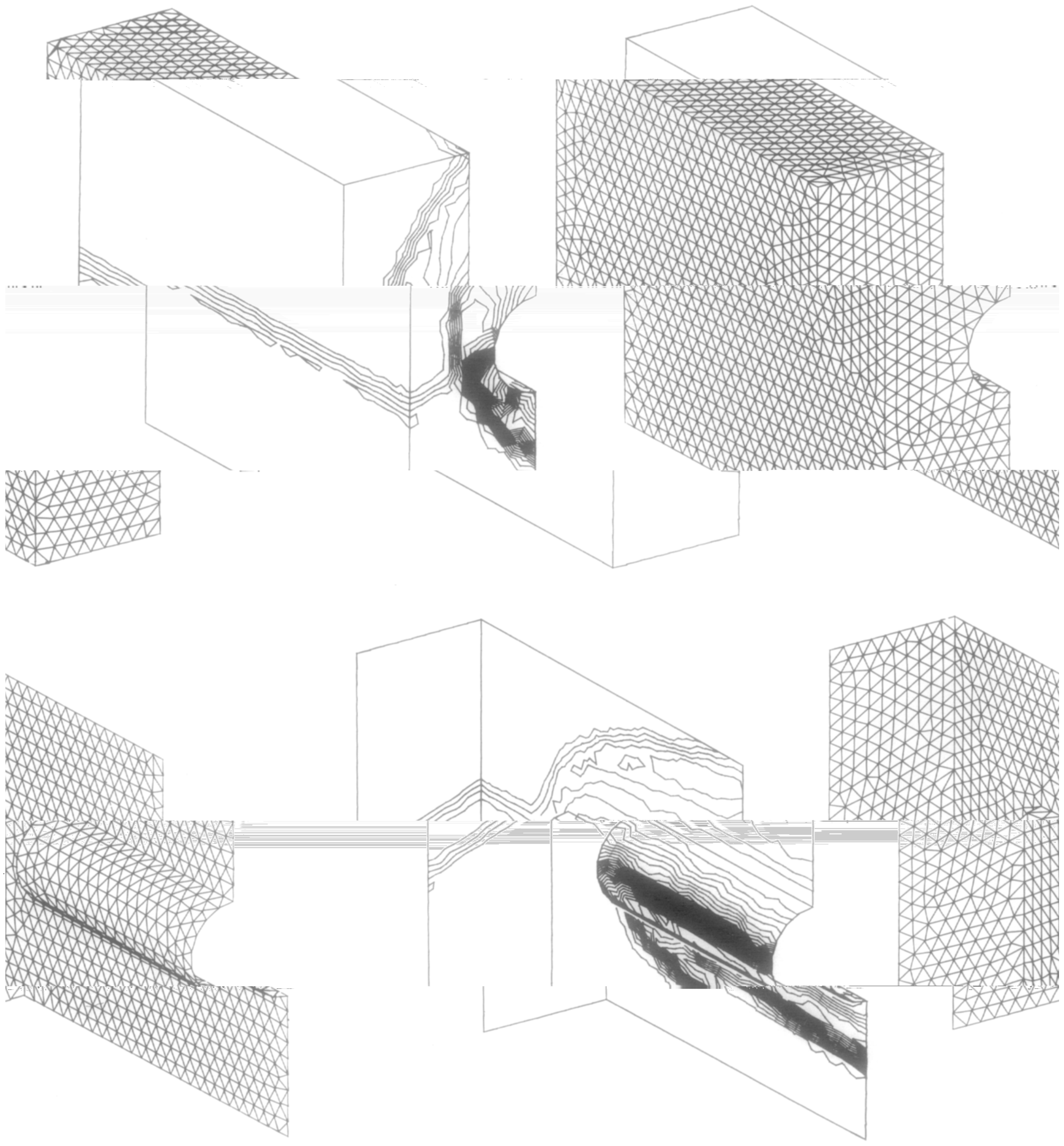


FIG. 14. First solution in the adaptive process: mesh employed and iso-density contours.

