NASA-CR-181,636

NASA Contractor Report 181636

NASA-CR-181636
19880019731

Adaptive Computational Methods for Aerothermal Heating Analysis

John M. Price and J. Tinsley Oden

Lockheed Missiles & Space Company, Inc.
Huntsville Engineering Center
Huntsville, AL 35807

Computational Mechanics Company
Austin, TX 78751

LIBRARY COPY

JUL 2 2 1988

LANGLEY RESEARCH CENTER
LIBRARY NASA
HAMPTON, VIRGINIA

Contract NAS1-17894

May 1988

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

NF00931

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181636		

4. Title and Subtitle
Adaptive Computational Methods for Aerothermal Heating Analysis

5. Report Date

6. Performing Organization Code

7. Author(s)

8. Performing Organization Report No.
LMSC-HEC TR F225772

10. Work Unit No.
506-43-31-03

9. Performing Organization Name and Address
Lockheed Missiles & Space Company, Inc.
Huntsville Engineering Center
4800 Bradford Blvd., Huntsville, AL 35807

11. Contract or Grant No.
NAS1-17894

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

13. Type of Report and Period Covered
Final Contractor Report
Sept. 84 to Dec. 87

14. Sponsoring Agency Code

15. Supplementary Notes
Technical Representative of the Contracting Officer: George C. Olsen, Langley Research Center.

16. Abstract

This report documents the development of adaptive gridding techniques for finite element analysis of fluid dynamics equations. The developmental work was done with the Euler equations with concentration on shock and inviscid flow field capturing. Ultimately this methodology is to be applied to a viscous analysis for the purpose of predicting accurate aerothermal loads on complex shapes subjected to high speed flow environments. The development of local error estimate strategies as a basis for refinement strategies is discussed as well as the refinement strategies themselves. The application of the strategies to triangular elements and a finite-element flux-corrected-transport numerical scheme are presented. The implementation of these strategies in the GIM/PAGE code for two-dimensional and three-dimensional applications is documented and demonstrated.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Aerothermal Heating Analysis *Adaptive Mesh Techniques* *Finite Element Methodology* *Error Estimates*	*Unclassified - Unlimited* Subject Category 34

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	*Unclassified*		

NASA FORM 1626 OCT 86

N88-29115#

CONTENTS

1. INTRODUCTION

This report is submitted in accordance with the reporting provisions of the subject contract and describes the research effort and its accomplishments for the contract period. This study was performed by personnel of the Computational Mechanics Section of the Lockheed-Huntsville Engineering Center (Lockheed-Huntsville), Huntsville, Alabama, and Computational Mechanics Company (COMCO), Austin, Texas. The Technical Representative of the NASA-Langley Contracting Officer was Dr. G.C. Olsen, Aerothermal Loads Branch, Mail Stop 395.

1.1 BACKGROUND

Finite element numerical methods are currently being used in computer codes for solving practical fluid flow problems. The advent of the supercomputer is one of the primary reasons for the success thus far. However, future problems are destined to be more complex and will no doubt tax even the fastest machines. In conjunction with the release of the next generation of supercomputers (Cyber 2xx/GF10), more powerful numerical algorithms will also be needed. Current methods utilize a grid of points to discretize the continuum which are fixed a priori and not changed during the computation. In addition, the order of the method, direction of differencing, and damping models, are all chosen by the code user.

The success of finite element and finite difference codes often depends on the user's ability to discretize the domain and/or selectively increase the order of the finite element shape function to capture strong gradients within the domain. Currently, this requires an a priori knowledge of the location and strength of sharp gradients that occur in the flow field. Even then, obtaining optimal discretization and interpolation is a lengthy and

costly iterative procedure. Strong flowfield gradients occur in shock waves, expansion regions, and viscous layers. The accurate determination of these regions is vitally important in determining the aerothermal loads on aerospace vehicles in supersonic flight. Body heating rates are particularly sensitive to the resolution of thermal gradients at the vehicle surface.

The next generation of finite element methods to impact the computational mechanics community will be the "self-adaptive" kind. In these advanced methods, logic is built into the code to choose the grid of points, move them around, choose the degree of approximation, and generally adapt itself to the physics of the flow. Not only does this provide more reliable and accurate results, but it frees the (non-expert) user from making these decisions before running the code.

1.2 OBJECTIVE

The objective of this contract is to develop new computation methods for aerothermal heating analysis which utilize adaptive strategies. The new methods will be tested initially in trial codes and then implemented in Lockheed's GIM/PAGE code. Finally, a test problem will be run and compared with experimental data for code verification.

2. WORK ACCOMPLISHED

Adaptive procedures may be placed in one of three basic categories:

1. Moving Meshes. The number of grid points is fixed, and the mesh is distorted so as to improve the quality of local approximations of the flow field and its gradient.

2. Mesh Refinement (h-method). The mesh is refined (i.e., the number of elements is increased, their dimension decreased) so that local accuracy is improved.

3. Subspace Enrichment (p-method). The local order of the approximation is increased to provide a more accurate solution. In finite element methods, the mesh is fixed while the local degree p of the polynomial shape functions is increased.

Regardless of which category an adaptive procedure falls into, it generally follows the steps shown in Fig. 2-1. The term structure refers to the basic mesh topology, the number of nodes and cells, the local order of the approximation, the numerical scheme, etc. It is the framework within which the solution is obtained. Using an initial structure, a solution is computed. The "goodness" of this solution is then determined. A measure of "goodness" can be obtained by computing a posteriori error estimates. The measure of solution "goodness" can also include such things as the cost of the solution in dollars and the manhours required to obtain solution. If the "goodness" criteria is met then a solution of a specified "goodness" has been obtained. If the "goodness" criteria are not met, then the structure of the mathematical approximation is changed in some rational manner. This may involve moving nodes, adding more nodes and cells, and increasing the local order of the approximation. A "better" solution is now computed. This process is repeated until the "goodness" criteria are met.

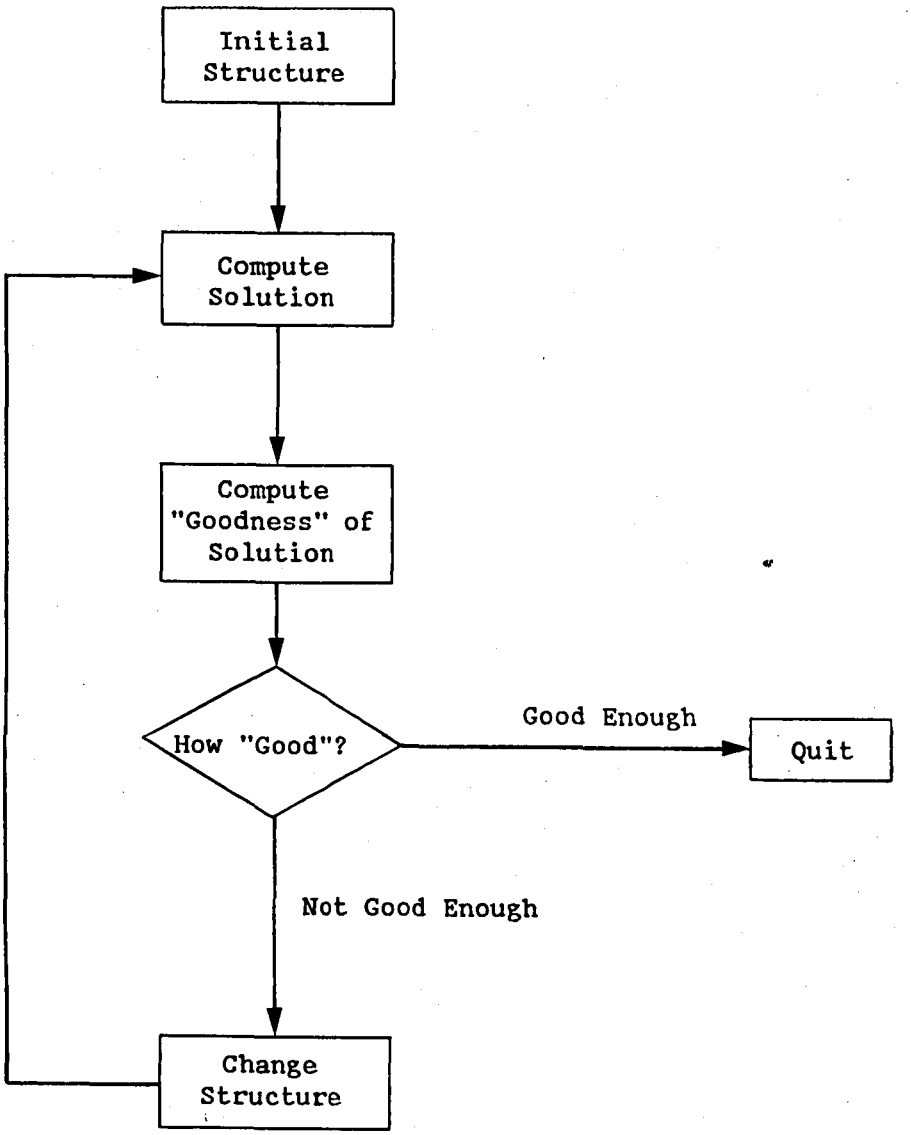


Fig. 2-1 Flow Chart of an Adaptive Procedure

This report consists of a collection of papers which document the various research efforts undertaken during this multiyear contract by Lockheed-Huntsville and COMCO personnel.

Appendix A documents recent advances in error estimation and adaptive methods for finite element calculations.

Appendix B documents the adaptive mesh strategy which is employed in several test codes as well as the GIM/PAGE code.

Appendix C documents the implementation of a class of adaptive procedures for time-dependent Euler equations in two dimensions.

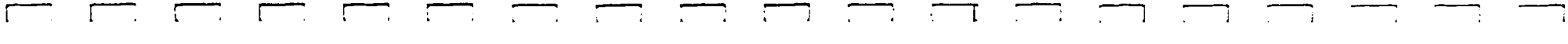
Appendix D documents implementation of an adaptive procedure which uses triangular elements and a FEM-FCT numerical scheme.

The implementation of the GIM/PAGE code with adaptivity is documented in Appendix E.

Development of a three-dimensional adaptive procedure is covered in Appendix F.

3. REMARKS AND CONCLUSIONS

Adaptive finite element methods will have a significant impact on computational fluid dynamics in the future. This report shows that adaptivity can be coupled to several numerical algorithms. Existing flow solvers can be enhanced with adaptivity. There is much work which needs to be done in the general area of overall adaptive strategy optimization. This involves the integration of both software and computer to realize an efficient analysis tool. Within a software/computer structure, a particular adaptive strategy may produce the least computationally expensive answers. This same adaptive strategy within another software/computer structure may perform very poorly. The type of data management technique may effect how an adaptive strategy performs within a particular software/computer structure. In summary, additional work should be done to determine the effect of the software/computer structure on an adaptive strategy.



Appendix A

RECENT ADVANCES IN ERROR ESTIMATION AND ADAPTIVE IMPROVEMENT OF FINITE ELEMENT CALCULATIONS

RECENT ADVANCES IN ERROR ESTIMATION AND ADAPTIVE IMPROVEMENT OF FINITE ELEMENT CALCULATIONS

J.T. Oden, P. Devloo, and M. Howe

Texas Institute for Computational Mechanics,
Department of Aerospace Engineering
and Engineering Mechanics
The University of Texas at Austin

Abstract. We collect in this article a synopsis of methods and results on adaptive finite element methods. We outline methods for constructing a-posteriori error estimates for linear and nonlinear problems in mechanics. Adaptive methods are described and a variety of numerical results are given on applications to problems in fluid mechanics.

1. INTRODUCTION

How good are the answers? What can be done to improve them? These questions arise with increasing frequency among users of modern computational mechanics codes. They are fundamental, in that such questions relate to the basic goals of computational mechanics: the use of computational methods and devices to simulate mechanical phenomena. Yet much of contemporary research in computational mechanics is concerned with a myriad of other issues which, important as they may be, do not consciously and directly focus on those primitive and fundamental questions. When one does focus on those queries, a sequence of natural constraints are met that have a profound effect on the way one approaches the development of modern codes, numerical schemes, algorithms, and data management techniques for computational mechanics applications:

Modulo natural deficiencies in the ability of the mathematical model itself to capture real physical behavior, we translate the first question into one that can be managed in mathematical terms: how accurate are the numerical solutions? The only plausible and general approach toward answering this question is to construct a-posteriori error estimates; i.e. to use the results of an initial calculation to estimate the local error in a finite element / finite difference approximation.

Having obtained an indication of "how good the answers are," one can proceed to the second question: what can be done to improve them? The answer is clearly to use adaptivity of the approximation in some way: to change the structure of the approximation to improve accuracy, where by "structure" we mean the basic mesh topology, the number and location of nodes and cells, the local order of the approximations, etc.

As is well known, there has emerged in the literature several methods for effectively altering

this structure: h-methods, in which the mesh is automatically refined to reduce error; p-methods, in which the local polynomial degree is increased; r-methods, in which a fixed number of nodal points are redistributed to reduce error; and combined methods, in which h - p, r - h, r - p, -- combinations are employed. A survey of the recent literature on such adaptive methods has been compiled by Oden and Demkowicz [20].

What is especially significant about these answers to the basic questions is that they have a great impact on the design of computational mechanics codes. To implement a rational adaptive scheme one must obey the following criteria in designing a programming strategy:

1. **Mesh Independence.** Since the mesh itself may well be changing as the solution evolves, it is necessary to have schemes which can be implemented on arbitrary unstructured or quasi-structured meshes. This first criterion makes obsolete most existing body-fitted coordinate schemes common in finite difference literature.

2. **Robustness.** Since the structure of the approximation is continually changing in an adaptive scheme, adaptive methods must be very stable under changes in mesh size, under mesh distortions, etc.

3. **Mathematical Basis.** Since a -posteriori error estimates are necessary for an effective adaptive scheme, it is necessary that a solid mathematical basis exist for the adaptive methods.

4. **Geometry Independence.** Modern computational methods, adaptive or not, must be able to cope with solution domains of arbitrary, complex geometry. The "real world" problems encountered in applications seldom have simple geometries for which many classical methods work well.

5. **Supercomputing.** The significant data management problems inherent in adaptive strategies must lend themselves to supercomputing strategies-vectorization, parallelism, etc.

6. **Efficiency.** Hopefully, when all features of an adaptive strategy are optimized in a program/computer structure, an efficient analysis tool will emerge. It is not necessary that the final product be capable of analyzing a given discretization as "fast" as possible; rather, the objective is overall optimization: to produce the best possible answers (in some sense) for a fixed level of computational effort.

In our opinion, it is very clear that only finite element methodologies can fulfill all of these criteria.

In this paper, we shall outline several recent advances in developments of the basis components of adaptive methods. We do not attempt to provide a thorough review of the literature, as this has already been the subject of a recent paper [20]. Rather, we provide summary comments in a few areas that we think stand out as important advances in the field. Naturally we are most familiar with our own efforts in this field, so we comment more fully on some of our own results.

Following this Introduction, we give a brief summary of a few recent advances in adaptive finite elements. This is followed by several sections on general ideas behind a-posteriori error estimation, h-method data management, algorithms for fluid-mechanics applications, and some new results on numerical experiments with our adaptive codes. Finally, we comment on future directions of research in this field.

2. RECENT ADVANCES

The state-of-the-art in adaptive finite element methods is adequately summarized in the volume of collected works and presentations made at the Lisbon conference of 1985. These have recently appeared under the editorship of Babuska, Zienkiewicz, Gago, and Oliveira [1]. Here one will find

information of the basic ideas of h, p, r methods together with numerous applications to problems in solid mechanics and fluid mechanics.

More recently a number of signigicant advances have been made which should be brought to view in the area of elliptic problems, we mention the important theoretical work by Guo and Babuska [14] on h-p methods. It is known that one can generally achieve a faster increase in local accuracy using p methods than h methods. By this it is meant that greater accuracy can be achieved with fewer degrees of freedom by increasing the local order p of the polynomial than by refining the mesh. This does not necessarily mean that the p methods offer a superior approach to solving elliptic problems, for one must add to this equation the significant factor of a data management scheme, which is often the life and death of an adaptive method. Babuska and his co-workers have shown, however, that the best possible approach to the accuracy problem, one leading to exponential convergence, is to simultaneously refine both h and p. The h-p methods have shown, in certain example problems, to produce exceptionally accurate results. At this writing, most of these results have been confined to one-dimensional problems and to linear elliptic problems in two-dimensions. There would appear to be some computational difficulties in extending these methods to time dependent problems, since there one must cope with the difficult issue of consistent mass matrices, stability and space-time approximation. However, it is possible that these difficulties may also be overcome with additional research.

A production finite element code based on p methods is now being promoted and sold. This is the PROBE code, and its successful implementation of the p method has already an impact on the design of linearly elastic structures, see [25]. The simple r methods produced by Diaz and Kikuchi, and Taylor [12] have been used effectively in classes of problems in which one wants to keep the number of degrees of freedom more or less constant. In particular, in problems such as metal forming simulations, where one must solve a large number of nonlinear partial differential equations, it is natural to try to achieve the best possible accuracy for a fixed number of nodal points. Some simple moving mesh algorithms have been proposed which are easy to implement and which apparently work well in two and three-dimensional problems. These have proved to be very effective for nonlinear problems in plasticity in nonlinear solid mechanics.

In general, moving mesh methods suffer from one defect: for a fixed number of nodes and fixed degree polynomial within each element, there is an inherent threshold of error which cannot be eliminated. Thus, with the exception of the work of Miller on moving finite element methods and the work mentioned above by Diaz and Kikuchi on r methods, most of the recent work on adaptive methods has focused on h-methods and p-methods.

Perhaps the most significant recent advances in adaptive finite element methods have come in the area of time dependent problems. We mention in this regard the important work of Flaherty and his co-workers (see, for example, [8]) who have developed effective numerical methods for certain classes of parabolic problems. Additional references on this subject can be found in these papers. We also mention the construction of adaptive characteristic Petrov-Galerkin methods by Demkowicz and Oden [9,10] which involves not only the construction of the local a-posteriori estimates but also the construction of near optimal schemes for nonlinear convection diffusion problems with small diffusion coefficients. These results have recently been extended to solve Euler equations in two-dimensions. [27].

One area in which adaptive methods appear to be making some in-roads is in supersonic gas dynamics and general fluid mechanics. Several effective numerical schemes have been proposed by Löhner, Morgan and Zienkiewicz [16,17,18], and the authors [19, 21]. These schemes have been used effectively to solve two-dimensional steady state and transient problems in compressible fluid mechanics.

More recently, Oden, Strouboulis and Devloo [19,23,24] have extended these methods to fluid mechanics in which moving domains are encountered. In particular, adaptive schemes have been developed for classes of problems in which flow interaction occurs due to the motion of one body or another through a flow field. Initial results on the application of adaptive methods to supersonic rotor-stator problems have produced some impressive results, some of which are

outlined later in this paper. These include examples in which adaptivity has resulted in a mesh with nearly 70 percent fewer elements than the uniform fine mesh required to produce equivalent accuracy.

We comment on some of the components of an effective adaptive scheme below.

3. A-POSTERIORI ERROR ESTIMATION

The great majority of results on a-posteriori error estimation that have appeared in the literature in recent years is restricted to linear elliptic problems; however, a great deal of precision and depth of results has been possible for problems in this class. In [21,24,9], we have described a general method for a-posteriori error estimation that is applicable to broad classes of linear and nonlinear problems, including parabolic and hyperbolic problems. Successful use of our method in determining error estimates for the finite element approximation of the Navier-Stokes equations has also been made [24].

An outline of the general method is provided by the abstract linear problem:

$$\text{Find } u \in V \text{ such that } a(u,v) = f(v) \qquad \forall\ v \in V \tag{3.1}$$

where $a(\cdot,\cdot)$ is a bilinear form on $V \times V$, V being a Banach space, and f is a linear functional on V. The Galerkin approximation of (3.1) in a finite-dimensional subspace V^h of V is characterized by the problem,

$$\text{Find } u^h \in V^h \text{ such that } a(u^h, v^h) = f(v^h) \qquad \forall\ v^h \in V^h \tag{3.2}$$

We suppose that $V \rightarrow H = H^* \rightarrow V^*$, the inclusions being dense and continuous, for a Hilbert (pivot) space H, V^* being the dual of V, etc. If $<.,.>$ denotes duality pairing on $V^* \times V$, then we generally have

$$a(u, v) = <Au, v>$$

If $V_A = \{v \in V \mid A v \in H\}$, then we also have $<Au,v> = (A u,v)_H : u,v \in V_A$ where $(\cdot,\cdot)$ is the inner product on H and $A \in \mathcal{L}(V_A, H.)$

In general, for finite element approximations, the form $a(\cdot,\cdot)$ can be expressed as the sum of contributions from an assembly of E subdomains:

$$u,v \in V:\ a(u,v) = \sum_e \{(A u,v)_e + \Gamma_e(u,v)\}$$

where $(\cdot,\cdot)$ denotes the H- inner product defined on restrictions of u and v to subdomain (element) e and $\Gamma_e(u,v)$ is the bilinear concomitant associated with boundary terms on the boundary of subdomain e.

Let $e^h = u - u^h$ denote the error and suppose that

$$\| v \|^2_A \overset{\text{def}}{=} a(v,v)$$

$$\|w\|_A = \sup_{v \in V} \frac{a(w,v)}{\|v\|_A}$$

Then

$$\|e^h\|_A = \sup_{v \in V} \frac{a(e^h,v)}{\|v\|_A}$$

$$= \sup_{v \in V} \|v\|_A^{-1} \{\sum_e (r^h, v)_e + \Gamma_e(e^h, v)\} \tag{3.3}$$

where $r^h = \Lambda u - \Lambda u^h = f - \Lambda u^h$ is the local residual. To eliminate r^h, we construct a local auxiliary problem, for a function θ_e defined by

$$a(\theta^e, v) = R_e(v); \quad e = 1,2, ..., E \tag{3.4}$$

where $R_e(v) = (r^h, v)_e + \Gamma_e(e^{h*}, v)\, e^{h*}$ being some appropriate approximation of e^h on the boundary. Setting Al_e the restriction of A over Ω_e and

$$\|\theta^e\|^2_{A,e} = a_e(\theta^e, \theta^e) = \langle Al_e\, \theta^e, \theta^e \rangle$$

introduce (3.4) into (3.3) to arrive at the a-posteriori estimate,

$$\|e^h\|_A \leq \{\sum \|\theta^e\|^2_{A,e}\}^{1/2} \tag{3.5}$$

The functions θ_e are local error indicators. Of course we do not wish to solve the E equations to obtain the θ_e. We are, thus, content to construct an approximate solution to (3.4) over some enriched subclass V_e^h of functions so as to produce approximations θ_e^h of θ_e. Several different methods of a-posteriori error estimation may result from different schemes for approximating (3.4). Alternatively, if one can derive local a-priori bounds such as $\|\theta^e\|_A \leq C \|R^e\|_*$, then (3.5) can be rewritten in terms of the residual functional R^e.

In many nonlinear problems, a step such as (3.3) may not hold, and instead, we bound the residual. For example

$$\|r^h\|_* = \sup_{v \in V} \frac{\langle Au - Au^h, v \rangle}{\|v\|} = \sup_{v \in V} \|v\|^{-1} \{\sum_e (r^h, v)_e + \Gamma_e(e^h, v)\}$$

$$\leq \{\sum_e \|\theta\|^2_{A,e}\}^{1/2}$$

We conclude this section with several remarks.

1. These examples provide global a-posteriori error (or residual) bounds in terms of local error indicators. By a special construction of test functions, truly local error estimates can be

obtained. For example, Demkowicz and Oden [9] studied a special Petrov-Galerkin method for the problem $-\varepsilon u'' + u = f$, and showed that the local error must satisfy the sharp a-posteriori estimate

$$\| e^h \|_{L^2(\Omega_e)} \leq \frac{h^2}{h^2 + \varepsilon\pi^2} \| r^h \|_{L^2(\Omega_e)}$$

where r^h is, again, the element residual.

2. For a time-dependent problem, such as

$$\int_\Omega (\partial u/\partial t + A(u)) \, v \, dx \, dy = \int_\Omega f \, v \, dx \, dy$$

for arbitrary test functions and v, and linear A, the fact that the error must be the function $e^h = u - u^h$ leads, by direct substitution, to the evolution equation,

$$\int_\Omega (\partial e^h/\partial t + A(e^h)) \, v \, dx \, dy = - \int_\Omega r^h \, v \, dx \, dy$$

Thus, using a higher order approximation E^h of e^h than that used in approximating u^h, we arrive naturally at a system of equations for the evolution of error,

$$\mathbf{M} \, \mathbf{E} + \mathbf{K} \, \mathbf{E} = \mathbf{R} \tag{3.6}$$

Various dynamic error estimators can be constructed depending on how one constructs the approximation E^h of e^h. In (3.6), **M** is the usual mass matrix associated with the approximation $E^h = \Sigma_j E_j (t) \psi_j (x)$, **E** is the vector of nodal errors E^j, **K** is the stiffness matrix, and **R** the residual vector.

3. For certain classes of problems, it is possible (or, at least, it may be assumed to be possible) to obtain an estimate,

$$\| \theta^e - \theta_h^e \|_{A,e} \leq C \| \theta_e - v^h \|_{A,e} \qquad \forall \, v^h \in V_e^h$$

where V_e^h is the special class of local test fucntions used in approximating the local auxilliary problems (3.4). Then $\| \theta_e - v^h \|_{A,e}$ may, in turn, be estimated using standard results from finite element interpolation theory (see Oden and Carey [22]). In particular, if θ^h is the interpolant of θ_e over Ω_e obtained using polynomials of degree $\leq k$, for an n - dimensional problem with quasi-uniform mesh refinements,

$$| \theta_e - \tilde{\theta}^h |_{m,q,\Omega_e} \leq c \, h_e^{n/q - n/p + k + 1 - m} \, | \theta_e |_{k+1,p,\Omega_e} \tag{3.7}$$

with $| \cdot |_{m,q,\Omega_e}$ the $W^{m,q}(\Omega_e)$ - seminorm $0 \leq p \leq \infty$, and $q = p / (p - 1)$. For the case $m = 0$, $k = 1$, $p = q = 2$, we obtain

$$\| \theta_e - \tilde{\theta}^h \|_{L^2(\Omega_e)} \leq C \, h_e^2 \, | \theta_e |_{2,2,\Omega_e}$$

and for $m = 0$, $k = 0$, $q = 1$, $p = \infty$, we have

$$\int_{\Omega_e} | \theta_e - \tilde{\theta}_h | \, d\Omega \approx h_e^2 | \theta_e - \tilde{\theta}_h |_{AVG} \leq C h_e^3 | \theta_e |_{1,\infty,e}$$

These estimates can judge the quality of the approximations of the local indicators, provided a means for computing estimates of the seminorms $| \theta_e |_{k+1,p,e}$ is developed.

4. FEATURES OF AN ALGORITHM APPROPRIATE FOR ADAPTIVE FEM

Earlier in this paper we listed criteria for the development of adaptive finite element codes for complex problems in solid and fluid mechanics. In this section, we summarize features of an adaptive code we have developed for two dimensional problems in compressible gas dynamics in which we have attempted to meet most of these criteria.

4.1 Preliminaries

We consider the motion of a perfect gas flowing through a two-dimensional domain $\Omega \subset IR^2$. If $\mathbf{U} = \mathbf{U}(x,t)$ is the vector of conservation variables with ρ the mass density, $\mathbf{m}$ the linear momentum and e the total energy, it satisfies the following weak initial -boundary value problem:

Find $\mathbf{U} \in \mathbf{V}$ such that

$$\int_0^T \int_\Omega \left(U^T \frac{\partial \phi}{\partial t} + Q(U) : \nabla \phi \right) d\Omega \, dt + \int_\Omega U_0^T \, \phi(\cdot, 0) \, d\Omega = \int_0^T \int_{\partial\Omega} F^T \phi \, ds \, dt \qquad (4.1)$$

$$\forall \phi \in \mathbf{W}$$

Here $\mathbf{Q}(\mathbf{U})$ is the Euler flux tensor,

$$Q(U) = \begin{bmatrix} m_1 & m_2 \\ \rho^{-1} m_1^2 + p(U) & \rho^{-1} m_1 m_2 \\ \rho^{-1} m_1 m_2 & \rho^{-1} m_1^2 + p(U) \\ \rho^{-1} m_1 (e + p(U)) & \rho^{-1} m_2 (e + p(U)) \end{bmatrix} \qquad (4.2)$$

$$p(U) = (\gamma - 1)(e - \rho^{-1} \, \mathbf{m} \cdot \mathbf{m} / 2)$$

where p is the thermodynamic pressure and γ is the ratio of specific heats.

Moreover,

$$\mathbf{V} = \{ v = \{ v_1, v_2, v_3, v_4 \}^T \mid v_i = v_i(x,t) \in L^\infty(0, T; L^1(\Omega)); \; i = 1,2,3,4 \} \qquad (4.3)$$

$$\mathbf{W} = \{ w = \{ w_1, w_2, w_3, w_4 \}^T \mid w_i \in C^1[\Omega, T]), \; w_i(x, T) = 0 \; ; \; i = 1,2,3,4 \} \qquad (4.4)$$

F is the actual prescribed flux through the boundary $\partial\Omega$ and the following notation is used

$$\mathbf{U}^T \frac{\partial \phi}{\partial t} = \sum_{\alpha=1}^{4} U_\alpha \frac{\partial \phi_\alpha}{\partial t}$$

$$\mathbf{Q} : \nabla \phi = \sum_{i=1}^{2} \sum_{\alpha=1}^{4} Q_{\alpha i} \frac{\partial \phi_\alpha}{\partial x_i}$$

Let us now consider an arbitrary time interval $[\tau_1, \tau_2] \subset [0, T]$ and modify the space of test functions to include functions which do not vanish at the final time, namely:

$$W^{\tau_1, \tau_2} = \{ w = \{ w_1, w_2, w_3, w_4 \}^T \mid w_i \in C^1 (\Omega \times [\tau_1, \tau_2]) ; \ i = 1,2,3,4 \}$$

Then we can state the weak-statement of the conservation laws over the space time subdomain $\Omega \times [\tau_1, \tau_2]$ as follows:

Find $\mathbf{U} \in \mathbf{V}^{\tau_1, \tau_2}$ such that

$$\int_\Omega (\mathbf{U}^T(\cdot, \tau_2) \phi (\cdot, \tau_2) \, d\Omega = \int_\Omega (\mathbf{U}^T(\cdot, \tau_1) \phi (\cdot, \tau_1) \, d\Omega$$

$$+ \int_{\tau_1}^{\tau_2} \int_\Omega (\mathbf{U}^T \frac{\partial \phi}{\partial t} + \mathbf{Q} : \nabla \phi) \, d\Omega \, dt - \int_{\tau_1}^{\tau_2} \int_{\partial\Omega} \mathbf{F}^T \phi \, d\gamma \, dt \qquad \forall \phi \in \mathbf{W}^{\tau_1, \tau_2} \qquad (4.5)$$

Here $\mathbf{V}^{\tau_1, \tau_2}$ is appropriately defined as the solution space over the strip $\Omega \times [\tau_1, \tau_2]$.

4.2 Solution Algorithm

We obtain a finite element approximation of (4.1) by partitioning the space-time domain $\Omega \times [0,T]$ into subdomain $\Omega \times [t_n, t_{n+1}]$ (with $0 = t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots t_N = T$) by discretizing each subdomain and by employing (4.5) using the discrete spaces of test and trial functions defined by the discretization. Moreover, by approximating the space-time integrals using numerical integration we get the following scheme [19]:

I. First Step:

For each element Ω_e, compute $\mathbf{U}_e^{n+1/2}$ such that,

$$\mathbf{U}_e^{n+1/2} \int_{\Omega_e} d\Omega = \int_{\Omega_e} \mathbf{U}_h^n \, d\Omega - \Delta t/2 \int_{\Omega_e} \text{div } \mathbf{Q} (\mathbf{U}_h^n) \, d\Omega \qquad (4.6)$$

II. Second Step:

Calculate $\mathbf{U}_h^{n+1}$ such that,

$$\int_\Omega \phi_h^T \mathbf{U}_h^{n+1} \, d\Omega = \int_\Omega \phi_h^T \mathbf{U}_h^n \, d\Omega + \Delta t \int_\Omega \mathbf{Q} (\mathbf{U}^{n+1/2}) : \nabla \phi_h \, d\Omega$$

$$- \Delta t \int_{\partial\Omega} \phi_h^T (Q(U_h^{n+1}) - Q(U^n))\, \mathbf{n}\, d\gamma \; - \Delta t \int_{\partial\Omega} \phi_h^T Q(U^n)\, \mathbf{n}\, d\gamma \qquad \forall\, \phi_h \qquad (4.7)$$

Here we assumed $\partial\phi_h/\partial t = 0$ (i.e. the spatial grid remains fixed), we let $\mathbf{F} = \mathbf{Q}\,\mathbf{n}$ and denote $U_h^n = U_h(\cdot, t_n)$.

Equations (4.6), (4.7) define a two-step TG/FELW (Taylor-Galerkin/Finite-Element-Lax Wendroff) method which has been introducted by Donea [13], studied by Baker et al. [2], refined by Löhner et al. [17] and others ([3], [19]). The second step of the scheme, as given in (4.7) involves a global calculation of the form:

$$\mathbf{M}\,\{\,\mathbf{U}\,\}^{n+1} = \{\,\mathbf{R}\,\} \qquad (4.9)$$

Here **M** denotes the consistent mass matrix, {R} the load vector whose definition can be easily deduced from (4.7) and $\{U\} = \{\,U_1, U_2, U_n, U_{n+1}, \ldots, U_n\}^T$ is the global vector of nodal unknowns. The inversion of the mass matrix can be performed by a Jacobi iteration [17] or a preconditioned Jacobi Conjugate Gradient [19].

The TG/FE-LW method provides us with a fast, multi-dimensional time stepping algorithm with a high resolution (high order of accuracy) in smooth regions of flow and which applies to unstructured adaptive grids. It is well known [17] that the algorithm suffers from a phenomenon of non-linear instability. To overcome this deficiency, artificial diffusion is added to stabilize the scheme in the presence of discontinuities ([18], [19]).

4.3 Flux-Corrected Transport

The theory of Flux Corrected Transport has been developed by Boris, Book and others ([4], [5], [6]) and it involves an attempt to systematically correct finite - difference transport schemes in order to avoid non-physical oscillations in the solution. Fully multi-dimensional FCT schemes have been constructed by Zalesak [26]. Recently Löhner et al. [18] presented a flux-correction procedure of the TG/FELW scheme for systems of conservation laws. In this section we give a short exposition of the FCT - TG/FE-LW algorithm which we employed in some of our adaptive calculations.

The FCT procedure consists of solving equation (4.9) by using a diffusion and an antidiffusion step. In the diffusion step a "strong" diffusion term is added to obtain a "transported and diffused" solution which is free of non-physical oscillations. In the antidiffusion step part, a "limited" amount of diffusion is subtracted from the right hand side (4.9) in order to steepen the solution at discontinuities and increase the accuracy in "smooth" regions of flow.

In particular, we have:

Step I: "Diffusion" Step

Compute $\{\,U_{td}^{n+1}\}$ from

$$\mathbf{M}\,\{\,U_{td}^{n+1}\} = \{\,\mathbf{R}\,\} + \{\,\mathbf{V}\,\} \qquad (4.10)$$

Here {V} denotes the vector of added diffusion with nodal contributions of the form:

$$V_i = \int_\Omega \left(D_x \frac{\partial \phi_{hi}}{\partial x}^T \frac{\partial U^n}{\partial x} + D_y \frac{\partial \phi_{hi}}{\partial y}^T \frac{\partial U^n}{\partial y} \right) d\Omega \qquad (4.11)$$

For a mesh of quadrilaterals we let

$$D_x = D_y = c\, A_e$$

where c is a constant and A_e denotes the area of element Ω_e.

Step II: "Antidiffusion " Step.

Compute $\{U^{n+1}\}$ as the limit of the sequence of iterates $\{ U^{n+1}_{[i]} \}$, i = 1,2,3,... defined by:

$$M_L \{ U^{n+1}_{[i+1]} - U^{n+1}_{td} \} = l \{ F \}_{[i]}$$

$$F = (M_L - M)\ \Delta U^{n+1}_{[i]} - V \qquad (4.12)$$

Here M_L denotes the lumped mass matrix and l denotes the flux limiting function which may be defined appropriately in order to prevent oscillations in the solution. In our applications we used the strategy of Zalesak [26] and Löhner et al. [18] to compute $l (F_{[i]})$.

4.4 An h Refinement / Unrefinement Strategy for Steady-State Solutions of High-Speed Compressible Flow

An adaptive procedure for steady-state solutions of equations of compressible gas dynamics involves the following steps:

For a given domain a coarse finite element mesh is defined which contains only a number of elements sufficient to model the basic geometric features of the flow domain (see Figure 1a). Each element in the initial mesh is assigned a "level" equal to zero. Then a finer mesh is generated by a bisection process, indicated in Figure 1b, in order to obtain an initial grid with the "group" structure. Note that when an element is refined a group of 4 elements is defined and each of the 4 new elements has a level one unit higher than the "parent" element.

1. For a given finite element grid determine the steady-state solution.

2. Compute error indicators θ_e over all M elements in the grid. Let

$$\theta_{MAX} = \max_{1 \le e \le M} \theta_e$$

3. We scan groups of 4 elements and compute

$$\theta^m_{GROUP} = \sum_{k=1}^{4} \theta_{m_k}$$

where m_k is the k-th element in group m.

4. Error tolerances are given by two real numbers, $0 < \alpha, \beta < 1$.

If $\theta_e \geq \beta\, \theta_{MAX}$

we **refine** element Ω_e by bisecting it into four new elements.

If $\theta^{m}_{GROUP} \leq \alpha\, \theta_{MAX}$

we **unrefine** group m by replacing this group with a single new element with the nodes coincident with the corner nodes of the group.

5. Go to step 1.

4.5 Numerical Examples

In this section we present examples of adaptive calculations of steady-state solutions of problems in high speed compressible flow. The error indicator employed in the numerical examples is given by the normalized gradient of the density:

$$\theta_e = A_e^{1/2}\, \frac{\max_{i=1,2} \left| \frac{\partial \rho_h}{\partial x_i} \right|}{\rho_h}$$

where ρ_h denotes an average value of the density of element Ω_e.

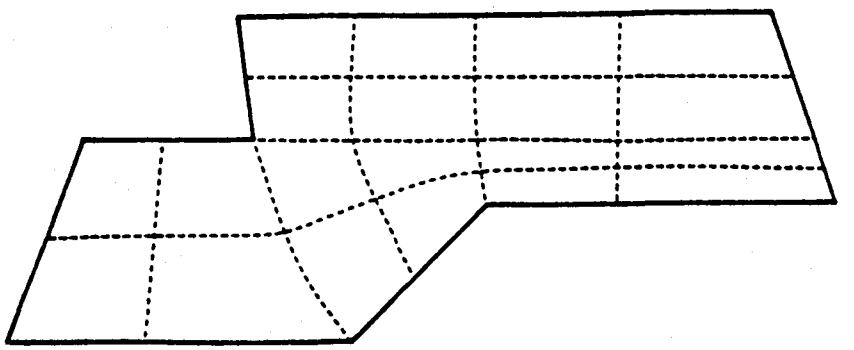
4.5.1 Supersonic Flow Over a 20° Ramp

We consider the problem of a Mach 3 flow (with $\gamma = 1.40$) over a 20° ramp. The gas enters with uniform flow conditions through the left boundary of the domain and develops an oblique shock at the root of the ramp.

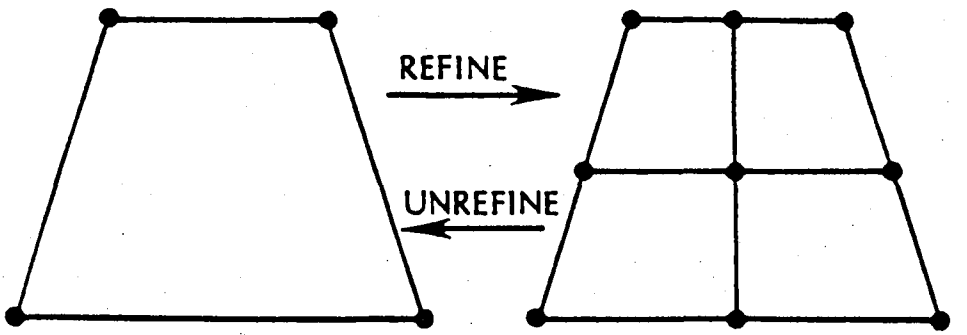
A coarse initial mesh with the computed pressure contours are illustrated in Fig. 2. Adaptive mesh results are shown in Figures 3 and 4 with one and two levels of refinement respectively. The constants for the adaptive scheme were chosen $\alpha = 0.05$, $\beta = 0.15$. The FCT version of the time-stepping algorithm was employed with $c = 0.125$. The results compare well with the exact solution except for some small disturbances downstream which are due to the artificial stagnation point at the tip of the corner. A three-dimensional view of the pressure is shown in Figure 5.

4.5.2 Supersonic Flow in Expansion Corner.

In this example, the steady supersonic flow through a 10° expansion is studied. The inflow Mach number was selected $M_\infty = 6$ with $\gamma = 1.38$. Figures 6 through 8 show the meshes



(a)



(b)

Figure 1. (a) A coarse initial mesh consisting of 4-element groups.
(b) The refinement and unrefinement of a group of elements.

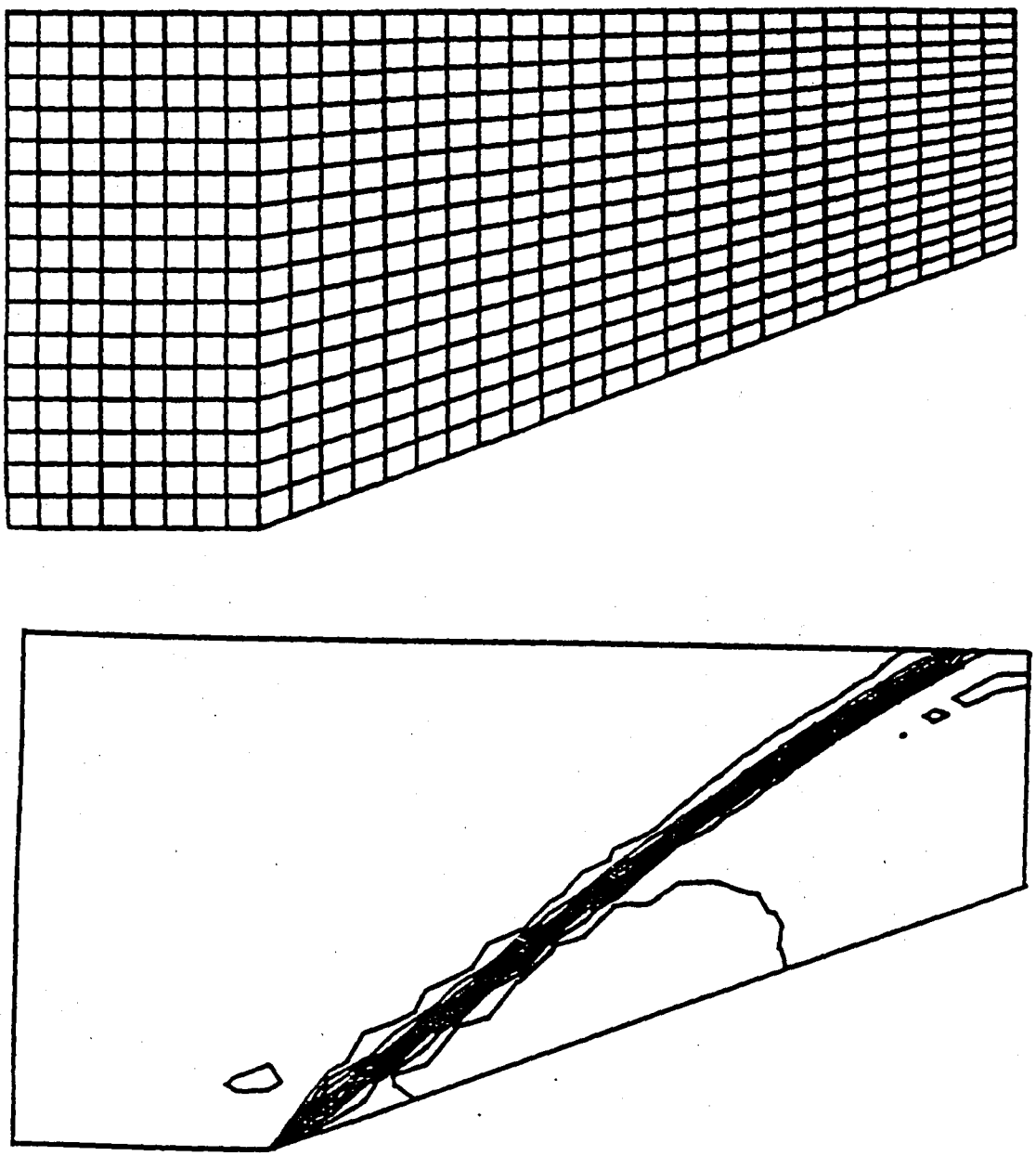


Figure 2. Supersonic flow over a 20° ramp.
Initial mesh and pressure contours.

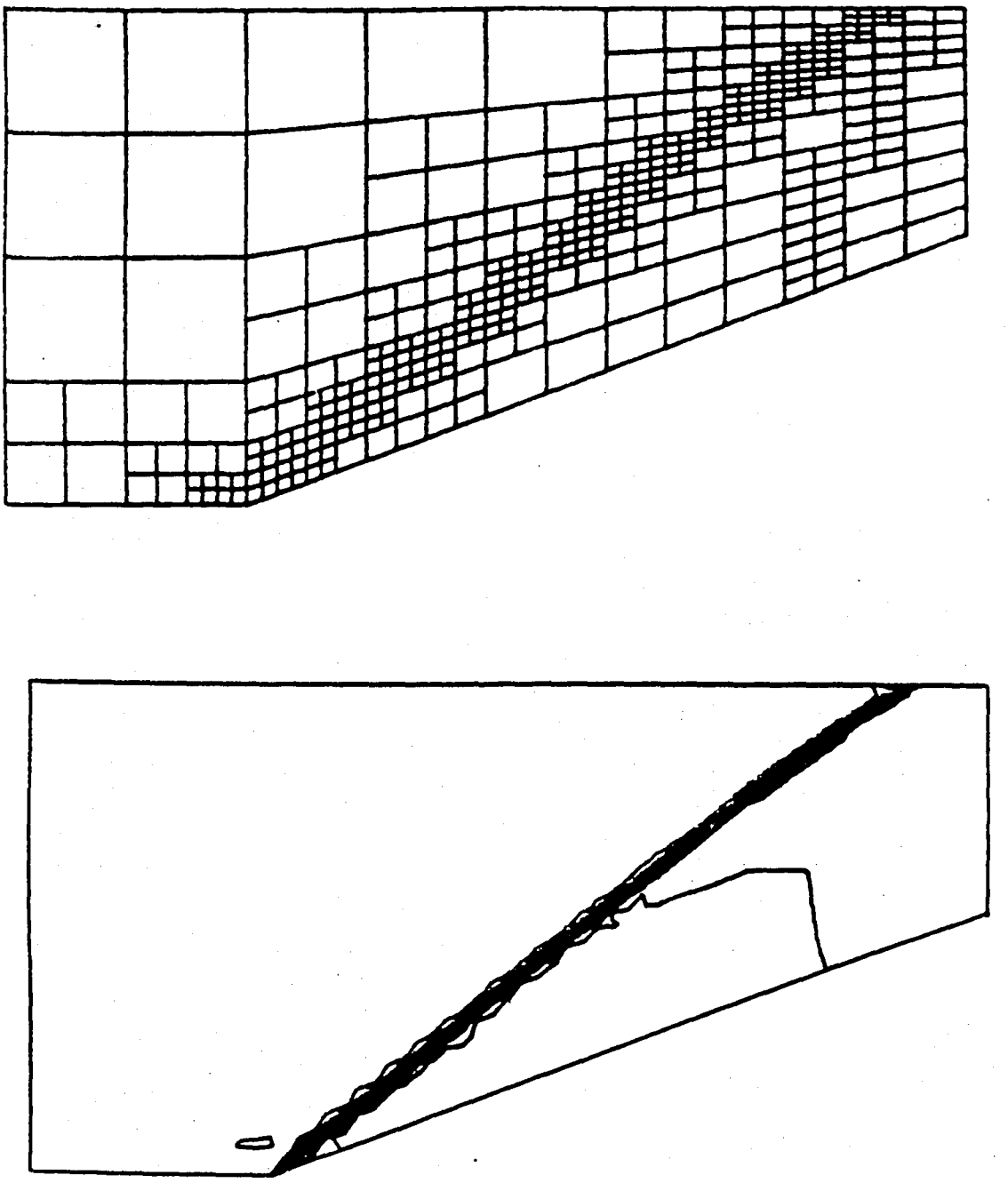


Figure 3. Supersonic flow over a 20° ramp.
Mesh and pressure contours obtained with one level of refinement.

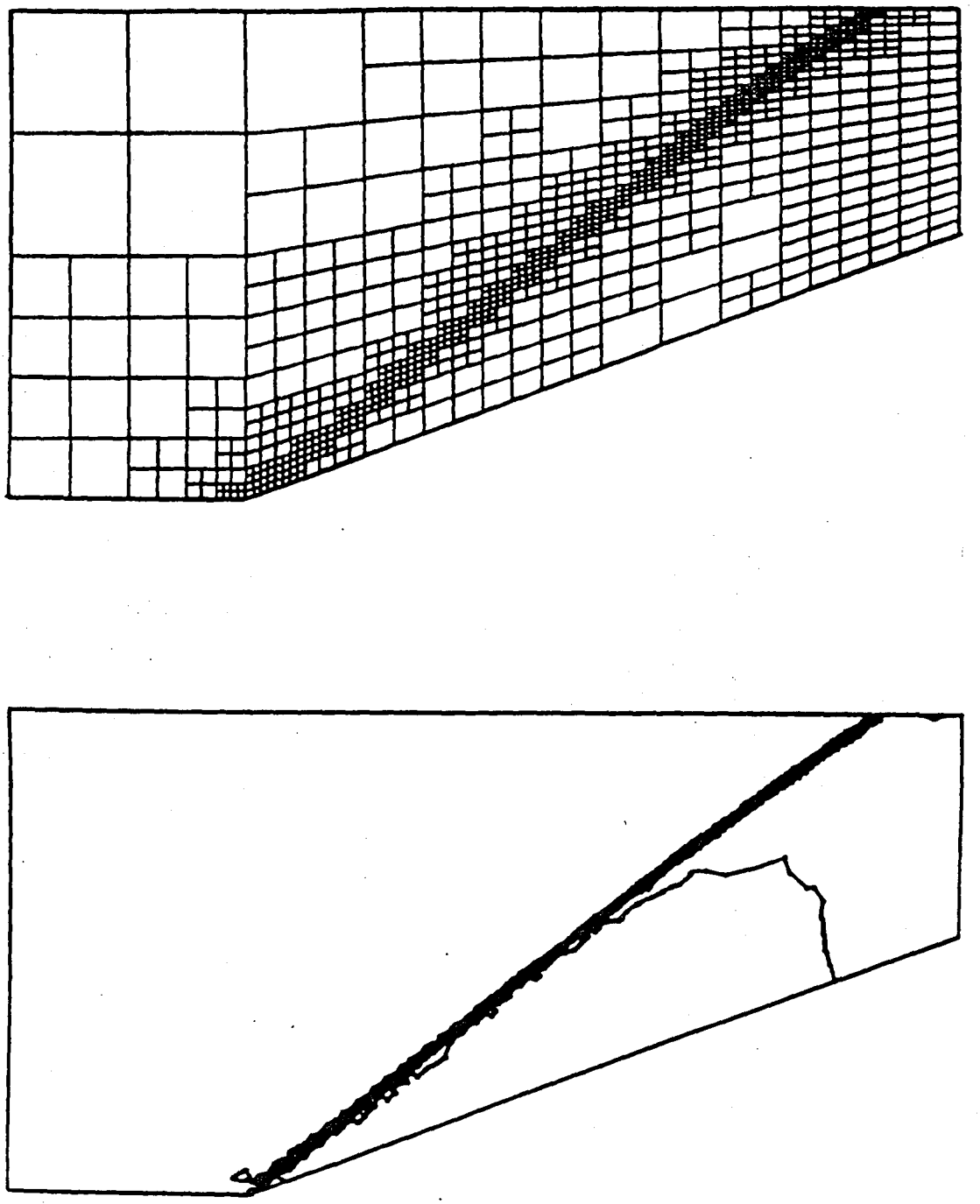


Figure 4. Supersonic flow over a 20° ramp.
Mesh and pressure contours obtained with two levels of refinement.

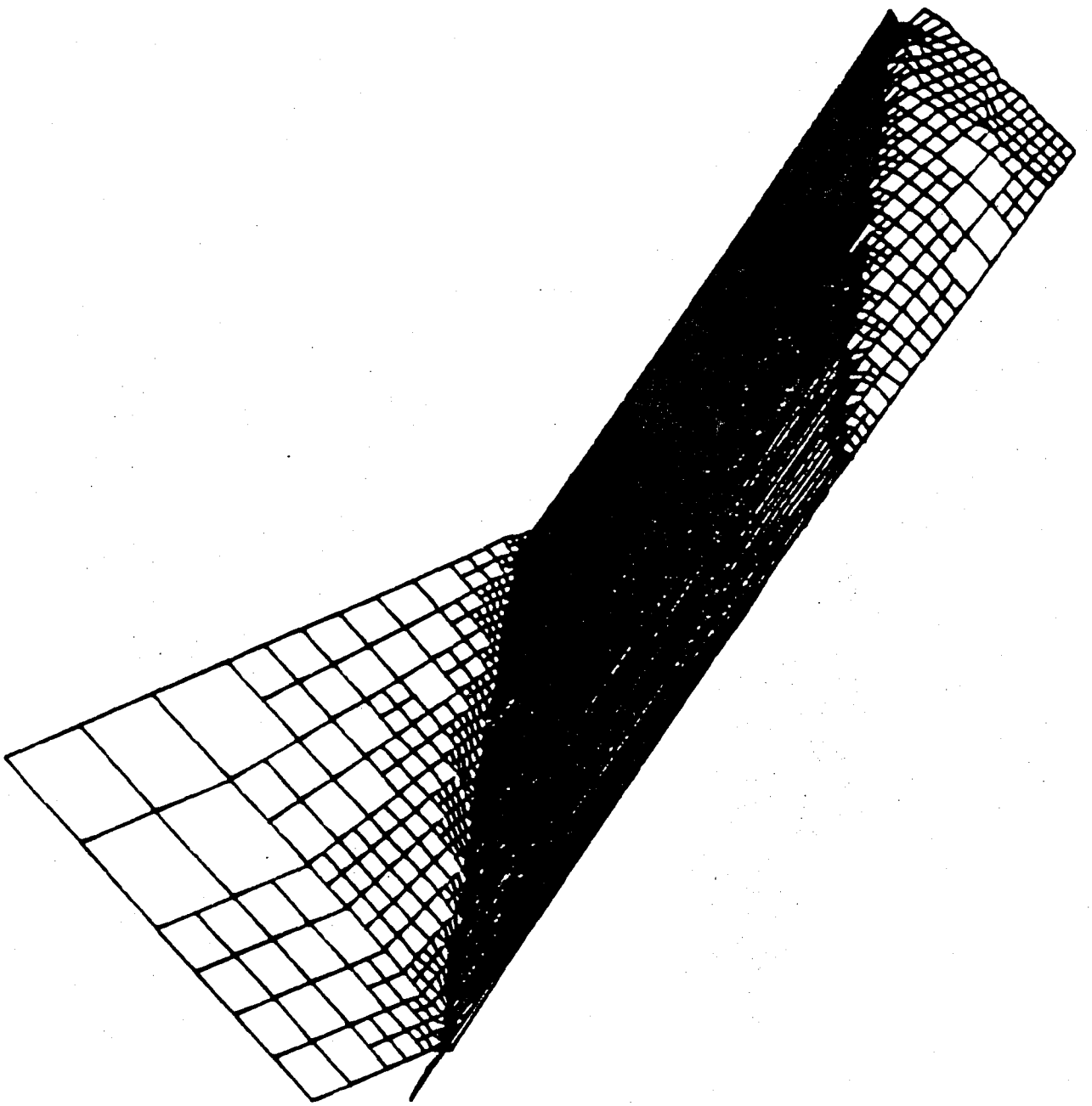


Figure 5. Supersonic flow over a 20^0 ramp.
Three-dimensional view of the converged pressure function obtained with two levels of refinement.

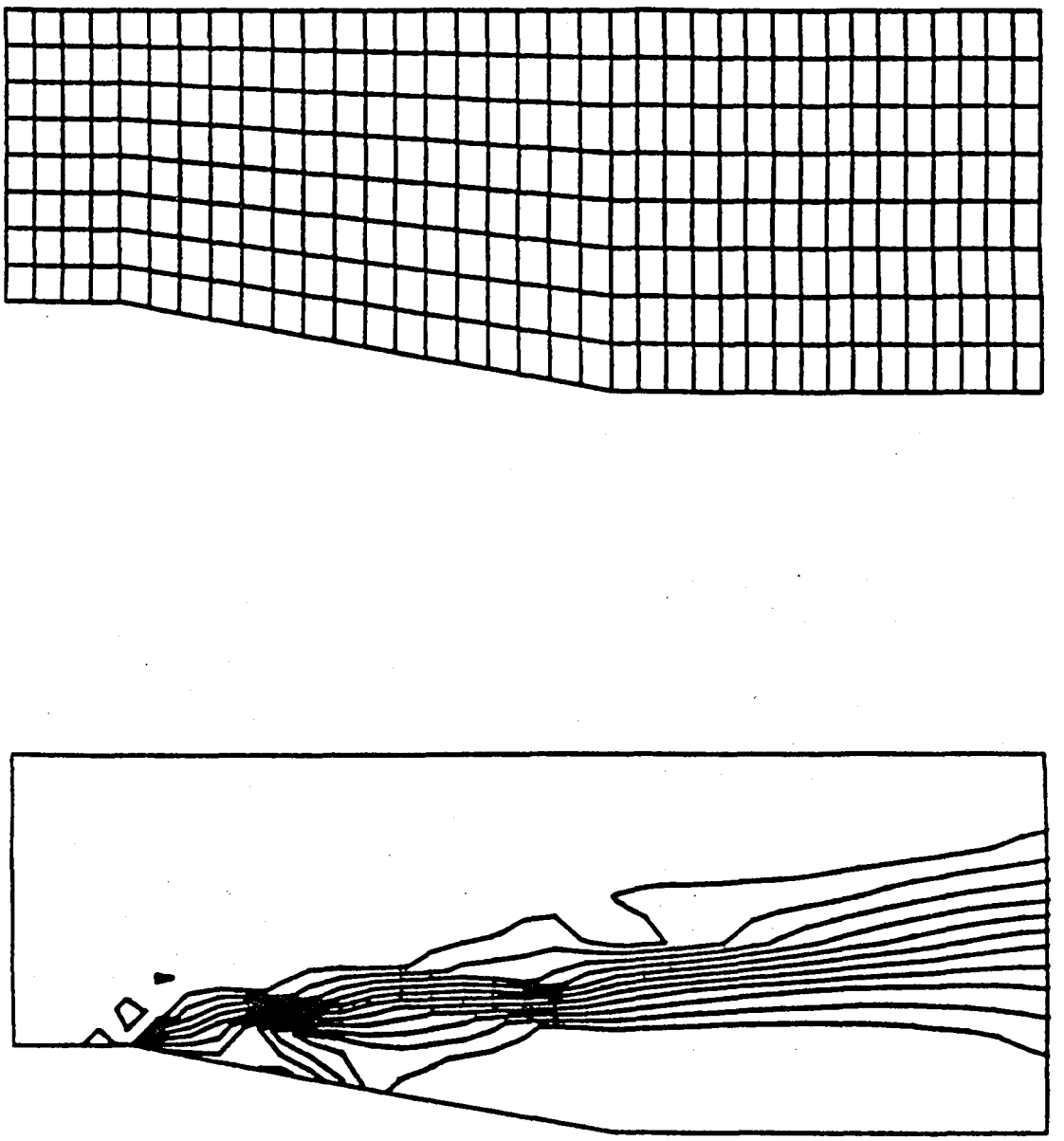


Figure 6. Supersonic expansion around a 10° corner. Initial mesh and density contours.

