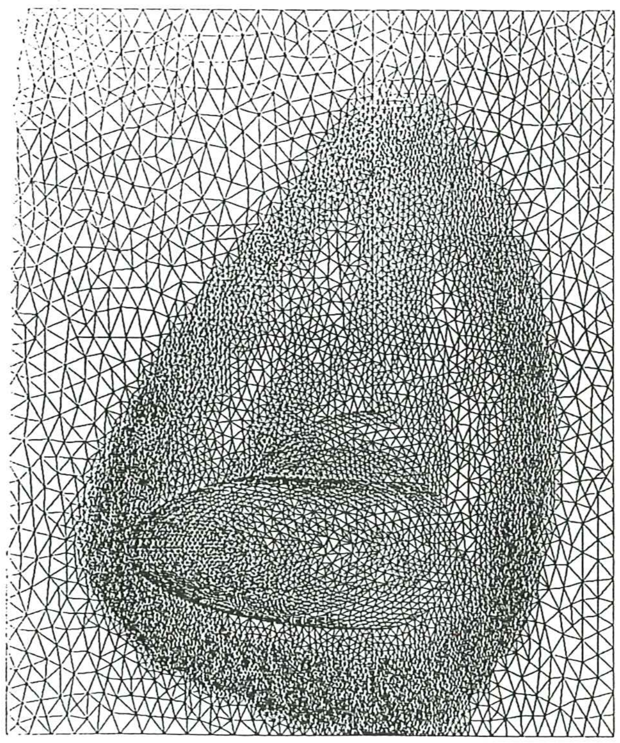
Adaptive Finite Element Computations of Hypersonic Inviscid Flows around a Double Ellipsoid

T. Fischer

E. Oñate



Publication CIMNE Nº 60, February 1995

Adaptive Finite Element Computations of Hypersonic Inviscid Flows around a Double Ellipsoid

T. Fischer
E. Oñate

Publication CIMNE Nº 60, February 1995

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

Adaptive Finite Element Computations of Hypersonic Inviscid Flows around a Double Ellipsoid

T. Fischer and E. Oñate

International Centre for Numerical Methods in Engineering, Universidad Politécnica de Cataluña, Gran Capitán s/n, 08034 Barcelona, Spain

1 ABSTRACT

This report documents the potential capabilities of adaptive inviscid flow calculations on unstructured meshes in three dimensions using the finite element method. The finite element formulation of the compressible Euler and Navier-Stokes equations is based on a two-step explicit Taylor-Galerkin scheme. Adaptive remeshing is applied to enhance the numerical solution in the vicinity of shocks. Particular emphasis is put on the generation of unstructured tetrahedral meshes as well as in the discussion of estimating the error in the numerical solution. Finally, an application to a well known 3D high speed flow problem is shown where adaptive remeshing techniques become essential to keep the computational cost within reasonable limits. Its results are compared to some seemingly best reference solutions using other algorithms.

2 INTRODUCTION

The Taylor-Galerkin approach has proved to be an efficient numerical procedure which is capable of simulating inviscid and laminar viscous high speed flows around complex geometries in two and three dimensions. The spatial discretization is typically based on the use of linear triangular finite elements (2D) or linear tetrahedral elements (3D). The time discretization involves two steps of a Taylor expansion of second order. The advantage of the use of a two-step algorithm with a lumped mass matrix formulation is a reduction of stored memory [1] resulting in a lean explicit algorithm. Artificial dissipation of Lapidus [2] or Jameson [3] is typically added to account for discontinuities such as shocks or boundary layers. To accelerate the convergence to steady state local time stepping may be used.

3 BASIC THEORY

3.1 Euler and Navier-Stokes equations

The Euler and Navier-Stokes set of equations for a compressible fluid in conservative form for laminar flux within a three dimensional domain without source terms read:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}_i}{\partial x_i} = \frac{\partial \mathbf{g}_i}{\partial x_i} \qquad i = 1,3 \tag{1}$$

where the nodal unknowns $\mathbf{u}$, the advective fluxes $\mathbf{f}_i$ and the viscous fluxes $\mathbf{g}_i$ are:

$$\mathbf{u} = \left\{ \begin{array}{c} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho \epsilon \end{array} \right\} \quad \mathbf{f}_i = \left\{ \begin{array}{c} \rho u_i \\ \rho u_i u_1 + p\delta_{1i} \\ \rho u_i u_2 + p\delta_{2i} \\ (\rho\epsilon + p) u_i \end{array} \right\} \quad \mathbf{g}_i = \left\{ \begin{array}{c} 0 \\ \sigma_{1i} \\ \sigma_{2i} \\ k\,\frac{\partial T}{\partial x_i} + u_j\,\sigma_{ji} \end{array} \right\} \tag{2}$$

Assuming a bulk viscosity of $(3\lambda + 2\mu) = 0$, the law of Navier-Poisson for a Newtonian fluid leads to the components of the viscous stress tensor:

$$\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{3}$$

In above: ρ is the density, p is the pressure, T is the temperature, u_i Cartesian components of the velocity, ϵ the total energy per unit mass, δ_{ij} the Kronecker delta, k is the material heat conductivity, and $\mu = \mu(T)$ is the dynamic coefficient of viscosity which is calculated from Sutherland's equation.

The pressure is obtained from the equation of state for a perfect gas:

$$p = (\gamma - 1)\rho(\epsilon - 0.5 u_j u_j)$$

where $\gamma = C_p/C_v$ is the ratio of specific heats . In the present computations we have taken $\gamma = 1.4$ which comes closest to air. In case of inviscid calculations, the viscous fluxes $\mathbf{g}_i$ are set to zero to obtain the Euler equations.

3.2 Taylor-Galerkin algorithm

The scheme chosen here is the well known Taylor-Galerkin method, successfully developed and used by different authors [1,4].

Non structured meshes of linear triangular finite elements are used [1].

Outline of the algorithm

The solution vector $\mathbf{u}$ is expanded in Taylor series as:

$$\Delta\mathbf{u} = \mathbf{u}^{n+1} - \mathbf{u}^n = \Delta t\frac{\partial \mathbf{u}^n}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 \mathbf{u}^n}{\partial t^2} + O(\Delta t^3) \tag{4}$$

where the superscript n indicates solution at time $n\Delta t$, etc. Hence using equation (1) and neglecting the $O(\Delta t^3)$ term the expansion reads:

$$\Delta \mathbf{u} = \Delta t \left(\frac{\partial \mathbf{g}_j}{\partial x_j} - \frac{\partial \mathbf{f}_i}{\partial x_i} \right)^n - \frac{\Delta t^2}{2} \left[\frac{\partial}{\partial x_i} \left(\mathbf{G}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right) - \frac{\partial}{\partial x_i} \left(\mathbf{A}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right) \right]^n \qquad (5)$$

where $\mathbf{A}_i = \partial \mathbf{f}_i / \partial \mathbf{u}$ and $\mathbf{G}_i = \partial \mathbf{g}_i / \partial \mathbf{u}$ are matrices of vector gradients, and where derivatives or products of order higher than two have been neglected.

