AIAA JOURNAL
Vol. 31, No. 10, October 1993

# Unsteady Euler Solutions for Arbitrarily Moving Bodies and Boundaries

J. Y. Trépanier,* M. Reggio,† M. Paraschivoiu,‡ and R. Camarero§
*École Polytechnique de Montréal, Montréal, Québec H3C 3A7, Canada*

**A methodology for the computation of unsteady Euler flows in configurations with moving boundaries is presented. The procedure is composed of a moving and adaptive grid management algorithm and a Lagrangian-Eulerian flow solver. The domain discretization is handled via unstructured triangular grids. The flow scheme uses a generalized version of Roe's approximate Riemann solver, which takes into account the mesh movement while satisfying in an intrinsic way the geometric conservation laws. Grid-flow coupling is obtained through error estimation. The methodology is applied to various flow problems showing moving shock waves and moving geometries.**

## I. Introduction

AMONG the many obstacles on the path toward the development of more advanced computational fluid dynamics (CFD) tools, the coupling between the discretization grid and the flow solver occupies an important place. Several layers have been added to the use of adaptive grids in terms of different adaptation strategies such as grid relocation, grid enrichment, and coarsening on structured and/or unstructured grids.[1-5] In the case of steady flowfields, these methods have been very successful. For unsteady flow problems, such as compressible flows with moving boundaries and/or moving discontinuities, it is thought that the use of such methods is inevitable. In fact, when rapid changes in the solution occur or when the computational domain itself evolves in time, it is almost impossible to design a unique and adequate grid for the whole computation process, and modifications in the grid connectivity become essential. This also necessitates the implementation of a numerical scheme that correctly handles grid movement. Along these lines, the approach presented by Ref. 4, based on a *global remeshing* strategy and a finite element flow solver, has shown great potential.

In this work, a separate development is described. The focus is on a *dynamic grid adaptation* procedure that uses triangles as the basic discretization elements. The method, dedicated to the Euler system of equations, couples a finite volume flow solver with a local remeshing approach. The remeshing amalgamates several algorithms basically dedicated to grid motion, grid adaptation, grid quality control, and grid size control. These ingredients constitute fundamental elements for the simulation of flows in complex domains with substantial deformations of the boundaries and with large ratios between the areas of the largest and smallest triangles appearing in the discretization.

The computational domain is defined by a set of curves whose kinematic hierarchy determines the grid pattern. The grid evolution is established on the basis of sequential operators related to the velocity of the moving curves and to nodal displacement. The latter provides an additional smoothing term to improve the quality of the transient grid. In addition to the purely geometric constraints that are imposed by regridding, grid adaptation is performed to resolve strong gradients in the flowfield.

A remeshing methodology based on algorithms for refining, coarsening, and curing has been devised. Grid refinement is achieved through triangle subdivision, whereas grid coarsening is obtained via node removal. A grid-quality parameter is used to trigger various cure actions on the grid. These actions are locally applied in any given region of the triangulation to satisfy the geometric- and flow-based remeshing requirements. A further control on the grid size distribution is carried out by imposing, in relation to a reference grid, size limiters on the triangles.

The solution of the flow-governing equations on arbitrarily static/moving grids has been tackled with a generalized Lagrangian-Eulerian formulation of Roe's Riemann solver. The flow solver[6] respects the so-called geometric conservation laws, which establish relations for the conservation of surfaces and volumes of the control cells.[7] Examples are provided to show the robustness of the method for complex cases.

## II. Governing Equations

The Euler equations for two-dimensional/axisymmetric compressible flows in a general moving reference frame may be written in integral form as

$$\frac{\partial}{\partial t}\int_{V(t)} U\eta \,\mathrm{d}V + \oint_{S(t)} \boldsymbol{n}\cdot \boldsymbol{F}\eta \,\mathrm{d}S = \int_{V(t)} g\eta \,\mathrm{d}V \tag{1}$$

where $U$ is the vector of dependent variables, $\boldsymbol{F}$ is the flux tensor, and $\boldsymbol{n}$ is the outward unit vector normal to the boundary $S(t)$, which encloses the time-dependent volume $V(t)$. The source term $g$ results from the axisymmetric formulation. The variable $\eta$ accounts for the flow geometry. In axisymmetric flows, $\eta$ is taken as the distance $y$ from the axis, whereas in plane-flow cases, it is taken as 1. In the latter case, $g = 0$. The open forms of $U$, $F$, and $g$ are

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix} \quad \boldsymbol{F} = \begin{bmatrix} \rho(u - w) \\ \rho(u - w)u + Ip \\ \rho(u - w)E + up \end{bmatrix} \quad g = \begin{bmatrix} 0 \\ e_y p/y \\ 0 \end{bmatrix} \tag{2}$$

where $\rho$ is the density, $\boldsymbol{u}$ is the fluid velocity, $\boldsymbol{w}$ is the velocity of the boundary of the volume, $E$ is the specific energy, $p$ is the pressure, $\boldsymbol{I}$ is the unit tensor, and $\boldsymbol{e}_y$ is the unit radial vector. The case $\boldsymbol{w} = \boldsymbol{u}$ corresponds to a Lagrangian system,[8-10] and $\boldsymbol{w} = 0$ is a Eulerian one. In the present formula-

Presented as Paper 92-0051 at the AIAA 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-9, 1992; received Jan. 31, 1993; revision received March 30, 1993; accepted for publication March 30, 1993. Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Assistant Professor, Department of Mechanical Engineering, C.P. 6079, Succ. "A." Member AIAA.

†Associate Professor, Department of Mechanical Engineering, C.P. 6079, Succ. "A."

‡Graduate Student, Department of Mechanical Engineering, C.P. 6079, Succ. "A." Student Member AIAA.

§Professor, Department of Mechanical Engineering, C.P. 6079, Succ. "A."

tion, $w$ can be arbitrarily specified, and its computation will be detailed in a following section.