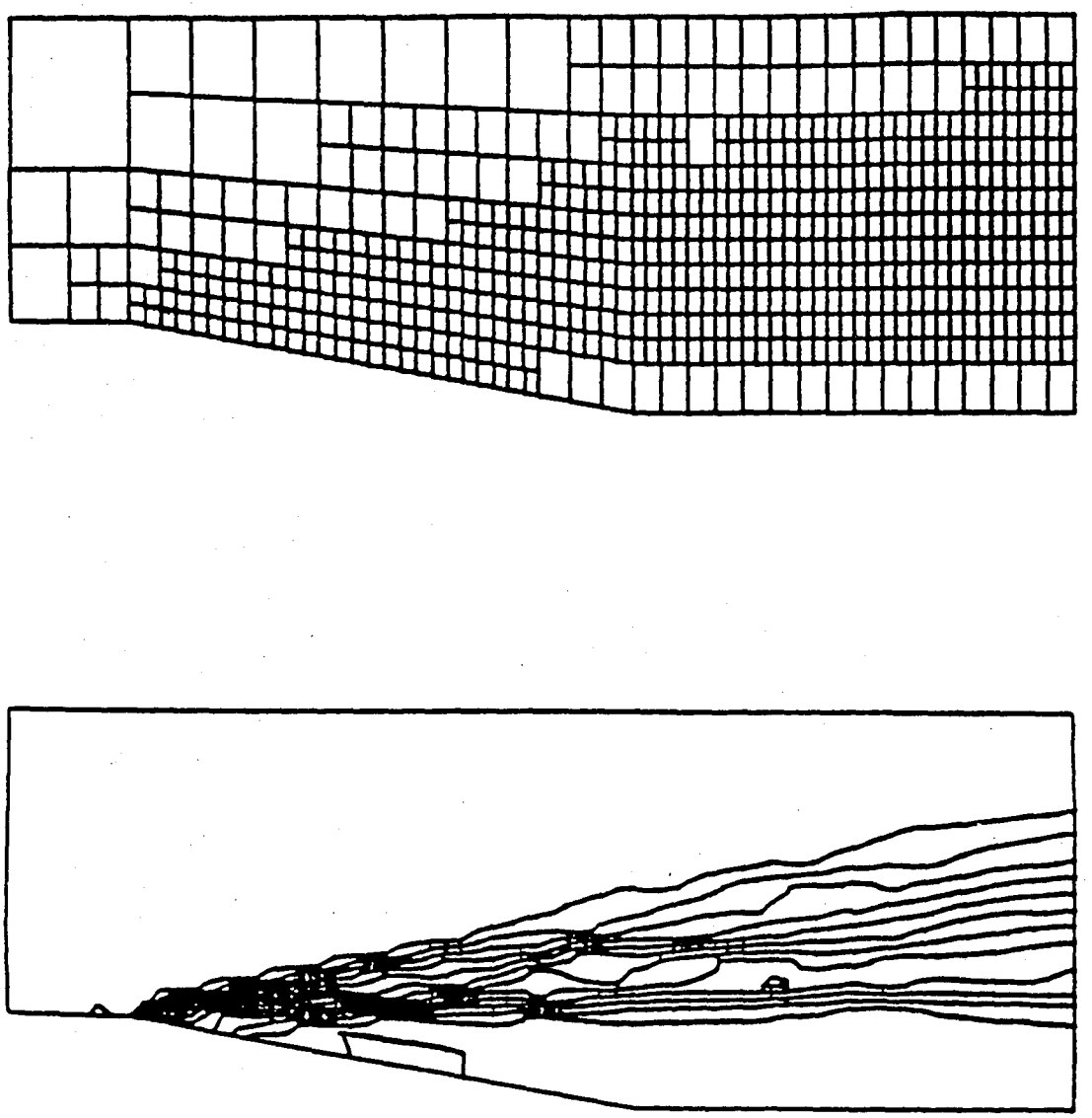


Figure 7. Supersonic expansion around a 10° corner.
Mesh and density contours obtained with one level of refinement.

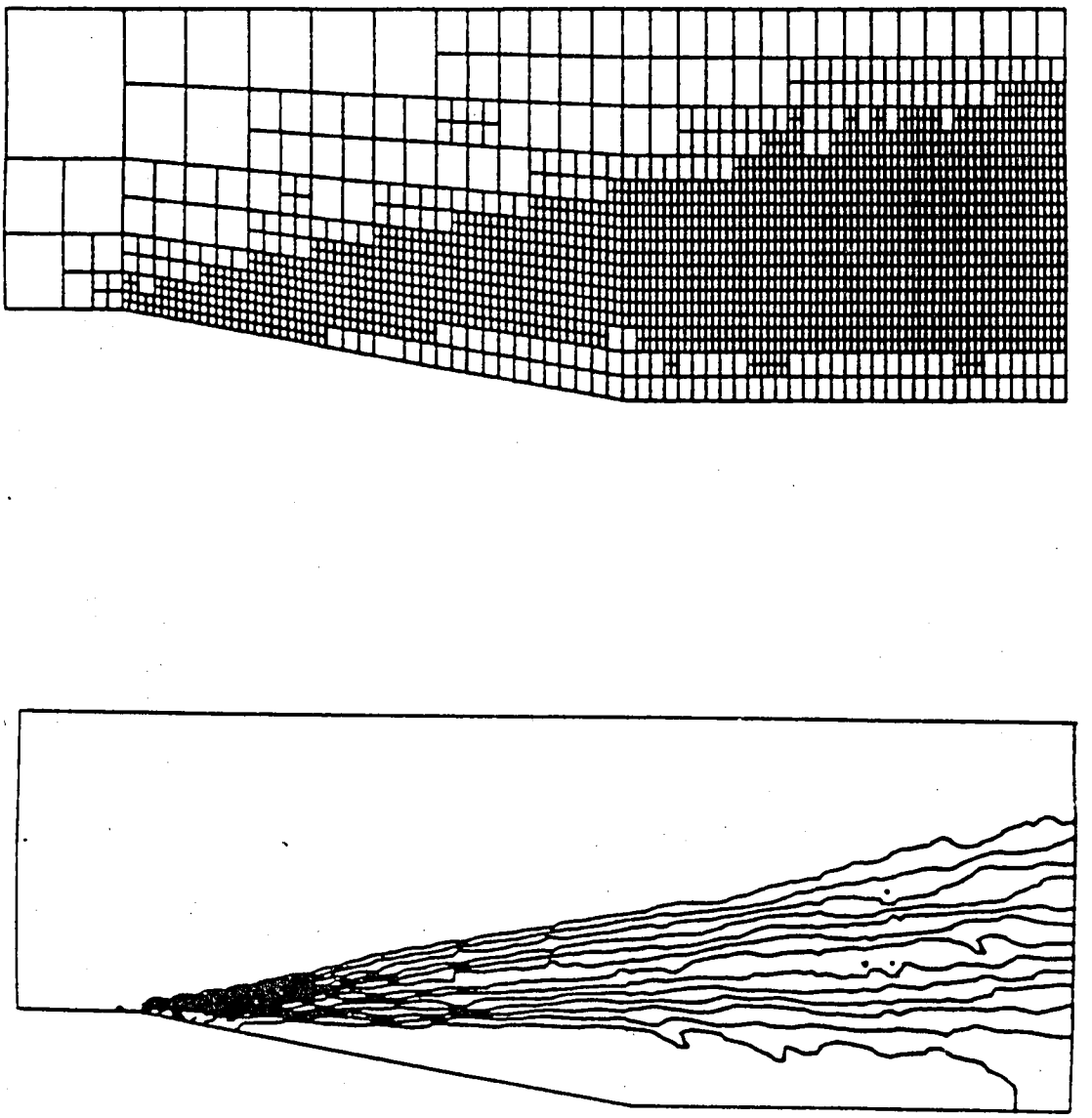


Figure 8. Supersonic expansion around a 10° corner.
Mesh and density contours obtained with two levels of refinement.

employed in the calculation with the corresponding density contours. The results were obtained with the FCT scheme with $\alpha = 0.05$, $\beta = 0.15$ and $c = 0.125$. Striking improvement in the solution is seen to result from the refinement procedure.

4.5.3 Asynchronous Time-Stepping Procedures

In the algorithms described in the previous paragraph the global timestep Δt is determined as the minimum allowable time step in the grid, namely:

$$\Delta t = \min_{e=1,\ldots,M} \Delta t_e ; \quad \Delta t_e = \frac{C \sqrt{A_e}}{|u| + c} \tag{4.14}$$

Here C denotes the C.F.L. number, c is the local speed of sound in the element and $|u|^2 = u^2_1 + u^2_2$.

From the definition (4.14) we see that since $\Delta t \sim C \sqrt{A_e} \sim h_e$ the timestep may be governed by the smallest element in the mesh. This choice of Δt guarantees stability and time-accuracy of the scheme. For steady-state calcumations however time-accuracy is not important and it may be more economical to employ asynchronous time-stepping by prescribing local time-steps.

Let us denote by Δt_j^{node} the nodal timestep of node j which is computed by the minimum of the time-steps of the elements which are connected to node j. Then, an Asychronous TG/FE-LW scheme may be employed as follows:

I. First Step:

For each element Ω_e, compute U_e^{n+1} such that:

$$U_e^{n+1/2} \int_{\Omega_e} d\Omega = \int_{\Omega_e} U_h^n \, d\Omega - \frac{\Delta t_e}{2} \int_{\Omega_e} \text{div } Q(U_h^n) \, d\Omega \tag{4.15}$$

II. Second Step:

Calculate $U_h^{n+1} = \sum_{j=1} U_h^{i,n+1} \phi_i$ such that,

$$\sum_{i=1}^{N} \left(\int_\Omega \phi_j^T \phi_i \, d\Omega \right) U_h^{i,n+1} = \sum_{i=1} \left(\int_\Omega \phi_j^T \phi_i \, d\Omega \right)$$

$$+ \Delta t_j^{node} \int_\Omega Q(U_h^{n+1/2}) : \nabla \phi_j \, d\Omega)$$

$$- \Delta t_e \int_{\partial\Omega} \phi_j^T (Q(U_h^{n+1/2}) - Q(U^n)) \, n \, d\gamma$$

$$+ \Delta t_j^{node} \int_{\partial\Omega} \phi_j^T Q(U_h^n) \, n \, d\gamma \qquad j = 1, 2, \ldots, N \tag{4.16}$$

We now demonstrate some of the features of the asynchronous time-stepping scheme using two numerical examples:

4.5.3.1. The Reflecting Shock Problem:

The statement of the problem is given in [19]. Figures 9 and 10 show the steady-state density contours obtained with the time-accurate and asynchronous algorithms, respectively. We note that the steady state was obtained after 130 time steps with the time-accurate scheme and after only 100 time-steps with the asynchronous scheme, which represents 30% of savings in computational effort.

4.5.3.2. NACA 0012 Airfoil in Supersonic Wind Tunnel:

We also considered the problem of a NACA 0012 airfoil in a supersonic wind tunnel with inflow Mach number $M_\infty = 3$, $\gamma = 1.4$ [19]. Figure 11 presents a comparison between the steady-state density contours obtained with the two schemes. The time-accurate scheme requires 585 time-steps to converge while the asynchronous scheme converged after 496 time-steps.

4.5.4. Elevon Cove Problem

The Elevon Cove problem has to do with supersonic flow past a complex swan-like geometry of a portion of the space shuttle elevon. The problem is described in [3]. Figures 12 and 13 show a preliminary calculation of the problem with our adaptive Euler code. The mesh shown does not correspond dto a later unrefined mesh. This mesh is not yet optimal, since the program was still attempting to compute a new mesh at the time calculations were stopped.

5. Features for an Adaptive Finite Element Algorithm for Transient Calculations

We now present an example of an h-refinement / unrefinement strategy for transient calculations. The basic steps of the algorithm are:

a) Advance the solution N time steps.

b) Do the following until no more elements can be refined:

(1) Compute the element error indicators θ_e.
(2) Refine all elements with $\theta_e \geq \beta\, \theta_{MAX}$

(3) Integrate the last N time steps with the updated (refined) mesh
(4) Go to (1).

c) Compute the element error indicators θ_e and unrefine all groups with $\theta^m_{GROUP} \leq \alpha\, \theta_{MAX}$

d) Go to a).

We note that the "do loop" in step b) converges when no more elements can be refined (the maximum level of refinement is fixed). Although the iteration in step b) guarantees a **"fully updated" mesh** it may lead to an expensive scheme if more than a few passes are required for convergence of the "do loop". A cheaper alternative is presented by the following "two-pass" scheme:

a) Advance the solution N time steps.

b) Compute the element error indicators θ_e.
c) Refine all elements with $\theta_e \geq \beta\, \theta_{MAX}$
d) Integrate the last N time steps with the refined mesh obtained in c)

e) Compute the element error indicators θ_e and
1) Unrefine all groups with
$\theta^m{}_{GROUP} \leq \alpha\, \theta_{MAX}$

2) Refine all elements with
$\theta_e \leq \beta\, \theta_{MAX}$

f) Go to step a).

In the following, we present two examples of adaptive refinement for transient problems.

5.1 Rotating Cone Problem [11]

We consider the following advection problem:

$$\frac{\partial U}{\partial t} + \text{div}\ (\mathbf{a}\, U) = 0$$

$$U(x,y,0) = \begin{cases} 0, & r \geq 150 \\ 250\,[1 + \cos\frac{\pi r}{150}], & r < 150 \end{cases}$$

Here, $r^2 = x^2 + (y - 250)^2$ is given by the vector $\mathbf{a}\,(R, \theta) = (R \cos\theta, -R \sin\theta)$ where R, θ are the polar coordinates indicated in Figure 14.

This problem has been solved by many authors and it is considered as a benchmark problem for algorithms for advection problems ([15], [11]). Here we show some results obtained with an adaptive SUPG algorithm [11]. Figure 15 shows some "fully updated" meshes which are obtained with the scheme outlined in the beginning of this section. For more details the reader should refer to [11].

5.2 A Problem of Supersonic Rotor-Stator Interaction

We applied the "two-pass" adaptive algorithm to a problem of supersonic rotor-stator interaction. We consider now two rows of doubly-parabolic airfoils with thickness to lenth ratio equal to 0.08. Figure 16 shows some of these airfoils and the initial finite element discretization of the domain. We assume that the stator and the rotor have the same number of airfoils and we perform the computation on domains corresponding in one rotor and one stator airfoil while the presence of the remaining airfoils is simulated by periodic boundary conditions. In the figures the domain of the rotor airfoil is drawn twice.

In the Figs. 17 through 25 we give the results of a supersonic calculation obtained with a dynamically adapted grid. The distance between consecutive airfoils of the rotor (and stator) is assumed equal to twice the airfoil length while the distance between the tail of the stator and the front tip of the rotor airfoil is taken equal to 0.2 of the airfoil length. We impose boundary conditions of supersonic inflow on the left boundary of the stator with the dependent variables equal to

$\rho = 1.4,\ \rho u = 4.2,\ \rho v = 0,\ \rho e = 8.8.$

The inflow boundary conditions correspond to a free stream Mach number equal to three. Boundary conditions of supersonic outflow were assumed on the right boundary of the domain of the rotor. The steady-state solution which is obtained by keeping the airfoild fixed was used as initial condition.

We have chosen $\beta = 0.19$, $\alpha = 0.06$ and we defined the group error indicator to be equal with the maximum element error indicator of the elements in the group. We did not specify N but instead we revised the mesh every time the fifth nodes [23,24] of the rotor mesh coincided with corner nodes of the stator mesh (this resulted in mesh revisions every 10-12 time steps). We also note that all interface elements have been refined beforehand with the maximum level of refinement to facilitate the application of the sliding interface algorithm.

In order to capture shocks of variable strength we used the "normalized" error indicators given in (4.13). It becomes clear from the numerical results that variable shocks are captured well and the mesh evolves dynamically to adapt to the solution of the rotor-stator problem.

Results are shown in Figures 17 - 25. The initial mesh is that shown in Fig. 16. The first adaptive calculation for a steady-state initial condition is shown in Fig. 17a. The corresponding computed pressure profiles are shown in Fig. 17b. Note the symmetry of the shock lines, the continuous pressure fields across the mesh interface, and the fact that both unrefinement and refinement of the mesh were required to achieve the accuracy limits specified. The rotor blades are then allowed to move with unit speed and the mesh is dynamically refined. Plots are shown of calculated adaptive meshes and pressure profiles for 1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8 and 1 cycle (period) of the motion, during which a rotor blade makes a complete revolution from its initial position in Fig. 17 back to the same position.

Several features of the computed meshes and solutions are noteworthy. In the initial steady-state case, only 16 unrefined elements appear. The size of these large elements, indicating small local error, is limited in the present caluclations by the distance from the tips of the rotor and stator blades: two elements in the present case since there must exist a sliding interface between them. A minor program modification could allow much larger elements in regions of small error. For the transient case, the number of larger elements (indicating substantial unrefinement) increases, and these regions of low error migrate over the mesh as solution evolves in time. Conversely, substantial refinement of the mesh is indicated at the interface and along shock lines. The method successfully captures shock interactions and the increasing density of pressure profiles downstream from the moving blades. The ratio of the number of elements in the adaptive mesh to that in the uniform fine mesh varies in time, but is typically 4,000 / 12,500, a reduction of 68 percent! The initial coarse mesh of Fig.16 contains around 4000 cells and is incapable of delivering the required accuracy, a fact not easily realized without an expensive computation.

6. Future Directions

We believe adaptive finite element methods will have a significant impact on computational fluid dynamics and computational structural mechanics in the future. These techniques, together with the modern parallel and array processors, will make obsolete many of the more popular methods in numerical analysis in use today. In particular, use of the body-fitted coordinate techniques, splitting methods such as ADI, etc. will probably lose some of their popularity since they are not well suited for problems with unstructured meshes.

It is likely that large gains are to be made in three-dimensional problems. Here more than anywhere else, one needs to do computations on a near optimal mesh where only a minimum number of degrees-of-freedom is required to produce a given level of accuracy.

It is likely that new advances in parallel and array processing will bring the p-methods and h-p methods to the forefront, since, at least from a theoretical point of view, array processors may have

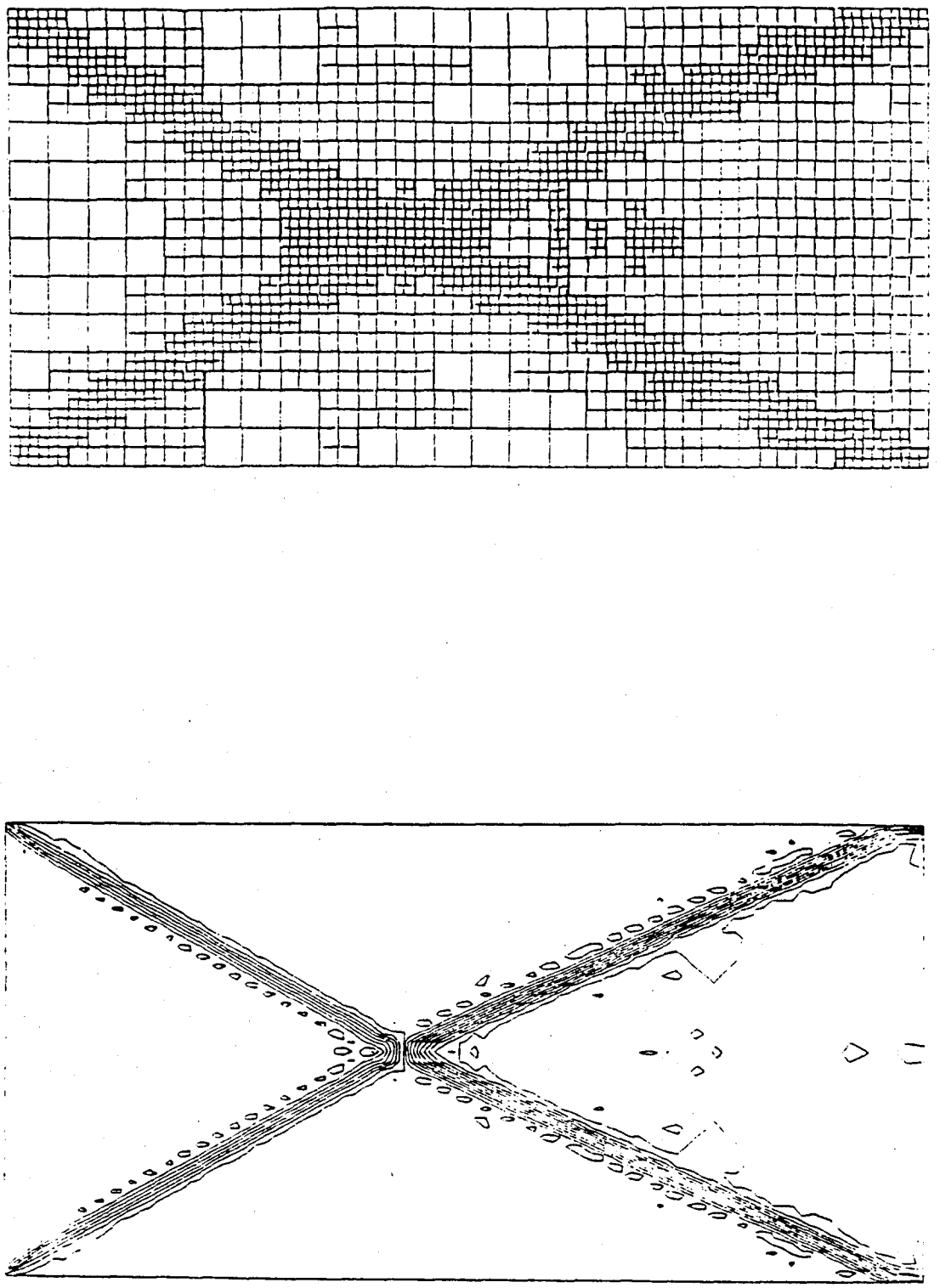


Figure 9. Reflecting shock problem.
Adaptive grid and density contours obtained with the time-accurate scheme.

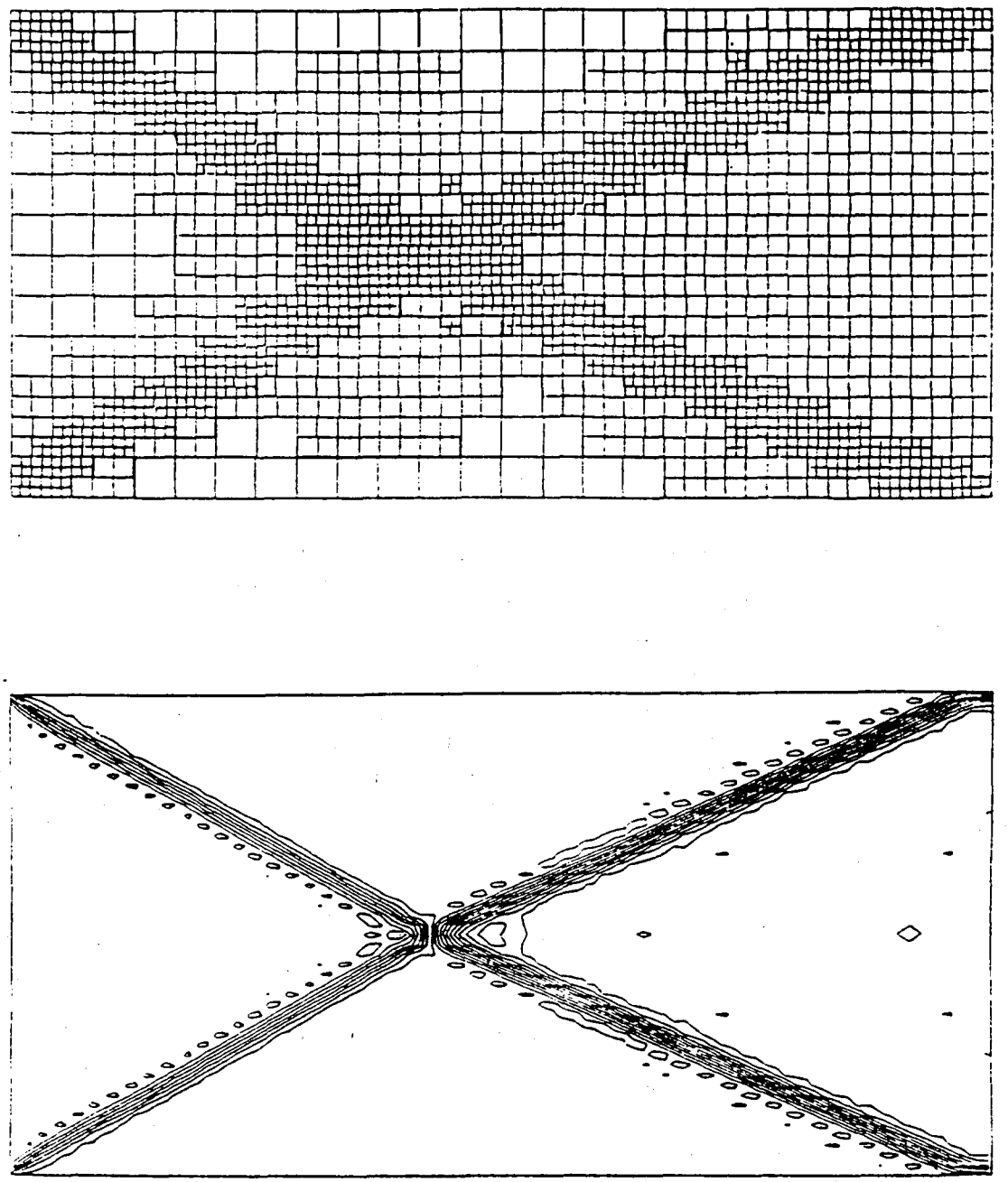


Figure 10. Reflecting shock problem.
Adaptive grid and density contours obtained with the asynchronous time-stepping scheme.

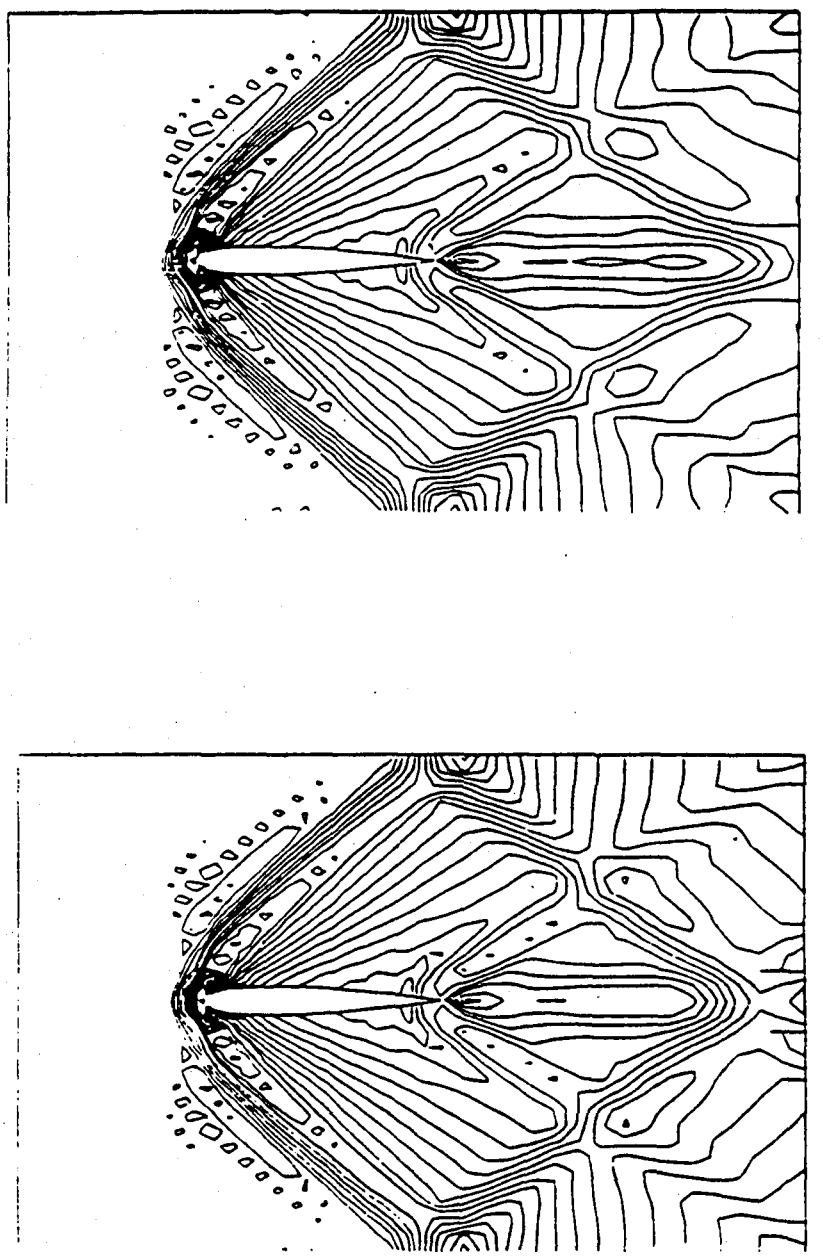


Figure 11. NACA 0012 airfoil in supersonic wind tunnel.
(a) Steady-state density contours obtained with the time-accurate scheme.
(b) Steady-state density contours obtained with the asynchronous time-stepping scheme.

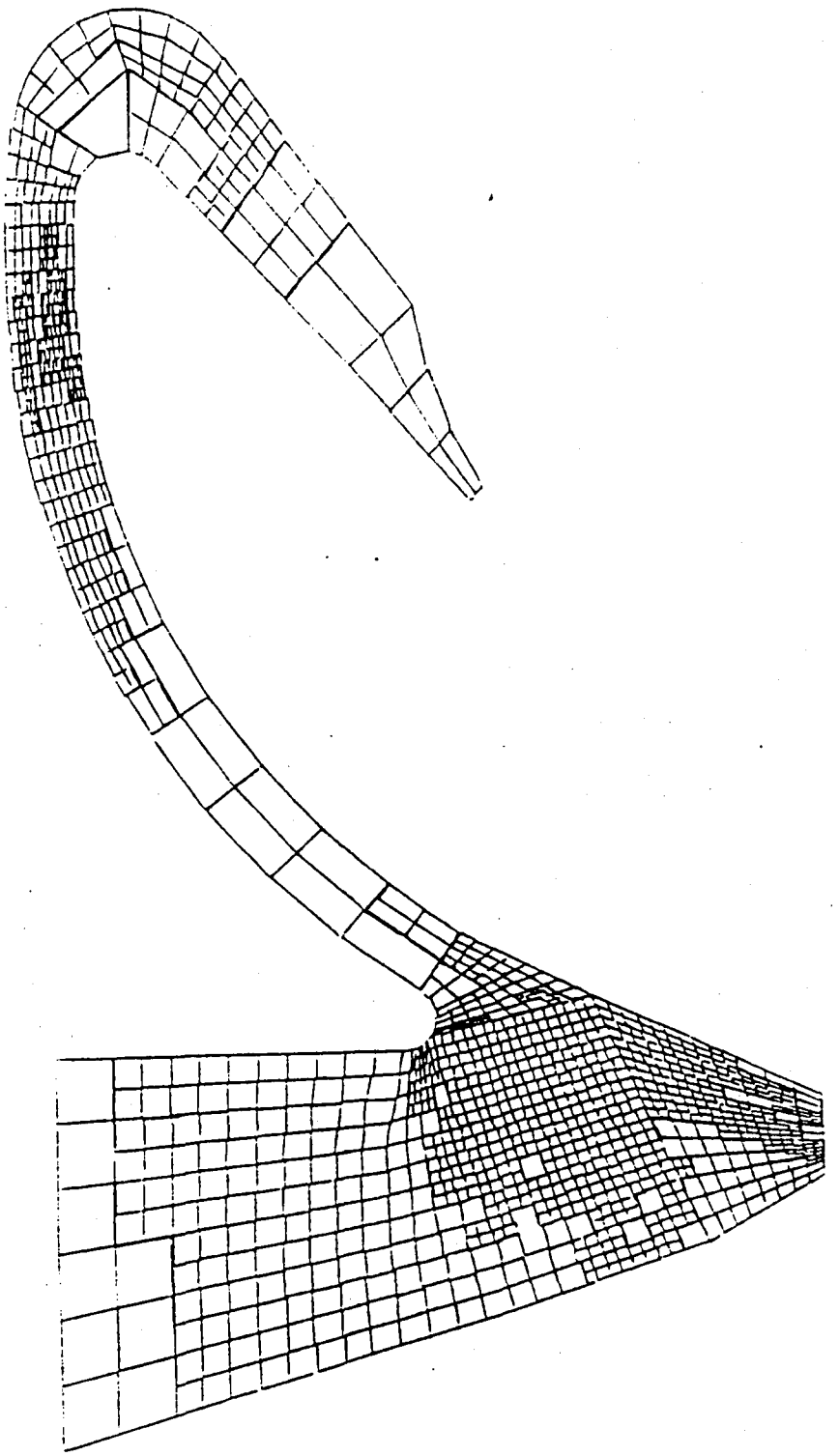


Figure 12. Elevon Cove Problem.
Adaptive Finite Element Grid.

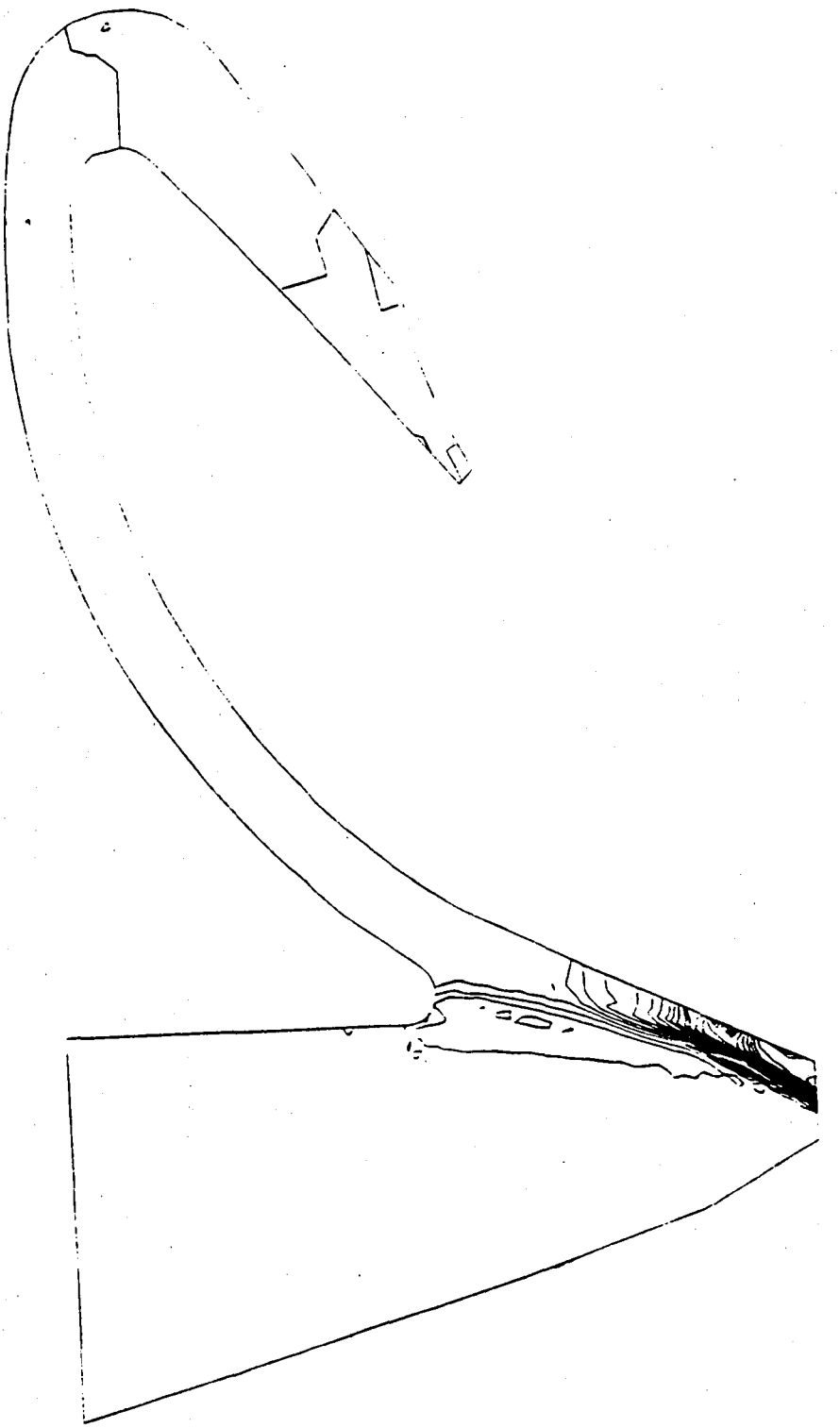


Figure 13. Elevon Cove Problem.
Density contours of the computed solution.

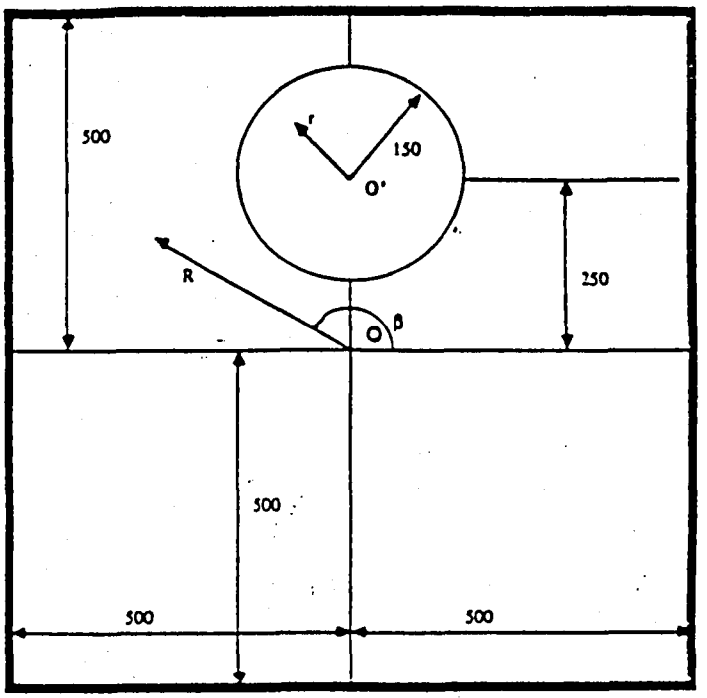


Figure 14. Rotating cone problem.
Problem statement.

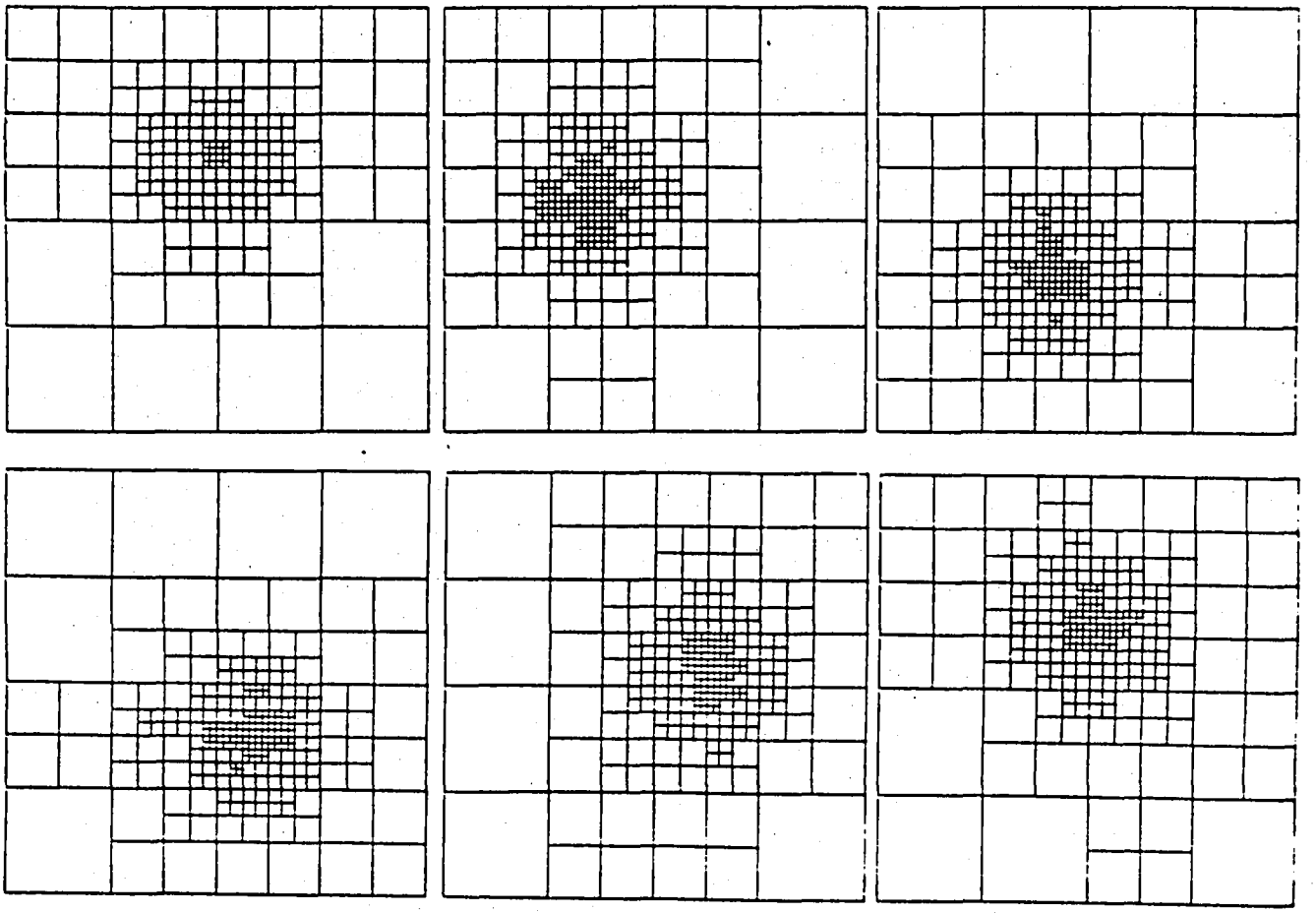


Figure 15. Rotating cone problem.
"Fully updated" adaptive grids.

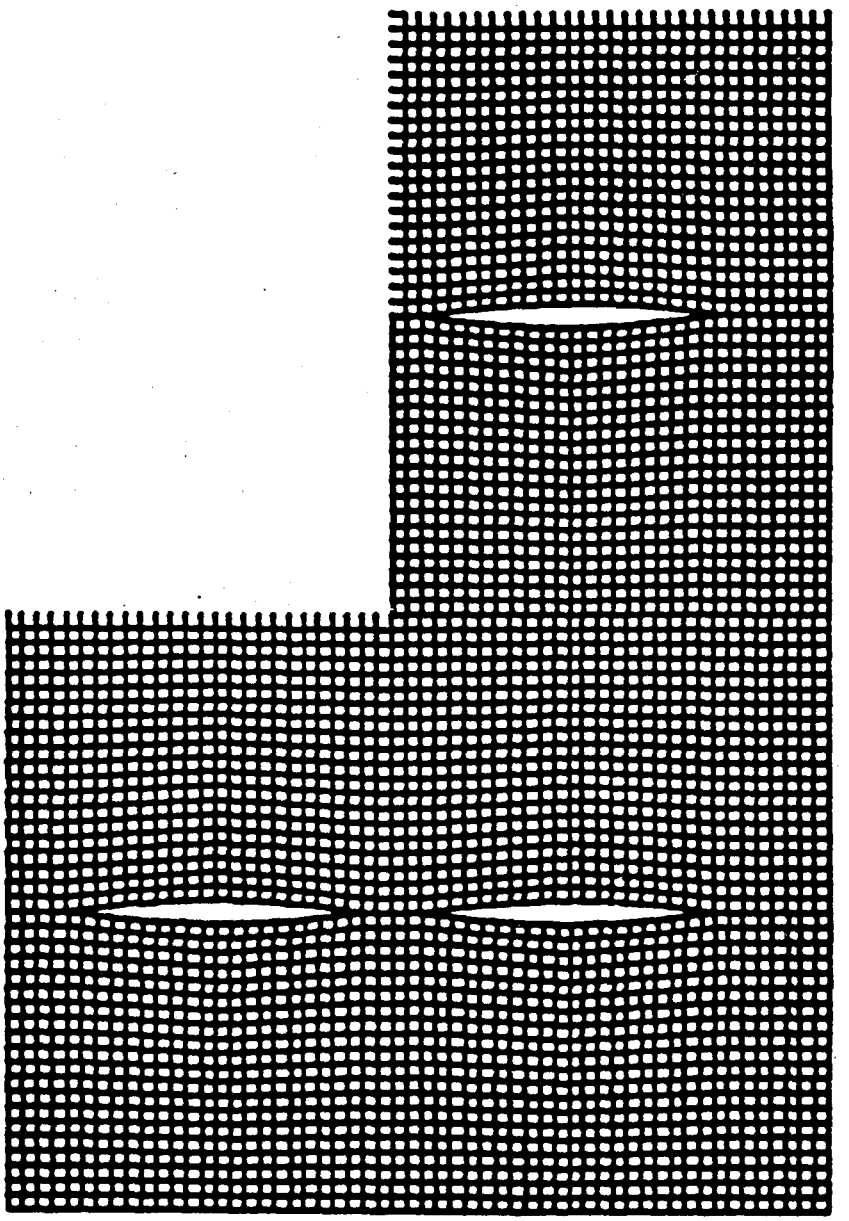


Figure 16. Supersonic flow interaction between rotor and stator airfoils. Initial finite element mesh employed in the calculation.

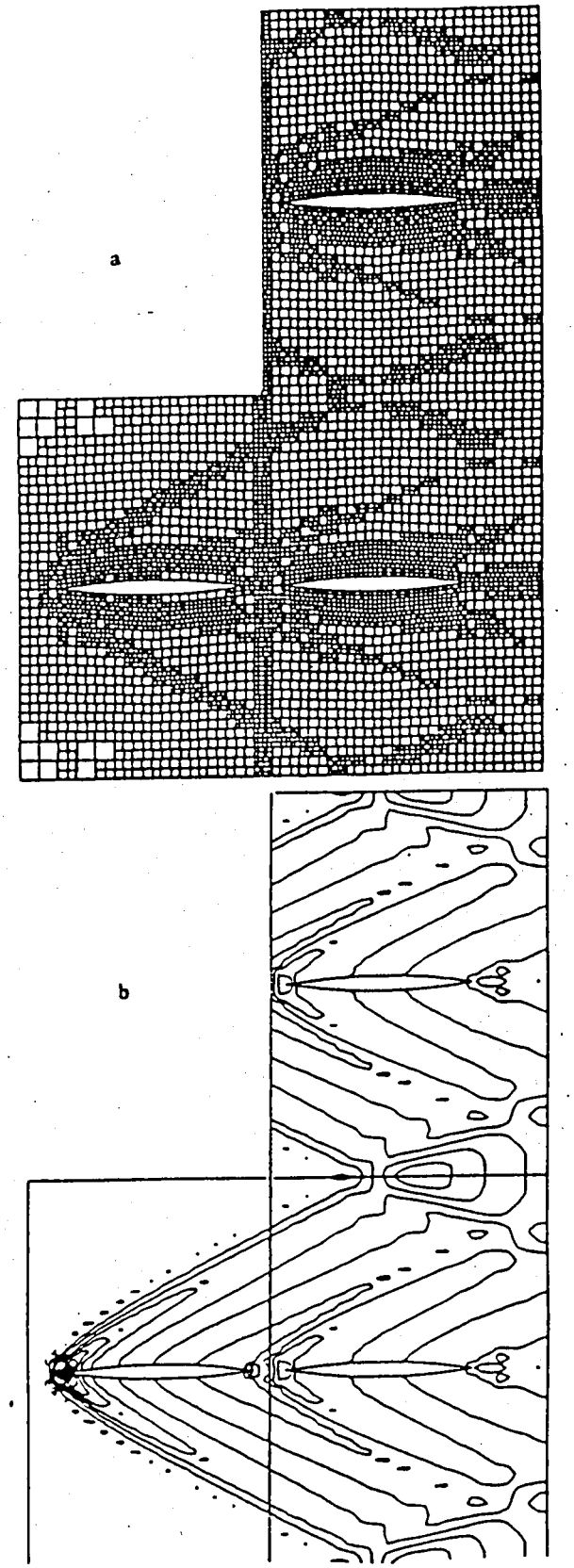


Figure 17. Supersonic flow interaction between rotor and stator airfoils:
(a) Initial adaptively refined mesh for steady-flow through rotor-stator configuration.
(b) Pressure contours for steady-flow through rotor-stator configuration.

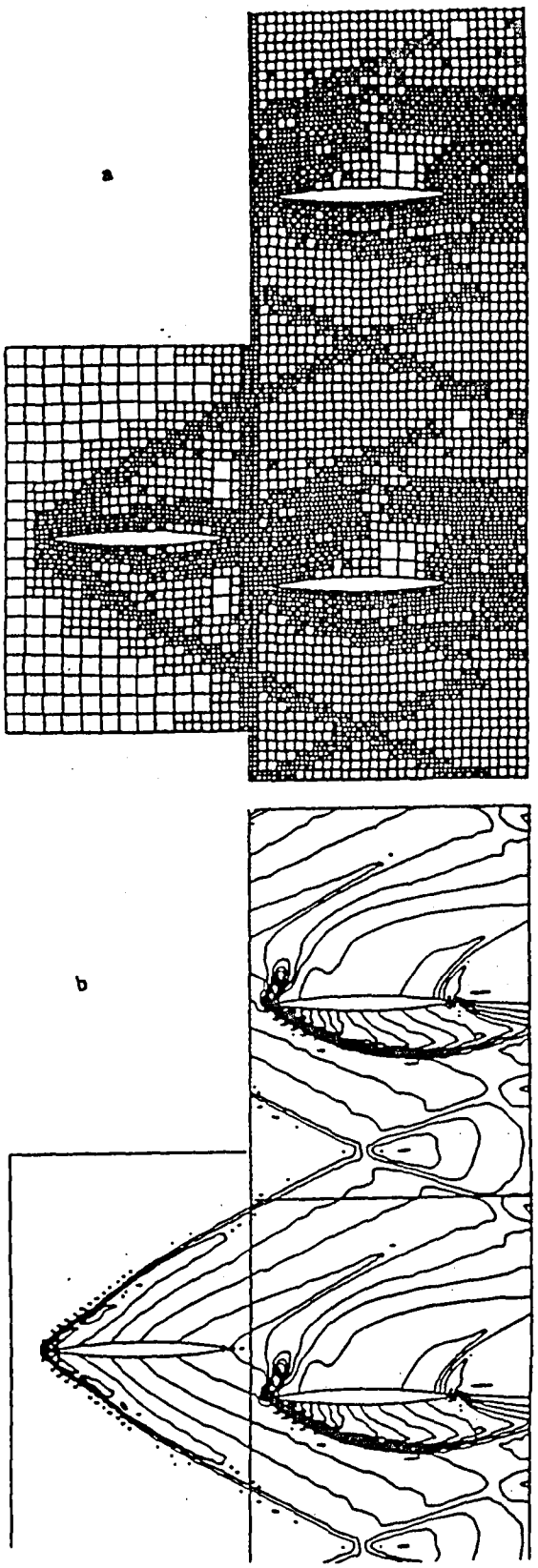


Figure 18. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 1/8 of the rotor cycle.
(b) Pressure contours at 1/8 of the rotor cycle.

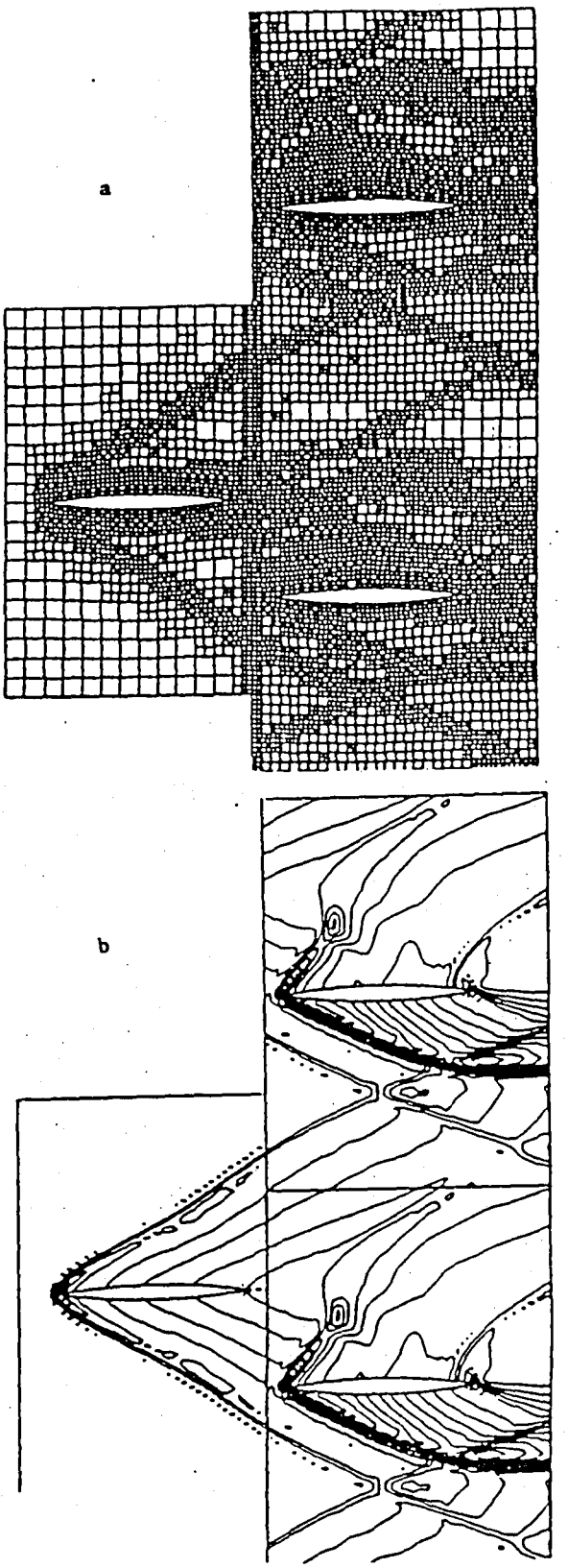


Figure 19. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 2/8 of the rotor cycle.
(b) Pressure contours at 2/8 of the rotor cycle.

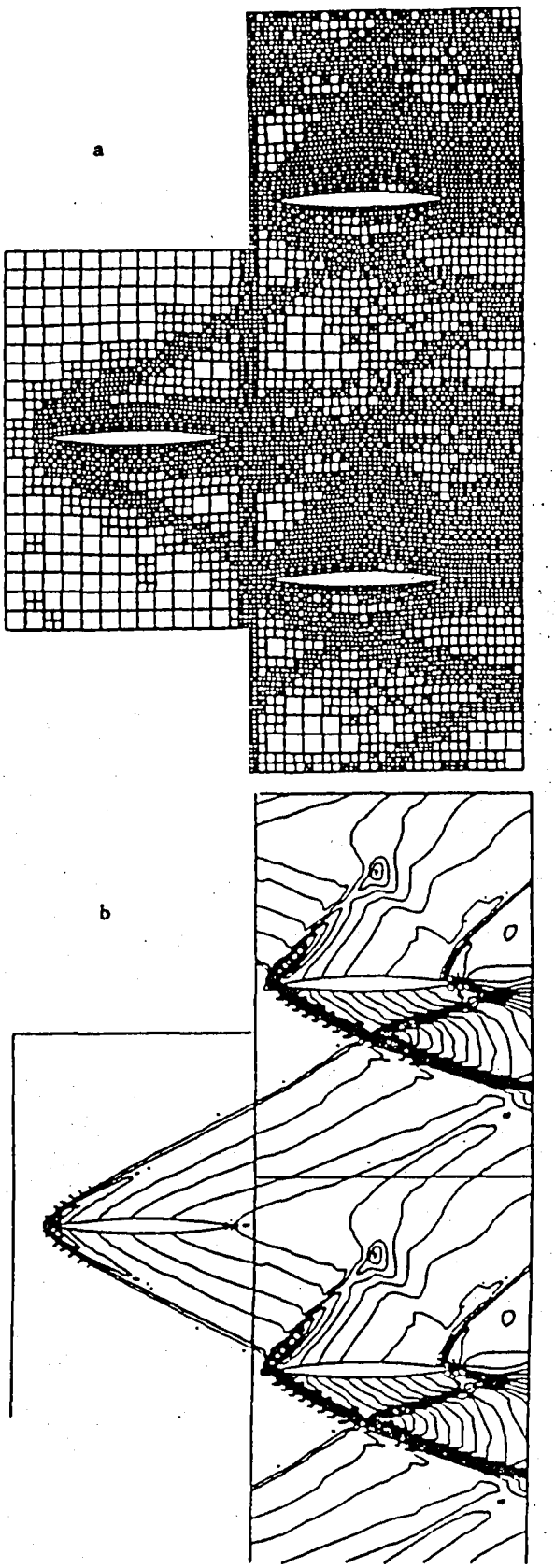


Figure 20. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 3/8 of the rotor cycle.
(b) Pressure contours at 3/8 of the rotor cycle.

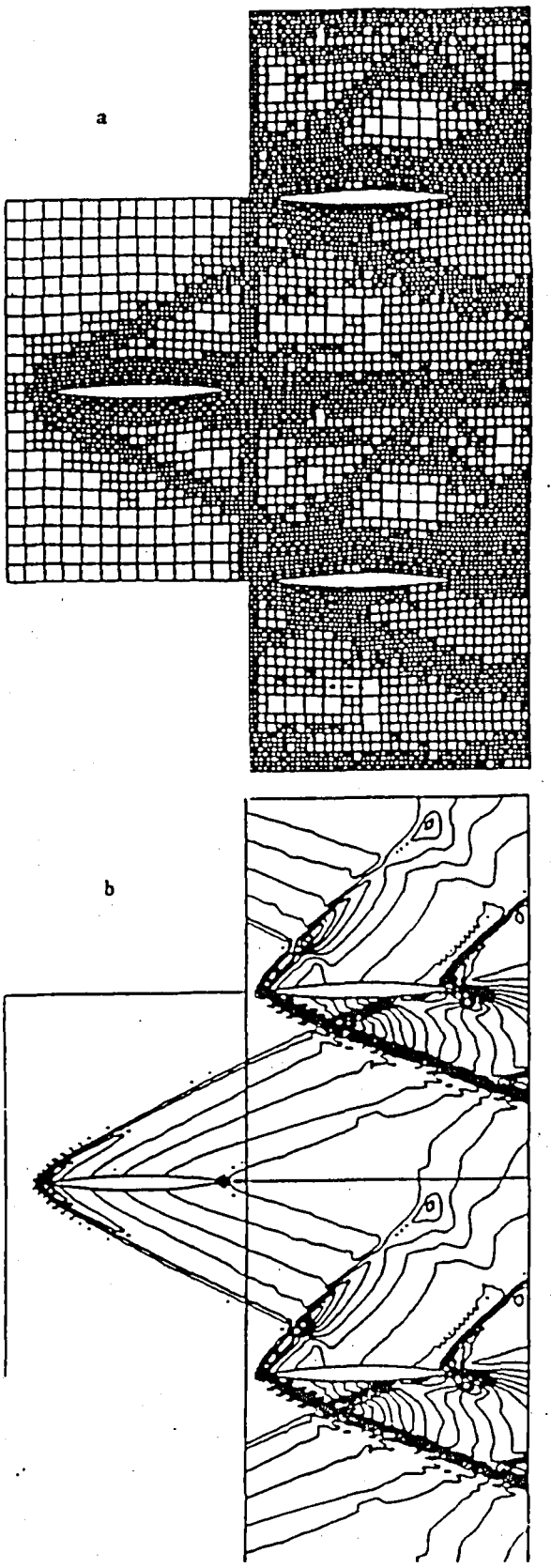


Figure 21. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 4/8 of the rotor cycle.
(b) Pressure contours at 4/8 of the rotor cycle.

