Pergamon

Engineering Fracture Mechanics Vol. 50, No. 5/6, pp. 819–847, 1995

0013-7944(94)E0062-L
0013-7944/95 $9.50 + 0.00

MESH ADAPTATION IN FLUID MECHANICS

RAINALD LÖHNER
CMEE, School of Engineering and Applied Science, The George Washington University, Washington, DC 20052, U.S.A.

Abstract—The development, application and impact of mesh adaptation procedures in the field of computational fluid mechanics (CFD) are reviewed. The discussion is restricted to unstructured (i.e. unordered) grids, such as those commonly encountered in finite element applications.

1. INTRODUCTION

THE DEVELOPMENT of self-adaptive mesh refinement techniques in computational fluid dynamics (CFD) and computational structural mechanics (CSM) was motivated by the same reasons:

(a) With mesh adaptation, the numerical solution to a specific problem, given the basic accuracy of the solver and the desired accuracy, should be achieved with the least number of degrees of freedom. This, in most cases, translates into the least amount of work for a given accuracy.
(b) With mesh adaptation, the user must no longer waste time choosing a grid that is suitable for the problem at hand; although this may seem unimportant for repetitive steady-state calculations, it is of the utmost importance for transient problems with traveling discontinuities (shocks, plastic fronts, etc.). In this way, adaptation adds a new dimension of user-friendliness to computational mechanics.

Given these very strong motivating reasons, the last decade has seen a tremendous surge in activity in this area. It is interesting to note that mesh adaptation in CFD and CSD both appeared in the early 1980s. In fact, the first conferences dedicated solely to this subject took place around 1984 [1, 2]. The present review focuses primarily on unstructured (i.e. unordered) grids, such as those commonly encountered in finite element applications. However, parallel developments in the area of structured grids (finite differences and finite volume techniques) are mentioned wherever appropriate.

Any adaptive refinement scheme is composed of three main ingredients:

an optimal-mesh criterion,
an error indicator, and
an algorithm or strategy to refine and coarsen the mesh.

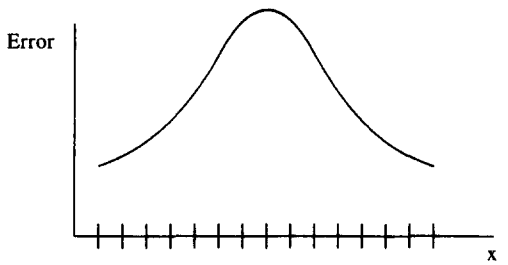
They give answers to the questions:

How should the optimal mesh be?
Where is refinement/coarsening required?
How should the refinement/coarsening be accomplished?

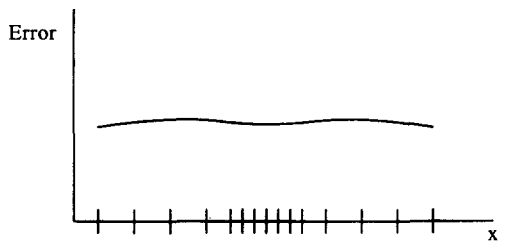
The topic of adaptation now being a decade old, it is not surprising that a variety of answers have been proposed by several authors for each of these questions. In the following, the most successful ones are discussed in more depth.

2. OPTIMAL MESH CRITERIA

Before designing an adaptive mesh procedure, the analyst should have a clear idea of what is to be achieved. Reduction of manual and computational work is the obvious answer, but in order to be more definite, a quantitative assessment of the optimality of the adaptive mesh procedure is necessary. This leads to the immediate question: What should the optimal mesh be like? The



(a) Before adaptation



(b) After adaptation

Fig. 1. Error distribution before and after refinement.

answer to this crucial question is seldom clear, as engineers do no always know *a priori* what constitutes a sufficiently accurate answer to the problem at hand. For example, if all that is requried is the lift of an airfoil (an integrated, global quantity), there is in most cases no need for an extremely accurate solution away from the surface. A number of researchers have followed guidelines with mathematical foundations, which are outlined below.

2.1. *Equidistribution of error*

The aim is to attain a grid in which the error is uniformly distributed in space. One can show that such a mesh has the smallest number of degrees of freedom (i.e. the smallest number of elements) for the general aim:

$$c^h \rightarrow \min \forall \mathbf{x} \in \Omega. \tag{1}$$

Conceptually, one can derive this criterion from the observation that the error will have the irregular distribution for the first mesh shown in Fig. 1a. If the number of degrees of freedom is kept the same, the distribution of element size and shape is all that may be varied. After repositioning of points, the error distribution in space will become more regular, as shown in Fig. 1b. One can also see that the general aim stated in eq. (1) will be achieved when the error is constant in the domain. This mesh optimality criterion is most often used in conjunction with steady-state problems.

2.2. *Local absolute error tolerances*

In many applications, the required error tolerances may not be the same at all locations. Moreover, instead of using the general minimization stated in eq. (1), one may desire to enforce absolute local bounds in certain regions of the domain:

$$c^h < c_l \; \forall \mathbf{x} \in \Omega_{\text{sub}}. \tag{2}$$

Mesh refinement or coarsening would then take place if the local error indicator exceeds or falls below given refinement or coarsening tolerances:

$$c^h > c_r \Rightarrow \text{refine}, \quad c^h < c_c \Rightarrow \text{coarsen}.$$

The calculation of skin-friction coefficients of airfoils is a possible application for such a mesh optimality criterion. The mesh optimality criterion based on local absolute error tolerance is most often used for transient problems.

3. ERROR INDICATORS/ESTIMATORS

Consider the task of trying to determine if the solution obtained on the present mesh and discretization is accurate. Intuitively, a number of criteria immediately come to mind: variations of key variables within elements, entropy levels, higher-order derivatives of the solutions, etc. All of them make the fundamental assumption that the solution on the present mesh is already in some form close to the exact solution. This assumption is reasonable for parabolic and elliptic problems, where, due to global minimization principles, local deficiences in the mesh only have a local effect. For hyperbolic problems, the assumption $u^h \approx u$ may be completely erroneous. Consider an airfoil at a high angle of attack. A coarse initial mesh may completely miss local separation bubbles at the leading edge that lead to massive separation in the back portion of the airfoil. Although the local error in the separation region may be very small, the global error in the field would be very large due to different physical behavior. *Any* error indicator presently in use could miss these features for bifurcation point regions, performing adaptation at the wrong places. On the other hand, the assumption $u^h \approx u$ is a very reasonable one for most initial grids and stable physics. As a matter of fact, it is not so difficult to attain, given that the nature of most engineering applications is such that: (a) developments are evolutionary, rather than revolutionary, i.e. a similar problem has been solved before (the airfoil being the extreme example), and (b) the controlability requirement of reliable products implies operation in a stable regime of the physics.

3.1. *Popular error indicators*

The most popular error indicators presently used in production codes may be grouped into the following categories.

3.1.1. *Jumps in indicator variables.* The simplest error indicator is obtained by evaluating the jump (i.e. the undivided difference) of some indicator variable like the Mach number, density or entropy within an element or along an edge. This error indicator implicitly makes the assumption

$$\epsilon_{el}^h = c_1 h |\nabla u|, \tag{3}$$

i.e. first order accuracy for the underlying scheme. Error indicators of this form have been used in industrial applications with success [3–9], even if the underlying numerical discretization was of higher than first order.

3.1.2. *Interpolation theory.* Making the assumption that the solution is smooth, one may approximate the error in the elements by a derivative one order higher than the element shape function. For 1D, this would result in the error indicator at the element level of the form

$$\epsilon_{el}^h = c_1 h^p \left| \frac{\partial^p u}{\partial x^p} \right|, \tag{4}$$

where the pth derivative is obtained by some recovery procedure, and for linear elements $p = 2$. The total error in the computational domain is then given by

$$\epsilon_\Omega^h = c_2 \left[\int h^p \left(\frac{\partial^p u}{\partial x^p} \right)^2 d\Omega \right]^{1/2}. \tag{5}$$

This error indicator gives superior results for smooth regions. On the other hand, at discontinuities the local value of ϵ_{el}^h will stay the same no matter how fine the mesh is made.

3.1.3. *Comparison of derivatives.* Again making the assumption that the solution is smooth, one may compare significant derivatives using schemes of different order. As an example, consider the following three approximations to a second derivative:

$$u_{,xx}|_{4_1} = \frac{1}{h^2}(u_{i-1} - 2u_i + u_{i+1}) - \tfrac{1}{12} h^2 u_{,IV}, \tag{6a}$$

$$u_{,xx}|_{4_2} = \frac{1}{4h^2}(u_{i-2} - 2_{u_i} + u_{i+2}) - \tfrac{1}{12}4h^2 u_{,\mathrm{IV}}, \tag{6b}$$

$$u_{,xx}|_6 = \frac{1}{12h^2}(-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}) + \tfrac{1}{90}h^4 u_{,\mathrm{VI}}. \tag{6c}$$

The assumption of smoothness in u would allow a good estimate of the error in the second derivatives from the difference of these three expressions [10]. Moreover, comparing eq. (6a) to eqs (6b, c) can give an indication as to whether it is more efficient to h-refine the mesh (reduction of h), or to increase the order of accuracy for the stencil (p-refinement). For unstructured grids, one may recover these derivatives with reconstruction procedures.

3.1.4. *Residuals of PDEs on adjacent grids.* Assume we have a node-centered scheme to discretize the PDEs at hand. At steady state, the residuals at the nodes will vanish. On the other hand, if the residuals are evaluated at the element level, non-vanishing residuals are observed in the regions that require further refinement. This error indicator has been used extensively in France [11]. Another possibility is to check locally the effect of higher order shape functions introduced at the element level or at element boundaries [12]. These so-called p-finement indicators have been used extensively for structural FEM applications [1, 13, 14].

3.1.5. *Energy norms.* Assume incompressible creeping flow. For this viscous-dominated flow the solution consists of a velocity field $\mathbf{v}$ with zero divergence that minimizes the dissipation energy functional

$$J(\mathbf{v}) = \int \mu(v^i_{,j} + v^j_{,i}) : (v^i_{,j} + v^j_{,i})\, \mathrm{d}\Omega = \int \sigma : \epsilon \, \mathrm{d}\Omega. \tag{7}$$

Here $\mathbf{v}$ denotes the velocity field, σ the viscous stress tensor and ϵ the strain rate tensor. The dissipation energy of the error $\mathbf{e} = \mathbf{v} - \mathbf{v}_h$ satisfies the relation

$$J(\mathbf{e}) = \int (\sigma_h - \sigma) : (\epsilon_h - \epsilon)\, \mathrm{d}\Omega. \tag{8}$$

The unknown exact stresses and strain rates σ, ϵ can be recovered at nodes through a least-squares projection. The final estimator, using recovered stresses σ_r and the strain rates ϵ_r, then becomes

$$J(\mathbf{e}) = \int (\sigma_h - \sigma_r) : (\epsilon_h - \epsilon_r)\, \mathrm{d}\Omega. \tag{9}$$

This error estimator, although derived for creeping flows, has also been shown to be useful for flows with moderate Reynolds number [15]. One can also see that this error indicator is closely related to the Zienkiewicz–Zhu error indicator [16–18].

3.1.6. *Energy of spatial modes.* For higher-order methods, such as spectral element methods, a very elegant way to measure errors and convergence is to separate the energy contents associated with the different shape functions. The decrease of energy contained in the higher-order shape functions gives a reliable measure of convergence [19, 20]. At the same time, this way of measuring errors provides an immediate strategy as to when to perform h-refinement (slow decrease of energy content with increasing shape function polynomial) or p-refinement (rapid decrease of energy content with increasing shape function polynomial).

3.1.7. *Other error indicators/estimators.* In a field that is developing so rapidly, it is not surprising that a variety of other error indicators and estimators have been proposed. The more theoretically inclined reader may wish to consult refs [21–23].

3.2. *Problems with multiple scales*

All of these error indicators have been used in practice to guide mesh adaptation procedures. They all work well for their respective area of application. However, they cannot be considered as generally applicable for problems with multiple intensity and/or length scales. None of them are dimensionless, implying that strong features (e.g. strong shocks) produce large error indicator values, whereas weak features (secondary shocks, contact discontinuities, shear layers) produce small ones. Thus, in the end, only the strong features of the flow would be refined, losing the weak ones. A number of ways have been proposed to circumvent this shortcoming.