Equations (1) and (2) represent the conservation of mass, momentum, and energy. In addition, an equation of state is considered in the general form:

$$p = p(\rho, e) \tag{3}$$

where $e = E - \frac{1}{2}\boldsymbol{u} \cdot \boldsymbol{u}$ is the specific internal energy. In the case of an ideal gas, Eq. (3) becomes

$$p = (\gamma - 1)\rho e \tag{4}$$

where $\gamma$ is a constant representing the ratio of specific heat capacities of the fluid.

## III. Flow Solver

### A. Static Grids

For a nonmoving grid, the flux between adjacent cells can be written, according to Roe,[11] as

$$F^F(Q) = \tfrac{1}{2}\left[F_R^F(Q) + F_L^F(Q) - \sum_{i=1}^{4}\alpha_i |\Lambda_i^F| e_i\right] \tag{5}$$

where the superscript $F$ has been used to denote a fixed mesh and the subscripts $R$ and $L$ have been used to characterize the right and left states at a given interface. In this expression, the total flux across an interface during a time step $\Delta t$ has been defined as

$$F(Q) = QU + \begin{bmatrix} 0 \\ nP \\ u_n P \end{bmatrix} \tag{6}$$

The "flux eigenvalues" appearing in Eq. (5) are given by

$$\Lambda^F = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \end{bmatrix}^F = \begin{bmatrix} Q + C \\ Q - C \\ Q \\ Q \end{bmatrix} \tag{7}$$

with "flux variables" $Q$, $C$, and $P$ defined as

$$\begin{aligned} Q &= u_n S \Delta t \\ C &= c S \Delta t \\ P &= p S \Delta t \end{aligned} \tag{8}$$

where $u_n$ represents the velocity normal to a given interface $S$, $c$ the speed of sound, $p$ the pressure, and $\Delta t$ the time step. The expressions for the wave strengths $\alpha_i$ and the eigenvector components $e_i$ are well documented in Ref. 11.

Once the flux variables are known, the properties $U$ are advanced to the new time position $n + 1$ using a finite volume approach given by

$$(U^{n+1} - U^n)V^n = -\sum_{k=1}^{N\,\text{sides}} F^F(Q_k) + gV\Delta t \tag{9}$$

### B. Moving Grids

From a physical point of view, the grid motion only effects the convective variables. In particular, the $Q$ term defined in Eqs. (8) changes to

$$Q^M = (u_n - w_n)S\Delta t \tag{10}$$

To calculate the new convective terms and eigenvalues, the velocity $w_n$ of the face of a control volume is required. This velocity has to be specified in terms of the geometry, otherwise the quantities referred to will not be correctly defined.[7] Specifically, it is essential to consider that the total volumetric increment of a deforming cell $V^{n+1} - V^n$ is composed of elementary $\Delta V$ increments along each of its faces. Accordingly, and associated with this facial volume increment $\Delta V$, the relevant facial velocity $w_n$ during a time step $\Delta t$ is defined by

$$w_n = \frac{\Delta V}{S\Delta t} \tag{11}$$

where $S$ represents the length of the side. Using this definition, the convective term $Q^M$ becomes

$$Q^M = u_n S\Delta t - \Delta V = Q^F - \Delta V$$

Applying this definition of $Q^M$, the flux for a moving grid can be expressed as

$$F^M(Q) = \frac{1}{2}\left[F_R^M(Q) + F_L^M(Q) - \sum_{i=1}^{4}\alpha_i |\Lambda_i^M| e_i\right] \tag{12}$$

with modified right and left fluxes and modified "flux eigenvalues" $\Lambda^M$ given by

$$F_R^M = F_R^F - \Delta V U_R$$

$$F_L^M = F_L^F - \Delta V U_L$$

and

$$\Lambda^M = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \end{bmatrix}^M = \begin{bmatrix} Q + C - \Delta V \\ Q - C - \Delta V \\ Q - \Delta V \\ Q - \Delta V \end{bmatrix} \tag{13}$$

In this case, the properties $U$ are advanced to the new time $n + 1$ by

$$U^{n+1}V^{n+1} - U^n V^n = -\sum_{k=1}^{N\,\text{sides}} F(Q_k^M) + gV\Delta t \tag{14}$$

From a practical point of view, and according to the explicit approach adopted, Eq. (14) may be rewritten in a more convenient form as

$$U^{n+1} = \frac{V^n}{V^{n+1}}\left\{U^n + \frac{1}{V^n}\left[-\sum_{k=1}^{N\,\text{sides}} F_k(Q_k^M) + gV^n\Delta t\right]\right\} \tag{15}$$

This formulation shows that the influence of the grid motion is felt through the volume increments $\Delta V$ along the faces of the control volume. Once the original $Q_k$ terms for static grids have been modified to $Q_k - \Delta V_k$, the $U^{n+1}$ variables may be computed similarly as for fixed grids, and a final scaling by the factor $V^n/V^{n+1}$ is applied.

As usual, and depending on the time level in which the flow variables on the right-hand side are assumed and on the method chosen to evaluate the $R$ and $L$ states at a given interface, the nature of the method will be defined. If piecewise constant states are assumed, then the scheme is first-order accurate in both space and time. For nonmoving grids, extensions to second order, performed after the algorithm proposed by Barth and Jespersen,[12] have been presented in a previous work.[13] In the study of moving meshes, an explicit approach has been adopted, and the flow variables are considered to be piecewise constants on each cell leading to a first-order scheme.

### C. Boundary Conditions

The boundary conditions are enforced by using the idea of image cells at the boundaries. In these, the flow properties are set according to the type of boundary, and then the Riemann solver is applied to compute the flux across these boundaries. For a static solid wall, the flow properties in the image cell are taken as those of the adjacent boundary cell, except that the normal component of the velocity is reflected to ensure the impervious condition. For a moving solid wall, a corresponding treatment is applied to the relative velocity ($u_n - w_n$). Specifically, this is negated. At a subsonic outlet boundary, the pressure is imposed whereas the remaining flow properties are extrapolated. For a supersonic outflow, all of the properties in the image cell are extrapolated from the adjacent interior cells. At inlet boundaries, the total pressure and temperature, as well as the flow angle, are imposed for a subsonic flow. The static pressure is taken from the neighboring interior cell. For a supersonic inlet, all of the flow properties are imposed.