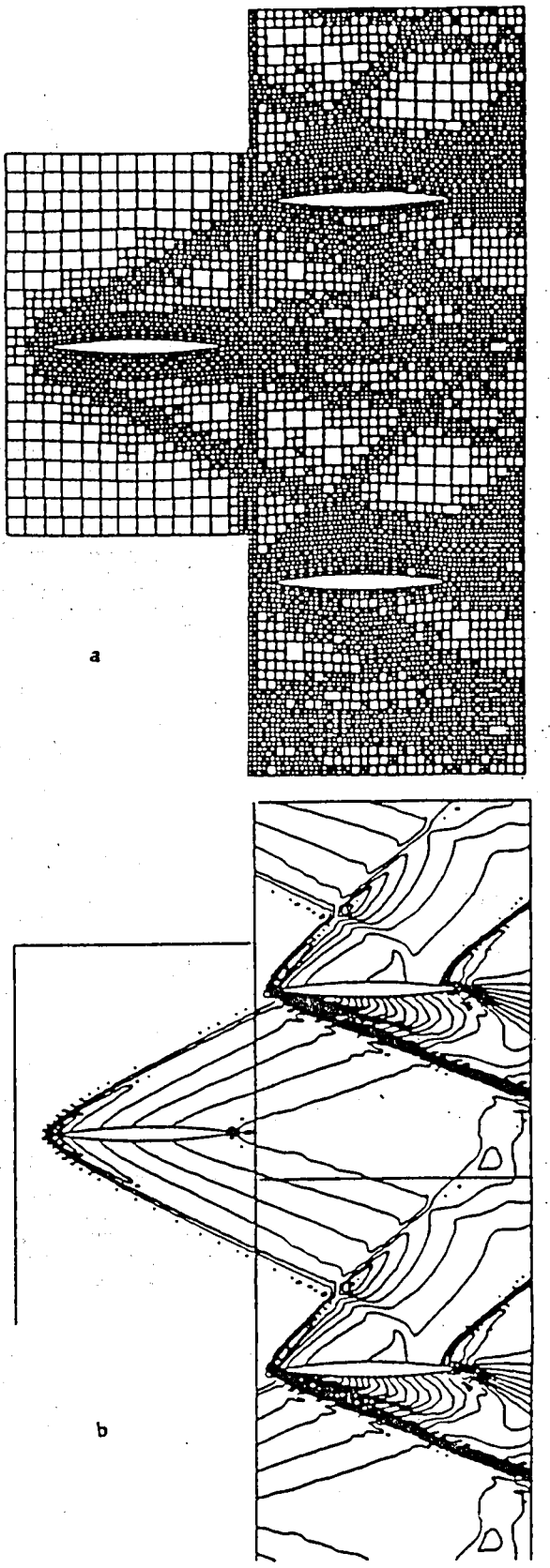


Figure 22. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 5/8 of the rotor cycle.
(b) Pressure contours at 5/8 of the rotor cycle.

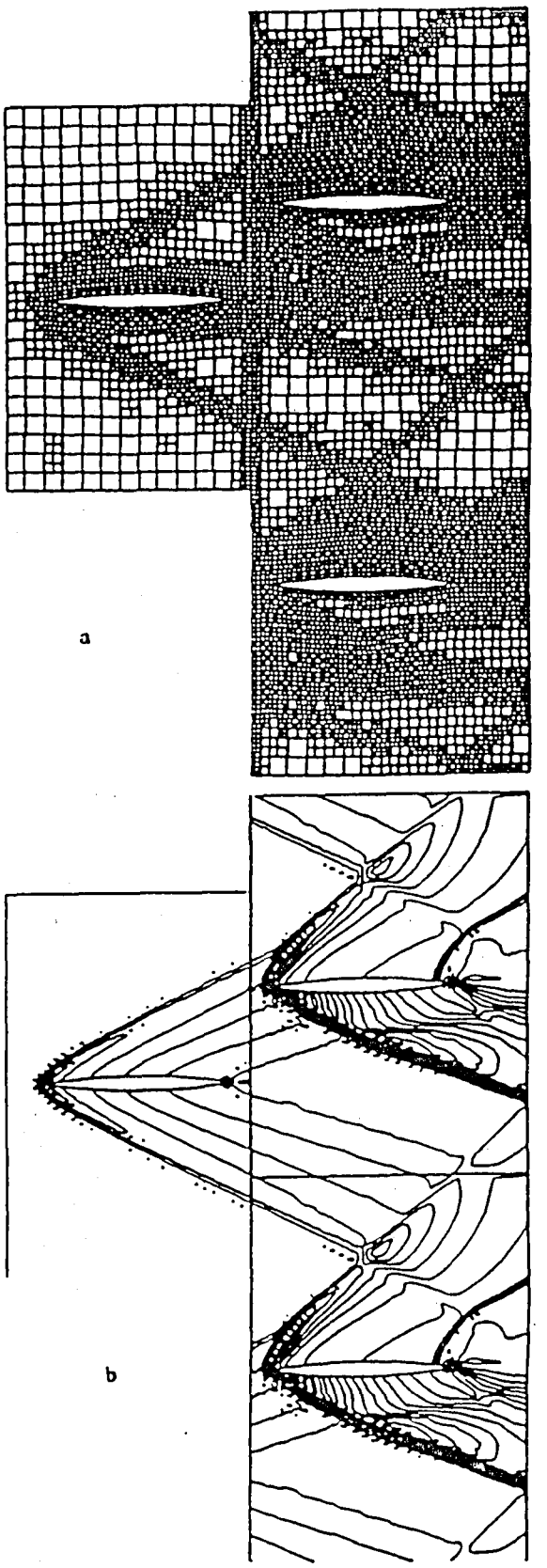


Figure 23. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 6/8 of the rotor cycle.
(b) Pressure contours at 6/8 of the rotor cycle.

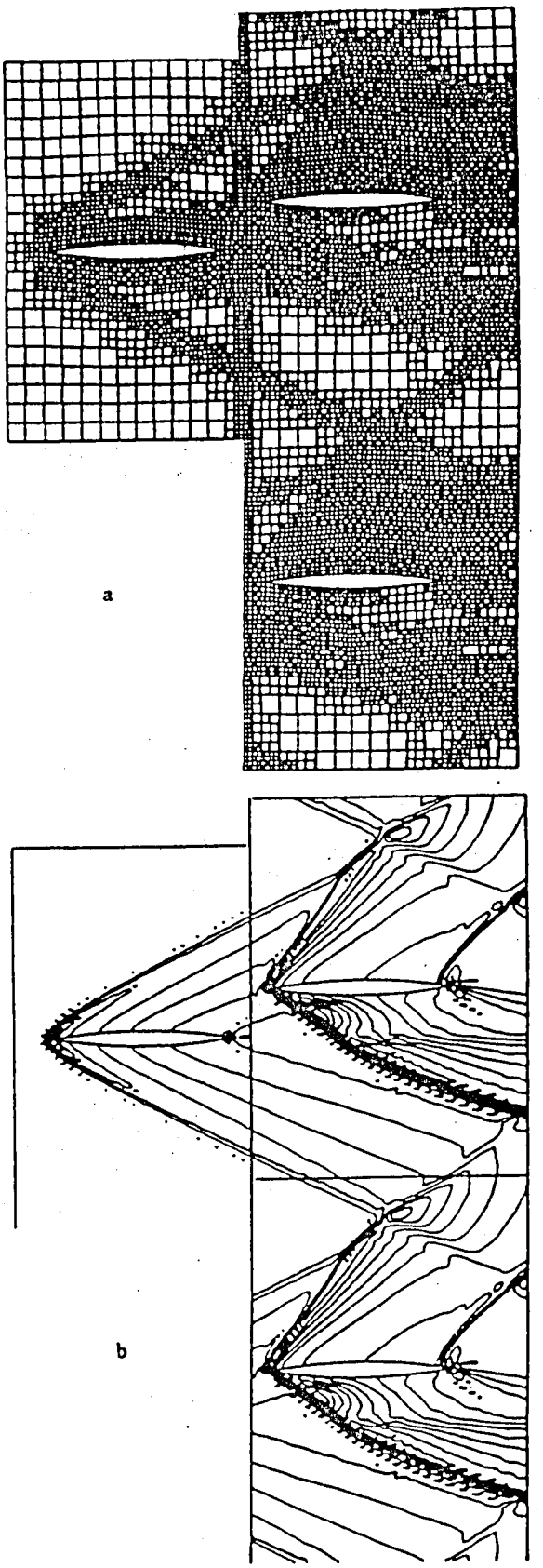


Figure 24. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh at 7/8 of the rotor cycle.
(b) Pressure contours at 7/8 of the rotor cycle.

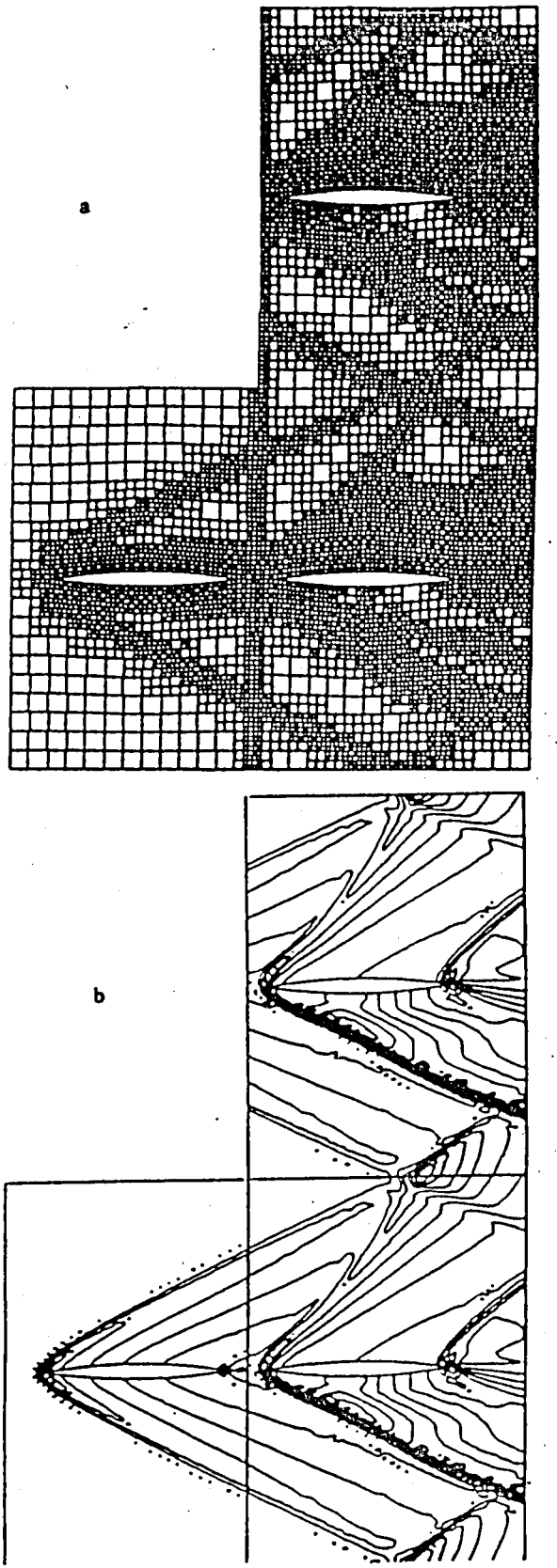


Figure 25. Supersonic flow interaction between rotor and stator airfoils.
(a) Adaptively refined mesh after one complete rotor cycle.
(b) Pressure contours after one complete rotor cycle.

the capability of handling significant local data management problems that might be associated with an implementation of p-methods. It is conceivable that a clever use of p-method philosophy in this computing environment may prove to be very effective for a wide class of problems.

There is another point of view that is emerging from the adaptive literature: that is that two general types of a-posteriori estimation can be used in effective adaptive procedures. In one case, only a rather crude error estimator may be satisfactory to establish trends in mesh adaptation that will lead to improved accuracies. The effectivity indices for such methods may not be close to unity, so that the actual error predicted may be quite far from the true error that exists in the approximate solution. Nevertheless, a scheme may result which is truly adaptive, in the sense that the actual local error is systematically reduced below some threshold. Parallel to these methods are methods in which a great deal of sophistication is used in an a-posteriori error estimation. Here, with additional expense, quite accurate estimates of local errors can be obtained. This leads one to speculate that there may emerge in the future adaptive codes with two or three levels of sophistication: one in which an adaptive scheme is used to produce a near optimal solution for a fixed level of effort; secondly, a post-processing operation in which very precise estimates of the local error are produced and presented, perhaps in terms of error contours, for residual evaluation toward obtaining a final evaluation of the quality of the solution. Again, if this quality is not acceptable to the analyst, he may choose to re-run the problem through additional adaptive cycles to produce furhter improvement in local solution quality.

Acknowledgements. The support of this work under contracts N00014-84-K-0409 from ONR and NAS3-24849, NAS-8-36647 from NASA is gratefully acknowledged. We also wish to thank Mrs. Franziska Haug for typing this manuscript.

REFERENCES

1. Babuska, I., Zienkiewicz, O.C., Gago, J.P. de S. R., and Oliveira, A., (Eds.), **Adaptive Methods and Error Refinement in Finite Element Computation**, John Wiley and Sons, Ltd, London, 1986.

2. Baker, A.J., and Kim, J.W., "Analyses on a Taylor Weak-Statement Algorithm for Hyperbolic Conservation Laws", *Technical Report, CFDL* 86-1, Comp. Fluid Dyn. Lab., Univ. of Tennessee, 1986.

3. Bey, K.S., Thornton, E.A., Dechaumpai, P., and Ramakrishnan, R., "A New Finite Element Approach for Prediction of Aerothermal Loads -- Progress in Inviscid Flow Computations", **AIAA 85-1533-CP.**

4. Book, D.L., Boris, J.P., and Hain, K., "Flux-Corrected Transport II: Generalizations of the Method", Journal of Computational Physics 18, 1975.

5. Boris, J.P., and Book, D.L., "Flux-Corrected Transport I. SHASTA, A Fluid Transport Algorithm that Works", Journal of Computational Physics 11, 1973.

6. Boris, J.P., and Book, D.L., "Flux-Corrected Transport III. Minimal Error FCT Algorithms", Journal of Computational Physics 20, 1976.

7. Burstein, S.Z., "Finite Difference Calculations for Hydrodynamic Flows Containing Discontinuities", Journal of Computational Physics 2, 1967.

8. Davis, S.F., and Flaherty, J.E., "An Adaptive Finite Element Method for Initial Value Problems for Partial Differential Equations", *SIAM Journal Sci. Stat. Comput.*, 3, 1982.

9. Demkowicz, L., and Oden, J.T., "An Adaptive Characteristic Petrov-Galerkin Finite Element Method for Convection-Dominated Linear and Nonlinear Parabolic Problems in One Space Variable", **TICOM Report 85-3**, The Univ. of Texas at Austin, April, 1985.

10. Demkowicz, L., and Oden, J.T., "An Adaptive Characteristic Petrov-Galerkin Finite Element Method for Convection-Dominated Linear and Nonlinear Parabolic Problems in Two Space Variables", *Comput. Meths. Appl. Mech. Engrg.*, 55, 1986.

11. Devloo, Ph., Oden, J.T., Strouboulis, T., "Implementation of an Adaptive Refinement Technique for the SUPG Algorithm", *Comput. Meths. Appl. Mech. Engrg.*, (to appear).

12. Diaz, A.R., Kikuchi, N., and Taylor, J.E., "A Method of Grid Optimization for Finite Element Methods", *Comput. Meths. Appl. Mech. Engrg.*, 41, 1983.

13. Donea, J., "A Taylor-Galerkin Method for Convective Transport Problems, *Internat. J. Numer. Meths. Engrg.*, 20, 1984.

14. Guo, B., and Babuksa, I., "The h-p Version of the Finite Element Method", **Note BN-1043**, Inst. Physical Sc. and Tech., Univ. of Maryland, July, 1985.

15. Hughes, J.T.R., Brooks, A., "A Theoretical Framework for Petrov-Galerkin Methods with Discontinuous Weighting Functions. Application to the Streamline Upwind Procedure", In: Gallagher, R.H., et al. (Ed.), "**Finite Elements in Fluids - Volume 4**", John Wiley and Sons, Ltd., London, 1982.

16. Löhner, R., Morgan, K., and Zienkiewicz, O.C., "Adaptive Grid Refinement for the Euler and Compressible Navier-Stokes Equations", In: Babuska, I. et al. (Ed.), **Adaptive Methods and Error Refinement in Finite Element Computations**, John Wiley and Sons, Ltd., London, 1986.

17. Löhner, R., Morgan, K., and Zienkiewicz, O.C., "An Adaptive Finite Element Procedure for Compressible High Speed Flows", *Comput. Meths. Appl. Mech. Engrg.*, 51, 1985.

18. Löhner, R., Morgan, K., Vahdati, M., Boris, J.P., and Book, D.L., "FEM-FCT: Combining Unstructured Grids with High Resolution".

19. Oden, J.T., Strouboulis, T., and Devloo, Ph., "Adaptive Finite Element Methods for the Analysis of Inviscid Compressible Flow: I. Fast Refinement / Unrefinement and Moving Mesh Methods for Unstructured Meshes", **TICOM Report 86-3**, The Univ. of Texas at Austin, 1986.

20. Oden, J.T., and Demkowicz, L., "Advances in Adaptive Improvements: A Survey of Adaptive Finite Element Methods in Computational Mechanics", **State-of-the-Art Surveys in Computational Mechanics**, Special ASME Publication, ASME, N.Y., 1987 (to appear).

21. Oden, J.T., Demkowicz, L., Strouboulis, T., and Devloo, Ph., "Adaptive Methods for Problems in Solid and Fluid Mechanics", In: Babuska, I. et al. (Ed.), **Adaptive Methods and Error Refinement in Finite Element Computations**, John Wiley and Sons, Ltd, London, 1986.

22. Oden, J.T., and Carey, G.F., "**Finite Elements: Mathematical Aspects**", Volume IV, Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1983.

23. Strouboulis, T., Devloo, Ph., and Oden, J.T., "A Moving-Grid Finite Element Algorithm for Supersonic Flow Interaction Between Moving Bodies", *Comput. Meths. Appl. Mechs. Engrg.*, (to appear)

24. Strouboulis, T., "Adaptive Finite Element Methods for Flow Problems in Regions with Moving Boundaries, Ph.D. Dissertation, The Univ. of Texas at Austin, August, 1986.

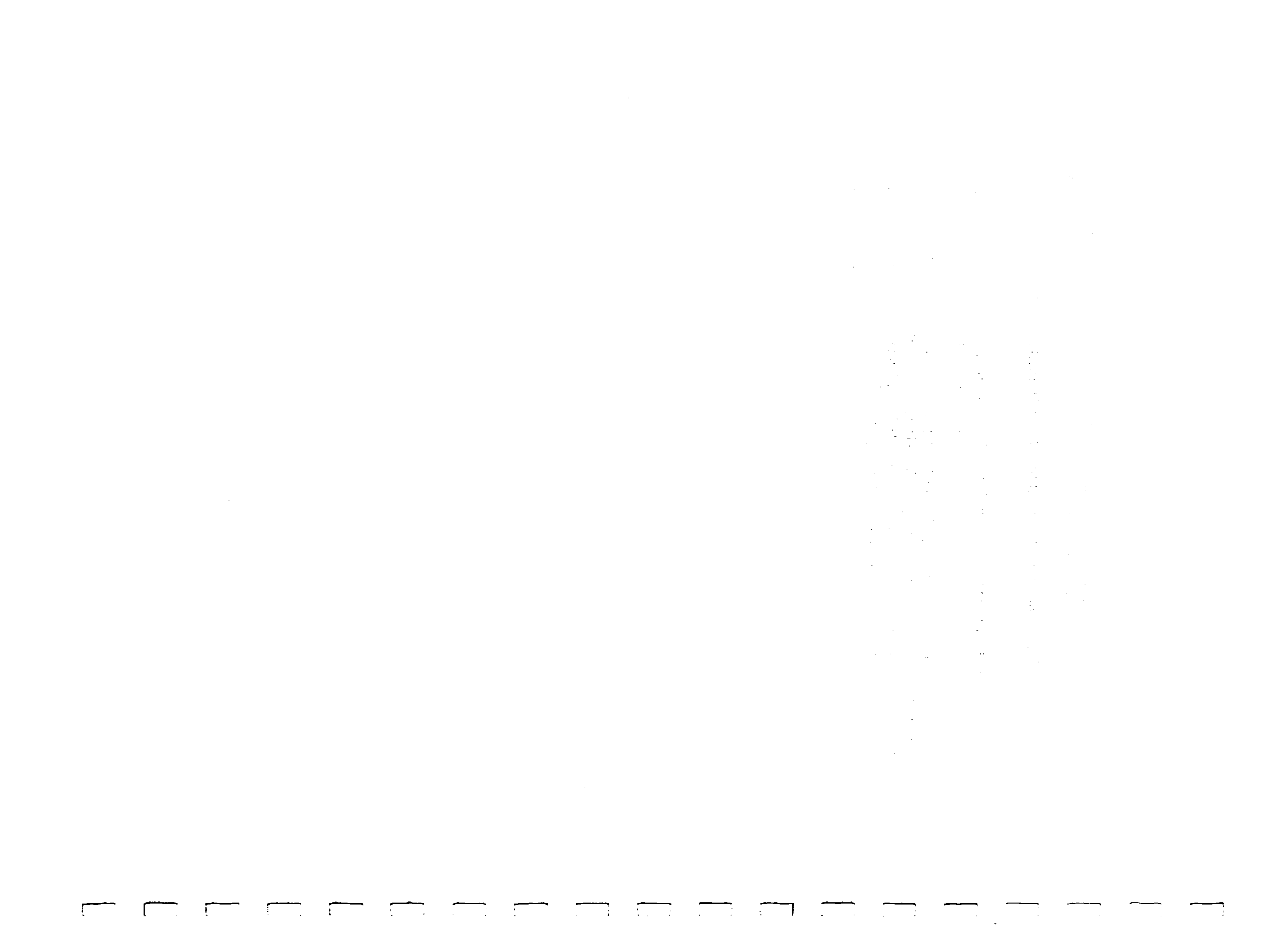
25. Szabo, B.A., **PROBE: Theoretical Manual**, Noetic Technologies, St. Louis, Missouri, 1985.

26. Zalesak, S.T., "Fully Multidimensional Flux-Corrected Transport Algorithm for Fluids", *Journal of Computational Physics*, 31, 1979.

27. Oden, J.T., Strouboulis, T. and Devloo, Ph., "Adaptive Finite Element Methods for the Analysis of Inviscid Compressible Flow: Fast Refinement/ Unrefinement and Moving Mesh Methods for Unstructured Meshes," Computational Methods in Applied Mechanics and Engineering, Vol. 59, No. 3, 1986, pp. 327-362.

Appendix B

ADAPTIVE MESH STRATEGY



Appendix B

The adaptive mesh strategy to be described is an h-method applied to hexahedral elements for three-dimensional and quadrilaterals for two-dimensionals wherein the mesh is refined or unrefined (coarsened) when a solution quality test function falls outside preassigned upper and lower bounds. For clarity the two-dimensional strategy is described. It differs from the three-dimensional strategy only in the number of elements which comprise a group (4 vs 8) and the number of new sub-elements created during a refinement (4 vs 8). A set of "adaptation" rules are listed which are used to implement this strategy.

1.1 General Description. The adaptive mesh strategy involves the following steps.

1. For a given domain L, such as that shown in Fig. B-1a, a coarse finite element mesh and an initial solution are available.

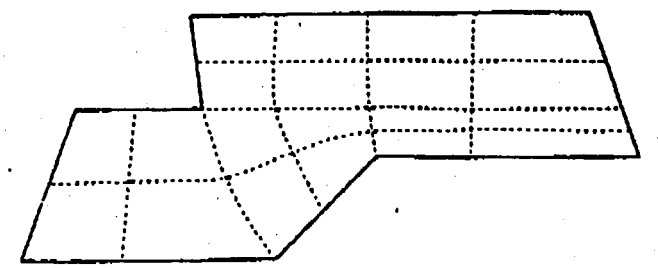
2. As the adaptive process will be designed to handle groups of four elements at a time, a finer starting grid is generated by a bisection process, indicated in Fig. B-1b, to obtain an initial set of element groups.

3. The adaptive procedure is initiated by computing solution quality indicators r_3 over all M elements in the grid. Let

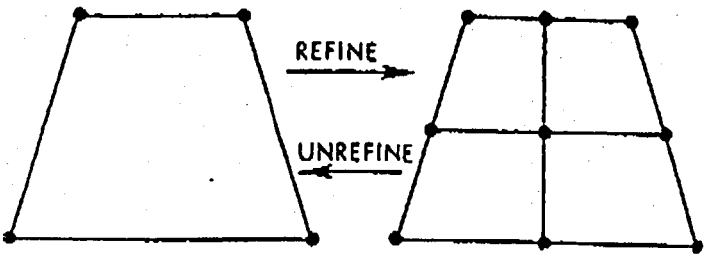
$$r_{MAX} = \max_{1 \le e \le M} r_e$$

4. Next, scan groups of a fixed number P of elements and compute

$$r^{k}_{GROUP} = \sum_{k=1}^{P} r_{e_k}$$



(a)



(b)

Fig. B-1 (a) A Coarse Initial Mesh Consisting of Four-Element Groups and (b) The Refinement and Unrefinement of a Group of Elements

where e_k is the element number for group k, P = 4 for two-dimensional grids and P = 8 for three-dimensional grids.

5. The solution quality bounds are defined by two real numbers, 0 a, ß 1. If

$$r_e \leq ß\, r_{MAX}$$

element r_e is <u>refined</u>. This is done by bisecting r_e into four new sub-elements. If

$$r^k_{GROUP} = ar_{MAX}$$

group k is <u>unrefined</u> by replacing this group with a single new element with nodes coincident with the corner nodes of the group. This is always possible because each group is itself the result of an initial bisectioning.

This general process can be followed for any choice of a solution quality indicator.

2.2 <u>Data Structures</u>. An important consideration in all adaptive schemes is the data structure and associated algorithms needed to handle the changing number of elements, their node locations and numbers, and the element labels.

As noted in the preceding paragraphs, the algorithm is designed to process (refine or unrefine) in groups of four elements at each local refinement/unrefinement step. Consider, for example, the case of an initial mesh of 20 square elements shown in Fig. B-2. Assign to each element in this mesh an element number, NEL = 1,2,...,NELEM and to each <u>global</u> node a label NODE. The array, NODES(J,NEL) relates the local node number J(J = 1,2,3,4) of element NEL to the global node number NODES. In addition, the coordinates X_J,Y_J

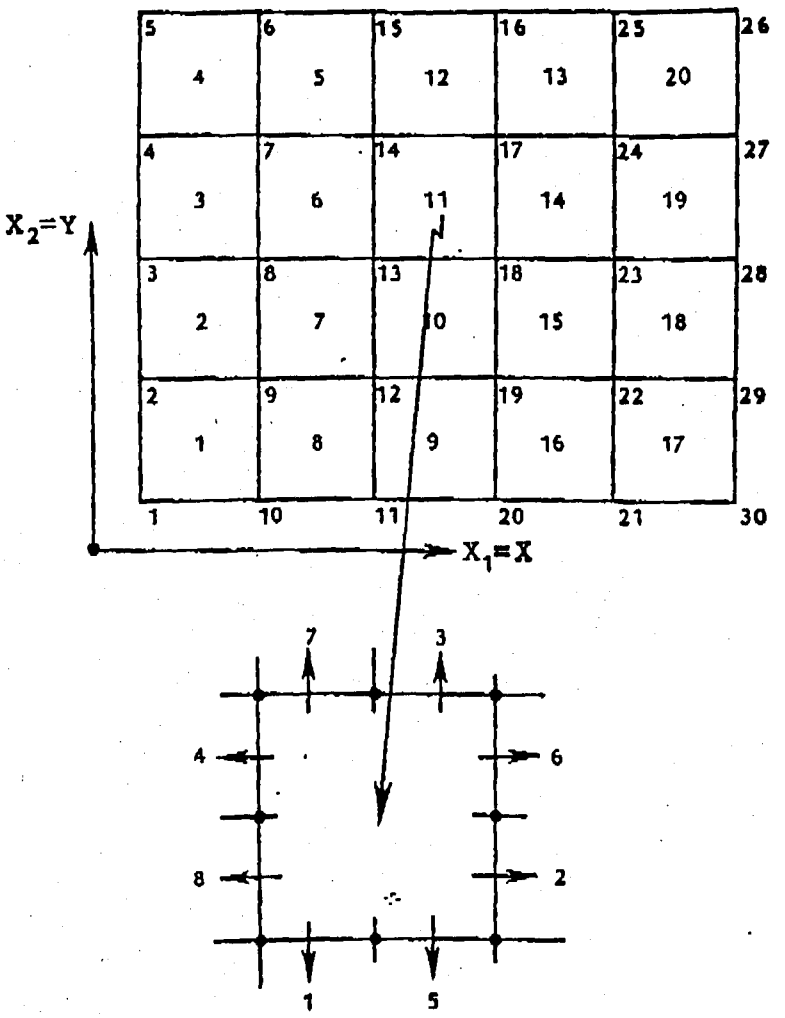


Fig. B-2 Mesh, Node, and Connectivity Numbering in a Model Problem

of each node are also provided relative to a fixed global coordinate system. File these numbers in two arrays,

NODES(J,NEL) = the array of global node number assigned to node J of element NEL

XDO(JCO,NODE) = the array of JCO -- coordinates of global node NODE(JCO = 1 or 2).

If a solution quality indicator signals that an element should be refined, say element 11 in the example, some system for assigning appropriate labels to the new elements and nodes must be devised. Toward this end, a convention can be established that defines the connectivity of the specified element with its neighbors in the mesh. This information is provided by a third connectivity array,

NELCON(NC,NEL) = the NC^{th} connection of element NEL, where NC = 1,2,...,8

As seen in Fig. B-2, each side of an element may be connected to two other elements so that NELCON is dimensioned thusly;

NELCON(8,MAXEL)

with MAXEL an appropriately large number.

The entire refinement process (or its inverse -- the unrefinement process) just described is accomplished by specifying a series of element levels. For example, the initial coarse mesh could be assigned level 0. When an element is refined, its sub-elements belong to a higher level, level 1, and when these sub-elements are refined, elements of level 2 result, and so on. In this way, if the maximum level any element in the mesh can achieve is limited, then the maximum number of elements the mesh can contain is also limited. In general, no such limit need be set.

Thus, the bookkeeping of element and node numbers evolved in a refinement process is monitored by the arrays NODES(.,.), XCO(.,.), NELCON(.,.), and an array LEVEL(NEL) which assigns a level number to element NEL. Initially, the same level can be assigned to all elements, and this level is an arbitrary parameter prescribed in advance by the user. Thus, provisions are now in hand for an arbitrary, dynamic renumbering of elements and nodes.

2.3 Adaptation Rules. Several rules must be established to successfully implement the refinement or coarsening of a mesh. The following "element" rules are employed:

1. An element may be refined only if its neighbors are at the same refinement level or higher.
2. If a "neighbor" element of an element to be refined is at a lower level of refinement, it must be refined first.
3. Refinement of an element results in creation of eight sub-elements for three-dimensional and four sub-elements for two-dimensional meshes.
4. To be eligible for coarsening a group of elements must not contain another group of elements and each element of the group to be coarsened must not be connected to a "neighbor" element of a higher level.

For example, if element 11 if Fig. B-2 is to be refined, we proceed through the following steps:

1. An intermediated node is common to two members of a group only.
2. An intermediate node that is created along a domain boundary cannot be constrained.
3. If an element and its neighbor both of which are at the same level are connected to a third element at a lower level, then the intermediate node which exists along the edge common with the third element is constrained.
4. If a group of elements is eligible for coarsening, then the intermediate constrained node along the edge common to an element which is not a member of the group will be eliminated.

5. If a group of elements is eligible for coarsening, then the node along the edge common to this group and its neighbor group will become constrained.

6. If a group of elements is eligible for coarsening, then the intermediate node along a domain boundary edge is eliminated.

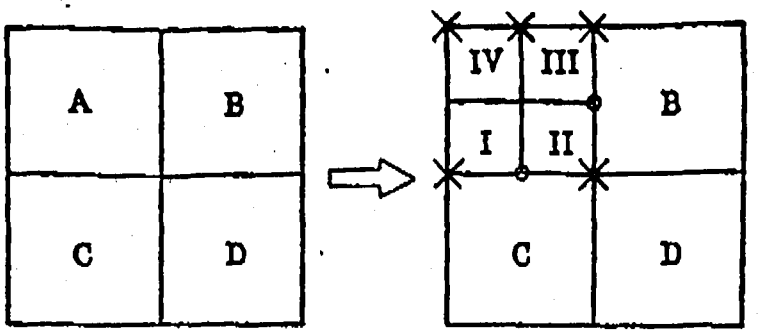
Use of the above rules can be illustrated by considering the uniform grid of four elements shown in Fig. B-3a. Suppose element A is marked for refinement. By applying element rules 1 and 3, element A is divided into subelements, I, II, III, IV as shown. Application of node rules 1 and 2 dictates that the nodes marked by circles be constrained. Nodes marked X are unconstrained.

Next, let element III be chosen for further refinement. Element III cannot be refined since one of its neighbors, B is at a lower level. Refinement of element III before element B would violate element rule 1. Therefore, element B is refined as shown in Fig. B-3b. Note that node B is no longer constrained, since node rule 2 no longer is satisfied. Node C1 remains constrained.

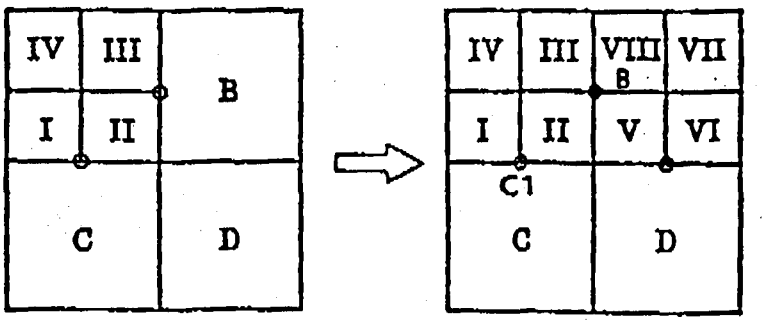
Now that element B has been divided into elements V, VI, VII, VIII, element rule 1 can be applied. Figure B-3c illustrates this division.

Suppose the group of elements V, VI, VII, VIII shown in Fig. B-3c is marked for coarsening. This group is not eligible for coarsening until the group of elements, a, ß, Y, w has been coarsened. Element VII has neighbors ß and w which are a higher level. This violates element rule 4.

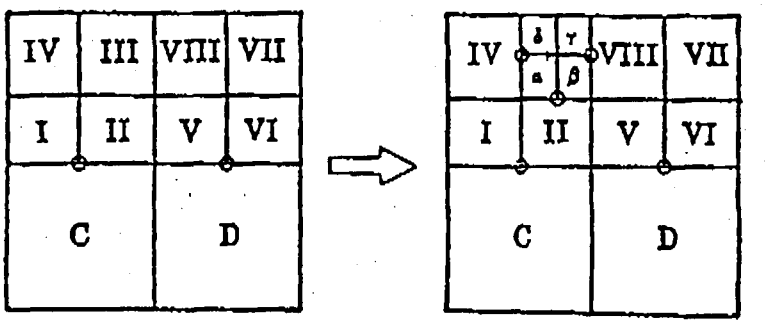
Now, let the group of elements, a, ß, Y, w be marked for coarsening. Element rule 4 is satisfied and elements a, ß, Y, w are replaced by element III. The intermediate constrained nodes associated with elements a, ß, Y, w are eliminated through use of node rule 4. The intermediate node along the upper domain boundary is eliminated using node rule 6.



(a)



(b)



(c)

Fig. B-3 Sequence of Refinements of a Uniform Mesh

Appendix C

ADAPTIVE FINITE ELEMENT METHODS FOR THE ANALYSIS OF INVISCID COMPRESSIBLE FLOW

Appendix C

I. Fast Refinement/Unrefinement and Moving Mesh Methods for Unstructured Meshes

J.T. Oden, P. Devloo, and M. Howe

Texas Institute for Computational Mechanics,
Department of Aerospace Engineering
and Engineering Mechanics
The University of Texas at Austin

ABSTRACT. New adaptive finite element methods are presented for the analysis of unsteady inviscid compressible flow in arbitrary two-dimensional domains. The procedures described herein are used in conjunction with a semi-explicit two-step algorithm for solving the time-dependent Euler equations in two space dimensions. Two schemes are presented for monitoring the evolution of error, and error estimates are used as a basis for a mesh refinement strategy. The capability of unrefinement (adaptively coarsening the mesh) is also included. The methods do not require a structured mesh and are applicable to quite general geometries.

1. INTRODUCTION

Many would agree that the most fundamental and important questions facing users of modern computational methods for flow predictions are the following:

I. How good are the answers?

II. How can one obtain the best possible answers for a fixed computational effort (or a fixed computing budget, fixed manpower level, or a fixed and limited computing capability)?

The first question is exceedingly difficult since it includes both the issue of the validity of the physical and mathematical model of the flow phenomena itself as well as the issue of the quality of the numerical approximation of the equations characterizing the model. To simplify matters for purposes of the present discussion, we shall dispense with the first issue and take for granted that the classical Navier-Stokes or, in the present paper, the Euler equations are adequate models of nature for the applications in mind. Thus, the first question reduces to a word: accuracy — how accurate are the numerical solutions?

The second question is seldom asked, but it is intrinsically connected to the first. It is common practice in applications of computational fluid dyanmics to the complex flow domains, to generate extremely fine finite difference meshes in hopes of capturing all important features of the flow, even though the location of these special points of interest changes in time. This leads some to employ fine meshes in

all positions of the flow domain where some important aspect of the flow might possibly manifest itself. The quality of results is generally judged by the invariance of solutions to further refinement if one can afford the cost of another calculation. The fact that coarse mesh solutions may be adequate in much of the domain at most instants of time cannot be exploited in traditional fixed mesh schemes.

After some thought about these issues, rather broad answers to the fundamental questions present themselves:

I. Accuracy. To determine the accuracy of a computed solution, one can attempt to develop reliable a-posteriori estimates of local error. In other words, one might hope to be able to develop procedures which use the evolving computed solution to determine sharp estimates of local errors in various norms over each mesh cell and at each time step.

II. "Optimal" Meshes. Use adaptive procedures to continually change the structure of the mesh -- the size of mesh cells, the location of grid points -- so as to keep the local errors within a preassigned limit.

Obviously, the second answer assumes that one has some means to measure the local quality of the numerical solution and, therefore, presumes the availability of some type of *a-posteriori* error estimate.

We describe, in this paper, algorithms and results developed in an attempt to more sharply resolve these answers, particularly that to question II, for a class of problem in compressible flow. More specifically, we describe here a class of very effective adaptive schemes for time-dependent Euler equations in two dimensions which employ both

mesh refinement (when the local error is large) and mesh "unrefinement" (when the local error is small) and which generate the appropriate mesh changes as the solution evolves in time. This requires that we estimate the local approximation errors at each time step. However, only an indication of the relative error between successive meshes is essential in our methods; the issue of very sharp *a-posteriori* estimates of local error (our answer to question I) is one of great concern to us and is the subject of other papers [8,18,19].

In designing an adaptive scheme for Euler equations, we keep the following guidelines in mind:

(1) Unstructured Grids. The method must be virtually grid independent and global-coordinate free. While an initial mesh can be defined to model the basic geometry of the flow domain and the initial data, thereafter it must be possible to automatically add or eliminate cells and grid points as needed to monitor local accuracy levels. This requirement considerably lessens the attractiveness of body-fitted coordinates, many elliptic/algebraic mesh generators, and various factorization algorithms which exploit such regular mesh topologies.

(2) General Geometries and Boundary Conditions. The method must be applicable to arbitrary flow domains with virtually arbitrary geometry, general in-flow and out-flow conditions, and general boundary conditions.

(3) Solid Mathematical Basis. Since, by its nature, any sound adaptive method must employ some type of local error

estimator, it is important that the methods employed have a reasonably firm mathematical basis, e.g., that *a-priori* or *a-posteriori* error estimates exist and that the convergence characteristics of the method are acceptable.

(4) High Accuracy. The method should be capable of delivering high-order accuracy.

(5) Robustness. The method must be numerically stable and not sensitive to singularities, distortions in the mesh, or irregularities in the data.

(6) Supercomputing. The method should lend itself to modern supercomputing methods for accelerating computational speed, such as easy vectorization or implementation on parallel processors, etc.

(7) Computational Efficiency. The method and the supporting algorithms and data structures must be computationally efficient.

We feel that these criteria can be best met by finite element methods. In the present work, we use as the basis of our adaptive schemes a semi-explicit method used by several other authors (e.g., [19,4,15,16,21]): a two-step Lax-Wendroff/Taylor-Galerkin scheme. It is far from optimal (and does not satisfy all of our criteria), but is perfectly adequate to use in conjunction with our adaptive scheme. Schemes which fulfill all of these criteria are under development and will be reported in subsequent papers.

We remark that there is a growing literature on adaptive methods in computational fluid mechanics. Adaptive procedures for incompressible viscous flow problems were developed by the authors in a series of recent

papers (see, e.g., [8,18,19]). These methods employed a variety of different adaptive strategies, but did not come as close to satisfying the above criteria as the methods discussed in the present work. The general subject of adaptive finite element methods is dealt with in a forthcoming volume of collected papers edited by Babuska, Zienkiewicz, Gago, and Oliveira [2]. For a survey of adaptive finite difference schemes, see the works of Anderson [1]. Also, Berger and Oliger [4] and Berger and Jameson [5] have recently developed adaptive finite difference methods for hyperbolic conservation laws. Still other types of adaptive methods for hyperbolic problems have been recently proposed by Demkowicz and Oden [10,11].

Following this Introduction, we develop weak formulations of a class of problems in compressible gas dynamics. These space-time formulations are shown to be the basis of a class of Lax-Wendroff/Taylor Galerkin schemes. Our derivation of this family of algorithms is nonstandard, in that we show that a two-step scheme follows easily from the use of a numerical quadrature scheme for evaluating appropriate flux integrals. In Section 3, finite element nodels of the space-time formulation are introduced, and in Section 4, we discuss the important issue of *a posteriori* error estimation. Section 5 of this paper is devoted to a detailed discussion of adaptive strategies. These include an h-method, wherein the mesh is refined or unrefined when local errors fall outside a preassigned upper and lower bound, and an r-method, in which the mesh is automatically distorted to equidistribute error. In Section 6 of the paper, we present the results of several numerical experiments on two-dimensional problems. These results illustrate that the performances of the adaptive schemes are quite acceptable for a class of complex flow problems.

2. PRELIMINARIES

We consider the motion of a perfect gas flowing through a domain Ω over a time interval $[0,T]$. We shall confine our attention to two-dimensional cases, $\Omega \subset \mathbb{R}^2$; we denote by D the space-time domain, $D = \Omega \times (0,T)$ and by $\partial\Omega$ the boundary of Ω . The motion of the gas is governed by the global balance laws of physics and the second law of thermodynamics. Thus, if $\underset{\sim}{U} = \underset{\sim}{U}(\underset{\sim}{x},t)$, $(\underset{\sim}{x},t) \in D$, is the 4-vector of conservation variables, $\underset{\sim}{U} = \{\rho , \underset{\sim}{m} , E\}^T$, with ρ the mass density, $\underset{\sim}{m}$ the linear momentum, and E the total energy, and if $d\Omega$ and dS denote Lebesque measures of area (volume) and length (area) of Ω and $\partial\Omega$ respectively, then we demand that $\underset{\sim}{U}$ satisfy the following system of conservation laws:

$$\frac{d}{dt}\int_{\Omega} \underset{\sim}{U}\, d\Omega = -\int_{\partial\Omega} \underset{\sim}{Q}(\underset{\sim}{U})\underset{\sim}{n}\, dS \tag{2.1}$$

Here, $\underset{\sim}{Q}(\underset{\sim}{U})$ is the flux and $\underset{\sim}{n}$ is the unit outward normal to $\partial\Omega$. If m_1 , m_2 denote Cartesian components of $\underset{\sim}{m}$, then

$$\underset{\sim}{U} = \{\rho , m_1 , m_2 , E\}^T$$

$$\underset{\sim}{Q}(\underset{\sim}{U}) = \left[\begin{array}{c|c} m_1 & m_2 \\ \rho^{-1}m_1^2 + p(\underset{\sim}{U}) & \rho^{-1}m_1m_2 \\ \rho^{-1}m_1m_2 & \rho^{-1}m_2^2 + p(\underset{\sim}{U}) \\ \rho^{-1}m_1\left(E + p(\underset{\sim}{U})\right) & \rho^{-1}m_2\left(E + p(\underset{\sim}{U})\right) \end{array}\right] \tag{2.2}$$

$$\underset{\sim}{n} = \{n_1 , n_2\}^T \; ; \quad p(\underset{\sim}{U}) = (\gamma - 1)(E - \rho^{-1}\, \underset{\sim}{m} \cdot \underset{\sim}{m}/2)$$

In (2.2), p is the thermodynamic pressure and γ is the ratio of specific heats, assumed here to be constant. In addition to (2.1), $\underset{\sim}{U}$ must satisfy the entropy production inequality

$$\frac{d}{dt}\int_{\Omega} \rho\eta(\underset{\sim}{U})\, d\Omega + \int_{\partial\Omega} \underset{\sim}{n} \cdot \left(\underset{\sim}{m}\eta(\underset{\sim}{U}) + \theta^{-1}\, \underset{\sim}{q}\right) dS \geq 0 \qquad (2.3)$$

with $\eta(\underset{\sim}{U})$ the entropy density of the gas, θ the absolute temperature, and $\underset{\sim}{q}$ the heat flux, as well as an initial condition,

$$\underset{\sim}{U}(\underset{\sim}{x},0) = \underset{\sim}{U}_0(\underset{\sim}{x}) \quad , \qquad \underset{\sim}{x} \in \Omega \qquad (2.4)$$

where $\underset{\sim}{U}_0$ is given.

It is of fundamental importance to note the smoothness requirements on $\underset{\sim}{U}$ in order that (2.1) make sense mathematically. Conservation laws (2.1) hold when the components of $\underset{\sim}{U}$ are bounded measurable (with respect to Lebesque measure in $\underset{\sim}{x}$) functions on D . Thus, we may seek solutions in the function space

$$V = \{\underset{\sim}{V} = \{V_1, V_2, V_3, V_4\}^T \mid V_i = V_i(\underset{\sim}{x},t)$$

$$\in L^{\infty}\left(0,T\, ;\, L^1(\Omega)\right)\, ;\, i = 1, 2, 3, 4\} \qquad (2.5)$$

In particular, (2.1) is <u>not</u> equivalent to the classical Euler equations,

$$\underset{\sim}{U}_t + \operatorname{div} \underset{\sim}{Q}(\underset{\sim}{U}) = \underset{\sim}{0} \qquad (2.6)$$

(with $\underset{\sim}{U}_t = \partial\underset{\sim}{U}/\partial t$ and $\operatorname{div} \underset{\sim}{Q} = \sum_i \partial Q_{\alpha i}/\partial x_i$) since solutions of (2.1) may not possess derivatives across surfaces in D . However, the conservation laws and initial conditions are fully equivalent to the following weak boundary-initial value problem:

Find $\underset{\sim}{U} \in \underset{\sim}{V}$ such that

$$\int_D \left(\underset{\sim}{U}^T \underset{\sim}{\phi}_t + \underset{\sim}{Q}(\underset{\sim}{U}) : \underset{\sim}{\nabla}\underset{\sim}{\phi}\right) d\Omega dt$$

$$+ \int_\Omega \underset{\sim}{U}_0^T \underset{\sim}{\phi}(\cdot,0) d\Omega = \int_0^T \oint_{\partial\Omega} \underset{\sim}{F}^T \underset{\sim}{\phi} \, dS \, dt \qquad (2.7)$$

for all $\underset{\sim}{\phi} \in W$

where $\underset{\sim}{F}$ is the actual prescribed flux through $\partial\Omega$ and W is a suitable space of test functions; e.g.,

$$W = \{\underset{\sim}{\phi} = \{\phi_1, \phi_2, \phi_3, \phi_4\}^T \mid \phi_i = \phi_i(\underset{\sim}{x},t),$$

$$\phi_i \in C^1(\bar{D}) \ , \ \phi_i(\underset{\sim}{x},T) = 0 \ , \quad i = 1, 2, 3, 4\} \qquad (2.8)$$

In (2.7), we use the notation

$$\underset{\sim}{U}^T \underset{\sim}{\phi}_t = \sum_{\alpha=1}^{4} U_\alpha \frac{\partial \phi_\alpha}{\partial t} \quad ; \quad \underset{\sim}{Q} : \underset{\sim}{\nabla}\underset{\sim}{\phi} = \sum_{i=1}^{2} \sum_{\alpha=1}^{4} Q_{\alpha i} \frac{\partial \phi_\alpha}{\partial x_i}$$

On the other hand, if $\underset{\sim}{U}$ is known to have integrable derivatives in D everywhere except on a family of surfaces $\{\Gamma_k\}_{k=1}^R$, then we may consider the problem

Find $\underset{\sim}{U} \in V$ such that

$$\int_D \left(\underset{\sim}{U}_t^T \underset{\sim}{\phi} + \left(\text{div } \underset{\sim}{Q}(\underset{\sim}{U})\right)^T \underset{\sim}{\phi}\right) d\Omega \, dt - \int_\Omega \underset{\sim}{U}^T(\cdot,0) \, \underset{\sim}{\phi}(\cdot,0) d\Omega$$

$$- \sum_{k=1}^{R} \int_{\Gamma_k} \underset{\sim}{\phi}^T \{s_k \, \|\underset{\sim}{U}\| - \|\underset{\sim}{Q} \, \underset{\sim}{n}\|\} dS \, dt$$

$$+ \int_\Omega \underset{\sim}{U}_0^T \underset{\sim}{\phi} \, d\Omega + \int_0^T \int_{\partial\Omega} (\underset{\sim}{Q}\underset{\sim}{n})^T \underset{\sim}{\phi} \, dS \, dt$$

$$= \int_0^T \int_{\partial\Omega} \underset{\sim}{F}^T \underset{\sim}{\phi} \, dS \, dt \qquad \text{for all } \underset{\sim}{\phi} \in W \qquad (2.10)$$

Here, s_k are the speeds of propagation of discontinuities across Γ_k and V and W are appropriately redefined, e.g.,

$$V = \{\underset{\sim}{V} = (V_1, V_2, V_3, V_4) \mid V_i \in L^\infty(D) ,$$

$$V_i \in H^1(D_{ki}) , \quad D_{ki} = D - \Gamma_{ki}\}$$

$$W = \{\underset{\sim}{\phi} \mid \phi_i \in C^0(\bar{D}) , \quad \phi_i(\underset{\sim}{x},T) = 0\}$$

where Γ_{ki} are the surfaces on which V_i suffers a jump. If $\underset{\sim}{F}$ is not a prescribed flux but is merely a notation for $\underset{\sim\sim}{Q}\underset{\sim}{n}$, then these flux terms cancel and do not appear in the formulation.

Consider an arbitrary time interval $[\tau_1,\tau_2] \subset [0,T]$ and include in W functions $\phi(\underset{\sim}{x},\tau_2) \neq 0$. Let ω be a subset of Ω such that $\omega \cap \bigcup_k \Gamma_k = \emptyset$, and let $\underset{\sim}{F} \equiv \underset{\sim\sim}{Q}\underset{\sim}{n}$. Then another weak statement of the system conservation laws over $\bar{\omega} \times [\tau_1,\tau_2]$ is:

Find $\underset{\sim}{U} \in V^{\omega,\tau}$ such that

$$\int_{\tau_1}^{\tau_2} \int_\omega \left(-\underset{\sim}{U}^T \underset{\sim}{\phi}_t + (\operatorname{div} \underset{\sim\sim}{Q})^T \underset{\sim}{\phi}\right) d\Omega \, dt$$

$$+ \int_\omega \left(\underset{\sim}{U}^T(\cdot,\tau_2) \, \underset{\sim}{\phi}(\cdot,\tau_2) - \underset{\sim}{U}^T(\cdot,\tau_1) \, \underset{\sim}{\phi}(\cdot,\tau_1)\right) d\Omega = 0 \qquad (2.11)$$

$$\text{for all } \underset{\sim}{\phi} \in W^{\omega,\tau}$$

with $V^{\omega,\tau}$ and $W^{\omega,\tau}$ appropriate spaces of trial and test functions.

Remark. It is well known that (2.7), (2.10) and (2.11) may all possess non-physical solutions since none of these formulations involve the entropy inequality (2.3). Thus, in general, we seek solutions to (2.7), (2.10) or (2.11) in the subset $K \subset V$; $K = \{v \in V , v$ satisfies (2.3) for appropriate θ , $\underset{\sim}{q}(\nabla\theta)\}$.

3. FINITE ELEMENT APPROXIMATIONS

Finite element approximations of the gas dynamics problem are obtained by a direct approximation of (2.10) or (2.11) on finite-dimensional spaces approximating the spaces V and W. The spatial domain Ω is partitioned into a collection T_h of finite elements Ω_e over which the components of trial functions $\underset{\sim}{V}$ are approximated by polynomials of degree k. In this way, we construct a family $\{V_h\}$ of finite dimensional spaces of the type

$$V_h = \{\underset{\sim}{v}^h = \{v_1^h, v_2^h, v_3^h, v_4^h\}^T \in V \mid$$

$$v_i^h \in P_k(\Omega_e), \quad i = 1, 2, 3, 4\} \tag{3.1}$$

where $P_k(\Omega_e)$ is the space of polynomials of degree k defined over Ω_e. Alternatively, we can use $v_i^h|_{\Omega_e} \in Q_k(\Omega_e)$, where $Q_k(\Omega_e)$ is the space of tensor products of polynomials of degree k on Ω_e (e.g., $Q_1(\Omega_e)$ is spanned by bilinear functions, $Q_2(\Omega_e)$ by biquadratics, etc.). In addition, a family $\{W_h\}$ of finite dimensional spaces of test functions is also constructed. We then consider Galerkin approximations of (2.7), (2.10) or (2.11) by seeking solutions to these equalities in V^h, with V and W replaced by V^h and W^h, respectively.

3.1 A Two-Step, Lax-Wendroff/Taylor-Galerkin Scheme. We next derive a special semi-discrete, weak formulation from (2.11) which provides the basis for the construction of a popular family of finite element schemes. We proceed with the following steps:

i) Partition the time interval $[0,T]$ according to $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$;

ii) Apply the weak balance law (2.11) to a typical time interval $[t_n, t_{n+1}]$ (with $\tau_1 = t_n$ and $\tau_2 = t_{n+1}$);

iii) Set $\underset{\sim}{\phi}_t = \underset{\sim}{0}$ in (2.11) suggesting the ultimate use of a time-invariant grid (we relax this assumption later);

iv) Replace the time integrations in (2.11) by the elementary midpoint quadrature rule

$$\int_{t_n}^{t_{n+1}} f(t)dt \simeq \Delta t \, f^{n+\frac{1}{2}}$$

$$\Delta t = t_{n+1} - t_n \, , \quad f^{n+\frac{1}{2}} = f(t_n + \Delta t/2)$$

Thus, with $\omega = \Omega$, we obtain the semidiscrete approximation

$$\int_\Omega \underset{\sim}{\phi}_h^T \underset{\sim}{U}^{n+1} \, d\Omega = \int_\Omega \underset{\sim}{\phi}_h^T \underset{\sim}{U}^n \, d\Omega + \Delta t \int_\Omega \underset{\sim}{Q}^{n+\frac{1}{2}} : \underset{\sim}{\nabla}\underset{\sim}{\phi}_h \, d\Omega$$

$$- \Delta t \oint_{\partial\Omega} \underset{\sim}{\phi}_h^T (\underset{\sim}{Q}^{n+\frac{1}{2}} \, \underset{\sim}{n}) \, dS$$

$$\text{for all } \underset{\sim}{\phi}_h \qquad (3.2)$$

where $\underset{\sim}{U}_h^n = \underset{\sim}{U}_h(\underset{\sim}{x}, t_n)$, etc., $\underset{\sim}{U}_h$ being the approximation of $\underset{\sim}{U}$, and $\underset{\sim}{Q}^{n+\frac{1}{2}}$ is the flux at the half step,

$$\underset{\sim}{Q}^{n+\frac{1}{2}} = \underset{\sim}{Q}(\underset{\sim}{U}_h^{n+\frac{1}{2}}) \qquad (3.3)$$

v) To obtain an approximation $\underset{\sim}{U}_h^{n+\frac{1}{2}}$, we use (2.11) again for time interval $[t_n, t_{n+\frac{1}{2}}]$, this time replacing the time integrals by a simple strip rule and integrating by parts the divergence terms

$$\int_{\Omega_e} \underset{\sim}{\phi}_h^T \underset{\sim}{U}_h^{n+\frac{1}{2}} \, d\Omega = \int_{\Omega_e} \underset{\sim}{\phi}_h^T \underset{\sim}{U}_h^n \, d\Omega$$

$$- \frac{\Delta t}{2} \int_{\Omega_e} \underset{\sim}{\phi}_h^T (\text{div } \underset{\sim}{Q}^n) d\Omega$$

$$\text{for all } \underset{\sim}{\phi}_h \qquad (3.4)$$

We thus arrive at the algorithm,

1) With $\left(\underset{\sim}{U}_h^n , \underset{\sim}{Q}^n = \underset{\sim}{Q}(\underset{\sim}{U}_h^n)\right)$ known at the n th time step, compute $\underset{\sim}{U}_h^{n+\frac{1}{2}}$ using (3.4)
2) Compute $\underset{\sim}{Q}^{n+\frac{1}{2}}$ using (3.3)
3) Compute $\underset{\sim}{U}_h^{n+\frac{1}{2}}$ using (3.4)
4) Go to 1)

This algorithm is the finite-element based two-step Lax-Wendroff/Taylor Galerkin scheme (see [20,7]). It is one of a family of methods advanced by Donea [12], studied by Baker and Kim [3], and successfully refined and used by Löhner et al. [15,16]) and Bey et al. [6] in finite-element applications in fluid dynamics. This semi-explicit method is of second order in time and can experience spurious oscillations near shocks and other types of irregularities in the solution. These deficiencies must be reckoned with in implementing the method.

3.2 Artificial Viscosity. As noted earlier, artificial viscosity terms are usually added to schemes such as that employed here so as to dampen out oscillations in the numerical solutions near shocks. The calculations described subsequently were performed adding a Lapidus-viscosity term, which, at time step t_n, is of the form

$$- \nabla \cdot \left(\underset{\sim}{c}(\underset{\sim}{u}) \cdot \nabla \underset{\sim}{U}^{n+\alpha}\right) \qquad (3.5)$$

where

$$c_i(\underset{\sim}{u}) = C \left| \frac{\partial u_i}{\partial x_i} \right| \quad \text{(no sum on } i \text{)}$$

C is the Lapidus constant, $u_i = m_i/\rho$ is the i^{th} component of the flow velocity, and α is a parameter which determines if viscosity is to be included implicitly ($\alpha = 1$) or explicitly ($\alpha = 0$). The viscosity term, written out in component form, is

$$- \sum_{k=1}^{2} \frac{\partial}{\partial x_k} \left(c_k(\underset{\sim}{u}) \frac{\partial}{\partial x_k} U_\beta^{n+\alpha} \right) ; \quad \beta = 1, 2, 3, 4$$

Setting $\alpha = 1$, we obtain for step 2 of the procedure (instead of (3.2)),

$$\int_\Omega \phi_h^T \underset{\sim}{U}_h^{n+1} \, d\Omega + \Delta t \int \sum_{\alpha=1}^{4} \sum_{i=1}^{2} c_i(\underset{\sim}{u}) \, U_{\alpha,1}^{n+1} \, \phi_{\alpha,i}^h \, d\Omega$$

$$- \Delta t \int_{\partial\Omega} \phi_h^T (\underset{\sim}{c}(\underset{\sim}{u}^n) \cdot \underset{\sim}{\nabla} \underset{\sim}{U}^n) \underset{\sim}{n} \, dS$$

$$= \int_\Omega \phi_h^T \underset{\sim}{U}_h^n \, d\Omega + \Delta t \int_\Omega \underset{\sim}{Q}^{n+\frac{1}{2}} : \underset{\sim}{\nabla} \phi_h \, d\Omega$$

$$- \Delta t \int_{\partial\Omega} \phi_h^T (\underset{\sim}{Q}^{n+\frac{1}{2}} \underset{\sim}{n}) d\Omega$$

for all admissible test functions ϕ_h.