3.2.1. *Stopping criteria.* The most common way to alleviate the problems encountered with multiple intensity/length scales is to introduce a stopping criterion for refinement. This can be a minimum element size or, for the case of h-refinement, a maximum level of subdivision. After the discretization is sufficiently fine enough to activate the stopping criterion, the remainder of the added degrees of freedom are introduced in regions with weaker features.

3.2.2. *Two-pass strategy.* A second option, proposed by Aftosmis [8, 24], is to separate the disparate intensity or length scales into two (or possibly several) passes over the mesh. In the first pass, the strong features of the flow are considered, and appropriate action is taken. A second pass is performed for the elements not marked in the first pass, identifying weak features of the flow, and appropriate action is taken. This procedure has worked well for viscous flows with shocks [8, 24], and can be combined with any of the error indicators described above.

3.2.3. *Non-dimensional error indicators.* An error indicator that allows even refinement across a variety of intensity and length scales was proposed in ref. [25]. In general terms, it is of the form

$$\text{error} = \frac{h^2|\text{second derivatives}|}{h|\text{first derivatives}| + c_n|\text{mean value}|}. \tag{10}$$

By dividing the second derivatives by the absolute value of the first derivatives, the error indicator becomes bounded, dimensionless and the "eating up" effect of strong features is avoided. The terms following ϵ are added as a noise filter in order not to refine wiggles or ripples which may appear due to loss of monotonicity. The value for ϵ thus depends on the algorithm chosen to solve the PDEs describing the physical process at hand. The multidimensional form of this error indicator is given by

$$E^I = \sqrt{\left[\frac{\sum_{k,l}\left(\int_\Omega N^I_{,k} N^J_{,l}\,\mathrm{d}\Omega \cdot U_J\right)^2}{\sum_{k,l}\left(\int_\Omega |N^I_{,k}|[|N^J_{,p} U_J| + c_n(|N^J_{,l}||U_J|)]\,\mathrm{d}\Omega\right)^2}\right]}, \tag{11}$$

where N^I denotes the shape function of node I. The fact that this error indicator is dimensionless allows the simultaneous use of several indicator variables. Because the error indicator is bounded ($0 \leqslant E^I \leqslant 1$), it can be used for whole classes of problems without having to be scaled to the problem at hand. This results in an important increase in user-friendliness, allowing non-expert users access to automatic self-adaptive procedures. This error indicator has been used successfully for many years on a variety of applications [25–38].

3.3. *Determination of element size and shape*

Having measured the error in the present solution, or at least having identified the regions of the computational domain that require further refinement or coarsening, the next question to be answered is the magnitude of mesh change required. This question is of minor importance for h-refinement or p-refinement, where the grid is simply subdivided further by factors of two in space or by adding the next higher degree polynomial to the space of available shape functions. On the other hand, for adaptive mesh movement or adaptive remeshing, where the element size and shape vary smoothly, it is necessary to obtain a more precise estimation of the required element size and shape. How to achieve this will be exemplified for the non-dimensional error indicator given by eq. (11). It is a simple matter to perform a similar analysis for all the other error indicators described. Defining the derivatives according to order as:

$$D_i^0 = c_n(|U_{i+1}| + 2|U_i| + |U_{i-1}|) \tag{12a}$$

$$D_i^1 = |U_{i+1} - U_i| + |U_i - U_{i-1}| \tag{12b}$$

$$D_i^2 = |U_{i+1} - 2U_i + U_{i-1}|, \tag{12c}$$

the error indicator on the present (old) grid E^{old} is given by:

$$E_i^{\text{old}} = \frac{D_i^2}{D_i^1 + D_i^0}. \tag{13}$$

The reduction of the current element size h^{old} by a fraction ξ to $h^{\text{new}} = \xi h^{\text{old}}$ will yield a new error indicator of the form

$$E_i^{\text{new}} = \frac{D_i^2 \xi^2}{D_i^1 \xi + D_i^0}. \tag{14}$$

Given the desired error indicator value E^{new} for the improved mesh, the reduction factor ξ is given by:

$$\xi = \frac{E^{\text{new}}}{E^{\text{old}}} \frac{1}{2} \left[\frac{D_i^1 + \sqrt{\left\{ (D_i^1)^2 + 4 D_i^0 \frac{E^{\text{old}}}{E^{\text{new}}} [D_i^1 + D_i^0] \right\}}}{[D_i^1 + D_i^0]} \right]. \tag{15}$$

Observe that for a smooth solution with $D^1 \ll D^0$, this results in $\xi = (E^{\text{new}}/E^{\text{old}})^{0.5}$, consistent with the second order accuracy of linear elements. Closer to discontinuities $D^1 \gg D^0$, and one obtains $\xi = E^{\text{new}}/E^{\text{old}}$, consistent with the first order error obtained in these regions.

This error indicator can be generalized to multidimensional situations by defining the following tensors:

$$(D^0)_{kl}^I = h^2 c_n \int_\Omega |N_{,k}^I| |N_{,l}^J| |U_J| \, \mathrm{d}\Omega, \tag{16}$$

$$(D^1)_{kl}^I = h^2 \int_\Omega |N_{,k}^I| |N_{,l}^J U_J| \, \mathrm{d}\Omega, \quad (D^2)_{kl}^I = h^2 \left| \int_\Omega N_{,k}^I N_{,l}^J \, \mathrm{d}\Omega U_J \right|, \tag{17}$$

which yield an error matrix $\mathbf{E}$ of the form:

$$\mathbf{E} = \begin{Bmatrix} E_{xx} & E_{yx} & E_{zx} \\ E_{xy} & E_{yy} & E_{zy} \\ E_{xz} & E_{yz} & E_{zz} \end{Bmatrix} = \mathbf{X} \cdot \begin{Bmatrix} E_{11} & 0 & 0 \\ 0 & E_{22} & 0 \\ 0 & 0 & E_{33} \end{Bmatrix} \cdot \mathbf{X}^{-1}. \tag{18}$$

The principle eigenvalues of this matrix are then used to obtain reduction parameters ξ_{jj} in the three associated eigenvector directions. Due to the symmetry of $\mathbf{E}$, this is an orthogonal system of eigenvectors that defines a local coordinate system.

We remark that the variation in element size required to meet a certain tolerance can be determined with any of the error indicators enumerated in Section 3.1. However, the variation in stretching, i.e. the shape of the elements for the adapted 3D mesh, requires an error indicator that is based on a tensor of second derivatives.

4. REFINEMENT STRATEGIES

Besides the mesh optimality criterion and the error indicator/estimator, the third ingredient of any adaptive refinement method is the refinement strategy, i.e. *how to refine* a given mesh. Three different families of refinement strategies have been considered to date.

4.1. *Mesh movement or repositioning (* r*-methods)*

The aim is to reposition the points in the field in order to obtain a better discretization for the problem at hand. The regions that require more elements tend to draw points and elements from regions where a coarser mesh can be tolerated. Three basic approaches have been used to date:

(a) The Moving Finite Element Method, where the position of points is viewed as a further unknown in a general functional to be minimized [39].
(b) Spring Systems, whereby the mesh is viewed as a system of springs whose stiffness is proportional to the error indicators [3, 11, 40–43].
(c) Optimization Methods, whereby the position of points is changed in order to minimize a functional [44, 45].

Mesh movement schemes are relatively simple to code, as the mesh topology is not allowed to change. They have the desirable property of aligning elements with features of lower dimensionality than the problem at hand. This stretching of elements can lead to considerable savings as compared to other methods. On the other hand, they are not flexible and general enough for production runs

that may exhibit complex physics. They are presently used mainly in conjunction with Finite Difference codes [41–43, 45, 46], where by the very nature of the method no topology changes are allowed. For unstructured grid codes, mesh movement has only been tried in academia or in conjunction with other methods [3] (see Section 4.4).

4.2. *Mesh enrichment (*h/p*-methods)*

In this case, degrees of freedom are added or taken from a mesh according to some rule. One may either split elements into new ones (h-refinement), or add further degrees of freedom with hierarchical shape functions (Fig. 2). The same may be accomplished with the addition of higher order shape functions (p-refinement), again either conventional polynomials [2], spectral [19, 20] functions, or hierarchical shape-functions [13]. For elliptic systems of PDEs, the combination of h- and p-refinement leads to exponential convergence rates [2, 47]. p-Refinement methods have so far not been used extensively in fluid mechanics. The author is not aware of any production or commercial CFD code that has a built-in p-refinement capability. Possible reasons for this lack of success, which stands in contrast to CSM applications [2, 14, 48, 49], are the following.

(a) The limited accuracy that is achievable for CFD applications due to monotonicity enforcement close to discontinuities [50], and the lack of accurate turbulence models [51, 52]. Indeed, a CFD solution that claims an accuracy better than 1% for a complex flow problem has to be considered with great scepticism. On the other hand, the major gains of high-order methods as compared to h-refinement in combination with low-order methods are in the range below 1% relative error.

(b) The desired accuracy of engineering applications, which seldom fall below 1% relative error. Given the uncertainties in boundary conditions, material parameters and source terms of typical engineering applications, 1% relative error can already be considered unnecessarily accurate.

(c) The much higher coding complexity of p-methods or h/p-methods as compared to straightforward h-methods. The only possible difficulty of h-methods is given by hanging nodes for quad- or brick-type elements (for triangles and tetrahedra even this problem disappears). Apart from this relatively small modification, the original one element-type flow solver can remain unchanged. On the other hand, any p-method requires the development and maintenance of a complete library for all the different types of elements. Further complications arise from the adaptation logic when h/p-type refinements are considered.

This is not to say that we may not see much activity in this area: boundary layers, shear layers, flames and other features that are currently being computed using under-resolved Navier–Stokes

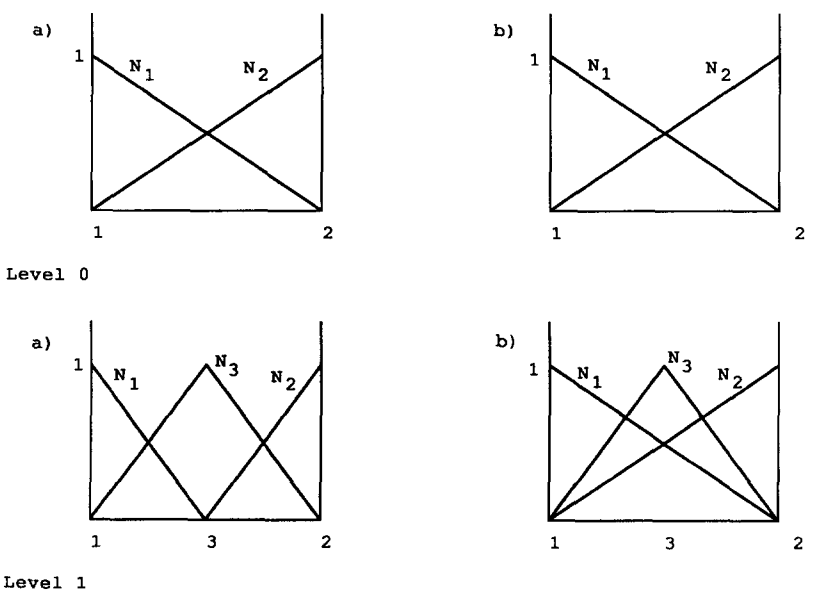


Fig. 2. h-Refinement using usual polynomials or hierarchical shape functions.

runs are, when properly resolved, smooth features that should be ideally suited to high-order discretizations.

4.2.1. h-*Enrichment*. By far the most successful mesh enrichment strategy has been *h*-enrichment. There are several reasons that can be given for this success:

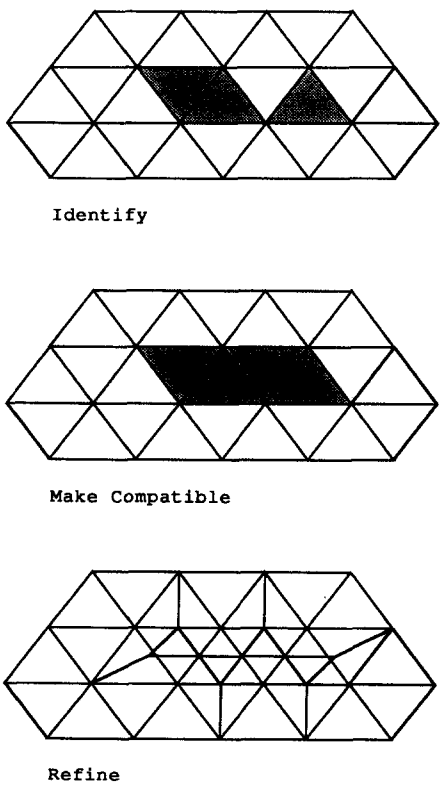
- Conservation is maintained naturally with *h*-refinement.
- No interpolations other than the ones naturally given by the element shape functions are required. Therefore, no numerical diffusion is introduced by the adaptive refinement procedure. This is in contrast to adaptive remeshing, where the grids before and after a mesh change may not have the same points in common. The required interpolations of the unknowns will result in an increased amount of numerical diffusion (see refs [28, 30]).
- *h*-Refinement is very well suited to vector and parallel processors. This is of particular importance for transient problems, where a mesh change is performed every 5–10 timesteps, and a large percentage of mesh points is affected in each mesh change [25, 30].