## IV. Grid Management

The grid management algorithm includes aspects related to the evolution of the geometry and elements related to the solution behavior. It also comprises a remeshing algorithm and issues concerning remeshing frequency and the maintenance of grid quality.

### A. Geometry-Induced Grid

A computational geometry is described by a set of curves whose movement will effect the grid layout. The evolution of the grid is performed in two steps. The first one directly takes into account the effect of the moving curves via their velocity; the second is an indirect action that provides an additional smoothing based on nodal displacement. According to this, the velocity of the grid nodes can be represented by

$$w = w_g + w_s$$

where $w_g$ is the geometric grid velocity and $w_s$ is the smoothing grid velocity.

Both these terms must respect the boundary conditions defined by the movement of the curves. The resulting grid velocity $w$ is then used by the flow solver to compute the volumetric variations of the computational cells, according to the work of Zhang et al.[7]

#### *1. Computation of the Geometric Term $w_g$*

A moving curve directly influences the grid nodes that lie on it. Two types of curve-node interaction are considered: Dirichlet and Neumann. In Dirichlet curve-node interaction, the velocity of the grid nodes that lie on a moving curve is set equal to the velocity of that curve. There is no relative motion of the nodes with respect to the curve. In a Neumann curve-node interaction, the nodal velocity is set equal to the normal component of the curve velocity at the position of the grid node. This is the minimal constraint that can be imposed on a grid node to remain on the curve. In this case, the grid node slides on the curve with a velocity relative to this and equal to the tangential component of the curve velocity. Some special cases, such as curve-curve interaction, must be taken into account and the assignment of Dirichlet or Neumann types to the curves are left to the user.

The motion of the nodes on the curves establishes a $w_g$ term only for these nodes. In practice this implies a rapid degeneration of the grid near the moving boundaries, and frequent remeshing will be necessary. To minimize these remeshings, the velocities of the nodes on the boundaries can be used in a boundary-value problem for calculating the $w_g$ velocity terms for the internal grid nodes. One way of posing this problem is to consider a Laplacian equation for each velocity component. This yields a very smooth velocity field, but one that is expensive to compute at each time step. Instead, a less rigorous but straightforward methodology, which consists simply of assigning to the internal nodes the mean velocity of their direct neighbors, has been adopted. This procedure is repeated for a few iterations. The final result is a diffusion-like operator that smoothes out the large variations in grid velocity, and this was found effective for the type of computations that were carried out.

#### *2. Computation of the Smoothing Term $w_s$*

The purpose of this term is to produce an additional smoothing of the transient grid evolution by considering nodal displacements. The new position of a grid node is obtained as the average of the position of its neighbors. The velocity of the grid nodes is then calculated by dividing the node translation by a time interval, which is related to the nondimensional time scale of the problem. When this action is applied on nodes located on a Neumann-type boundary curve, the resulting smoothing grid velocity $w_s$ must be tangential to it. Consequently, the normal component of $w_s$ is dropped out, and the nodes are allowed to slide on this curve.

In spite of the use of both the $w_g$ and the $w_s$ terms, tangled grids can arise in some regions. In such cases, procedures described in the following sections will be applied to eliminate the problem.

### B. Flow-Induced Grid

At any time during the computation, the flowfield can ask for grid adaptation in some parts of the domain. It is now well accepted that the optimal grid for the computation of a numerical solution is the one for which the local error is the same for all of the computational cells.[14] Among the various alternatives for obtaining an error estimation, we have followed an a posteriori approach, which starts by projecting the solution into a higher order subspace. In particular, we have applied the technique presented by Barth and Jespersen[12] in which a piecewise constant solution in each triangular element is projected into a piecewise linear solution. The error in the solution is then estimated to be the integration of the difference between the two fields.

Once the error has been estimated, we then need a rule to determine the next grid for the continuation of the solution. The goal is to make the error on the next grid uniform, and this can be achieved if it is known how the error behaves as the grid is adapted. This depends on the flow solver. For the current first-order Roe-based scheme, the error is estimated to be proportional to the grid size, and the characteristics of the next grid are thus obtained by scaling.

Some peculiarities of compressible flow solutions must nevertheless be taken into account when an attempt is made to use this type of error estimation directly. In practice, very high ratios between the areas of the largest and smallest triangles appearing in the discretization can be required. This sometimes results in extremely small triangles in regions of high gradients, which will harm the convergence of the computation. To limit this problem, some triangle-sized limiters in the computational domain have been imposed. The implementation of this idea will be detailed in a subsequent section.

### C. Dynamic Remeshing Technique

A dynamic remeshing technique is needed to satisfy the geometry- and flow-induced grid requirements. For transient flow computations requiring frequent remeshing, a global remeshing can become expensive. Instead of following this avenue, the proposed meshing system consists of a series of local actions that can refine, coarsen, cure, and/or smooth the evolving grid. These operations are performed in a loop whereby each element is tested against a set of objective functions until a given tolerance is reached.

#### *1. Grid Refinement*

Refinement of the grid is obtained through triangle subdivision. A triangle that must be refined is branched into two triangles by cutting it on its longest side. This process is carried out on all of the triangles requiring this option followed by a reconnection of unmatched sides. When a side representing a curved boundary is cut, the new node inserted on that side is relocated on the boundary. The boundaries are a set of curves, kept in the background, that are defined with a larger number of points than used for the initial grid. Thus, the relocation improves the accuracy of the curve representation rather than harms it.

#### *2. Grid Coarsening*

Coarsening of the grid is achieved through node removal followed by a local remeshing. A node is selected for removal if all of its neighboring triangles are to be coarsened. This operation leaves an open polygon that is then remeshed by recursively removing from it the triangle with the highest quality. Some restrictions are imposed to maintain the topology of the triangulation, such as protection from removing nodes located at the intersection of two or more curves or nodes located at sharp corners of the geometry.