3.3 Details of the Finite Element Algorithm. The details of the implementation of the algorithm described above are crucial to successful computations. In this work, we use meshes of four-node quadrilateral (Q_1) elements over which the components $U_\alpha (\alpha = 1,2,3,4)$ of $\underset{\sim}{U}$ are piecewise bilinear functions. Similar approximations and algorithms are used by Bey

et al. [6]. In addition, so-called group approximations of the flux $Q_{\alpha i}(\alpha = 1,2,3,4;\ i = 1,2)$ are employed so that these components are also piecewise bilinear functions determined by their values at element nodes. In general, this finite element approximation will be of the form,

$$U_\alpha \cong \sum_{j=1}^{N} U_\alpha^j(t)\ \phi_j(\underset{\sim}{x})$$

$$Q_{\alpha i} \cong \sum_{j=1}^{N} Q_{\alpha i}^j\ (t)\ \phi_j(\underset{\sim}{x}) \tag{3.7}$$

where N denotes the total number of nodes in the discretization, and U_α^j, $Q_{\alpha i}^j$ are values of $\underset{\sim}{U}^h$ $\underset{\sim}{Q}^h$ at node j, and ϕ_j are the global piecewise bilinear basis functions.

As noted earlier, we advance the solution in time in two steps. It is important to note that the first step is essentially local, computed over each element, while the second is global and contains the artificial viscosity terms:

First Step: For each element Ω_e, calculate a constant element vector $U_{\alpha,e}^{n+\frac{1}{2}}$ from

$$U_{\alpha,e}^{n+\frac{1}{2}} \int_{\Omega_e} d\Omega = \sum_{i=1}^{4} \{ \left(\int_{\Omega_e} \phi_i d\Omega \right) U_\alpha^{i,n}$$

$$- \frac{\Delta t}{2} \left(\int_{\Omega_e} \frac{\partial \phi_i}{\partial x_\beta} d\Omega \right) Q_{\alpha\beta}^{i,n} \} \tag{3.8}$$

Second Step: For each node j, calculate $U_\alpha^{j,n+1}$ by solving the following system of equations

$$\sum_{j=1}^{N} \left\{ \int_{\Omega} \left(\phi_i \phi_j + \tau_\beta \frac{\partial \phi_i}{\partial x_\beta} \frac{\partial \phi_j}{\partial x_\beta} \right) d\Omega \right\} U_\alpha^{j,n+1} = \sum_{j=1}^{N} \left(\int_{\Omega} \phi_i \phi_j \, d\Omega \right) U_\alpha^{j,n}$$

$$+ \Delta t \int_{\Omega} Q_{\alpha\beta}^{n+\frac{1}{2}} \frac{\partial \phi_i}{\partial x_\beta} \, d\Omega - \Delta t \int_{\partial\Omega} n_\beta \left(Q_{\alpha\beta}^{n+\frac{1}{2}} - \bar{Q}_{\alpha\beta}^{n} \right) \phi_i \, ds$$

$$- \Delta t \int_{\partial\Omega} n_\beta \, Q_{\alpha\beta}^{n} \, \phi_i \, ds \qquad (3.9)$$

Here, $\bar{Q}^n$ denotes the elementwise averaged value of the flux. The coefficients τ_β are defined to be constant over each element,

$$\tau_\beta = cA_e \left| \overline{\frac{\partial u_\beta^h}{\partial x_\beta}} \right|$$

where c is a global constant ($c = 1$ in the examples), A_e denotes the area of Ω_e, u_β^h denote the components of the fluid velocity.

To speed up the calculation, we precalculate and store the following element integrals before the time stepping is started:

$$\int_{\Omega_e} \phi_i \, d\Omega \,, \quad \int_{\Omega_e} \frac{\partial \phi_i}{\partial x_\beta} \, d\Omega$$

$$\int_{\Omega_e} \phi_i \phi_j \, d\Omega \,, \quad \int_{\Omega_e} \frac{\partial \phi_i}{\partial x_\beta} \frac{\partial \phi_j}{\partial x_\beta} \, d\Omega$$

$$i,j = 1,2,3,4; \quad \beta = 1,2,3,4$$

An element-by-element Jacobi Conjugate Gradient method is used to obtain the solution of the matrix problem in the second step. Due to the structure of the mass matrix, the iterative solver requires only a few iterations to converge fully.

3.4 Boundary Conditions. In the finite element schemes developed here, we implement the following three types of boundary conditions:

(a) Supersonic Inflow. On the part of the boundary with supersonic inflow, the values of all the conservation variables are imposed;

(b) Supersonic Outflow. On the outflow part of the boundary, the values of the conservation variables and the normal flux are unknown. Boundary conditions of supersonic outflow are implemented by adding the contribution of the boundary integral of the normal flux to the right-hand side of the equations of the second step; and

(c) Solid Boundaries. On a solid boundary, the normal component of the velocity $u_n = u_\beta n_\beta$ is zero. We note that, in general, the nodal directions are not uniquely defined. In such calculations, we compute the normal directions at the nodes which satisfy global mass conservation at the steady state, namely,

$$n_\beta^i = \int_\Omega \frac{\partial \phi_i}{\partial x_\beta} \, d\Omega \Big/ \sqrt{\sum_{\beta=1}^{2} \left(\int_\Omega \frac{\partial \phi_i}{\partial x_\beta} \, d\Omega \right)^2}$$

3.5 Hourglass Instabilities. We now show that the Taylor-Galerkin scheme presented above can propagate undetected spurious solutions. To demonstrate this, let us consider the scheme applied on the 2-D Burger's equations on a mesh of rectangular elements. Burger's equations may be obtained from the above formulation by redefining the flux as follows:

$$\{Q_{\alpha 1}\} = \begin{Bmatrix} 0 \\ U_2^2 \\ \\ U_2 U_3 \\ 0 \end{Bmatrix}, \quad \{Q_{\alpha 2}\} = \begin{Bmatrix} 0 \\ U_3 U_2 \\ \\ U_3^2 \\ 0 \end{Bmatrix}$$

Consider a rectangular element with the following nodal solution at time t_n :

$$\{U_\alpha^1\}^n = [0,\ 1,\ 1,\ 0]^t$$

$$\{U_\alpha^2\}^n = [0,\ 1,\ -1,\ 0]^t$$

$$\{U_\alpha^3\}^n = [0,\ 1,\ 1,\ 0]^t$$

$$\{U_\alpha^4\}^n = [0,\ 1,\ -1,\ 0]^t$$

Then the scheme gives,

$$\{U_\alpha^e\}^{n+\frac{1}{2}} = [0,\ 1,\ 0,\ 0]^t$$

and, by letting $c = 0$ (no artificial viscosity), we get:

$$\{U_\alpha\}^{n+1} = \{U_\alpha\}^n$$

This means that the scheme propagates "hourglass" solutions undetected. This fact explains why in the numerical examples the method produced oscillations near the outflow boundaries. This hourglassing phenomenon can be eliminated by considering each quadrilateral as a patch of two triangular elements joined along one diagonal of a quadrilateral.

4. ERRORS

The adaptive finite element methods described here involve two basic components:

1) Error estimation -- the determination of *a-posteriori* estimates of the evolution of error in the numerical solution; and

2) Adaptation -- the automatic restructuring of the approximation so as to reduce local element errors and the computational effort.

In this paper, adaptive procedures are based on estimates of error in a single principal dependent variable, such as the density, pressure or the entropy. We shall choose the density ρ as the driving factor in adaptivity, although other choices could be used in the algorithms developed here. Two basic procedures are used to estimate local element errors.

4.1 Evolution Equation for Error. Consider the continuity equation for the evolution of mass density through a domain Ω with known flow velocity $\underline{u}$. A weak form of the continuity equation is

$$\int_\Omega \phi\, \rho_t \, d\Omega = - \int_\Omega \nabla\cdot(\underline{u}\, \rho)\phi \, d\Omega$$

$$\text{for all} \quad \phi \in W \tag{4.1}$$

A semi-discrete Galerkin approximation of (4.1) consists of seeking an approximate density ρ^h such that, over some suitable finite-dimensional space of test functions W_h,

$$\int_\Omega \phi_h\, \rho_t^h \, d\Omega - \int_\Omega \nabla\cdot(\underline{u}\, \rho^h)\phi_h \, d\Omega$$

$$\text{for all} \quad \phi_h \in W_h \tag{4.2}$$

If $W_h \subset W$, we may choose $\phi = \phi_h$ in (4.1), subtract (4.2) and obtain the following evolution equation for the error $e^h(\underset{\sim}{x},t) = \rho(\underset{\sim}{x},t) - \rho^h(\underset{\sim}{x},t)$:

$$\int_\Omega \phi_h \, e^h_t \, d\Omega = \int_\Omega - \nabla\cdot(\underset{\sim}{u} \, e^h)\phi_h \, d\Omega$$

$$\text{for all } \phi_h \in W_h \tag{4.3}$$

The exact and approximate solutions are related according to

$$\rho = \rho^h + e^h \tag{4.4}$$

where e^h is the approximation error. Thus, the error satisfies the evolution equation,

$$\int_\Omega \left(\phi e^h_t + \underset{\sim}{\nabla}\cdot\underset{\sim}{u} \, e^h\phi\right) d\Omega = \langle r_h,\phi\rangle$$

$$\text{for all } \phi \in W \tag{4.5}$$

where $\langle r_h,\phi\rangle$ is the residual functional,

$$\langle r_h,\phi\rangle = -\int_\Omega \left(\rho^h_t\phi + \underset{\sim}{\nabla}\cdot\underset{\sim}{u} \, \rho^h\phi\right) d\Omega \tag{4.6}$$

If we replace ϕ by ϕ_h, $\langle r_h,\phi\rangle = 0$ by (4.3) and the evolution equation reduces to merely the orthogonality condition (4.3), which is automatically satisfied by error.

We obtain an approximate evolution equation for the error as follows: let E^h denote a fine-grid approximation of e^h; i.e.,

$$e^h(\underset{\sim}{x},t) \simeq E^h(\underset{\sim}{x},t) = \sum_N E^N(t) \, \psi_N(\underset{\sim}{x}) \tag{4.7}$$

where $\psi_N(\underset{\sim}{x})$ denotes a polynomial basis function defined on a subgrid of finer mesh size than that used to calculate ρ^h. Then, introduction of (4.7) into (4.5), and replacing ϕ by ψ_N gives

$$\sum_{M} \left(m_{NM} \dot{E}^{M} + k(\underset{\sim}{u})_{NM} E^{M} \right) - r_N(t) = 0$$

$$N = 1, 2, \ldots, \bar{N} \qquad (4.8)$$

where

$$\left.\begin{aligned} m_{NM} &= \int_{\Omega} \psi_N \psi_M \, d\Omega \\ k(\underset{\sim}{u})_{NM} &= \int_{\Omega} \underset{\sim}{\nabla} \cdot (\underset{\sim}{u} \psi_N) \psi_M \, d\Omega \\ r_N &= \langle r_h, \psi_N \rangle \end{aligned}\right\} \qquad (4.9)$$

Many possible ways for implementing (4.9) present themselves. These equations, for example, need not be global in the sense that an element-by--element or patch of elements in a fine mesh obtained through a mesh refinement may produce sufficient accuracy to allow for an adequate indication of the evolution of error. The local velocities $\underset{\sim}{u}$ and residual r_h can be interpolated using Q_1-approximations on a fine mesh level. Several of these alternatives are under study and are to be the subject of a forthcoming report.

4.2 Interpolation Errors. Let u be a smooth function defined over a regular domain Ω. The $W^{r,p}(\Omega)$-semi-norm of u is defined by

$$|u|_{W^{r,p}(\Omega)} = \left\{ \int_{\Omega} \sum_{\substack{i+j=r \\ i,j \geq 0}} \left| \frac{\partial^{i+j} u}{\partial x_1^i \partial x_2^j} \right|^p d\Omega \right\}^{1/p} \qquad (4.10)$$

where $1 \leq p \leq \infty$ and r is a non-negative integer.

The Sobolev norm of u is

$$\|u\|_{W^{r,p}(\Omega)u} = \left\{ \sum_{k=0}^{r} |u|^p_{W^{k,p}(\Omega)} \right\}^{1/p} \tag{4.11}$$

Let G be an arbitrary convex subdomain (a finite element) of Ω over which u is interpolated by a function $\tilde{u}_h$ which contains complete piecewise polynomials of degree k. Then, it can be shown (see Oden and Carey [17]) that the local interpolation error in the $W^{m,p}(G)$-semi-norm is

$$|u - \tilde{u}_h|_{W^{m,p}(G)} \leq C \frac{h^{k+1}}{\rho^m} \cdot h^{\frac{n}{p'} - \frac{n}{p}} |u|_{W^{k+1,p}(G)} \tag{4.12}$$

where

h = the diameter of the domain G

ρ = the diameter of the largest sphere that can be inscribed inside G

n = the dimension of the domain Ω

$p' = p/(p-1)$

C = a constant independent of h, ρ, and u.

If ρ is proportional to h and if it remains proportional in refinements of G defined by parametrically reducing h, we have

$$|E^h|_{m,p,G} \leq C\, h^{\frac{n}{p'} - \frac{n}{p} + k+1-m} |u|_{k+1,p} \tag{4.13}$$

with $|\cdot|_{m,p,G} = |\cdot|_{W^{m,p}(G)}$, etc. and $E^h = u - \tilde{u}_h$.

Such estimates can be used to devise crude adaptive schemes. Suppose that u on the right side of (4.13) is replaced by a finite element approximation u_h and that $|u_h|_{k+1,p} = |u|_{k+1,p} + O(h)$. Then, (4.13)

indicates that the local error in the $W^{m,p}(G)$ seminorm is proportional to the error indicator, $h^{n/p' - n/p + k+1 - m} |u|_{k+1,p}$. Some choices are:

i) $n = 2$, $m = 0$, $k = 1$, $p = p' = 2$

$$\|E^h\|_{L^2(G)} \leq C\, h^2 |u|_{2,2,G}$$

In this case, one must approximate the $W^{2,2}$-semi-norm of u over G; i.e., the L^2-norm of second partial derivatives of u.

ii) $n = 2$, $p = \infty$, $p' = 1$, $k = 0$, $m = 0$.

$$|E^h|_{L^1(G)} = Ch^2 |E^h_{\text{average}}|$$

$$\leq Ch^3 |u|_{1,\infty,G}$$

$$= Ch^3 \max_{\underset{\sim}{x} \in G} |\underset{\sim}{\nabla} \cdot \underset{\sim}{u}(x)|$$

Such estimates can give only rough indications of local errors in sufficiently fine meshes. However, they are usually easy to implement and our experience is that they can provide a very effective basis for mesh refinement strategies.

5. ADAPTIVE MESH STRATEGIES

Let us suppose that we can calculate an error indicator Θ_e for each finite element Ω_e in a given mesh at a time t. This indicator is, in general, a real number representing the local error in a suitable norm, and it is computed using one of the procedures described in the preceding section. The decision to adapt the numerical procedure (to refine the mesh or

to move nodal points) is based on whether or not local error indicators exceed preassigned tolerances. We shall describe two adaptive procedures in this section.

5.1 An h-Refinement/Unrefinement Method. Our h-procedure involves the following steps.

1) For a given domain Ω, such as that shown in Figure 1a, a coarse finite element mesh is constructed which contains only a number of elements sufficient to model basic geometrical features of the flow domain.

2) As our adaptive process will be designed to handle groups of four elements at a time, we generate a finer starting grid by a bisection process, indicated in Figure 1b, to obtain an initial set of element groups.

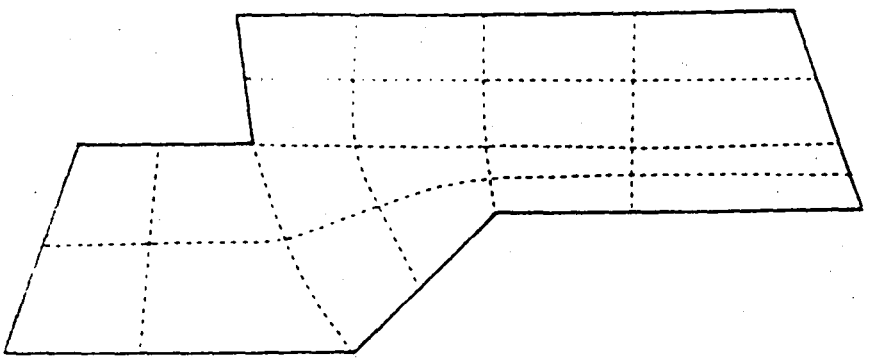
3) We initiate the numerical solution procedures on this initial coarse grid, and compute error indicators θ_e over all M elements in the grid. Let

$$\theta_{MAX} = \max_{1 \le e \le M} \theta_e$$

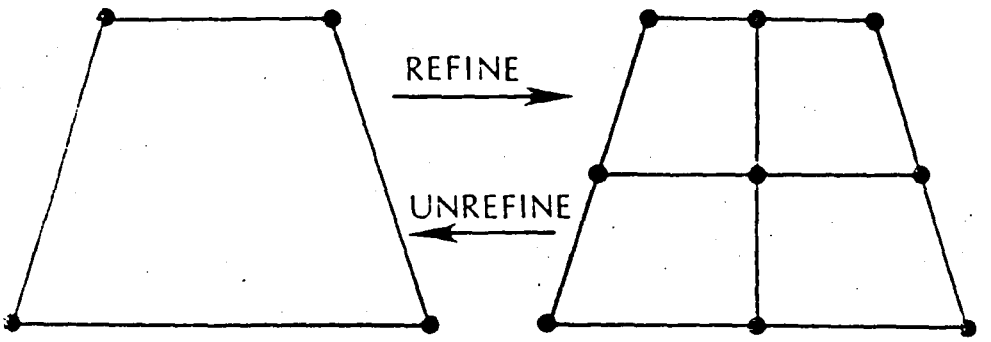
4) Next, we scan groups of a fixed number P of elements and compute

$$\theta^k_{GROUP} = \sum_{k=1}^{P} \theta_{e_k}$$

where e_k is the element number for group k. We take $P = 4$ in our current codes.



(a)



(b)

Figure 1. (a) A coarse initial mesh consisting of 4-element groups and (b) the refinement and unrefinement of a group of elements.

5) Error tolerances are defined by two real numbers, $0 < \alpha,\beta < 1$. If

$$\theta_e \geq \beta\, \theta_{MAX}$$

we <u>refine</u> element θ_e. This is done by bisecting θ_e into four new subelements. If

$$\theta^k_{GROUP} \leq \alpha\theta_{MAX}$$

we <u>unrefine</u> group k by replacing this group with a single new element with nodes coincident with the corner nodes of the group. This is always possible because each group is itself the result of an initial bisectioning.

This general process can be followed for any choice of an error indicator. Moreover, it can also be implemented at each time step in the numerical schemes discussed in Section 3.

5.2 <u>Data Structures</u>. An important consideration in all adaptive schemes is the data structure and associated algorithms needed to handle the changing number of elements, their node locations and numbers, and the element labels.

As noted in the preceding paragraphs, the algorithm is designed to process (refine or unrefine) in groups of four elements at each local refinement/unrefinement step. Consider, for example, the case of an initial mesh of 20 square elements shown in Figure 2. We assign to each element in this mesh an element number, NEL = 1,2,...,NELEM and to each <u>global</u> node a label NODE. The array, NODES(J,NEL) relates the local node number J(J = 1,2,3,4) of element NEL to the global node number NODES. In addition, the coordinates X_J, Y_J of each node are also provided relative to a fixed global coordinate system. We file these numbers in two arrays,

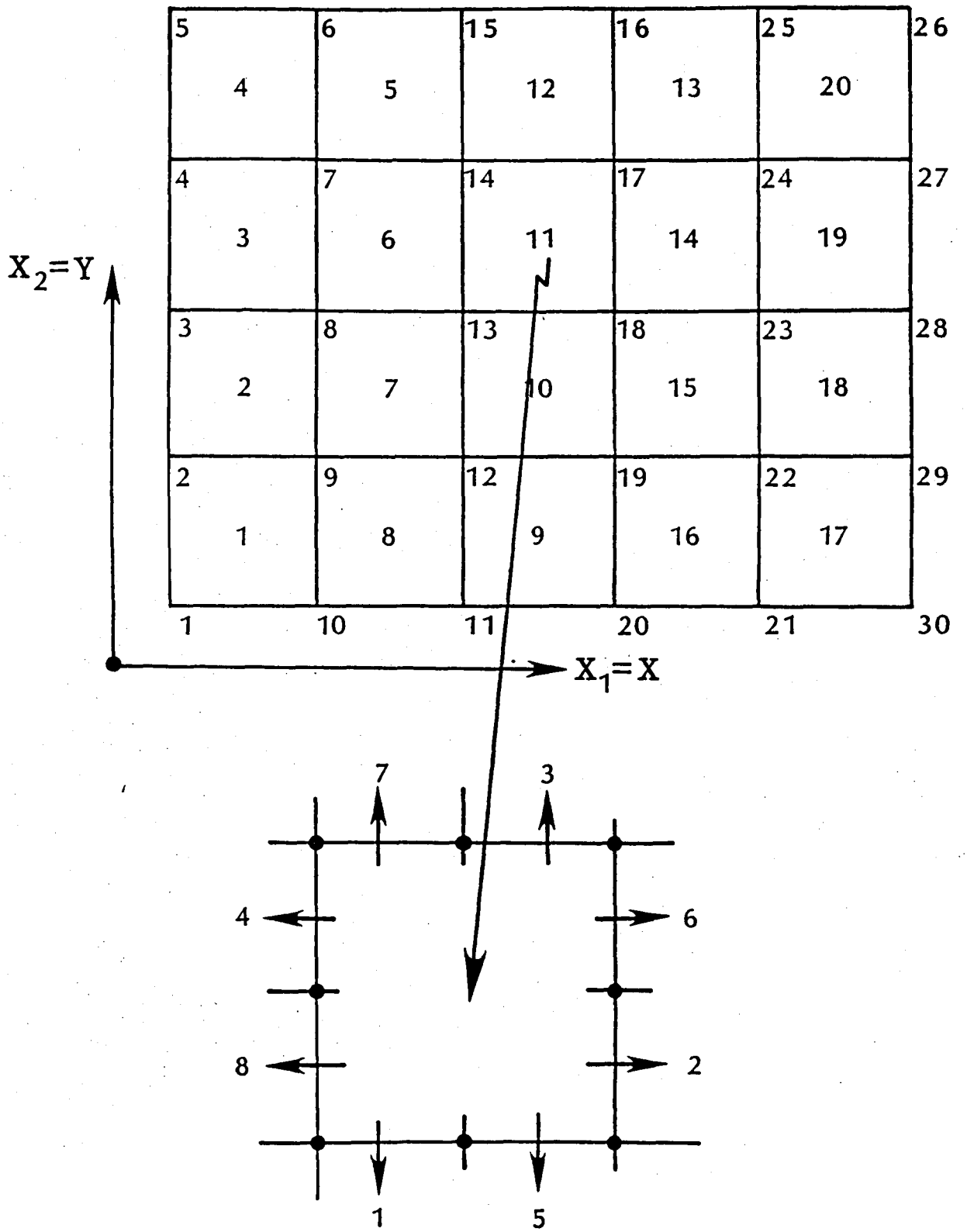


Figure 2. Mesh, node, and connectivity numbering in a model problem.

NODES(J,NEL) = the array of global node numbers assigned to node J of element NEL

XCO(JCO,NODE) = the array of JCO -- coordiantes of global node NODE(JCO = 1 or 2) .

Suppose that an error indicator is computed that signals that an element should be refined, say element 11, in the example. We must have some system for assigning appropriate labels to the new elements and nodes. Toward this end, we can establish a convention that defines the connectivity of the specified element with its neighbors in the mesh. This information is provided by a third connectivity array,

NELCON(NC,NEL) = the NC^{th} connection of element NEL NC = 1,2,...,8 ;

As seen in Figure 2, each side of an element may be connected to two other elements so that Dimension NELCON = (8,MAXEL), (with MAXEL an appropriately large number).

The entire refinement (or its inverse -- the unrefinement process) just described is accomplished by specifying a series of element levels. For example, the initial coarse mesh could be assigned level 0 . When an element is refined, its subelements belong to a higher level, level 1, and when these sub-elements are refined, elements of level 2 result, and so on. In this way, if the maximum level any element in the mesh can achieve is limited, then the maximum number of elements the mesh can contain is also limited. In general, no such limit need be set.

Thus, the bookkeeping of element and node numbers evolving in a refinement process is monitored by the arrays NODES(·,·) , XCO(·,·) , NELCON(·,·) , and an array LEVEL(NEL) which assigns a level number to

element NEL . Initially, the same level can be assigned to all elements, and this level is arbitrary parameter prescribed in advance by the user. Thus, provisions are now in hand for an arbitrary, dynamic renumbering of elements and nodes. If, for example, for the mesh in Figure 2, if element 11 is to be refined, we proceed through the following steps:

(1) Loop over the neighbors of element 11 (which is made possible with the NELCON array) and check the level of the neighboring elements relative to the level of element 11;

(2) If any neighboring element has a level lower than 11, then the element cannot be refined at this stage;

(3) If 11 can be refined (as is the case in Fig. 2), we generate new element numbers (thus changing NELEM and new node numbers for unconstrained nodes);

(4) Compute the connectivity matrix NELCON for the new elements;

(5) Adapt the connectivity matrices for the neighboring elements (since the refinement of 11 has now changed this connectivity); and

(6) Interpolate the solution between the unconstrained nodes.

It is clear that some strategy is needed to test if a designated element is appropriately connected for a refinement to take place.

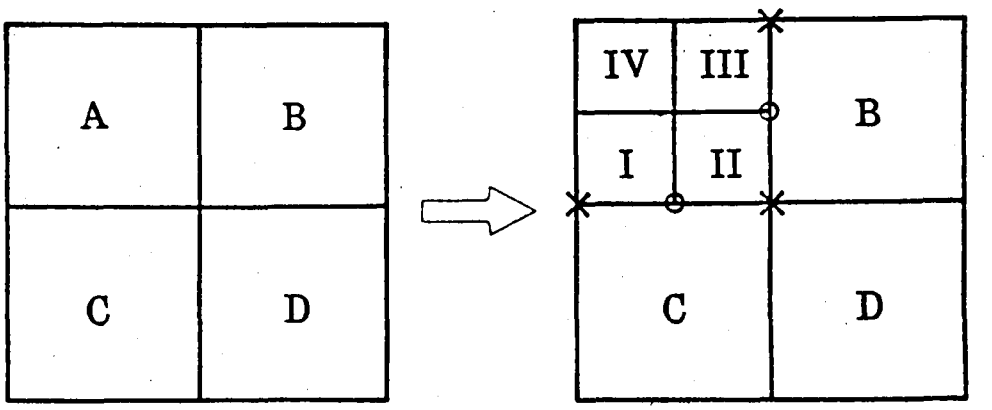
Consider, for example, the uniform grid of four elements shown in Figure 3a and suppose that the error estimators dictate that element A is to be refined. Thus, A is divided into four elements, I, II, III, IV, as shown, and the solution values at the junction nodes, shown circled in the figure, are constrained to coincide with the averaged values between those marked X . Note that the connectivities change in this process, e.g., the connectivities 4 and 8 of element B are different.

Next, assume that an additional refinement is required, and that we must next refine element III. We impose the restriction that each element side can have no more than two elements connected to it. Thus, before III can be refined, element B must first be refined, as indicated in Figure 3b. The constrained node B1 in Figure 3a now becomes active, while node C1 remains a constrained node. With B bisected, we proceed to refine III into sub-elements $\alpha,\beta,\gamma,\delta$, and new constrained nodes, again circled in Figure 3c, are produced. In this case, only element B had to be refined first in order to refine III, but, in general, the number of elements that must be refined in order to refine a particular element cannot be specified. The following subroutine determines the necessary refinements prerequisite to refining an element NEL1:

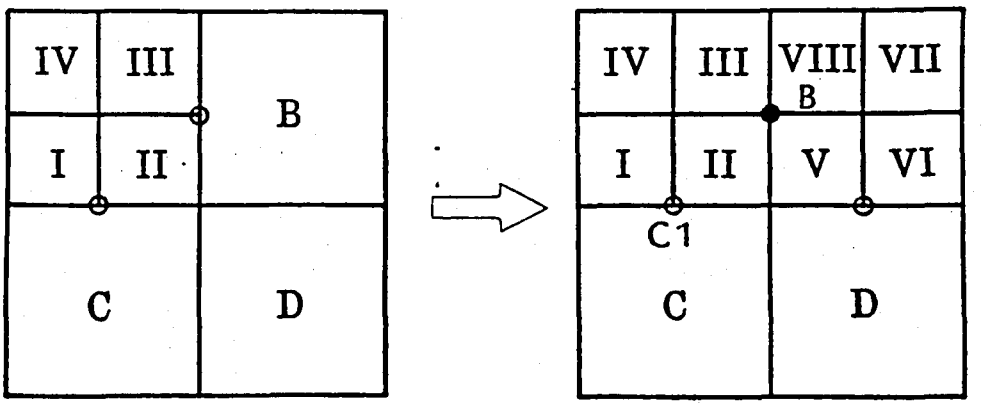
SUBROUTINE DIVIDE(NEL1,NEL2)

NEL1 = the input element that needs to be refined

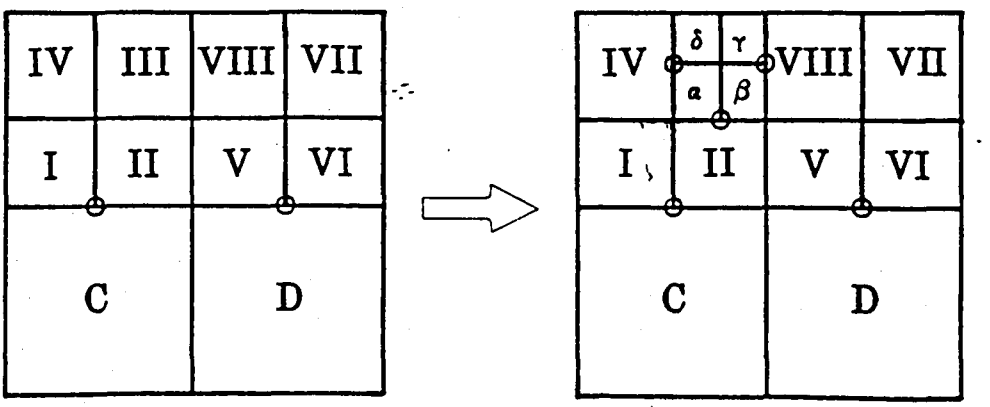
NEL2 = output element = { NEL1 if NEL1 has been divided
NELD = element that needs to be divided prior to NEL1



(a)



(b)



(c)

Figure 3. Sequence of refinements of a uniform mesh.

Then, symbolically, we have the algorithm (for the example in Fig. 3),

```
Repeat
   NEL1 = III
   CALL DIVIDE (NEL1,NEL2)
   WHILE (NEL2.NE.NEL1)
      NEL1 = NEL2
      CALL DIVIDE (NEL1,NEL2)
   END WHILE
UNTIL (NEL2.EQ.III)
```

5.3 Moving Mesh (Node Redistribution) Methods. Another family of adaptive schemes we have considered is a node-redistribution scheme which progressively moves a fixed number of nodes as to reduce local error. One basis for such schemes is to equidistribute error at each time step.

For example, let θ_e be an error indicator for element Ω_e in a mesh containing a fixed number M of elements in a two-dimensional mesh. Let $h = h(x_1,x_2)$ be a mesh function such that

$$h(x_1,x_2) = h_e = \text{dia}(\Omega_e) \quad \text{for} \quad (x_1,x_2) \in \Omega_e$$

and note that, approximately,

$$M = \int_\Omega \frac{d\Omega}{h^2} \tag{5.1}$$

with $d\Omega = dx_1 dx_2$ (this being exact for domains which are unions of square elements). Let $\theta = \theta(x_1,x_2)$ be mesh function which gives the local error indicator when evaluated at a point ($\theta = \theta_e$ for $\underset{\sim}{x} \in \Omega_e$). We wish to minimize the total error indicator functional,

$$J(\theta) = \sum_{e=1}^{M} \int_\Omega \theta_e^2 \, d\Omega \tag{5.2}$$

subject to the constraint (5.1). Using Lagrange multipliers, this leads to the optimality condition,

$$\delta\left(J + \lambda\left(\int_{\Omega} h^{-2}\, d\Omega - M\right)\right) = 0 ,$$

or

$$\sum_e 2 \int_{\Omega_e} \left(\theta_e \frac{\partial \theta}{\partial h} - \lambda h^{-3}\right) \delta h d\Omega = 0$$

or

$$h_e^3\, \theta_e \frac{\partial \theta_e}{\partial h} - \lambda = 0$$

Suppose that $\text{meas}(\Omega_e) = \sigma_o h_e^2$ and that θ_e is of the form $\theta_e = h_e^{\sigma} f(u)$. Then, integrating this last result over a typical element gives

$$\int_{\Omega_e} \sigma h_e^3\, \theta_e\, h_e^{\sigma-1} f(u) d\Omega = \lambda \sigma_o h_e^2$$

Hence, the optimal mesh size distribution results when

$$\int_{\Omega_e} \theta_e^2\, d\Omega = \lambda \sigma_o / \sigma = \text{CONST.} \qquad (5.3)$$

In other words, to obtain the optimal mesh, we must <u>equidistribute</u> the indicators $\int \theta_e^2$.

To use this result to redistribute nodes, we proceed as follows (cf. Diaz et al. [12]):

1) Generate an initial (generally regular) mesh with a fixed number M of elements and compute a trial solution on this mesh at one time step;

2) Compute the corresponding error indicators θ_e ;

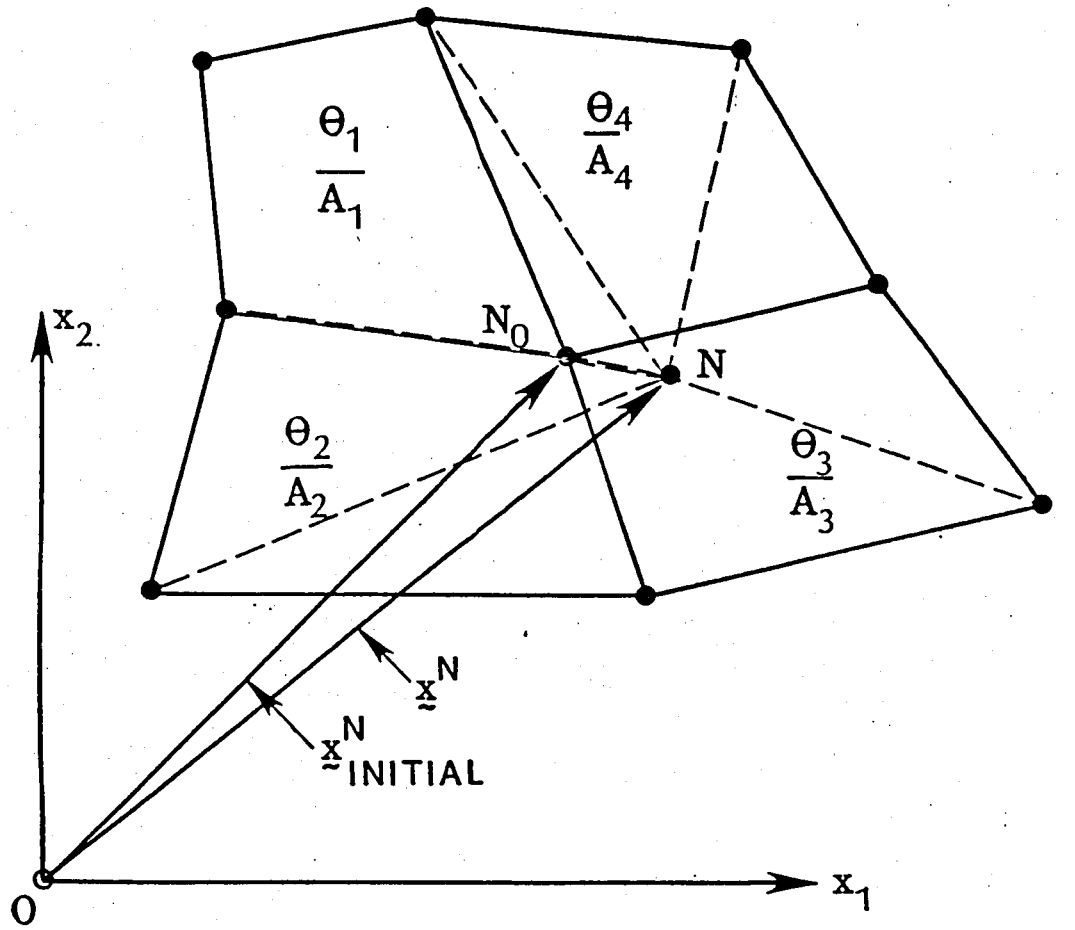


Figure 4. Calculation of area center-of-error $\underset{\sim}{x}^N$ to equidistribute element error indicators in a cluster of four elements.

3) For a group k of P elements (with P always 4 in this work), let A_{e_i} denote the area of element i in the group. The area-weighted indicators for group k are the P-numbers,

$$\theta_{ei}/A_{ei}$$

4) Let y_{e_i} denote a vector from the origin of a global coordinate system to the centroid of element e_i of group k. Then the <u>center of error</u> of group k is defined as the vector

$$\underset{\sim}{x}^k = \frac{\sum_{i=1}^{4} y_{e_i} \left(\frac{\theta_{e_i}}{A_{e_i}}\right)}{\sum_{i=1}^{4} \left(\frac{\theta_{e_i}}{A_{e_i}}\right)} \tag{5.4}$$

5) Relocate the node at the center of group k to lie at the vertex of $\underset{\sim}{x}^k$;

6) Continue this sequence of operations over each group h of four elements until the new location of each node does not change more than a preassigned tolerance.

This process should approximately equidistribute the element error indicators.

6. NUMERICAL EXAMPLES

In this section, we present the results of several numerical experiments on representative test problems. Six examples are presented,

the first five involving steady-state solutions and the last a transient problem. In the steady-state examples, one of the following two strategies is used:

(A) A.1. The numerical solution is computed on a fixed mesh and is advanced in time until a steady state is reached.

A.2. After convergence to a steady state, error indicators Θ_e are computed over each element. In the calculations discussed below, we employ the interpolation estimates and use

$$\Theta_e = A_e |\rho^h|^2_{2,2,\Omega_e} \simeq A_e \oint_{\partial\Omega_e} \|\partial\rho^h/\partial n\| dS \qquad (6.1)$$

where A_e is the area of the element.

A.3. The mesh is refined/unrefined using the criteria and algorithms discussed in the preceding section.

(B) B.1. Same as step A.1.

B.2. After convergence to a steady state, error indicators Θ_e are computed according to

$$\Theta_e = A_e \int_{\Omega_e} \nabla\rho^h \cdot \nabla\rho^h \, d\Omega \qquad (6.2)$$

B.3. In applying the node redistribution (moving mesh) algorithm, a modified error indicator $\tilde{\Theta}_e$ is employed which is designed to be always greater than unity even when $\Theta_e \simeq 0$. In particular, we use

$$\tilde{\Theta}_e = 1 + \frac{\alpha\Theta_e}{\beta + \gamma\Theta_e}$$

In our examples, $\alpha = 81$, $\beta = 1$, and $\gamma = 8$.

B.4. Nodes are redistributed a total of K times using the procedure described in Section 5.3. In the examples, we take only two iterations (K = 2).

We proceed to the examples.

6.1 Shock Reflection Problem. We begin with a problem for which an exact solution is known and which has been used as a benchmark problem by others.

The problem involves the steady flow of a perfect gas in a rectangular duct in which density, velocity, and energy are prescribed in each of four triangular wedges in such a way that the appropriate jump conditions (the Rankine-Hugoniot conditions) are exactly satisfied. Thus, a problem of shock reflection for which an exact solution is known is obtained. Dimensions and data are given in Figure 5. In this and all the other problems, the solution is considered to have converged to steady state when the magnitude of the L^2-norm of the density is reduced by three orders of magnitude.

The time step is monitored by the formula

$$\Delta t = \min_e \left\{ \frac{0.50\sqrt{A_e}}{|\underset{\sim}{u}| + C} \right\}$$

Here, $C^2 = \frac{\gamma P}{\rho}$ and $|\underset{\sim}{u}|^2 = u_1^2 + u_2^2$, $\gamma = 1.40$. The constants multiplying the artificial viscous terms were selected locally as:

$$\tau_x = A_e \left| \overline{\frac{\partial u^n}{\partial x}} \right|_e , \qquad \tau_y = A_e \left| \overline{\frac{\partial v^n}{\partial y}} \right|_e$$

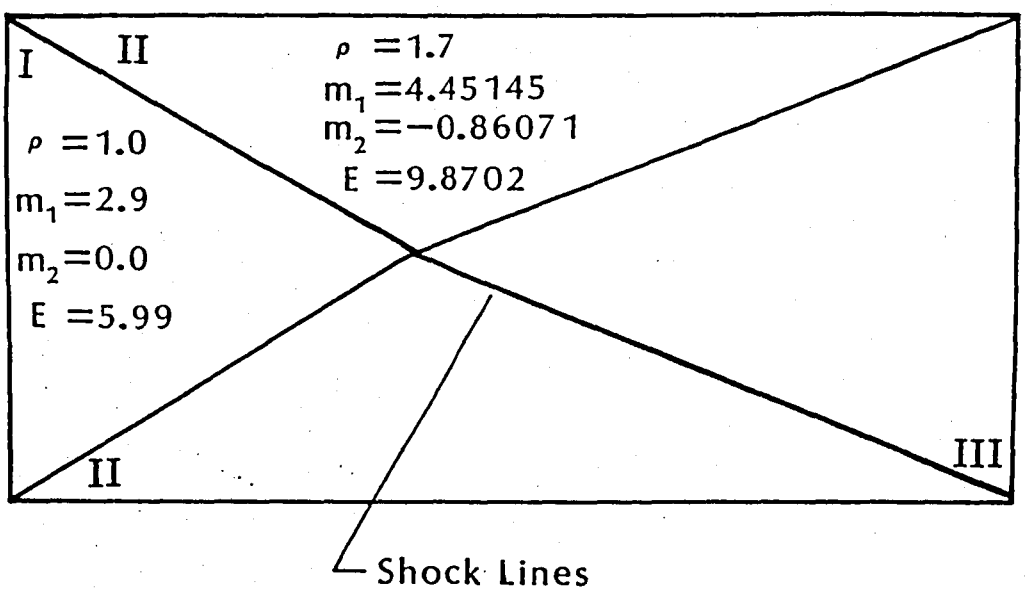


Figure 5. A shock reflection problem. Inflow values of the conservation variables are prescribed as indicated in regions I and II, and outflow values are computed in III to satisfy the conservation laws.

where the bar denotes average element values. A Lapidus constant of 1.0 was used in all calculations.

The results of a uniform coarse initial mesh approximation are shown in Figure 6. The computed density contours are also shown in this figure. Note that only a rough indication of the location of the shock is possible with this mesh.

A much better resolution is given in Figure 7 where the adaptively refined mesh shown is computed with refinement parameters $\alpha = 0.10$, $\beta = 0.50$ (recall Section 5). Note that no "unrefinement" appears to have taken place with these parameter choices, but that the simple error estimation scheme is capable of detecting the general area of the shock line. The much improved density profiles are indicated in the figure.

Still better results are obtained with the same α and β but with two levels of refinement, as indicated in Figure 8. Note that in this case large elements appear in the mesh, indicating unrefinement as well as refinement. of the original mesh. The corresponding density surface is given in Figure 9 where quite sharp shock fronts are observed. Note some spurious oscillation is encountered near the outflow boundary, as should be expected from the deficiencies of the algorithm noted in Section 4.

The same problem was also analyzed using the node redistribution algorithm discussed in Section 5.3 with 20 node redistribution iterations. Results are shown in Figure 10. There, the original coarse initial mesh of Figure 6 is progressively distorted to conform to the reflected shock locations. Corresponding density contours are also given in the figure.

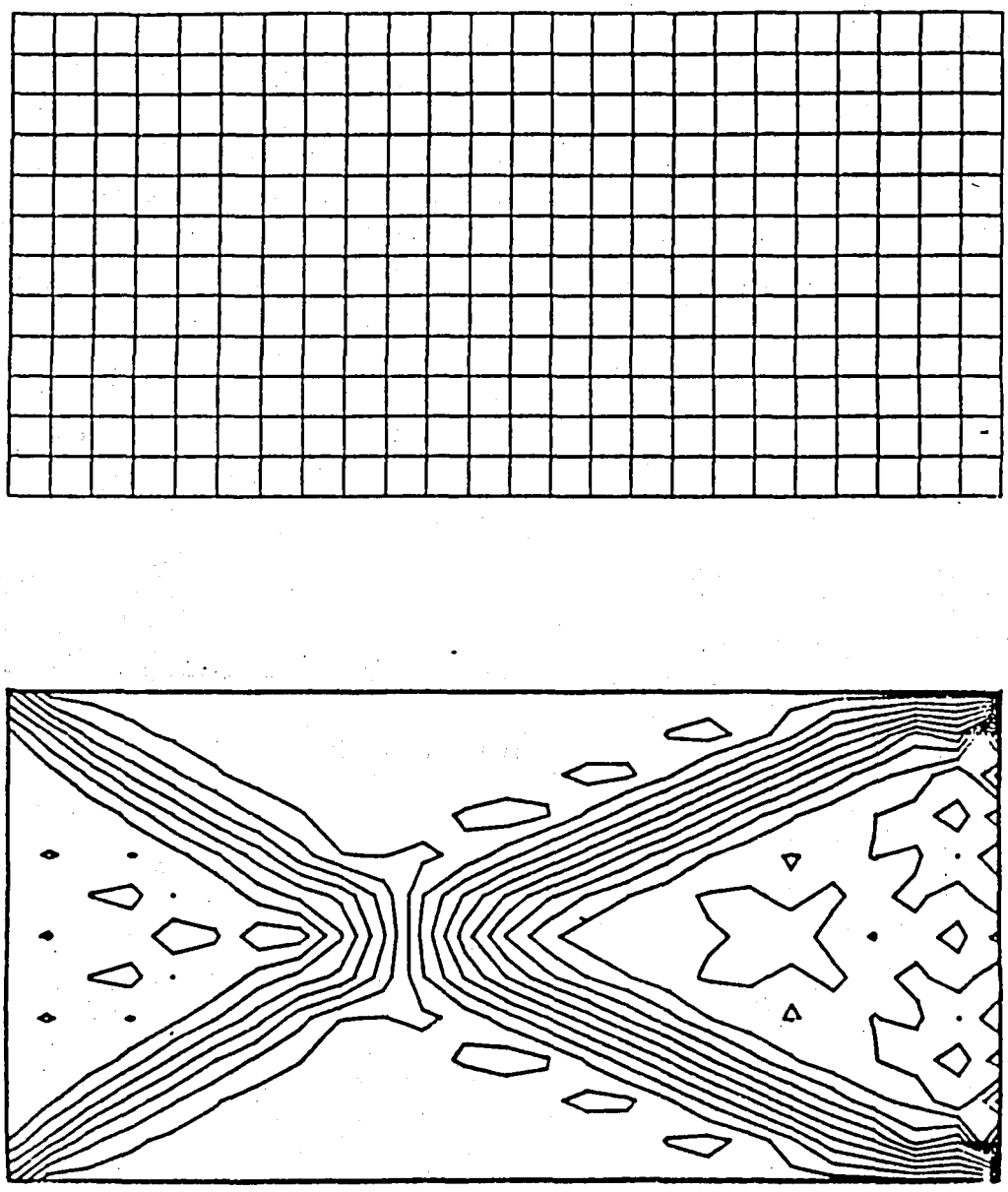


Figure 6. Reflecting shock problem.

Initial mesh and density contours.

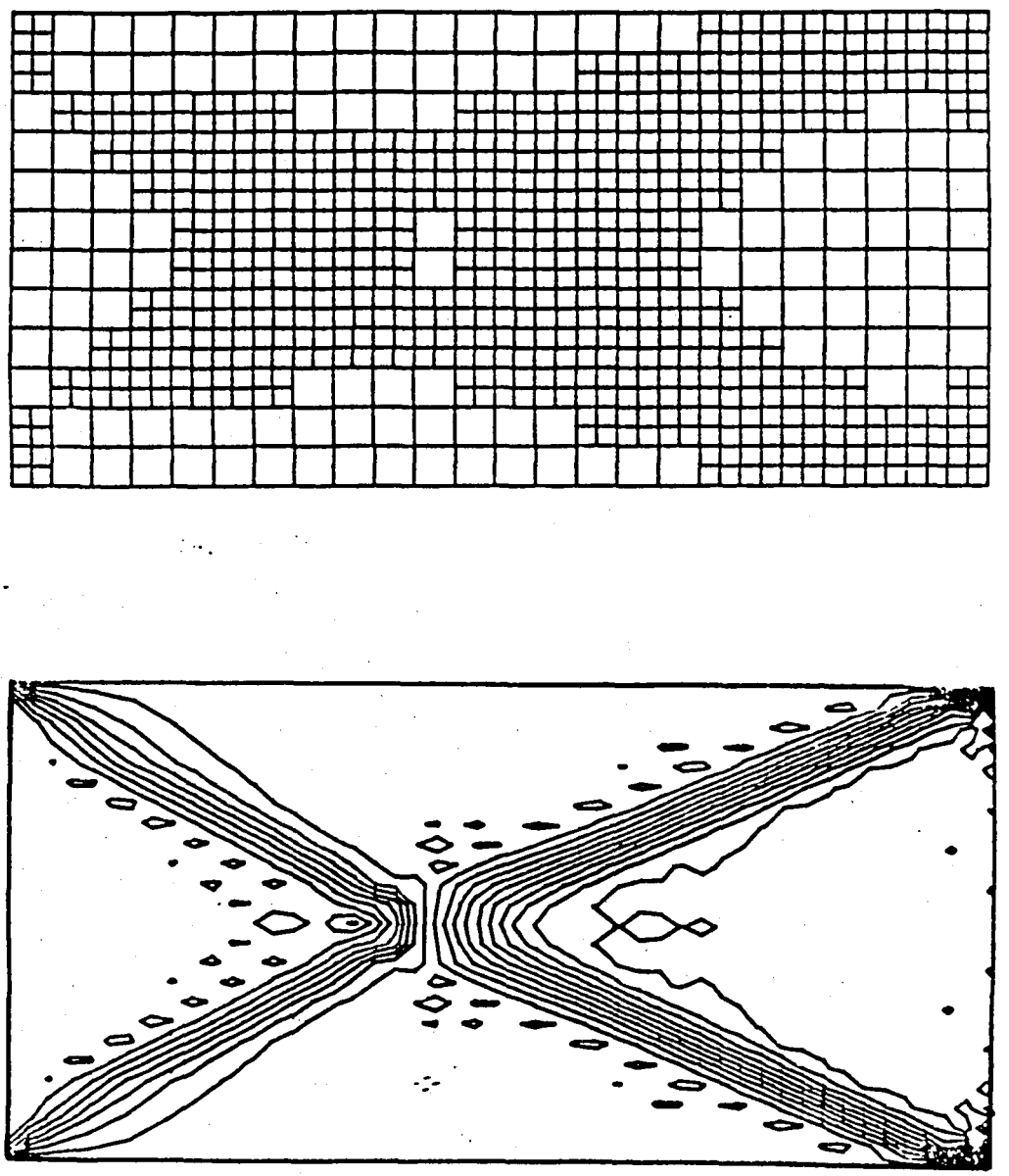


Figure 7. Reflecting shock problem. Mesh and density contours obtained with one level of refinement (α = 0.10, β = 0.50).

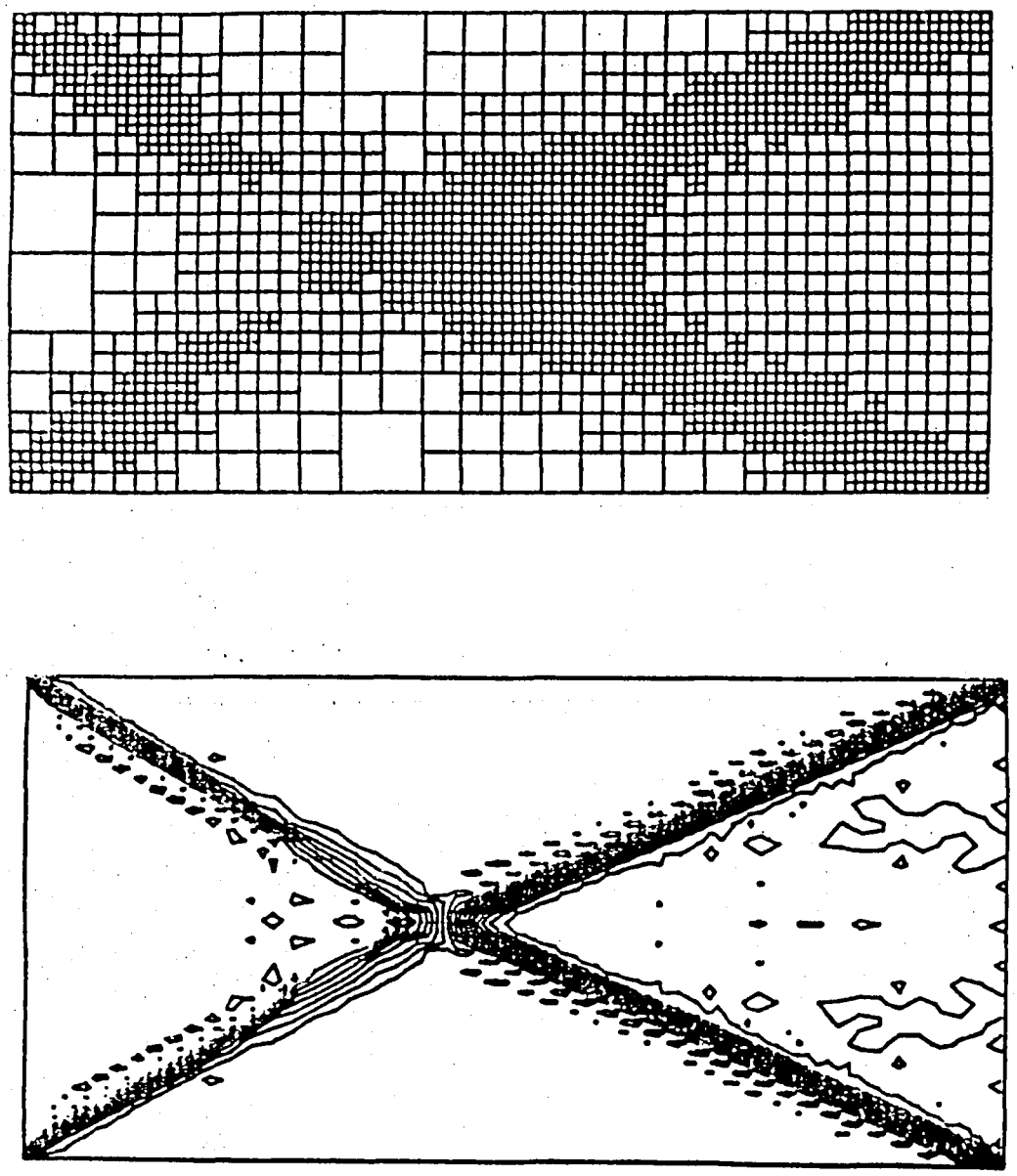


Figure 8. Reflecting shock problem. Mesh and density contours obtained with two levels of refinement (α = 0.10, β = 0.50).

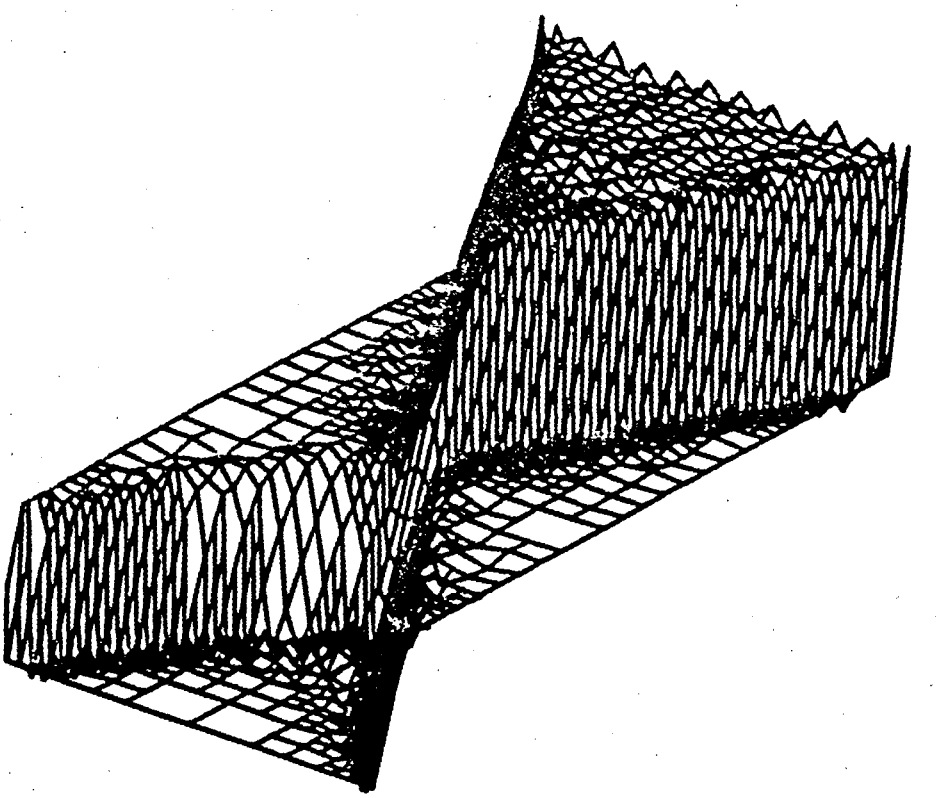


Figure 9. Reflecting shock problem. 3-D view of the converged density function obtained with two levels of refinement.