The three main variants used to date achieve mesh enrichment/coarsening through (see Fig. 3):

(a) classic subdivision of elements into four (2D) or eight (3D) after compatibility tests;
(b) recursive subdivision of the largest edge side with subsequent compatibility tests;
(c) agglomeration of cells with blocked subdivisions after compatibility tests.

The compatibility tests are necessary to ensure that an orderly transition occurs between fine and coarse mesh regions, and to avoid hanging nodes for triangular and tetrahedral elements.

4.2.1.1. *Classic* h-*refinement*. This simplest form of mesh refinement/coarsening seeks to achieve higher mesh densities by subdividing elements by four (2D) or eight (3D). In order to have a smooth transition between refined and unrefined mesh regions, compatibility tests are required. For



a) Classic H-Refinement

Fig. 3(a)

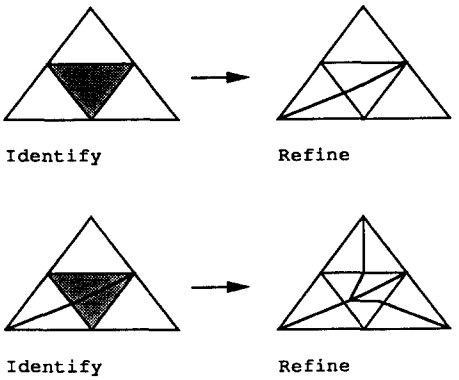


Fig. 3 (b and c)

Fig. 3. Possible h-refinement strategies.

triangles and tetrahedra, badly formed elements due to refinement are avoided by restricting the possible refinement patterns. For tetrahedra, subdivision is only allowed into two (along a side), four (along a face) or eight. These cases are denoted as 1:2, 1:4 and 1:8, respectively. At the same time, a 1:2 or 1:4 tetrahedron can only be refined further to a 1:4 tetrahedron, or by first going back to a 1:8 tetrahedron with subsequent further refinement of the eight sub-elements. We call these the 2:4, 2:8+ and 4:8+ refinement cases. The refinement cases are summarized in Fig. 4. This restrictive set of refinement rules avoids the appearance of ill-deformed elements. At the same time, it considerably simplifies the refinement/coarsening logic. An interesting phenomenon that does not appear in 2D is the apparently free choice of the inner diagonal for the 1:8 refinement case. As shown in Fig. 5, one can place the inner four elements around the inner diagonals 5–10, 6–8 or 7–9. Choosing the shortest inner diagonal produces the smallest amount of distorted tetrahedra in the refined grid. When coarsening, again only a limited number of cases that are compatible with refinement is allowed. The coarsening cases become 8:4; 8:2, 8:1, 4:2, 4:1, 2:1 (Fig. 6). Similar possible refinement patterns have been devised for triangles [25, 53], quads [4, 5, 8, 54–56] and bricks [24, 55, 57, 58].

A considerable simplification in logic can be achieved by limiting the number of refinement/coarsening levels per mesh change to one [25, 33]. The main algorithmic steps then become:

- identify the elements to be refined/coarsened;
- make elements to be refined compatible by expanding the refinement region;
- make elements to be coarsened compatible by shrinking the coarsening region;
- refine/coarsen the mesh;
- correct the location of new boundary points according to the surface definition data available;
- interpolate the unknowns and boundary conditions.

Most of the steps listed above are easily vectorizable and parallelizable, leading to an extremely fast adaptation procedure. This makes h-refinement one of the few adaptation procedures suitable

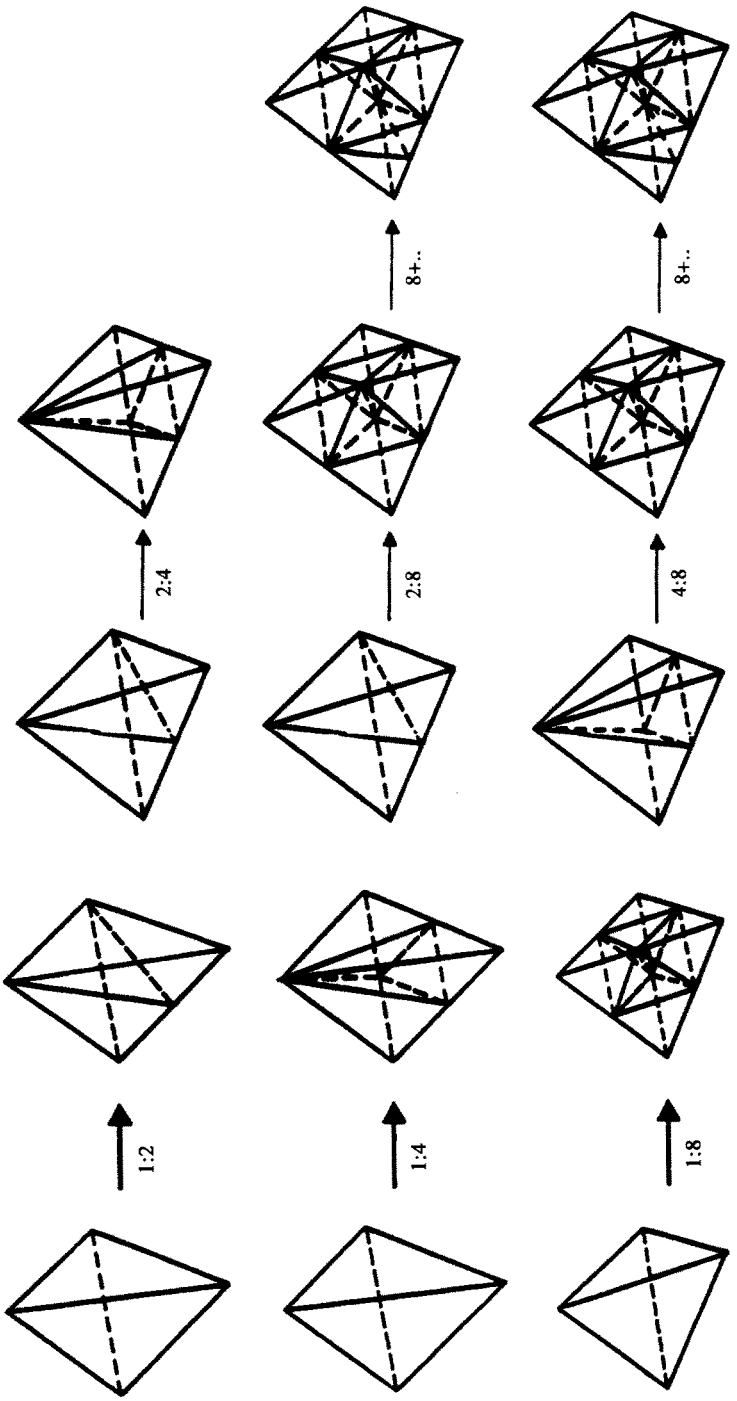


Fig. 4. Refinement cases for tetrahedra.

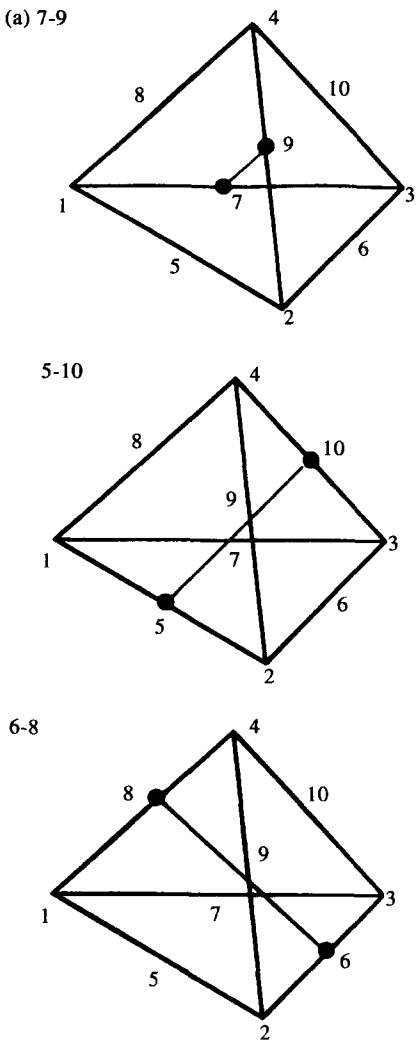


Fig. 5. (a) (*See caption on page* 831).

for strongly unsteady flows [34–36, 38]. h-Refinement has had the biggest impact in the field, leading to saving ratios in CPU and memory requirements in excess of 1:100 [34], the equivalent of two supercomputer generations. It has allowed the routine simulation of problems that had been intractable to date, opening new areas of applications in CFD [31, 33, 35, 36, 38].

4.2.1.2. *Recursive subdivision of the largest edge.* The separate coding and maintenance of allowable refinement and coarsening patterns observed for classic h-refinement in conjunction with triangles and tetrahedra can be avoided by using a recursive subdivision algorithm proposed by Rivara [59–61]. Whenever an element has been marked for refinement, the largest edge of the element is subdivided, producing the 1:2 refinement pattern in Fig. 4. Hanging nodes are then reconnected to produce a mesh without hanging nodes. The elements are checked again for refinement, and the procedure is repeated until a fine enough mesh has been obtained. This algorithm may be summarized as follows:

- Until the expected error is decreased sufficiently:
- identify the elements to be refined/coarsened;
- subdivide the largest edge of all elements marked for refinement;
- subdivide further elements with hanging nodes until no hanging nodes are present.

4.2.1.3. *Agglomeration of cells with blocked subdivisions.* The main aim of this type of refinement, which so far has been used only with quad and brick elements, is to achieve locally ordered blocks

of data, so that the amount of (expensive) indirect addressing operations required by the flow solver can be minimized. The main algorithmic steps can be summarized as follows:

- identify the elements to be refined/coarsened;
- identify possible blocking patterns;
- make blocking patterns compatible;
- refine/coarsen the mesh;
- correct the location of new boundary points according to the surface definition data available;
- interpolate the unknowns and boundary conditions.

As one can see, the main difference between this algorithm and the classic h-refinement is the pattern recognition phase and the compatibility checking phase that is now operating on blocks instead of individual elements. In particular, the pattern recognition phase has been the subject of considerable research [62], producing impressive results [58, 63].

4.3. *Remeshing (*m*-methods)*

The third family of refinement strategies is based on the existence of automatic grid generators. The grid generator is used in combination with an error indicator based on the present discretization and solution to remesh the computational domain either globally or locally, in order to produce a more suitable discretization. The main advantages offered by adaptive remeshing methods are the following.

(a) The possibility of stretching elements when adapting features that are of lower dimensionality than the problem at hand (e.g. shock surfaces in 3D), which leads to considerable savings as compared to h-refinement.

(b) The ability to accommodate in a straightforward manner problems with moving bodies or free surfaces. For this class of problems a fixed mesh structure will, in most cases, lead to badly distorted elements. This implies that at least a partial regeneration of the computational domain is required. An observation made from practical runs is that as the bodies move through the flow field, the positions of relevant flow features will change. Therefore, in most of the computational domain, a new mesh distribution will be required.