### 3. *Grid Curing*

Grid curing comprises operations that are performed on the triangulation to increase grid quality. From a purely geometric point of view, the quality of each triangle is computed as a scalar number given by

$$\text{quality} = \frac{4\sqrt{3}\,\text{area}}{a^2 + b^2 + c^2}$$

where $a$, $b$, and $c$ are the lengths of the sides of the triangle.

This leads to a quality of 1 for an equilateral triangle and 0 for a degenerated triangle. This number serves as a measure to influence various operations of the cure procedure. A more general approach for defining the quality of stretched triangular grids is presented in Ref. 15. Two basic cure procedures have been employed in the present work, and they are 1) the swapping of diagonals and 2) the coarse cure, which deletes bad triangles.

The swapping of diagonals is a well-known technique for obtaining a Delaunay triangulation from an existing triangulation.[16] The algorithm consists in examining each pair of adjacent triangles and selecting from the two possible configurations, for the diagonal side, the one that maximizes the minimum quality in the triangulation. This procedure is repeated until no more swapping occurs.

The coarse-cure procedure examines the triangulation and marks triangles with a quality below 0.4 as bad triangles. Then, for each bad triangle, the node opposed to the longest side is deleted with a procedure similar to the coarsening procedure.

### 4. *Solution Transfer Between Grids*

One important characteristic of the dynamic grid adaptation procedure is the relatively easy transfer of the solution between successive grids. At each adaptation cycle, a local relation is kept between the newly generated grid and the previous one. Then the solution transfer procedure uses these pointers to transfer and/or interpolate the solution on the new grid. With this technique, there is no need for any searches. Nevertheless, as realized by Formaggia et al.,[17] a critical point to deal with when applying grid adaptivity is the conservativeness of the flow variables.

For the present first-order scheme, in which piecewise constant variables have been assumed, the transfer of information is fully conservative during the refinement operation, because this simply implies triangle subdivision. The coarsening operation interpolates from the previous mesh the flow variables at the centroid of the new triangles. Although this action is not strictly conservative, coarsening is called only in low-gradient regions, thus the error is small and can be tolerated.

### 5. *Remeshing Algorithm*

The goal of the remeshing algorithm is to produce a triangulation meeting a field of sizes dictated by the error estimator. The proposed remeshing process is described in Table 1. The first step determines which triangles require refinement or coarsening, and a corresponding code is attributed to each triangle. In practice, these actions are discrete operations on the grid, and some care must be taken in setting the triangle code to avoid possible oscillations in the remeshing process, i.e., to ensure a quasismooth grid convergence. To do so, the average performance of the two basic operators, refinement and coarsening, is first evaluated. The refinement operator produces triangles of an area half that of their parent. The coarsening operator, which, on average, will operate on nodes surrounded by six triangles, produces new triangles with areas of about 1.5 times the average parent area. From this basic data, the codes on triangles have been set according to the following inequalities:

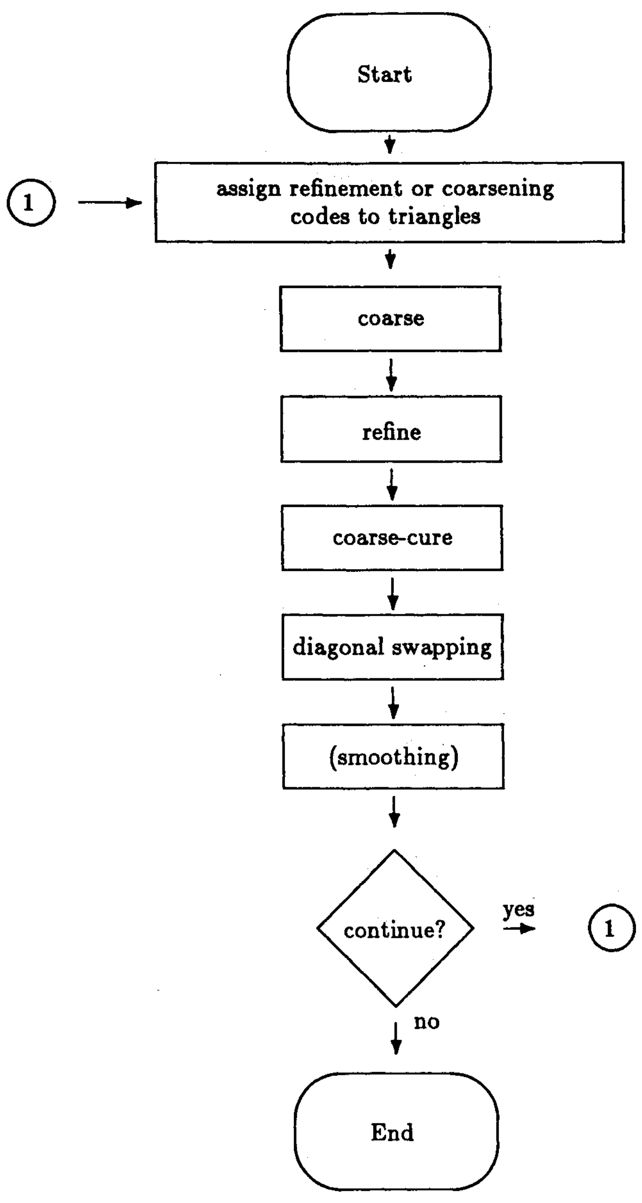
```
IF   actual area > 3/2 required area,
  THEN  set a refinement code
IF   actual area < 3/4 required area
  THEN  set a coarsening code
```

In choosing these limits, we expect that a refined triangle will not require coarsening later and that a coarsened triangle will not require refinement later. The final mesh will meet the requirement within these limits, i.e., with no triangles larger than 3/2 and only a few triangles smaller than 3/4 of the required area.