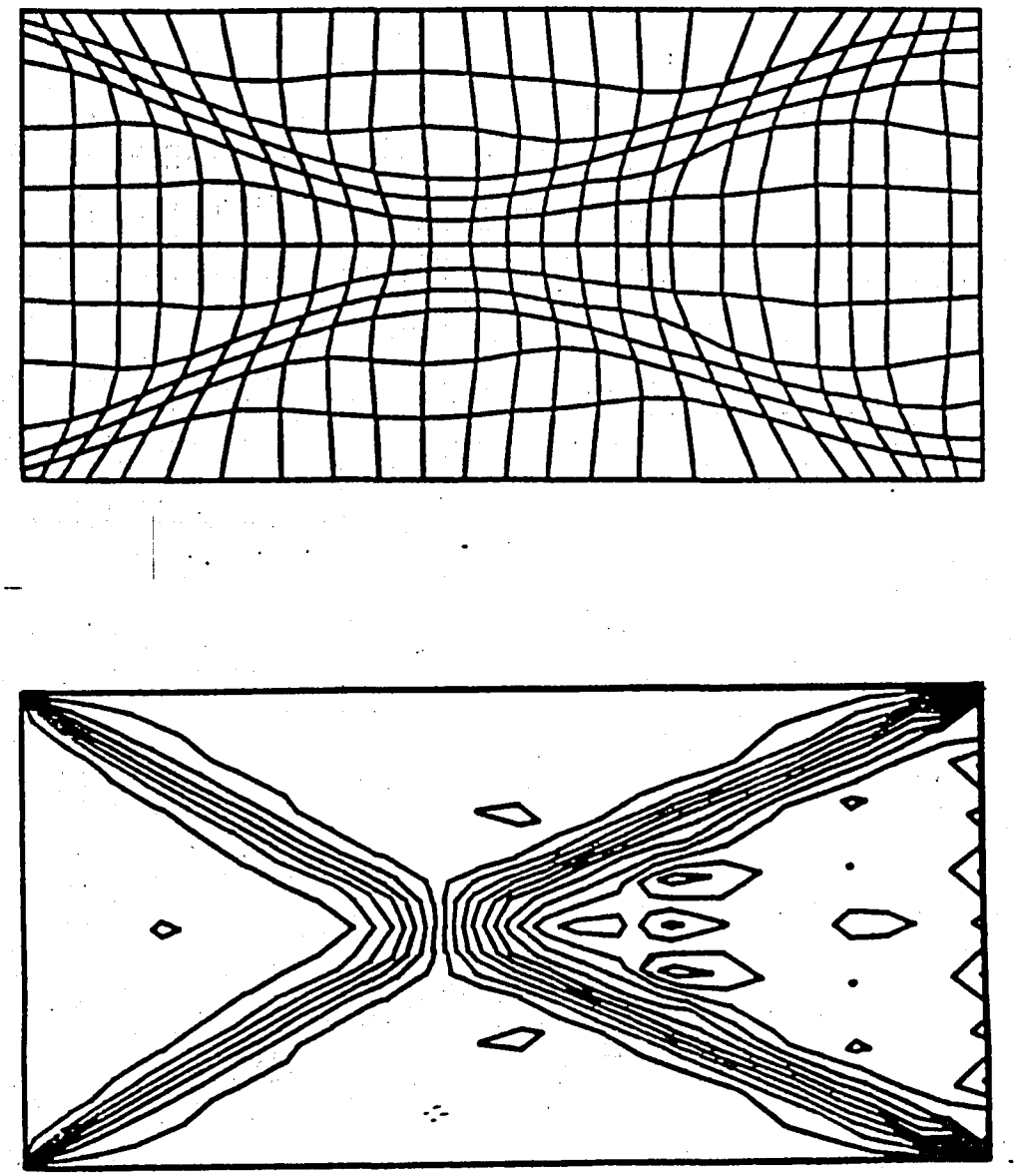


Figure 10. Reflecting shock problem. Mesh and density contours obtained after 10 applications of the mesh redistribution algorithm.

6.2 NACA 0012 Airfoil in Supersonic Wind Tunnel. In this example, the supersonic flow through a narrow wind tunnel containing a NACA 0012 airfoil is studied. The inflow Mach number was set at $M_\infty = 2$, with $\gamma = 1.40$ and symmetry is exploited to reduce the computational effort.

The initial coarse mesh and density computed contours are given in Figure 11. Note that the critical features of the solution -- the reflected shock and contact discontinuity -- are lost with this coarse mesh. A refined/unrefined mesh, obtained with parameters $\alpha = 0.10$, $\beta = 0.10$ is shown in Figure 12 together with a greatly improved density approximation. In these and subsequent calculations, a CFL number of 0.5 and a Lapidus constant of 1.0 were employed. Results of a node-redistribution scheme for the coarse mesh are shown in Figure 13. In these results, ten iterations of the node redistribution algorithm were used.

6.3 Supersonic Flow in a Wind Tunnel with a Step. The steady-state solution of the problem of a wind tunnel with a step introduced into the flow is next considered. The inflow Mach number was selected $M_\infty = 3.0$ and $\gamma = 1.40$. The initial coarse mesh is shown in Figure 14 with the corresponding density profiles, and results of the adaptive refinement/unrefinement scheme with $\alpha = 0.15$ and $\beta = 0.20$ are shown in Figure 15. The mesh refinement algorithm was also used, with the mesh and density profiles obtained after 10 iterations shown in Figure 16. We see that the adaptive scheme captures well the features of the flow including the contact discontinuity at the top near the point of reflection of the bow shock. However, some oscillations are present downstream, and they are believed to be due to the non-monotonicity of the solution algorithm. The

results presented for the refinement-unrefinement procedure have been constrained by a maximum number of 2000 nodes or 2000 elements that can be allowed. In the refined mesh shown, this constraint has been achieved.

6.4 Supersonic Flow Over a 20° Ramp. We next consider the steady supersonic flow through a conduit with a 20-degree ramp. The gas (with $\gamma = 1.4$) enters as a uniform $M = 3.0$ flow through the left side of the ramp and a shock develops at the ramp root. A coarse initial mesh and the computed density contours are illustrated in Figure 17. For this problem, a reasonably good indication of the orientation of the shock is obtained.

Adaptive mesh results are shown in Figures 18 and 19 for choices of the parameters of $\alpha = 0.20$ and $\beta = 0.50$ with one and two levels of refinement, respectively. Notice that spurious oscillations at the outflow boundary above the ramp root, due to the hourglass oscillations described in Section 3, cause unnecessary refinements in this region. Similarly, in regions between the shock and the ramps, some unnecessary refinement results from oscillations in the numerical solution. Nevertheless, striking improvement in the quality of the solution is seen to result from the refinement procedure.

In this particular problem, the node redistribution algorithm works remarkably well. A computed distorted coarse mesh, obtained after ten applications of the node redistribution algorithms, is shown in Figure 20 with the resulting density contours.

6.5 Blunt Leading Edge of 8' HTT Panel Holder in Hypersonic Flow. The problem of the blunt leading edge of the 8' HTT panel holder in a supersonic flow field with freestream Mach number $M_\infty = 6.57$, $\gamma = 1.38$

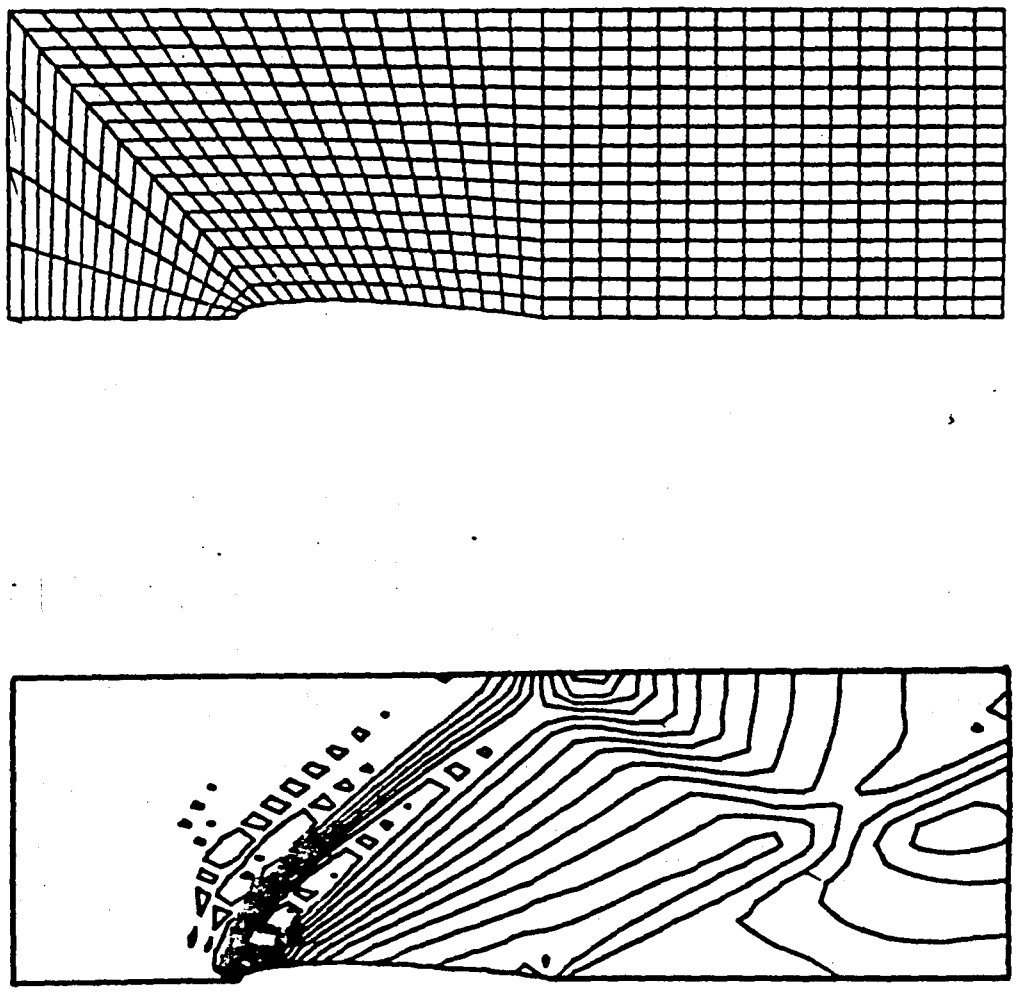


Figure 11. NACA 0012 airfoil in supersonic wind tunnel. Initial mesh and density contours.

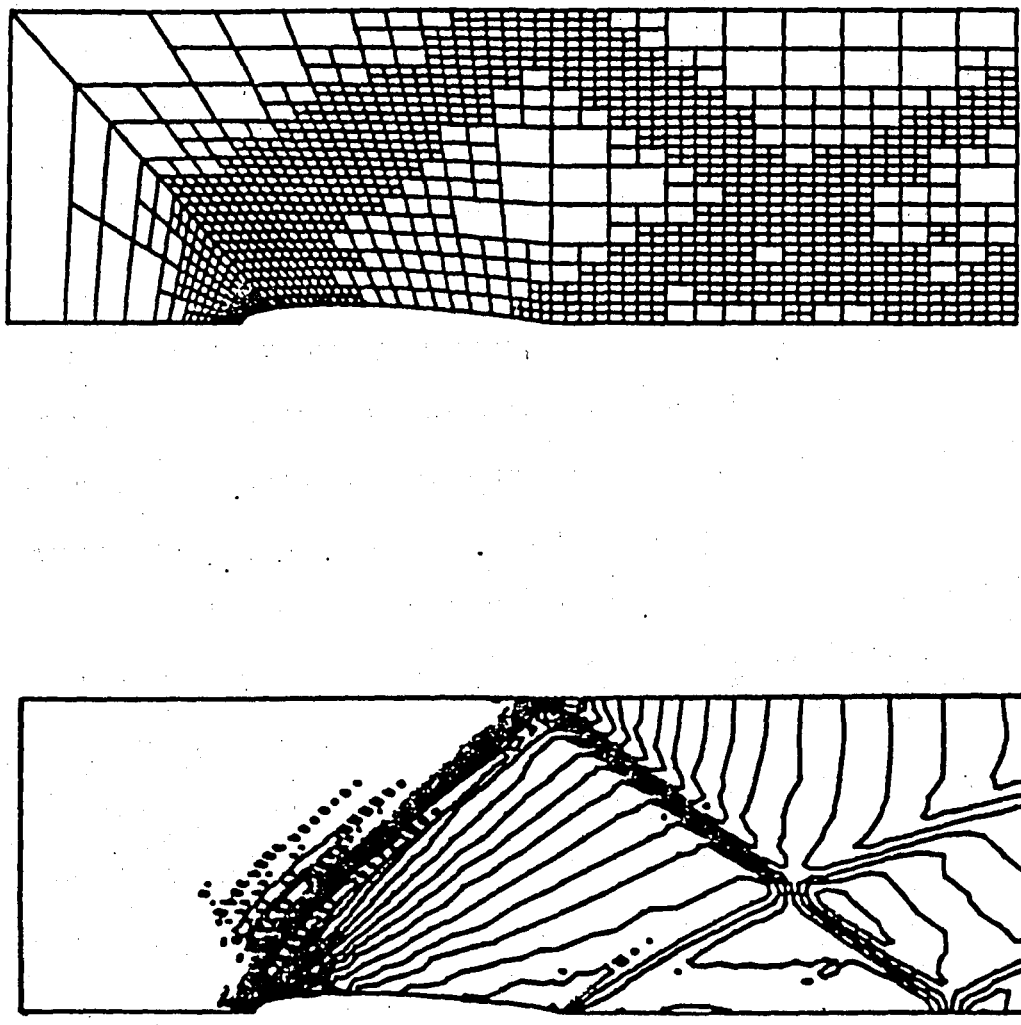


Figure 12. NACA 0012 airfoil in supersonic wind tunnel. Mesh and density contours obtained with one level of refinement (α = 0.10, β = 0.10)

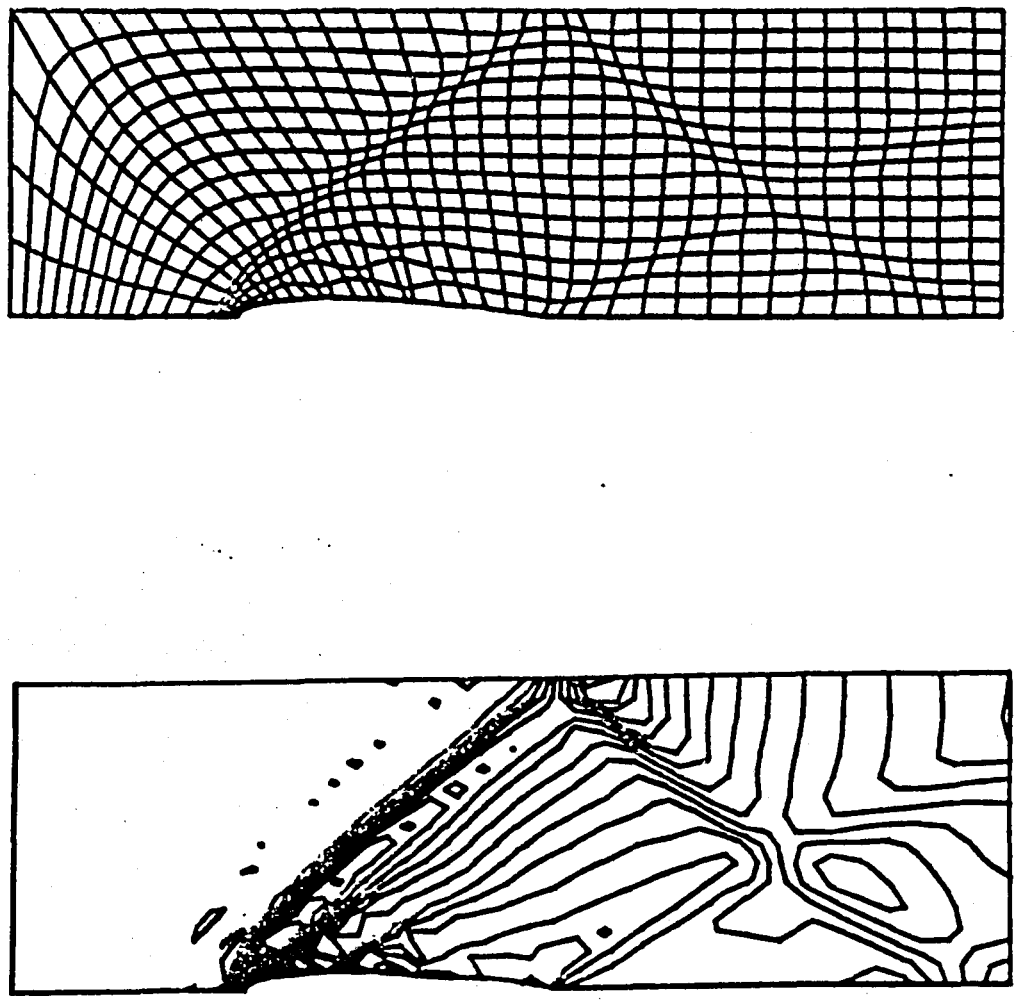


Figure 13. NACA 0012 airfoil in supersonic wind tunnel. Mesh and density contours obtained after 10 applications of the mesh redistribution algorithm.

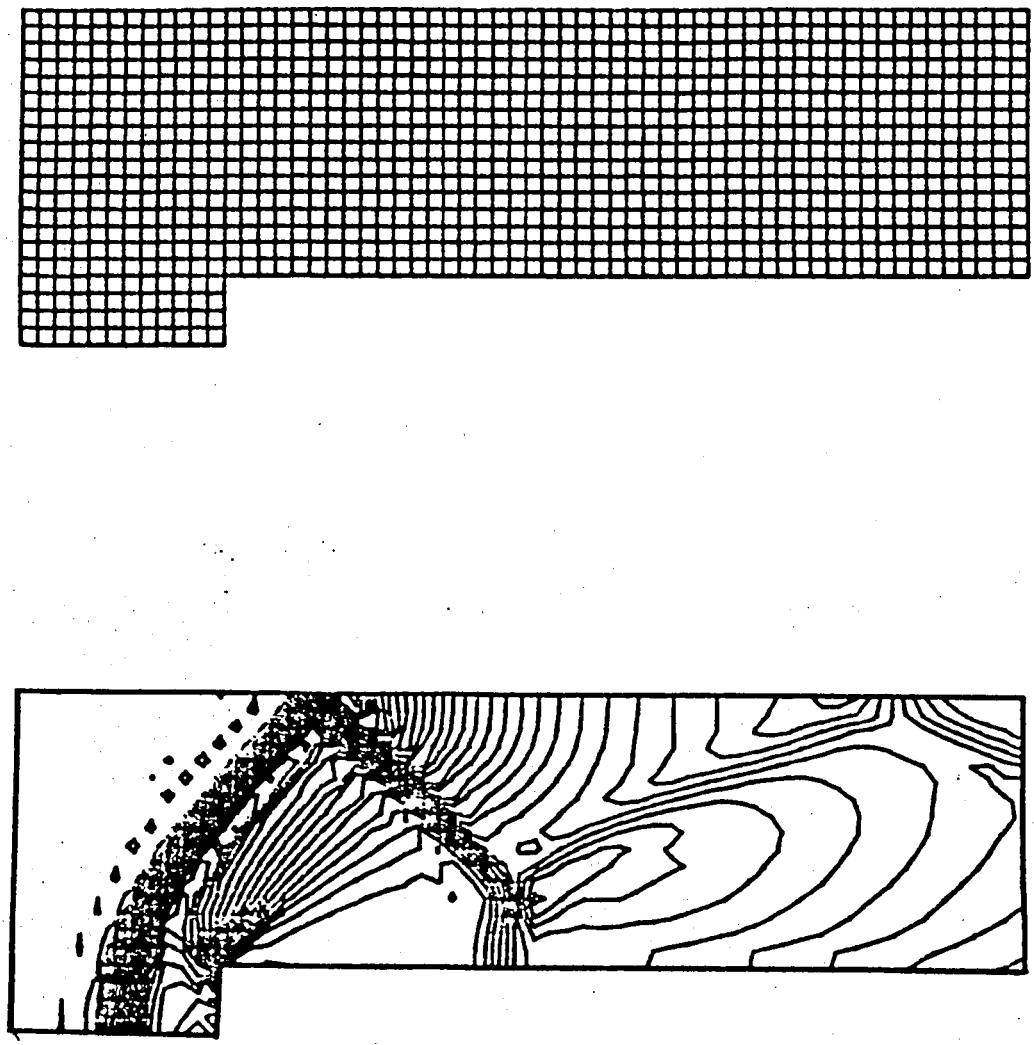


Figure 14. Supersonic flow in a wind tunnel with a step. Initial mesh and density contours.

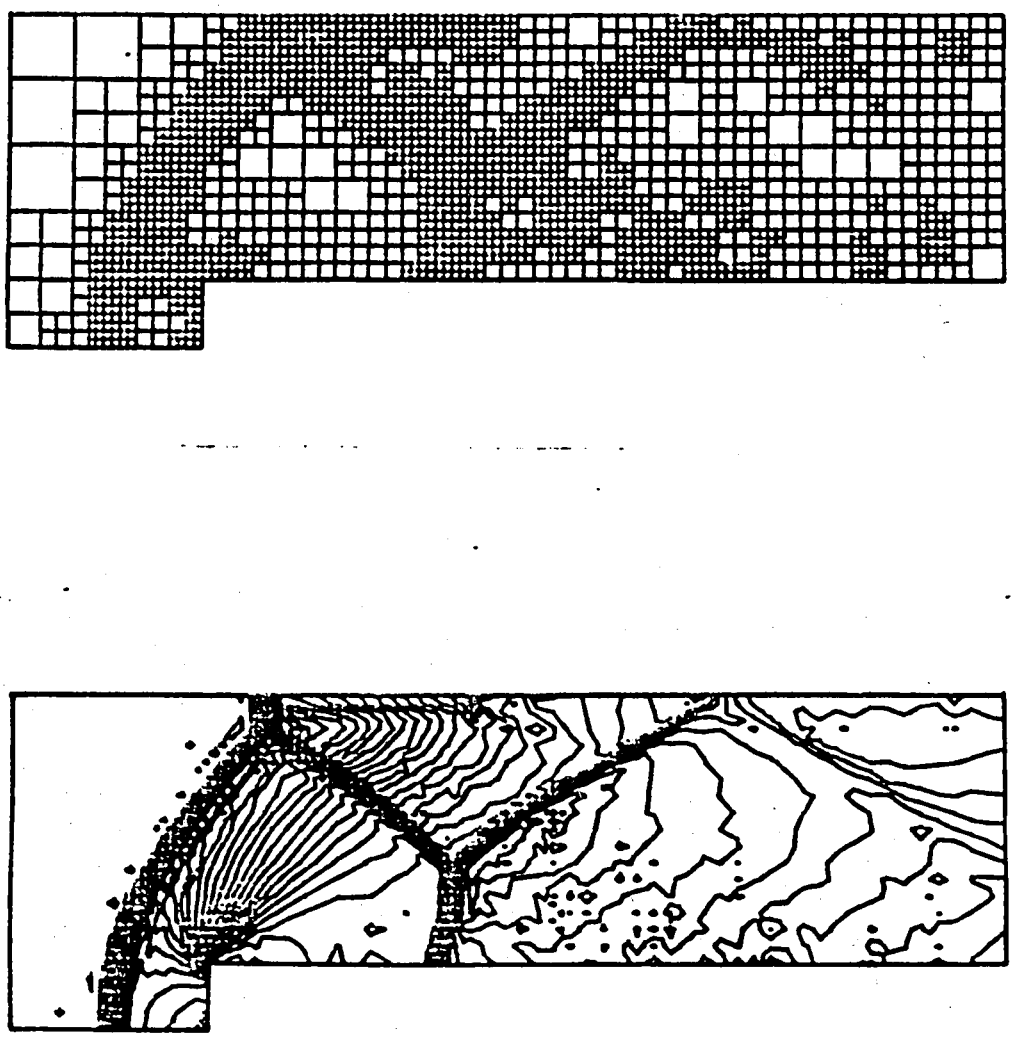


Figure 15. Supersonic flow in a wind tunnel with step. Mesh and density contours obtained with one level of refinement ($\alpha = 0.15$, $\beta = 0.20$).

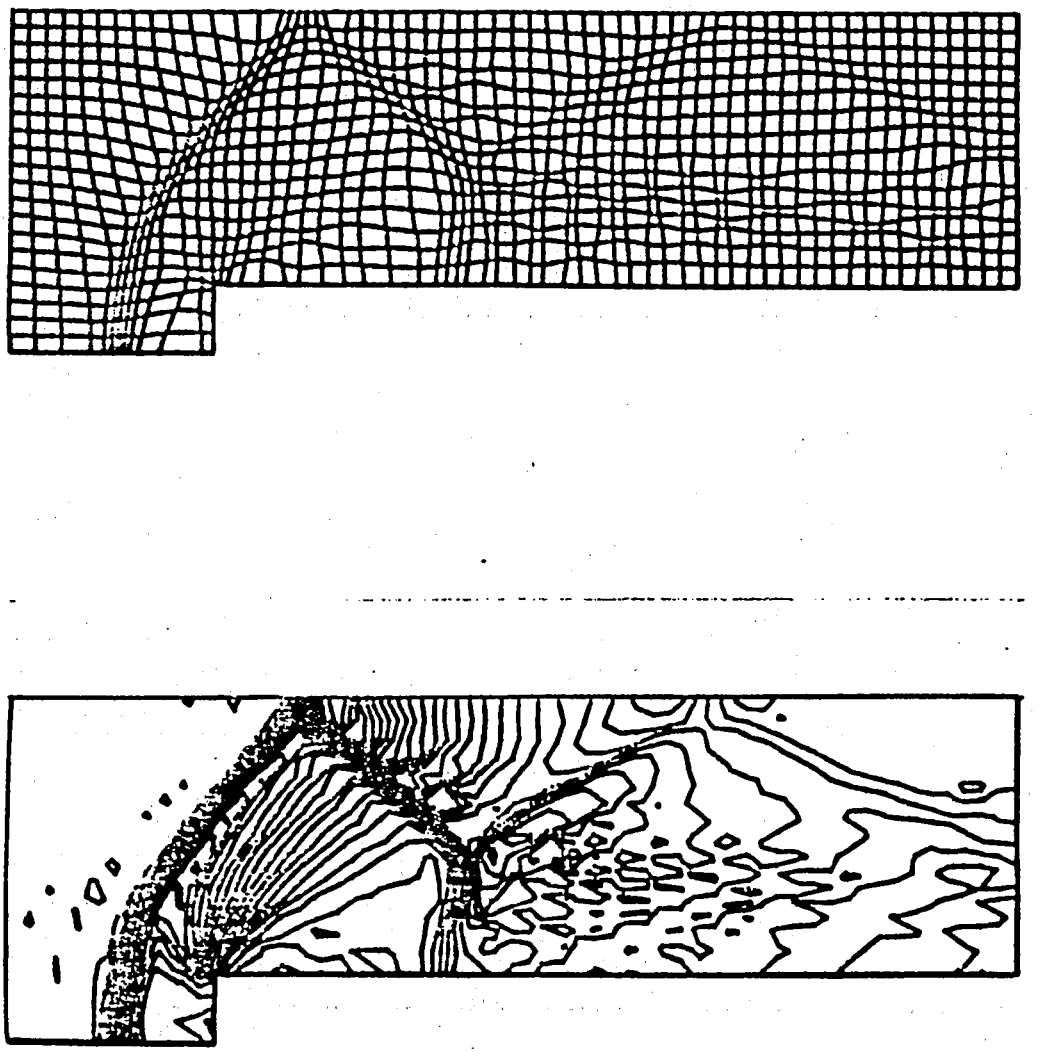


Figure 16. Supersonic flow in a wind tunnel with a step. Mesh and density contours obtained after 10 applications of the mesh redistribution algorithm.

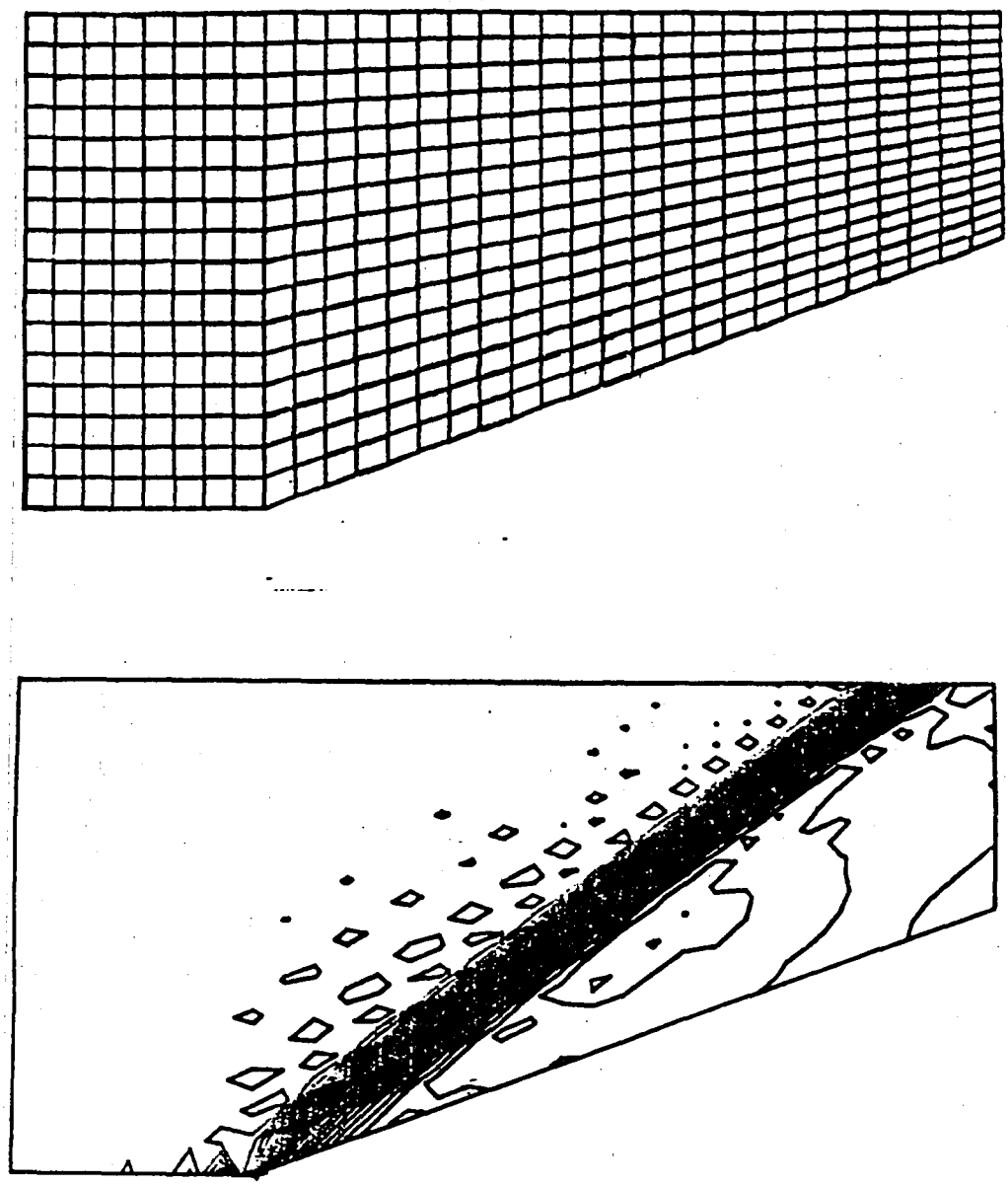


Figure 17. Supersonic flow over a 20° ramp.

Initial mesh and desnity contours.

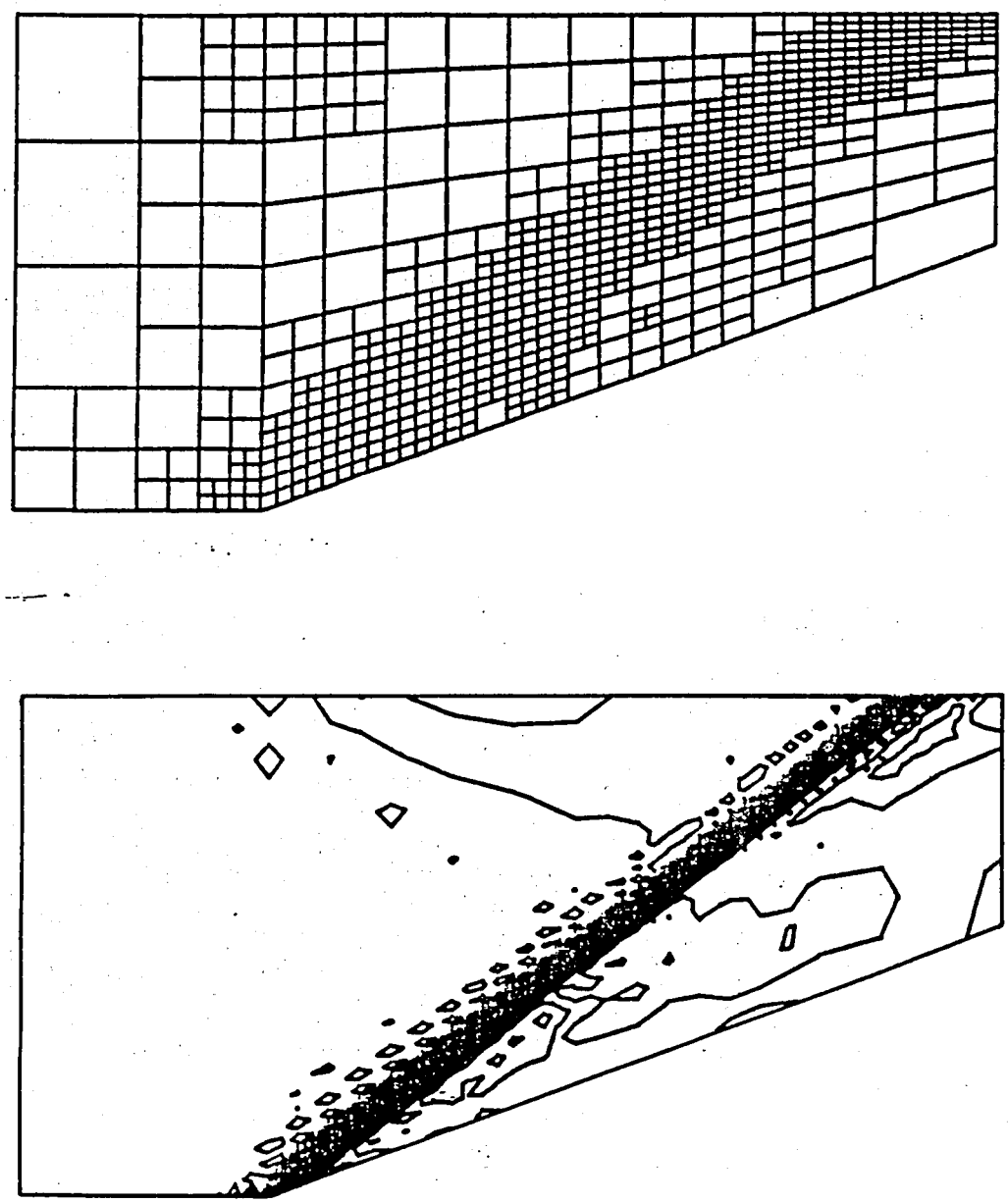


Figure 18. Supersonic flow over a 20° ramp. Mesh and density contours obtained with one level of refinement ($\alpha = 0.20$, $\beta = 0.50$).

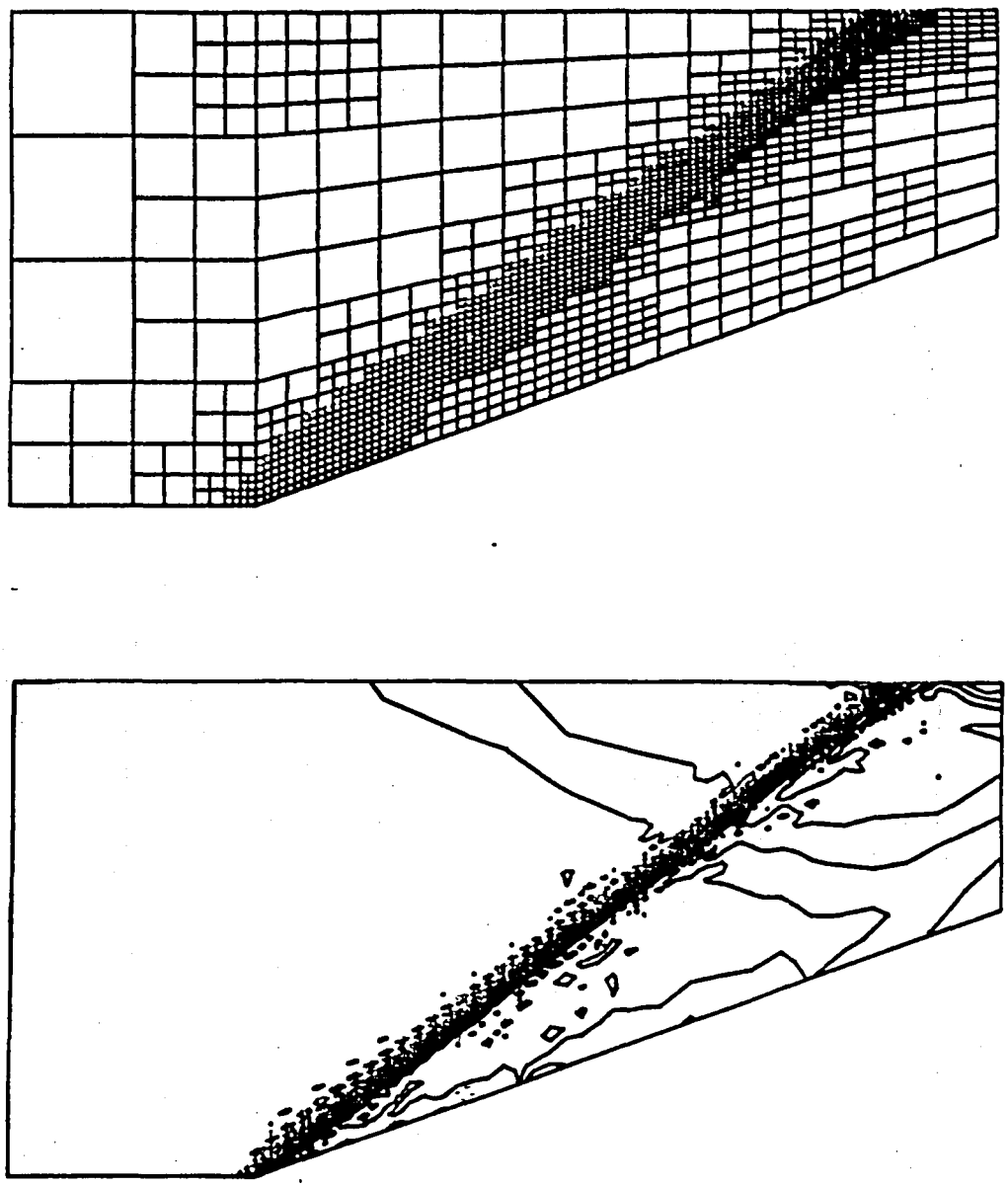


Figure 19. Supersonic flow over a 20° ramp. Mesh and density contours obtained with two levels of refinement ($\alpha = 0.20$, $\beta = 0.50$).

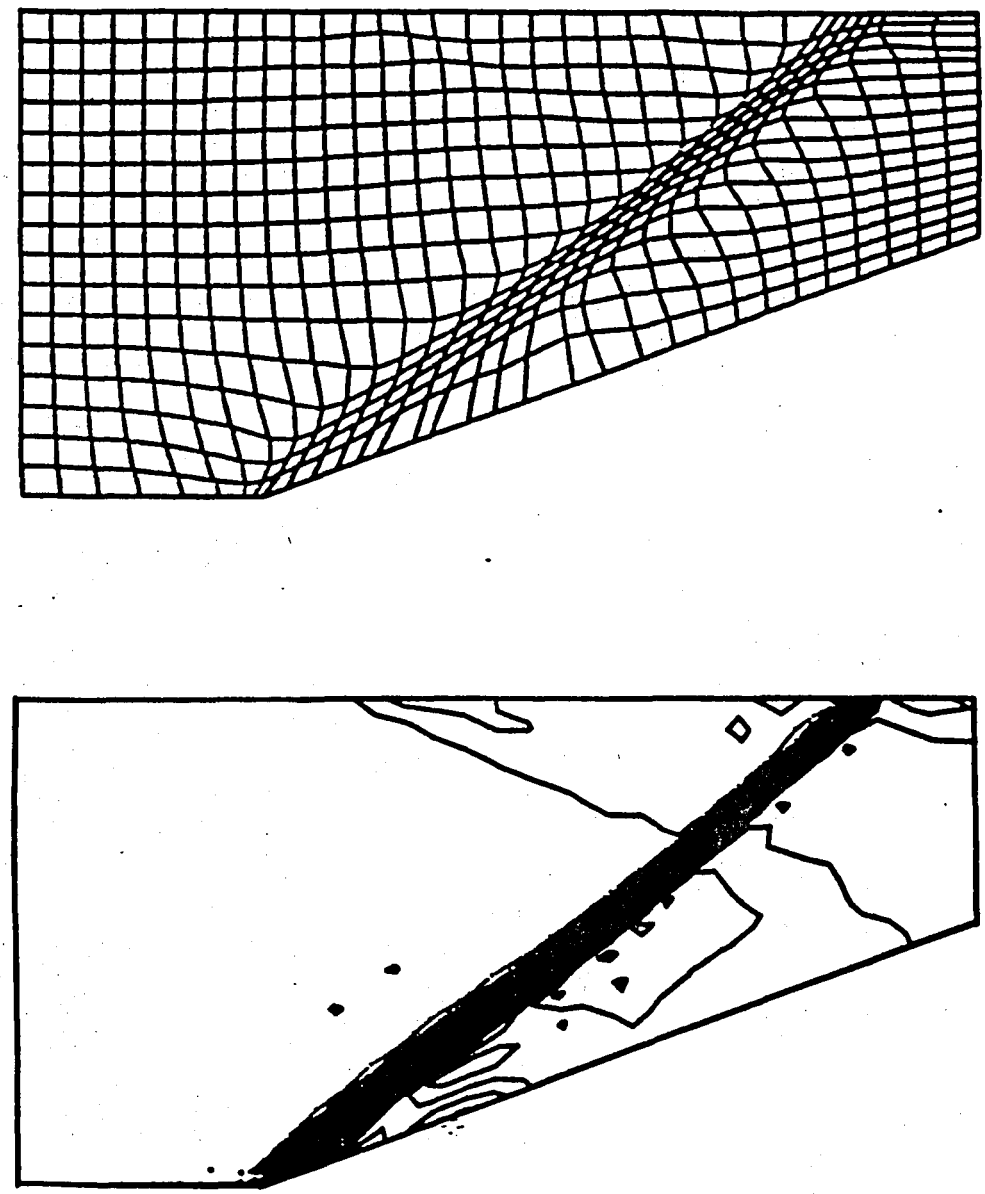


Figure 20. Supersonic flow over a 20° ramp. Mesh and density contours obtained after 10 applications of the mesh redistribution algorithm.

and $0°$ angle of attack was solved to obtain the steady-state solution. This problem has also been studied by Bey et al. [6].

A coarse mesh solution is indicated in Figure 21 and an adaptively refined/unrefined mesh and solution, obtained for $\alpha = 0.05$ and $\beta = 0.15$, are shown in Figure 22. A distorted mesh and corresponding density map are indicated in Figure 23. In this particular problem, neither the h-method nor the r-method gave particularly good results, as a poor approximation of the solution between the shock and blunt body results from spurious oscillations in the basic time-marching algorithm. In the case of mesh adaptation using redistribution, the solution actually diverges after four passes through the adaptive scheme due to the badly graded (hourglassed) mesh produced from the oscillations of the adaptive scheme downstream of the shock.

6.6 Transient Adaptive Solution for Supersonic Flow Over a 20° Ramp. In all the examples presented above, a time-accurate time stepping scheme is used, but the adaptive scheme was not used stepwise for the transient solution since our primary interest was to increase accuracy in the steady-state solution. The adaptive method used to track transient fronts is described as follows:

1) Choose a structured mesh with the finest mesh size to be allowed in the calculation to be the initial mesh. This is done to avoid large variations of the time-step during the time-stepping.
2) Ever N time steps ($N = 50$ in the present problem) go through the refinement-unrefinement process (only unrefinement after the first N time steps).

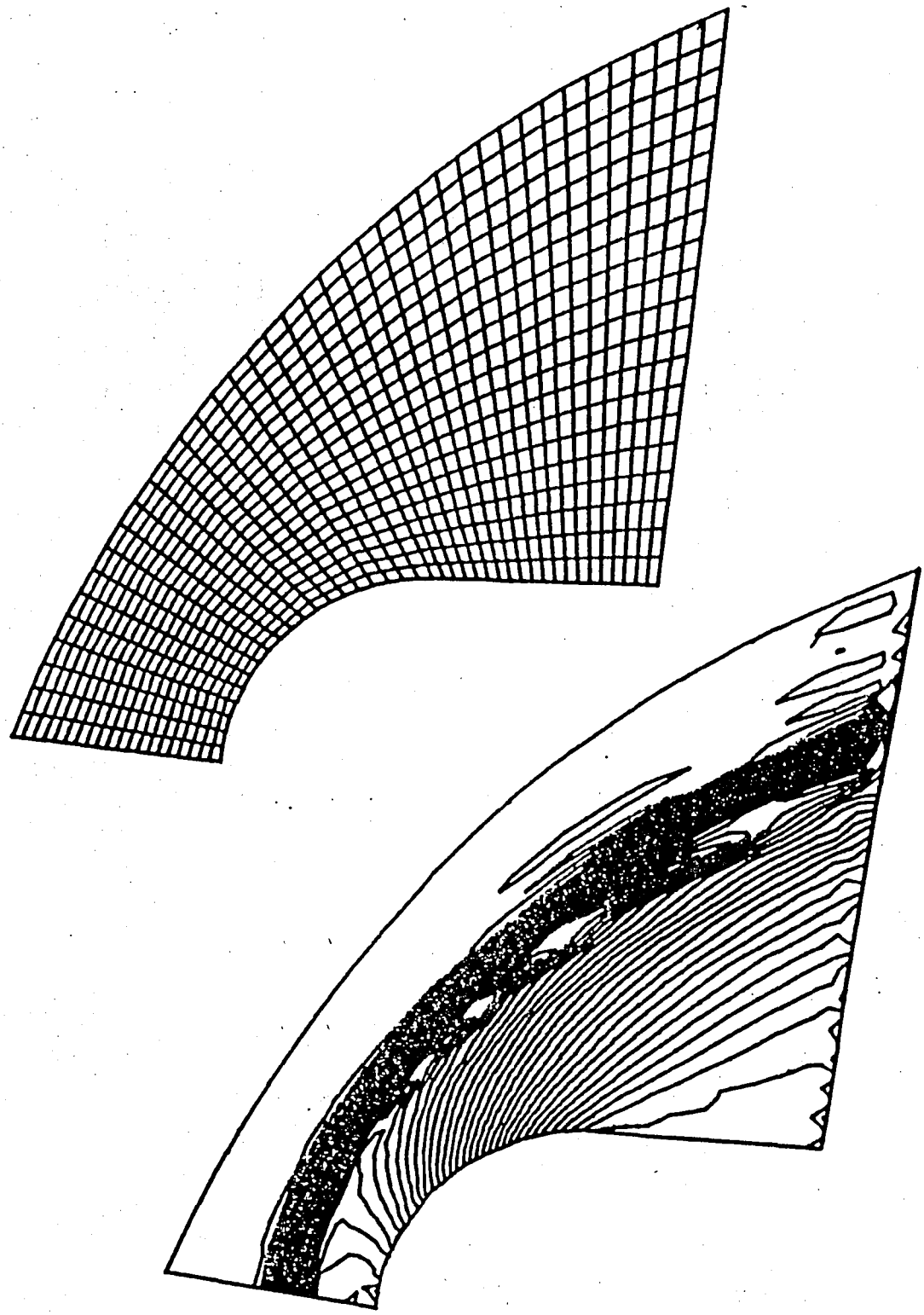


Figure 21. Blunt leading edge in hypersonic flow field.

Initial mesh and density contours.

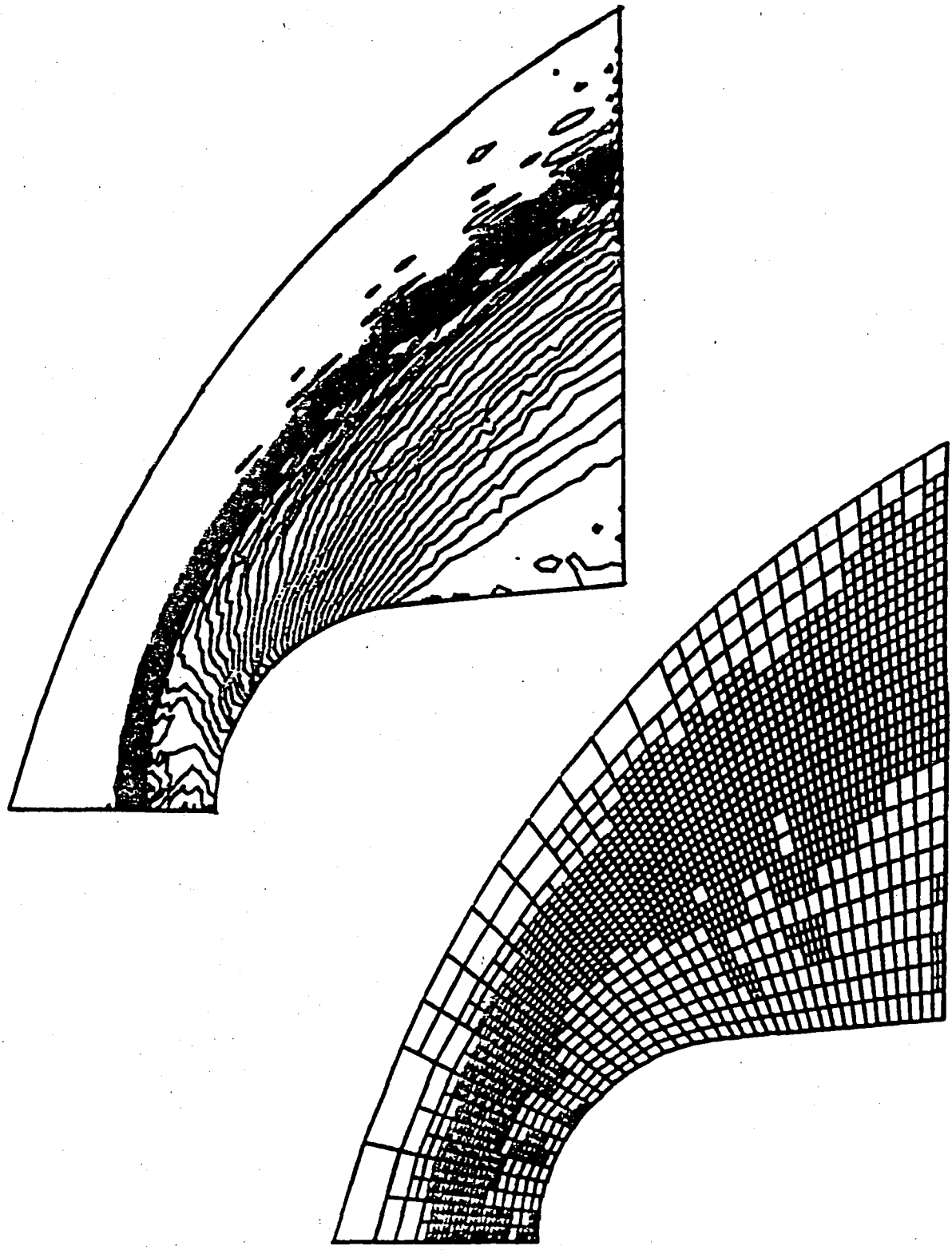


Figure 22 Blunt leading edge in hypersonic flow field. Mesh and density contours obtained after, with one level of refinement (= 0.05, = 0.15).

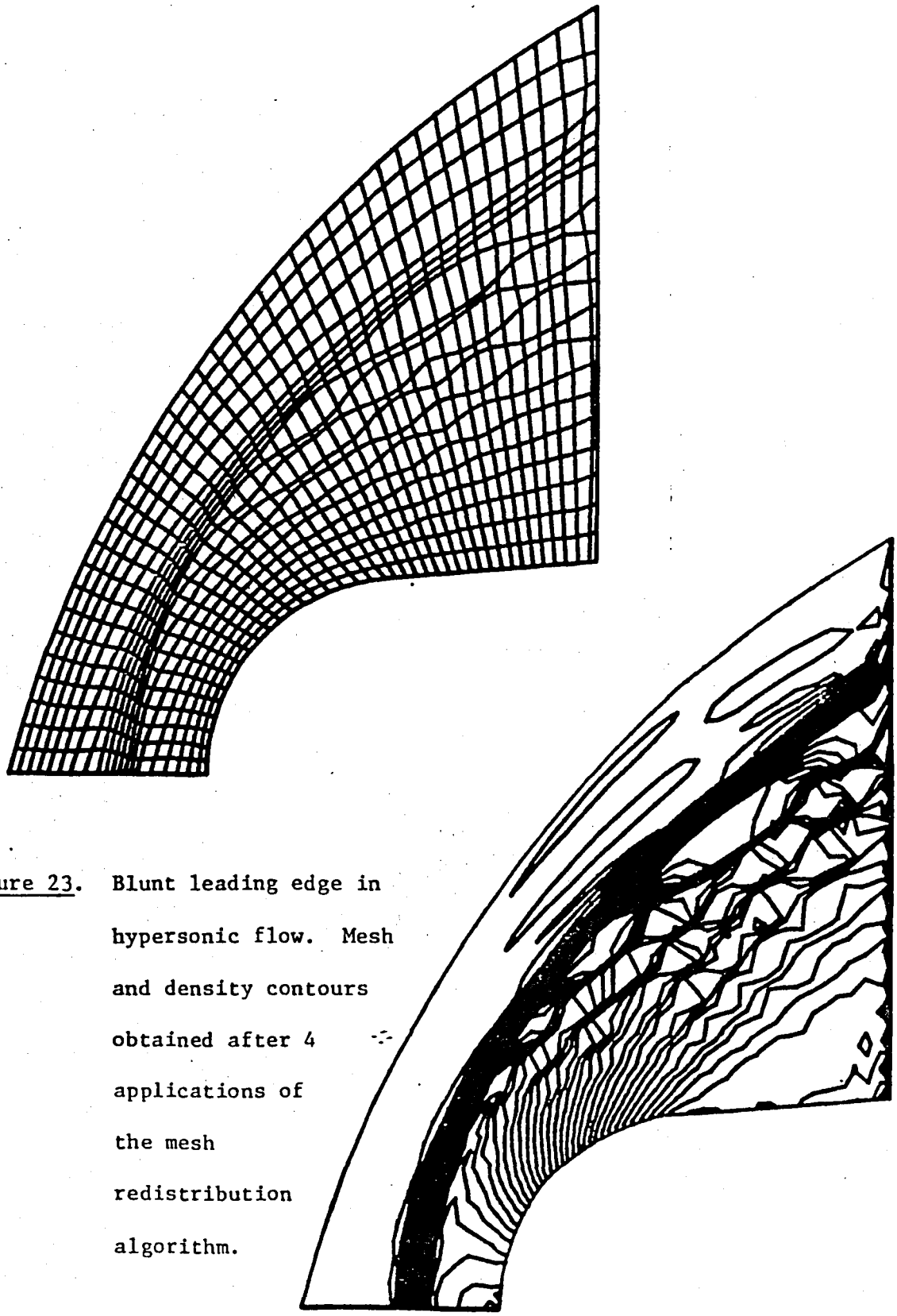


Figure 23. Blunt leading edge in hypersonic flow. Mesh and density contours obtained after 4 applications of the mesh redistribution algorithm.

The above adaptive strategy was employed to solve the problem of a 20° ramp which is suddenly introduced in a supersonic flow field with $M_\infty = 3.0$, $\gamma = 1.40$. The solution was integrated to steady state, and it is demonstrated that the mesh adapts to the shock front as the shock front moves from its initial to its steady-state position.

The initial coarse mesh is shown in Figure 24 and the evolution of a refined/unrefined mesh for various time intervals is illustrated in successive figures, Figures 25-29. The refinement parameters used were $\alpha = 0.05$ and $\beta = 0.25$, and a total of 250 time steps were used to track the solution from its initial to the final steady state. The final steady result is similar to that obtained earlier and shown in Figures 18 and 19.

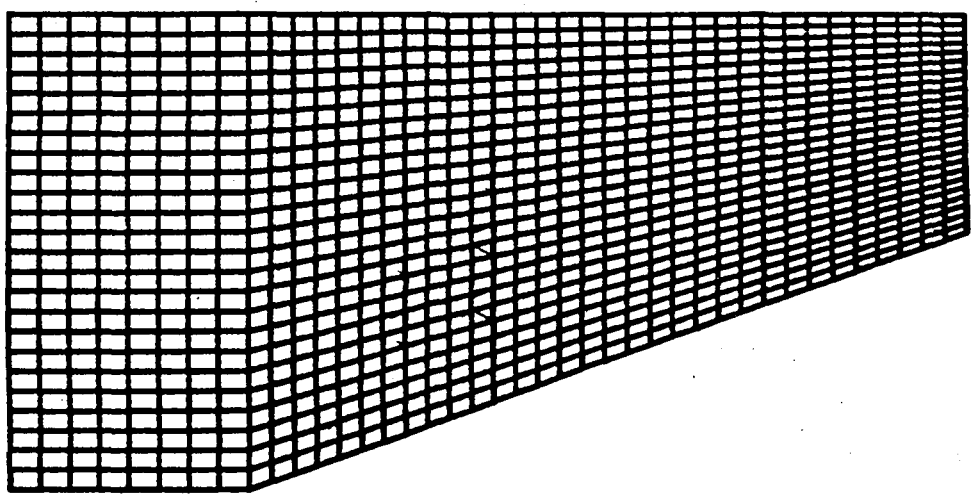


Figure 24. Transient supersonic flow over a 20° ramp. Initial mesh.

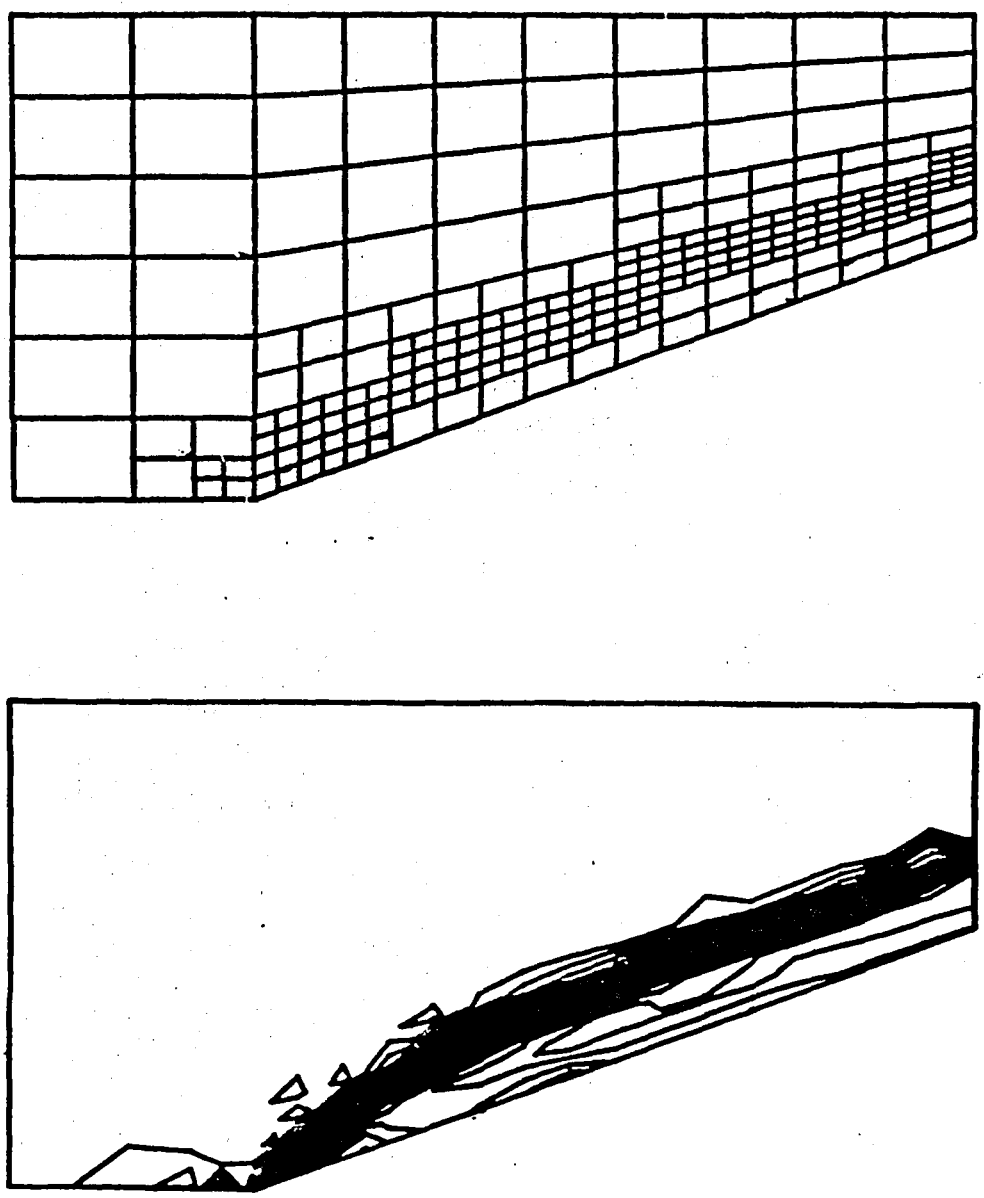


Figure 25. Transient supersonic flow over a 20° ramp. Mesh and density contours after 50 time steps ($\alpha = 0.05$, $\beta = 0.25$).