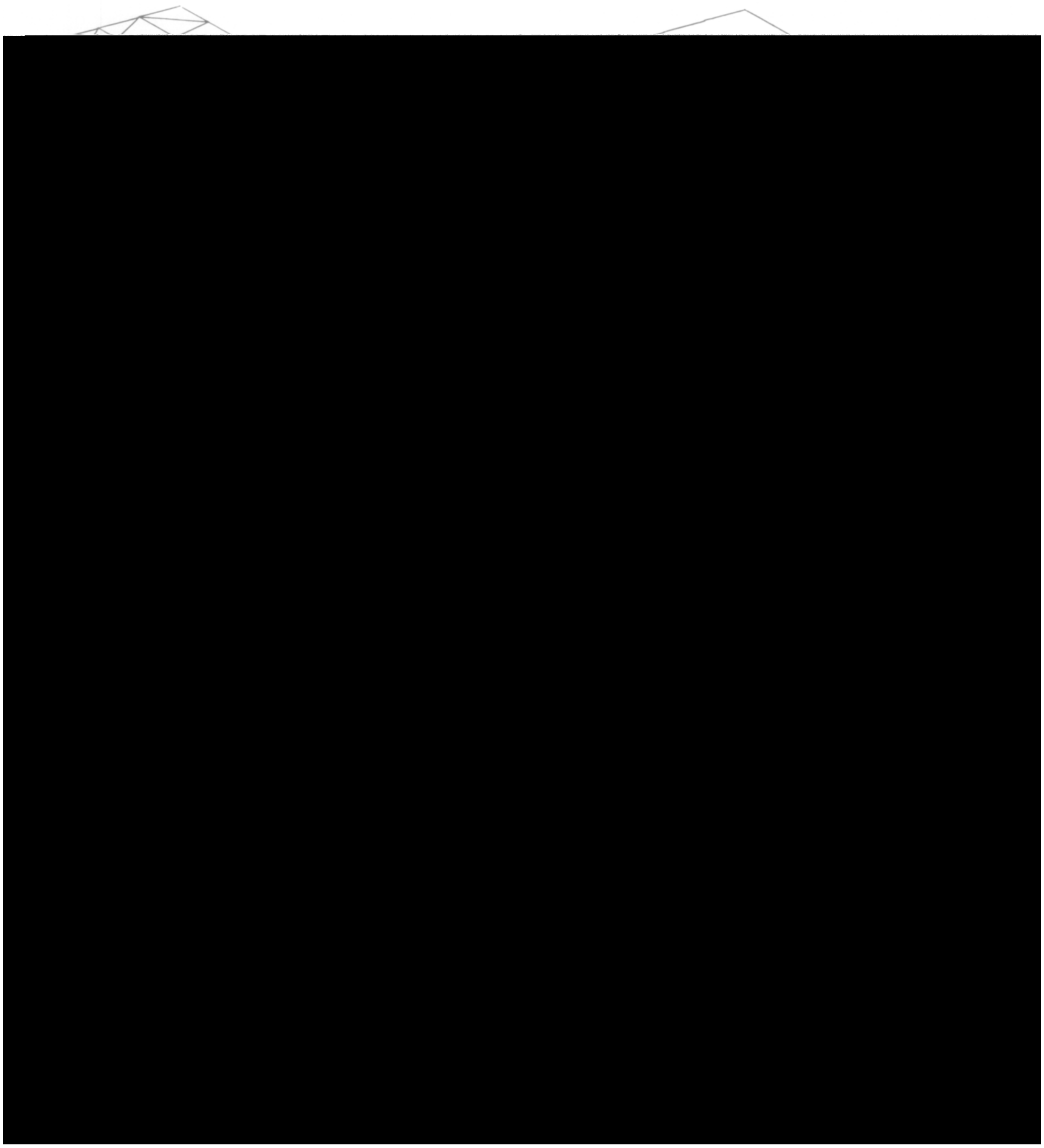


FIG. 15. Second solution in the adaptive process: mesh employed and iso-density contours.

final mesh and solution which are depicted in Fig. 16. Even though these results are still far from being mesh independent, the improvement in the solution quality during the adaptivity process can be noted by examining, in Fig. 17, the comparison between the results of experiment and the values of the computed pressure coefficient on the cylinder surface in a cross section located halfway along the cylinder.