Equation (5) can be now discretized using C_0 finite elements [4]:

$$\mathbf{u}^n = \sum_i N_i \mathbf{a}_i^n = \mathbf{N}\mathbf{a}^n \qquad (6)$$

Where $\mathbf{N}_i$ is the interpolation (linear) function of node i and $\mathbf{a}_i^n$ the nodal values of the unknown $\mathbf{u}$ at time n.

The integral form of the discretized equation, where standard Galerkin weighting functions are used, may be written as:

$$\int_\Omega \mathbf{N}^T \Delta \mathbf{u} d\Omega =$$

$$\Delta t \int_\Omega \mathbf{N}^T \left[\frac{\partial \mathbf{g}_j}{\partial x_j} - \frac{\partial \mathbf{f}_i}{\partial x_i} \right]^n d\Omega - \frac{\Delta t^2}{2} \int_\Omega \mathbf{N}^T \left[\frac{\partial}{\partial x_i} \left(\mathbf{G}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right) - \frac{\partial}{\partial x_i} \left(\mathbf{A}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right) \right]^n d\Omega \qquad (7)$$

Where $\Delta \mathbf{u} = \mathbf{u}^{n+1} - \mathbf{u}^n$. At this point it should be noted that $\mathbf{A}_i$ and $\mathbf{G}_i$ are matrices of vector gradients that have to be computed and stored (time and memory consuming task).

Integration by parts using Greens theorem in equation (7) gives:

$$\int_\Omega \mathbf{N}^T \Delta \mathbf{u} d\Omega =$$

$$\Delta t \left\{ \int_\Omega \mathbf{N}^T \left(- \frac{\partial \mathbf{f}_i}{\partial x_i} \right) d\Omega - \int_\Omega \frac{\partial \mathbf{N}^T}{\partial x_j} \mathbf{g}_j d\Omega + \int_\Gamma \left(\mathbf{N}^T \mathbf{g}_j \right)_n d\Gamma \right\}^n - \qquad (8)$$

$$- \frac{\Delta t^2}{2} \left\{ \int_\Omega \frac{\partial \mathbf{N}^T}{\partial x_i} \left(\mathbf{A}_i \frac{\partial \mathbf{f}_k}{\partial x_k} - \mathbf{G}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right) d\Omega - \int_\Gamma \mathbf{N}^T \left(\mathbf{A}_i \frac{\partial \mathbf{f}_k}{\partial x_k} - \mathbf{G}_i \frac{\partial \mathbf{f}_k}{\partial x_k} \right)_n d\Gamma \right\}^n$$

Subscript n indicates normal projections. Superscript n means values at time n. The finite element interpolation leads to a system of algebraic equations:

$$\mathbf{M}\,\Delta\mathbf{u}^n \;=\; \mathbf{RHS}^n \tag{9}$$

$$\text{where} \quad \mathbf{M}_{ij} = \int_\Omega \mathbf{N}_i\mathbf{N}_j d\Omega \tag{10}$$

To preserve the explicitness of the algorithm when a steady state solution is to be obtained, equation (9) can be solved using a lumped mass matrix. In other cases, a Jacobi-type iteration procedure may be performed to recover the properties of the full mass matrix [1].

Two-step algorithm

To avoid the computation and storage of $\mathbf{A}_i$ and $\mathbf{G}_i$, an alternative algorithm has been developed. In the first step, a Taylor expansion at time $n+1/2$ is performed (neglecting at this stage the viscous terms):

$$\mathbf{u}^{n+1/2} \simeq \mathbf{u}^n + \frac{\Delta t}{2}\frac{\partial \mathbf{u}^n}{\partial t} = \mathbf{u}^n - \frac{\Delta t}{2}\frac{\partial \mathbf{f}_i^n}{\partial x_i} \tag{11}$$

From this step $\mathbf{u}^{n+1/2}$ is obtained. For the second step, again the use of Taylor expansion for $\mathbf{f}_i^{n+1/2}$ and equation (1) neglecting second order derivatives, provides the tools to replace $\mathbf{A}_i$ and $\mathbf{G}_i$ by values of fluxes at time $n+1/2$ [1]. Replacing in equation (8) $\mathbf{A}_i$ and $\mathbf{G}_i$ by the values obtained and integrating by parts $\partial \mathbf{f}_i/\partial x_i$, we have:

$$\left(\int_\Omega \mathbf{N}^T\mathbf{N}d\Omega\right)\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) \;=\; \mathbf{M}\,\Delta\mathbf{u} \;=$$

$$\Delta t\left[\int_\Omega \frac{\partial \mathbf{N}^T}{\partial x_k}\left(\mathbf{f}_i^{n+1/2} - \mathbf{g}_j^n\right)d\Omega - \int_\Gamma \mathbf{N}^T\left(\mathbf{f}_i^{n+1/2} - \mathbf{g}_j^n\right)_n d\Gamma\right] \tag{12}$$

Subscript i and j indicate mean nodal values and elemental values respectively. Recall that subscript n refers to boundary normal values.

Stability

The stability of the algorithm is preserved if the time step meets the following criteria, for the viscous case:

$$\Delta t_{crit} = \frac{Ch_e}{(|\mathbf{v}| + c) + \frac{4\mu}{\rho_{min}\mathrm{PrRe}h_e}} \qquad (13)$$

for the inviscid case:

$$\Delta t_{crit} = \frac{Ch_e}{(|\mathbf{v}| + c)} \qquad (14)$$

where c the local speed of sound, C is the Courant number, μ is the dynamic viscosity, ρ_{min} is the minimum element density, $\mathbf{v}$ is the velocity vector, and h_e is a mesh spacing value taken to be the minimum of the element. The Courant number is chosen to be less than 1 to ensure the stability of the explicit Taylor-Galerkin algorithm.

In order to enhance the performance of the program and to accelerate the solution towards steady state, local time stepping can be used.