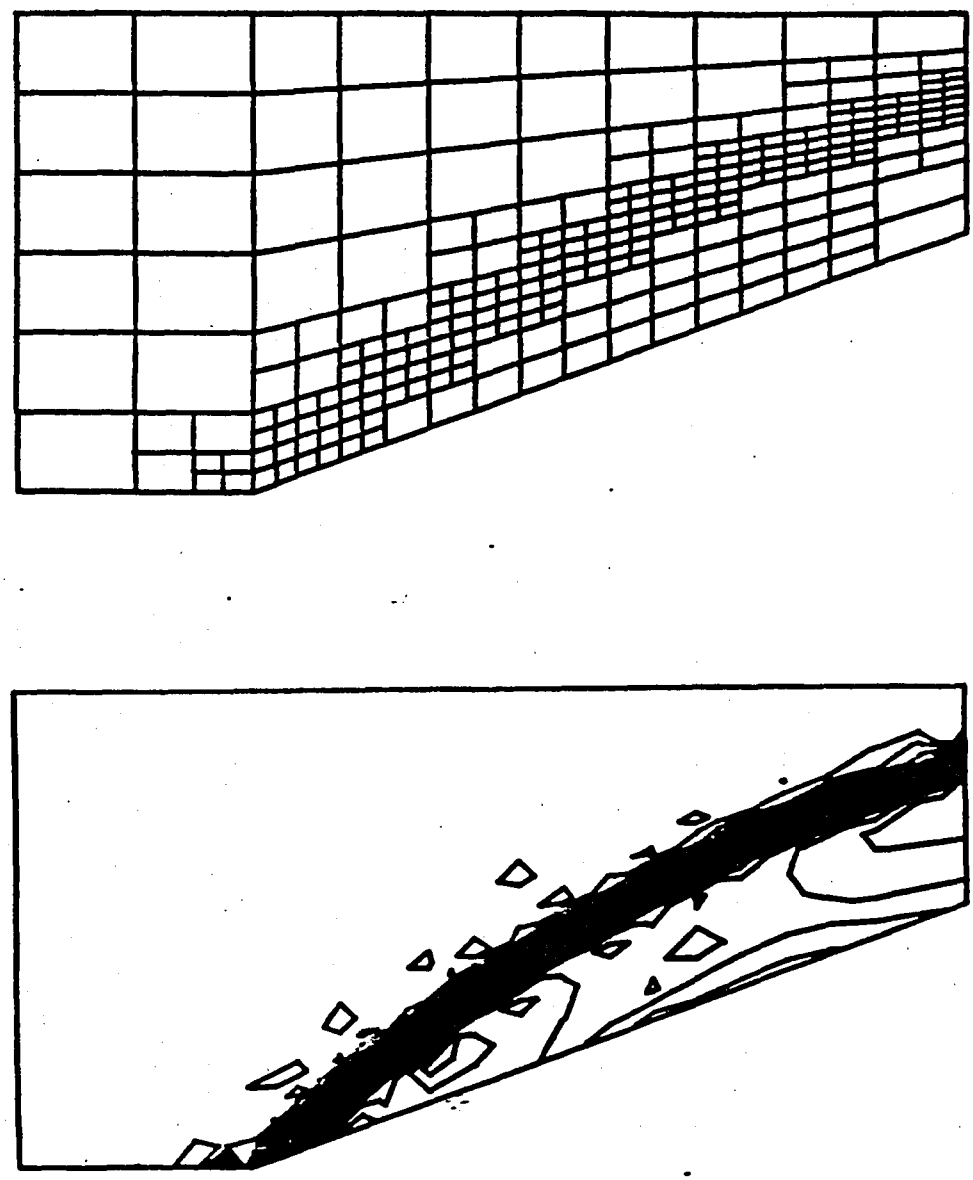


Figure 26. Transient supersonic flow over a 20° ramp. Mesh and density contours after 100 time steps (α = 0.05, β = 0.25).

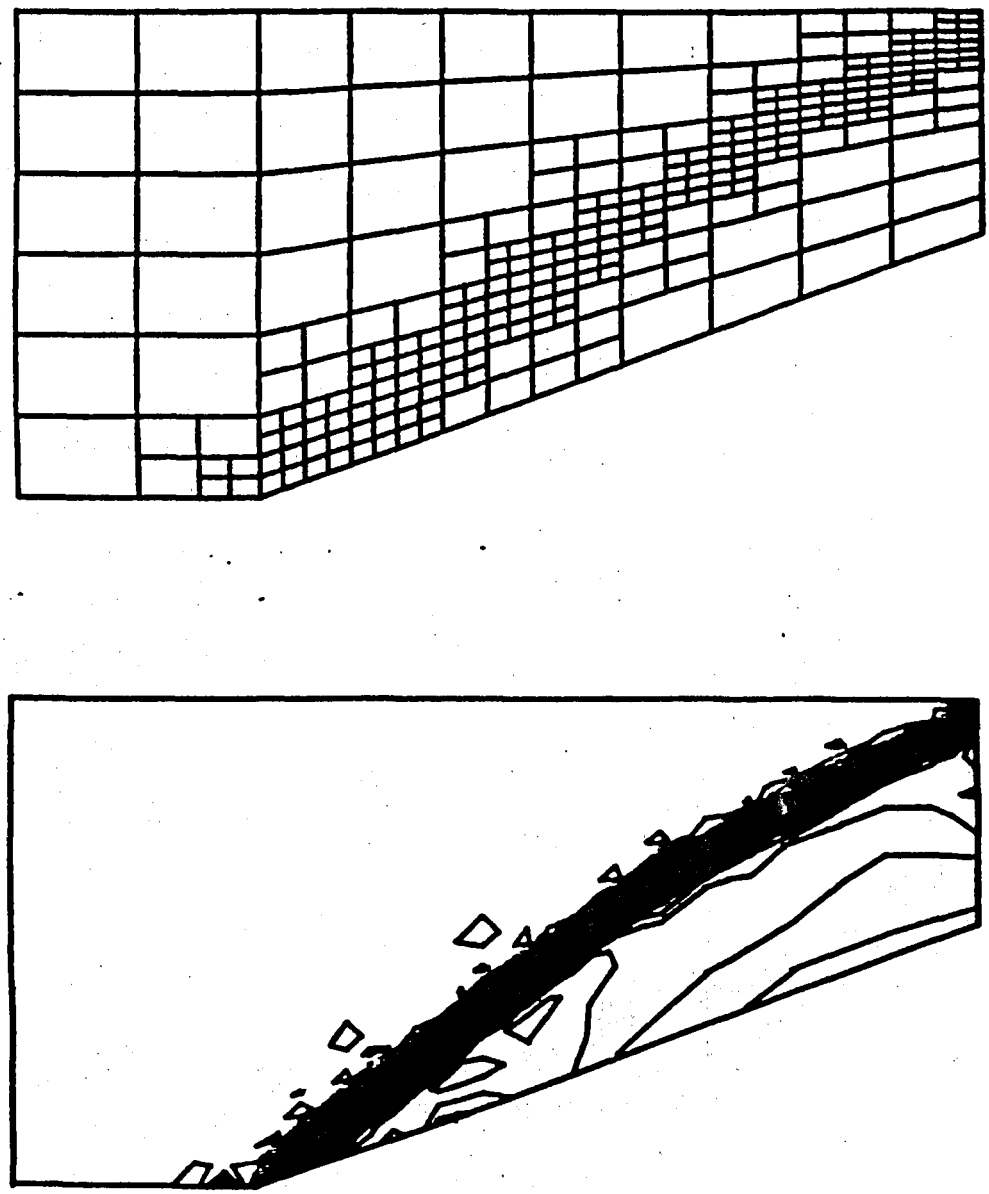


Figure 27. Transient supersonic flow over a 20° ramp. Mesh and density contours after 150 time steps ($\alpha = 0.05$, $\beta = 0.25$).

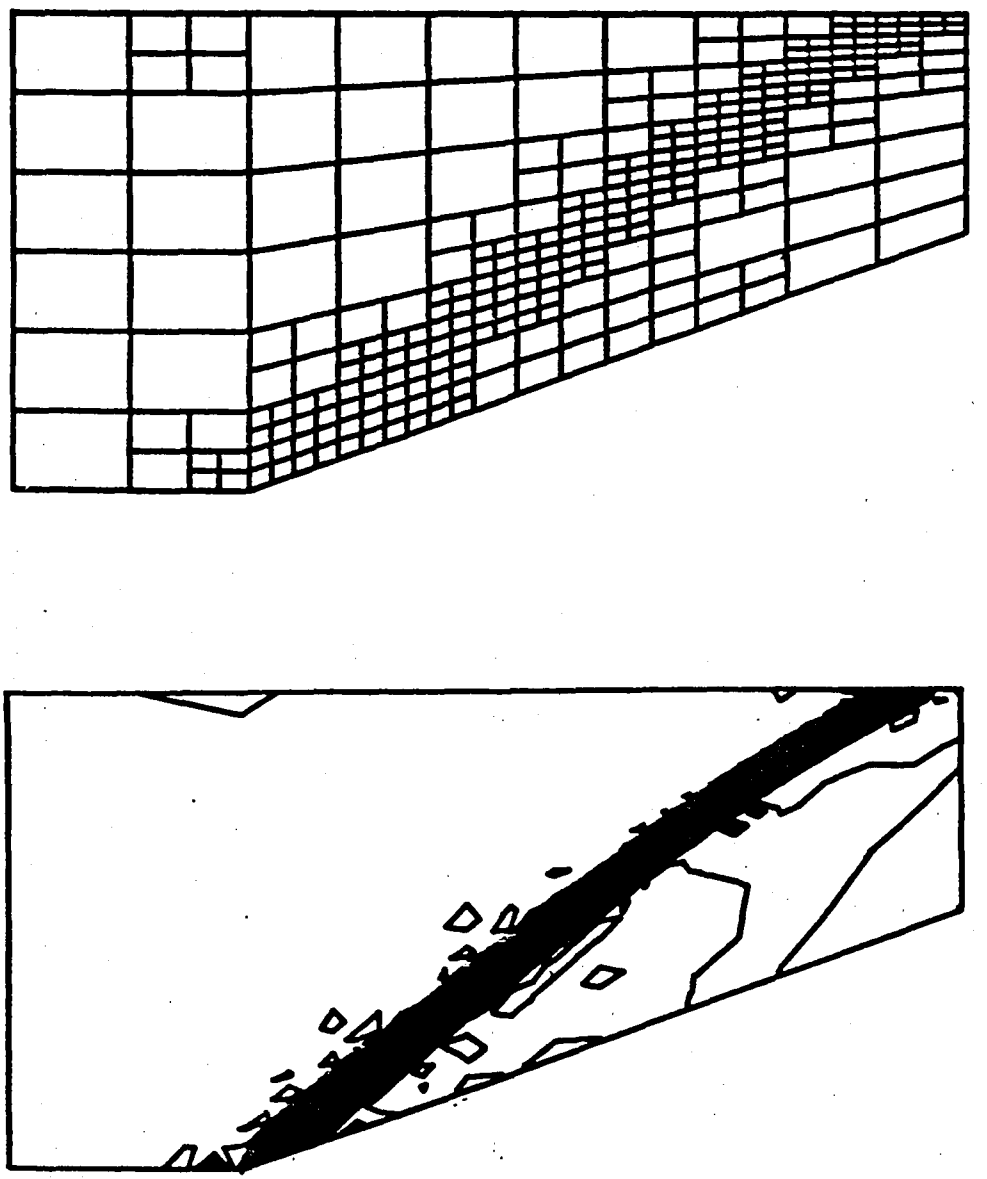


Figure 28. Transient supersonic flow over a 20° ramp. Mesh and density contours after 200 time steps (α = 0.05, β = 0.25).

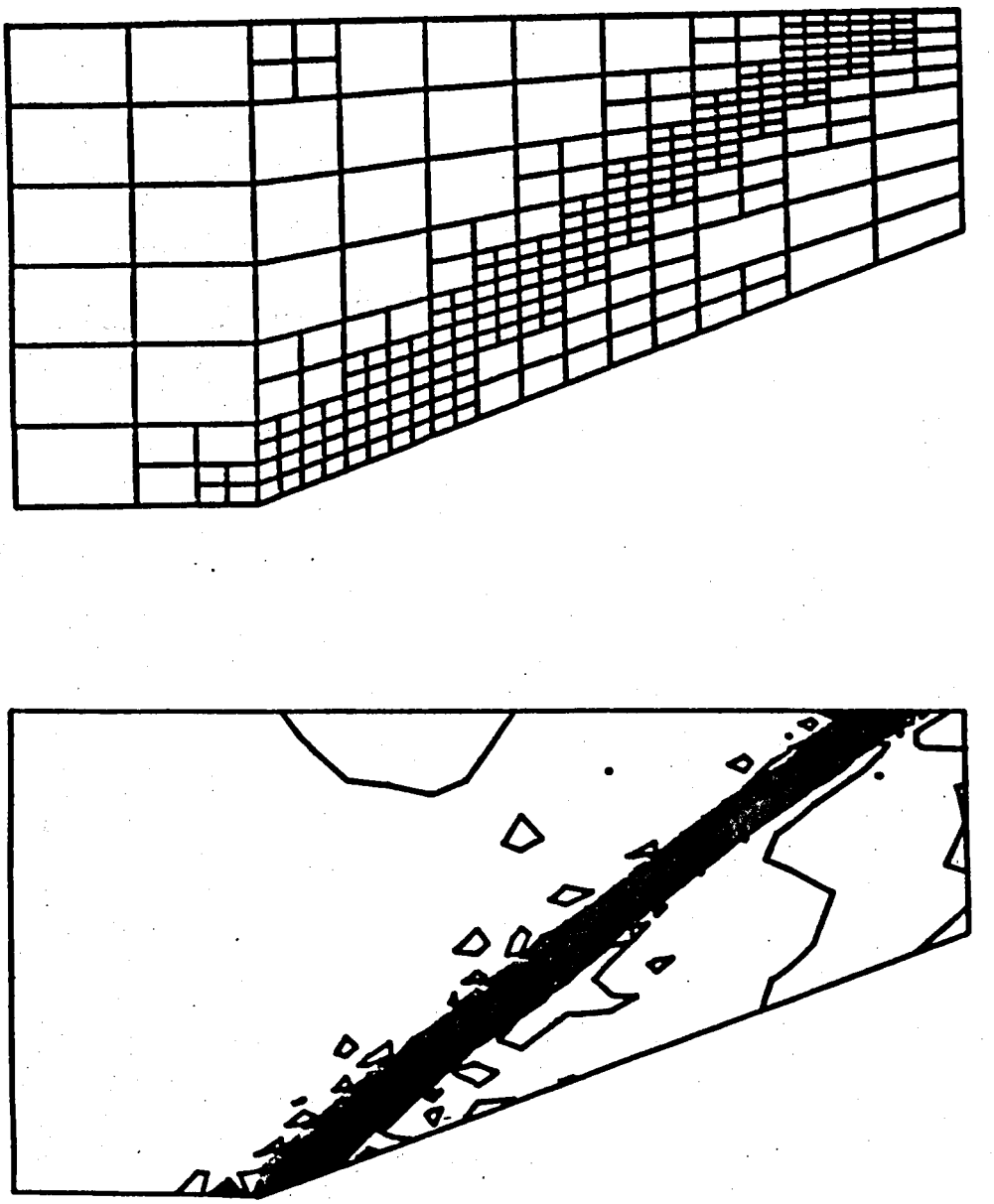


Figure 29. Transient supersonic flow over a 20° ramp. Mesh and density contours after convergence to steady-state, (α = 0.05, β = 0.25).

ACKNOWLEDGMENT

The bulk of the work reported here is a component of a program for the development of finite element methodologies for aerothermal heating analysis and related computational fluid/structural dynamics problems supported by NASA-Langley Research Center. The support of Dr. A. Wieting and Dr. G. Olsen of NASA is gratefully acknowledged. Much of the preliminary work on this effort was done in collaboration with Mr. Lawrence W. Spradley of Lockheed Missiles & Space Company, Huntsville Engineering Center. The adaptive data structures discussed in Section 5 were developed in a study of adaptive finite element methods in solid mechanics sponsored by the Office of Daval Research. The support of some of the effort of Drs. P. Devloo and T. Strouboulis of COMCO, Inc., through Contract NAS8-36647 with NASA-Marshall Space Flight Center is also acknowledged.

REFERENCES

1. Anderson, D. A., "Adaptive Mesh Schemes Based on Grid Speeds," in K. Ghia, U. Ghia (Eds.), Advances in Grid Generation, ASME Special Publication, FED-Vol. 5, New York 1983.

2. Babuska, I., O. C. Zienkiewic, J. P. de S. R. Gago and A. de Oliveira (Eds.), Adaptive Methods and Error Refinement in Finite Element Computation, John Wiley and Sons, Ltd., London 1986.

3. Baker, A. J. and J. W. Kim, "Analyses on a Taylor Weak-Statement Algorithm for Hyperbolic Conservation Laws," Technical Rept., CFDL/86-1, Comp. Fluid Dyn. Lab, University of Tennessee, 1986.

4. Berger, M. and A. Jameson, "Automatic Adaptive Grid Refinements for the Euler Equations," MAE Report, 1633, Princeton University, October 1983.

5. Berger, M. J. and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," Journal of Computational Physics, Vol. 53, No. 3 (1984), 484-512.

6. Bey, K. S., E. A. Thornton, P. Dechaumpai and R. Ramakrishnan, "A New Finite Element Approach for Prediction of Aerothermal Loads -- Progress in Inviscid Flow Computations," AIAA Paper No. 85-1533-CP.

7. Burstein, S. Z., "Finite Difference Calculations for Hydrodynamic Flows Containing Discontinuities," Journal of Computational Physics, 2 (1967), 198-222.

8. Demkowicz, L, J. T. Oden and T. Strouboulis, "Adaptive Finite Elements for Flow Problems with Moving Boundaries. Part I: Variational Principles and *A Posteriori* Estimates," Computer Methods in Applied Mechanics and Engineering, 46 (1984), 217-251.

9. Demkowicz, L., J. T. Oden and T. Strouboulis, "An Adaptive p-Version Finite Element Method for Transient Flow Problems with Moving Boundaries," in R. H. Gallagher, G. F. Carey, J. T. Oden and O. C. Zienkiewicz (Eds.), Finite Elements in Fluids, Vol. 6, John Wiley and Sons, Ltd., London 1985.

10. Demkowicz, L. and J. T. Oden, "An Adaptive Characteristic Petrov-Galerkin Finite Element Method for Convection Dominated Linear and Non-Linear Parabolic Problems in One Space Variable," TICOM Report 85-3, The University of Texas at Austin, April 1985.

11. Demkowicz, L. and J. T. Oden, "An Adaptive Characteristic Petrov-Galerkin Finite Element Mthod for Convection-Dominated Linear and Nonlinear Parabolic Problems in Two Space Variables," to appear in Computer Methods in Applied Mechanics and Engineering.

12. Diaz, A. R., N. Kikuchi and J. E. Taylor, "A Method of Grid Optimization for Finite Element Methods," Comp. Meth. in Appl. Mech. and Eng., 41, 1983, 29-45.

13. Donéa, J., "A Taylor Galerkin Method for Convective Transport Problems," Int'l. Journal for Numerical Methods in Engineering, 20 (1984), 101-119.

14. Lapidus, A., "A Detached Shock Calculation by Second-Order Finite Differences," Journal of Computational Physics, 2 (1967), 154-177.

15. Lohner, R., K. Morgan and O. C. Zienkiewicz, "Adaptive Grid Refinement for the Euler and Compressible Navier-Stokes Equations," in I. Babuska, O. C. Zienkiewicz, J. P. de S. R. Gago and A. de Oliveira (Eds.), Adaptive Methods and Error Refinement in Finite Element Computation, John Wiley and Sons, Ltd., London 1986.

16. Lohner, R., K. Morgan and O. C. Zienkiewicz, "An Adaptive Finite Element Procedure for Compressible High Speed Flows," to appear in Computer Methods in Applied Mechanics and Engineering.

17. Oden, J. T. and G. F. Carey, Finite Elements: Mathematical Aspects, Prentice-Hall, Englewood Cliffs, 1981.

18. Oden, J. T., L. Demkowicz, T. Strouboulis and P. Devloo, "Adaptive Methods for Problems in Solid and Fluid Mechanics," in I. Babuska, O. C. Zienkiewicz, J. P. de S. R. Gago and A. de Oliveira (Eds.), Adaptive Methods and Error Refinement in Finite Element Computation, John Wiley and Sons, Ltd., London 1986.

19. Strouboulis, T., J. T. Oden and L. Demkowicz, "*A-Posteriori* Error Estimates for Some Galerkin Space-Time Approximations of the Unsteady Navier-Stokes Equations," TICOM Report 85-5, The University of Texas at Austin, June 1985.

20. von Neumann, J. and R. D. Richtmeyer, "A Method for Numerical Calculation of Hydrodynamic Shocks," Journal of Applied Physics, 21 (1950), 232-236.

21. Zienkiewicz, O. C., R. Lohner and K. Morgan, "High Speed Inviscid Compressible Flow by the Finite Element Method," in J. R. Whiteman (Ed.), The Mathematics of Finite Elements and Applications, V-MAFELAP 1984, Academic Press, Ltd., London 1985.

Appendix D

ADAPTIVE FINITE ELEMENT METHODS FOR THE ANALYSIS OF INVISCID COMPRESSIBLE FLOWS USING A FEM-FCT SCHEME

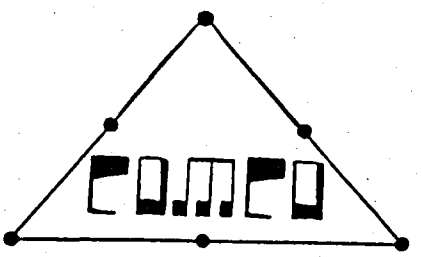
ADAPTIVE FINITE ELEMENT METHODS
FOR THE ANALYSIS OF
INVISCID COMPRESSIBLE FLOWS

TR-87-06

An Final Technical Report
to the
LOCKHEED MISSILES & SPACE COMPANY
HUNTSVILLE, ALABAMA

pertaining to
Contract No. SM80D7030F

October, 1987



THE COMPUTATIONAL MECHANICS COMPANY, INC.

Computational Mechanics Company, Inc.
4804 Avenue H
Austin, Texas 78751

CONTENTS

CONTENTS (Concluded)

LIST OF FIGURES

THE COMPUTATIONAL MECHANICS COMPANY, INC.
4804 Avenue H, Austin, Texas 78751

TECHNICAL LETTER - FINAL REPORT

Contract # SM80D7030
September 30, 1987

Chapter 1

This report concerns an adaptive finite element code, capable of solving transient and steady-state problems in compressible inviscid fluid flow. Unstructured triangular finite element meshes were used for the basis of the adaptivity.

Some research has been carried out in the area of adaptivity (see [2],[4],[7] and [10]), although many different approaches have been pursued. For example, an approach involving quadrilateral elements was seen to provide accurate results in [4]. Devloo, [4], used one type of quadrilaterial element division, resulting in the generation of contrained nodes (i.e., the nodal solution is contrained in terms of other nearly nodal solutions.) This approach combined with a scheme (FEM-FCT, see Chapter 3), capable of capturing line discontinuities in the flow, was seen to improve the accuracy of the results. Others, such as Bank [2], have opted for triangular elements and two different types of division. This choice of two divisions results in a mesh of unconstrained nodes, noted to provide accurate pictures of fluid-flow interaction.

In all these references, adaptivity was seen to improve the solution and the speed with which the solution was computed. Considering the types of problems (i.e. compressible fluid flow), an adaptive triangular element approach with a suitable numerical scheme was hence adopted as the basis of this report. The adaptive part of the code forms the basis of the work carried out an an already coded

FCTG-FEM "solver" (that used by Strouboulis in [13]). Parts of the "solver" had to be changed to accommodate triangular elements. Full details of the adaptive part are presented in Chapter 4.

Several classical fluid flow problems were analyzed. The initial conditions and solutions were obtained from references such as [11], [12], thereby enabling valid comparisons to be made. The problems all involved some form of line discontinuities such as shock/shock-interaction. The adaptivity was seen to help capture the true form of these discontinuities, providing similar solutions to those contained in [12]. The shock resolutions were accurate enough to justify the use of such a scheme.

A brief summary of this report would thus be as follows: Chapters 2 and 3 describe the FEM-FCT numerical scheme used; Chapter 4 describes the adaptive strategy; Chapter 5 includes numerical examples and; Chapter 6 gives suggestions and recommendations for further developments.

CHAPTER 2

FINITE ELEMENT APPROXIMATIONS

Before describing the details of the Flux-Corrected Method (adapted to the Finite Element Method - See Chapter 3), a brief background to the class of problems and method of solution will be given.

All of the problems analysed involve the time-dependent flow of a compressible inviscid fluid in two-dimensions. If we denote the domain of flow by Ω and the time interval of this flow by [0,T] then the true domain D is defined by $D = \Omega \times [0,T]$. A sketch of such a domain is given in Figure 2.1 including $\partial\Omega$, the boundary of the domain. This domain is then used to compute a solution vector U, consisting of the following four components :

$$\mathbf{U} = \left[\rho \, , \, \rho u \, , \, \rho v \, , \, \rho e \right]^t \qquad (2.1)$$

where : ρ = density

ρu = linear momentum in the x-direction.

ρv = linear momentum in the y-direction.

ρe = total energy.

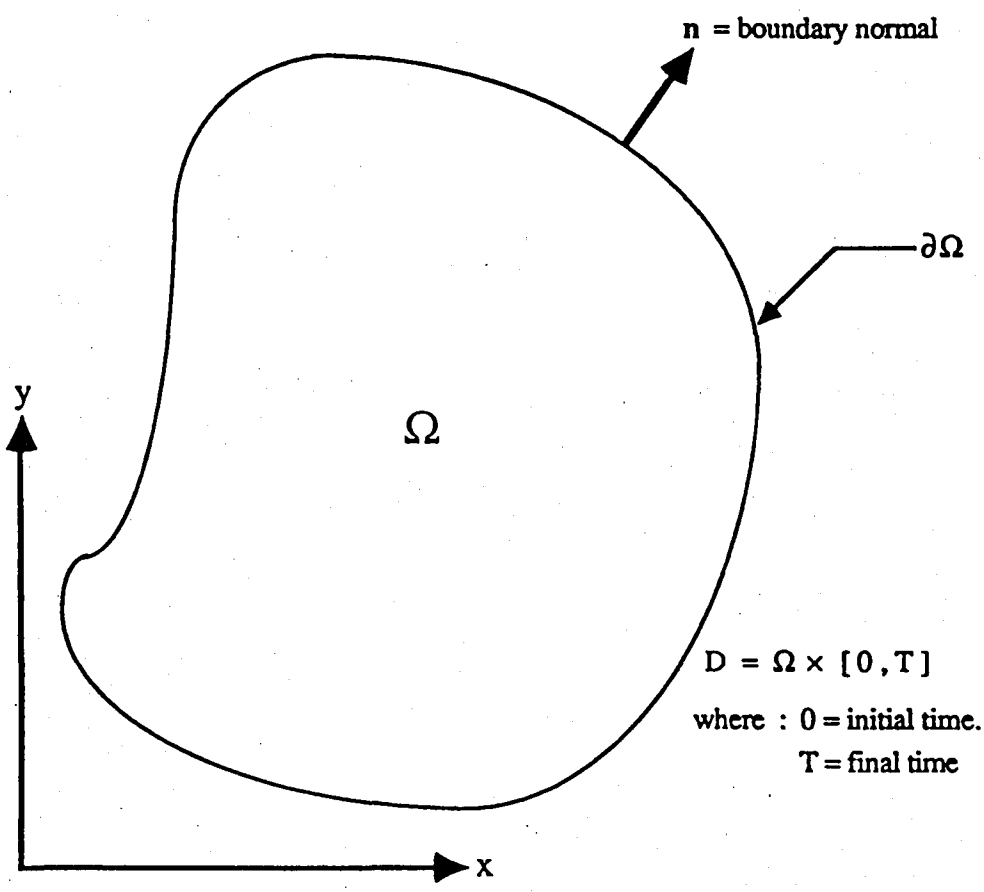


Figure 2.1 The space-time domain (D) of problems

These four conservation variables can be used to calculate the following four primitive variables : ρ , u , v and p.

where : u = velocity in the x-direction.

v = velocity in the y-direction.

p = pressure.

Note that the pressure is calculated from the following formula for a perfect gas :

$$p = (\gamma - 1)\rho\left[e - \frac{u^2 + v^2}{2}\right]$$

where : ρ , e , u and v are as specified in equation (2.1).

γ = ratio of specific heats for the gas/liquid under consideration.

The fluid flow is governed by the balance laws of mass, momentum and energy conservation. These equations, expressed in integral form, are known as the Euler equations for flow of a compressible inviscid fluid. This integral form can be written as follows :

$$\frac{d}{dt}\int_{\Omega} U \, d\Omega = -\int_{\partial\Omega} Q \cdot n \, ds \qquad (2.2)$$

where : U = solution vector.

$d\Omega$, ds = Lebesgue measures of length and area.

n = vector normal to the boundary.

Q = pair of flux vectors E, F.

The form of Q is as follows :

$$Q = [\, E \,,\, F \,] = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho v u \\ \rho u v & \rho v^2 + p \\ (\rho e + p) u & (\rho v + p) u \end{bmatrix}$$

Besides satisfying equation (2.2), U must also satisfy an initial condition given at $t = 0$,ie.

$$U(x, 0) = U_0(x) \qquad x \in \Omega$$

Now to obtain a Galerkin approximation, we must consider the differential form of equation (2.2).

$$\frac{\partial U}{\partial t} + \nabla \cdot Q = 0 \qquad (2.3)$$

$$\text{where} : \nabla \cdot Q = \sum_{i=1}^{2} \frac{\partial Q_{\alpha i}}{\partial x_i}$$

If we multiply equation (2.3) by a test function φ^T and integrate, we obtain the following weak boundary-value problem which is equivalent to the conservation laws, the initial conditions and jump conditions :

$$\frac{\partial}{\partial t}\int_{\Omega} \varphi^T U \, d\Omega = \int_{\Omega} Q \cdot \nabla \varphi^T - \int_{\partial\Omega} \varphi^T Q \, ds \qquad (2.4)$$

$$\text{where: } \varphi^T U = \sum_{i=1}^{4} U_i \, \varphi_i$$

$$Q \cdot \nabla \varphi = \sum_{i=1}^{2} \sum_{j=1}^{4} Q_{ji} \frac{\partial \varphi_j}{\partial x_i}$$

CHAPTER 3

FLUX-CORRECTED TRANSPORT METHOD

In Chapter 2, a weak variational statement of the conservation laws was formulated, governing the flow of a fluid through a region Ω (see equation 2.4). For such flows, point and line discontinuities can occur in the primitive variables in use. Depending upon the scheme being used, these discontinuities can cause problems. In particular it is known that those schemes of order greater than one will tend to cause oscillations in the solution at and about such discontinuities. If these oscillations are large enough, the solution will eventually become unstable and diverge. A method able to deal with these oscillations would hence enhance the solution greatly.

The Flux-Corrected Transport (FCT) Method is such a method as it employs the use of both a high- and low-order numerical scheme. The idea behind this is to use the high-order scheme in areas where the primitive variables change smoothly and not abruptly. The low-order scheme is then employed in those areas where the variables vary abruptly (such as along the line of a shock-wave). The combination of these two schemes near such discontinuities tends to provide an accurate picture although slight oscillations can still mar the solution. For this reason, a strong diffusion term is added to the low-order solution which tends to reduce the magnitude of these spurious solution oscillations.

Before a detailed description of the high- and low- order schemes is given, a rough outline of the FCT method is given:

If we discretize equation (2.3) with respect to time, we obtain an equation of the following form (using the standard high-order method) :

$$U^{n+1} = U^n + \Delta U^h \qquad (3.1)$$

where : ΔU^h = the increment in the solution vector corresponding to a change in time from t_n to t_{n+1}.

The FCT method computes a ΔU^h of high enough order to capture the solution with few oscillations. Rewriting equation (3.1) in terms of low- and high-order contributions yields the following :

$$U^{n+1} = U^n + \Delta U^l + (\Delta U^h - \Delta U^l)$$
$$= U^l + (\Delta U^h - \Delta U^l)$$

where : U^l = low-order solution at $t=t_{n+1}$.

ΔU^l = low-order increment in U from $t=t_n$ to $t=t_{n+1}$.

Hence the FCT method limits the second term above and sets it equal to ΔU^*

$$U^{n+1} = U^l + \Delta U^* \qquad (3.2)$$

where : $\Delta U^* = (\Delta U^h - \Delta U^l)$

Obviously ΔU^* must be "limited" very carefully so as to avoid either oscillations or lack of resolution in the solution. Before describing any finer details of the FCT method, the high- and low- order schemes will be described.

3.1 High-order Scheme : Taylor-Galerkin method.

First we discretize U with respect to time using the midpoint formula :

$$U^{n+1} = U^n + 2\left(\frac{\Delta t}{2}\right)\left(\frac{\partial U}{\partial t}\right)^{n+\frac{1}{2}} + O(\Delta t^3)$$

$$\equiv U^n + \Delta t\left(\frac{\partial U}{\partial t}\right)^{n+\frac{1}{2}} \qquad (3.3)$$

where : U^{n+1} = solution vector at $t=t^{n+1}$.

U^n = solution vector at $t=t^n$.

$(\partial U / \partial t)^{n+1/2}$= solution derivative at $t=t^{n+1/2}$.

From Chapter 2 and the governing differential equation (equaton 2.3) :

$$\left(\frac{\partial U}{\partial t}\right)^{n+\frac{1}{2}} = -\left(\frac{\partial E}{\partial x}\right)^{n+\frac{1}{2}} - \left(\frac{\partial F}{\partial y}\right)^{n+\frac{1}{2}} \qquad (3.4)$$

(obtained by substituting : $\nabla \cdot Q = \nabla \cdot [E, F]$

$$= \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} \quad)$$

If equation (3.4) is substituted into equation (3.3), the variational form of equation (3.3) becomes :

$$\int_{\Omega} \varphi_j^T U^{n+1} d\tau = \int_{\Omega} \varphi_j^T U^n d\tau - \Delta t \int_{\Omega} \varphi_j^T \left(\left(\frac{\partial E}{\partial x} \right)^{n+\frac{1}{2}} + \left(\frac{\partial F}{\partial y} \right)^{n+\frac{1}{2}} \right) d\tau \qquad (3.5)$$

Now, we know that :

$$\frac{\partial}{\partial x}\left(E^{n+\frac{1}{2}} \varphi_j^T \right) = E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} + \varphi_j^T \left(\frac{\partial E}{\partial x} \right)^{n+\frac{1}{2}} \quad \text{or}$$

$$\left(\frac{\partial E}{\partial x} \right)^{n+\frac{1}{2}} \varphi_j^T = \frac{\partial}{\partial x}\left(E^{n+\frac{1}{2}} \varphi_j^T \right) - E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x}$$

Similarly for F :

$$\left(\frac{\partial F}{\partial y} \right)^{n+\frac{1}{2}} \varphi_j^T = \frac{\partial}{\partial y}\left(F^{n+\frac{1}{2}} \varphi_j^T \right) - F^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial y}$$

Using the above, equation (3.5) becomes :

$$\int_{\Omega} U^{n+1} \varphi_j^T d\tau = \int_{\Omega} U^n \varphi_j^T d\tau$$

$$+\Delta t \int_{\Omega} \left(E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} + F^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial y} \right) d\tau$$

$$-\Delta t \int_{\Omega} \left(\frac{\partial}{\partial x} \left(E^{n+\frac{1}{2}} \varphi_j^T \right) + \frac{\partial}{\partial y} \left(F^{n+\frac{1}{2}} \varphi_j^T \right) \right) d\tau$$

$$= \int_{\Omega} U^n \varphi_j^T d\tau$$

$$+\Delta t \int_{\Omega} \left(E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} + F^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial y} \right) d\tau$$

$$-\Delta t \int_{\partial\Omega} \left(E^{n+\frac{1}{2}} n_x + F^{n+\frac{1}{2}} n_y \right) ds \qquad (3.6)$$

where : n_x , n_y = rectangular vector components of the normal to the boundary (see Figure 2.1).

$\partial\Omega$ = boundary of Ω (see Figure 2.1).

Finally, the solution vector U is discretized by :

$$U^{n+1} = \sum_{i=1}^{NNODE} U_i^{n+1} \varphi_i^T \quad , \quad U^n = \sum_{i=1}^{NNODE} U_i^{n+1} \varphi_i^T$$

where : NNODE= number of nodal points.

φ_i = Galerkin shape functions (provide a basis of finite dimension).

Equation (3.6) thus becomes :

$$\sum_{i=1}^{NNODE} U_i^{n+1} \int_\Omega \varphi_i^T \varphi_j^T \, d\tau = \sum_{i=1}^{NNODE} U_i^n \int_\Omega \varphi_i^T \varphi_j^T \, d\tau$$

$$+ \Delta t \int_\Omega \left(E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} + F^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} \right) d\tau$$

$$- \Delta t \int_{\partial\Omega} \left(E^{n+\frac{1}{2}} n_x \varphi_j^T + F^{n+\frac{1}{2}} n_y \varphi_j^T \right) d\tau$$

If we denote $\int_\Omega \varphi_i^T \varphi_j^T \, d\tau$ as M_{ij}, the mass matrix, equation (3.6) can be written as follows :

$$\underline{M}\, U^{n+1} = \underline{M}\, U^n + Q \qquad (3.7)$$

where : $$Q = \Delta t \int_\Omega \left(E^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial x} + F^{n+\frac{1}{2}} \frac{\partial \varphi_j^T}{\partial y} \right) d\tau$$

$$- \Delta t \int_{\partial\Omega} \left(E^{n+\frac{1}{2}} n_x \varphi_j^T + F^{n+\frac{1}{2}} n_y \varphi_j^T \right) ds$$

Note that Q is expressed in terms of $E^{n+1/2}$ and $F^{n+1/2}$ which are both functions of $U^{n+1/2}$. Everything else in equation (3.7) is known or can be computed

(e.g. U^{n+1}) if $U^{n+1/2}$ can be computed. To calculate $U^{n+1/2}$, we again use equation (2.3), formed in Chapter 2. The process is identical to the one for computing U^{n+1} except for the time interval which spreads from $t=t^n$ to $t=t^{n+1/2}$. Hence we use :

$$U^{n+\frac{1}{2}} = U^n + \frac{\Delta t}{2}\left(\frac{\partial U^n}{\partial t}\right) + O(\Delta t^3)$$

$$\approx U^n - \frac{\Delta t}{2}\left(E_x^n + F_y^n\right)$$

The following weak statement is made:

$$\int_\Omega U^{n+\frac{1}{2}} \varphi_j^T d\tau = \int_\Omega U^n \varphi_j^T d\tau - \frac{\Delta t}{2}\int_\Omega (E_x^n + F_y^n)\varphi_j^T d\tau \quad (3.8)$$

U^n (and hence E_x , F_y) are known, enabling $U^{n+1/2}$ to be computed. A brief summary of the high-order process is as follows :

(1) Using equation (3.8), compute $U^{n+1/2}$, knowing U^n, E^n and F^n.

(2) Substitute $U^{n+1/2}$ into equation (3.7) and compute U^{n+1}.

(3) Repeat process to advance a further time-step.

3.2 Low-order scheme : Lumped mass matrix.

As mentioned earlier, the low-order scheme is used really to capture line or point discontinuities. As such, it should provide an oscillation-free solution to

prevent possible later numerical instability. The scheme uses a lumped mass-matrix $\underline{M}_l$ instead of the matrix used in the high-order scheme ($\underline{M}$). $\underline{M}_l$ is obtained by summing the elements of each row and placing the sum in the diagonal position ,e.g.

$$\underline{M}_l = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} \qquad \underline{M} = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 2 \end{bmatrix}$$

The form of the resulting equation is similar to equation (3.7) except for the addition of a strong diffusion term. This diffusion term, V , aids in dampening out any oscillations in the solution near the discontinuities (shocks) and is included as follows :

$$\underline{M}_l \, U_l^{n+1} = \underline{M}_l \, U_l + Q + V \qquad (3.9)$$

3.3 Iterative form.

The low- and high- order schemes have the following form :

$$\underline{M} \, U^{n+1} = \underline{M} \, U^n + Q \quad \text{(high-order)}$$

$$\underline{M}_l \, U^{n+1} = \underline{M}_l \, U^n + Q + V \quad \text{(low-order)}$$

Equation (3.7) can be rewritten as :

$$\underline{M}\,U^{n+1} + \underline{M}_1\,U^{n+1} - \underline{M}_1\,U^{n+1} = \underline{M}\,U^{n} + Q$$

$$\therefore\ (\underline{M} - \underline{M}_1)\,U^{n+1} + \underline{M}_1\,U^{n+1} = \underline{M}\,U^{n} + Q$$

The iteration is as follows :

$$\underline{M}_1\ U^{n+1}_{[i]} = \underline{M}\,U^{n} + Q - (\underline{M} - \underline{M}_1)\,U^{n+1}_{[i-1]} \qquad (3.10)$$

where : $U^{n+1}_{[i]}$ = "i 'th" iteration of U^{n+1}

$$U^{n+1}_{[0]} = U^{n}$$

3.4 Flux limiting procedure :

If we take equations (3.9) and (3.10) and subtract them, we get :

$$\underline{M}_l\,(U_{[i]}^{n+1} - U_l^{n+1}) = (\underline{M} - \underline{M}_l)\,U^n - (\underline{M} - \underline{M}_l)\,U_{[i-1]}^{n+1} - V \quad \text{or}$$

$$\underline{M}_l\,U_{[i]}^{n+1} = \underline{M}_l\,U_l^{n+1} + (\underline{M} - \underline{M}_l)\,U^n$$

$$-(\underline{M} - \underline{M}_l)\,U_{[i-1]}^{n+1} - V \qquad \text{or}$$

$$U_{[i]}^{n+1} = U_l^{n+1} + f_{[i-1]}$$

which is similar in form to equation (3.2). The limited increment ΔU^* then corresponds to $f_{[i-1]}$.

3.5 Error estimate.

As the "solver" based on the FCT-FEM method was to be used with adaptivity (see Chapter 4), an error estimator, capable of identifying abrupt changes in the solution vector U, was needed. The error estimator had to be accurate enough to control refinement/unrefinement in areas in the mesh of large/small error. Many references were available in the area of error estimates (see [11], [12]) enabling an appropriate error estimator to be chosen. The form of this estimator is given by the

normalized gradient of one of the four conservation variables. In this case density was chosen :

$$\theta_e = \frac{A_e^{\frac{1}{2}} \max_{i=1,2} \left| \frac{\partial \rho_h}{\partial x_i} \right|}{\rho_h}$$

where : A_e = area of the element.

ρ_h = average value of the density over element Ω_e

3.6 Implementation of the Flux-Corrected Transport method.

An existing FCT-FEM "solver" using quadrilateral elements was used (see [13]). As this "solver" was written with quadrilateral elements in mind, it had to be converted in order to use triangular elements. This involved going through the code and changing all loops and subroutines involving elemental calculations (e.g. number of sides/nodes per element changed from 4 to 3). The common blocks and variables/arrays were then used as the basis for the adaptive part of the code (see Chapter 4). This ensured a convenient way of linking the FCT-FEM "solver" and the adaptive algorithm described in Chapter 6.

CHAPTER 4

MESH REFINEMENT STRATEGY

Many refinement strategies presently involve meshes consisting of quadrilateral elements. These strategies have to use constrained nodes (ie nodes generated by the refinement whose nodal values are constrained by the nodal values of the two nodes on the same element side) . These constraints have to be incorporated when computing a finite element solution and hence result in a more complicated and less efficient code in general. When considering a triangular finite element mesh, however, one can devise refinement methods involving two alternative element divisions which would eliminate the need for constrained nodes. Such a method would thus improve the speed with which the code computed the solution. This is important as the majority of the running time is spent on computing a solution rather than on refining/unrefining the mesh after a specified time interval. This indicates that the more efficient and quicker adaptive codes involve triangular elements and unconstrained nodes.

Taking the above into account, an adaptive triangular element strategy was adopted. Two different element sub-divisions are used in this strategy so certain unsatisfactory refinement/unrefinement arrangements can occur unless particular rules are devised and implemented. For the sake of ease of reading, the type of sub-divisions will be described before the rules governing refinement/unrefinement.

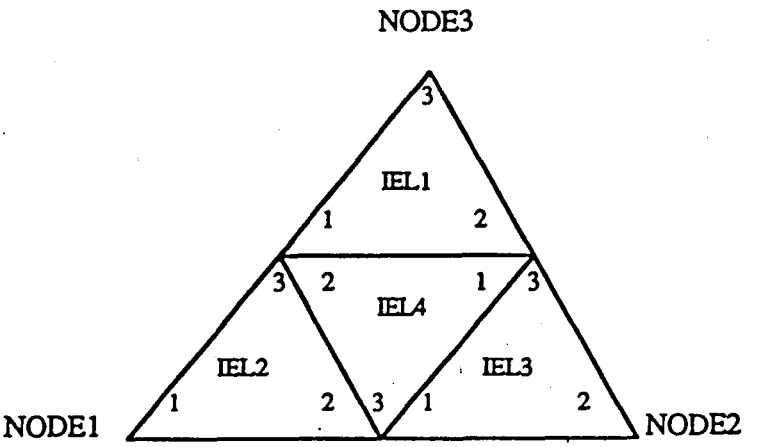
4.1 Types and uses of each division type

The two types of division, following the example described in [1], are known as "regular" division and "green" division. The "regular" division results in the generation of four identical triangular elements which are all geometrically similar to the original "father" element. This node is then connected to the vertex on the opposite side, thus dividing the element into two unsimilar elements.(See Figure 4.1 for a schematic representation)

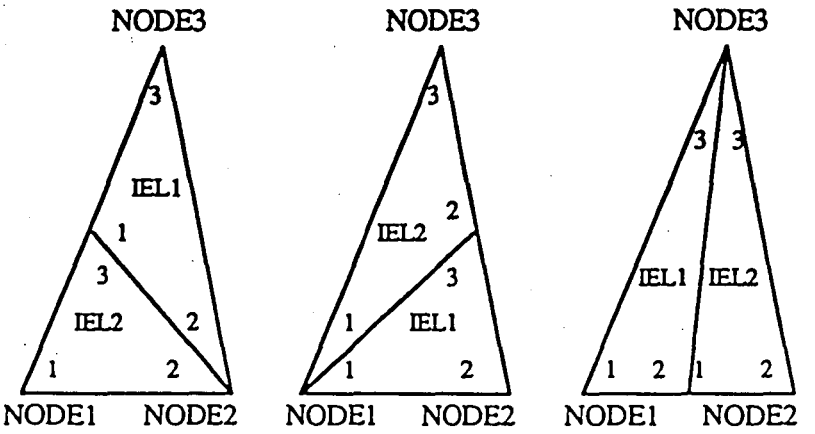
The way in which these alternative divisions are used is as follows:

(1)Based upon some a-posteriori error estimate, "regular" division is carried out throughout the mesh in such a way that there are no other elements with more than one constrained node per side, i.e. after "regular" refinement, the mesh consists of only triangular elements and degenerate quadrilateral elements (degenerate because of the fourth node introduced by refinement-nodes 1-3 are such nodes for the shaded elements in Figure 4.2 a).

(2) "Green" refinement is then carried out throughout the mesh on all of the degenerate quadrilateral elements. This effectively "cleans up" the mesh and yields no elements with constrained nodes (refer to Figure 4.2 b).

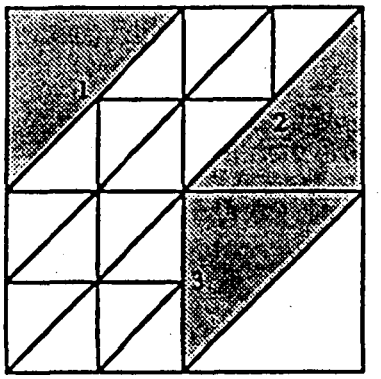


a) "Regular" division

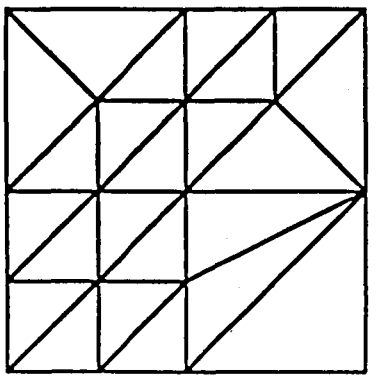


b) "Green" division

Figure 4.1 : Types of division.



a) Mesh after "regular" refinement



b) The mesh after "cleaning up" the degenerate quadrilateral elements

Figure 4.2 : The effects on a mesh of "cleaning up".

4.2 General Rules governing Refinement/Unrefinement

As the refinement/unrefinement must produce reasonably shaped elements without constrained nodes, two general rules can be immediately defined:

(1) The interior angle of each element must be bounded from zero to ensure a reasonable solution ,i.e. long, slender elements are not desirable.

(2) The size difference between neighboring elements must not be such that constrained nodes are produced.

4.3 Rules governing successive refinements

Rule number (1) above prohibited long, slender elements. The combination of both "green" and "regular" divisions, however, can contradict this rule as can be seen in Figure 4.3. These figures show situations which could contradict rule (1) and the solutions to these situations.

Rule (2) dissallowed constrained nodes. Figure 4.4 shows situations which could lead to at least two constrained nodes if care is not taken in refinement. Depending on the element numbering, the "cleaning up" of the mesh (see Figure 4.2) with "green" divisions would reduce these two nodes to one constrained node which is still unsatisfactory.The following logic is used to solve these problems:

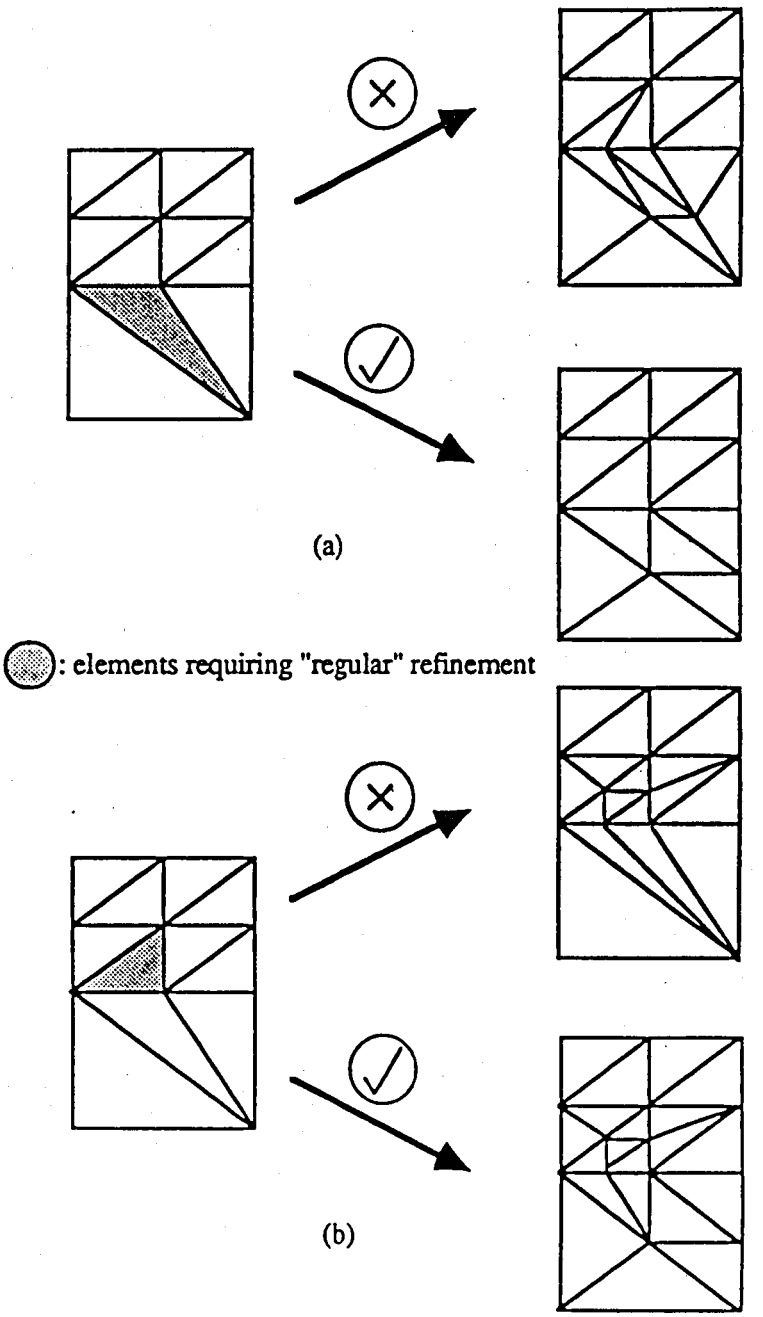
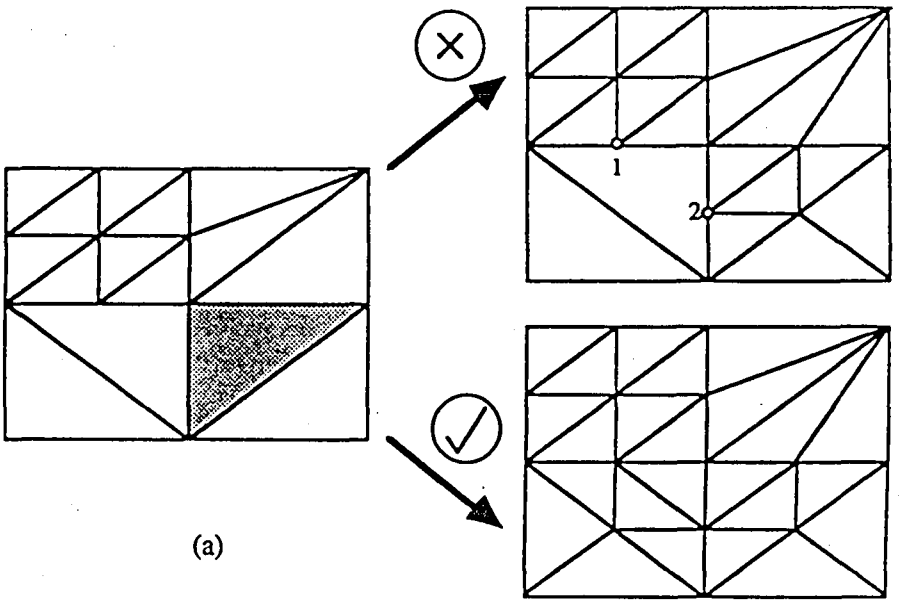
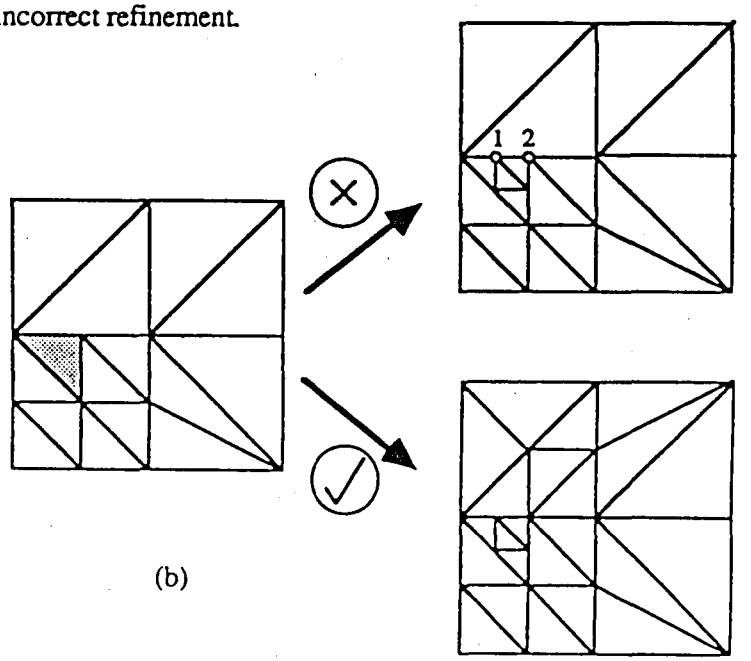


Figure 4.3 Correct and incorrect successive refinements



(a)

o : constrained nodes resulting from incorrect refinement.



(b)

Figure 4.4 Further examples of correct and incorrect successive refinements

NOTE: In the following, NEL1=the element that is to be refined due to a large a-priori error estimate.

NEIG= any neighbouring elements of NEL1

NNEI= any neighbouring elements of NEIG

LEVEL(NEL)= level of refinement of NEL (=0 for a non-refined element)

(1) If NEL1 is a "green" triangle , NEL must be "un-greened" and then"regularly" divided before NEL can be divided(Figure 4.3 a)

(2) If any NEIG is a "green" Triangle, NEIG must be " un-greened" and "regularly" divided bfore NEL1 can be divided(Figure 4.3 b)

(3) If any NEIG has one or more "regularly" divided neighbours(NNEIG), then NEIG must be "regularly" refined before NEL can be dealt with (see Figure 4.4 a).

(4) If LEVEL(NEL) is greater than LEVEL(NEIG), NEIG must be refined "regularly".(Figure 4.4 b)

4.4 Rules governing successive unrefinements.

In the case of unrefinement, the two general rules (see section 4.2) must again be obeyed. If refinement has been carried out in the correct manner, unrefinement cannot cause long, slender elements (rule (1)). Hence rule (1) will automatically be obeyed in unrefinement. Constrained nodes (rule (2)), however, can occur if certain rules are not obeyed. One clear case can be seen in Figure 4.5 where removal of the shaded group would cause two constrained nodes to occur, destroying the continuity of the solution. The following logic is hence used:

NOTE: In the following: NG1= group number requiring unrefinement.
NEIGRP=any neighbouring group of NG1.
NNEIGRP=any neighbouring group of NEIGRP.
"REGULAR" group : group of "regular" elements.
"GREEN" group : group of "green" elements.

(1) If NG1 is a "green" group, unrefinement is not carried out.

(2) If NG1 contains a further group (NG2), group NG2 must be unrefined before NG1 is considered again (see Figure 4.6) .

(3) If two or more of the three NEIGRP's are "green" groups, unrefinement is immediately possible (see Figure 4.7).

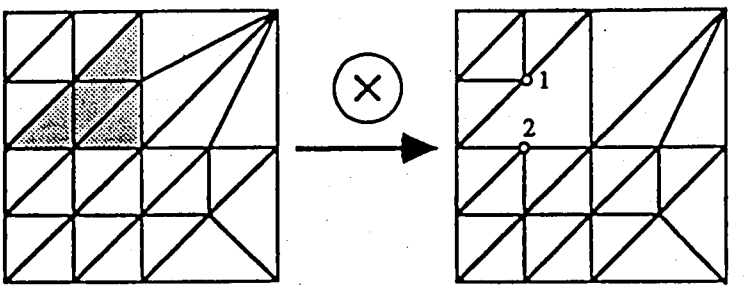


Figure 4.5 Incorrect unrefinement introduces two constrained nodes.

