JOURNAL OF COMPUTATIONAL PHYSICS 72, 449–466 (1987)

Adaptive Remeshing for Compressible Flow Computations

J. PERAIRE, M. VAHDATI, K. MORGAN, AND O. C. ZIENKIEWICZ

Institute for Numerical Methods in Engineering, University College, Swansea SA2 8PP United Kingdom

Received May 2, 1986; revised September 10, 1986

An adaptive mesh procedure for improving the quality of steady state solutions of the Euler equations in two dimensions is described. The procedure is implemented in conjunction with a finite element solution algorithm, using linear triangular elements, and an explicit time-stepping scheme. The mesh adaptation is accomplished by regeneration using information provided by the computed solution on the current mesh. The meshes produced are characterised by the appearance of stretched elements in the vicinity of one-dimensional flow features and can exhibit a large variation in element size. This allows solutions of enhanced quality to be produced in a computationally efficient manner.

1. INTRODUCTION

Numerical methods used for the analysis of problems involving compressible flow must be capable of providing adequate definition of the narrow regions of high gradients (e.g., shocks) which frequently occur and which are normally found to be embedded in large areas in which the flow variables vary slowly. As the location of these high gradient regions is not known to the analyst a priori, it is apparent that adaptive mesh methods, with a posteriori error estimators, will have an important role to play in the development of efficient solution techniques for such problems. To date, successful grid adaptation techniques have been based on either mesh enrichment or mesh movement. These techniques have generally been implemented in conjunction with explicit time integration procedures and have been designed for the analysis of steady state problems.

Mesh movement can be accomplished by advancing the solution towards steady state and, at certain times, replacing the mesh sides by springs of a certain stiffness. The nodes are then moved until the spring system is in equilibrium. The spring strengths are normally based on the local solution gradient. In two dimensions, such techniques have been used with finite volume methods on quadrilaterals [1] and finite element methods on triangles [2]. Since new nodes are not added, and nodal connectivities are not changed, a drawback of this method is that the accuracy of the final computation is limited by the structure and resolution of the initial grid.

Mesh enrichment algorithms have been used in finite element methods with

0021-9991/87 $3.00

triangles and tetrahedra [3, 4] and finite volume methods with quadrilaterals [5]. The basic approach is to advance the solution toward steady state on an initial (coarse) grid and then to use an error or refinement indicator to mark the computational cells which should be refined. The marked cells are automatically subdivided, the computation proceeds and the process is repeated until the analyst is satisfied with the solution quality. Although this method has proved to be effective in practice, it also suffers from certain drawbacks, e.g., in the solution of two-dimensional problems the areas in the vicinity of one-dimensional flow features, such as shocks and boundary layers, are not refined in an efficient manner and the number of elements employed increases rapidly with each refinement.

Recently, Löhner and Morgan [6], in the context of the finite element method, have attempted to reduce some of these problems by implementing a process which achieves directional refinement and which also removes elements from those regions of the flow where the error indicator is small. Although the programming of this refinement algorithm was complex, the quality of the results produced encouraged further research.

In this paper we describe an approach in which the adaptive refinement of the grid is accomplished by means of a remeshing procedure which is based upon information provided by the computed solution. The two-dimensional compressible flow equations are solved by using an explicit finite element method [7] with a triangular mesh and the remeshing is performed by a mesh generator which allows a significant variation in the mesh spacing through the region of interest. The mesh generator also allows efficient definition of one-dimensional flow features by generating elements which may be stretched in the direction of the feature. The performance of the proposed approach is illustrated by using it to solve problems of regular shock reflection at a wall, supersonic expansion flow around a corner, and hypersonic flow past a blunt body at high angle of attack.

2. Generation of the Initial Mesh

2.1. *The Background Grid*

The problem of generating a mesh over a two-dimensional region of arbitrary shape is considerably simplified if unstructured triangular meshes are employed [8, 9]. For the method to be described here, this process is started by constructing by hand a coarse background grid of 3-noded triangular elements which completely covers the solution domain of interest. This is illustrated in Fig. 2.1 which shows a possible background grid consisting of only four elements, for a problem of expansion flow around a corner. For the elements to be generated, it is convenient to define a node spacing δ, the value of a stretching parameter s and a direction of stretching $\boldsymbol{\alpha}$. The generated elements will then have typical length $s\delta$ in the direction parallel to $\boldsymbol{\alpha}$ and a typical length δ in the direction normal to $\boldsymbol{\alpha}$ (see Fig. 2.2). The background grid is used to provide a piecewise linear spatial distribution for these

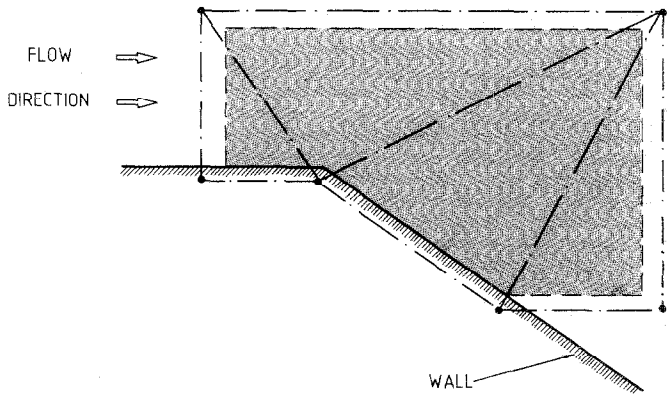

FIG. 2.1. Coarse background grid used to cover the computational domain (shaded) for a problem of supersonic expansion around a corner.