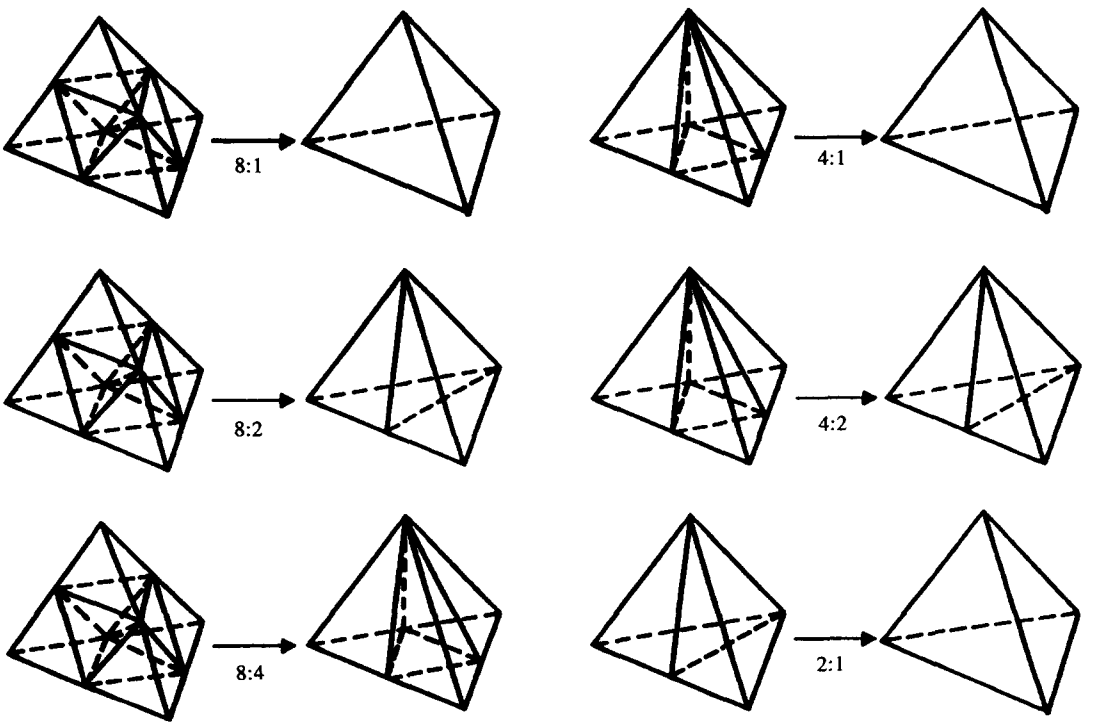


Fig. 6(e)

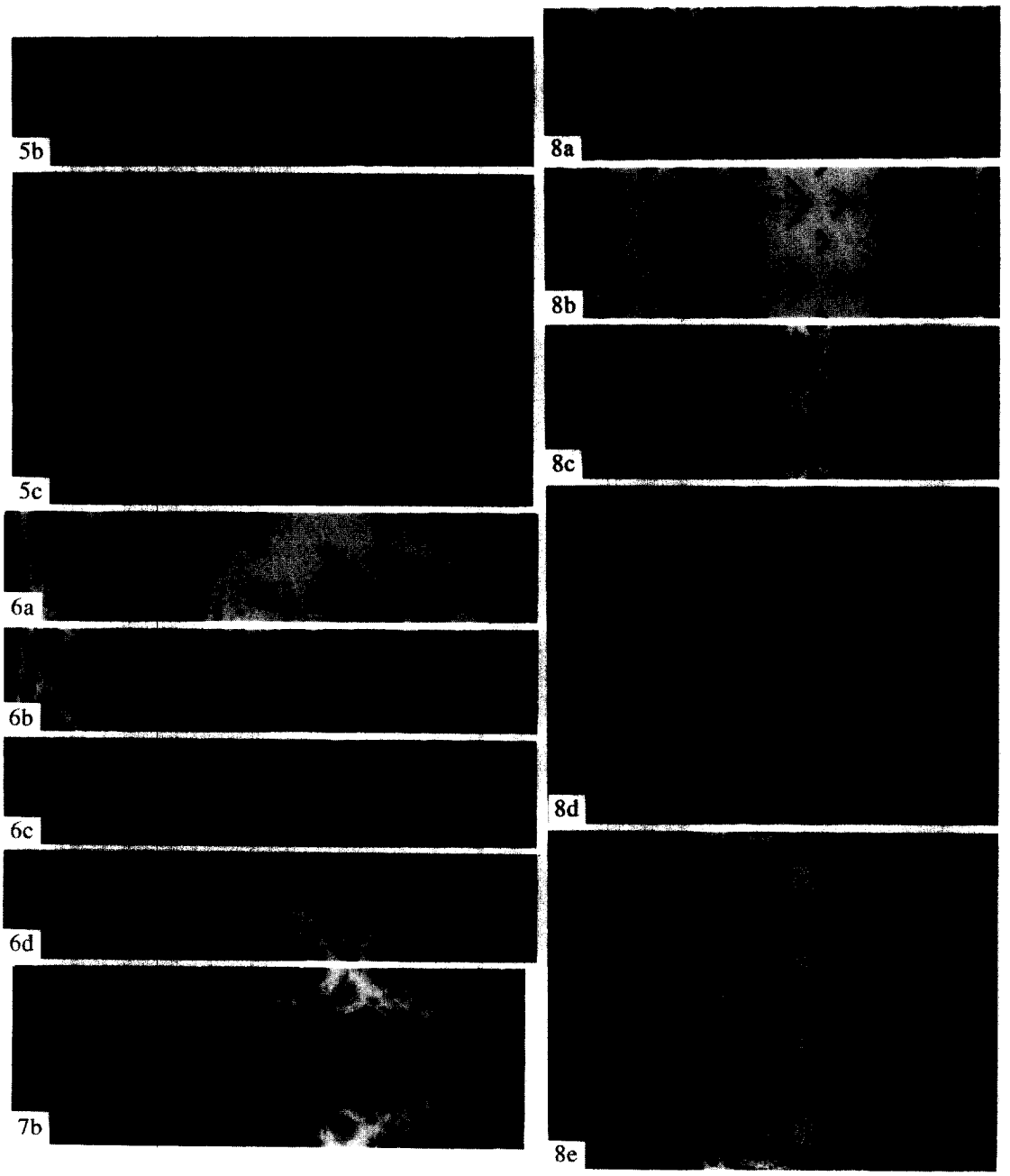


Fig. 5. Choice of inner diagonal for h-refinement on tetrahedra.

Fig. 6. Coarsening cases for tetrahedra.

Fig. 7. Convection between concentric cylinders.

Fig. 8. Shock–object interaction in 2D.

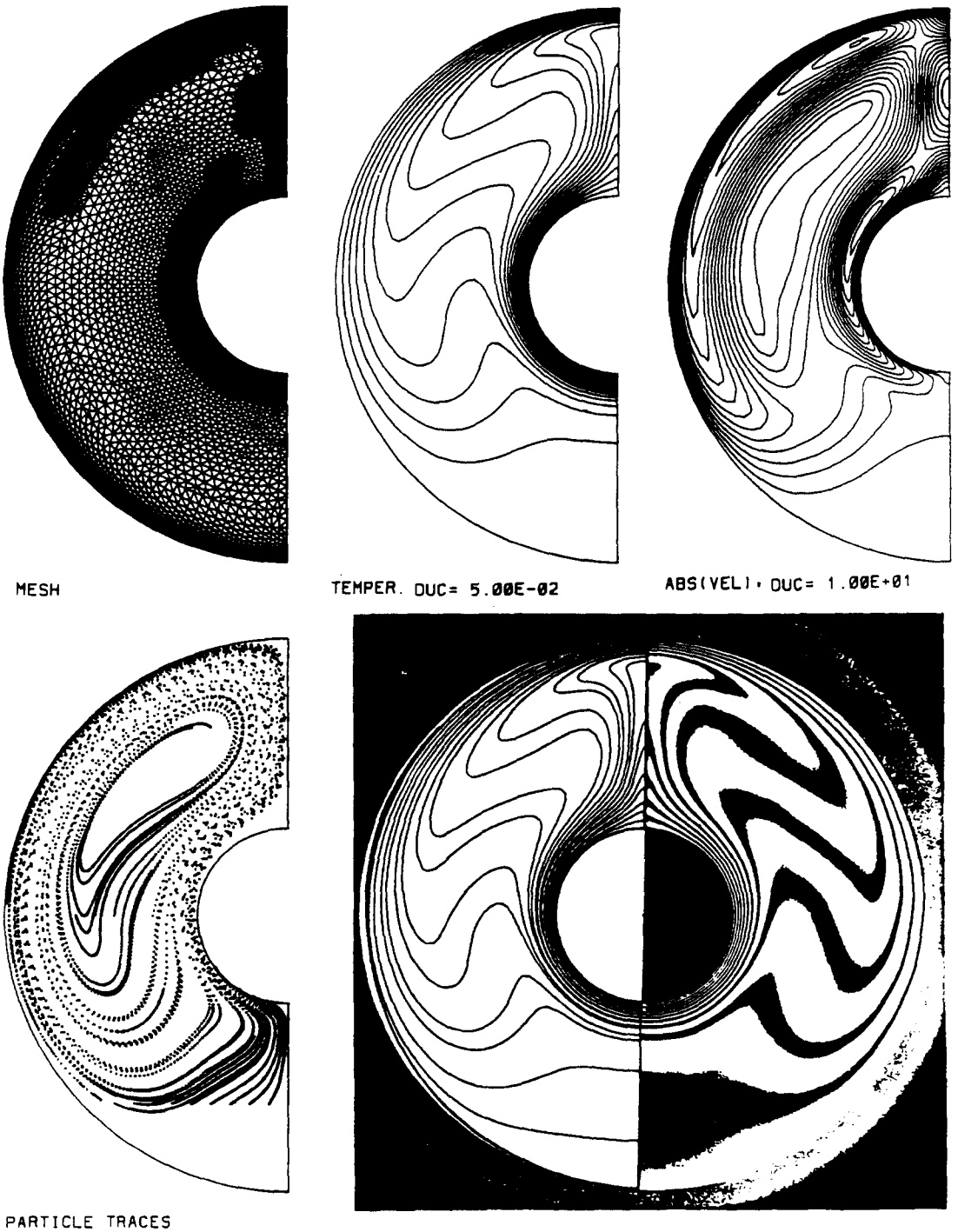


Fig. 7(a) (*see caption on page 831*)

Any of the automatic grid generation techniques currently available—advancing front [26, 27, 64–66], Delaunay triangulations [6, 7, 67] and modified quadtree/octree [68, 69]—may be employed to regenerate the mesh. The basic algorithmic steps are outlined below for the advancing front method, but any other automatic grid generation technique can be employed instead. The steps required for one *adaptive remeshing* are as follows:

- obtain the error indicator matrix for the gridpoints of the present grid;
- given the error indicator matrix, get the element size, element stretching and stretching direction for the new grid;

- using the old grid as the background grid, remesh the computational domain using the advancing front technique;
- if further levels of gobal h-refinement are desired, refine the new grid globally;
- interpolate the solution from the old grid to the new one.

For adaptive remeshing using the advancing front method, see refs [15, 28, 30, 32, 37, 64, 65, 70–74]; using Delaunay triangulations, see refs [6, 7]; and using the modified quad/octree approach, see ref. [12].

4.3.1. *Local remeshing.* For some classes of problems, e.g. subsonic store separation, the mesh may become distorted at a rate that is higher than the movement of physically relevant features. Another observation made from practical simulations is that the appearance of badly distorted elements occurs at a frequency that is much higher than expected from the element size prescribed. Given the relatively high cost of global remeshing, local remeshing in the vicinity of the distorted elements becomes an attractive option for these situations. The steps required are as follows:

- Identify the badly distorted elements in the layers that move, writing them into a list LEREM (1:NEREM).
- Add to this list the elements surrounding these badly distorted elements.
- Form holes in the present mesh by:
  forming a new background mesh with the elements stored in the list LEREM:
  deleting the elements stored in LEREM from the current mesh;
  removing all unused points from the grid thus obtained.
- Recompute the error indicators and new element distribution for the background grid.
- Regrid the holes using the advancing front method.

Typically, only a very small number of elements (<10) becomes so distorted that a remeshing is required. Thus, local remeshing is a very economical tool that allows reductions in CPU requirements by more than 60% for typical runs [32, 74].

4.4. *Combinations*

From the characteristics listed above, it is clear that all of the possible mesh refinement strategies exhibit weaknesses. Not surprisingly, some hybrid methods have been pursued that seek to compensate the weaknesses of the individual methods with their respective strengths. The most common are the following.

4.4.1. *Mesh enrichment followed by mesh movement.* In this approach, which has been used mainly for steady-state applications [3], the physical complexity is first accounted for by classic h-enrichment. The mesh is then further improved by moving the points. In this way, the flexibility of the h-enrichment technique is combined with the points. In this way, the flexibility of the h-enrichment technique is combined with the directionality feature of mesh movement.

4.4.2. *Mesh regeneration and mesh enrichment.* In this approach, which has been used for strong unsteady problems with moving bodies [75], the physical complexity is treated by classic h-enrichment. Moving bodies and/or changing topologies are treated by remeshing.