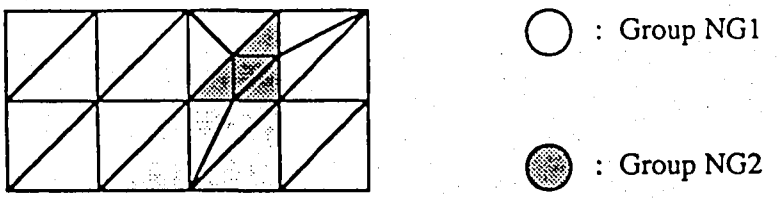


Figure 4.6 Unrefinement of group NG1 which contains group NG2 as a member

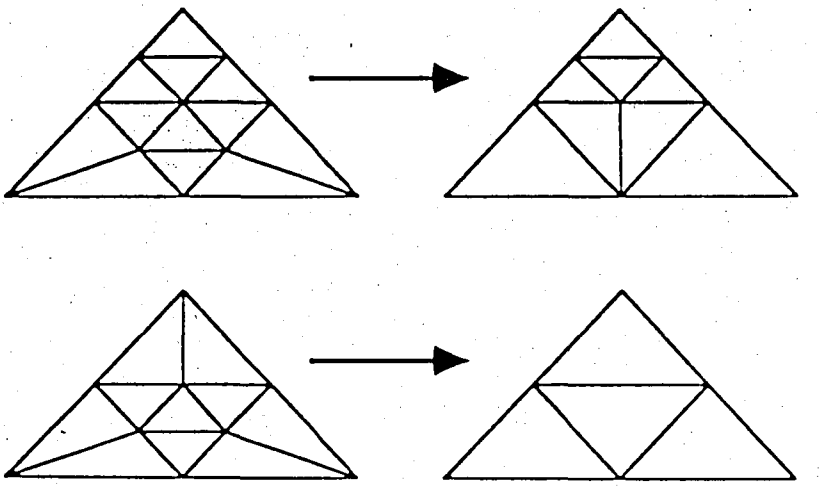


Figure 4.7 Two cases that can be immediately unrefined.

(4) If two or more of the three NEIGRP's are "regular" groups,then the three NNEIGRP's of these NEIGRP's must be examined:

a) If two or more NEIGRP's have one or more "green" NNEIGRP's unrefinement is carried out (see Figure 4.8).

b) If one or more NEIGRP's have one or more "green" groups as neighbours *and* one NEIGRP is a "green" group, unrefinement is possible (see Figure 4.9).

c) If neither situation a) nor b) exists, then unrefinement is deemed impossible.

4.5 Practical implementation of Refinement/Unrefinement

Before describing the subroutines which control refinement and unrefinement, a brief outline of the data structures used is necessary. The data arrays necessary for refinement/unrefinement will be explained in detail as well as the original data structures contained within the finite element "solver".

4.5.1. Data structures used:

The original code (ie the "solver") needs the following data structures to provide a solution:

NELEM= number of elements.

NNODE= number of nodes.

NODES(I,NEL)= Global node number of the I'th node of element "NEL".

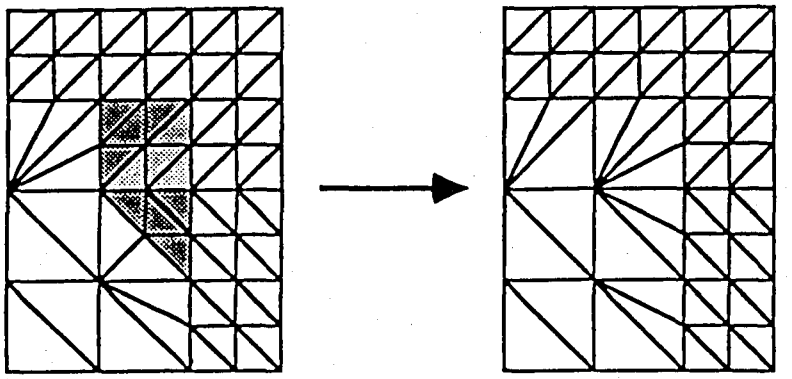


Figure 4.8 Two neighbouring "regular" groups (⊗) have one or more "green" groups as neighbours.

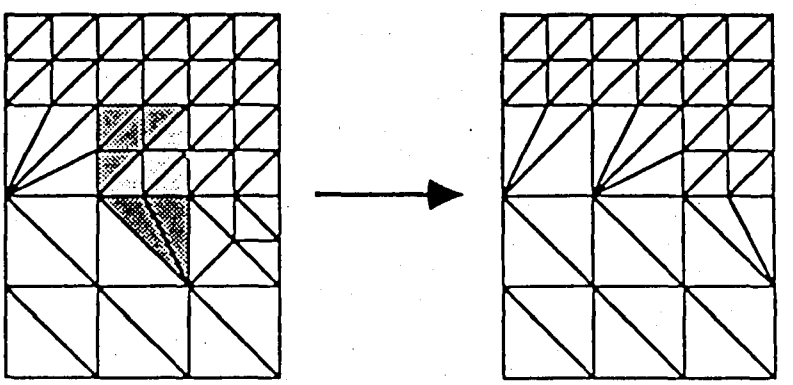


Figure 4.9 Group NG1 has one :

a) neighbouring "regular" group with one "green" neighbouring group(⊙)

b) one neighbouring "green" group(⊗)

X(I,INOD)= I'th coordinate of global node number "INOD".

QTN(I,INOD)= I'th component of the solution for global node "INOD".

The following arrays were introduced for refinement:

NELCON(I,NEL)= I'th connection/neighbor of element "NEL".

LEVEL(NEL)= level of refinement of element "NEL".

ERR(NEL)= an a-posteriori estimate of the local error of the solution in element "NEL".

A brief explanation of the NELCON and LEVEL arrays follows:

(1) NELCON array : this array stores the neighbours of a particular element. The connectivity numbering of an arbitrary element is shown in Figure 4.10. The dimension of NELCON is (6,MAXEL) as each side of the element can be connected to two different elements(MAXEL = maximum number of elements).

(2) LEVEL array : every time an element is refined, the level of the resulting two or four elements ("green" or "regular" division) is increased by one.

For unrefinement to occurr, we need three additional data arrays . One of these is used to store some history of the refinement process and the other as an error indicator for each group of elements generated by refinement (unlike ERR(NEL) which is specified per element).

NELGRP(I,NG)= I'th member of group number NG.

If > 0 : Refers to an element.

If < 0 : Refers to a group of elements.

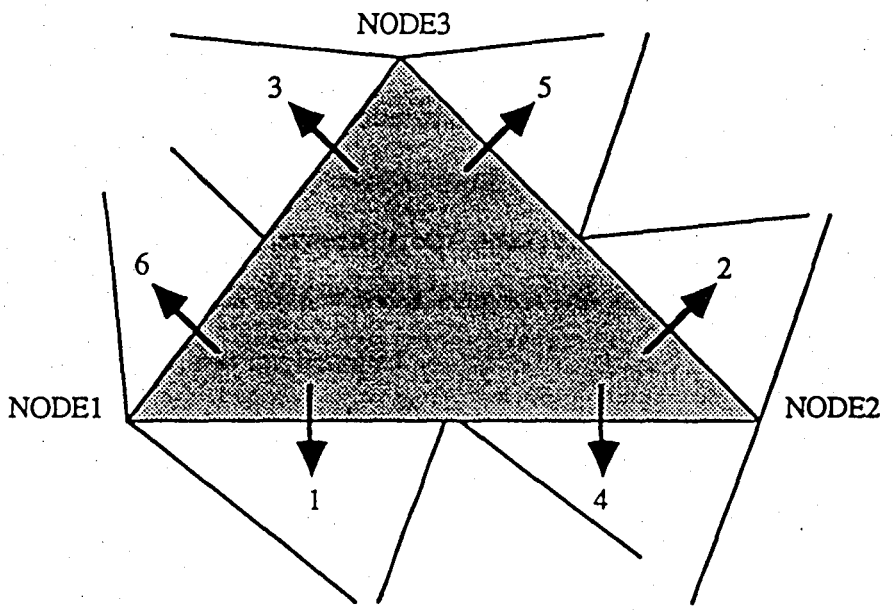


Figure 4.10 The connectivity of an element.

GRERR(NG)= an error estimate for groups of elements generated by refinement (GRERR= the sum of ERR's of each element in the group).

IELGRP(NEL)= the number of the group containing element NEL.

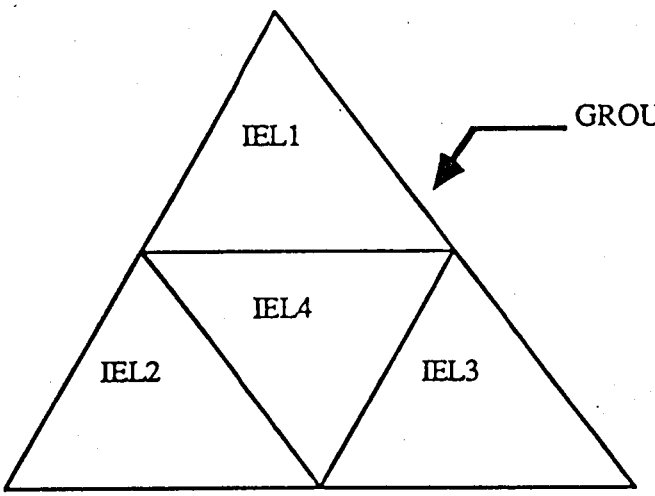
For a schematic representation of the NELGRP array, see Figure 4.11, noting the differences between "green" and "regular" groups. Lastly, in order to control the generation of new nodes, elements and groups, an additional structure obtained from [2] was implemented.

INODFR(I)= an array containing a list of "free" nodes, ie nodes not currently being used, "freed" by unrefinement .

INELFR(I)= an array containing a list of "free" elements caused by unrefinement (as opposed to INODFR(I) which contains "free" nodes) .

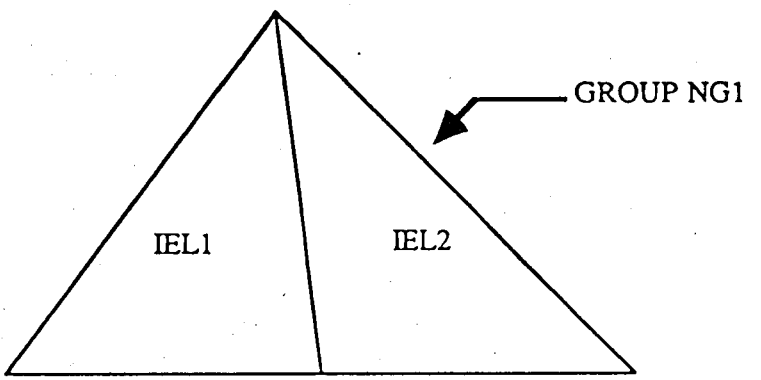
IGRFR(I)= an array containing a list of "free" groups caused by unrefinement.

IELCH(I)= an array containing a list of elements which have either been refined or unrefined and are also still in use (as opposed to "free" elements).



NELGRP(1,NG1) = IEL1
NELGRP(2,NG1) = IEL2
NELGRP(3,NG1) = IEL3
NELGRP(4,NG1) = IEL4

a) "Regular" group



NELGRP(1,NG1) = 0
NELGRP(2,NG1) = IEL1
NELGRP(3,NG1) = IEL2
NELGRP(4,NG1) = 0

b) "Green" group

Figure 4.11 Schematic representation of NELGRP array.

4.5.2 Subroutines controlling Refinement/Unrefinement.

4.5.2.1 Refinement.

Refinement is obtained mainly through the use of subroutines REFINE and DIVIDE(NEL1,NEL2). REFINE decides which elements need to be refined based on our error array ERR(NEL) and provides DIVIDE with these element numbers (see Figure 4.12 for a flowchart). Subroutine DIVIDE then operates as follows:

SUBROUTINE DIVIDE(NEL1,NEL2)

NEL1 : (input)

NEL2 : (output) =NEL1 if NEL1has been been refined.

= NEL0 if NEL0 has to be refined first to enable NEL1 to be refined.All the rules outlined in section 4.3 are implemented in DIVIDE.

Once all the elements have been considered for "regular" refinement, a final loop over the elements (in REFINE) identifies any degenerate quadrilateral elements and "green's" these elements to ensure no constrained nodes. Note that once an element is to be divided, the following steps are necessary:

(1) Generate new element numbers.

(2) Change NELCON array of neighbours.

(3) Generate NELCON array for new elements.

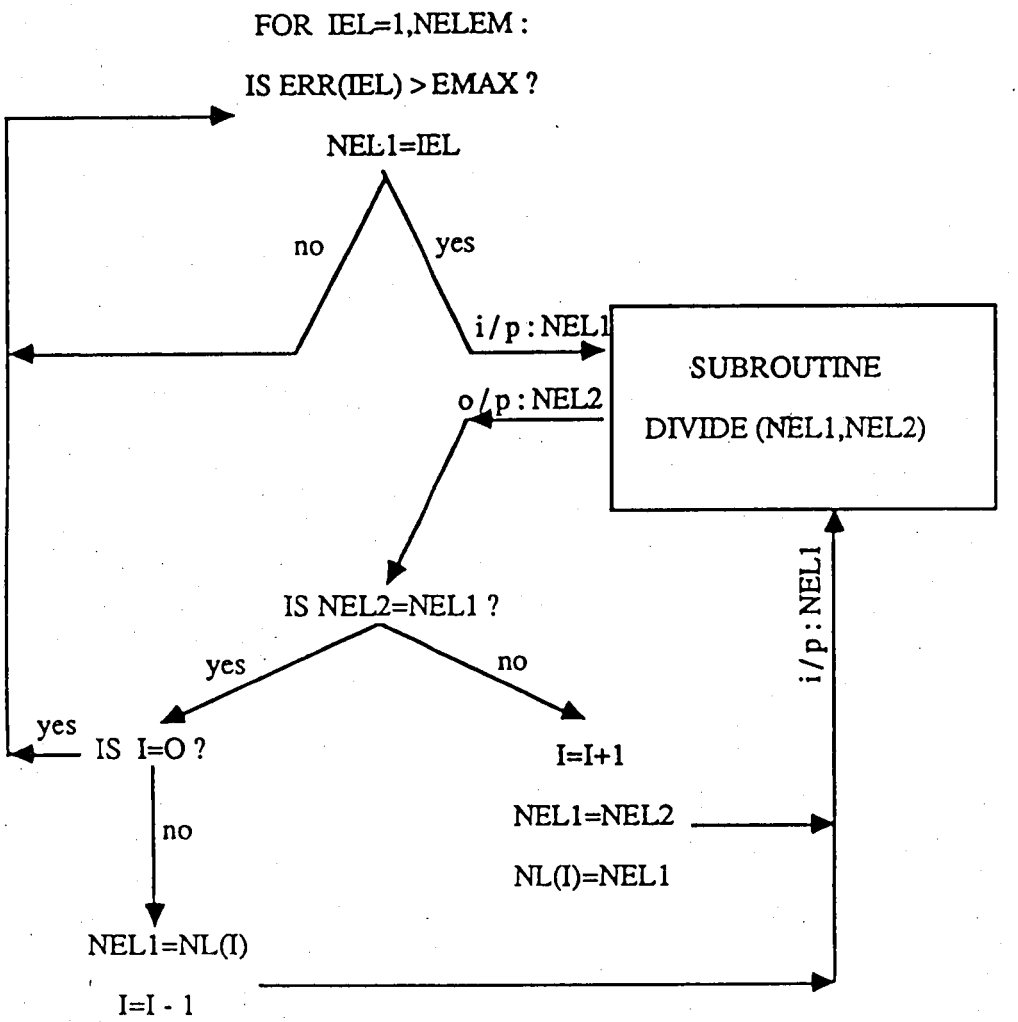


Figure 4.12 Flowchart of how subroutine REFINE calls subroutine DIVIDE

(4) Generate new nodal numbers.

(5) Fill in the NODES array for the new elements.

(6) Interpolate the solution over the new nodes.

4.5.2.2 Unrefinement.

Unrefinement is obtained through the use of subroutines REFINE, UNREFINE(NG1,NG2) and UNDIVIDE(NG) (see Figure 4.13). REFINE again decides which groups require unrefinement (based this time on the group error array GRERR(NG)) and provides UNREFINE with these group numbers as follows. Subroutine UNREFINE then operates as follows:

SUBROUTINE UNREFINE(NG1,NG2)

NG1 : (input) : group to be unrefined.

NG2 : (output) = NG1 if NG1 has been successfully unrefined.

= NG0 if Group NG0 has to be unrefined first in order to unrefine NG1 (see Section 4.4 for unrefinement rules).

= 0 if unrefinement of group NG1 is impossible.

NOTE : that if NG2 has a group error which does not demand unrefinement, then NG1 will not be unrefined although its group error does require unrefinement. This

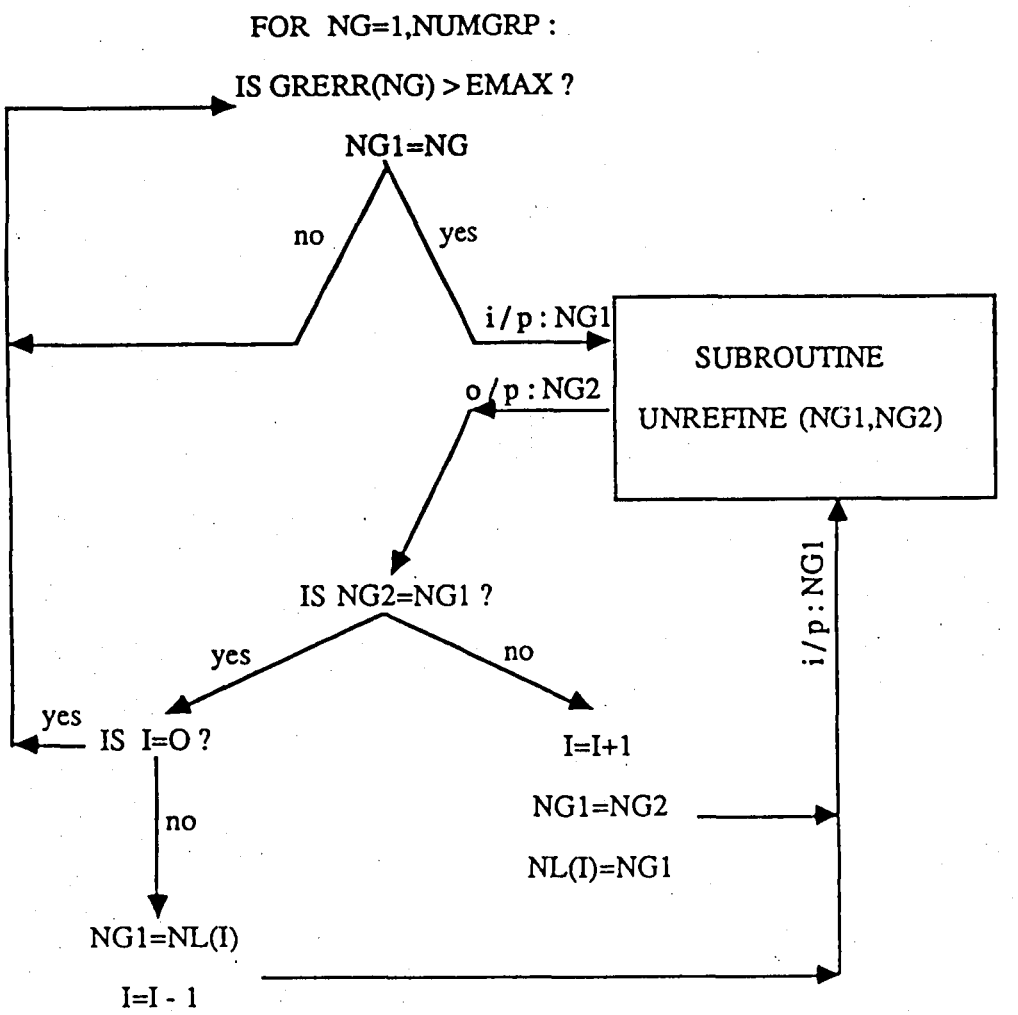


Figure 4.13 Flowchart of how subroutine REFINE calls subroutine UNREFINE

is unlike Subroutine DIVIDE(NEL1,NEL2) which will refine any elements just to divide NEL1.

Again once a group is to be unrefined, the following steps are required:

(1) Select one element number from the group (of two to four elements).

(2) Adapt the NODES array of the neighbouring elements.

(3) Adapt the NODES and NELCON arrays for the selected element.

(4) Add one or three elements to the list of free elements depending upon whether the group was "green" or "regular".

(5) Add the group number to the list of free nodes.

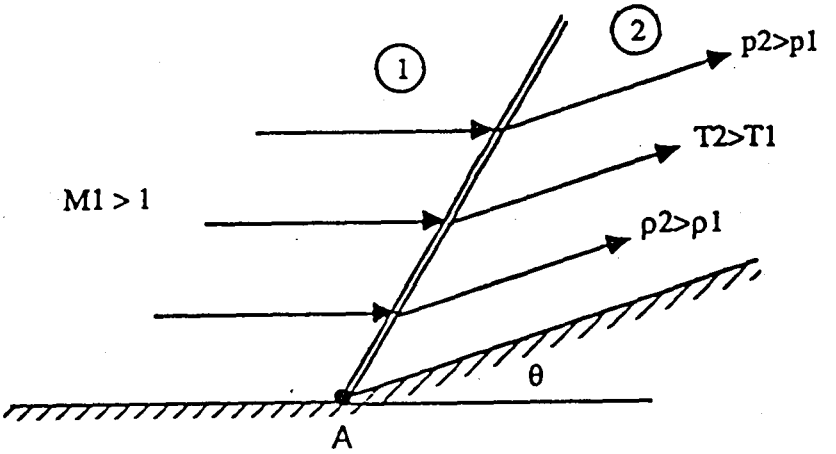
CHAPTER 5

NUMERICAL EXAMPLES

In this section, examples of both steady-state and transient problems in supersonic compressible flow are presented. The Flux-Controlled Transport (FCT) algorithm together with the adaptive refinement/unrefinement algorithm was seen to produce very good results especially in those problems involving shock waves (i.e. line discontinuities) and Prandtl-Meyer expansion waves. In almost all of the examples exact theoretical solutions were available to compare with the computed solutions, allowing direct comparisons. The FCT algorithm was employed in all the problems beside problem 5.2 with a value of c = 0.125.

5.1 Supersonic flow over a 20° concave corner (ramp).

In this case, supersonic fluid flow over a 20° ramp was observed. When supersonic flow is deflected upwards through an angle θ, the flow streamlines have to change direction very abruptly. This takes place across the shock wave which is oblique to the initial flow direction and stems from the point at which the flow is deflected. All the deflections are alike meaning that the flow remains parallel after the shock. Across the shock, the fluid velocity decreases and the density, pressure and temperature all increase. Refer to Figure 5.1 for a sketch together with the initial coarse mesh.



(a) Supersonic flow over a concave corner.

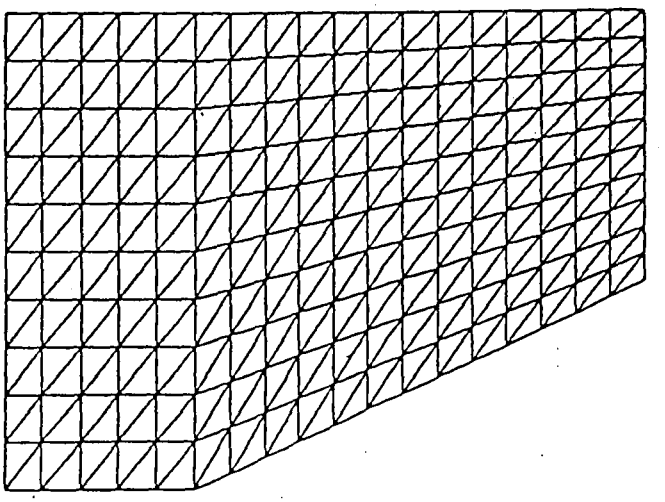


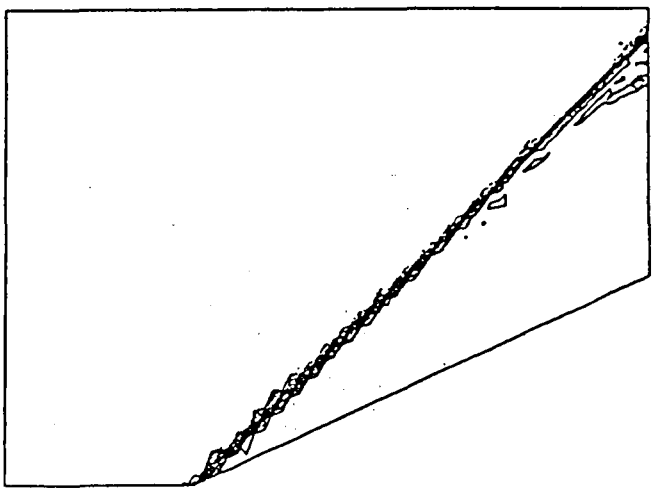
Figure 5.1 Sketch of supersonic flow over a concave corner plus initial mesh.

Two levels of refinement were allowed and the constants controlling refinement/unrefinement were $\beta = 0.20$ and $\alpha = 0.01$, i.e. the error estimate for each element ideally lies between α*EMAX and β*EMAX (EMAX is the maximum element error estimate, selected by looping over all the elements) . Uniform inflow conditions of Mach 3 and γ=1.40 ($\gamma = c_p/c_v$) were specified. After less than 500 time steps, the shock profile had been accurately captured except for some small disturbances which dissapeared after further time-stepping. The solution at this time is shown in Figure 5.2 together with the final refined mesh. The concentrated area of level 2 elements (level 0 = unrefined elements) correlates excellently with the line of the shock. The slight oscillatory nature of the solution is characteristic of the FCT method.

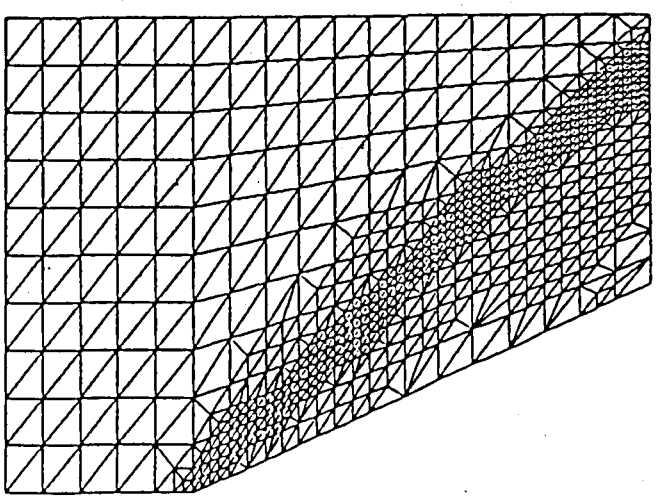
5.2 Supersonic flow over a convex 10° corner.

In contrast to flow over a concave corner (section 5.1), flow over a convex corner results in the fluid being deflected away from itself to remain parallel with the surface. This change in direction is accomplished through an expansion wave which is centered at the point at which the corner begins. The flow streamlines are all uniformly curved by the expansion fan until they are again parallel to the surface. Unlike the discontinuities across a shock wave, the flow properties change smoothly and continuously over an expansion wave. In addition the flow velocity increases whereas density, pressure and temperature all decrease (see Figure 5.3 for a sketch)

Two levels of refinement were allowed together with the following refinement/unrefinement constants γ=1.38 , β=0.20 and α=0.01. The inflow

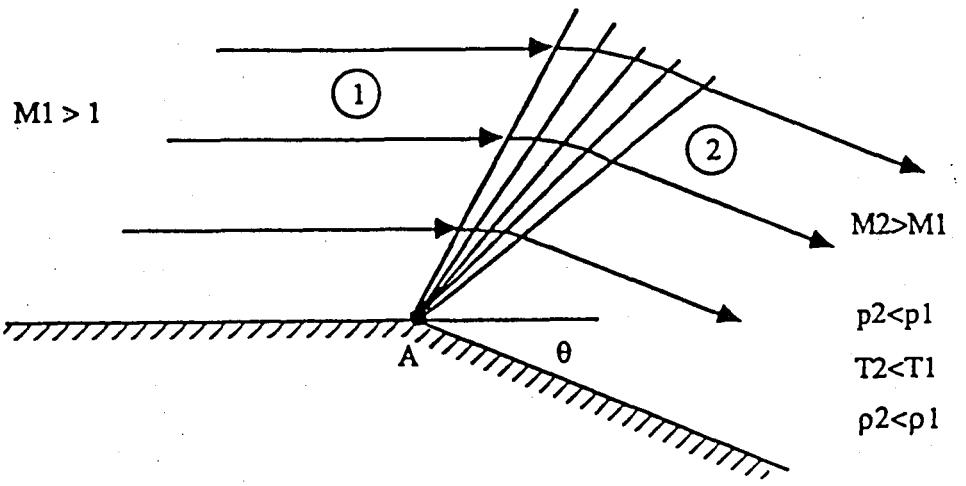


(a) Final computed solution

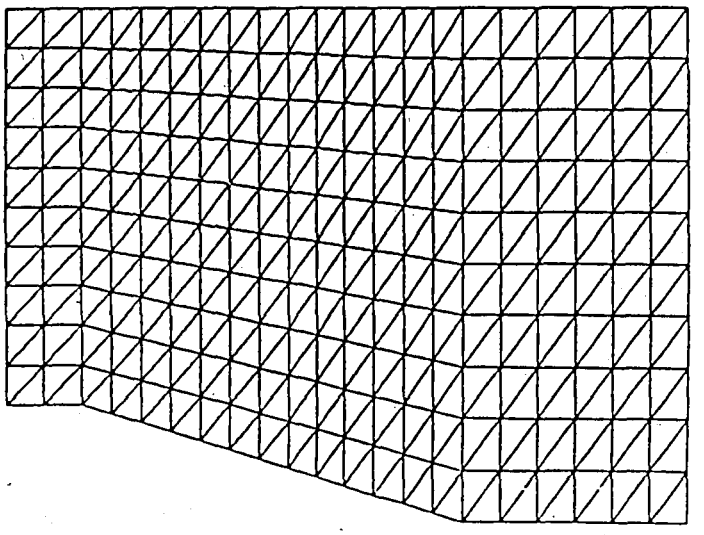


(b) Final refined mesh (maximum level of refinement =2)

Figure 5.2 Final solution and the corresponding refined mesh.



(a) Supersonic flow over a convex corner.



(b) Initial mesh

Figure 5.3 Sketch of an expansion fan and the initial coarse mesh.

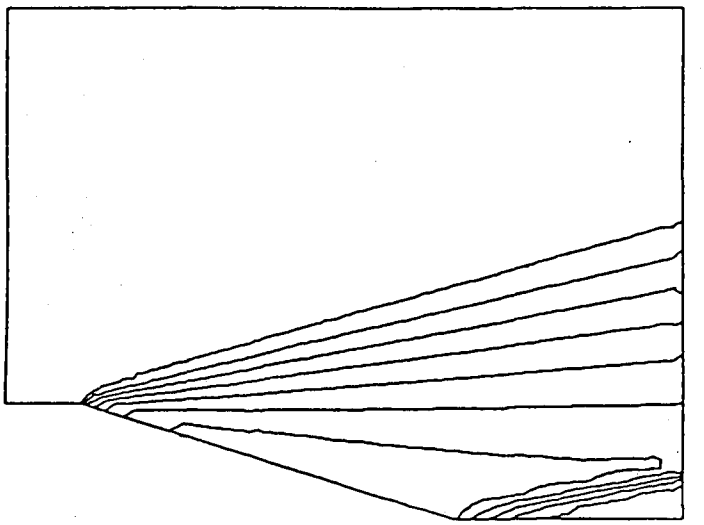
conditions were specified as a uniform Mach 6 flow. Various values of c were allowed ranging from c= 0.0 to 1.0 . The difference in the amount of diffusion correlates to the value of c as can be observed from Figure 5.5. The final solution (c=0.125) and the corresponding refined mesh are shown in Figure 5.4.

Again the results compared very well with the exact solution. The area of the expansion fan is made up from level 2 elements as expected. Because of the high-speed inflow conditions, a shock wave is generated at the second (concave) corner after the expansion wave. This is as expected remembering that the fluid velocity increases through an expansion wave, ensuring continued supersonic (if not hypersonic) flow.

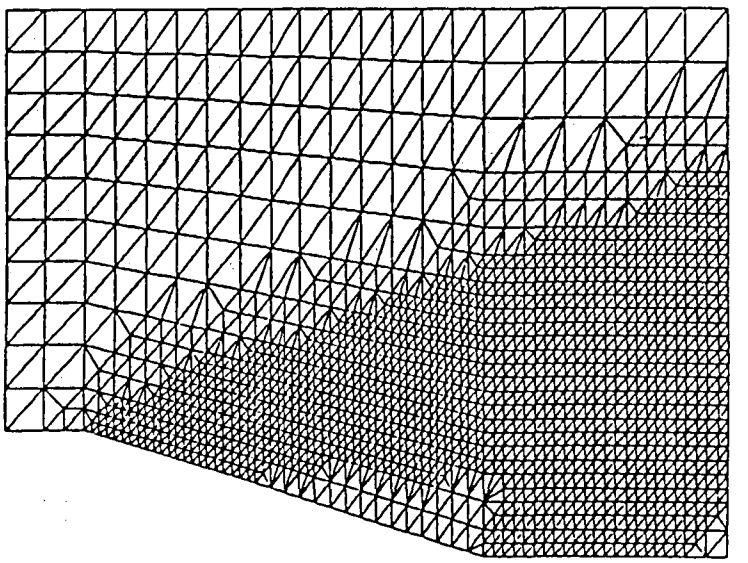
5.3 Intersection of two shock waves of the same family.

A sketch of the double-ramp problem (appropriate to this class of problems) is given in Figure 5.6 as is the initial coarse mesh. The supersonic inflow conditions are specified so as to generate two different oblique shock waves (similar to problem 5.1) , one at A and one at B. Shock wave BC, because of the increased ramp angle at point B, will be inclined at a steeper angle than shock wave AC. Hence the two shock waves intersect at point C resulting in the propagation of a single shock CD.

Now the flow direction and pressure in region 3, p_3 and θ_3, result from the upstream conditions in area 2.Likewise p_2 and θ_2 are a result of upstream conditions in area 1, so p_3 and θ_3 are affected by both shocks AC and BC. Properties in area 5



(a) Final computed solution



(b) Final refined mesh

Figure 5.4 Final solution and the corresponding mesh.

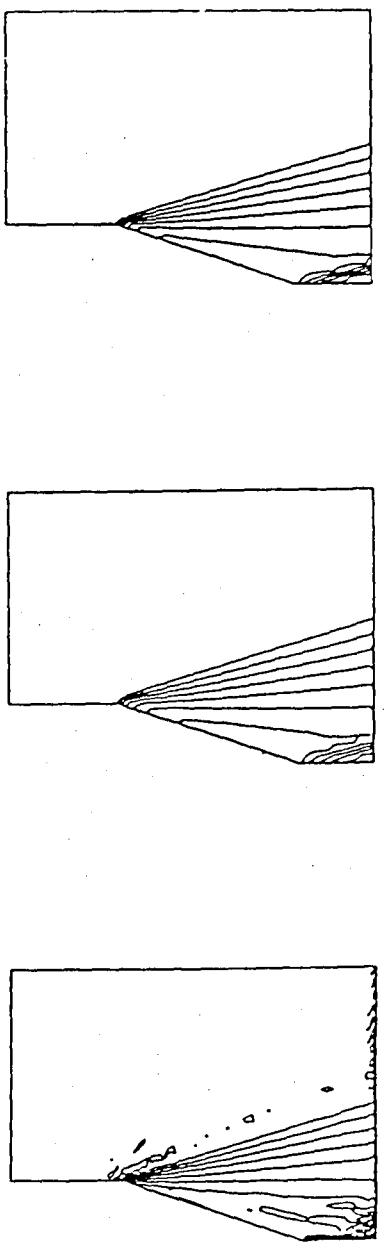
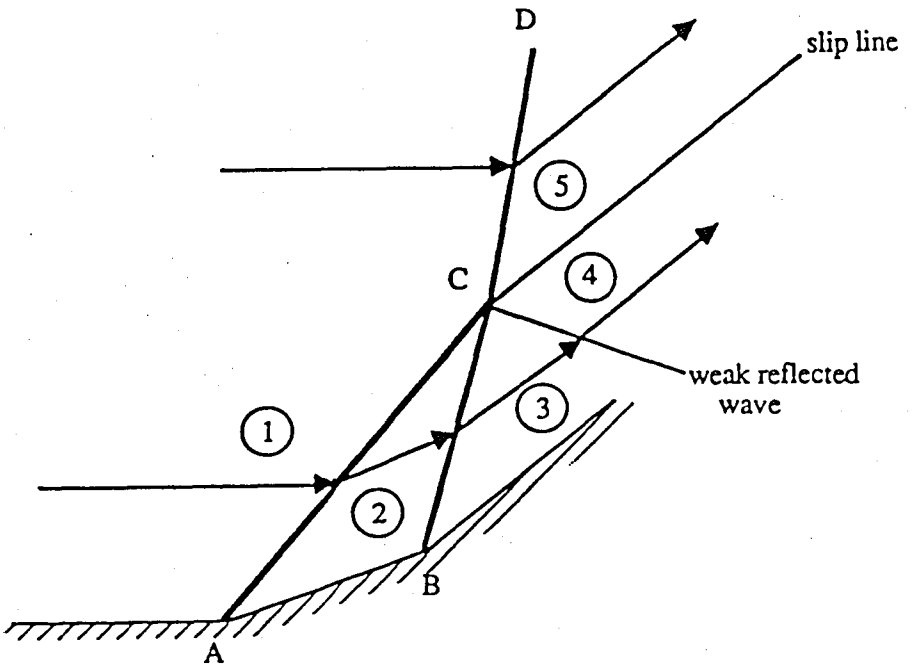
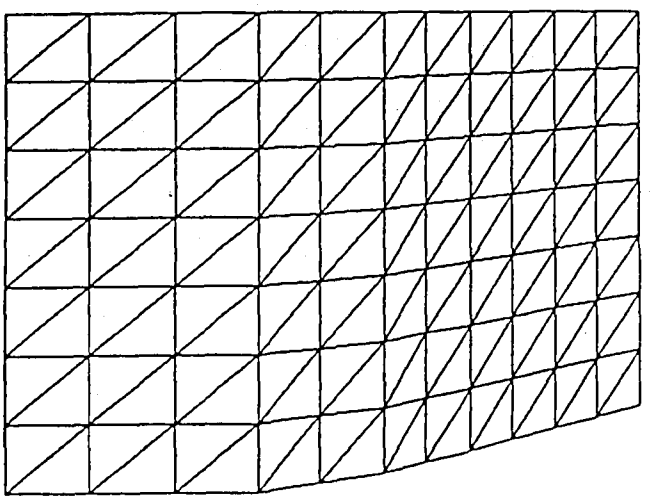


Figure 5.5 Examples of expansion waves corresponding to different values of "c".



(a) Intersection of two shocks generated by a double ramp.



(b) Initial coarse mesh

Figure 5.6 Sketch of two shocks intersecting and the initial mesh.

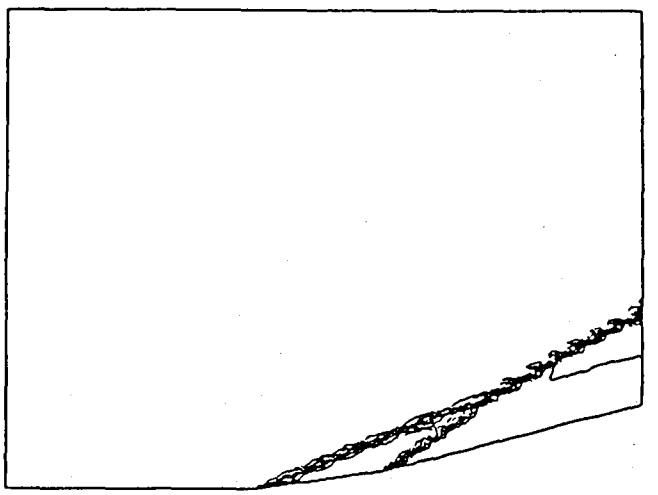
(p_5 and θ_5), however, are processed by a single shock CD. As the entropy changes across the single shock (CD) and the two shocks (AC and BC) will be different, a slip line must originate from point C (ie a line across which the pressures and flow directions are equal). To get $p_5=p_3$ and $\theta_5=\theta_3$ simultaneously is virtually impossible and hence a weak reflected wave is generated from point C (either a weak shock or an expansion wave). All this wave really does is to ensure that $p_4=p_5$ and $\theta_4=\theta_5$, satisfying all shock relations.

Three levels of refinement were specified with γ=1.38, β=0.20 and α=0.01. The inflow conditions were uniform Mach 5 flow with c=0.125.

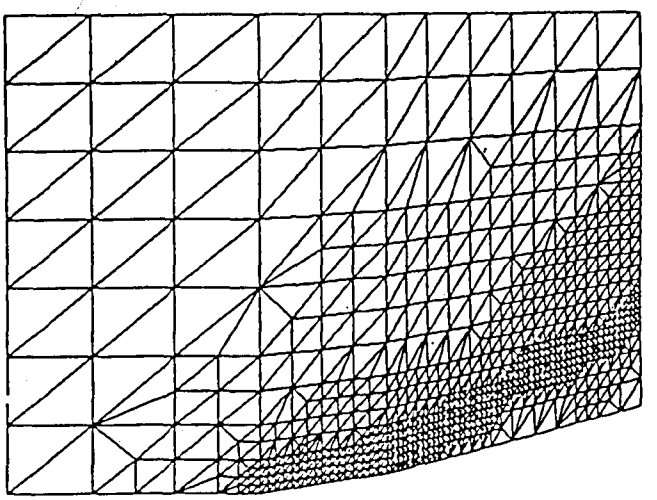
The results as can be seen in Figure 5.7 correspond excellently to the expected appearance (that contained in Figure 5.6). Two shock waves are generated and intersect, combining to form a third wave. A weak shock wave can be seen to originate from point C to satisfy all physical relations between area 1 and area 5, and between areas 1, 2, 3 and 4. The area of most refinement is concentrated about all four shock waves (three strong waves and one weak wave) as was expected (see Figure 5.6).

5.4 Shock Reflection problem.

This problem is generated by specifying initial conditions corresponding to four different triangular areas. It is important that these initial values of density, momentum, and energy satisfy the Rankine-Hugoniot jump conditions over a shock



(a) Final computed solution for the double-ramp problem



(b) Final refined mesh

Figure 5.7 Final solution and the final mesh for double-ramp problem

wave. Although this problem might seem a little contrived, the solution is identical to, say, that of a conventional supersonic diffuser. In this case, the domain is a lot less complicated (see Figure 5.8).

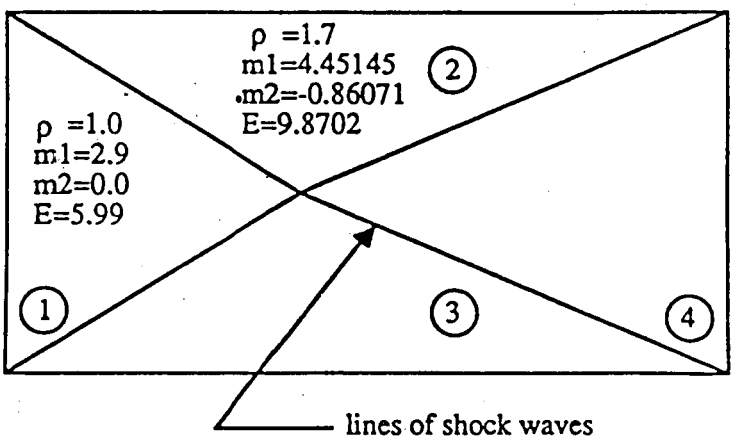
The inflow conditions correspond to a Mach number of 3.5 and γ=1.4. The refinement/unrefinement parameters are again β=0.20 and α=0.01.

Figure 5.9 is a plot of the interpolated initial conditions and the mesh that results from initial refinement (before time-stepping has begun). The computed solution (after 500 time-steps) is then shown with a further refined mesh in Figure 5.10. One can observe that more elements have resulted from additional refinement. The mesh compares favourably with those obtained in [12] as does the converged solution. The FCT algorithm again gives the solution an oscillatory appearance which does not detract from the overall appearance of the solution.

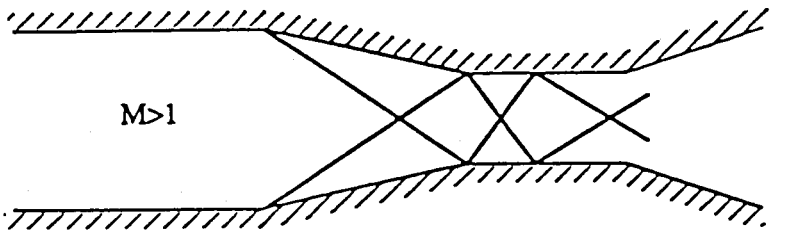
5.5 Supersonic flow over a step.

This problem is analagous to flow in a wind tunnel or a long tube over a small step (in this case the step height was a quarter of the tube diameter). The mesh and solution plots have all been intentionally scaled incorrectly for reasons of detail. The initial mesh in both scales is given in Figure 5.11 to illustrate this.

The inflow conditions were specified as Mach 3 and γ=1.4 on the left vertical mesh edge (the right vertical edge was specified as outflow). All other edges correspond to no-flow conditions.

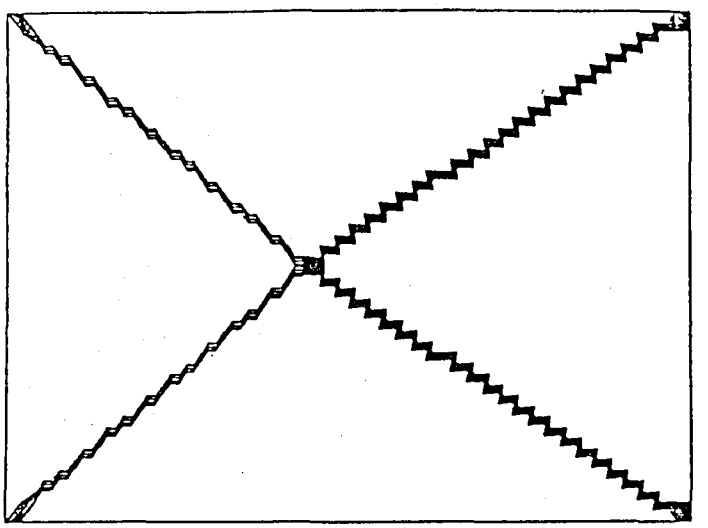


(a) Initial conditions

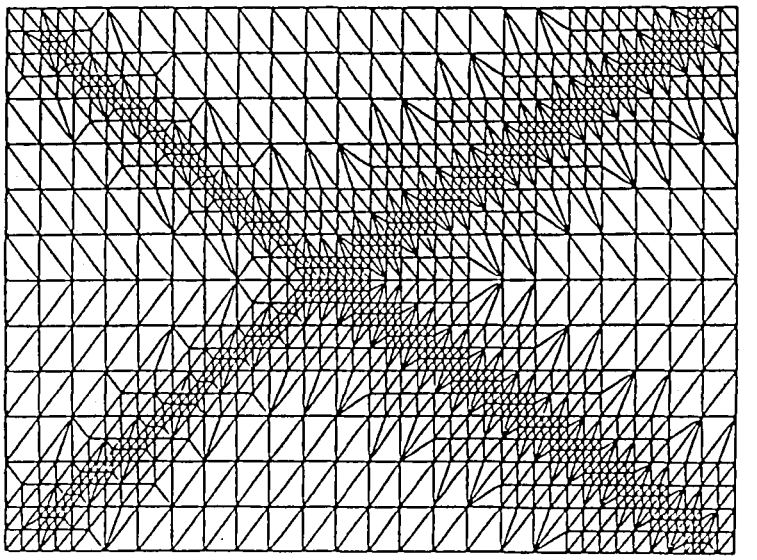


(b) Reflected waves inside a diffuser

Figure 5.8 Initial conditions for reflected wave problem and a physical example.

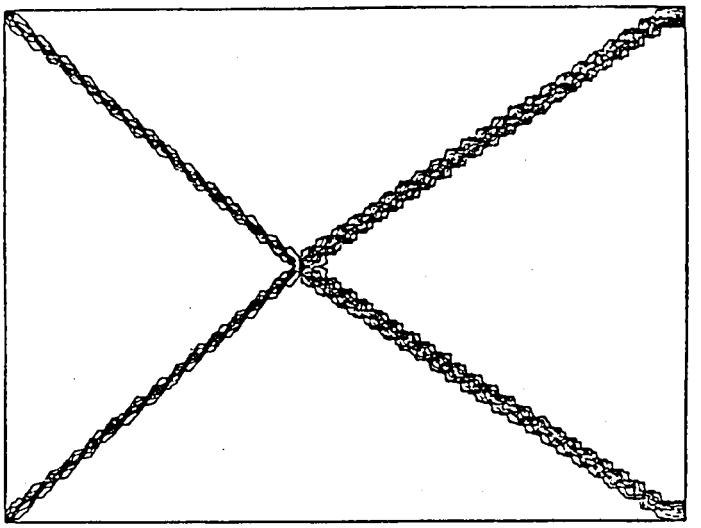


(a) Initial interpolated conditions

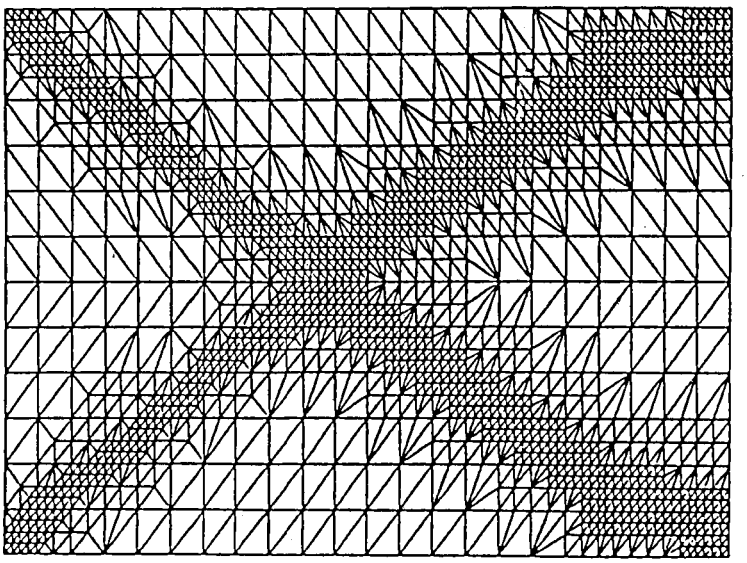


(b) Initial refined mesh (time = 0 sec.)

Figure 5.9 Initial conditions and mesh for reflected wave problem.

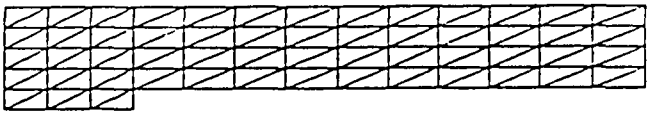


(a) Final computed solution (nsteps = 500)

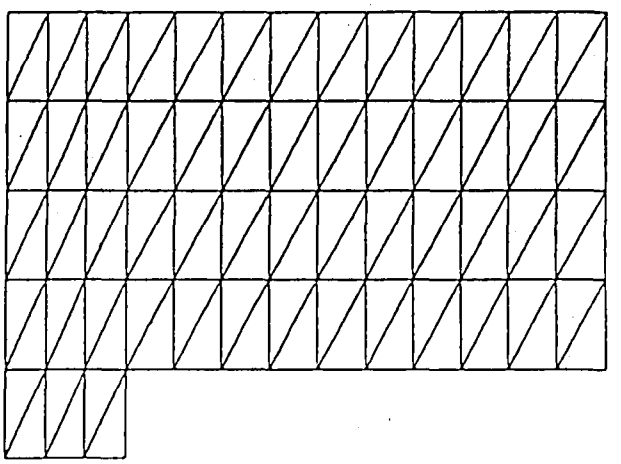


(b) Final refined mesh (time = tfinal)

Figure 5.10 Final solution and mesh for reflected wave problem.



(a) Equal scale

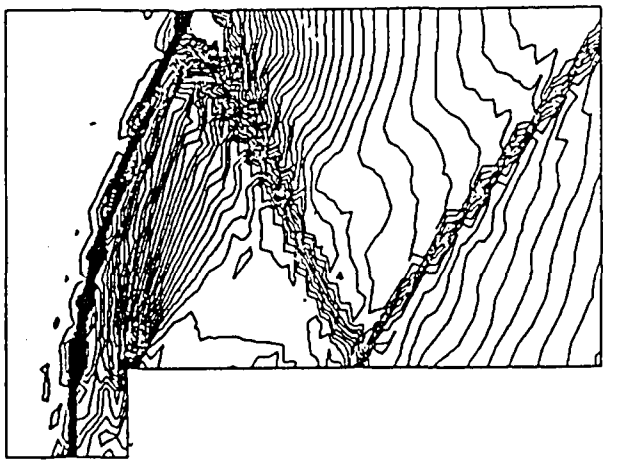


(b) Unequal scale

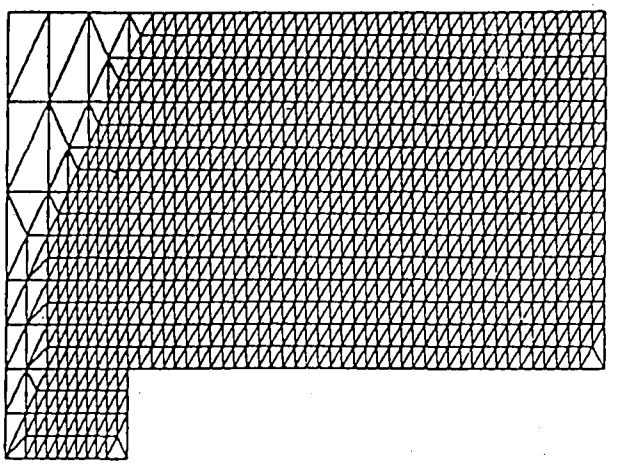
Figure 5.11 Initial meshes (equal and unequal scales).

The solution to this problem involves several shocks/ shock interactions and took 1000 time-steps to converge. The problem was run for two cases of different MAXLEV (= maximum level of refinement for an element). The final converged solutions and the corresponding refined meshes are included in Figures 5.12 and 5.13.

Figure 5.12 (MAXLEV=2) shows a similar shape solution to that contained in Figure 5.13 (MAXLEV=3) although the shock's resolution is better defined and clearer in Figure 5.13. The basic form of the solution involves an upstream bow wave caused by the step. This wave then reflects off first the upper surface then the lower before exitting. A Prandtl-Meyer expansion wave is generated at the corner of the step. The areas of large error (corresponding to the areas contining the shocks) are again well "covered" by the smaller (more refined) elements. Because of the complexity of the solution, the oscillatory nature of the FCT method is clearer especially in the second and third (reflected) waves.

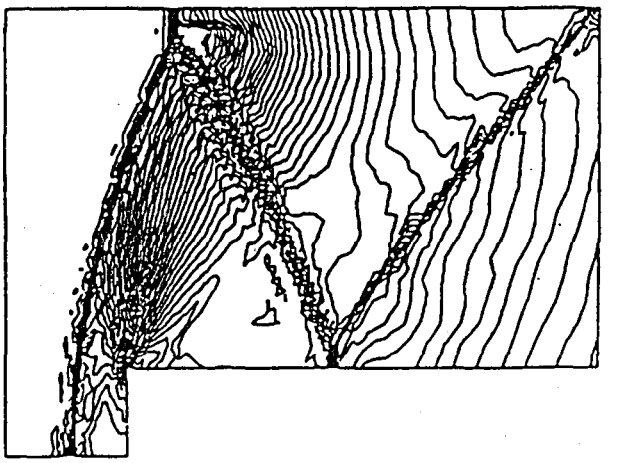


(a) Final computed solution

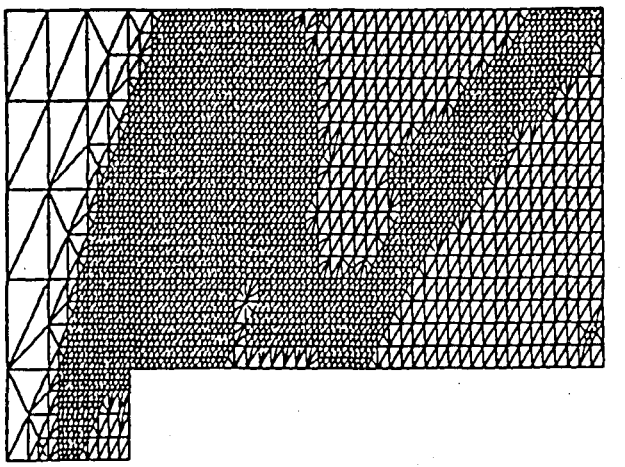


(b) Final refined mesh

Figure 5.12 Final solution and corresponding refined mesh (MAXLEV=2).



(a) Final computed solution



(b) Final refined mesh

Figure 5.13 Final solution and corresponding refined mesh (MAXLEV=3).

CHAPTER 6

RECOMMENDATIONS AND CONCLUSIONS

The main aim of this report was to provide a code capable of providing high-resolution solutions to a variety of problems involving strongly unsteady compressible flow. This was to be done with the minimum number of elements necessary.

As far as these two objectives are concerned the code performed very well, providing both accurate pictures of shocks and shock interaction as well as optimal meshes (due to the refinement/unrefinement capabilities of the code). The combination of the FEM-FCT numerical scheme and adaptivity was seen to be well suited to the class of problems of analysis. All the solutions were true to the calculated or experimentally-obtained exact solutions.

The success of this scheme would seem to prompt the development of a three-dimensional code. There are several references to the multi-dimensionality of the Flux-Controlled Transport method (see References [14],[10]). The types of refinement/unrefinement could also be generalized to three-dimensions. In the case of "green" divisions (see Chapter 4), the three-dimensional equivalent would generate four tetrahedral elements. Similarly, "regular" division would cause the generation of eight geometrically similar (to the parent element) tetrahedra. The

combination of these two divisions would result in three-dimensional meshes with no constrained nodes, again indicating possible time savings in the solution computation. Three-dimensional solutions to the integral form of the Euler equations would have great use and potential, providing information about the effects of surface shocks in three dimensions.

BIBLIOGRAPHY

[1] Baker, A.J., and Kim, J.W., "A Taylor Weak-Statement Algorithm for Hyperbolic Conservation Laws", Technical Report, CFDL 86-1, Comp. Fluid Dyn. Lab., University of Tennessee, 1986.

[2] Bank, R.E., and Sherman, A.H., "A Refinement Algorithm and Dynamic Data Structure for Finite Element Meshes", Centre for Numerical Analysis Report, University of Texas, October 1980.

[3] Devloo, Phillipe, Oden, J.T., and Strouboulis, T., "Implementation of an Adaptive Refinement Technique for the SUPG Algorithm", Computer Methods in Appl. Mech. and Eng., Vol. 61, pp 339-358, 1987.

[4] Devloo, P., "An H-P Adaptive Finite Element Method for Steady Compressible Flow", Ph.D. dissertation, The University Of Texas at Austin, August, 1987.

[5] Donea, Jean, "A Taylor-Galerkin Method for Convective Transport Problems", Int. J. for Num. Meth. in Eng., Vol. 20, pp. 101-119, 1984.

[6] Jameson, A., and Baker, T. J., "Euler Calculations for a complete aircraft", Proceedings of the 10th International Conference on Numerical Methods in Fluid Dynamics, Beijing, China, June 1986.

[7] Jameson, A., and Baker, T. J., "Improvements to the Aircraft Euler Method", AAIA Publication, 1987.

[8] Lohner, R., "The Efficient Simulation of Strongly Unsteady Flows by the Finite Element Method", American Institute of Aeronautics and Astronautics, 25th Aerospace Sciences Meeting, Reno, Nevada, Jan. 12-15, 1987.

[9] Lohner, R., Morgan, K., Vahdati, M., Boris, J. P. and Book, D.L., "FEM-FCT: Combining Unstructured Grids with High Resolution", Submitted to J. Comp. Physics, 1986.