parameters over the grid to be generated. Thus, at each node on the background grid, the nodal values of δ, s, and $\boldsymbol{\alpha}$ must be specified. During the generation process the local values of these quantities will be obtained by linear interpolation, over the triangles of the background grid, between the specified nodal values. For the initial mesh, the location of one-dimensional features is not known in general and so the value $s = 1$ (i.e., no stretching) is normally specified. The node spacing δ can also be defined to be uniform but a variation of δ can be achieved (by suitable construction of the background grid) if it is apparent that increased mesh resolution

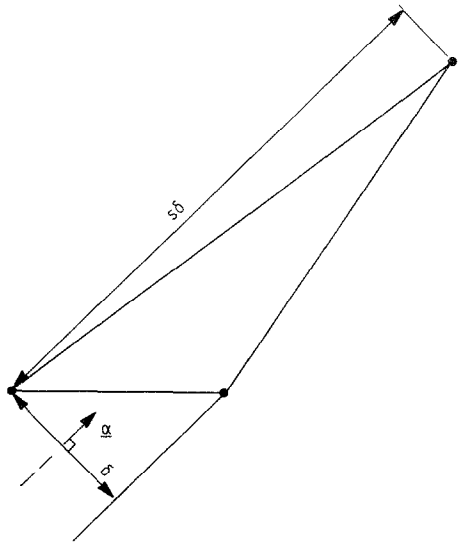

FIG. 2.2. The definition of the mesh parameters δ, s, and $\boldsymbol{\alpha}$.

is required in certain regions of the flow domain, e.g., in the vicinity of the corner in the problem of Fig. 2.1. Note that if δ is required to be uniform initially and no stretching is to be specified, then the background grid need only consist of a single element which completely covers the solution domain.

2.2. *Generation of Boundary Nodes*

The boundary of the solution domain is represented by the union of closed loops of curved segments and boundary nodes are placed at the points of intersection of these segments. For simply connected regions there is only one closed loop, whereas for multi-connected regions there will be as many internal loops as the number of openings inside the domain. The segments of the exterior boundary are defined in an anti-clockwise manner while the segments of interior boundaries are specified in a clockwise fashion. This means that, as the boundary curve is traversed, the region to be triangulated always lies to the left. Before beginning the process of generating triangles within the region of interest, the positioning of additional nodes on the boundaries of the region has to be performed. Each boundary segment is considered in turn and nodal points are generated on the boundary segments, with the spacing of the points being determined by interpolated values of δ, s, and $\boldsymbol{\alpha}$.

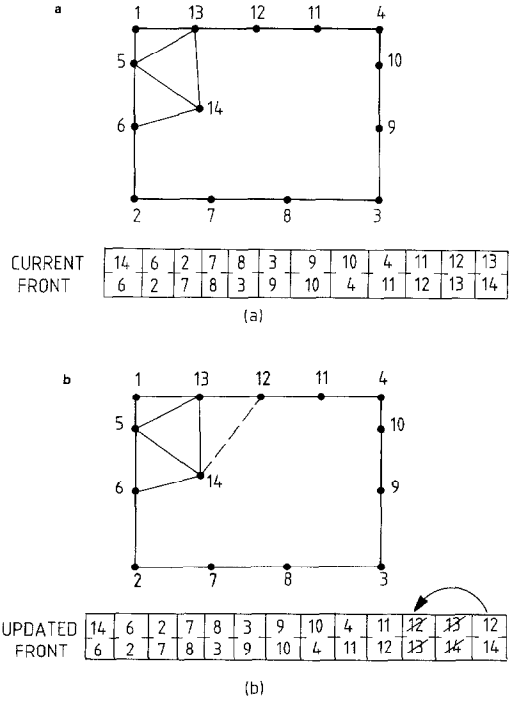

FIG. 2.3. Updating the front during the triangle generation process: (a) The form of the front at a certain stage. (b) The updated front following the generation of a new element.

2.3. *Triangle Generation*

The mesh generation algorithm utilises the concept of a generation front in a form which is very similar to that proposed by Lo [9]. At the start of the process the front consists of the sequence of straight line segments which connect consecutive boundary nodes. During the generation process, any straight line segment which is available to form an element side is termed active, whereas any segment which is no longer active is removed from the front. Thus, while the domain boundary will always remain the same, the generation front will change continuously and has to be updated whenever a new element is formed. This updating process is illustrated in Fig. 2.3.

The following steps are involved in the process of generating a new triangle in the mesh:

(a) An active side in the front is chosen as a base. If large variations in δ are present in the background grid then it is advantageous to look for the smallest active side, but if δ is constant or varying slowly the last active side in the front is used.

(b) Suppose the chosen side joins nodes A and B. Determine the local mesh parameters δ_M, s_M, and $\boldsymbol{\alpha}_M$ at the mid-point M of AB by interpolating over the background grid. Make a local rotation of coordinates so that $\boldsymbol{\alpha}_M$ lies along the x_1 axis and scale the x_1 coordinate by a factor s_M. In the new coordinate system, a triangle which is as regular as possible will be generated.

(c) Determine δ_1 according to

$$\delta_1 = \begin{cases} 0.55\,AB & \delta_M < 0.55AB \\ \delta_M & 0.55AB < \delta_M < 2AB \\ 2AB & 2AB < \delta_M. \end{cases} \tag{2.1}$$

The inequalities used here are necessary to ensure geometrical compatibility and to ensure that elements with excessive distortion are not generated. Different inequalities can be devised but the values shown have worked well in practice. Now construct the point C which is distance δ_1 from A and from B.