5. EXAMPLES

A number of examples, computed with some of the author's codes [76], are given to demonstrate some of the possible applications of adaptive mesh procedures.

5.1. *Convection between concentric cylinders*

This case, taken from van Dyke [77], shows the use of h-refinement for buoyancy driven flows. The ratio of diameters is $d_o/d_i = 3.14$, and the Grashof number based on the gap width is $Gr = 122{,}000$. The Reynolds and Prandtl numbers were $Re = 1.0$ and $Pr = 0.71$, respectively. After obtaining a first solution, mesh adaptation was invoked to resolve in more detail the temperature

gradients. The resulting flow and temperature fields are displayed in Fig. 7. The comparison with the results presented in van Dyke [77] is very good. It is noteworthy to remark that the plates where refinement took place are rather non-intuitive. On the other hand, the results obtained on the original, unrefined mesh did not exhibit a good comparison to the experiment.

5.2. *Shock–object interaction in 2D*

Figure 8a–e shows a case taken from Baum and Löhner [36]. They show classic h-refinement for strongly unsteady flows at its best. For this class of problems a new mesh is required every 5–7 timesteps, strict conservation of mass, momentum and energy during refinement is critical, and the introduction of dissipation due to information loss during interpolation when remeshing would be

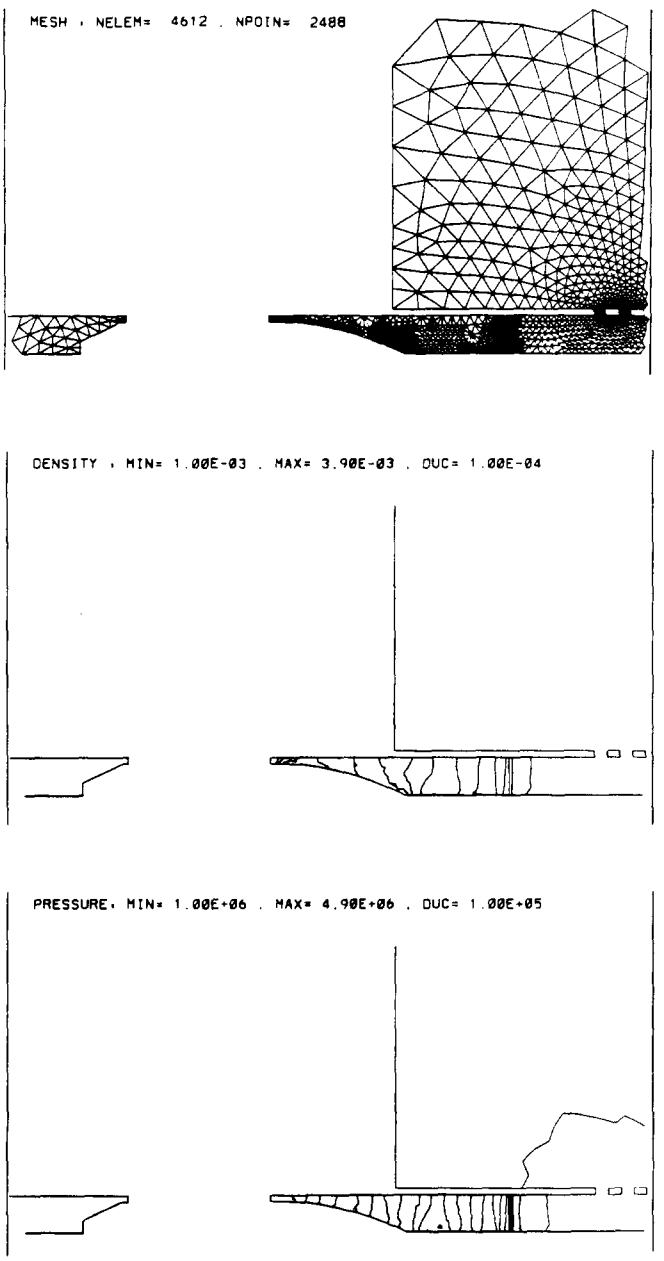


Fig. 9(a)

disastrous for the accuracy. A maximum of six levels of refinement were specified for this case, yielding meshes that on average have 300 Ktria and 150 Kpoints. Observe the detail in the physics that is achievable through adaptation. Notice, furthermore, the small extent of the regions that require refinement as compared to the overall domain. The equivalent uniform mesh run would have required more than two orders of magnitude more elements, CPU time and memory, pushing the limits of available supercomputers.

5.3. *Projectile exiting muzzle*

The problem definition, as well as the resulting flow fields and grids, are shown in Fig. 9a–f. The gas was modeled as ideal, and the density and pressure ratios were set to 1:40 and 1:2000,

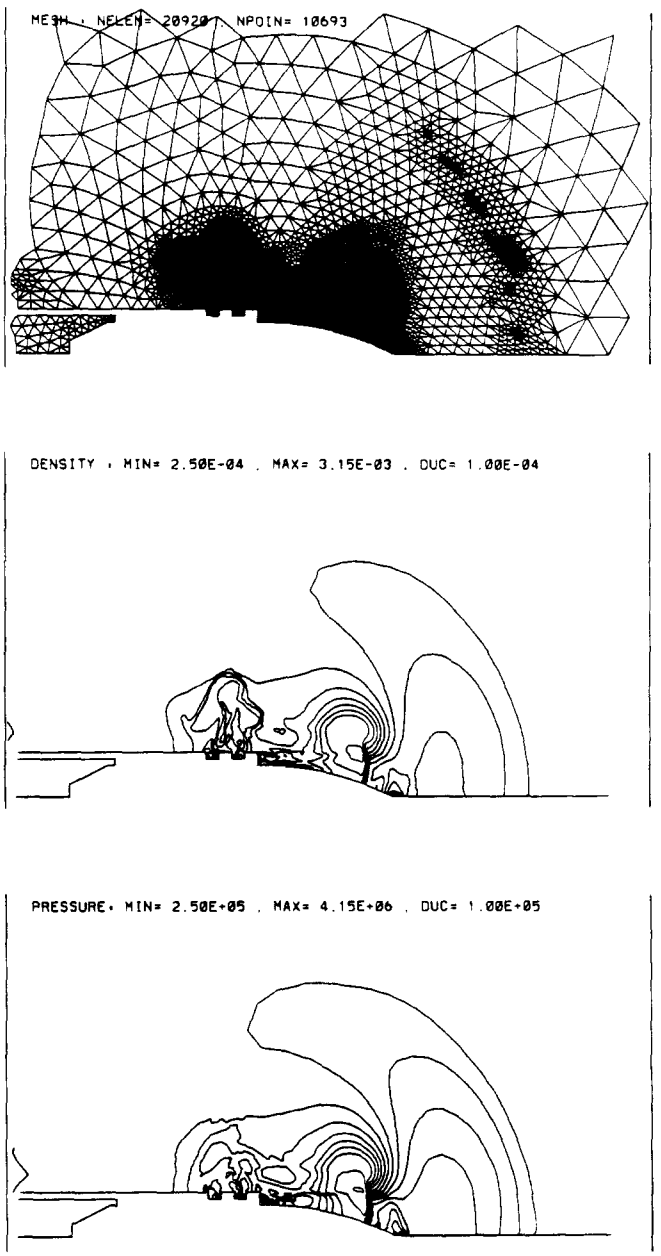


Fig. 9(b)

respectively. The Euler equations were solved in cylindrical coordinates. This particular flow problem exhibits considerable mesh movement and topology changes. It demonstrates the use of classic h-refinement in combination with remeshing. The flow through the pressure vents can be discerned clearly from Fig. 9e, f. For more details, see ref. [75].

5.4. *Shock–object interaction in 3D*

Figure 10a–c shows a case taken from refs [33, 35]. The object under consideration is a common main battlefield tank. A maximum of two layers of refinement were specified close to the

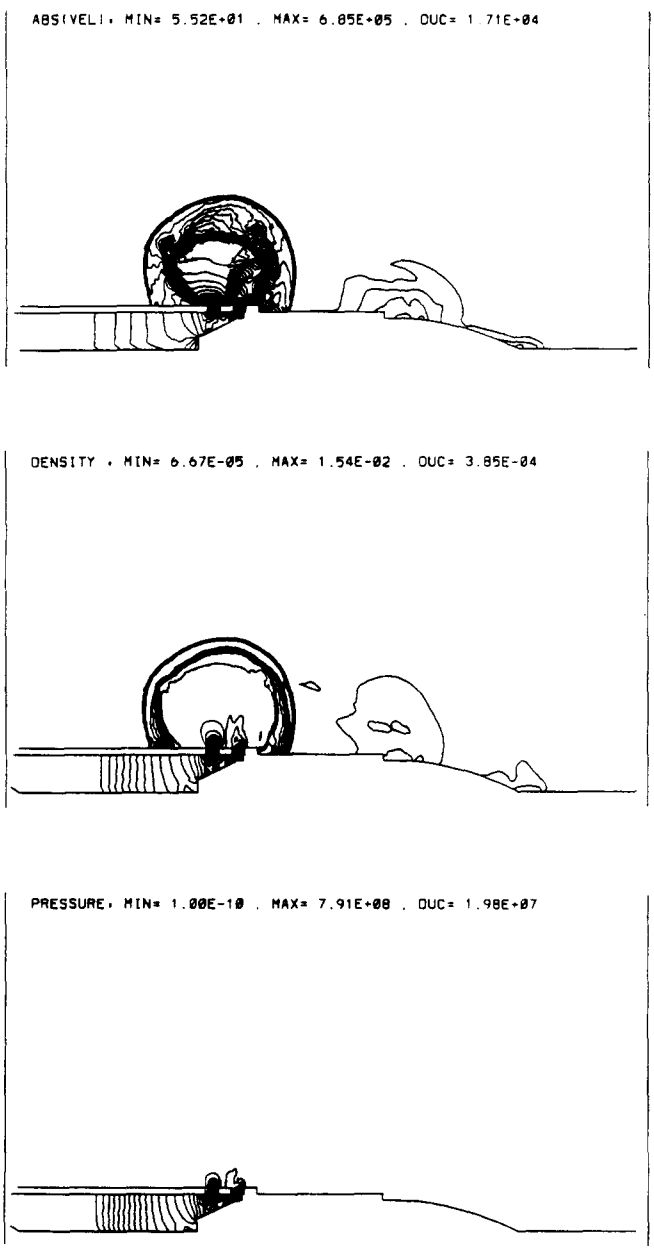


Fig. 9(c)

Fig. 9. Projectile exiting muzzle.

tank, whereas only one level of refinement was employed further away. The original, unrefined, but strongly graded mesh consisted of approximately 100 Ktetra and 20 Kpoints. During the run, the mesh size increased to approximately 1.6 Mtetra and 280 Kpoints. This represents an increase factor of 1:16. Although seemingly high, the corresponding global h-refinement would have resulted in a 1:64 size increase. A second important factor is that most of the elements of the original mesh are close to the body, where most of the refinement is going to take place. Figure 10a–c shows surface gridding and pressure contours at selected times during the run. The extent of mesh refinement is clearly discernible, as well as the location and interaction of shocks.

A complete run like the one shown here takes approximately 35 h of single-processor CRAY-YMP time, and 110 Mwords of memory. Given that it takes 3–5 days to grid up a configuration like the one shown, and another 3–5 days to plot and understand the data, these CPU and memory requirements are not deemed excessive. Moreover, given the DO-loop lengths encountered in a run like the one shown, microtasking works very well. On an eight-processor CRAY-YMP, speed-ups in excess of 1:6 have been obtained. This reduces run times to 6 h, an overnight run. From these figures, it also becomes apparent that without adaptive mesh refinement, a run like this would be impossible on today's hardware.

5.5. *Object falling into supersonic free stream (2D)*

The problem statement is as follows: an object is placed in a cavity surrounded by a free stream at $M_\infty = 1.5$. After the steady-state solution is reached (time $T = 0.0$), a body motion is prescribed, and the resulting flow field disturbance is computed. Adaptive remeshing was performed every 100 timesteps initially, while at later times the grid was modified every 50 timesteps. One level of global h-refinement was used to accelerate the grid regeneration. The maximum stretching ratio specified was $S = 5.0$. Figure 11a, b shows different stages during the computation at times $T = 60$ and $T = 160$. One can clearly see how the location and strength of the shocks change due to the motion of the object. Notice how the directionality of the flow features is reflected in the mesh.

