New Developments in Adaptive Finite Element Methods in Computational Fluid Dynamics

J. Tinsley Oden

Texas Institute for Computational Mechanics
The University of Texas at Austin

MSC Annual User's Conference
Los Angeles, California

March, 1990

New Developments in Adaptive Finite Element Methods in Computational Fluid Dynamics

J. Tinsley Oden

Texas Institute for Computational Mechanics
The University of Texas at Austin

Foreword and Abstract:

Slightly over two decades have past since finite element methods were first applied to the Navier-Stokes equations of fluid dynamics [1–6]. For much of the intervening twenty years, their impact on practical CFD (computational fluid dynamics) calculations has been relatively small compared to that of conventional finite difference schemes, especially in calculations of high–speed flows. Over the last five years, however, this situation has dramatically changed. During recent years, new finite element techniques have appeared which are finding application to an increasing list of flow simulation problems. The versatility, reliability, and generality of these techniques suggest that the technology for developing general–purpose CFD codes is at hand, and that these new tools will provide a new dimension to engineering analysis and design.

At the heart of these new finite element methods is the notion of *adaptivity*: the capability of numerical algorithms to modify themselves during a calculation to accommodate changing properties of numerical solutions. These modifications can be manifested in changes in the structure of the mesh, the order of the polynomial shape functions, the location of nodes, the form of the equation solver, the explicit or implicit character of time marching schemes, or in combinations of all of these features. When added to the well–known ability of finite elements to handle complex geometries and boundary conditions, modeling techniques with truly remarkable and powerful capabilities emerge.

The present paper provides a general overview of adaptive methods in CFD and describes recent progress toward developing general adaptive finite element codes for broad classes of flow simulation.

1 Introduction

The goal of adaptive methods is *optimal control* of the computational process: to control the computation so as to produce the "best" results for the least effort. Differences in various adaptive schemes hinge on how this optimal is defined, on how the control parameters are selected and measured, and on how the "effort" is defined. In most of the results described here, the control parameter is the *numerical error* of the solution in each finite element, so that one attempts to control the accuracy of the computation. We also mention some strategies in which numerical stability is also a factor, so that the time step, for example, is also controlled to maintain stability at minimum computational cost.

There is a wonderful byproduct of such a control strategy: if the error in a computation can indeed be estimated, then the user has a ready–made measure of the reliability of his calculation. A good adaptive scheme of the type discussed here employs sound, efficient, and reliable *a posteriori error estimators*, to estimate the evolution of error during a computational process. In principle, such error estimators provide the user with a quantitative answer to a question asked with increasing frequency in computer analyses: *How good are the answers?*

The best adaptive procedures function independently of the user, who merely prescribes a level of error he can tolerate or a dollar–value (the cost) he is willing to endure to complete a flow simulation. Thereafter, the adaptive code makes the decisions necessary to produce solutions within the user–specified limits. Once the control parameters (the element errors or some reasonable approximation of them) are available, the code attempts to adjust them to meet control objectives—to minimize the error. Typically, the control of error is achieved by refining the mesh in areas of the solution domain where errors are too large and by coarsening the mesh (using larger elements) where the error is small, or relocating nodes to increase nodal densities near regions of high error. Also, one could increase the degree of the local shape functions in an element of fixed size and expect the accuracy of the approximation to be increased. Thus, to summarize, there are several broad types of mesh adaptivity that can be used to control error:

> *h-methods*—in which the element size h is used to control error. Error is reduced by refining the mesh or by regenerating a new finer mesh. To obtain an *optimal h-mesh* (one with the least error possible for a fixed number of refinements), provisions for also coarsening a mesh must be included in the adaptive strategy.
>
> *r-methods*—in which a fixed number of elements of a given order are used in a FE mesh, but in which nodes are *relocated* to reduce error in certain regions. The r-methods are thus moving grid methods in which the number of degrees of freedom is fixed, but the node locations are adapted to control error.

p-methods—in which the polynomial degree p of the element shape functions is raised or lowered to control error. These p-refinement methods are mathematically akin to spectral methods used, for instance, in turbulence simulations, and some prefer to call these approaches *spectral adaptive methods.* The MSC/PROBE program falls into this category.

combined methods—in which combinations of h-, r-, and/or p-adaptivity are used. Some surprising and unexpected results have recently been obtained using such combined adaptive techniques.

There is a growing literature on adaptive finite element and finite differences methods for broad classes of problems. Surveys of the state of the subject primarily as it applies to elliptic problems can be found in [7, 8]; surveys of recent advances in adaptive methods for CFD can be found in [9–15] and other work on the subject is given in [16–21].

The objective of this presentation is to describe representative applications of adaptive finite element methods to several classes of problems in CFD and to outline the major features and considerations that are encountered in implementing particular adaptive strategies. We also describe applications of an adaptive finite element code under development called *ADAPT*®, designed to exploit many of the advantages of adaptive methods:

2 The Intelligent Adaptive Code

Adaptivity in a CFD code is just one example of the use of a *smart algorithm* in flow simulations. A smart algorithm is a numerical procedure that changes its structure and form during a calculation to accommodate changing features of the numerical solution. Together, smart algorithms and adaptive methods have the potential of removing from the hands of the user many *ad hoc* decisions in computer modeling that can affect the quality and efficiency of results. Ideally, such schemes provide a CFD tool that can be used as follows:

1. The user supplies a very crude mesh—perhaps only roughly defining the flow domain of interest, and he specifies tolerances he will accept in the error (or in the cost of the calculation).

2. Thereafter, the computer code computes an initial solution, checks error and stability tolerances, and decides if an adaptation is needed to improve the solution; if so:

3. The code chooses an optimal approximation structure—an algorithm—for most effectively reducing the error. The mesh, spectral order, and properties of the solution algorithm are changed accordingly and a new solution is produced. The adapted mesh also adapts itself more closely to fill up the actual flow domain.

4. Post–processing routines enhance the solution, present it to the user, plot estimated errors, all without interference from the analyst.

The engineer need not know at this point how the code chose to compute the solution in hand. The intelligent engineer, however, would be expected to interrogate the solution to determine which mesh structure and algorithm were ultimately used to complete the analysis.

3 *ADAPT®*: Towards a General Adaptive Package for CFD

The general adaptive philosophy outlined above has been used as the basis for the design of a new family of CFD codes. For several years, the author and his colleagues have been developing a general adaptive finite element code for the analyses of both compressible and incompressible flows called *ADAPT®*. The code features a general Preprocessor which receives data and parameter sets for specific options that are to be used for a given calculation, and a rich collection of algorithms and special formulations for many classes of flow problems. Typically, the data include:

- Initial mesh parameters (generally including specification of the coarsest level permitted in mesh "unrefinements")
- Data for boundary and initial conditions (*ADAPT®* monitors flow conditions at boundaries and automatically changes them when the local physics dictates a change is needed)
- Identification of flow types to be analyzed
- Fluid variables and coefficients (e.g., viscosities, thermal conductivities)
- Error tolerances in specified norms (optional, if unspecified, the code will choose a norm appropriate for the problem being analyzed)
- Flow solvers (also optional, but if left unspecified, *ADAPT®* will choose an initial algorithm appropriate for the specified applications)
- Grid velocities for interaction problems (rotor–stator interaction, burning boundaries, etc.)
- Chemical species equations

- Turbulence model identification, etc.

The Preprocessor supplies data to the main computational unit, called the kernel, where major data management and utilities are handled.

The *ADAPT®* code uses *h*-, *r*-, and *h-r* adaptivity. A companion code called *PHASER®*, is based on *h-p* adaptive strategies but is not specifically aimed at CFD applications. The *ADAPT®* code is designed to analyze the following broad classes of flow problems:

- supersonic, transonic, and subsonic compressible viscous flow
- incompressible viscous flow for wide ranges of Reynolds numbers
- inviscid flow—including real gas models
- chemically–reacting flow, including finite chemistry, nonequilibrium, radiative, turbulent flow
- fluid–structure interaction, including general problems with moving boundaries
- turbomachinery and other internal flows including a multi–blade, multi–stage, rotor–stator flow interaction.

Some representative results obtained using this code are described in subsequent sections of this paper.

4 Examples of Strategies and Results

An h-Refinement/Unrefinement Method. One *h*-procedure involves the following steps:

1. For a given domain Ω, a coarse initial finite element mesh is constructed which contains only a number of elements sufficient to model basic geometrical features of the flow domain. The initial mesh can be quite arbitrary, and can be generated by the pre–processing units of the code or by any convenient independent processor that interfaces with the code (*ADAPT®* has its own mesh generation and pre–processing directories, but also interfaces with standard CFD mesh generators, such as GRAPE or FLOW3D, etc.).

2. The adaptive process is designed to handle groups of N elements at a time (four for the two–dimensional case and eight for the three–dimensional case), a finer starting grid can be generated by a bisection process to obtain an initial set of element groups. The

user specifies the number of bisections needed to define the "initial" mesh. This will determine the largest element size that can be used in a flow simulation. Obviously, the larger the size of an element for a fixed Ω, the smaller the size of the problem that must be solved to deliver a solution of specified accuracy.