Artificial dissipation

To account for discontinuities in the solution of fluid flow using a high order scheme (e.g. Taylor expansion) a certain kind of artificial viscosity or diffusion must be introduced. First an algorithm of the Lapidus kind [2] and then an approach similar to Jameson [3] are described.

a) *Lapidus Algorithm*

It can be shown that for the ith node of an unidimensional solution, a second derivative of the unknown can be obtained as:

$$C\left[\frac{\partial}{\partial x}(h^2\frac{\partial \phi}{\partial x})\right]_i = [\mathbf{M}_l^{-1}(\mathbf{M}_c - \mathbf{M}_l)\phi]_i \qquad (15)$$

$\mathbf{M}_l$ is the diagonalized or lumped mass matrix, $\mathbf{M}_c$ is the consistent mass matrix, C is a constant, and ϕ is the unknowns variable.

An extension of this idea to a multidimensional problem gives:

$$\Delta \mathbf{u}_s = \mathbf{u}_s^{n+1} - \mathbf{u}^{n+1} = \Delta t \frac{\partial}{\partial \mathbf{l}}\left(k^\star \frac{\partial(\mathbf{v}^{n+1})}{\partial \mathbf{l}}\right) \qquad (16)$$

$k^\star$ is variable to determine the amount of viscosity necessary, $\mathbf{l}$ is the normalized vector of velocity gradients, $\Delta \mathbf{u}_s$ are incremental smoothed values, $\mathbf{u}_s^{n+1}$ is the smoothed solution at time n+1, and $\mathbf{v}$ is the vector of velocity.

$k^\star$ and vector $\mathbf{l}$ are obtained as follows:

$$k^\star = C_k (h^{el})^2 \left| \frac{\partial(\mathbf{v}\mathbf{l})}{\partial \mathbf{l}} \right| \quad ; \quad C_k : \textit{Lapidus constant} \tag{17}$$

$$\mathbf{l} = \left\{ \begin{array}{c} l_1 \\ l_2 \\ l_3 \end{array} \right\} = \frac{1}{|\mathbf{grad}\ \mathbf{v}^2|} \left\{ \begin{array}{c} \mathbf{grad}_1 \mathbf{v}^2 \\ \mathbf{grad}_2 \mathbf{v}^2 \\ \mathbf{grad}_3 \mathbf{v}^2 \end{array} \right\} \tag{18}$$

b) *2nd and 4th order artificial dissipation*

Instead of the artificial viscosity of Lapidus, an artificial dissipation of a blend of second and fourth order can be chosen [3]. This formulation is better suited if stronger shocks and shocks of different strength appear.

The general expression for the added dissipation flux $\mathbf{f}_d$, in a formulation using mass matrices, is as follows:

$$\mathbf{M}_l \mathbf{f}_d = \alpha^{(2)}(\mathbf{M}_c - \mathbf{M}_l)\mathbf{u} - \alpha^{(4)}(\mathbf{M}_c - \mathbf{M}_l)\mathbf{M}_l^{-1}(\mathbf{M}_c - \mathbf{M}_l)\mathbf{u} \tag{19}$$

This is the additional term to be added to all the equations with the modification of the energy equation where the enthalpy is constant at steady state. Hence, the last component of $\mathbf{u}$, $\rho\epsilon$, is replaced by the expression ρH.

The two coefficients, $\alpha^{(2)}$ and $\alpha^{(4)}$, are obtained from the expressions

$$\alpha^{(2)} = c^{(2)}(|\mathbf{v}| + c)S_e \tag{20}$$

$$\alpha^{(4)} = \max\left[0, c^{(4)} - c^{(2)} S_e\right] \tag{21}$$

and a pressure switched diffusion coefficient for each node which is calculated according to the following term:

$$S_e = \max_{i \epsilon el} \frac{|(\mathbf{M}_c - \mathbf{M}_l)\ p_i|}{\mathbf{M}_c\ p_i} \tag{22}$$

The terms $c^{(2)}$ and $c^{(4)}$ are user defined constants that can be tuned according to the desired amount of diffusion to be added.

3.3 Adaptive Solution

Error estimator

Due to the features of the high speed flow to be solved, strong variations may appear in very localized regions of the domain, whereas the rest of the solution remains smooth. To increase the efficiency of the method a remeshing based on *a posteriori* error estimator is performed [5].

The element mean quadratic error for unidimensional elliptic problems is:

$$E^{el} = C(h^{el})^2 \left|\frac{\partial^2 \phi}{\partial x^2}\right|_{el} \tag{23}$$

h^{el} is the 1D element length, ϕ is the variable chosen, and C is a constant($C = 1/11$ for linear elements).

If the criterion of equidistribution of the error among the elements is adopted, then:

$$(h^{el})^2 \left|\frac{\partial^2 \mathbf{v}}{\partial x^2}\right|_{el} = \text{constant} \tag{24}$$

For 2D and 3D problems, equation (23) may be extended as follows (in a heuristic approach):

$$(\delta_i)^2 |\lambda_i| = \text{constant} \qquad i = 1,3 \tag{25}$$

where λ_i are the eigenvalues of the Hessian matrix $C_{ij} = \frac{\partial^2 \phi}{\partial x_i \partial x_j}$ and δ_i is the corresponding element size.

This has the advantage of providing *directionality* to the error estimator. It is a desirable property, since some discontinuities like shocks present a strong variation in one direction and smooth behavior in the direction normal to the first. It is then useful, to take advantage of this to produce meshes with *stretched* elements in the mentioned regions.

Due to the vector character of **u**, the variable **u** is not well suited for the error estimation. A scalar variable should be chosen. In the test cases presented in this study the Mach number, has been used as the error variable. The new element sizes δ_1 and δ_2 in the principal directions are defined according to:

$$(\delta_1)^2 \, |\lambda_1| = (\delta_2)^2 \, |\lambda_2| = \delta_r \, \lambda\text{max} \tag{26}$$

λ_1 and λ_2 being the eigenvalues of matrix **C**, δ_r is a user specified constant (size) and λ_{max} is the maximum eigenvalue over the whole mesh.

Mesh generation

A finite element mesh of linear triangles of three nodes or linear tetrehedral elements of four nodes is generated using the advancing front technique [1,4] or the Delaunay triangulation. The element sizes are decided in accordance with the sizes specified by a background grid. The initial background grid may be a very simple mesh, and the subsequent grids are those used for the latest or seemingly best computation, where the sizes are decided as specified in the previous paragraph. However in three dimensions, the specifications of the background grid should be limited to about 5000 nodes, to limit the memory needs of the related data structure created by the mesh generation routine. Hence, some routines have been written to reduce redundant information stored at neighboring nodes. One of them is to remove all those points whose patches surrounding that node contain either minimum or maximum spacing information. Another is to geometrically limit the distance between neighboring nodes to a user defined minimum.