TABLE I

Mesh	Elements	Points	Boundary	$\delta_{\min}$	$\delta_{\max}$	C	N_e
1	51190	10041	4976	1.0	1.0	—	55775
2	100071	18660	6004	0.5	3.0	0.85	100836
3	171800	31083	7406	0.18	3.0	0.8	168420

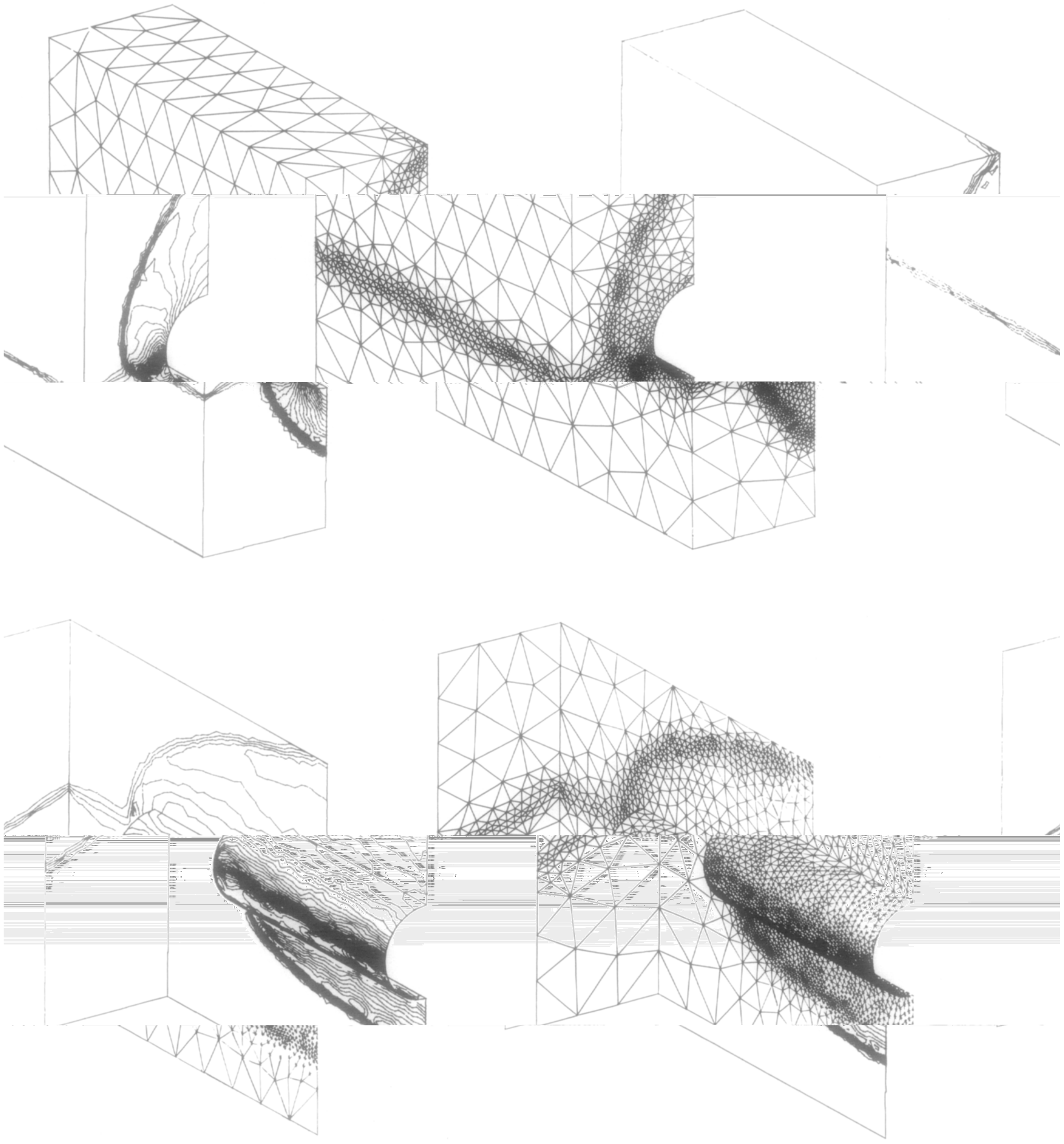


FIG. 16. Third solution in the adaptive process: mesh employed and iso-density contours.