## D. Global Remeshing Algorithm

A global grid control strategy is needed to link the algorithms described earlier. This strategy controls two parameters: the imposition of size limiters for the triangulation and the frequency of remeshing.

**Table 1 Global remeshing algorithm**



### 1. *Size Limiters*

As already mentioned, a strategy is needed to limit the minimum and maximum sizes of the triangles computed by the error estimates. To achieve this control, an initial grid or "reference grid" is first devised. Then local limits on the smallest and largest triangle areas are specified in terms of a ratio of the reference grid. For computations in geometries that need very high ratios of initial grid sizes, this approach is more flexible than the specification of absolute minimum and maximum sizes. Moreover, this technique has the ability to deal with both geometric and flow-grid requirements during a transient solution process. The geometric grid requirement means that, during a transient solution process in a geometry with moving boundaries, the grid must adapt correctly to these boundary changes. This is accomplished by setting the minimum and maximum ratio limits equal to 1.0, i.e., by forcing the grid to remain approximately the reference grid. Conversely, the flow-grid requirements refer to the adaptation of the grid to the solution itself. This is done by setting the minimum and maximum ratio limits to, say, 0.1 and 5.0, leading to minimum and maximum triangle areas of 10 and 500% of the reference grid, respectively. The minimum ratio limit $\Rightarrow 0$ and maximum ratio limit $\Rightarrow \infty$ represent the removal of any constraint on triangle size. Finally, the characteristics of the reference grid are transported by the current grid along its movement.

### 2. *Frequency of Remeshing*

The frequency of remeshing is determined by two conditions: the geometric requirement and the flow requirement. A geometric remeshing is performed each time that the $\Delta t_{\text{Grid}}$, which is defined as the minimum time interval needed to reduce one of the triangle areas by one-half, is reached. In this grid adaptation step, very few triangles are normally involved. A flow remeshing is carried out after a certain number of iterations on the flow solution. The frequency of this action is determined by a user-controlled variable, as are the minimum and maximum area ratio limits.