3. The numerical solution procedure is initiated on this initial coarse grid, and error indicators θ_e are computed over all M elements in the grid. Suppose that $u = u(x,y)$ is the exact solutions for some two–dimensional problem and that $U = U(x,y)$ is its finite element approximation. The approximation error is the function

$$e(x,y) = u(x,y) - U(x,y)$$

and the local error indicators for element Ω_e may approximate error norms of the following type:

$$(\theta_e)_1 = \text{average error} = \frac{1}{|\Omega_e|}\int_{\Omega_e} |e|\, d\Omega \quad (|\Omega_e| = \text{area or volume of } \Omega_e)$$

$$(\theta_e)_2 = \text{error in energy} = \int_{\Omega_e} \left(e_x^2 + e_y^2 + e^2\right) d\Omega \text{ (for example)}$$

$$(\theta_e)_3 = \text{maximum change in error} = \max_{(x,y)\in\Omega_e} |\nabla e(x,y)|$$

etc. Let

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

4. Next, the element groups are scanned of a fixed number P of elements and the code computes

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

where e_k is the element for group k. *ADAPT*® uses $P = 4$ in two–dimensional problems and $P = 8$ in three dimensions.

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

$$\theta_e \ge \beta\theta_{\text{MAX}}$$

is *refined* element θ_e. This is done by bisecting θ_e into four (eight) new subelements in two–dimensional (three–dimensional) cases. If

$$\theta^k_{\text{GROUP}} \le \alpha\theta_{\text{MAX}}$$

the group k is coarsened by replacing this group with a single new element with nodes coincident with the corner nodes of the group.

This general process can be followed for any choice of an error indicator. Moreover, it can also be implemented at each time step.

One possible adaptive scheme for time–dependent problems is:

1. Advance the solution N time steps Δt using an appropriate time–marching scheme.
2. Calculate error estimates.
3. Refine the mesh (the need for some refinement means that the last N time steps were computed using the "wrong" mesh—one with too much error; thus:)
4. Redo the N time–step calculations using the new refined mesh.
5. Redo the error estimation.
6. Unrefine (coarsen) the mesh.
7. Go to 1.

There are several rather obvious alternative versions of this algorithm, but this is the approach used in the sample calculations presented later in this paper. One could, for example, continue to loop through steps 2 through 7 and not allow the solution to advance until there was no change in mesh structure. This would produce a very accurate but a very slow scheme.

r-Methods. The moving mesh algorithms incorporated in *ADAPT*® are designed primarily for flow– and fluid–structure interaction problems. These include:

- rotor–stator flow interactions in turbomachinery, in which flows around moving blades are encountered,
- fluid–structure interactions, in which elastic structures undergo large–amplitude motions in a viscous flowfield, and
- burning, receding, or growing boundary problems in which the boundary of the flow domain is changing due to special effects such as burning, melting, or injection of other materials.

In all cases, the key to the analysis is the specification of the so–called *grid velocity* $\boldsymbol{v}_G$. In rotor–stator flow–simulation, $\boldsymbol{v}_G$ is specified as the relative velocity of blades in various stages of the turbine *ADAPT*® employs a sliding mesh algorithm which assigns a different mesh to each stage and then slides the neighboring meshes along one another at specified mesh interfaces. In the fluid–structure simulations, three types of meshes are used: an Eulerian

mesh in the fluid, a Lagrangian mesh in the structure, and a referential mesh attached to both the structure and the Eulerian mesh which is stretched according to a special scheme that prevents severe element distortions.

To model the burning or growth of a flow boundary, nodes on the mesh boundary are advanced normal to the current boundary each time step a distance equal to the product of the burning or growth rate and the time step. These new points are then connected by cubic splines to define the new fluid boundary. A check of mesh smoothness is also made with each advance of the boundary line and, if necessary, gridpoints are relocated to preserve mesh integrity. The rate of boundary recession, of course, may be a function of the computed flow itself, so that v_G is then computed element-by-element at each time step over the current mesh.

Figures 1–5 illustrate typical results obtained using these methods. Figures 1–3 contain results of a transient, two– and three–stage, two– and three–dimensional rotor–stator calculations. Note that the time–dependent h-adaptive scheme described earlier is also employed so that a near–optimal mesh is used throughout a flow simulation. Thus, in Figs. 1–3 we see an uneven and unstructured mesh adapted to satisfy the error control criterion mentioned earlier.

Figures 4–8 show other representative results. There we see the same fluid–structure interaction simulations and a simulation of the evolution of a burning erosion pocket in a solid propellant rocket motor in which the flow domain is changing with a flow dependent burning of the propellant boundary. Note that here we have a *combined h-r method*: while the boundary moves, the mesh is also refined to meet preset accuracy requirements. These h-r results are among the first every produced by a CFD code and represent an application of the *ADAPT*® code.

Three–dimensional examples are illustrated in Fig. 9. These are test results for the non–equilibrium chemistry routines in *ADAPT*®.

p- and h-p Methods. The idea of increasing the local order of an approximation while keeping mesh sizes fixed is a natural one in the case of problems with thin boundary layers or singularities. Some results on CFD calculations done using hierarchical p- and h-p finite element methods are to be cited below. In these calculations, element shape functions of the form

$$\varphi(\xi,\eta) = \sum_{i,j} \chi_i(\xi)\chi_j(\eta)$$

are used, where

$$\chi_i(\xi) = \text{ polynomial of degree } \leq p \text{ in } \xi,\ -1 \leq \xi \leq 1$$

The degrees of freedom for such elements are not purely nodal values of the solution but are also values of tangential derivatives at the midpoints of element boundaries or mixed

derivatives at the centroid. The resulting element matrices have hierarchical structure, in that those correesponding to an approximation of degree p are submatrices of those corresponding to approximations of degree larger than p.

The h-p methods employ both h-refinement/unrefinement and p-enrichment/unenrichment and represent one of the most complex and delicate adaptive data structures. However, exceptionally fast rates of convergence can be obtained by using these techniques, and this often offsets the cost of implementation.

Sample adaptive h-p results are shown in Figs. 10–11. Those in Figs. 10 and 11 demonstrate results on *incompressible* flow in two– and three–dimensions.

The Optimal Mesh. One of the basic issues that must be resolved in an adaptive calculation is to determine the best possible change in the mesh structure to most efficiently and quickly reduce local error. In other words, for a given local error measure, what should be done to improve the solution, decrease h or increase p, or both? While this "optimal path" question is, to a great extent, still open, we have employed one technique that is reasonably effective. It is based on the following analysis: Let

θ_e = error indicator density for element Ω_e in a regular 2D finite element mesh.

h = piecewise constant mesh size function;

$$h(x,y) = h_e = \text{ dia } (\Omega_e) \text{ for } (x,y) \in \Omega_e$$

p = piecewise constant polynomial degree functions,

$p(x,y) = p_e$ = highest polynomial degree of shape functions supported by element Ω_e for $(x,y) \in \Omega_e$.

The total error is given by the functional,

$$J(h,p) = \sum_e \int_{\Omega_e} \theta_e(h,p) d\Omega$$

The total number of degrees of freedom active in such an h-p approximation is roughly given by the functional

$$N = \int_\Omega n(p,h) d\Omega$$

where $n(p,h)$ defines the density of degrees of freedom and is obviously dependent on the local elementwise mesh parameters h_e and p_e. The *optimization problem* is to find a particular h-p distribution, (h,p) such that J is a minimum subject to the constraint the N = constant:

$$J(h,p) = \inf_{N=\text{const}} J(h,p)$$

Several optimization schemes have been developed to treat this problem (see Rachowicz, et al. [17]). In particular, a one–step optimization algorithm has been used successfully in one–dimensional and two–dimensional elliptic problems. Some results are given later.

5 Adaptive and Automatic Algorithm Selection

During a calculation on an adaptive mesh, how can one be sure that the algorithm or flow solver selected to solve the discrete problem be still appropriate for the condition prevailing at that time in the behavior of the solution? Better still, why should one expect a single flow solver to be effective everywhere, at a given time instant, throughout the current mesh? Answers: one cannot expect a single algorithm to remain effective and it may be necessary to use different algorithms at different portions of the mesh at different times.

Explicit/Implicit Methods. An excellent example of smart algorithms tied to an adaptive structure are those which are capable of using explicit or implicit time marching schemes depending on local accuracy and stability requirements. A typical scenario is as follows:

- We wish to advance the calculation of a transient flow problem one time step. The mesh sizes h_e of the elements Ω_e are determined by an h-adaptive scheme so as to satisfy present accuracy tolerances.

- The time–accurate scheme is, for example, based on a second–order (in time) Taylor-Galerkin method built around the Taylor expansion,

$$u(t+\Delta t) - u(t) - (1-\alpha)\frac{\Delta t}{2}u_t(t)$$

$$+ (1-\alpha)^2\frac{\Delta t^2}{2}u_{tt}(t) = (1-\alpha)\frac{\Delta t}{2}u_t(t+\Delta t)$$

$$+ (1-\alpha)^2\frac{\Delta t^2}{2}u_{tt}(t+\Delta t) + 0(\Delta t^3)$$

  where α is an *implicitness parameter*, $0 \leq \alpha \leq 1$: if $\alpha = 0$ the scheme is explicit (or "weakly" implicit) and if $\alpha > 0$, the scheme is implicit.

- For an explicit scheme to be numerically stable, it must satisfy a CFL condition of the type

$$\frac{c\Delta t}{|u|h_e} < 1$$

  (c a constant) so that if h_e is small (to maintain accuracy), then Δt is also small to maintain stability requirements. These explicit schemes can be very slow for fine meshes, requirring many time steps.

- A better idea is to *fix the time step*:

$$\Delta t = \Delta t_{\text{optimal}}$$

and to use explicit schemes in those elements in which the CFL condition is satisfied and implicit schemes in those elements where the condition is violated, both for a fixed time step. The result is that the flow domain Ω is partitioned into two subdomains, Ω_E and Ω_I, with explicit algorithms used in Ω_E and implicit algorithms used in Ω_I.

- It has been found through numerical experimentation that such explicit/implicit schemes can result in 40- to 60-percent savings in computational time over purely explicit methods for representative flow problems.