The characteristics of each mesh are summarised in Table I which shows the number of tetrahedral elements, nodal points, number of triangles on the boundary of the domain, and the values of the generation parameters defined above. A more detailed view of the shock interaction is given in Fig. 18, which displays the computed density contours on the final mesh on three planes normal to the axis of the cylinder. Finally, an idea of the mesh adaptation occurring in the interior of the three-dimensional domain can be gained by examining in Fig. 19 the intersection of the three meshes with a plane which is normal to the cylinder axis, halfway along the cylinder.

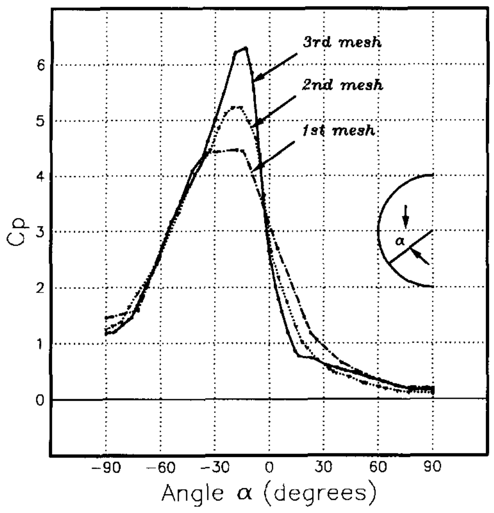


FIG. 17. Computed pressure coefficient on the cylinder surface in a cross section located halfway along the cylinder.

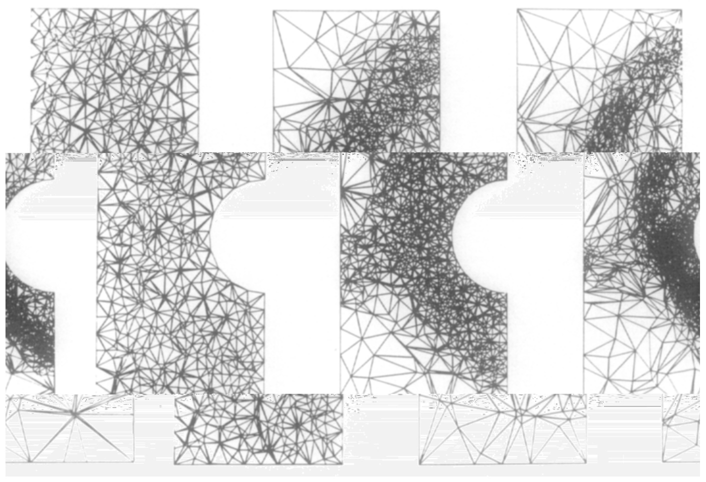


FIG. 19. Halfway cross sections normal to the cylinder axis of the three meshes employed.

These computations were performed on a Cray 2 and it is of interest to give an indication of the computational requirements of the proposed procedure. The meshes were generated at a rate of 6500 elements per minute, which means that the final mesh employed was generated in 26 min. The flow solution was deemed to be converged on this mesh after 1200 explicit timesteps and this required 31 min.

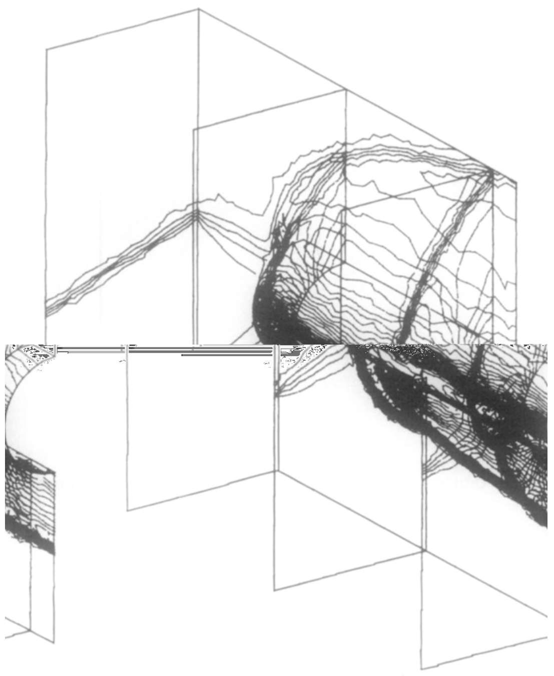


FIG. 18. View of the iso-density contours for the final mesh on the cylinder surface and three planes normal to the cylinder axis.