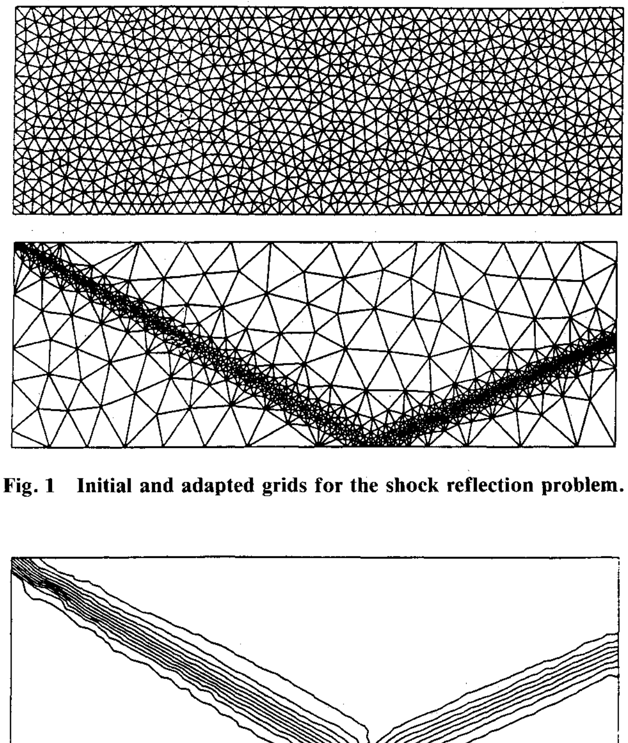


**Fig. 1 Initial and adapted grids for the shock reflection problem.**

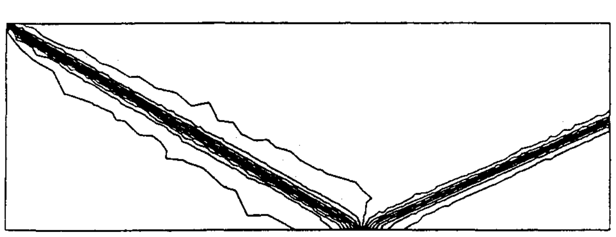


**Fig. 2 Mach number isolines computed with the initial and adapted grids.**

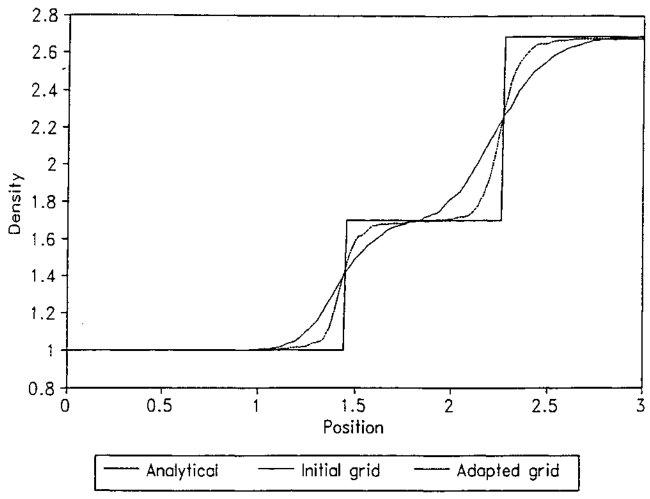


**Fig. 3 Comparison of the density variations on the two grids.**

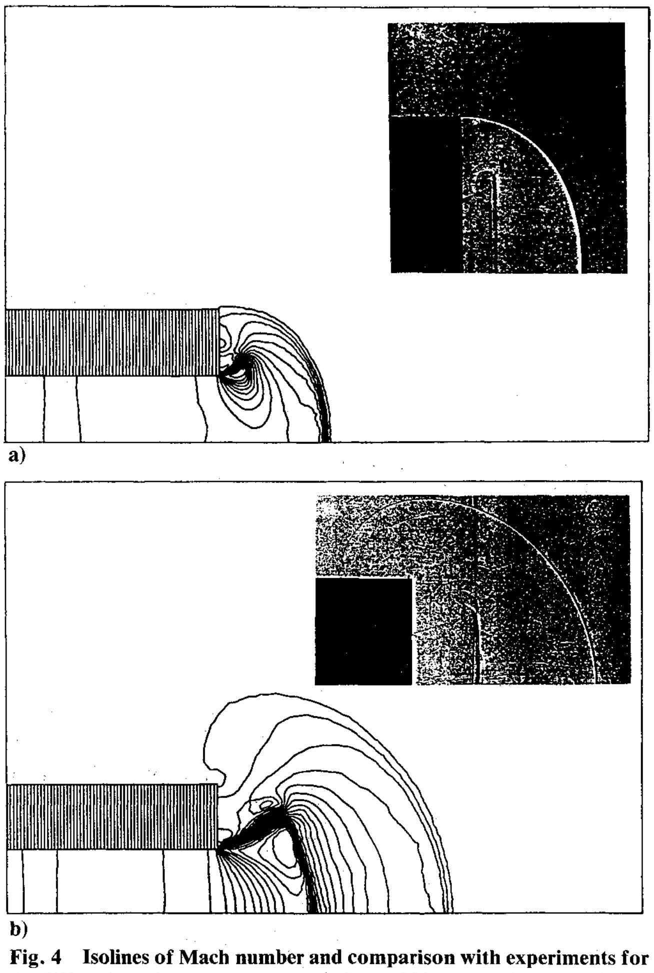


**Fig. 4 Isolines of Mach number and comparison with experiments for the diffraction of the shock wave: a) time = 200 μs and b) time = 500 μs.**

## V. Results

The methodology has been applied to the computation of various unsteady compressible flows. Basic validations have

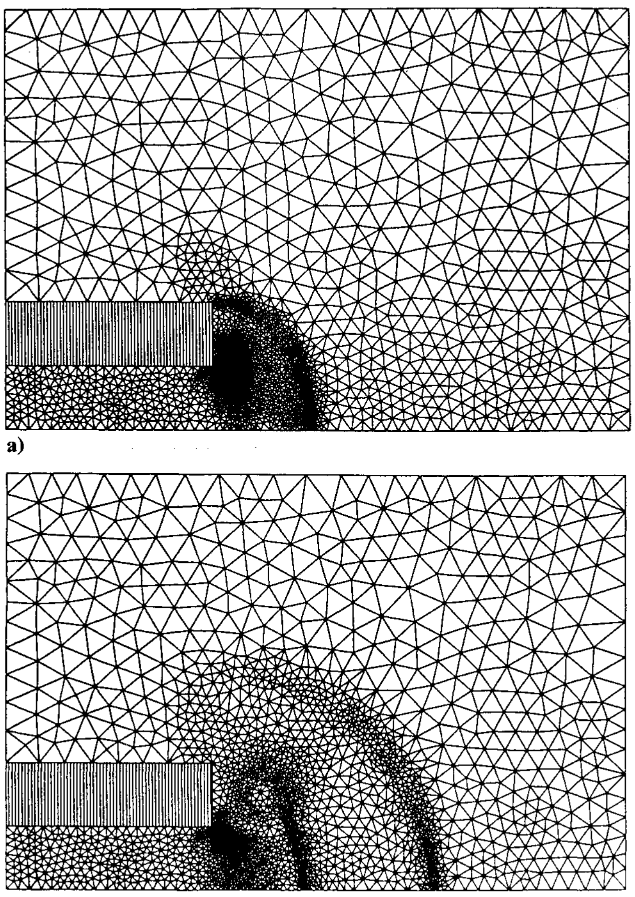


Fig. 5 Adapted grid corresponding to the solutions of Fig. 4: a) time = 200 $\mu$s and b) time = 500 $\mu$s.

already been reported in Ref. 6, and applications to oscillating airfoils can be found in Refs. 18 and 19. In this paper, we first demonstrate the correctness of the solver in a well-known shock reflection problem. Then three computations of unsteady flows in complex configurations will be presented to illustrate the versatility of the method.

## A. Oblique Shock Reflection

The first example concerns a shock reflection on a wall. Figure 1 illustrates the initial grid and the grid obtained after adaptation. To show the effectiveness of the mesh adaptivity, the number of elements in the adapted grid has been kept as close as possible to that of the initial grid. Figure 2 illustrates the computed Mach number isolines.

A further comparison of the solutions is depicted in Fig. 3, where the density distribution at $y = 0.2$ is presented for both cases. It is appreciated that the performance of the adaptation is important, as expected.

## B. Blast Waves from a Shock Tube

This example concerns the simulation of the diffraction of a shock wave exiting a cylindrical shock tube. This configuration has been investigated previously by Cooke and Fansler[20] using Harten's total variation diminishing scheme on a structured grid made up of almost 16,000 grid nodes. The purpose of this computation is to show how dynamic grid adaptation management can respond to a rapidly moving flow discontinuity. A shock wave is positioned initially inside the tube with the conditions across it being given by the Rankine-Hugoniot equations, together with a prescribed pressure jump. Figure 4 presents the results in the form of Mach number isolines, and these are compared to the shadowgraphs experimentally obtained.[20] The principal characteristics of this flow are captured by the solver and also by the grid management, as illustrated in Fig. 5. Figure 6 shows a comparison of the computations and the experimental data of Ref. 20 of the overpressure at two locations in the domain. The current computation compares well with the numerical results in Ref. 20, with many fewer nodes. However, we note that the slip line is not felt by the error estimate and that, consequently, the grid is not refined in this region. This has been attributed to the choice of density as the key variable, which is not strongly discontinuous across the slip line. Some work is still needed in this area to find a better error estimate.

## C. Exploding Pressure Vessel

The second problem investigated is the simulation of the explosion of a pressure vessel. The details of this geometry can be found in Ref. 4. The configuration shows two concentric cylinders where the inner one is filled with a high pressure gas (40 atm) and closed with a lid.