5.6. *Object falling into supersonic free stream (3D)*

The computational domain is shown in Figure 12a, b. The store is a slender object, resembling a missile. Figure 12a, b shows the mesh on the surface of the computational domain at time $T = 0.0$. The corresponding pressure contours (30) are shown in Fig. 12c. Figure 12e–f shows the surface mesh and the pressure contours (60) at time $T = 8.5$. One can clearly discern the extent of mesh adaptation, as well as the considerable change in shock strengths and shock positions that occurred due to body motion. The meshes shown here have approximately 300 Ktetra and 55 Kpoints. A run like the one shown takes about 6 h of Cray-2 time. For more details, see ref. [32].

6. CONCLUSIONS

Adaptive mesh refinement has progressed rapidly over the last decade. The main types of optimal mesh criteria, error indicators/estimators and refinement strategies have been reviewed and compared. Although the advances have been impressive, major areas require significant further development before adaptivity will become commonplace in computational mechanics. Among the most obvious, we mention:

- reliable quantitative error estimators for high Re Navier–Stokes equations;
- reliable quantitative error estimators for transient problems;
- adaptive refinement procedures capable of automatically producing elements with very high aspect ratio ($>1:1000$) cells in arbitrary directions;
- porting of adaptive refinement procedures to parallel computers.

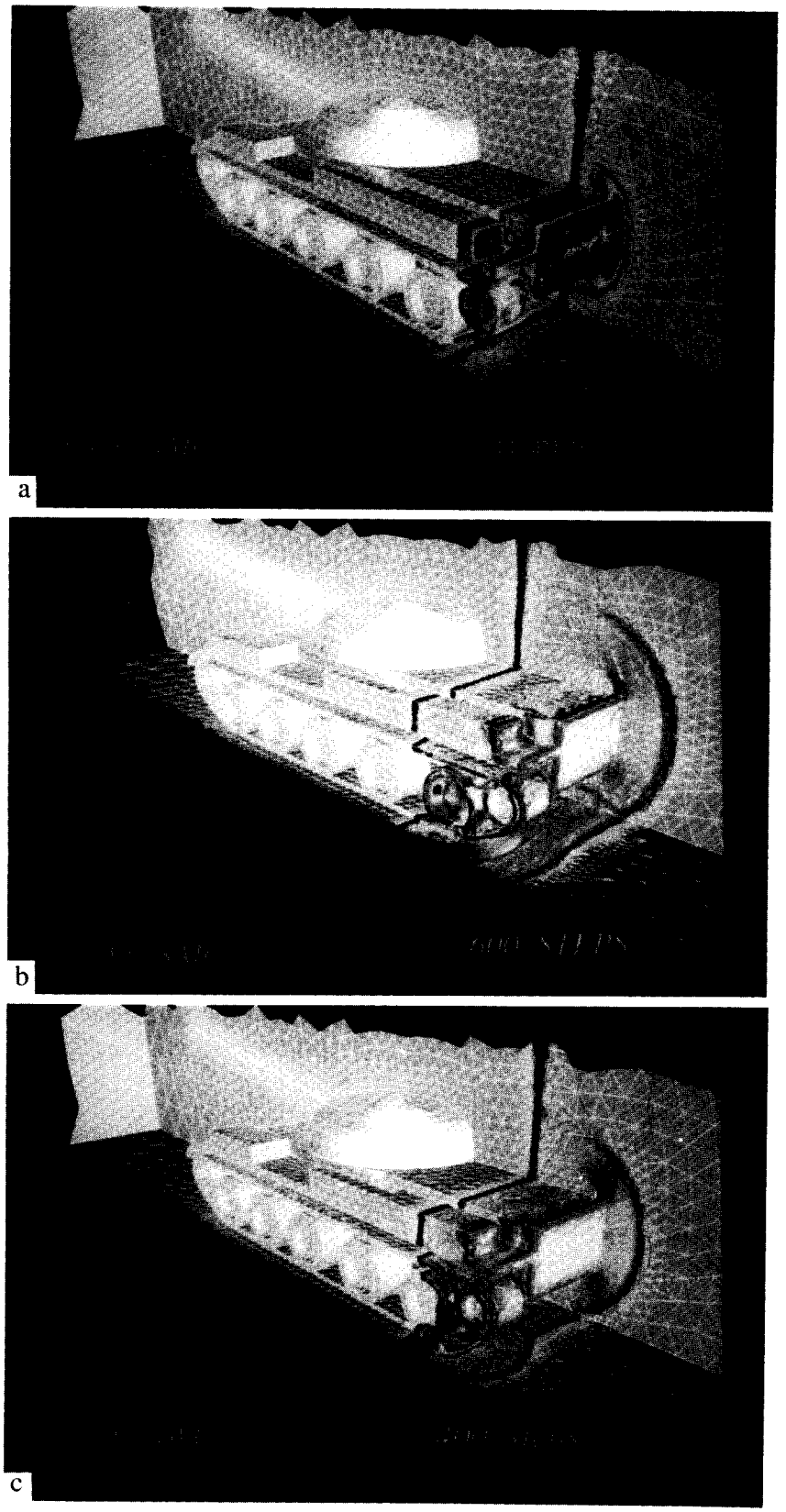


Fig. 10 (a, b and c)

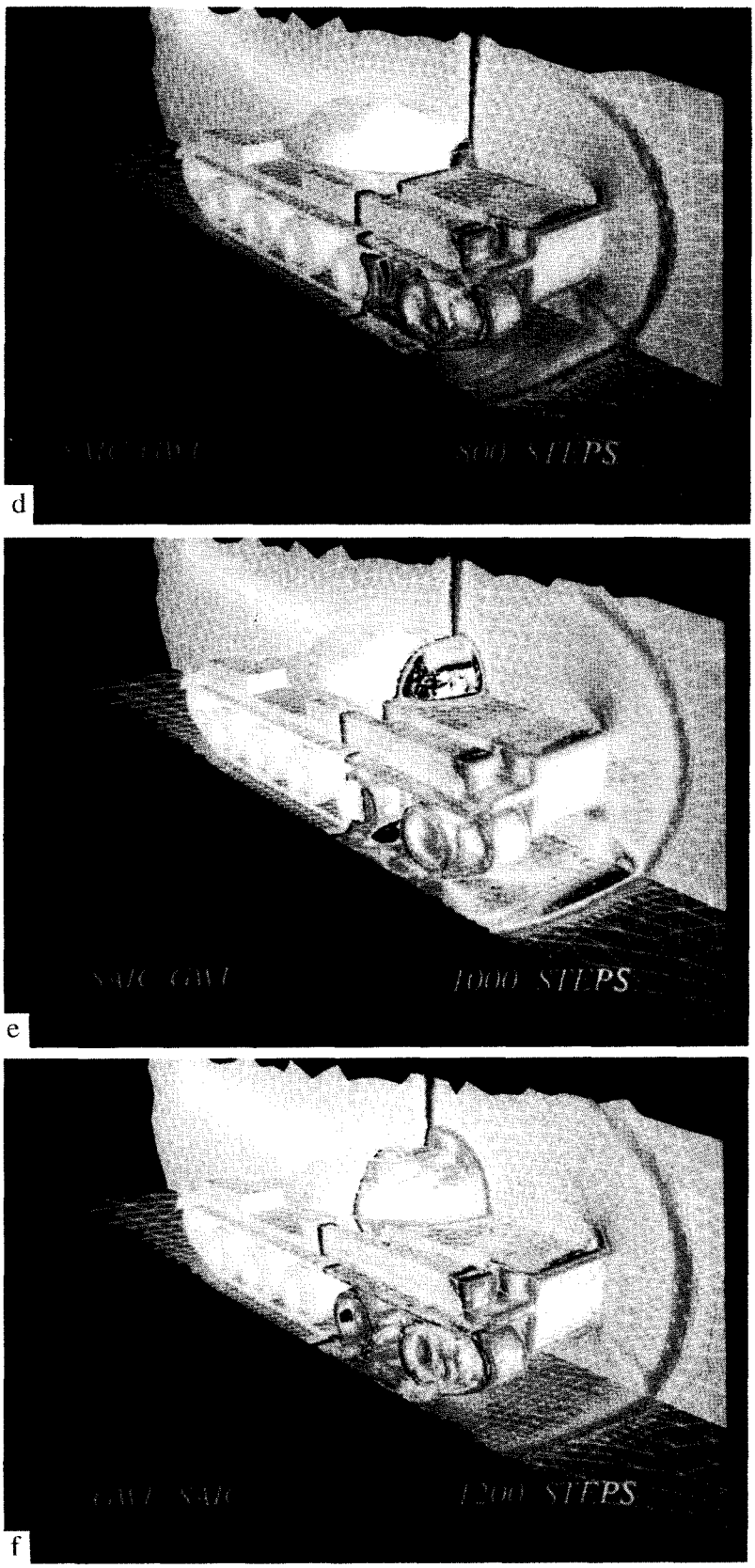


Fig. 10 (d, e and f)

Fig. 10. Shock–object interaction in 3D.

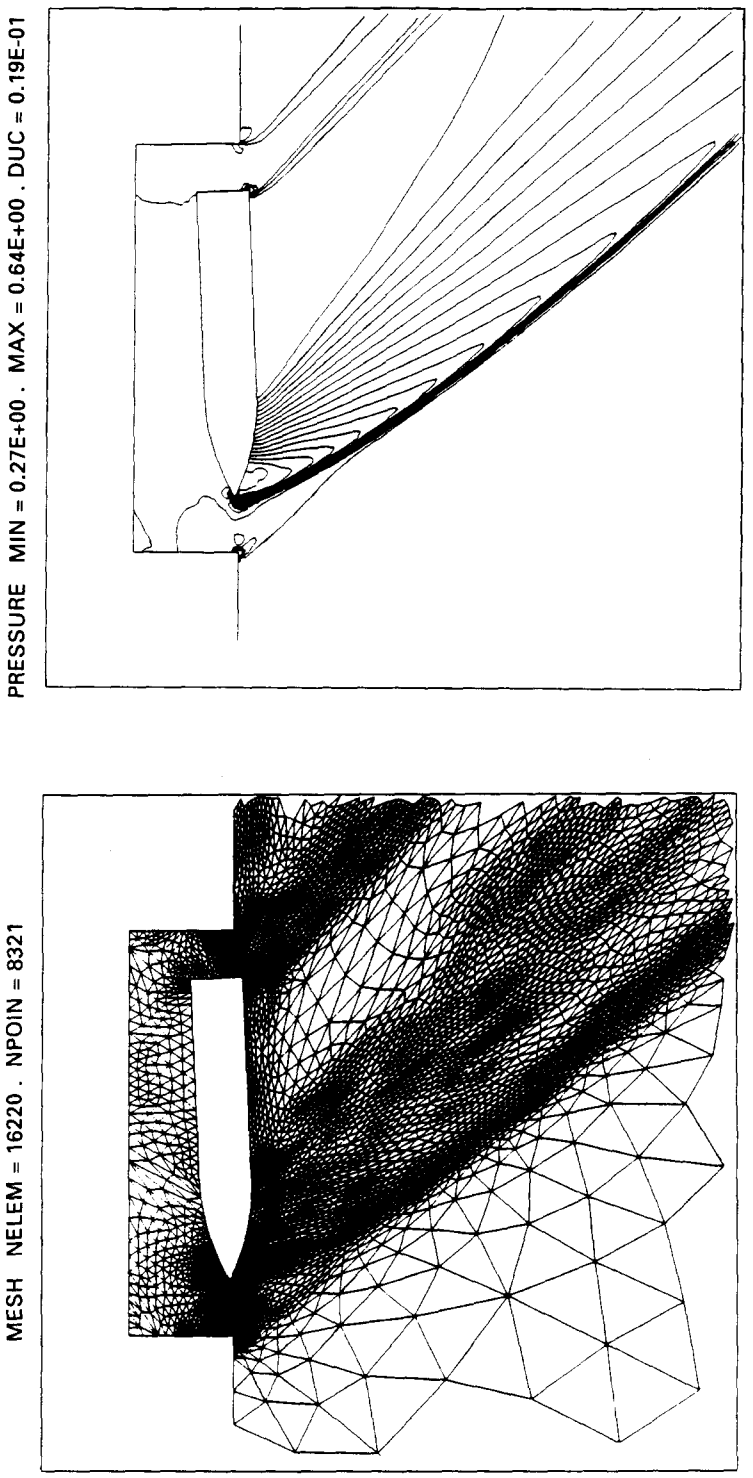
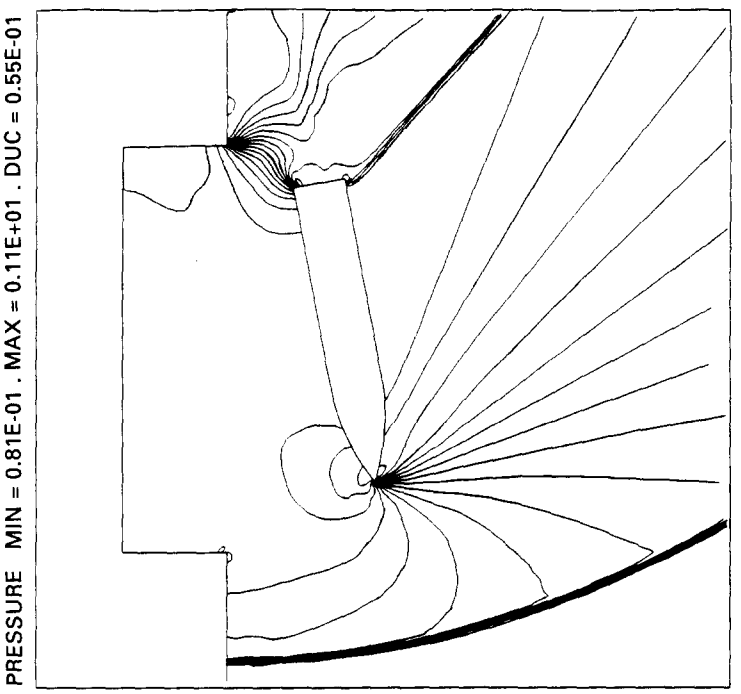