7. CONCLUSIONS

A complete procedure for the adaptive solution of the compressible Euler equations in general three-dimensional regions has been presented. The flexibility of the method means that adapted solutions of enhanced accuracy can be systematically and rapidly produced. In the example presented, the adaptivity decreases the minimum mesh spacing by a factor of five, while the number of degrees of freedom is increased by less than a factor of 3.5. Based upon our previous experiences [46] with alternative adaptive strategies, such as classical or directional enrichment, this method certainly appears to make better use of available degrees of freedom.

One important area of the approach which needs further improvement is the error estimating procedure. It is apparent from the computations reported here that the use of the techniques which are currently available can lead to a viable method of solution for a certain class of problems. However, the development of a rigorous error estimating theory is regarded as an essential prerequisite to the routine use of adaptive mesh techniques in the industrial environment.

ACKNOWLEDGMENTS

We acknowledge the partial support for this work provided by Dr. A. R. Wieting, Aerothermal Loads Branch, NASA Langley Research Center, under Research Contracts NAGW-478 and NAGW-1809, and by Avions Marcel Dassault Aviation under the HERMES program.

REFERENCES

1. J. F. Dannenhoffer and J. R. Baron, AIAA Paper 84-0005, 1984 (unpublished).

2. J. T. Oden, T. Strouboulis, and Ph. Devloo, *Finite Elements in Fluids*, Vol. 7 (Wiley, Chichester, 1988), p. 223.
3. R. Löhner, K. Morgan, J. Peraire, and O. C. Zienkiewicz, AIAA Paper 85-1531-CP, 1985 (unpublished).
4. B. Stoufflet, J. Periaux, F. Fezoui, and A. Dervieux, AIAA Paper 87-0560, 1987 (unpublished).
5. D. J. Mavriplis, ICASE Report 88-47, 1988 (unpublished).
6. J. Peraire, K. Morgan, J. Peiró, and O. C. Zienkiewicz, AIAA Paper 87-0559, 1987 (unpublished).
7. P. A. Gnoffo, *AIAA J.* **21**, 1249 (1983).
8. K. Nakahashi and G. S. Deiwert, AIAA Paper 85-1225, 1985 (unpublished).
9. J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *J. Comput. Phys.* **72**, 449 (1987).
10. R. Löhner and P. Parikh, *Int. J. Numer. Methods Fluids* **8**, 1135 (1988).
11. G. L. D. Siden, W. N. Dawes, and P. J. Albraten, ASME Paper 89-GT-211, 1989 (unpublished).
12. P. Hansbo, Ph.D. thesis, Dept. Structural Mechanics, Chalmers University of Technology, Sweden, 1989 (unpublished).
13. L. Formaggia, J. Peraire, and K. Morgan, *Appl. Math. Modelling* **12**, 175 (1988).
14. R. Löhner, AIAA Paper 88-3737, 1988 (unpublished).
15. J. Probert, O. Hassan, J. Peraire, and K. Morgan, in *Proceedings, 5th Int. Symp. Numer. Methods Eng.* (Springer-Verlag, Berlin, 1989), p. 801.
16. O. C. Zienkiewicz and J. Z. Zhu, *Int. J. Numer. Methods Eng.* **24**, 337 (1987).
17. M. Pastor, J. Peraire, and O. C. Zienkiewicz, Adaptive remeshing for shear band localisation problems, *Ing. Arch.* **61**, 30 (1991).
18. H. Jin and N.-E. Wiberg, Dept. of Structural Mechanics Report 89:3, Chalmers University of Technology, Sweden, 1989 (unpublished).