4 RESULTS

The particular objective of this study was to analyze the capabilities of Euler calculations on three dimensional unstructured meshes using adaptive remeshing techniques and compare them to other known solutions used by other authors employing different techniques. A well known test case was chosen for comparison, Test case 6.1.7 of the I. Workshop on Hypersonic Flows for Reentry Problems. The two seemingly good computations from ONERA (Guillen, Borrel) [6] and DLR (Radespiel, Herrmann, Longa) [7] presented at the workshop were chosen for comparison. ONERA uses a MUSCL type finite volume method on structured grids, whereas DLR uses a Jameson type Runge-Kutta finite volume method on structured grids, both of which are proven methods in the CFD industry.

Test case

The example corresponds to the hypersonic inviscid flow past a double ellipsoid under the following conditions: $M_\infty = 8.15$, $\alpha = 30°$. All the calculations were obtained on a Silicon Graphics R4000 workstation, with the exception of the largest mesh which was run on a CRAY-YMP.

Mesh and Results

The sequence of the initial meshes and refinement evolution can be seen in Figure 1. A total of four different meshes have been generated. The first coarse mesh contains 38734 nodes and 175872 elements. From this a second mesh was generated containing 22940 nodes. After a third refinement process two meshes were obtained, a coarse mesh of 35808 nodes and a fine mesh of 75006 nodes. Note at this stage the remeshing process has produced many small elements in the vicinity of the two shocks, which appear from this calculation. At this stage stretching of the elements has not been introduced yet, because of our fear of distorting the overall quality of the mesh. This feature could reduce to some amount the necessary amounts of points and elements.

Figure 2 shows Mach number contours for the fine and coarse mesh respectively, where you can see a clear improvement of the resolution of the shock. Observe in Fig. 3 density contours for the fine mesh. For a more detailed analysis the Mach contour lines are shown in Fig. 4 in the symmetry plane only. Note that the canopy shock has been captured much more precisely using the finer grid, also the principal shock shows a crisper pattern than the coarse mesh. Figure 5 compares the pressure coefficient (c_p) of the last two meshes for the plane of symmetry, whereas Fig. 6 compares the c_p of the fine mesh to different authors. The pressure peak is much better resolved using the more refined mesh and compares very well to other codes using structured meshes.

The final figures document the evolution of residuals and forces of the finest grid. The convergence was of 5 orders of magnitude (Fig. 7), although minor oscillations are present at some stage of the calculation. However, this did not affect the overall convergence to steady state, which is documented by the history of the non dimensional forces Fx and Fy, represented in Figure 8.

The cpu-time of the final calculation was about 7,6 hours on a Cray YMP.

5 CONCLUSIONS

An effective solution algorithm of the Euler equations for unstructured three dimensional meshes has been presented. Given the necessary computational resources, this algorithm is capable of solving inviscid flows around complex geometries. Especially when adaptive remeshing techniques become essential for the feasibility of the computations, this explicit scheme remains effective in practical applications.

The next step will be to show similar capabilities of this method for the viscous case, solving the laminar Navier-Stokes equation or even adding turbulence modeling. However, this will again be demanding on the mesh generation capabilities, which must be able to effectively place nodes close to the boundary.

ACKNOWLEDGMENTS

The help of CESCA (Centro de Supercomputación de Cataluña) supplying cpu time on the Cray YMP is gratefully acknowledged.

6 REFERENCES

1.-Peraire, J. *A finite element method for convective dominated flows.* PhD Thesis. University College of Swansea, 1986

2.-Lapidus A. *A Detached Shock Calculation by Second Order Finite Differences.* J. of Computational Physics 2, 154-177, 1967

3.-Jameson A. *Transonic aerofoil calculations using the Euler equations*, in Numerical Methods in Aeronautical Fluid Dynamics, 1982

4.-Zienkiewicz O.C., Taylor *The Finite Element Method* 4th Edition, Volume 2,Mc Graw Hill, 1991

5.-Oñate, E. *Error estimations and adaptive refinement techniques for structural and fluid flow problems* in Mecánica Computational Vol. 11, S. Idelsohn, Asociación Argentina de Mecánica Computational, 1991

6.-Guillen Ph., Borrel M. *Contribution to the workshop on hypersonic flows for reentry problems*, Workshop on hypersonic flows for reentry problems, Antibes, 22-25 January 1990

7.-Radespiel R., Herrmann U., Longo J.M.A. *Problem 6: Flow over a double ellipsoid*, Workshop on hypersonic flows for reentry problems, Antibes, 22-25 January 1990