(d) Determine all the active nodes which lie within the circle with centre at C and radius nAB. (There is no unique choice for the value of n which should be adopted, but the value $n=5$ has been used for the computations reported in this paper.) These nodes are ordered according to their distance from C, with the first nodes in the list being the closest to C, and are denoted by $N_1, N_2, \ldots, N_p$.

(e) Place C at the head of this list unless

$$AN_1 < 1.5\delta_1 \quad \text{and} \quad BN_1 < 1.5\delta_1. \tag{2.2}$$

(f) The required connecting point N_j is then taken to be the first node in the list which is such that the interior of the triangle ABN_j does not contain any other

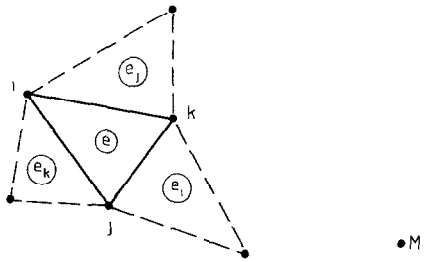

FIG 2.4. The searching algorithm to locate point M on the background grid. The area coordinates L_i, L_j, and L_k of element e are evaluated at M. Here $L_i(M) < L_k(M) < L_j(M)$ and the next element to be checked is element e_i.

node N_i in the list (excluding C) and such that the line MN_j does not intersect any existing sides in the front. The new element is formed, the coordinates are transformed back to the original space, and the front is updated as indicated previously.

The triangle generation process ceases when the number of active sides in the front is reduced to zero.

2.4. *Searching Algorithm*

Each time the values of δ_M, s_M, and $\boldsymbol{\alpha}_M$ are required during stage (b) above, they have to be obtained by locating point M within an element of the background grid. An efficient search algorithm has been implemented which requires, for each element e of the background grid, the knowledge of the three surrounding elements which have sides in common with element e. Given the coordinates of M and a starting element of the background grid, the three area coordinates [10] of M are determined. If each area coordinate lies between zero and one then the element contains the point M. If not, the node for which the area coordinate is a minimum (see Fig. 2.4) is found and this indicates the next element to be checked. In this manner, the necessity of searching over all the elements in the background grid is avoided.

3. SOLUTION OF THE EULER EQUATIONS

The equations governing two-dimensional compressible flow are considered in the form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_j}{\partial x_j} = \mathbf{0}, \tag{3.1}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho \varepsilon \end{pmatrix}, \qquad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho\varepsilon + p) \end{pmatrix}. \tag{3.2}$$

Here ρ, p, and ε denote the density, pressure, and specific total energy of the fluid, respectively, and u_i is the component of the fluid velocity in the direction x_i. The equation set is completed by the addition of the state equation

$$p = (\gamma - 1)\,\rho(\varepsilon - 0.5 u_j u_j), \tag{3.3}$$

where γ is the ratio of the specific heats.

An explicit finite element procedure for the solution of Eq. (3.1) has been widely reported [3, 11] and has recently been improved by the incorporation of a flux-corrected transport [FCT] algorithm [7, 12]. For the steady state problems to be considered in this paper, the FCT algorithm has been employed and the convergence is accelerated by using local timesteps [13].

4. Adaptive Remeshing

The procedures outlined above enable an initial approximation to the steady state solution to be obtained for a given problem. The solution quality can be improved by adaptively refining the mesh [3–6]. Here this mesh adaptation is achieved by using the computed solution to determine "optimum" nodal values for δ, s, and $\boldsymbol{\alpha}$. The mesh is then regenerated with the initial computational mesh now acting as a background grid. To determine the values for the mesh parameters, it is necessary to use the initial solution to give some indication of the error magnitude and direction. The Euler equations are expressed in terms of a vector unknown $\mathbf{U}$, whereas the development of a practical error indicator is most easily assomplished in terms of a single scalar function. Thus, for the Euler system, a certain "key" scalar variable should be identified and then the error indication process can be performed solely in terms of this variable. In this paper we follow previous work [3, 5, 6, 11] and base the error indication on the density variable ρ.

The construction of the error estimator can take various forms depending upon the error norm that we wish to consider. To simplify matters, we shall discuss the problem assuming initially one-dimensional behaviour in which the variable $\rho(x_1)$ is approximated by $\hat{\rho}(x_1)$, using linear interpolation. The first derivative of the approximation will be discontinuous in each element but, by projection or variational recovery [14], its nodal values can be approximated. The second derivative can be evaluated in a similar fashion.

The knowledge of the second derivative within an element e allows a determination of the local error E_e, defined by

$$E_e = \rho - \hat{\rho}, \tag{4.1}$$

provided we assume that the nodal error is zero. For then, to the next order of

approximation, this error is simply the departure of the quadratic from the linear approximation and can be expressed as

$$E_e = \frac{1}{2} x_1 (h_e - x_1) \left. \frac{d^2\hat{\rho}}{dx_1^2} \right|_e \tag{4.2}$$

where here the origin of x_1 has been placed at one end of the element and h_e denotes the element length. The root mean square value, E_e^{RMS}, of the error over the element can then be computed as

$$E_e^{\mathrm{RMS}} = \left\{ \int_0^{h_e} \frac{E_e^2 \, dx_1}{h_e} \right\}^{1/2} \approx \frac{1}{11} h_e^2 \left| \frac{d^2\hat{\rho}}{dx_1^2} \right|_e . \tag{4.3}$$

In many other situations [15], it has been demonstrated that equal distribution of the element error leads to an optimal mesh and, in what follows, we employ the same criterion. The simple measure of Eq. (4.3) gives an estimate of the error in each element and this is used to guide the refinement process by the requirement that the error should be equal in all elements, i.e.,

$$h_e^2 \left| \frac{d^2\hat{\rho}}{dx_1^2} \right|_e = \text{constant}. \tag{4.4}$$