- In the spirit of *optimal control* of the computational process mentioned in the introduction, explicit/implicit schemes are being incorporated into *ADAPT®* which attempt to choose $\Delta t_{\text{optimal}}$ so as to minimize the computational cost functions,

$$C = C(\Delta t) = \frac{\text{cost in advancing a numerical solution}}{\text{real time}}$$

  By assuming that the number of elements per time step is fixed, the above cost functions can be formulated in terms of a time step and $\Delta t_{\text{optimal}}$ can be obtained.

6 Concluding Comments

New developments in adaptive finite element methods have made possible significant advances in computational fluid dynamics. These techniques have also reached a level of maturity that permits their implementation in general–purpose CFD codes. Results suggest that such codes may soon be added to the analysis tools available for practical engineering analyses. In addition, adaptive methods provide quantitative measures of reliability of engineering calculations, as they naturally yield elementwise estimates of computational error. These features represent an important step toward optimal computation procedures: those which produce the best results in some sense for the least effort and/or cost. Equally important, they relieve the user from traditional responsibilities of making *ad hoc* decisions about mesh structure and even algorithm selections.

Acknowledgements

The examples presented in this work are the result of collaborations with several colleagues over the past five years. Much of our work at TICOM and at the Computational Mechanics Company is supported by contracts from the NASA Langley, Marshall and Ames offices. Theoretical work on error estimation and *h-p* methods was initiated under ONR support and continues under contracts with the NASA Marshall Space Flight Center and the NASA Langley Research Center.

7 References

1. Oden, J. T., "A General Theory of Finite Elements. Part 1. Topological Considerations," *International Journal of Numerical Methods in Engineering*, Vol. 1, pp. 147–159, 1969 and "Part 2. Applications," Vol. 1, No. 3, pp. 247–259, 1969.

2. Oden, J. T., and Symogyi, D., "Finite Element Applications in Fluid Dynamics," *Journal of the Engineering Mechanics Division*, ASCE, Vol. 95, No. EM3, June, pp. 821–826, 1969.

3. Oden, J. T., "A Finite Element Analogue of the Navier-Stokes Equations," *Journal of the Engineering Mechanics Division*, ASCE, Vol. 96, EM4, 1970.

4. Oden, J. T., "The Finite Element Method in Fluid Mechanics," **Lectures on Finite Elements in Continuum Mechanics**, Edited by J. T. Oden and E. R. A. Oliveira, Proceedings, NATO Advanced Study Institute, Lisbon, 1971, and UAH Press, Huntsville, pp. 151–186, 1973.

5. Oden, J. T., "Finite Element Formulation of Problems of Finite Deformation and Irreversible Thermodynamics of Nonlinear Continua—A Survey and Extensions of Recent Developments," **Recent Advances in Matrix Methods of Structural Analysis and Design**, Edited by R. H. Gallagher, Y. Yamada, and J. T. Oden, Univ. of Huntsville Press, pp. 693–724, 1971.

6. Oden, J. T., and Wellford, L. C. Jr., "Analysis of Flow of Incompressible Viscous Fluids by the Finite Element Method," *AIAA Journal*, Vol. 10, No. 12, pp. 1590–1599, 1972.

7. Babuška, I., Zienkiewicz, O. C., Gago, J., and Oliveira, E. R. A. (Eds.), **Accuracy Estimates and Adaptive Refinements in Finite Element Computations**, John Wiley and Sons, Ltd., Chichester, 1986.

8. Oden, J. T., and Demkowicz, L., "Advances in Adaptive Improvements: A Survey of Adaptive Methods in Computational Fluid Mechanics," **State of the Art Surveys in Computational Mechanics**, Edited by A. K. Noor and J. T. Oden, American Society of Mechanical Engineers, N.Y., 1988.

9. Oden, J. T., "Adaptive Finite Element Methods for Problems in Solid and Fluid Mechanics," **Finite Element Theory and Application Overview**, Edited by R. Voight, Springer-Verlag, N.Y., 1988.

10. Oden, J. T., "Adaptive FEM in Complex Flow Problems," **The Mathematics of Finite Elements with Applications**, Edited by J. R. Whiteman, London Academic Press, Lt., Vol. 6, pp. 1–29, 1988.

11. Oden, J. T., "Smart Algorithms and Adaptive Methods in Computational Fluid Dynamics," **Proceedings CANCAM (Canadian Congress on Applied Mechanics)**, Ottowa, Canada, 1989.

12. Oden, J. T., "Progress in Adaptive Methods in Computational Fluid Dynamics," **Adaptive Methods for Partial Differential Equations**, Ed. by J. Flaherty, et al., SIAM Publications, Philadelphia, 1989.

13. Oden, J. T., Strouboulis, T., and Devloo, P., "Adaptive Finite Element Methods for the Analysis of Inviscid Compressible Flow: I. Fast Refinement/Unrefinement and Moving Mesh Methods for Unstructured Meshes," *Computer Methods in Appl. Mech. and Engrg.*, Vol. 59, No. 3, 1986.

14. Oden, J. T., Strouboulis, T., and Devloo, P., "Adaptive Finite Element Methods for Compressible Flow Problems," **Numerical Methods for Compressible Flows—Finite Difference, Element and Volume Techniques**, Edited by T. E. Tezduyar and T. J. R. Hughes, AMD–Vol. 78, ASME, New York, pp. 115–126, 1987.

15. Oden, J. T., Strouboulis, T., and Devloo, P., "Adaptive Finite Element Methods for High–Speed Compressible Flows," *International Journal for Numerical Methods in Fluids*, Vol. 7, pp. 1211–1228, 1987.

16. Oden, J. T., Bass, J. M., Huang, C. Y., and Berry, C. W., "Recent Results on Smart Algorithms and Adaptive Methods for Two– and Three–Dimensional Problems in Computational Fluid Mechanics," *Computers and Structures*, (to appear).

17. Rachowicz, W., Oden, J. T., and Demkowicz, L., "Toward a Universal *h-p* Adaptive Finite Element Strategy, Part 3. A study of the Design of *h-p* Meshes," *Computer Methods in Applied Mechanics and Engineering*, **77**, pp. 181–212, 1989.

18. Oden, J. T., "Smart Algorithms and Adaptive Methods in Fluid–Structure Interaction," *AIAA Aerospace Sciences Meeting*, AIAA-90-0577, Reno, NV, 1990.

19. Demkowicz, L., Oden, J. T., Rachowicz, W., and Hardy, O., "Toward a Universal *h-p* Adaptive Finite Element, Part 1. Constrained Approximation and Data Structures," *Computer Methods in Applied Mechanics and Engineering*, **77**, pp. 79–112, 1989.

20. Bass, J. M., and Oden, J. T., "Adaptive Computational Methods for Chemically–Reacting Radiative Flows," *International Journal of Engineering Science*, Vol. 26, No. 9, pp. 959–992, 1988.

21. Oden, J. T., Demkowicz, L., Westermann, T., and Rachowicz, W., "Toward a Universal *h-p* Adaptive Finite Element Strategy, Part 2. *A Posteriori* Error Estimates," *Computer Methods in Applied Mechanics and Engineering*, **77**, pp. 113–180, 1989.