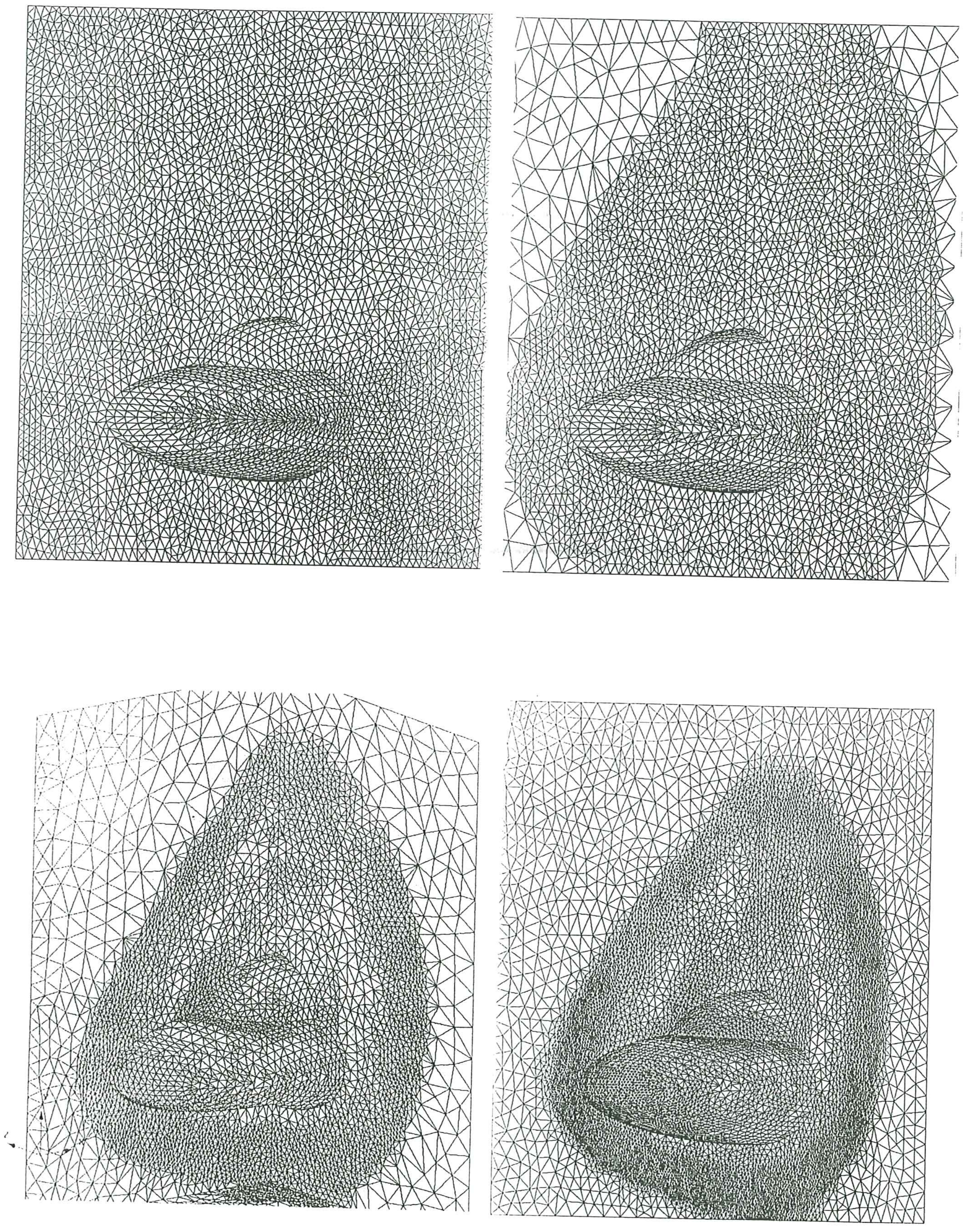
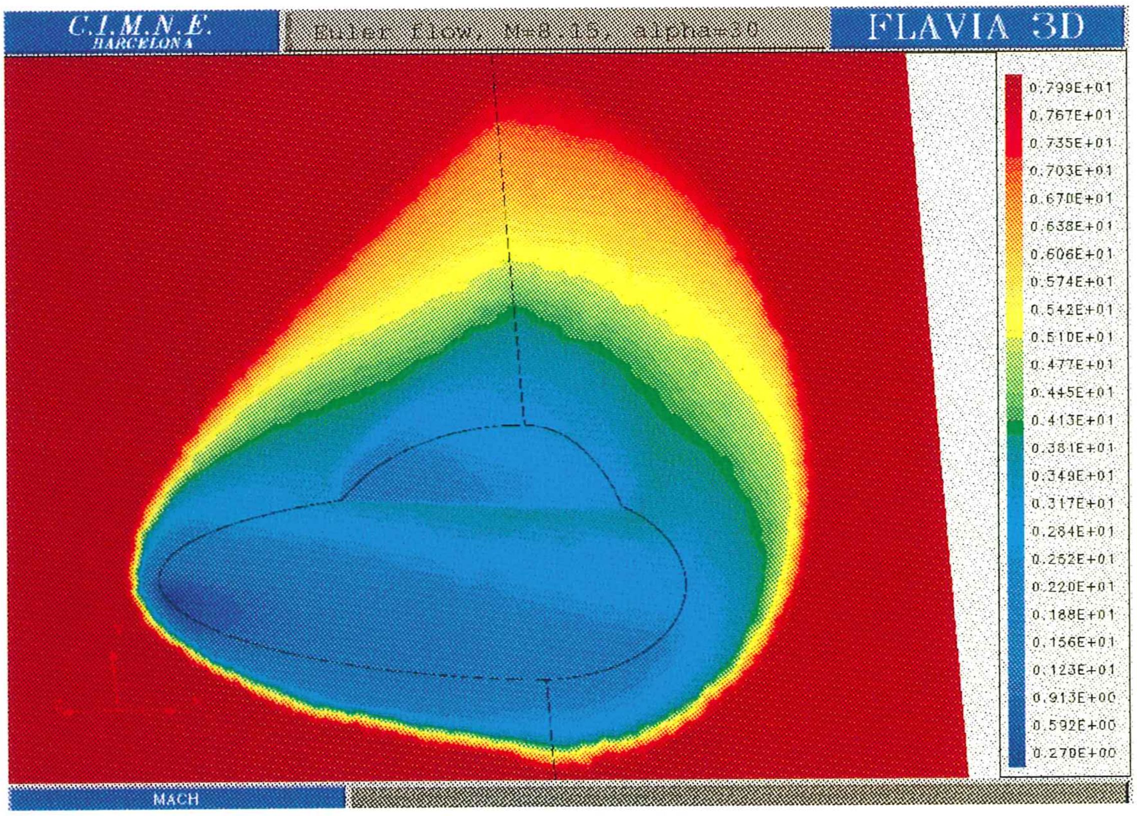
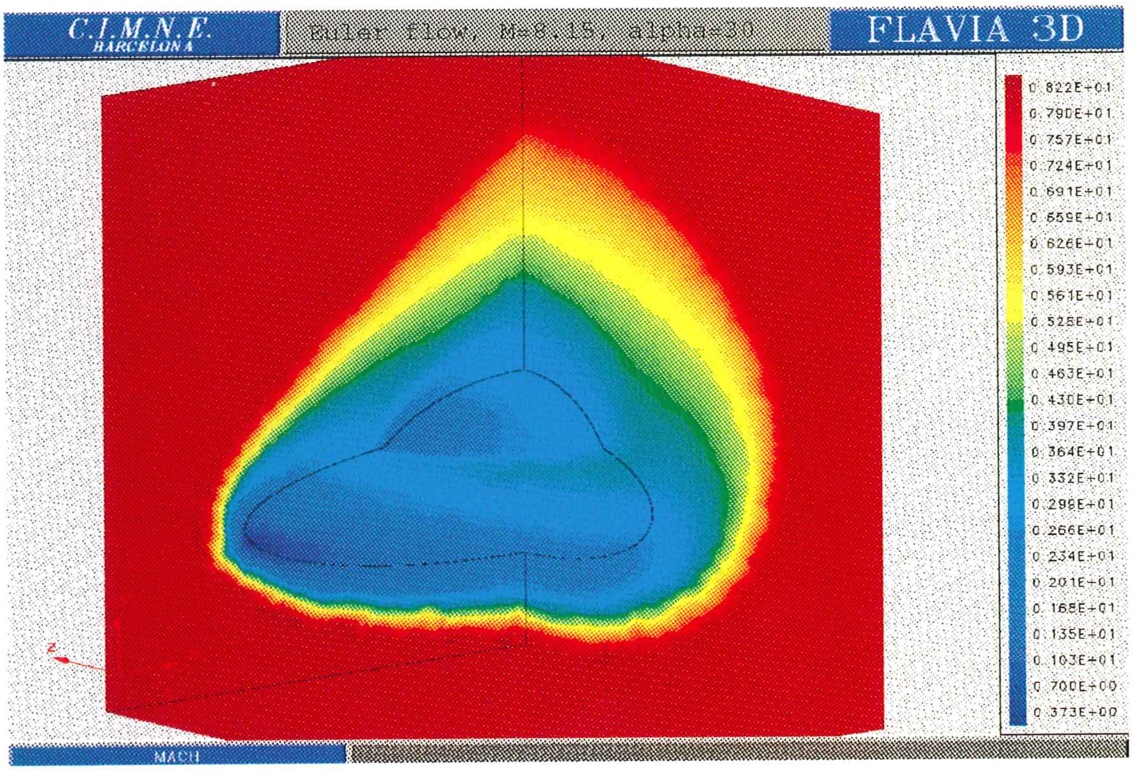