With the second derivative evaluated at each node P on the current mesh, Eq. (4.4) suggests that the new mesh be generated with the local spacing δ_P determined according to the condition

$$\delta_P^2 \left| \frac{d^2\hat{\rho}}{dx_1^2} \right|_P = \text{constant}. \tag{4.5}$$

For two-dimensional problems involving a density function $\rho(x_1, x_2)$, a matrix $\mathbf{m}_P$ of second derivatives can be constructed at each node P on the current mesh such that

$$m_{ij}|_P = \left. \frac{\partial^2\hat{\rho}}{\partial x_i \, \partial x_j} \right|_P . \tag{4.6}$$

The local principal directions X_1 and X_2 are determined and the corresponding quantities

$$\lambda_{1P} = \left. \frac{\partial^2\hat{\rho}}{\partial X_1^2} \right|_P , \qquad \lambda_{2P} = \left. \frac{\partial^2\hat{\rho}}{\partial X_2^2} \right|_P , \qquad |\lambda_{1P}| > |\lambda_{2P}| \tag{4.7}$$

are obtained. Once this is done, the one-dimensional process outlined above is applied in each direction separately, leading to the requirement that

$$\delta_{(1)P}^2 \, |\lambda_{1P}| = \delta_{(2)P}^2 \, |\lambda_{2P}| = \text{constant} = \delta_{\min}^2 \, |\lambda_1|_{\max}, \tag{4.8}$$

where $\delta_{(1)P}$ and $\delta_{(2)P}$ denote node spacings in the X_1 and X_2 directions, respectively, $|\lambda_1|_{\max}$ is the maximum value of $|\lambda_{1P}|$ over each node in the current mesh, and $\delta_{\min}$ is a user-specified minimum value for δ in the new mesh. The magnitude of the local stretching parameter s_P is simply defined to be the ratio of the two spacings, i.e.,

$$s_P = \left(\frac{|\lambda_{1P}|}{|\lambda_{2P}|}\right)^{1/2}. \tag{4.9}$$

It is apparent that, in regions of uniform flow, the computed values of δ_P will be very large. Practical mesh generation constraints therefore require the user to specify a maximum allowable value $\delta_{\max}$ for the local spacing on the new mesh. Then, if δ_P computed according to Eq. (4.8) is such that $\delta_P > \delta_{\max}$, the value of δ_P is set equal to $\delta_{\max}$. Similarly, a maximum allowable value $s_{\max}$ for the stretching on the new mesh is prescribed by the user, i.e., if the computed local stretching s_P exceeds $s_{\max}$, then s_P is replaced by $s_{\max}$. It should be noted that Eq. (4.9) is such that high stretching can be expected only in the vicinity of one-dimensional flow features with low curvature.

The mesh is regenerated according to the computed distribution of the mesh parameters and the solution of the problem recomputed on the new mesh. The process can be repeated on a sequence of meshes until the analyst is satisfied with the quality of the solution produced. The increase in definition of the flow features is achieved by decreasing the value of $\delta_{\min}$. The value of $\delta_{\min}$ is the major parameter governing the number of elements in the new mesh. For the problems considered here, $\delta_{\min}$ is reduced by a factor of $2 \sim 3$ in each remeshing and it is found that the number of degrees of freedom on the new mesh is not much larger (sometimes even smaller) than the number employed in the original mesh. More sophisticated methods for choosing $\delta_{\min}$ can obviously be devised to automatically control the number of elements in the new mesh.

5. Examples

In this paper we will demonstrate the numerical performance of the proposed approach by considering the solution of three different problems.

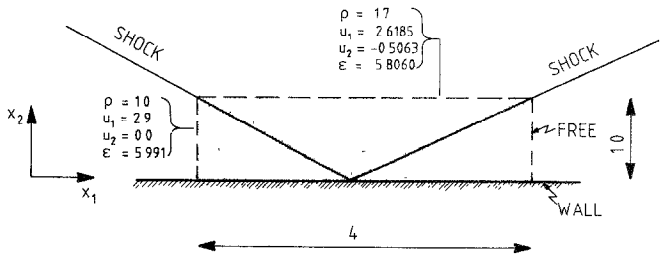

FIG. 5.1. Regular shock reflection at a wall: the problem definition and the computational domain (shaded).