PRESSURE MIN = 0.27E+00 . MAX = 0.64E+00 . DUC = 0.19E-01
MESH NELEM = 16220 . NPOIN = 8321

Fig. 11(a)



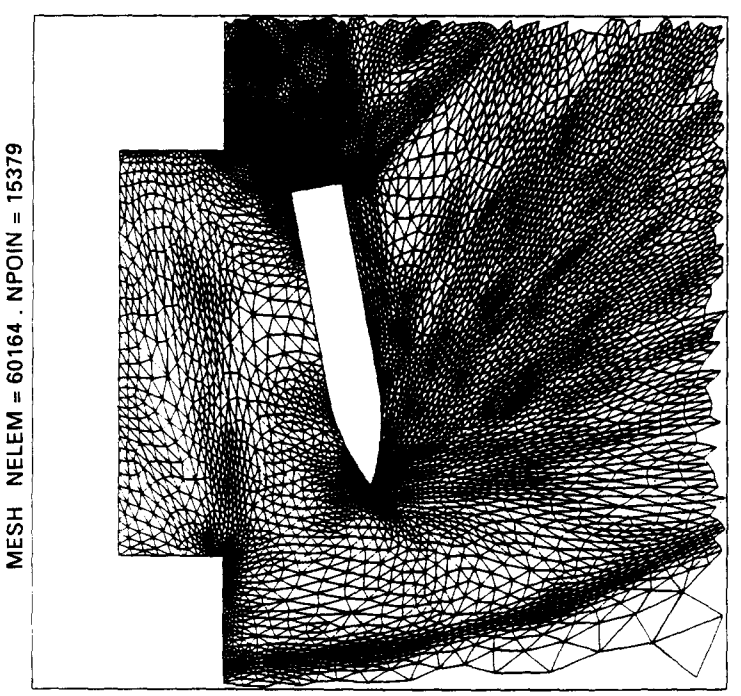


Fig. 11(b)

Fig. 11. Object falling into supersonic free stream (2D).

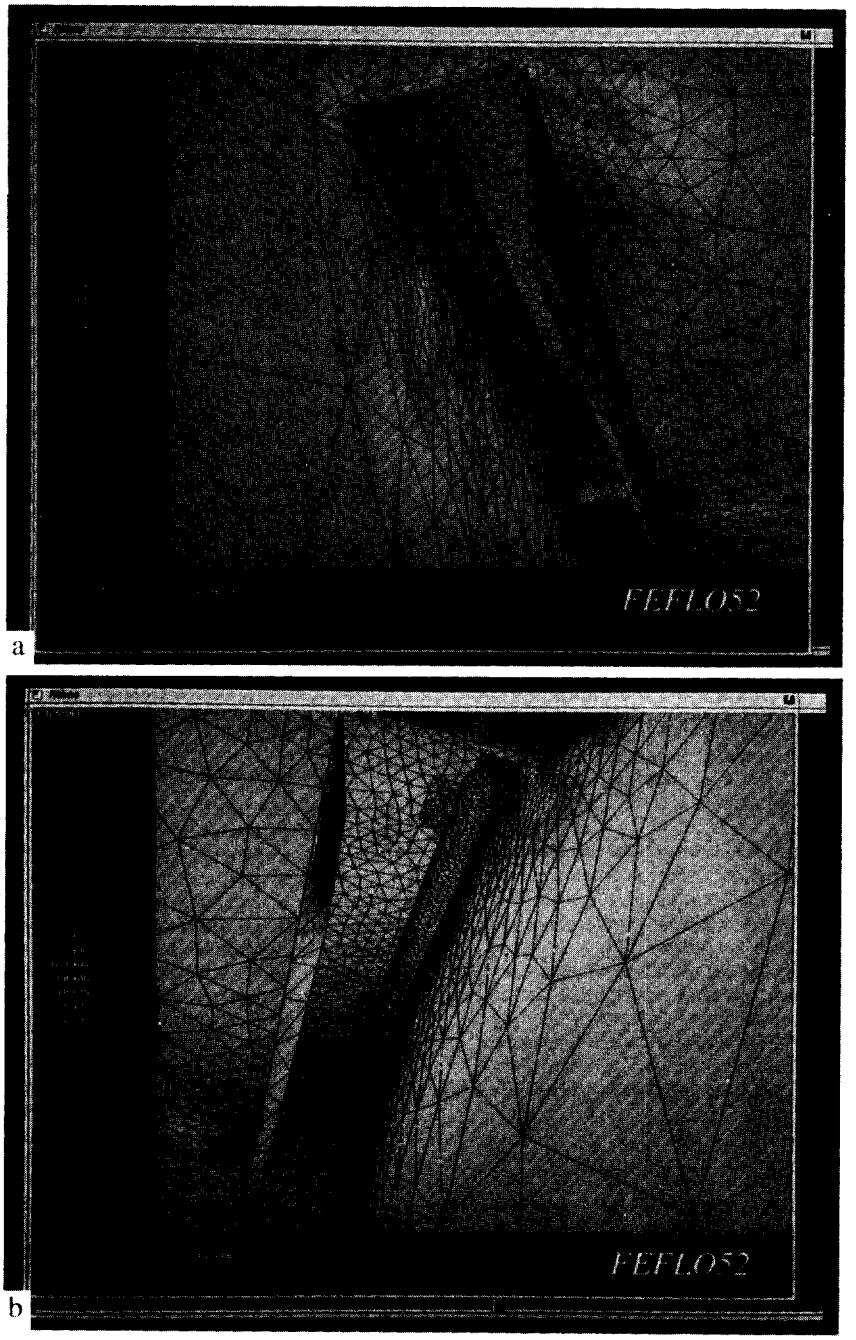


Fig. 12 (a and b)

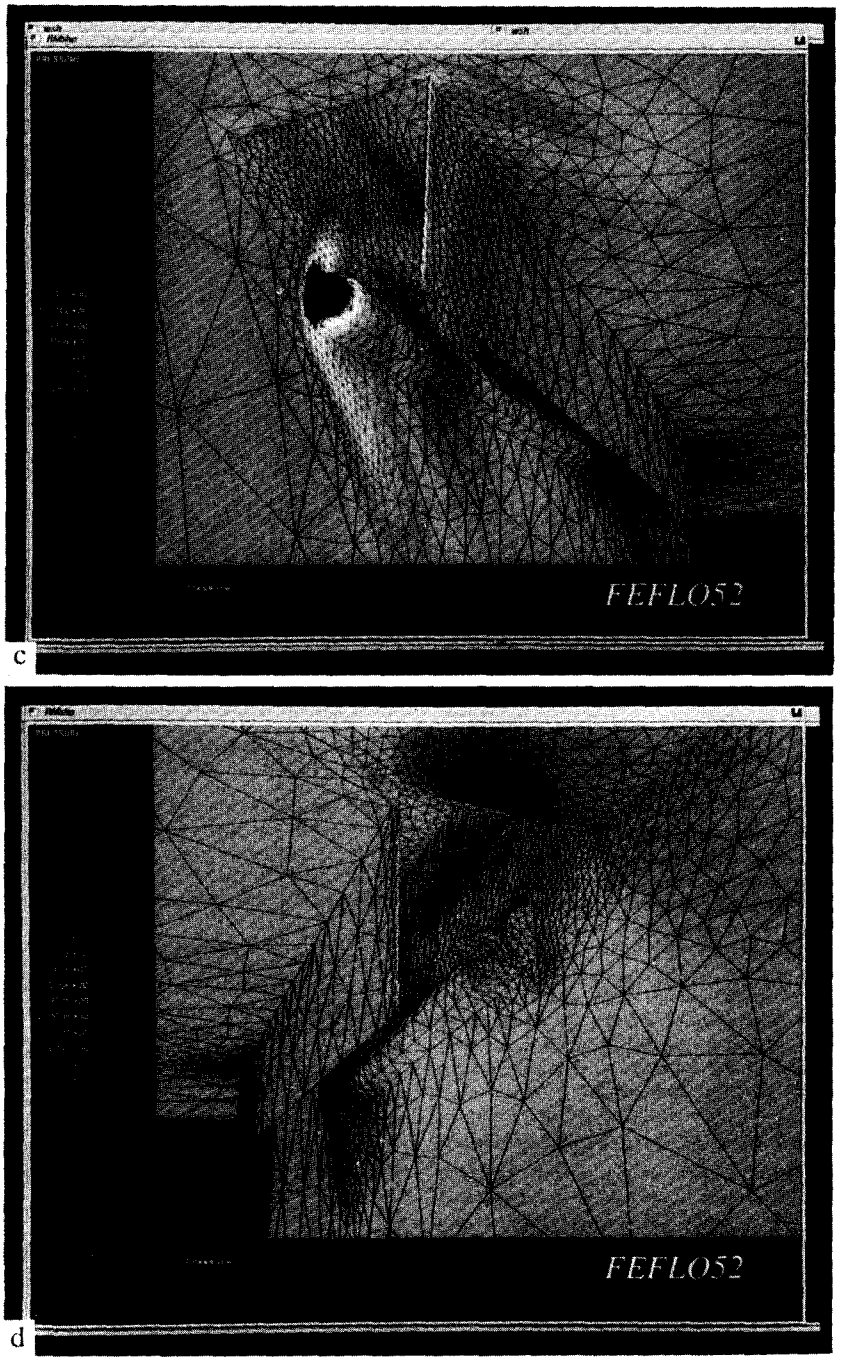


Fig. 12 (c and d)

Fig. 12. Object falling into supersonic free stream (3D).

Acknowledgements—I would like to acknowledge many fruitful and stimulating discussions during the course of the years with Profs O. C. Zienkiewicz and K. Morgan from Swansea, Prof. J. Peraire from Imperial College, as well as Drs J. P. Boris, D. L. Book, S. T. Zalesak and R. Ramamurti from NRL-LCP&FD, Dr M. D. Salas from NASA LARC, Drs J. D. Baum and H. Luo from SAIC, and the members of my own team at the George Washington University: Drs Jean Cabello, B. Petitjean, D. Martin and E. Mestreau. The support of CRAY Research in the form of many free hours on a CRAY-2 over the years, and IBM for providing me with a RISC-6000/550 for private use at home is also gratefully acknowledged.