The simulation begins when the lid starts accelerating upward under the influence of the pressure difference. The lid velocity is prescribed as in Ref. 4. Figure 7 shows a sequence of grids and Mach number isolines in parallel. The flow structure is very complex, comprising shocks, slip lines, and regions of recirculating flow. A direct comparison with the experimental data of Ref. 4 has not yet been possible due to a lack of information concerning the exact experimental setup. However, the computation is a good illustration of the robustness of the proposed algorithms.

## D. Operation of a Circuit Breaker

This electrical engineering application presents a challenging CFD problem where difficulties are found in geometrical

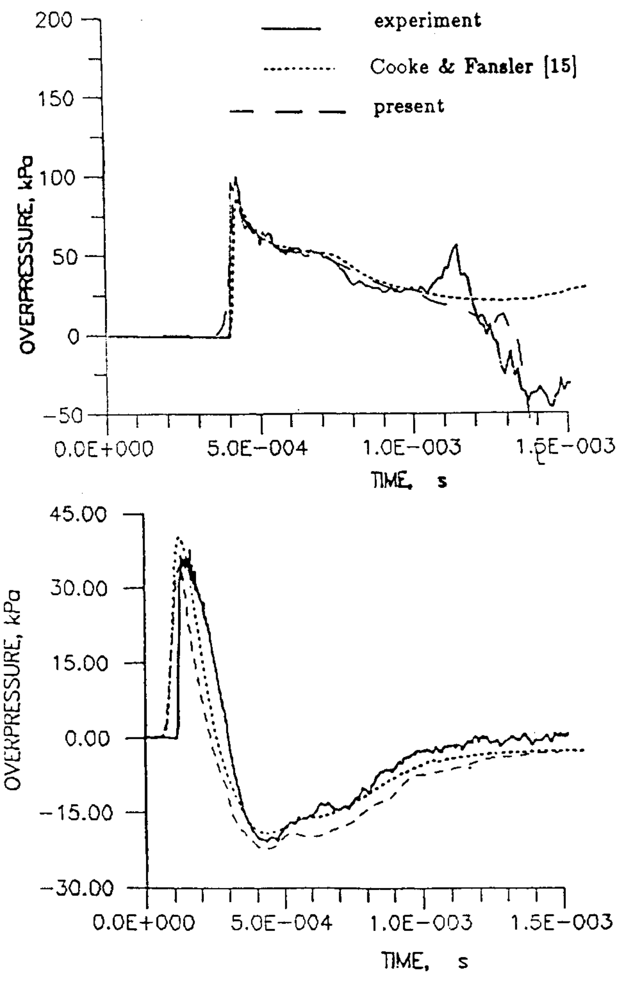


Fig. 6 Comparison of overpressures for the blast wave computation.

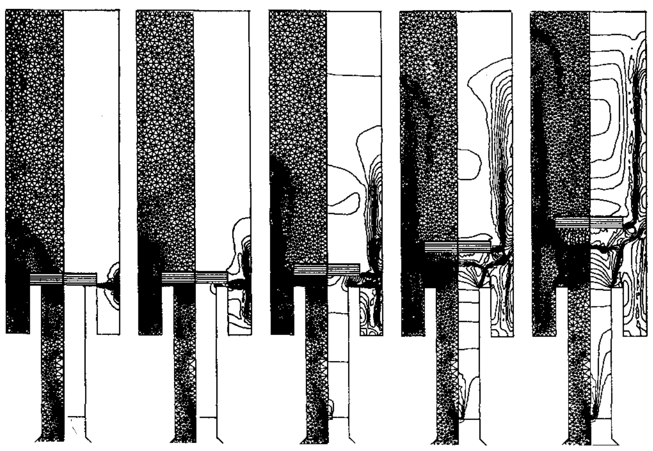


Fig. 7 Isolines of Mach number and grid for the exploding pressure vessel.

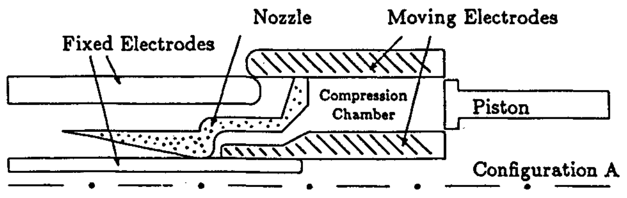


Fig. 8 Geometry for the circuit-breaker problem.

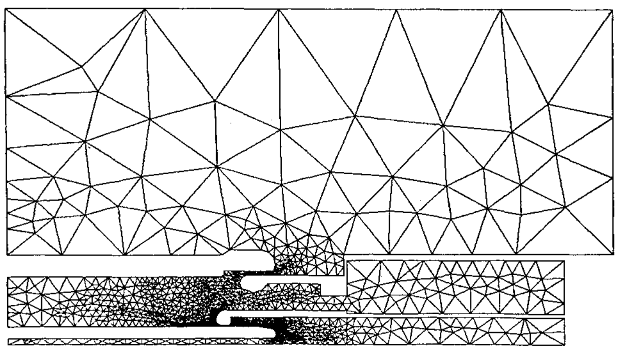


Fig. 9 Initial grid for the circuit-breaker problem.