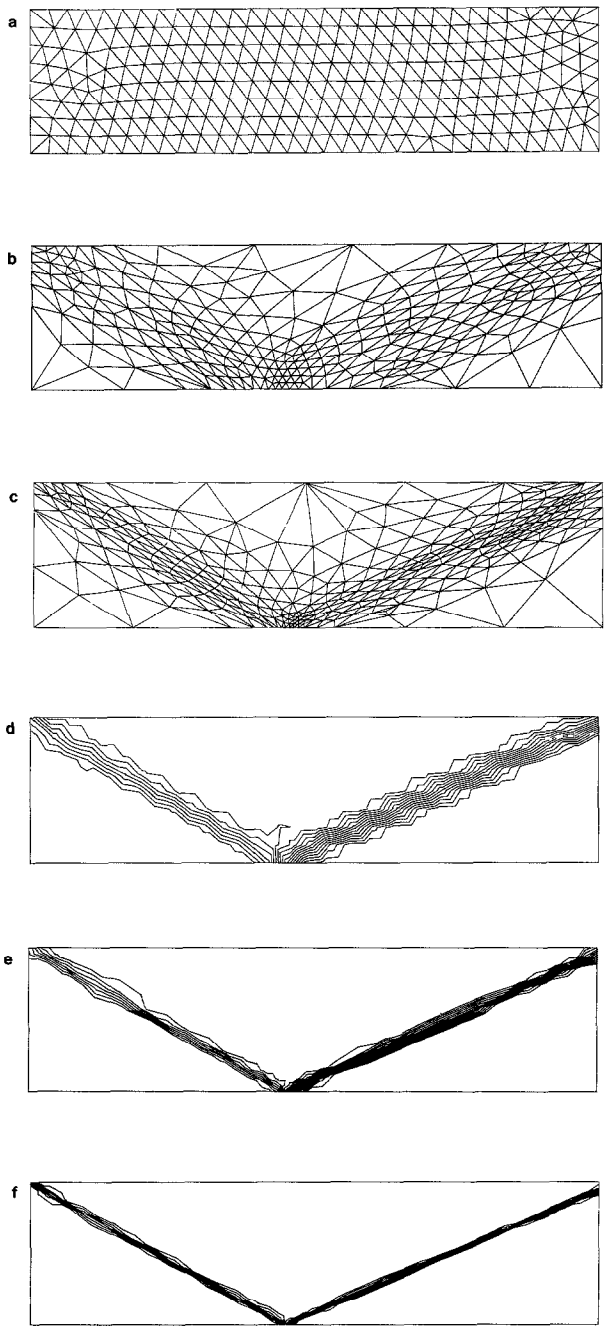

FIG. 5.2. Regular shock reflection at a wall: the sequence of meshes employed and the corresponding pressure contours.

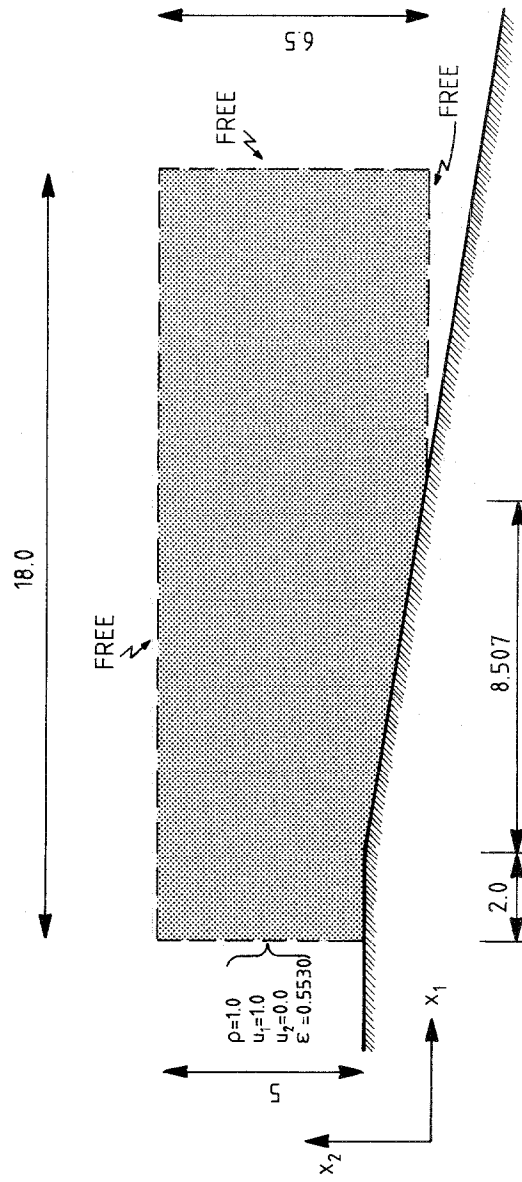

FIG. 5.3. Supersonic expansion around a corner: the problem definition and the computational domain (shaded).