Figure 1: Adaptive remeshing process showing the mesh history.



a) Mach contours for fine mesh



b) Mach contours for coarse mesh

Figure 2

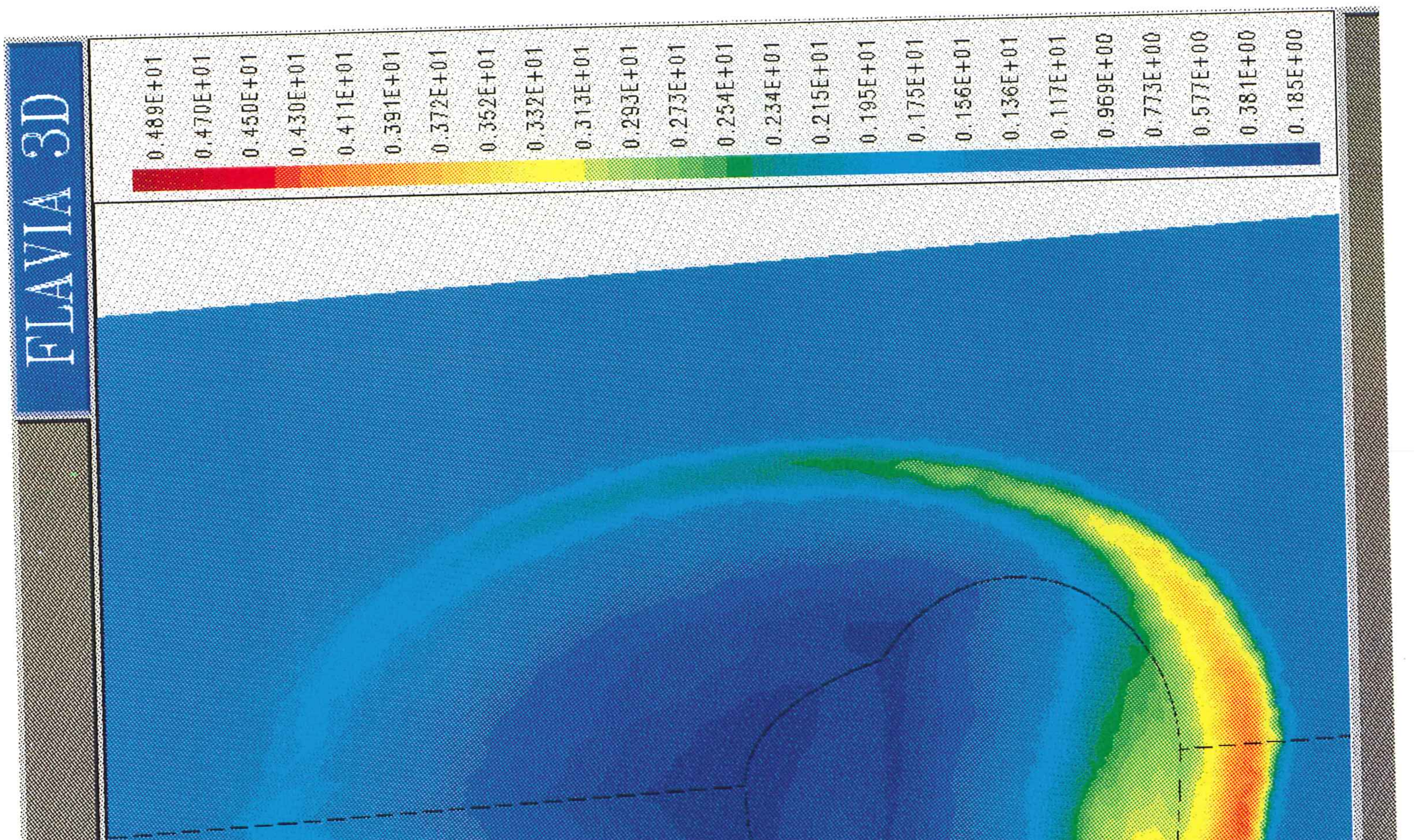
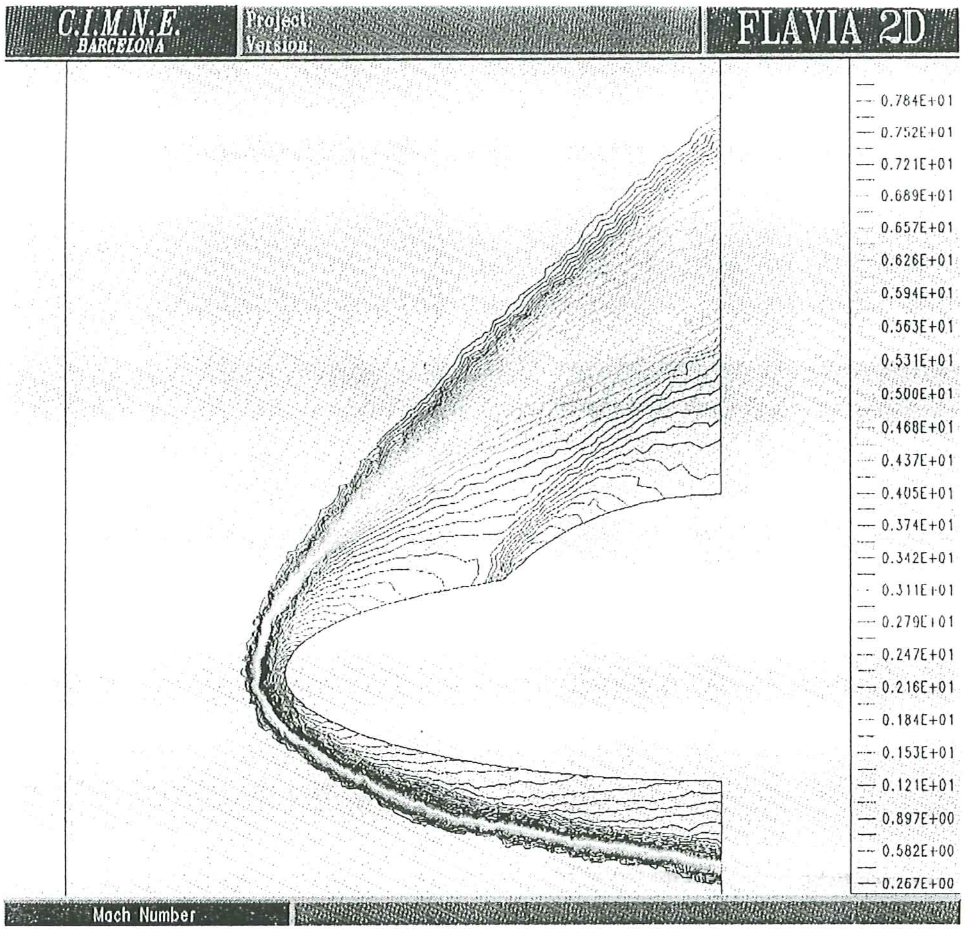