REFERENCES

[1] I. Babuska, J. Chandra and J. E. Flaherty (Eds), *Adaptive Computational Methods for Partial Differential Equations*. SIAM, Philadelphia, PA (1983).
[2] I. Babuska, O. C. Zienkiewicz, J. Gago and E. R. de A. Oliverira (Eds), *Accuracy Estimates and Adapative Refinements in Finite Element Computations*. John Wiley, New York (1986).
[3] B. Palmerio and A. Dervieux, Application of a FEM moving node adaptive method to accurate shock capturing. *Proc. First Int. Conf. on Numerical Grid Generation in CFD*, Landshut, W. Germany (14–17 July 1986).
[4] J. F. Dannenhoffer and J. R. Baron, Robust grid adaptation for complex transonic flows. AIAA-86-0495 (1986).
[5] J. G. Kallinderis and J. R. Baron, Adaptation methods for a new Navier–Stokes algorithm. AIAA-87-1167-CP, Hawaii (1987).
[6] D. J. Mavriplis, Adaptive mesh generation for viscous flows using delaunay triangulation. *J. Comput. Phys.* **90,** 271–291 (1990).
[7] D. J. Mavriplis, Turbulent flow calculations using unstructured and adaptive meshes. *Int. J. numer. Meth. Fluids* **13,** 1131–1152 (1991).
[8] M. Aftosmis and N. Kroll, A quadrilateral based second-order TVD method for unstructured adaptive meshes. AIAA-91-0124 (1991).
[9] D. DeZeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations. *J. Comput. Phys.* **104,** 56–68 (1993).
[10] W. Schönauer, K. Raith and K. Glotz, The principle of difference quotients as a key to the self-adaptive solution of nonlinear partial differential equations. *Comput. Meth. appl. Mech. Engng* **28,** 327–359 (1981).
[11] B. Palmerior, V. Billey, A. Dervieux and J. Periaux, Self-adaptive mesh refinements and finite element methods for solving the Euler equations. *Proc. ICFD Conf. Numer. Meth. for Fluid Dyn.*, Reading, U.K. (Mar. 1985).
[12] P. L. Baehmann, M. S. Shepard and J. E. Flaherty, *A posteriori* error estimation for triangular and tetrahedral quadratic elements using interior residuals. *Int. J. numer. Meth. Engng* **34,** 979–996 (1992).
[13] O. C. Zienkiewicz, J. P. de S. R. Gago and D. W. Kelly, The hierarchical concept in finite element analysis. *Comput. Structures* **16,** 53–65 (1983).
[14] D. A. Dunavant and B. A. Szabo, *A posteriori* error indicators for the *P*-version of the finite element method. *Int. J. numer. Meth. Engng* **19,** 1851–1870 (1983).
[15] J.-F. Hétu and D. H. Pelletier, Adaptive remeshing for viscous incompressible flows. *AIAA J.* **30,** 1986–1992 (1992).
[16] J. Z. Zhu and O. C. Zienkiewicz, A simple error estimator and adaptive procedure for practical engineering analysis. *Int. J. numer. Meth. Engng* **24,** 337–357 (1987).
[17] O. C. Zienkiewicz, Y. C. Liu and G. C. Huang, Error estimation and adaptivity in flow formulation of forming processes. *Int. J. numer. Meth. Engng* **25,** 23–42 (1988).
[18] M. Ainsworth, J. Z. Zhu, A. W. Craig and O. C. Zienkiewicz, Analysis of the Zienkiewicz–Zhu *a posteriori* error estimate in the finite element method. *Int. J. numer. Meth. Engng* **28,** 2161–2174 (1989).
[19] C. Mavriplis, *A posteriori* error estimators for adaptive spectral element techniques. *Proc. 8th GAMM Conf. Numer. Meth. Fluid Mech., NNFM 29*, pp. 333–342, Vieweg (1990).
[20] C. Mavriplis, Adaptive mesh strategies for the spectral element method. ICASE Rep. 92–36 (1992).
[21] T. Strouboulis and J. T. Oden, *A posteriori* error estimation for finite element approximations in fluid mechanics. *Comput. Meth. appl. Mech. Engng* **78,** 201–242 (1990).
[22] T. Strouboulis and K. A. Haque, Recent experiences with error estimation and adaptivity; Part 1: review of error estimators for scalar elliptic problems. *Comput. Meth. appl. Mech. Engng* **97,** 399–436 (1992).
[23] C. Johnsson and P. Hansbo, Adaptive finite element methods in computational mechanics. *Comput. Meth. appl. Mech. Engng* **101,** 143–181 (1992).
[24] M. Aftosmis, A second-order TVD method for the solution of the 3-D Euler and Navier–Stokes equations on adaptively refined meshes. *Proc. 13th Int. Conf. Numer. Meth. Fluid Dyn.*, Rome, Italy (1992).
[25] R. Löhner, An adaptive finite element scheme for transient problems in CFD. *Comput. Meth. appl. Mech. Engng* **61,** 323–338 (1987).
[26] R. Löhner, Some useful data structures for the generation of unstructured grids. *Commun. appl. numer. Meth.* **4,** 123–135 (1988).
[27] R. Löhner and P. Parikh, Three-dimensional grid generation by the advancing front method. *Int. J. numer. Meth. Fluids* **8,** 1135–1149 (1988).
[28] R. Löhner, An adaptive finite element solver for transient problems with moving bodies. *Comput. Structures* **30,** 303–317 (1988).
[29] R. Löhner, Adaptive *H*-refinement on 3-D unstructured grids for transient problems. AIAA-89-0365 (1989).
[30] R. Löhner, Adaptive remeshing for transient problems. *Comput. Meth. appl. Mech. Engng* **75,** 195–214 (1989).
[31] R. Löhner and J. D. Baum, Numerical simulation of shock interaction with complex geometry three-dimensional structures using a new adaptive *H*-refinement scheme on unstructured grids. AIAA-90-0700 (1990).
[32] R. Löhner, Three-dimensional fluid–structure interaction using a finite element solver and adaptive remeshing. *Comput. Syst. Engng* **1,** 257–272 (1990).
[33] R. Löhner and J. D. Bam, Adaptive *H*-refinement on 3-D unstructured grids for transient problems. *Int. J. numer. Meth. Fluids* **14,** 1407–1419 (1992).
[34] J. D. Baum and R. Löhner, Numerical simulation of shock-elevated box interaction using an adaptive finite element shock capturing scheme. AIAA-89-0653 (1989).

[35] J. D. Baum and R. Löhner, Numerical simulation of shock interaction with a modern main battlefield tank. AIAA-91-1666 (1991).
[36] J. D. Baum and R. Löhner, Numerical simulation of passive shock deflector using an adaptive finite element scheme on unstructured grids. AIAA-92-0448 (1992).
[37] E. Loth, J. D. Baum and R. Löhner, Formation of shocks within axisymmetric nozzles. *AIAA J.* **30,** 268–270 (1992).
[38] S. Sivier, E. Loth, J. D. Baum and R. Löhner, Vorticity produced by shock wave diffraction. *Shock Waves* **2,** 31–41 (1992).
[39] K. Miller and R. N. Miller, *SIAM J. numer. Anal.* **18,** 1019 (1981).
[40] A. R. Diaz, N. Kikuchi and J. E. Taylor, A method for grid optimization for the finite element method. *Comput. Meth. Appl. Mech. Engng* **41,** 29–45 (1983).
[41] P. A. Gnoffo, A finite-volume, adaptive grid algorithm applied to planetary entry flowfields. *AIAA J.* **21,** 1249–1254 (1983).
[42] K. Nakahashi and G. S. Deiwert, A three-dimensional adaptive grid method. AIAA-85-0486 (1985).
[43] R. Carcaillet, S. R. Kennon and G. S. Dulikravitch, Generation of solution-adaptive computational grids using optimization. *Comput. Meth. appl. Mech. Engng* **57,** 279–295 (1986).
[44] J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.* **46,** 342–368 (1986).
[45] O.-P. Jacquotte and J. Cabello, Three-dimensional grid generation method based on a variational principle. *Rech. Aerosp.* **1990-4,** 8–19 (1990).
[46] J. F. Thompson, A survey of dynamically adapted grids in the numerical solution of partial differential equations. *Appl. numer. Math.* **1,** 3 (1985).
[47] P. Devloo, J. T. Oden and P. Pattani, An *H–P* adaptive finite element method for the numerical simulation of compressible flows. *Comput. Meth. appl. Mech. Engng* **70,** 203–235 (1988).
[48] D. W. Wang, I. N. Katz and B. A. Szabo, *H*- and *P*-version finite element analyses of a rhombic plate. *Int. J. numer. Meth. Engng* **20,** 1399–1405 (1984).
[49] B. A. Szabo, Estimation and control of error based on *P*-convergence, in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, Chapter 3. John Wiley, New York (1986).
[50] S. K. Godunov, *Mat. Sb.* **47,** 271–306 (1959).
[51] W. K. Anderson and D. L. Bonhaus, Navier–Stokes computations and experimental comparisons for multielement airfoil configurations. AIAA-93-0645 (1993).
[52] E. Hystopolulos and R. L. Simpson, Critical evaluation of recent second-order closure models. AIAA-93-0081 (1993).
[53] J. T. Batina, Vortex-dominated conical-flow computations using unstructured adaptively-refined meshes. *AIAA J.* **28,** 1925–1932 (1990).
[54] J. T. Oden, P. Devloo and T. Strouboulis, Adaptive finite element methods for the analysis of inviscid compressible flow: I. Fast refinement/unrefinement and moving mesh methods for unstructured meshes. *Comput. Meth. appl. Mech. Engng* **59,** 327–362 (1986).
[55] R. A. Shapiro and E. M. Murman, Adaptive finite element methods for the Euler equations. AIAA-88-0034 (1988).
[56] R. Davis and J. Dannenhoffer, Adaptive grid embedding Navier–Stokes technique for cascade flows. AIAA-89-0204 (1989).
[57] M. B. Bieterman, J. E. Bussoletti, G. L. Hilmes, F. T. Johnson, R. G. Melvin and D. P. Young, An adaptive grid method for analysis of 3-D aircraft configurations. *Comput. Meth. appl. Mech. Engng* **101,** 225–249 (1992).
[58] R. L. Davis and J. F. Dannenhoffer, 3-D adaptive grid-embedding Euler technique. AIAA-93-0330 (1993).
[59] M. C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *Int. J. numer. Meth. Engng* **21,** 745–756 (1984).
[60] M. C. Rivara, Selective refinement/derefinement algorithms for sequences of nested triangulations. *Int. J. numer. Meth. Engng* **28,** 2889–2906 (1990).
[61] M. C. Rivara, A 3-D refinement algorithm suitable for adaptive and multigrid techniques. *Commun. appl. numer. Meth.* **8,** 281–290 (1992).
[62] M. J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **53,** 484–512 (1984).
[63] M. J. Berger and R. J. LeVeque, An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA-CP (1989).
[64] J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, Adaptive remeshing for compressible flow computations. *J. Comput. Phys.* **72,** 449–466 (1987).
[65] J. Peraire, K. Morgan and J. Peiro, Unstructured finite element mesh generation and adaptive procedures for CFD. AGARD-CP-464, 18 (1990).
[66] R. Löhner, J. Camberos and M. Merriam, Parallel unstructured grid generation. *Comput. Meth. appl. Mech. Engng* **95,** 343–357 (1992).
[67] T. J. Baker, Developments and trends in three-dimensional mesh generation. *Appl. numer. Math.* **5,** 275–304 (1989).
[68] M. A. Yerry and M. S. Shepard, Automatic three-dimensional mesh generation by the modified-octree technique. *Int. J. numer. Meth. Engng* **20,** 1965–1900 (1984).
[69] M. S. Shepard and M. K. Georges, Automatic three-dimensional mesh generation by the finite octree technique. *Int. J. numer. Meth. Engng* **32,** 709–749 (1991).
[70] R. Tilch, Ph.D. Thesis, CERFACS, Toulouse, France (1991).
[71] C. J. Huang and S. J. Wu, Global and local remeshing algorithm for compressible flows. *J. Comput. Phys.* **102,** 98–113 (1992).
[72] T. Strouboulis and K. A. Haque, Recent experiences with error estimation and adaptivity; Part 2: error estimation for *H*-adaptive approximations on grids of triangles and quadrilaterals. *Comput. Meth. appl. Mech. Engng* **97** (1992).
[73] J. Peraire, K. Morgan and J. Peiro, Adaptive remeshing in 3-D. *J. Comput. Phys.* (1992).
[74] J. D. Baum and R. Löhner, Numerical simulation of pilot/seat ejection from an F-16. AIAA-93-0783 (1993).
[75] N. K. Winsor, S. A. Goldstein and R. Löhner, Numerical modelling of inerior ballistics from initiation through projectile launch. *Proc. 42nd Aeroballistic Range Association Meeting*, Adelaide, Australia (22–25 October 1991).
[76] The FEFLO Family of CFD Codes. *Proc. 2nd Int. ESI PAM-Workshop*, Paris (Nov. 1991).

[77] M. van Dyke, *An Album of Fluid Motion*, p. 122. Parabolic Press (1989).
[78] R. Biswas and R. Strawn, A new procedure for dynamic adaptation of three-dimensional unstructured grids. AIAA-93-0672 (1993).
[79] P. L. George, *Automatic Mesh Generation*. John Wiley, New York (1991).