19. W. Atamaz-Sibai and E. Hinton, in *Proceedings, 3rd NUMETA Conf.*, edited by G. N. Pande and J. Middleton (Balkema, Amsterdam, 1990), p. 1044.
20. H. C. Huang and R. W. Lewis, in *Proceedings, 6th. Int. Conf. Numer. Methods Thermal Problems*, edited by R. W. Lewis and K. Morgan (Pineridge, Swansea, UK, 1989), p. 1029.
21. I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture* (Ellis Horwood, Chichester, 1981).
22. J. C. Ferguson, *J. Assoc. Comput. Mach.* **11**, 221 (1964).
23. S. A. Coons, Report MAC-TR-41, Project MAC, MIT, 1967 (unpublished).
24. J. Peiró, Ph.D. thesis, University of Wales, UK, 1989 (unpublished).
25. A. J. George, Ph.D. thesis, Stanford University, STAN-CS-71-208, 1971 (unpublished).
26. S. H. Lo, *Int. J. Numer. Methods Eng.* **21**, 1403 (1985).
27. T. J. Baker, in *AGARD Conference Proceedings*, No. 464 (AGARD, Paris, 1989), p. 20-1.
28. J. C. Cavendish, D. A. Field, and W. H. Frey, *Int. J. Numer. Methods Eng.* **21**, 329 (1985).
29. J. J. Stoker, *Differential Geometry* (Wiley Interscience, New York, 1969).
30. J. Bonet and J. Peraire, An alternating digital tree (ADT) algorithm for geometric searching and intersection problems, *Int. J. Numer. Methods Eng.* **31**, 1 (1991).
31. D. E. Knuth, *The Art of Computer Programming. Vol.* 1. *Fundamental Algorithms* (Addison–Wesley, Reading, MA, 1969).
32. R. Löhner, K. Morgan, J. Peraire, and O. C. Zienkiewicz, AIAA Paper 85-1531-CP, 1985 (unpublished).
33. J. Peraire, L. Formaggia, J. Peiro, K. Morgan, and O. C. Zienkiewicz, *Int. J. Numer. Methods Eng.* **26**, 2135 (1988).
34. J. Peraire, K. Morgan, and J. Peiró, in *AGARD Conference Proceedings*, No. 464, (AGARD, Paris, 1989), p. 18-1.
35. L. Formaggia, J. Peraire, K. Morgan, and J. Peiró, in *Proceedings, 4th International Symposium on Science and Engineering on Cray Supercomputers*, (CRAY Research Inc., Minneapolis, 1988), p. 45.
36. R. R. Thareja, J. R. Stewart, O. Hassan, K. Morgan, and J. Peraire, *Int. J. Numer. Methods Fluids* **9**, 405 (1989).
37. M. Vahdati, K. Morgan, J. Peraire, and O. Hassan, in *Proceedings, R. Aeronaut. Soc. Int. Conf. on Hypersonic Aerodynamics* (Royal Aeronautical Society, London, 1989), p. 12.1.
38. P. G. Ciarlet, *The Finite Element Method for Elliptic Problems* (North Holland, Amsterdam, 1978).
39. J. T. Oden, "Grid Optimisation and Adaptive Meshes for Finite Element Methods," University of Texas at Austin Notes, 1983 (unpublished).
40. O. C. Zienkiewicz and K. Morgan, *Finite Elements and Approximation* (Wiley, New York, 1983).
41. J. Donea, S. Giuliani, H. Laval, and L. Quartapelle, *Comput. Methods Appl. Mech. Eng.* **45**, 123 (1984).
42. B. Edney, FFA Report 115, Aeronautical Research Institute of Sweden, 1968 (unpublished).
43. A. R. Wieting, NASA TM 100484, 1987 (unpublished).
44. G. H. Klopfer and H. C. Yee, AIAA Paper 88-0233, 1988 (unpublished).
45. R. R. Thareja, J. R. Stewart, O. Hassan, K. Morgan, and J. Peraire, AIAA Paper 88-0036, 1988 (unpublished).
46. J. Peraire, K. Morgan, and J. Peiró, "Unstructured Mesh Methods for CFD," von Karman Institute for Fluid Dynamics Lecture Series 1990-06, 1990 (unpublished).