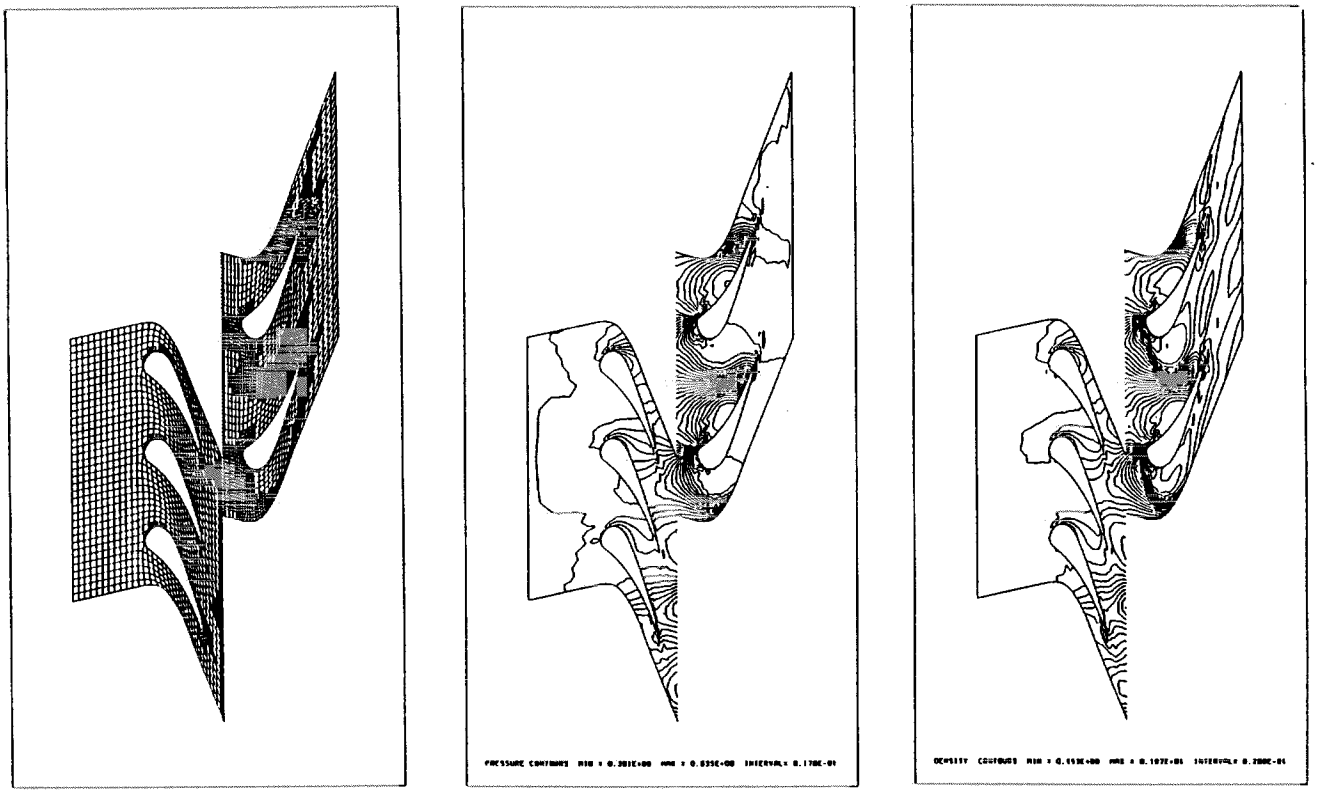
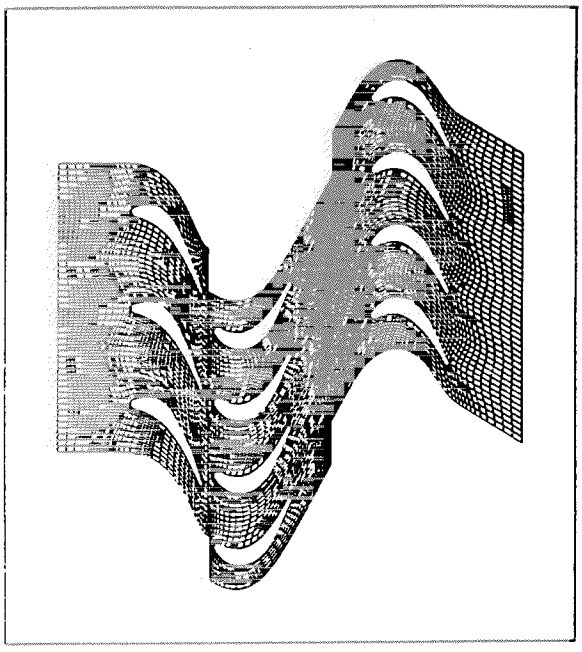
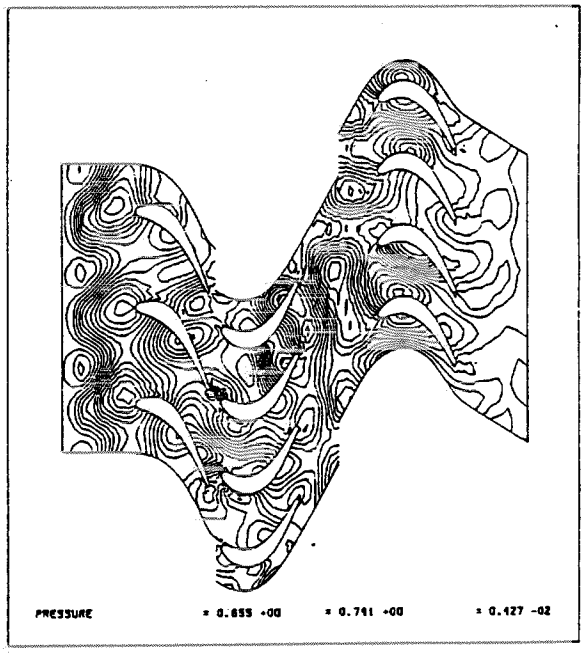


Figure 1: Simulation of transonic cascade flow past moving turbine blades in a turbine engine. From left to right: a) an instantaneous adapted mesh showing optimal distribution of mesh sizes on either side of meshes moving relative to one another along a sliding interface, b) pressure in contours and c) density contours. The number of elements in the mesh at this time instant (200 time steps) is 1152, around 50 percent of an equivalent fine mesh capable of producing the same resolution of the flow variables. Flow conditions: inflow Mach number = 0.16, relative blade velocities: $v_G = v_L - v_R = 0.16$.





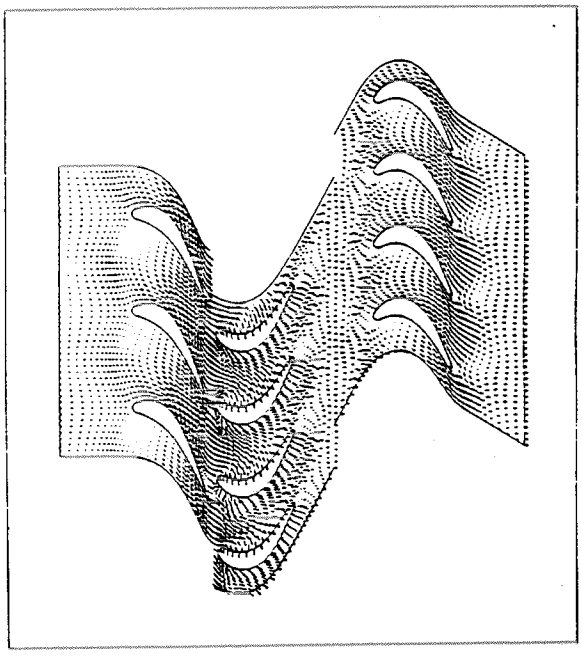


Figure 2: Another dynamic rotor–stator flow interaction calculation, this case involving three moving interfaces: multiple turbine blades and multiple stages. Shown here are the adapted h-adapted mesh, pressure contours, and velocity vectors after 1100 time steps for a transonic cascade flow simulation.

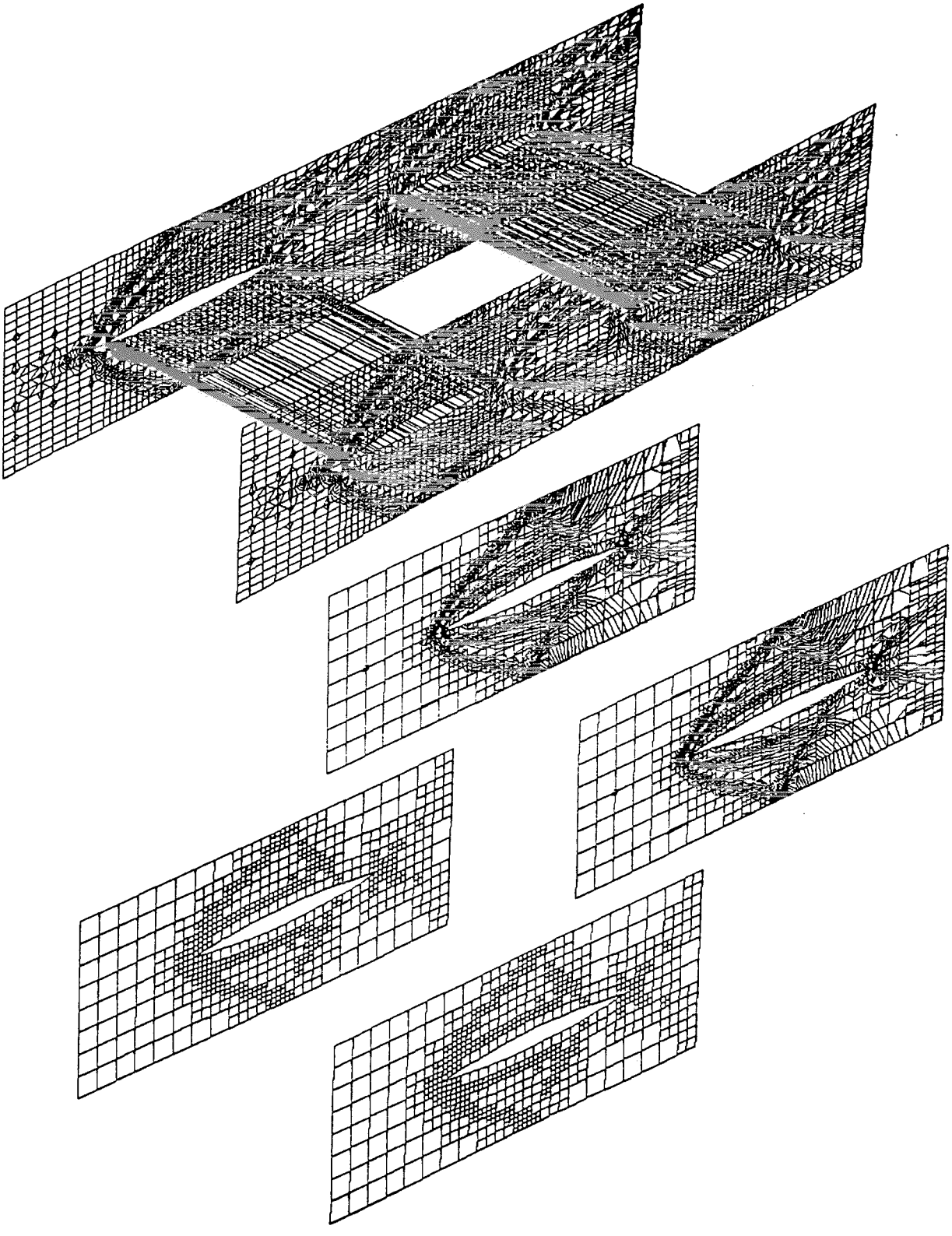
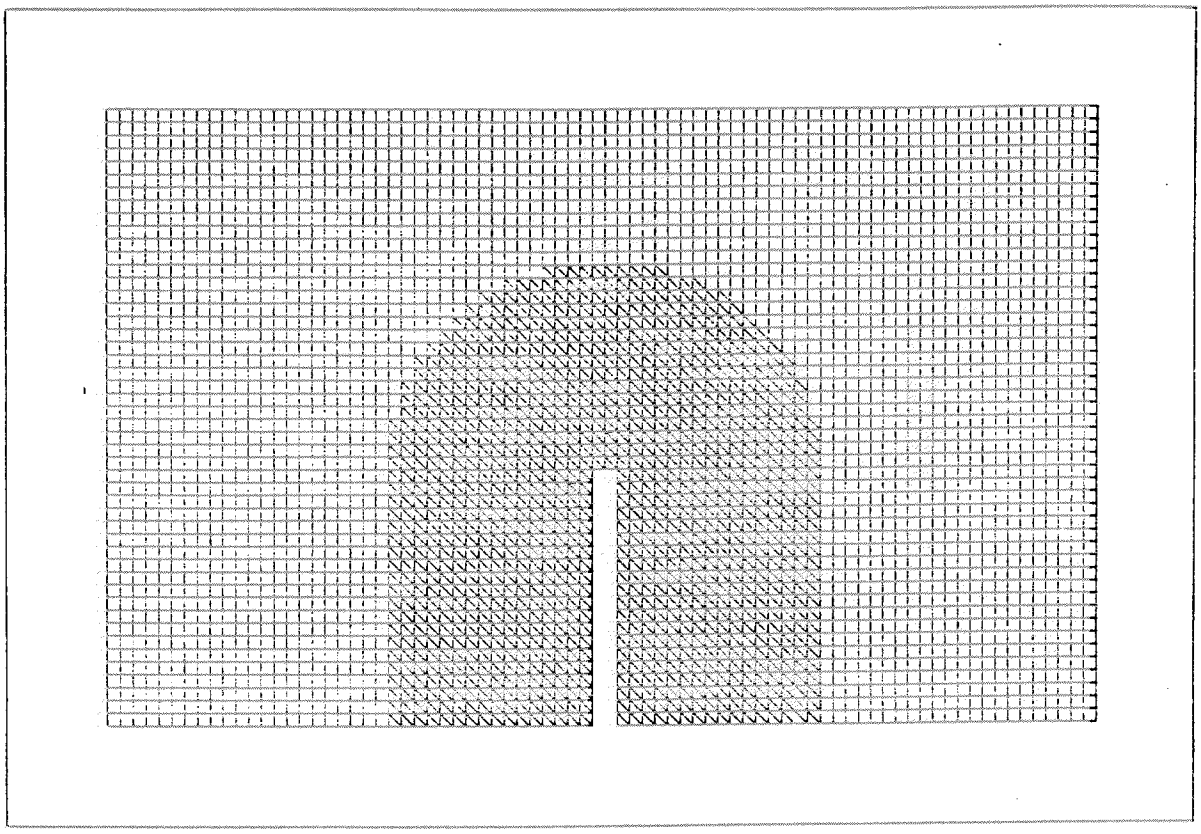
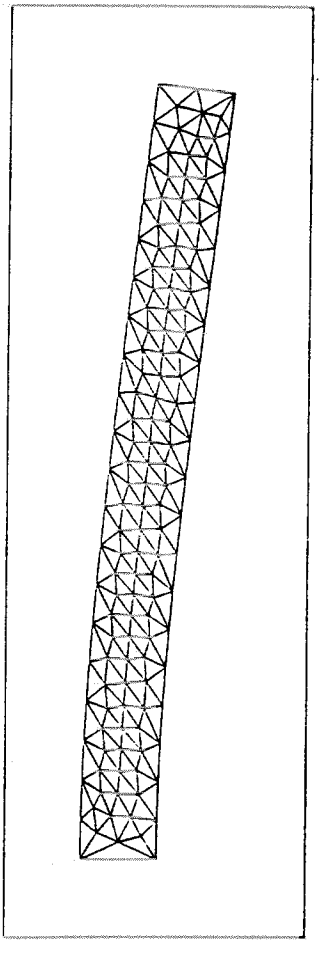