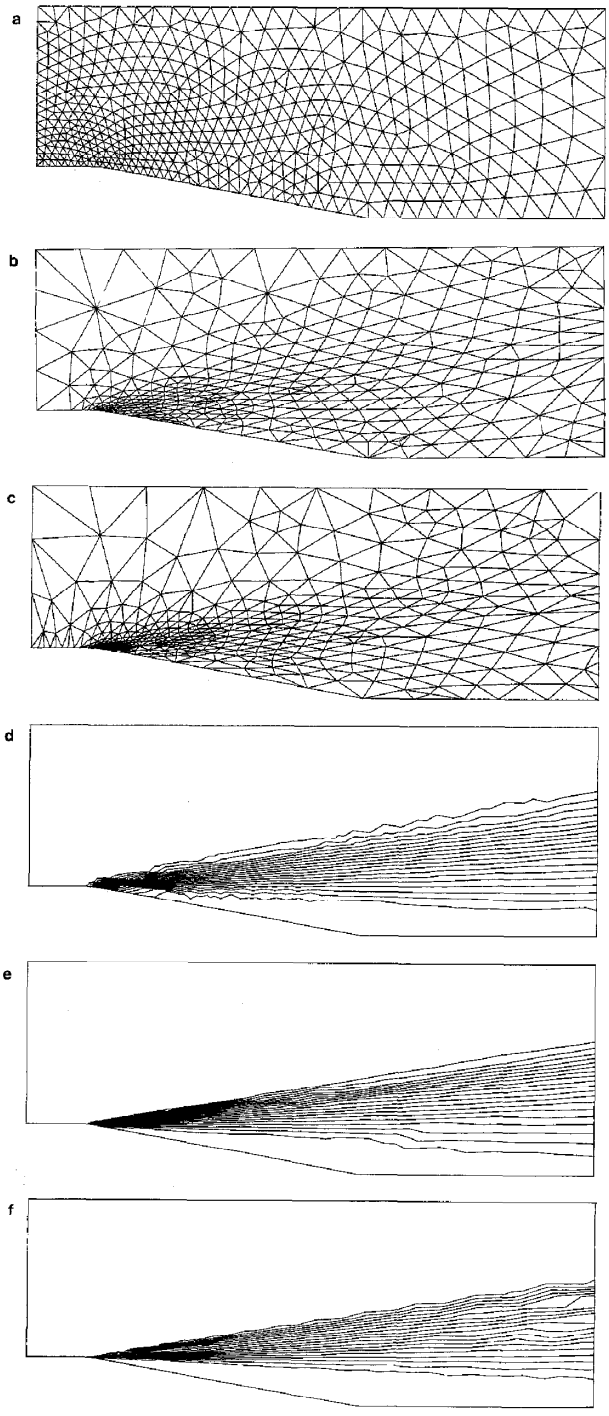

FIG. 5.4. Supersonic expansion around a corner: the sequence of meshes employed and the corresponding density contours.

TABLE I

Details of the Sequence of Meshes Employed
for the Problem of Regular Shock Reflection at a Wall

Mesh	Elements	Nodes	δ_{min}	δ_{max}	s_{max}
1	478	279	0.13	0.13	1.0
2	479	265	0.05	0.60	3.0
3	528	285	0.02	0.75	6.0

(a) Regular shock reflection at a wall. This is a standard test problem and has been widely discussed in the literature. The problem definition is given in Fig. 5.1 and the sequence of meshes employed with the corresponding computed pressure contour distributions are shown in Fig. 5.2. Details of the meshes used are shown in Table I.

(b) Supersonic expansion around a corner. The case considered is that of a flow at Mach 6 with $\gamma = 1.38$. The problem definition is shown in Fig. 5.3 and the sequence of meshes used together with the corresponding computed density contour distribution is given in Fig. 5.4. The variation of the density along the wall in each case is shown in Fig. 5.5. Details of the meshes used are shown in Table II and it can be observed that the initially generated mesh contains 977 elements whereas the final mesh has only 758 elements.

(c) Hypersonic flow past a blunt body at high angle of attack. The problem definition is given in Fig. 5.6 and represents a Mach 25 flow at 20° angle of attack. The sequence of meshes employed and the corresponding density and pressure contours are shown in Fig. 5.7. The distribution of velocity vectors at steady state on the final mesh is shown in Fig. 5.8. Details of the meshes used are given in Table III. For this problem, the generation of the final mesh required 4.15 min of CPU time on a VAX-750 while the flow analysis reduced the residual by three orders of magnitude in 72 min on the same mesh. This indicates the small relative computational effort spent in the regeneration process.

TABLE II

Details of the Sequence of Meshes Employed
for the Problem of Supersonic Expansion around a Corner

Mesh	Elements	Nodes	δ_{min}	δ_{max}	s_{max}
1	977	538	0.20	1.00	1.0
2	599	329	0.10	1.50	3.0
3	758	408	0.05	2.00	4.5

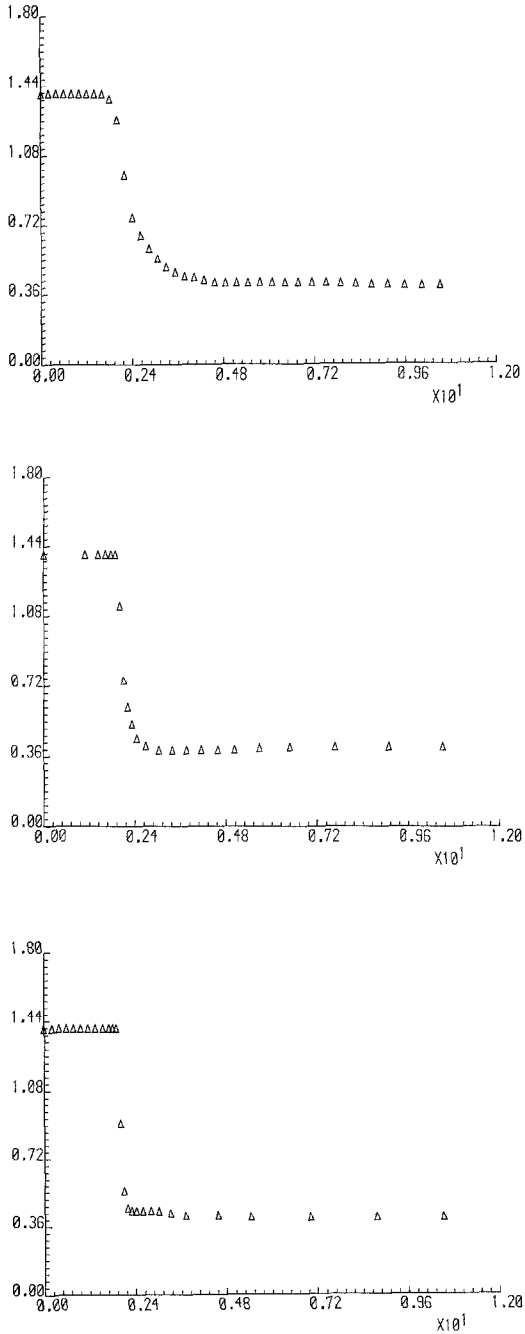

FIG. 5.5. Supersonic expansion around a corner: the variation of the density along the wall for each mesh employed.