Figure 3: Density contours for the fine mesh



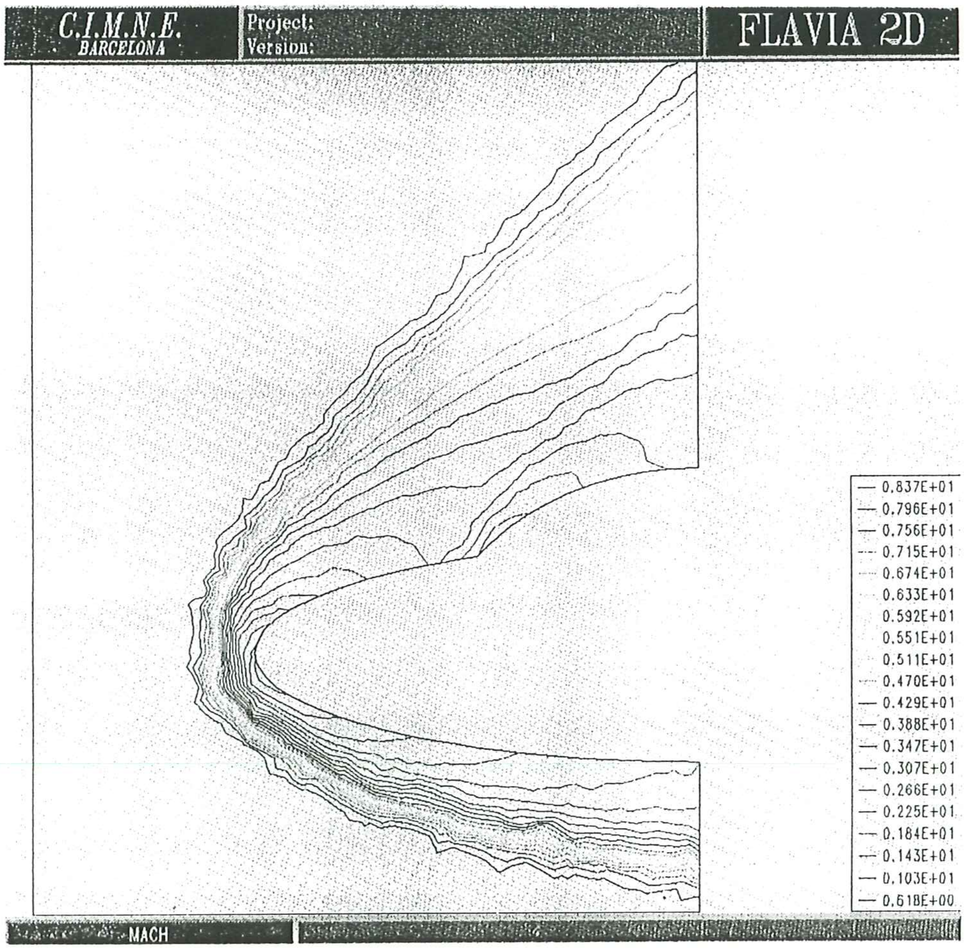


Figure 4: Comparison of Mach number lines for the fine and coarse mesh. Note the higher resolution of the shock for the fine mesh.

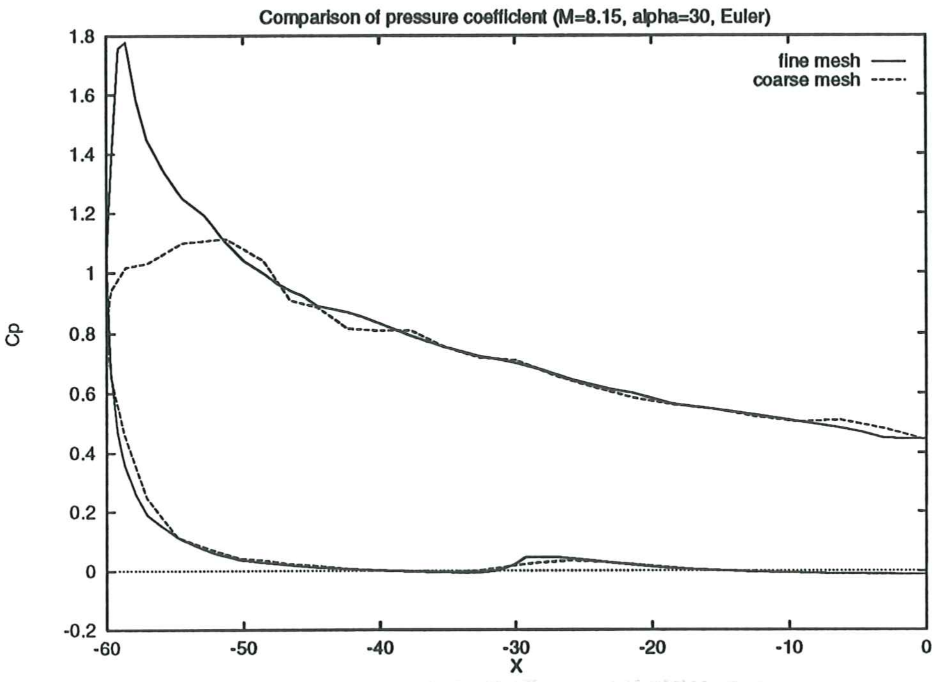


Figure 5: Comparison of c_p of the fine and coarse mesh.