Figure 3: A three–dimensional rotor–stator flow simulation performed with *ADAPT*®-3D (see [16]).





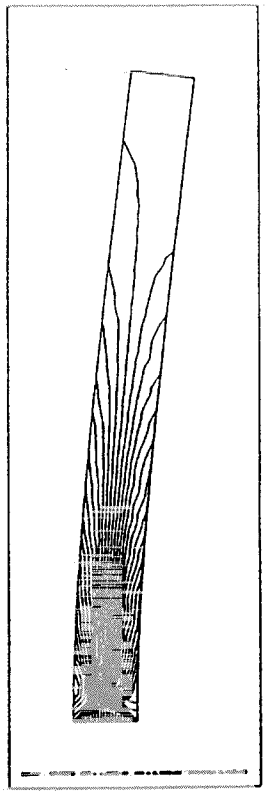


Figure 4: Example of three different types of meshes for a model fluid–structure interaction simulation: an outside Eulerian mesh through which a compressible fluid flows, a transition referential mesh, and a Lagrangian mesh to model deformation of an elastic beam in the flowfield. Pressures and shear stresses developed by the fluid on the beam cause it to deform. Instantaneously, contours of principal stresses (shown on the deformed beam) are obtained.

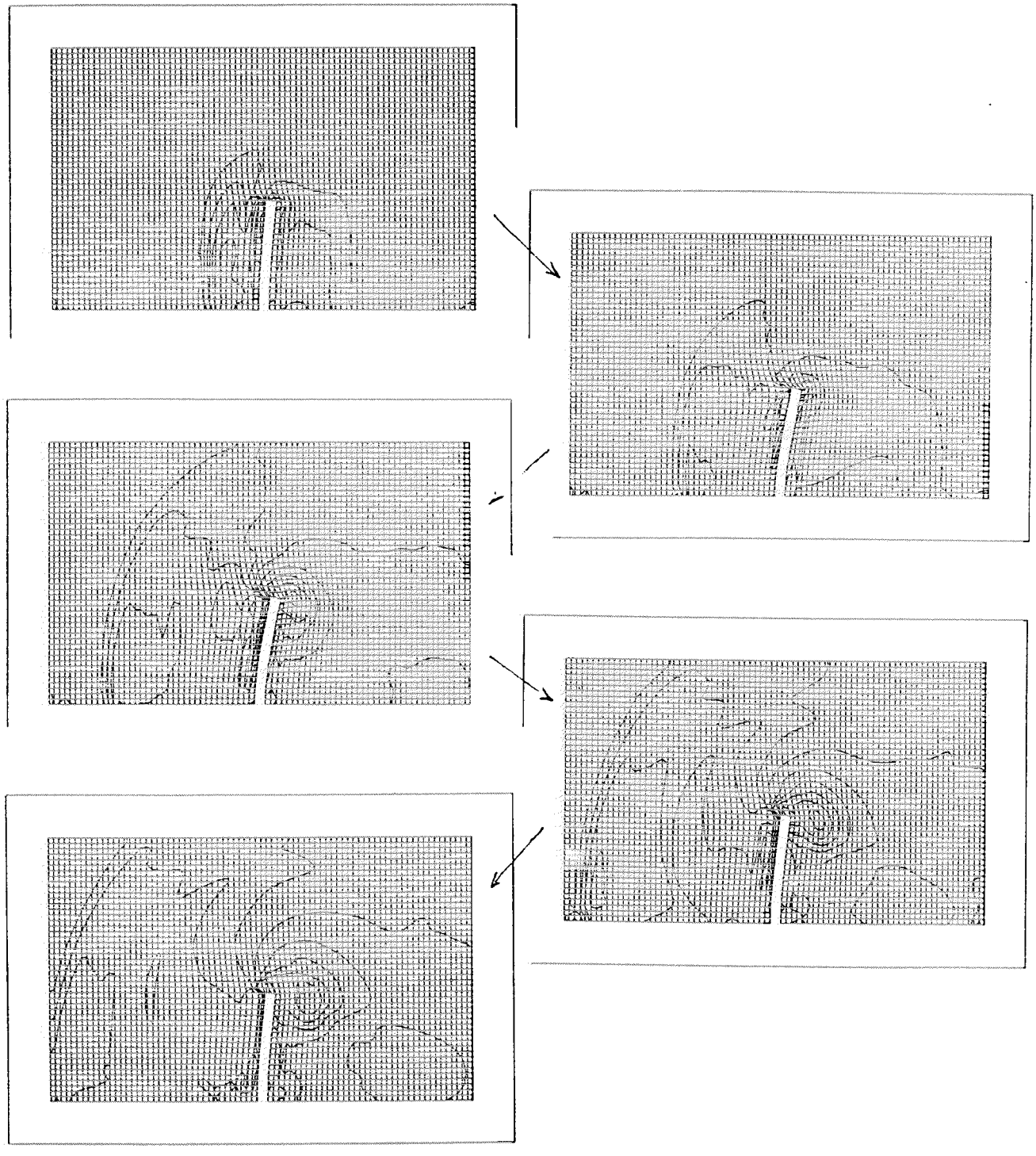


Figure 5: Time history of motion of an elastic beam undergoing large-amplitude motions in a viscous compressible flowfield. Shown are contours of density at various times, one frame per 40 time steps is shown. Note the bow shock developing and moving out of the flow domain and the development of a vortex on the downstream side of the beam.