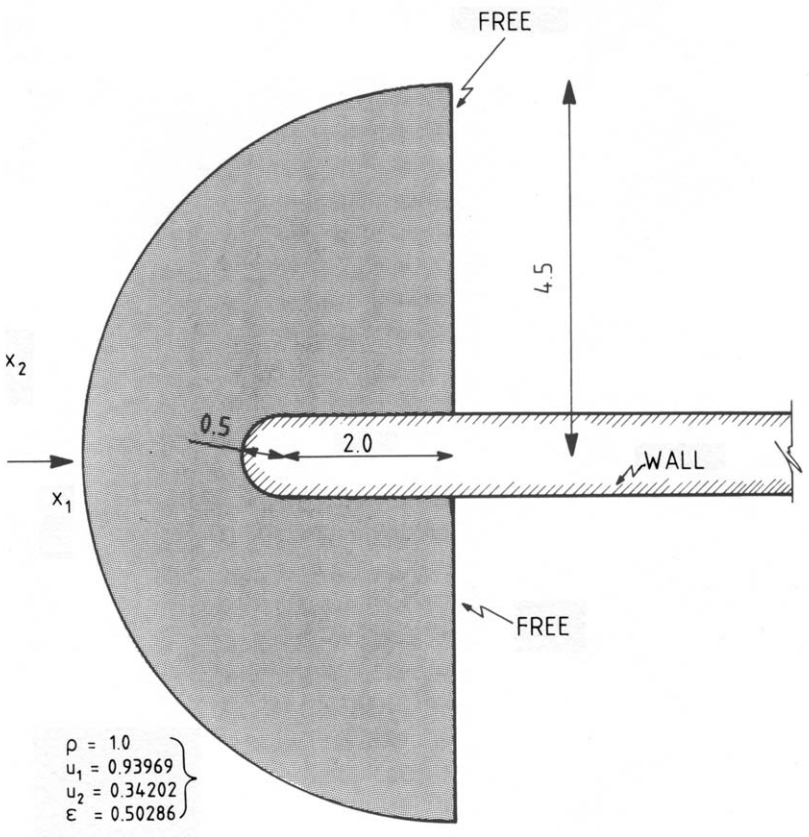

FIG. 5.6. Hypersonic flow past a blunt body at high angle of attack: the problem definition and the computational domain (shaded).

TABLE III

Details of the Sequence of Meshes Employed for the Problem of Hypersonic Flow Past a Blunt Body at High Angle of Attack

Mesh	Elements	Nodes	$\delta_{\min}$	$\delta_{\max}$	$s_{\max}$
1	978	547	0.25	0.25	1.0
2	696	383	0.10	1.20	3.0
3	1574	821	0.04	2.50	4.5