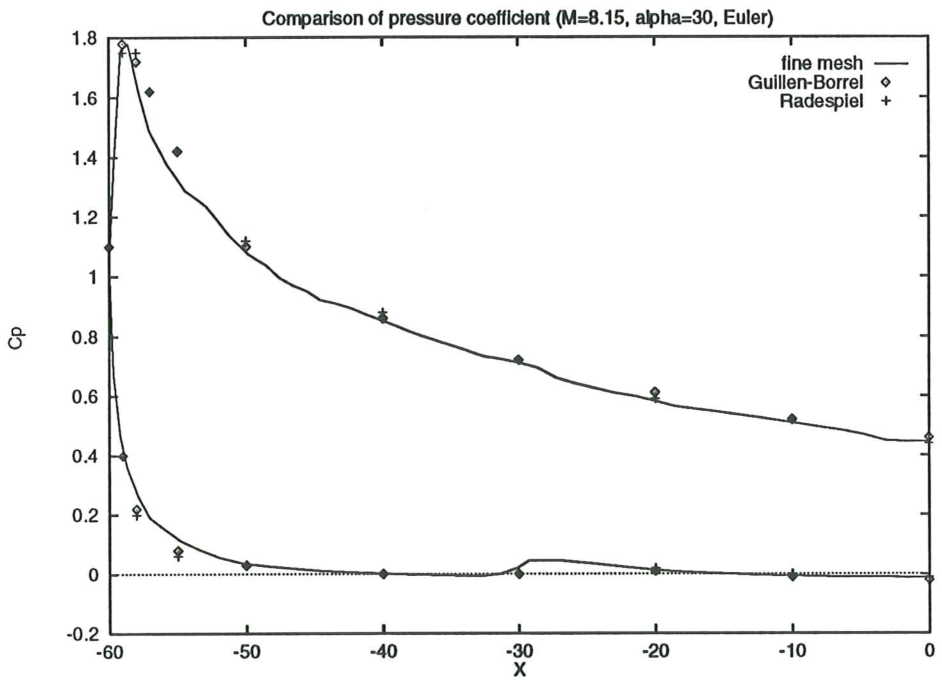


Figure 6: Comparison of c_p of fine mesh with different authors.

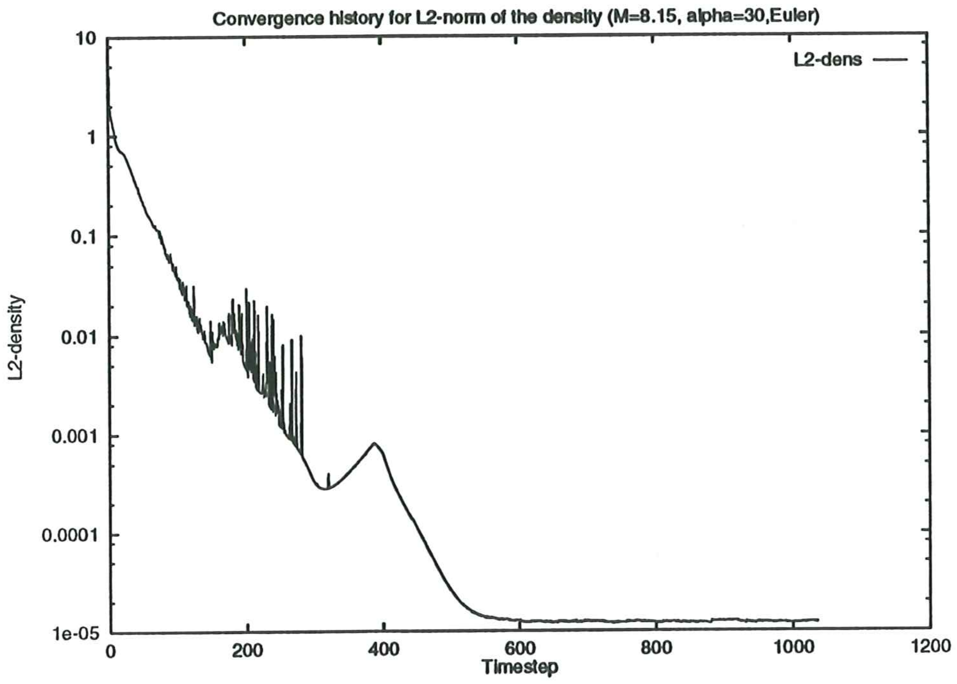


Figure 7: Convergence history of the density residuals.

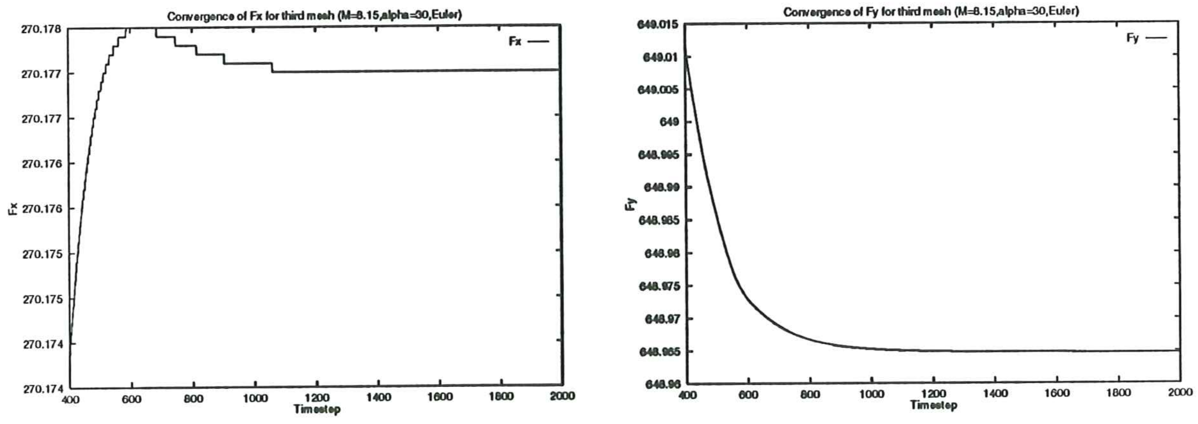


Figure 8: Convergence of the pressure forces.