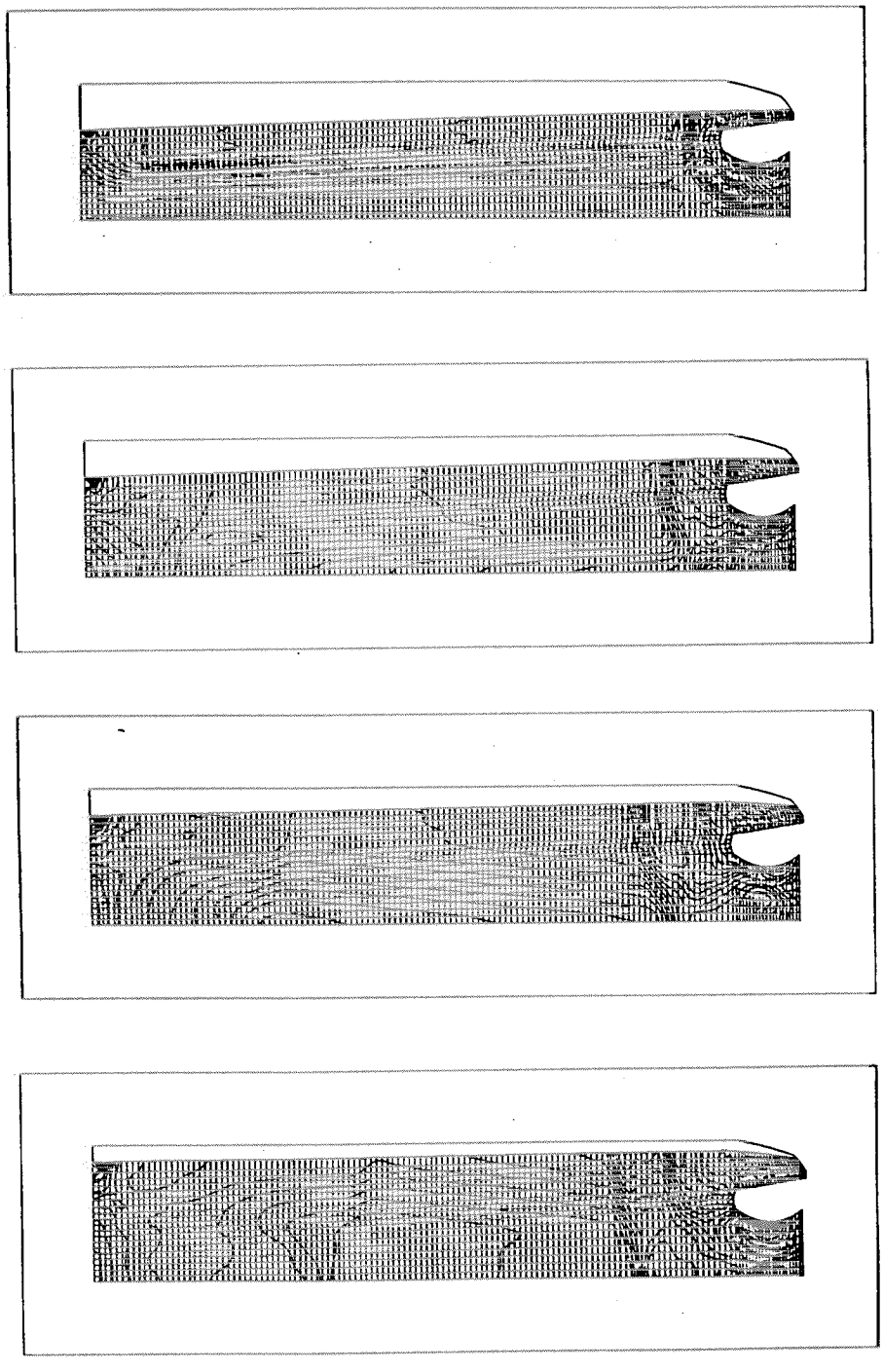


Figure 6: Example of a flow calculation on a domain with a moving boundary: burning of a solid propellant boundary with a hot gas flowing from left to right through the flow domain. Contours of gas density are shown.

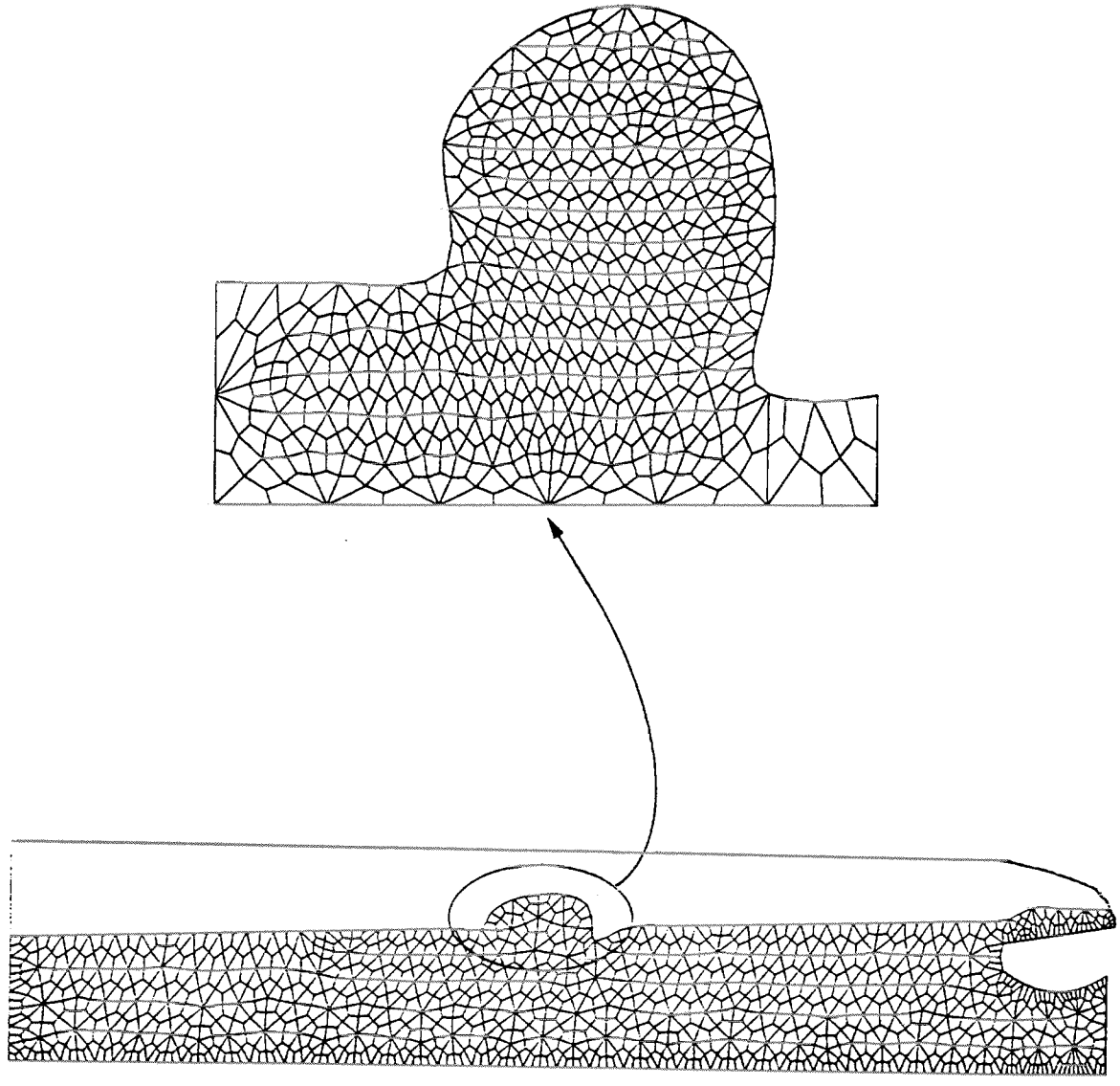


Figure 7: Unstructured moving mesh simulating an erosion pocket growing in a solid–propellant motor case.