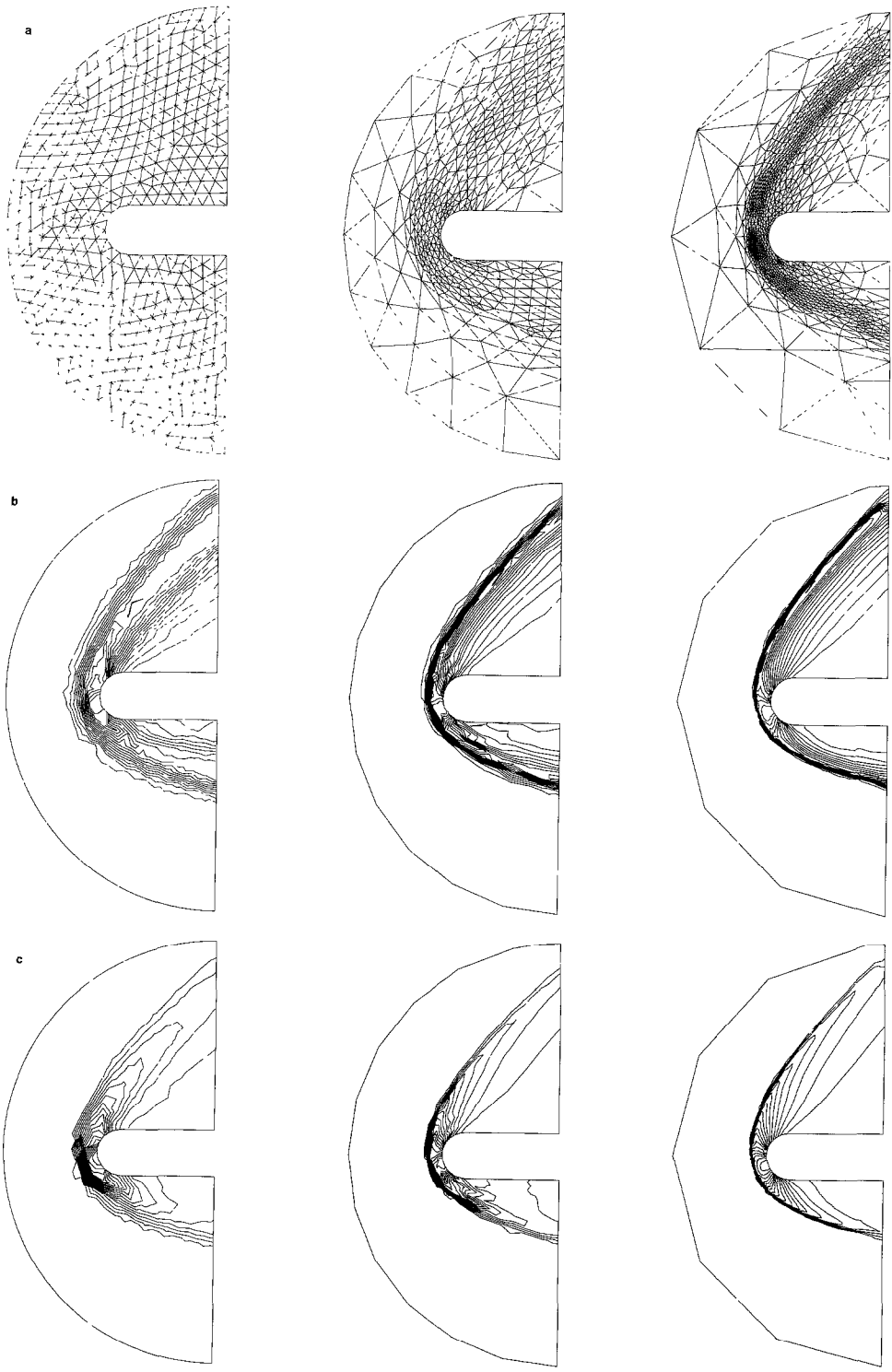

FIG. 5.7. Hypersonic flow past a blunt body at high angle of attack: (a) the sequence of meshes employed and the corresponding (b) density and (c) pressure contours.

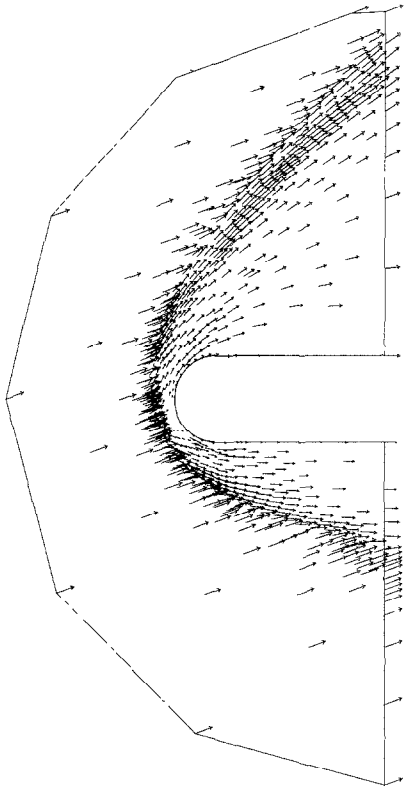

FIG. 5.8. Hypersonic flow past a blunt body at high angle of attack: the velocity vector distribution on the final mesh.

6. CONCLUSIONS

We have successfully demonstrated the use of an adaptive mesh regeneration procedure for the solution of two-dimensional problems involving compressible high speed flow. A full analysis of the influence of the highly irregular grids, produced by adaptation, on the accuracy of the solution has not been performed. However, the numerical computations here presented indicate that in such cases good accuracy can still be maintained. An essential feature of the procedure is its ability to gradually improve the quality of the solution without necessarily significantly increasing the total number of unknowns at each stage. We are now beginning to apply these ideas to the solution of three-dimensional problems and it is expected that for such problems this feature will prove to be even more important.

ACKNOWLEDGMENTS

The authors thank the Aerothermal Loads Branch of the NASA Langley Research Center for partial support of this work under Grant NAGW-478 and especially A. R. Wieting and K. S. Bey for their continued interest and encouragement. M. Vahdati gratefully acknowledges the support, in the form of a Research Studentship, of the U.K. Science and Engineering Research Council.

REFERENCES

1. P. A. GNOFFO, *AIAA J.* **21**, No. 9, 1249 (1983).
2. R. LÖHNER, K. MORGAN, AND O. C. ZIENKIEWICZ, "Adaptive Grid Refinement for the Compressible Euler Equations," *in Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, edited by I. Babuska *et al.* (Wiley, New York, 1986), p. 281.
3. R. LÖHNER, K. MORGAN, J. PERAIRE, AND O. C. ZIENKIEWICZ, AIAA Paper 85-1531-CP, 1985 (unpublished).
4. F. ANGRAND, V. BILLEY, A. DERVIEUX, J. PERIAUX, C. POULETTY, AND B. STOUFFLET, AIAA Paper 85-1706, 1985 (unpublished).
5. J. F. DANNENHOFFER AND J. R. BARON, AIAA Paper 85-0484, 1985 (unpublished).
6. R. LÖHNER AND K. MORGAN, AIAA Paper 86-0499, 1986 (unpublished).
7. R. LÖHNER, K. MORGAN, M. VAHDATI, J. P. BORIS, AND D. L. BOOK, University College of Swansea Report C/R/539/86, 1986 (unpublished).
8. J. C. CAVENDISH, *Int. J. Numer. Methods Engng.* **8**, 679 (1974).
9. S. H. LO, *Int. J. Numer. Methods Engng.* **21**, 1403 (1985).
10. O. C. ZIENKIEWICZ AND K. MORGAN, *Finite Elements and Approximation* (Wiley, New York, 1983).
11. R. LÖHNER, K. MORGAN, AND O. C. ZIENKIEWICZ, *Comput. Meth. Appl. Mech. Engng.* **51**, 441 (1985).
12. K. MORGAN, R. LÖHNER, J. R. JONES, J. PERAIRE, AND VAHDATI, in *Proceedings, 6th International Symposium on Finite Element Methods in Flow Problems, Antibes, France, 1986*, edited by M. O. Bristeau *et al.* (INRIA, Paris, 1986), p. 89.
13. A. JAMESON AND W. SCHMIDT, *Comput. Meth. Appl. Mech. Engng.* **51**, 467 (1985).
14. R. LÖHNER, K. MORGAN, AND O. C. ZIENKIEWICZ, Lecture Notes in Physics Vol. 218 (Springer-Verlag, Berlin, 1985), p. 388.
15. J. T. ODEN, "Grid Optimisation and Adaptive Meshes for Finite Element Methods", University of Texas at Austin Notes, 1983 (unpublished).