[10] Lohner, R., "FEM-FCT and Adaptive Refinement Schemes for Strongly Unsteady Flows",

[11] Oden, J.T., Strouboulis, T., and Devloo, P., "Adaptive Finite Element methods for the Analysis of Inviscid Compressible Flow : Part 1.Fast Refinement/Unrefinement and moving mesh methods for unstructured meshes", TICOM Report 86-3, The University of Texas at Austin, 1986.

[12] Oden, J.T., Strouboulis, T., Devloo, P., and Howe, M.T., "Recent Advances in Error Estimation and Adaptive Improvement of Finite Element Calculations", Computational Mechanics, Advances and Trends, Ed. A.K. Noor, ASME, pp. 369-410, 1986.

[13] Strouboulis, T., "Adaptive Finite Element methods for Flow Problems with Moving Boundary", Ph.D. dissertation, The University of Texas at Austin, August, 1986.

[14] Zalasak, S.T., "Fully Multidimensional Flux-Corrected Transport Algorithm for Fluids", Journal of Computational Physics, Vol. 31, pp. 335-362, 1979.

Appendix E

IMPLEMENTATION OF AN ADAPTIVE STRATEGY USING THE GIM/PAGE CODE

Appendix E

This appendix documents the implementation of an adaptive strategy using Lockheed-Huntsville GIM/PAGE code as the flow solver. A concise and complete discussion of GIM/PAGE methodology and its application to practical flow problems can be found in the work of Spradley et al.*

The implementation of the adaptive strategy using the GIM/PAGE code is possible because the adaptive strategy does not have to be used with any particular solution algorithm. For this reason, many existing CFD codes can be made adaptive without a major effort. The CFD code must be able to properly treat "constrained" nodes which will exist along the interface between the different levels of refinement. If this capability is not present it must be added. Routines must be added to transfer information between the CFD code and the adaptive strategy.

A general adaptive strategy for the computation of steady-state solutions of the equations of compressible gas dynamics involves the following steps:

1. For a given mesh, compute the steady-state solution.
2. Compute the local error for the mesh.
3. Survey the error field and determine where mesh restructuring is needed.
4. Refine or coarsen the mesh as needed.
5. Go to step 1.

The general tendency is to combine the adaptive strategy and flow solver into one large, complex computer program. However, a closer examination of the general adaptive procedure suggest that the development of another large

*Spradley, L.W., Stalnaker, J.F., Robinson, M.A., and Xiques, K.E., "Finite Element Algorithms for Compressible Flow Computation on a Supercomputer," Finite Elements in Fluids (eds R.H. Gallagher, G. Carey, J.T. Oden and O.G. Zienkiewicz), Vol. 6, 1985.

computer program can be avoided. The Lockheed-Huntsville PAGE code can already perform step 1 of the adaptive strategy. Furthermore, steps 2 through 4 do not require the resources of a supercomputer. They could be executed using the less expensive resources of a front-end machine or workstation.

This adaptive strategy using the GIM/PAGE code has been implemented through a Solution-Adaptive Analysis System (SAAS). This system operates in a computational environment consisting of a supercomputer and a front-end machine. The SAAS concept is shown in Fig. E-1. The adaptive processor and the adaptive database together with pre- and post-processors used to transfer data to and from the PAGE code reside on the front-end side of the SAAS environment. The PAGE code is used on the supercomputer side of SAAS. This segregation of the mesh restructuring from the flowfield computation will allow SAAS to be used with other existing CFD codes. Only custom pre- and post processors need to be developed to facilitate data transfer between the adaptive processor and the CFD codes. Currently, only pre- and post-processors which interface with the PAGE code are available.

The SAAS performs solution-adaptation through its adaptive processor. Utilizing the current geometry and solution, the adaptive processor first assesses the solution quality over the entire domain. It then refines or coarsens the mesh as required based on user-supplied refinement and coarsening tolerances. When refinement (coarsening) is indicated, the physical grid spacing in each of the computational coordinate directions is halved (doubled), the local nodal connectivity is reconfigured, and all geometric and flowfield variables are updated. Flow field quantities at grid points which fall on the boundaries between refined and non-refined elements are constrained to values determined by linear interpolation between the associated connected nodes. The pre- and post-processors convert the geometry and solution data to and from PAGE code format. The PAGE code is then used to compute a new solution using the adapted geometry. The SAAS work environment creates no computer resource penalty during the actual flowfield integration. This is because the grid refinement is removed from the integration process and is performed on a less expensive workstation or front-end computer.

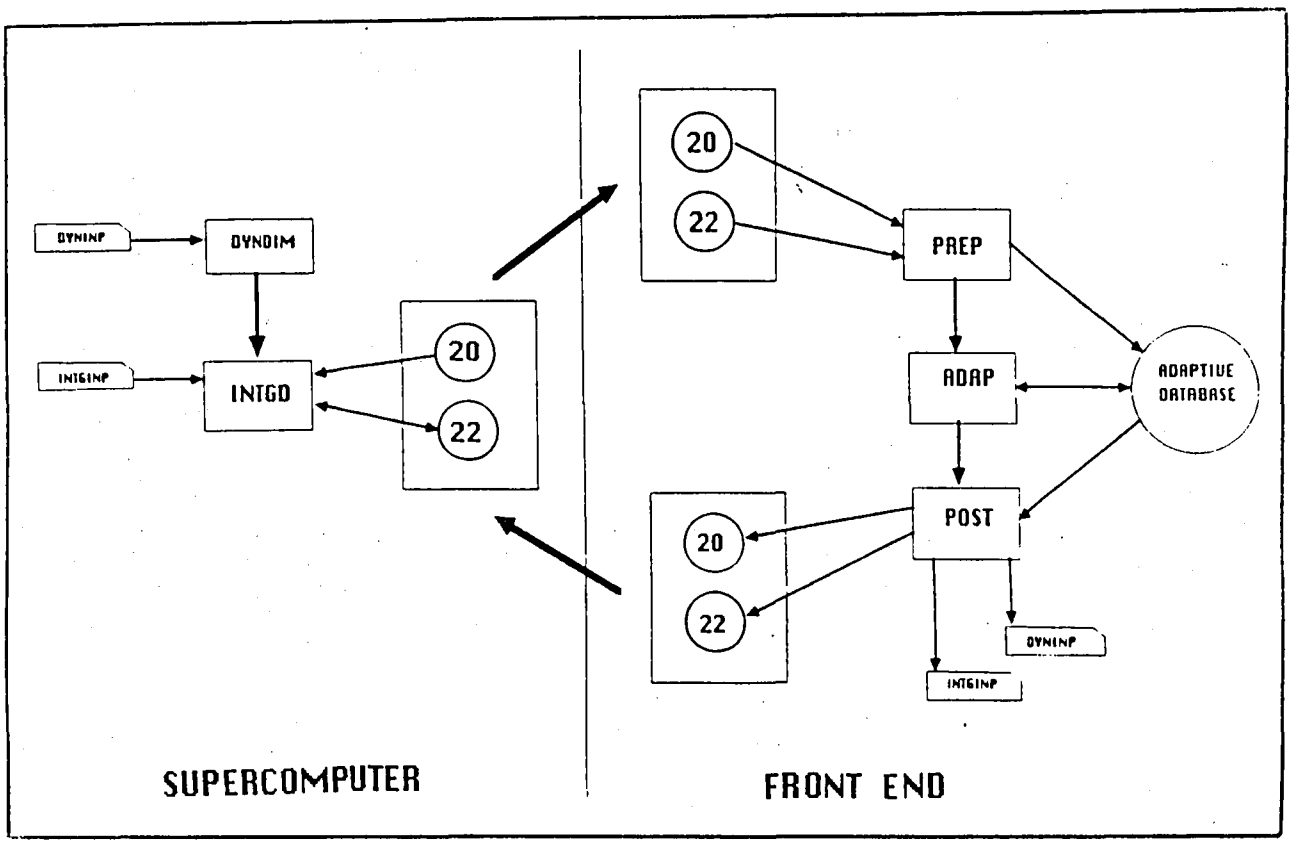


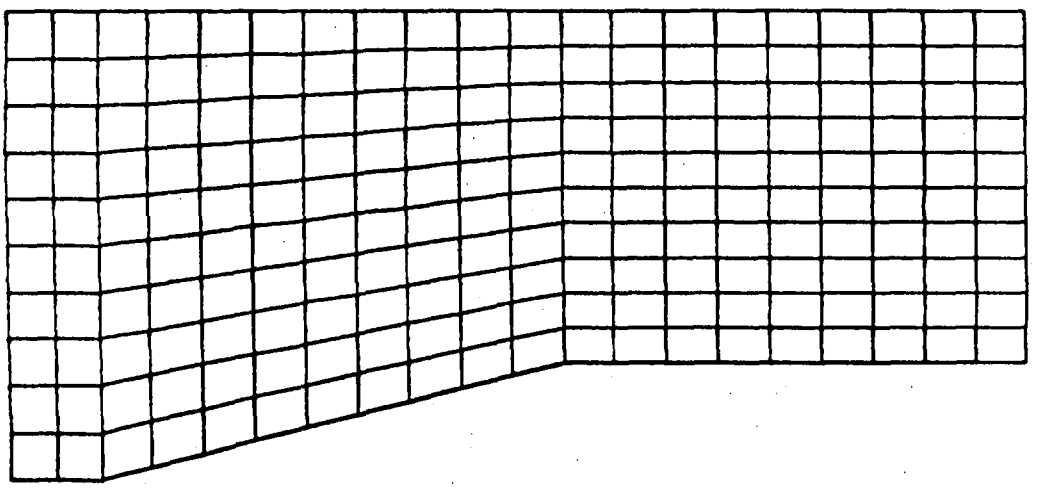
Fig. E-1 Self-Adaptive Analysis System

The following test cases were run to determine the utility of the SAAS adaptive work environment. The refinement threshold was held at 0.65. First, the flow of an ideal Mach 2.4 freestream up a 14.04 deg ramp was computed on a relatively coarse grid using the PAGE code. The grid and resulting solution are shown in Fig. E-2. SAAS was used to adapt the grid and compute an enhanced solution using the PAGE code. The adapted grid and the subsequent PAGE code solution are shown in Fig. E-3.

Second, a viscous shock-expansion test case was run on the same configuration and with the same inflow. A Reynolds number of 1000 was used. SAAS was used to adapt the mesh four times. The PAGE code was used to compute an intermediate solution between each adaptation. Figure E-4 shows the final grid and subsequent PAGE code solution.

Figures 5 and 6 show an inviscid shock reflection calculation. The conditions were the same as the previous cases with a solid upper boundary. This case was run to test the ability of SAAS to capture and refine multiple interactions of shocks and expansions. The mesh shown in Fig. E-5 was refined twice yielding the pressure results shown in Fig. E-6.

PAGE CODE GRID



2-D SHOCK-EXPANSION TEST CASE

Fig. E-2 Two-Dimensional Shock-Expansion Test Case

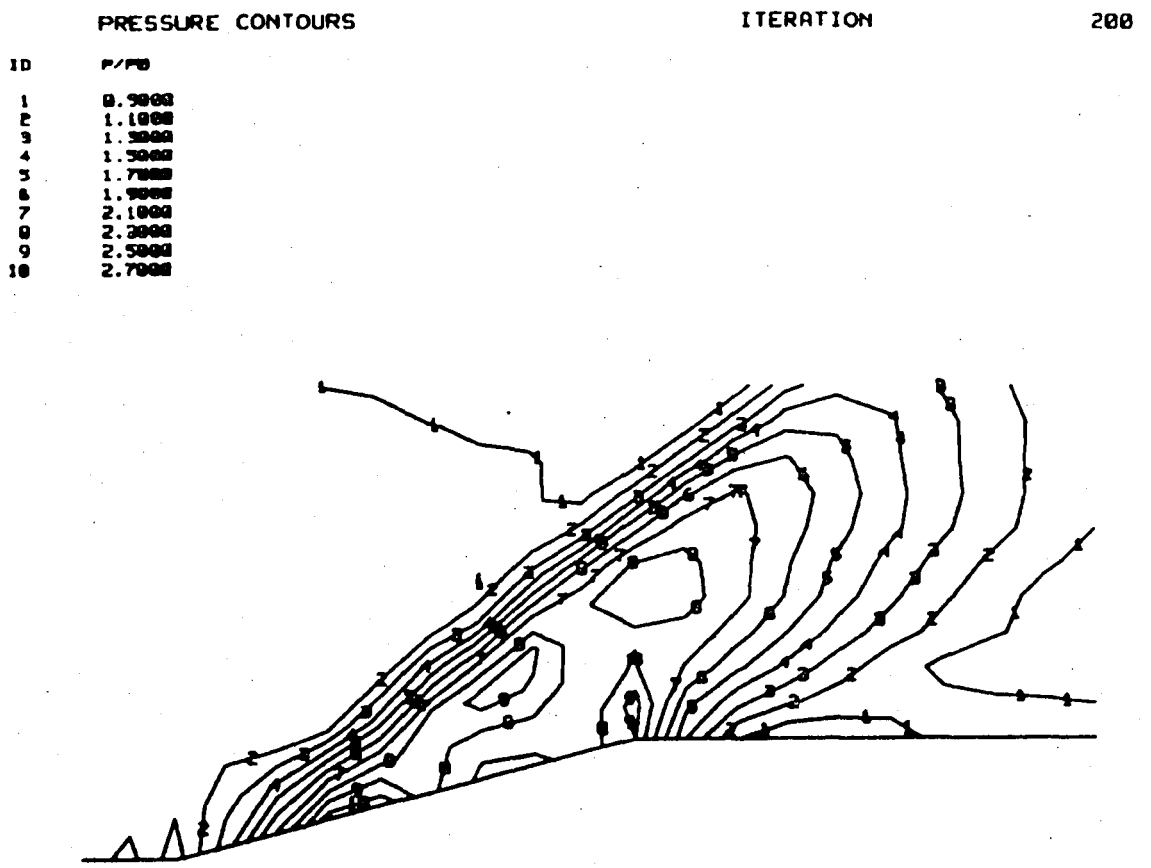


Fig. E-2 Two-Dimensional Shock-Expansion Test Case (Concluded)

PAGE CODE GRID

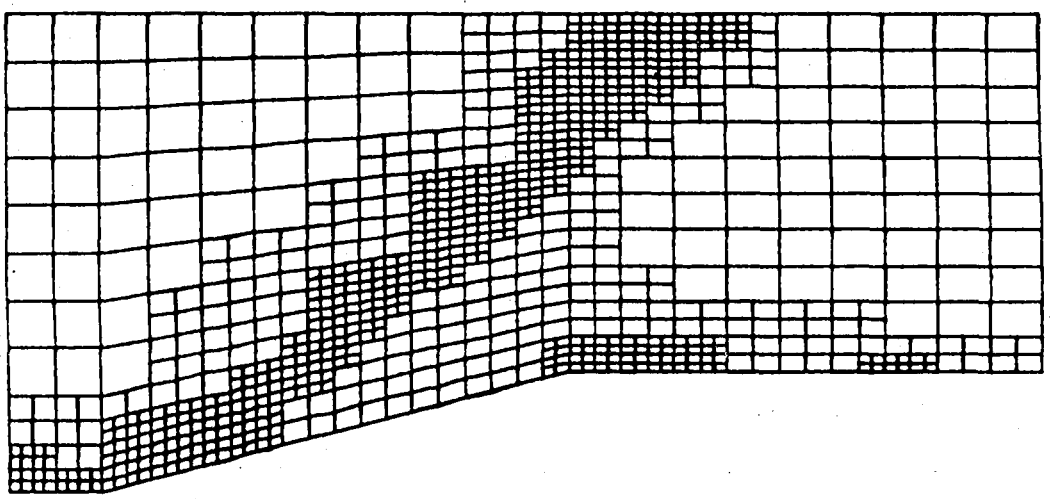


Fig. E-3 Two-Dimensional Shock-Expansion Test Case
Refined Grid

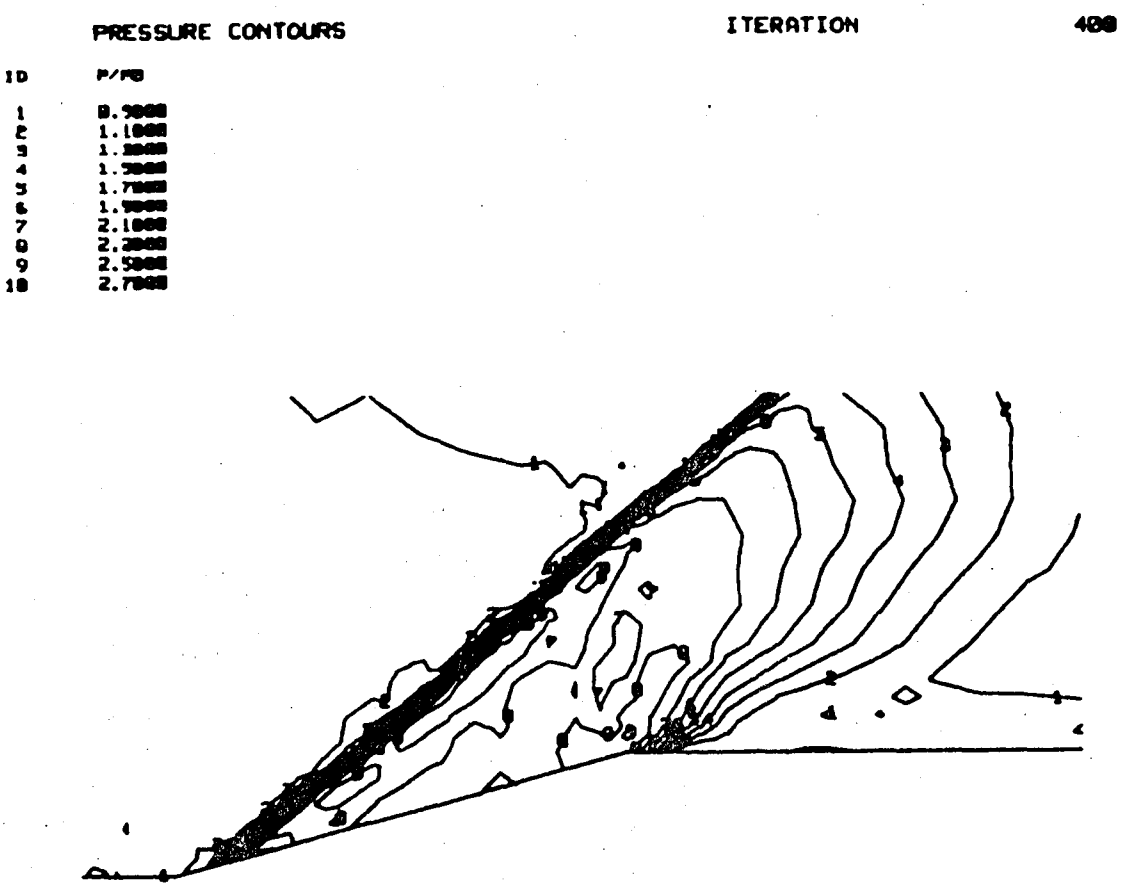


Fig. E-3 Two-Dimensional Shock-Expansion Test Case
Refined Grid (Concluded)

PAGE CODE GRID

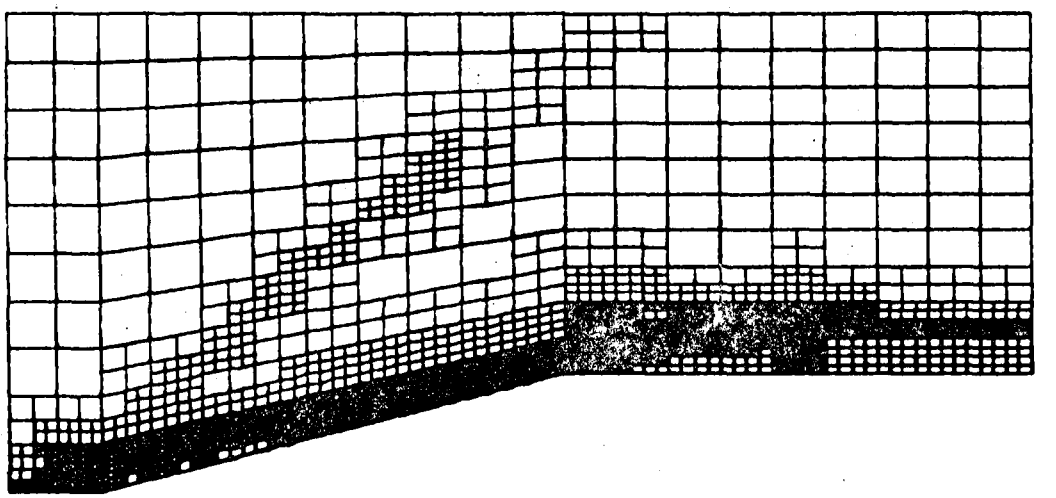


Fig. E-4 Two-Dimensional Shock-Expansion Test Case Refined Grid

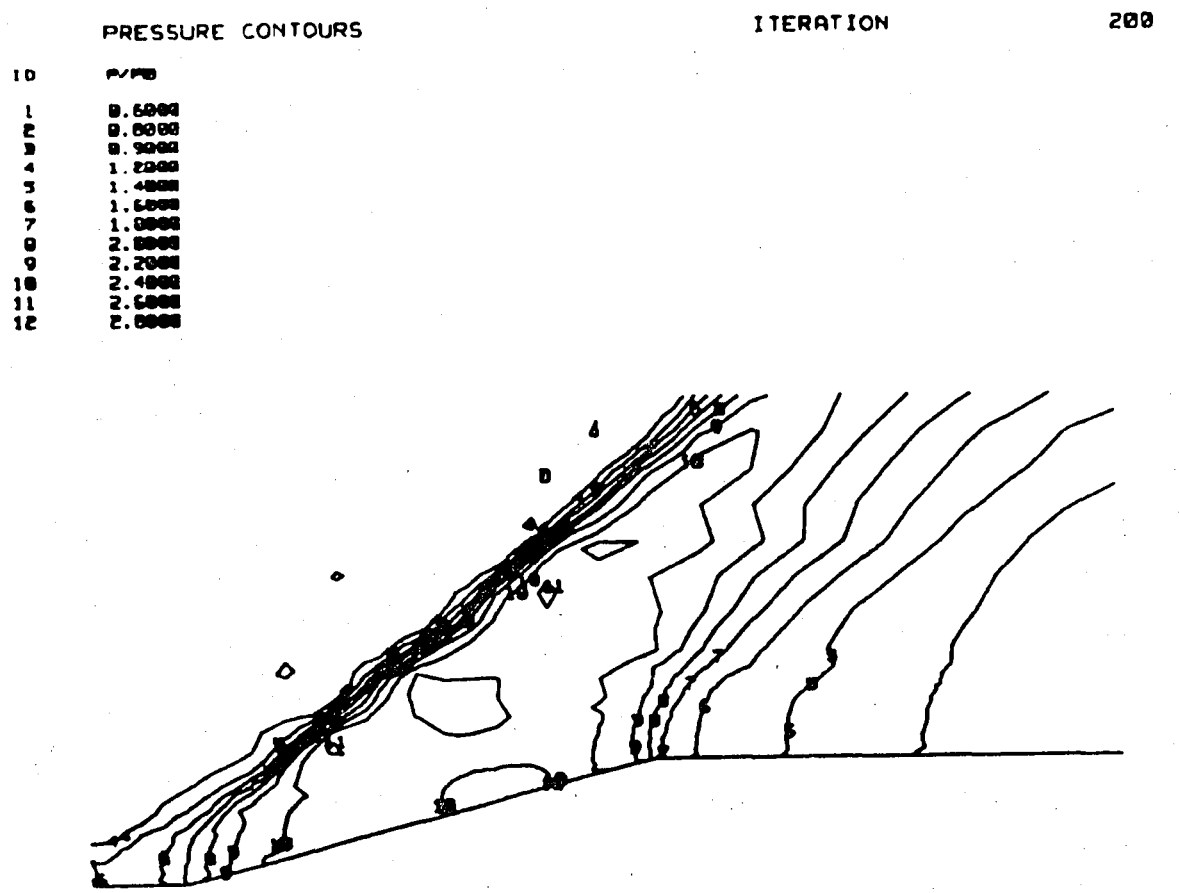


Fig. E-4 Two-Dimensional Shock-Expansion Test Case Refined Grid (Concluded)

PAGE CODE GRID

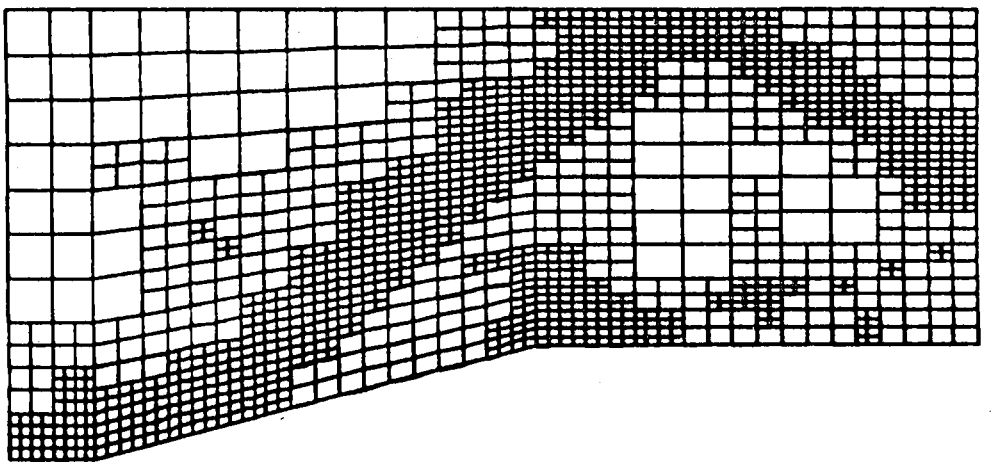


Fig. E-5 Two-Dimensional Shock-Expansion Test Case
Refined Grid

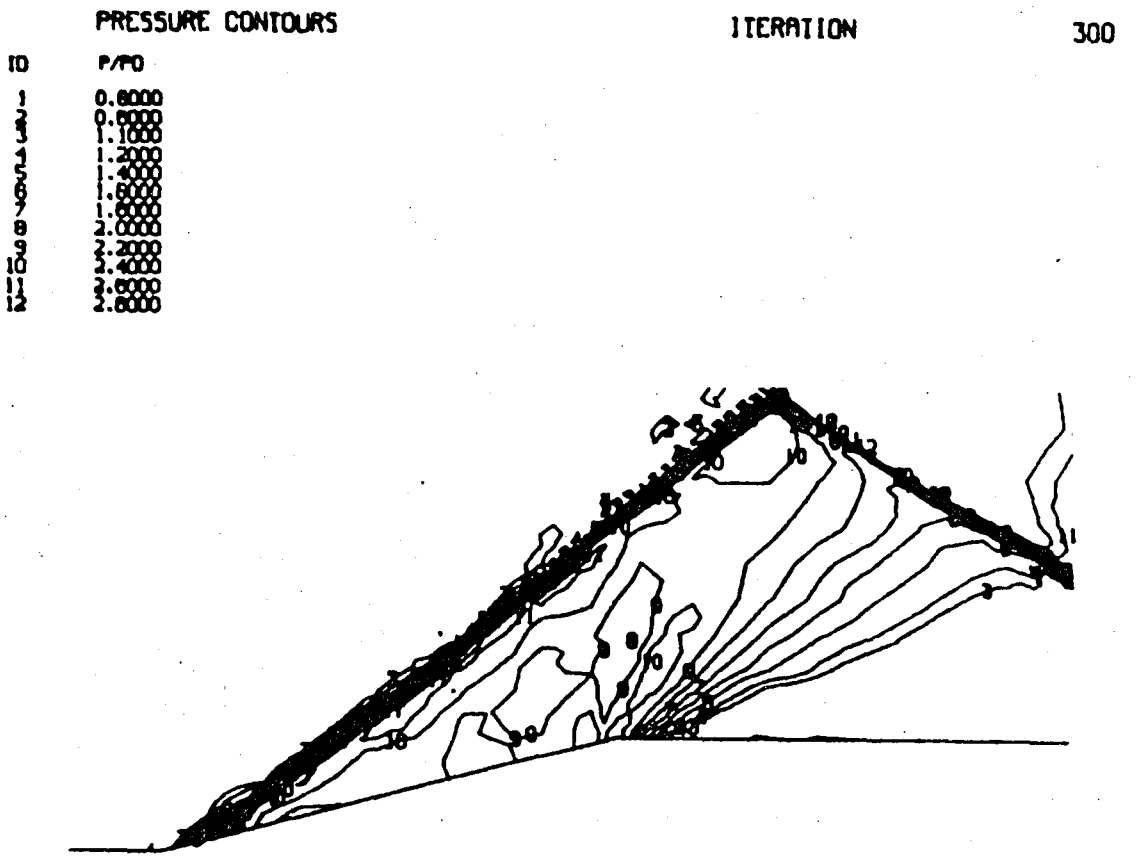


Fig. E-6 Two-Dimensional Shock-Expansion Test Case Refined Grid

Appendix F

THREE-DIMENSIONAL ADAPTIVE SCHEME

Appendix F

The three-dimensional adaptive scheme is divided into subprocesses each of which performs a specific task or function. The subprocesses are:

1. Feature detection
2. Mesh adaption
3. Boundary condition application
4. Data translation

These subprocesses surround a central data pool as shown in Fig. F-1. This arrangement is essentially the same organization used by Dannenhoffer and Baron (Ref. 1) in their hybrid expert system.

Two-way data transfer is allowed between each subprocess and the central data pool. There is no communications between any two subprocesses. As Dannenhoffer and others have previously stated, this type of communication structure benefits the overall system design and development. Each subprocess may be developed and tested separately. Each subprocess is independent of the other subprocess. Any algorithm change in a subprocess will not impact any other subprocess. All requests for data by any subprocess is handled by a collection of procedures which read and write data to the data pool. The data pool consists of a collection of files which can reside on more than one directory or machine.

The data manager processes information transfer requests between any subprocess and the data pool. The data manager keeps track of available elements and groups, retrieves information about a particular element or group and checks for non-existing data request.

The feature detection subprocess determines which regions of the mesh should be refined or unrefined. The mesh adaptation subprocess does the

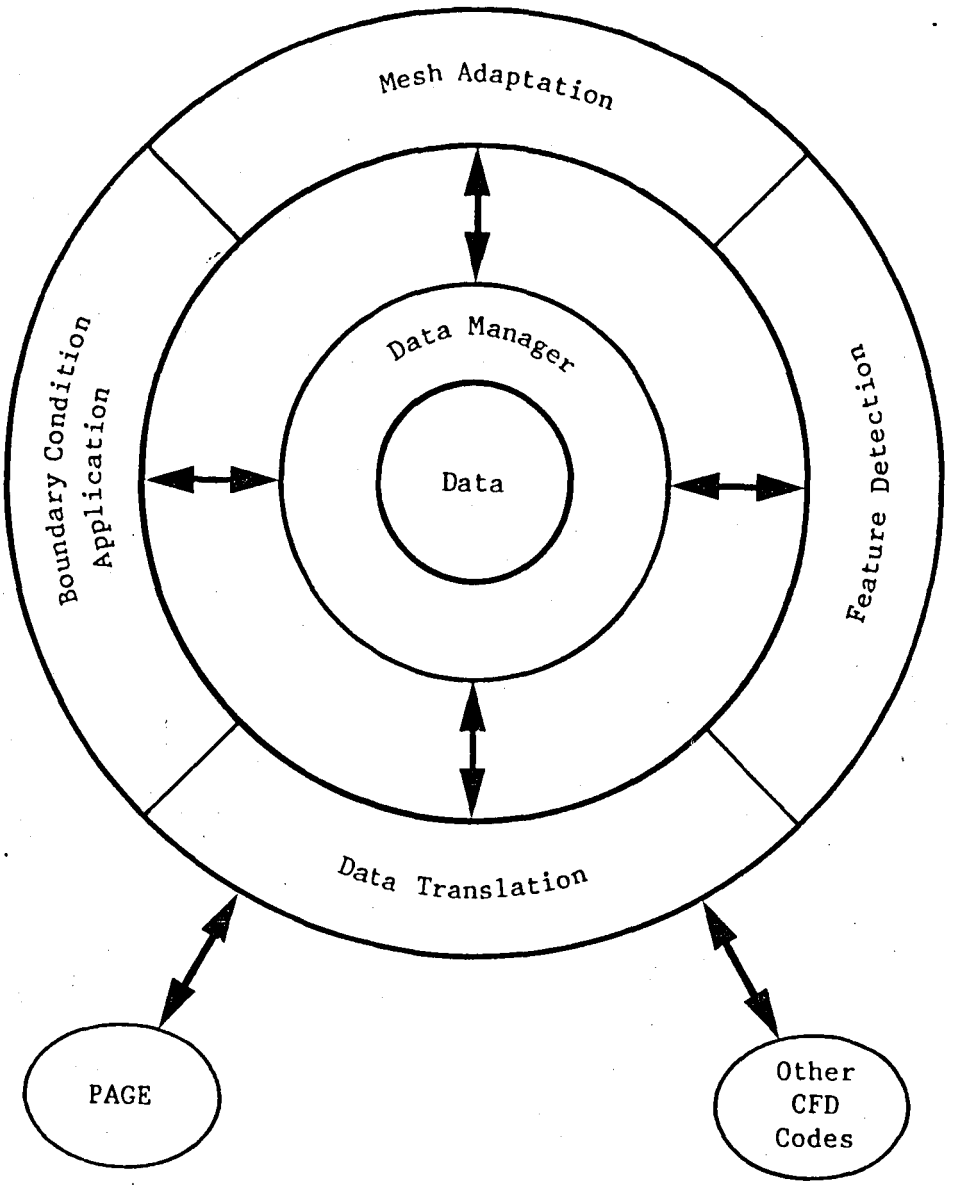


Fig. F-1 System Organization

mesh refinement and unrefinement. The boundary condition application subprocess applies boundary conditions on the adapted mesh. The data translation subprocess converts the data to a form which can be used by a CFD code.

The rest of this appendix describes the mesh adaptation subprocess. The other subprocesses have not been completed except for the data manager.

Mesh Adaptation

The mesh adaptation subprocess refines and unrefines the mesh using a list of instructions generated by the feature detection subprocess. This list resides in the data pool.

Two pieces of information are necessary to adapt the mesh. These are the relationship of an element with its neighbors (connectivity array) and the lineage of the element (group array). Using the connectivity and group arrays, the mesh can be adapted.

Connectivity Array

A connectivity array is employed within the integer space to facilitate mesh refinement and unrefinement. This array keeps track of the dynamic topological relationships between the elements of the mesh. Mesh adaptation is performed by operating on this connectivity array.

The three-dimensional connectivity array consists of a 4 x 4 x 4 array. Each element of this array is referred to as an atom. The array consists of a group of core atoms surrounded by neighboring atoms and neutral atoms. A connectivity array is illustrated in Fig. F-2. There is one connectivity array for each nonboundary element in the mesh. This array contains the topological data which relates one element with its immediate neighbors. Using this array the relative level of an element with its neighbor can be easily determined. Constrained nodes can also be identified.

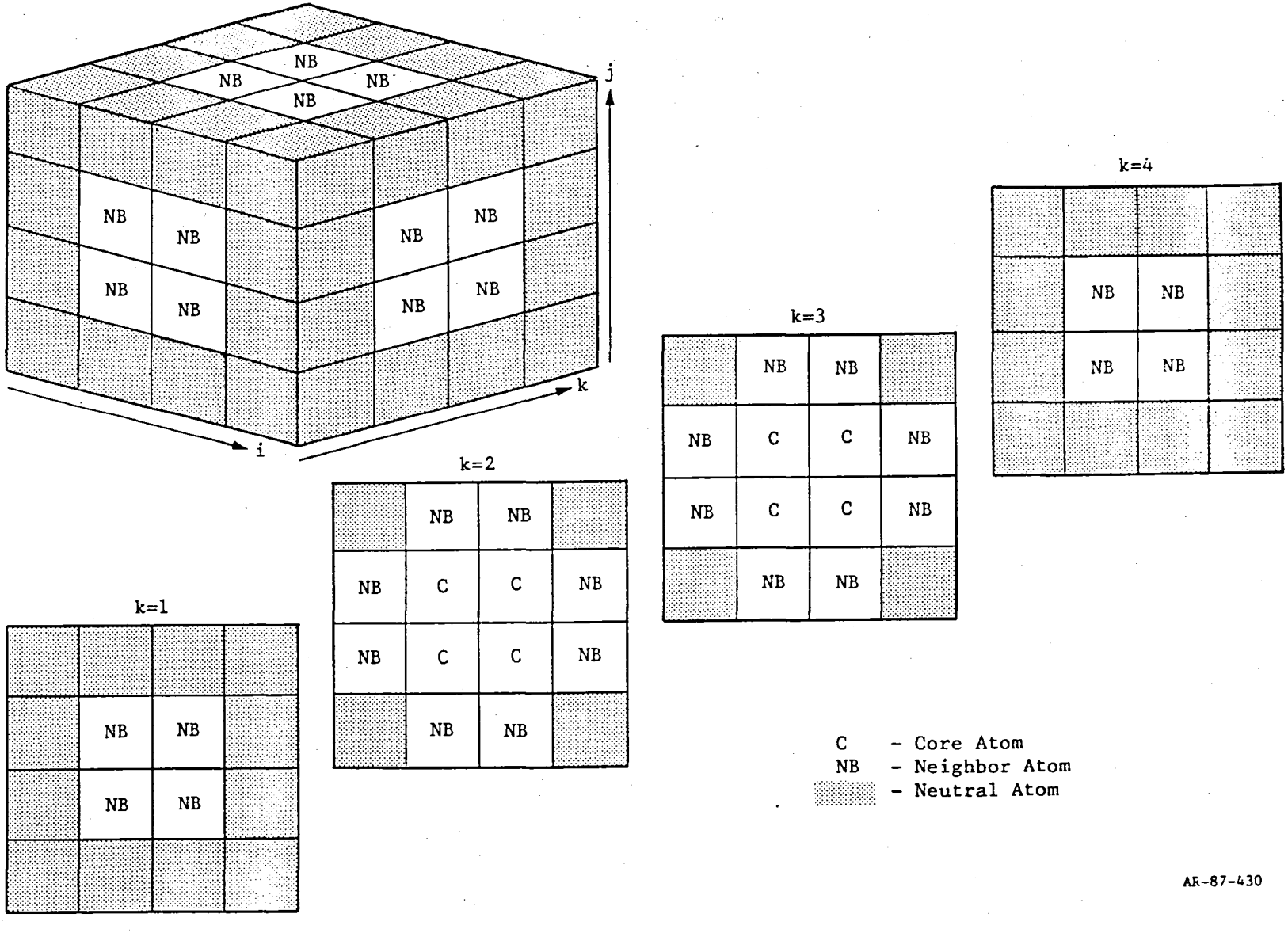


Fig. F-2 Connectivity Array

Group Array

The group array is used to facilitate the unrefinement process. This array contains the elements which are created when an existing element is refined. It keeps track of element lineage.

Refinement Algorithm

The following algorithm outlines the steps required to perform the refinement operation for a single element.

Step 1. Copy the connectivity array of the element to work the array.

Step 2. Get the next available group and initialize the new group array.

Step 3. For each core location of the work array:
- 3a. Get the next available element and place it in the core location.
- 3b. Initialize next available element's connectivity array.

Step 4. Copy core locations of the work array to the new group array.

Step 5. For each group location of the group array:
- 5a. Get the element stored in that group location.
- 5b. Transfer connectivity data from the work array to each element's connectivity array.

Step 6. For each neighbor location of the work array:
- 6a. Get the element stored in the neighbor location.
- 6b. Update the neighbor element's connectivity array.

Step 7. Update the old group array which the element belonged to.

Step 8. Delete the element connectivity array and add element to the available element heap.

This refinement algorithm is illustrated in Fig. F-3. A single group of eight elements is successively refined until it consists of 13 groups totaling 98 elements. One can see from these figures that there is a maximum of one level of refinement difference between any two adjacent elements. An exploded view of the final grid is shown in Fig. F-4.

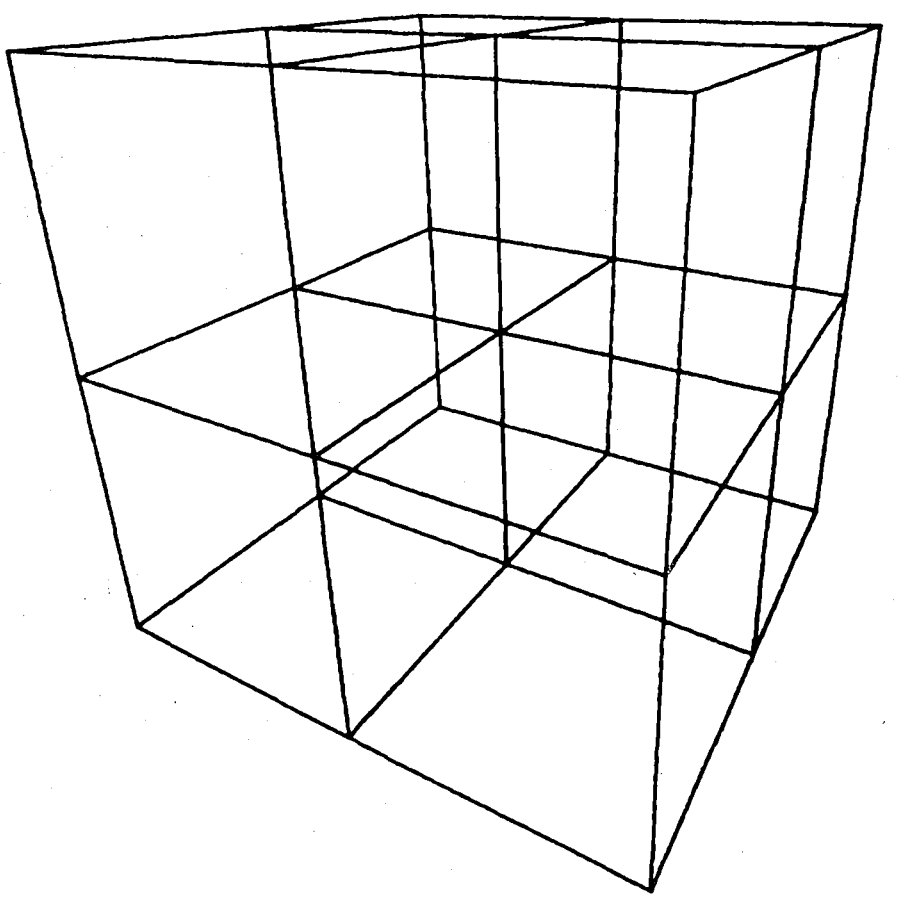


Fig. F-3a Original Mesh

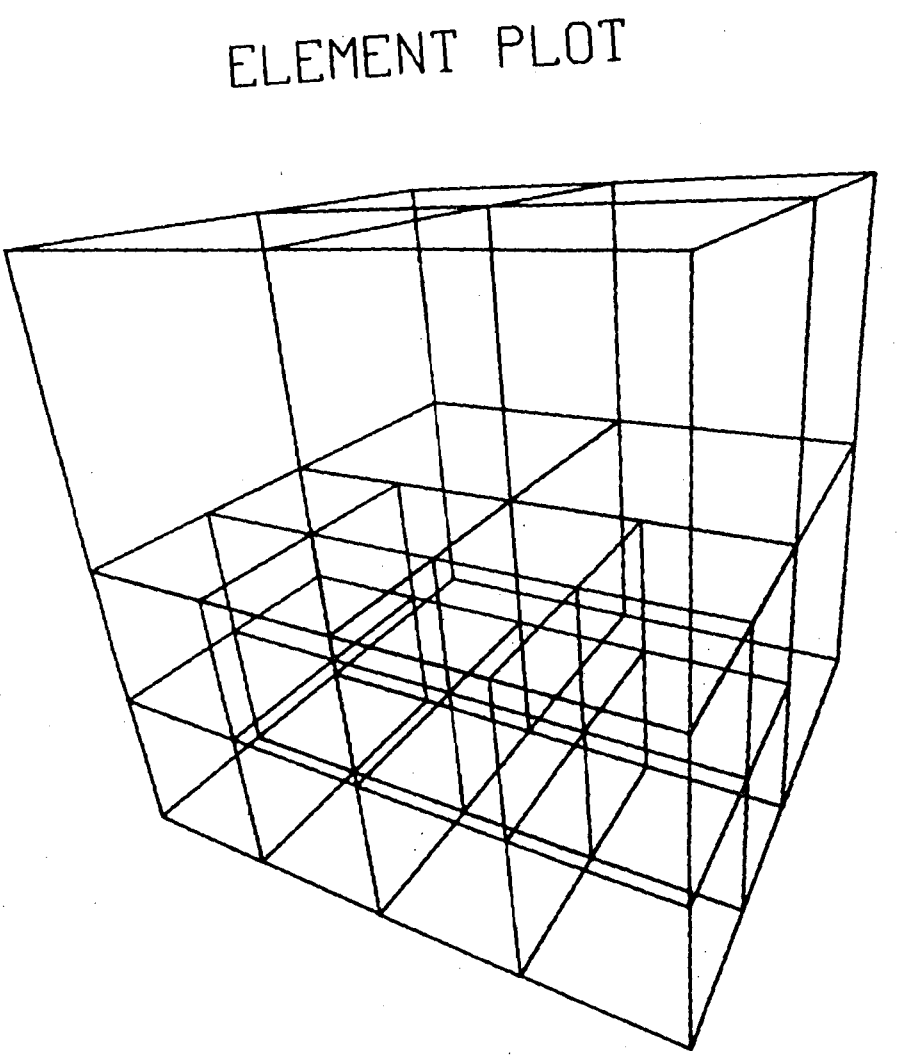


Fig. F-3b First Refinement Pass

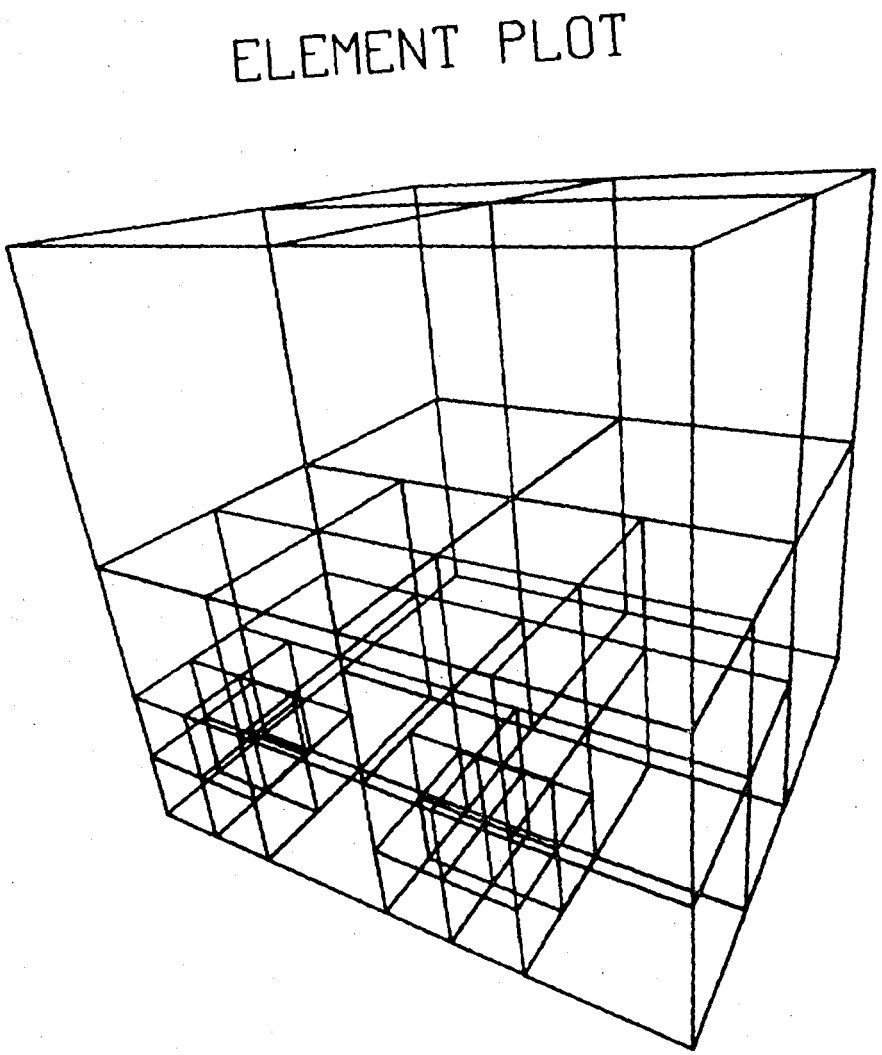


Fig. F-3c Second Refinement Pass

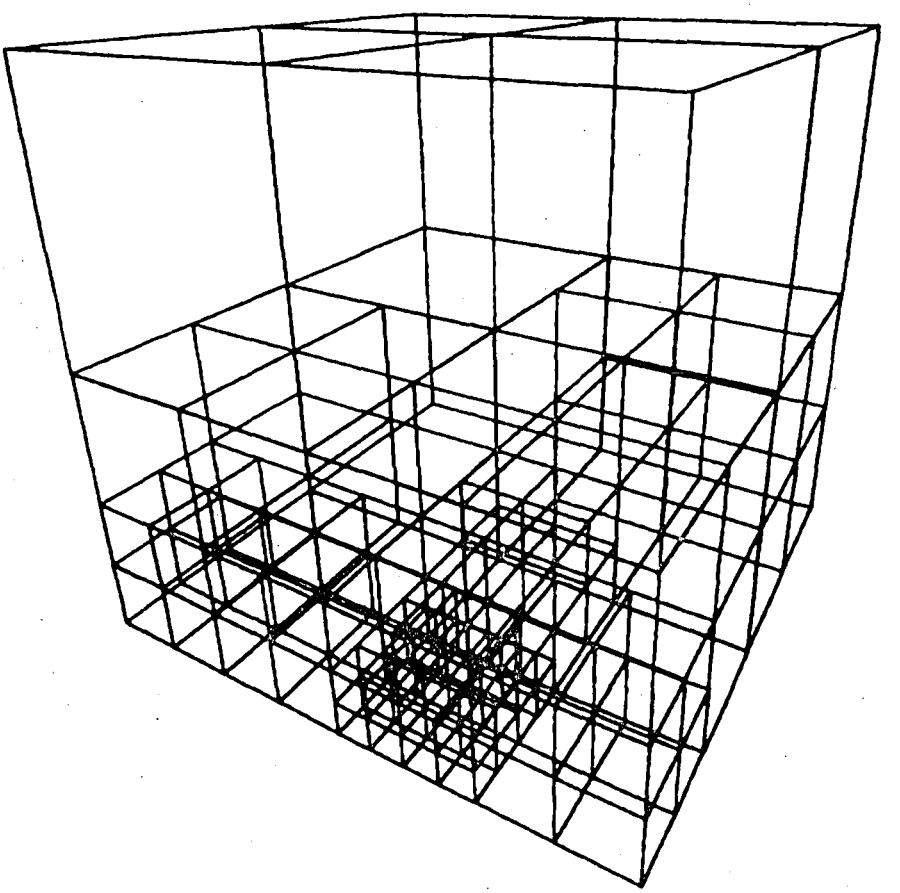


Fig. F-3d Third Refinement Pass

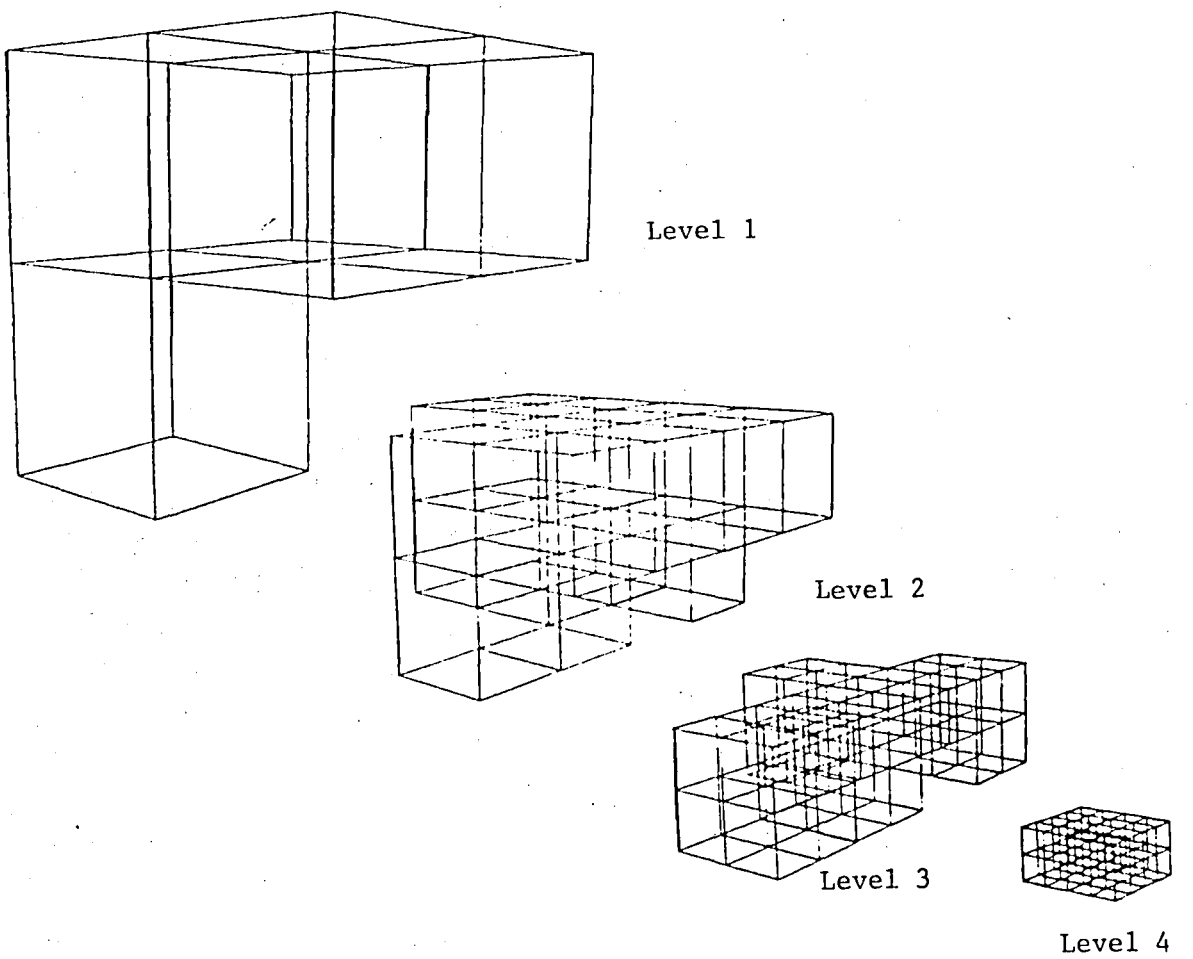


Fig. F-4 Exploded View of the Grid After Three Refinement Passes

Unrefinement Algorithm

The following algorithm outlines the steps required to perform the unrefinement operation for a single group.

Step 1: Copy the group array to the core of the work array.

Step 2: For each core location of the work array: (a) Get the element stored in the core location and (b) transfer connectivity data to the work array.

Step 3: Get next available element.

Step 4: Store this element in each core location of the work array.

Step 5: For the group array: (a) Get an element for the group array, (b) update the connectivity array of each neighbor of this element, and (c) add the element to the available element heap.

Step 6: If this is an embedded group then replace any reference to the group by the next available element.

Step 7: Add each member of the group array to the next available element heap.

Step 8: Add group to the next available group heap.

Figure F-5 is a refinement/unrefinement cycle using the above algorithms.

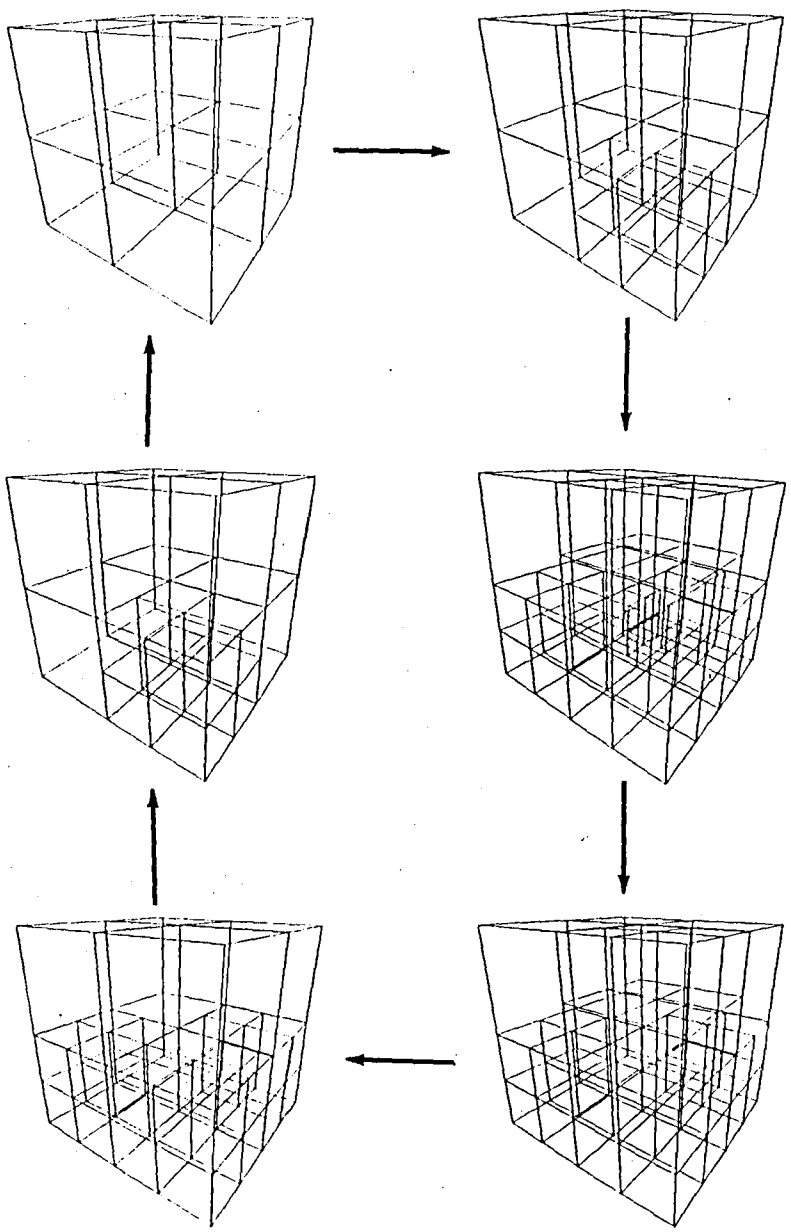


Fig. F-5 A Mesh Adaption Cycle

REFERENCE

1. Dannenhoffer, J.F., and J.R. Baron, "A Hybrid Expert System for Complex CFD Problems," AIAA Paper 87-1111, 1987.

NASA Contractor Report 181636
Distribution List
NAS1-17894

	No. Copies
NASA-Langley REsearch Center Hampton, VA 235665-5225 Attn: 151/Research Information Office 395/Dr. George C. Olsen	2 85 + orig.
NASA-Ames Research Center Attn: 202-3/Library Moffett Field, CA 94035	1
NASA-Dryden Flight Research Facility Ames Research Center Attn: Library P.O. Box 273 Edwards, CA 93523	1
NASA-Goddard Space Flight Center Attn: Library Greenbelt, MD 20771	1
NASA-Marshall Space Flight Center Attn: AS24L/Library Marshall Space Flight Center, AL 38412	1
Jet Propulsion Laboratory Attn: 111-113/Library 4800 Oak Grove Drive Pasadena, CA 91103	1
NASA-Lewis Research Center Attn: 60-3/Library 21000 Brookpark Road Cleveland, OH 44135	1
NASA-John F. Kennedy Space Center Attn: NWSI-D/Library Kennedy Space Center, FL 32899	1
National Aeronautics and Space Administration Attn: RM Washington, DC 20546-0001	1
NASA Scientific and Technical Information Facility P.O. Box 8757 Baltimore/Washington International Airport, MD 21240	5

End of Document