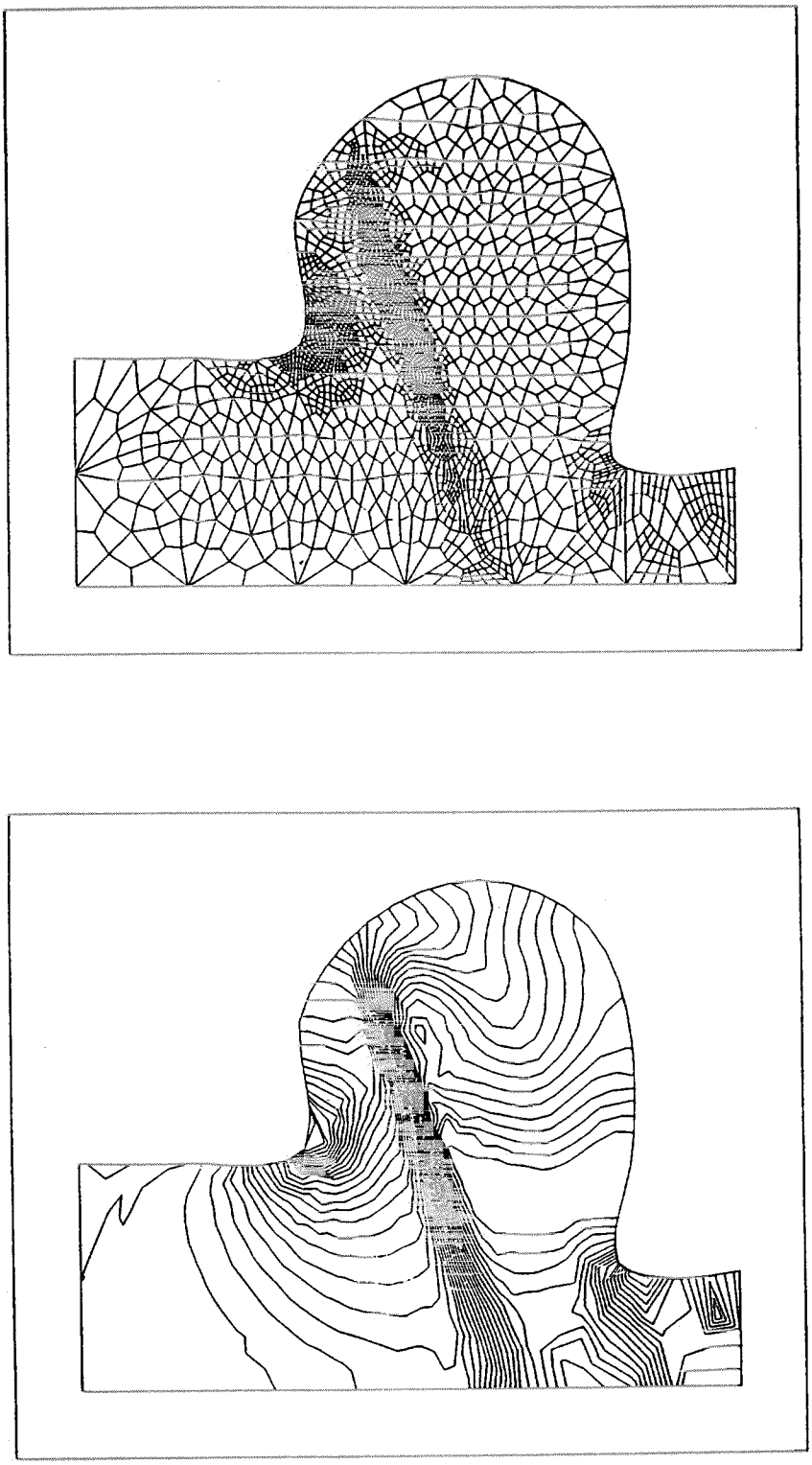


Figure 8: a) Simultaneous h-refinement around shock developed in erosion pocket and b) contours of gas density. This is an example of an adaptive h-r method as the boundary nodes are changing in location after each time step while the mesh is being refined to control solution error.

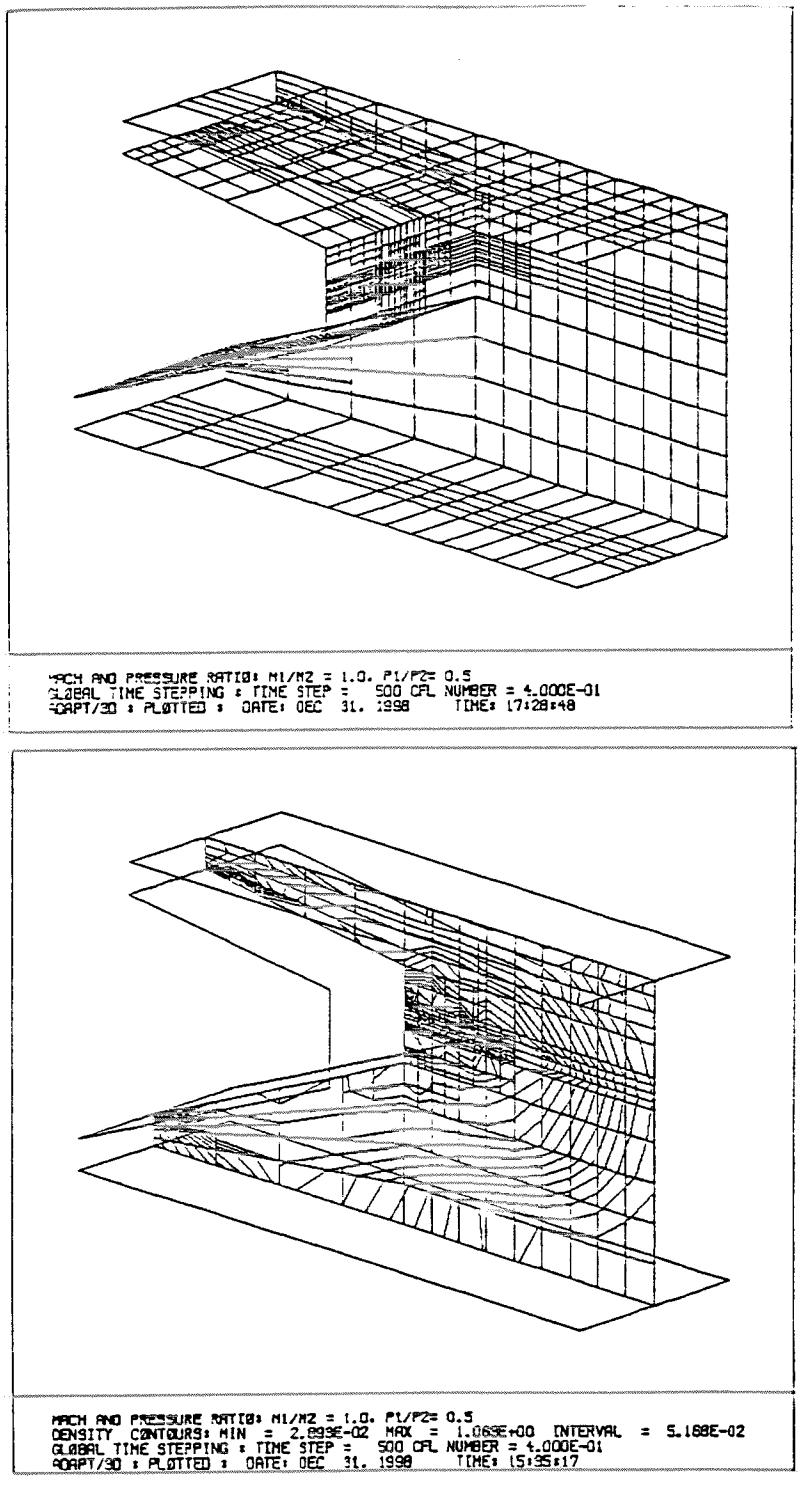


Figure 9: Three–dimensional, chemically–reacting flow simulation. Shown here is a coarse mesh model of a chemical laser simulation in which the adaptive strategy generates a fine mesh near the laser nozzle to capture fine–scale chemical kinetics: a) a refined mesh, b) density contours. The time scale of the chemical reactions is several orders-of-magnitude smaller than that of the compressible flowfield. This generally requires the use of implicit integration schemes for the chemistry modeling while explicit/implicit methods are used for the flow calculation.

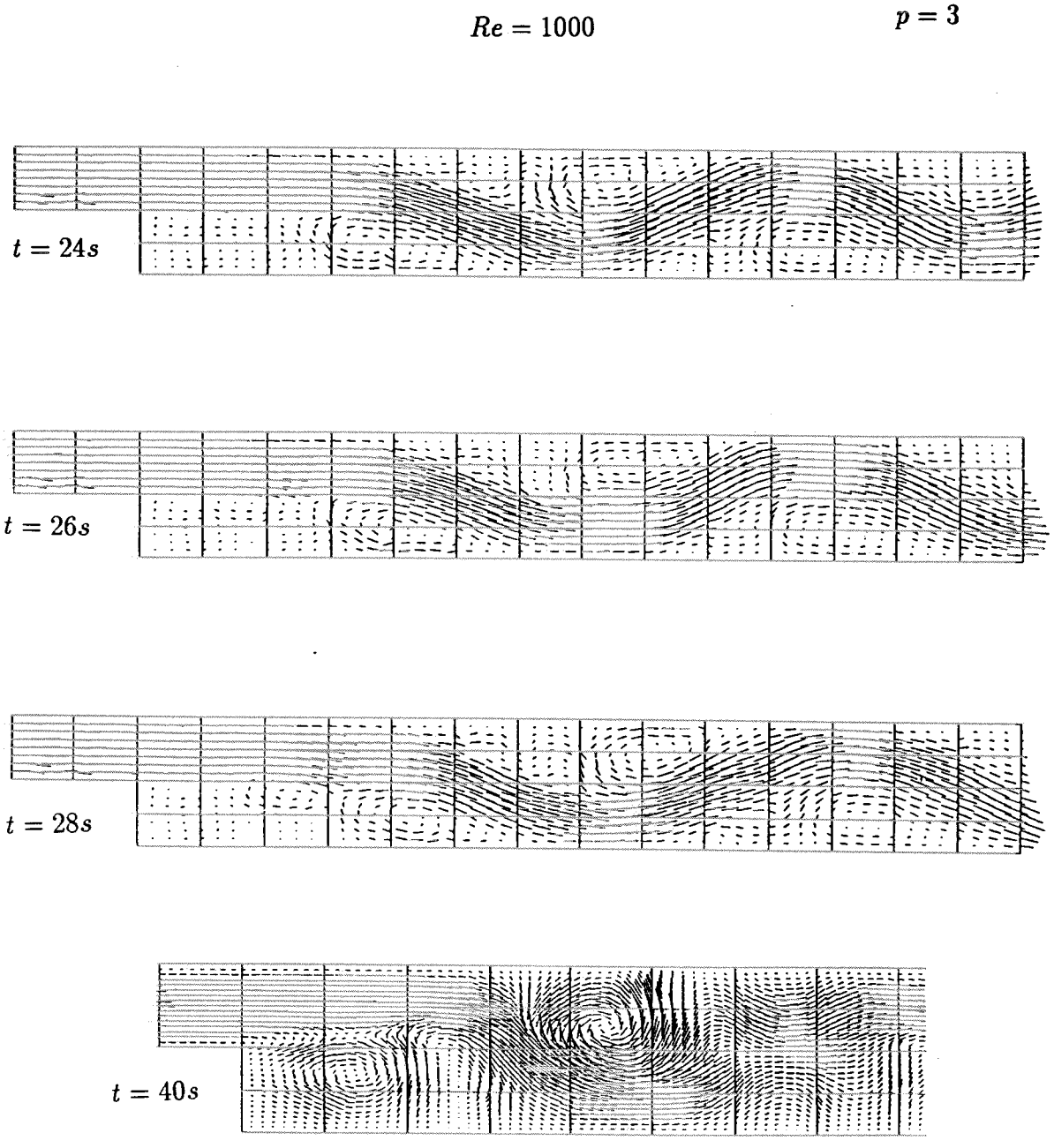


Figure 10: A coarser mesh simulation of transient incompressible flow over a step at a Reynolds number of 1000. Shown is the progressive development of the flowfield at three time steps on a cubic ($p = 3$) mesh and, finally, an unusually detailed simulation of complex flow structures for a $p = 4$ mesh at a later time.

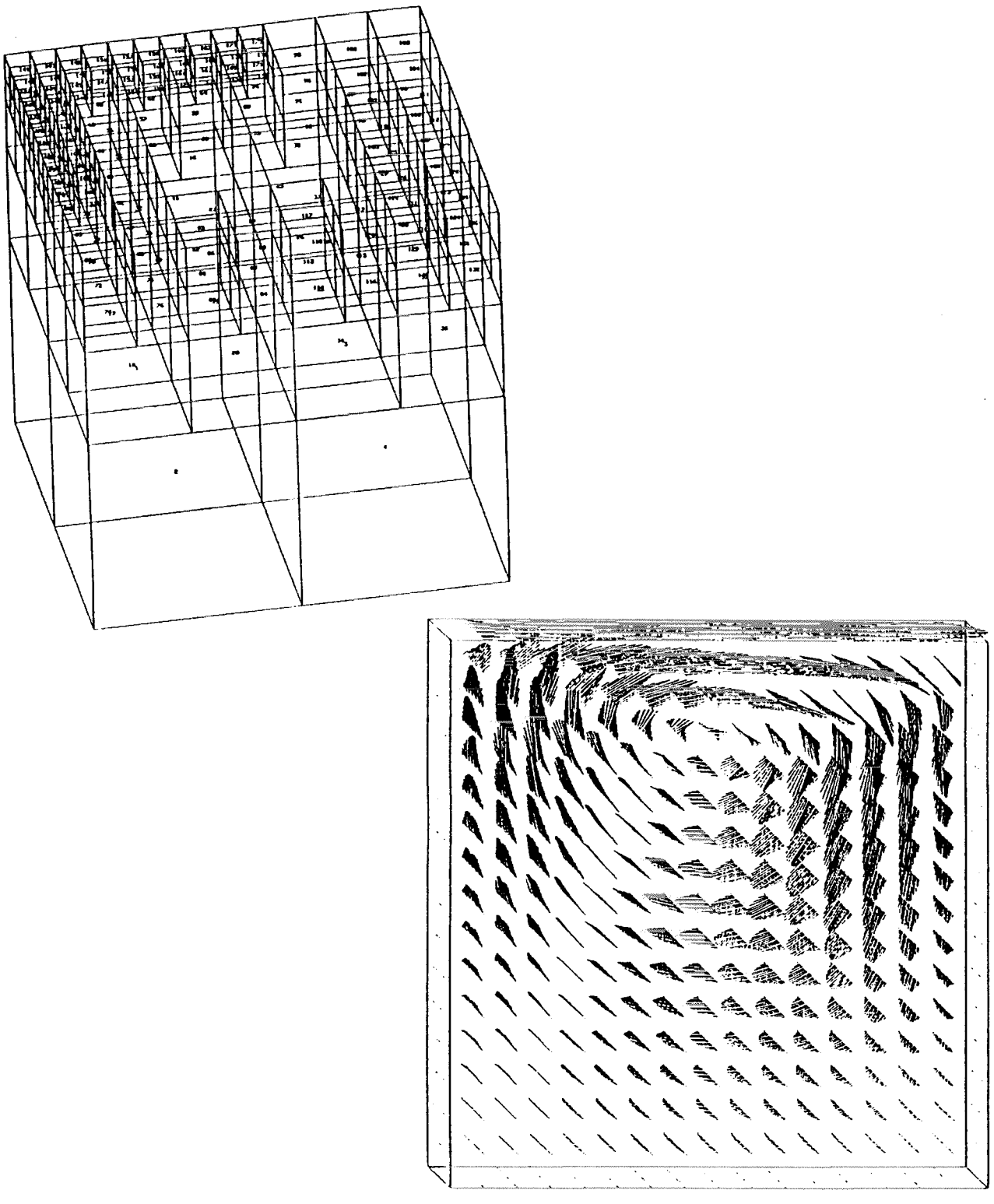


Figure 11: A three–dimensional adapted h-p mesh for Stokesian flow over a rectangular cavity: a) the adapted mesh for a uniform p of $p = 3$ and b) computed velocity–vector distribution.

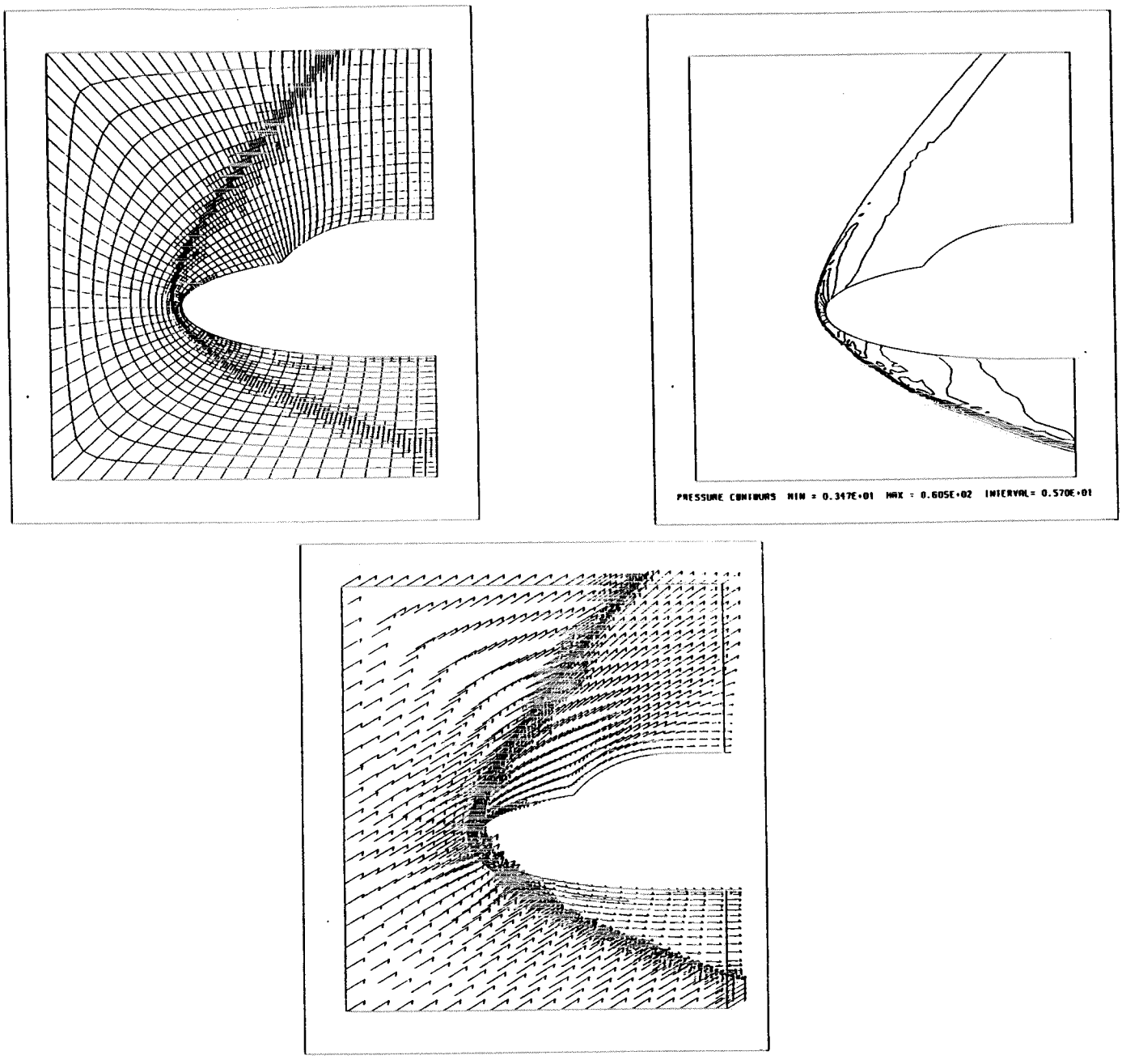


Figure 12: Supersonic flow past a double–ellipse shuttle–like body for flow at a thirty degree angle of attack. These calculations were done using the h-adaptive flow solver and the adaptive explicit–implicit scheme: a) the adapted mesh showing fine–grid resolution of the shock, b) pressure contours, and c) velocity vector distributions. Here an optimal time step is selected successively throughout the evolution from a uniform flowfield to a steady state flow. The mesh size is selected dynamically to meet accuracy requirements while the time step is calculated to maintain numerical stability at minimum computational cost; each element may be assigned a different type of time–marching scheme, implicit or explicit. Often only a few small elements need be advanced using implicit schemes to substantially reduce computing times.