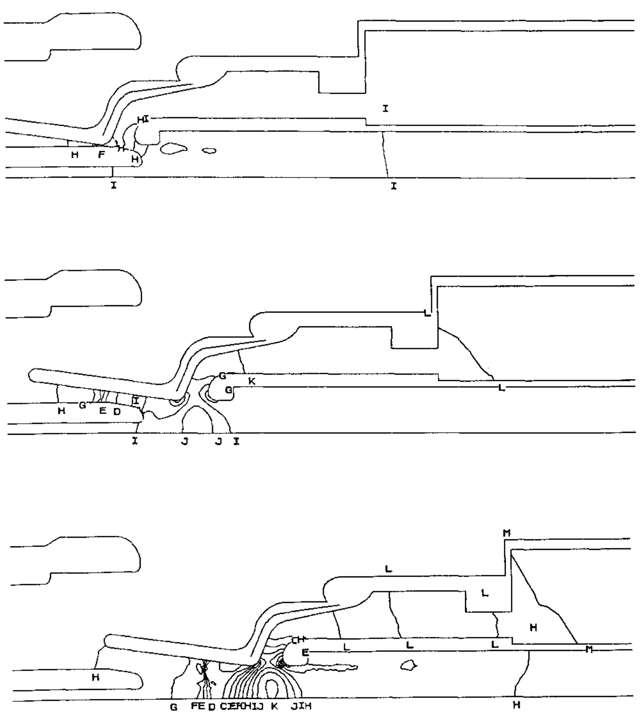


Fig. 10 Isolines of density for the circuit-breaker problem.

aspects, the real-gas effects, and the presence of strong source terms. Figure 8 shows a sketch of the inside of the axisymmetric breaking chamber. During operation, the moving electrodes and the piston move together to the left under the action of a spring mechanism. For the simulation, the opening velocity was prescribed following a polynomial form.

In most operations of the breaker, the electrical current passing through the fluid in the form of an arc is small and can be neglected. The major problem in these cases is to determine the effect of the flow on the distribution of density in the interelectrode region, which can be related to the dielectric withstand of the apparatus.

Figure 9 illustrates the initial grid in the chamber, and Fig. 10 presents a sequence of solutions at different times in the form of isolines of density. Under the action of the piston, the pressure in the compression chamber reaches almost three times the initial pressure, and this forces a sonic flow at the throat. Beyond the throat, the flow expands in the diffuser and is forced to adjust to the conditions near the electrodes through a strong shock wave. The computations have shown that the density in the electrode region is lower than the initial density in the chamber.

## VI. Conclusion

A methodology for the numerical simulation of unsteady compressible flows within complex geometries with moving boundaries has been presented, implemented, and tested. The procedure is based on the coupling of a finite volume Euler flow solver with an algorithm for moving and adaptive triangular grid management. The flow simulation uses an extended version of Roe's scheme written in a Lagrangian-Eulerian form.

The grid-flow coupling has been obtained by means of an error estimate based on a piecewise linear reconstruction of the solution. The resulting grid-flow coupling methodology enables the efficient generation of well-suited grids for compressible flow problems that respond to both the geometric requirements and the flow phenomena. The capability of the flow solver grid management technology was first shown in a standard shock reflection test case. Then the simulation of three realistic problems involving unsteady flows within complex geometries was carried out. The results for this new class of problems indicate good potential for industrial applications.

## References

[1]Löhner, R., "An Adaptive Finite Element Solver for Transient Problems with Moving Bodies," *Computers and Structures*, Vol. 30, No. 1/2, 1988, pp. 303-317.

[2]Mavriplis, D. J., "Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation," *Journal of Computational Physics*, Vol. 90, No. 2, 1990, pp. 271-291.

[3]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, No. 11, 1987, pp. 1211-1228.

[4]Probert, E. J., Hassan, O., Morgan, K., and Peraire, J., "An Adaptive Finite-Element Methods for Transient Compressible Flows with Moving Boundaries," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 751-765.

[5]Thompson, J. F., "A Survey of Dynamically-Adapted Grids in the Numerical Solution of Partial Differential Equations," AIAA Paper 84-1606, June 1984.

[6]Trépanier, J. Y., Reggio, M., Zhang, H., and Camarero, R., "A Finite Volume Method for Solving the Euler Equations on Arbitrary Lagrangian-Eulerian Grids," *Computers and Fluids*, Vol. 20, No. 4, 1991, pp. 399-409.

[7]Zhang, H., Reggio, M., Trépanier, J. Y., and Camarero, R., "Discrete Form of the GCL for Moving Meshes and Its Implementa-

tion in CFD Schemes," *Computers and Fluids,* Vol. 22, No. 1, 1993, pp. 9–23.

[8]Harlow, F., and Amdsen, A., "A Numerical Fluid Dynamics Calculation Method for All Flow Speeds," *Journal of Computational Physics,* Vol. 8, 1971, pp. 197–213.

[9]Donea, J., "Arbitrary Lagrangian-Eulerian Grids Arbitrary Lagrangian-Eulerian Finite Element Methods," *Computational Methods for Transient Analysis,* edited by T. Belytschko and T. Hughes, Elsevier Science Publishers B.V., The Netherlands, 1983, pp. 473–516.

[10]Trease, H. E., "Two-Dimensional Free Lagrangian Hydrodynamics," Ph.D. Thesis, Univ. of Illinois, Urbana-Champaign, IL, 1981.

[11]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics,* Vol. 43, No. 2, 1981, pp. 357–372.

[12]Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 89-0366, Jan. 1989.

[13]Trépanier, J. Y., Reggio, M., and Ait Ali Yahia, D., "An Implicit Flux-Difference Splitting Method for Solving the Euler Equations on Adaptive Triangular Grids," *International Journal of Numerical Methods for Heat and Fluid Flow,* Vol. 3, No. 1, 1993, pp. 63–77.

[14]Babuska, I., and Rheinboldt, W. C., "Error Estimates for Adaptive Finite Element Computations," *SIAM Journal of Numerical Analysis,* Vol. 15, No. 4, 1978, pp. 736–754.

[15]Zhang, H., Trèpanier, J. Y., Reggio, M., and Camarero, R., "A Navier-Stokes Solver for Stretched Triangular Grids," AIAA 30th Aerospace Sciences Meeting, AIAA Paper 92-0183, Reno, NV, Jan. 1992.

[16]Lawson, C. L., "Software for c1 Interpolation," *Mathematical Software III,* Academic Press, New York, 1977, pp. 166–194.

[17]Formaggia, L., Peraire, J., and Morgan, K., "Solution of a Store Separation Using a Finite Element Method," *Applied Mathematical Modelling,* Vol. 12, 1988, pp. 175–181.

[18]Trèpanier, J. Y., Zhang, H., Reggio, M., and Paraschivoiu, M., "Periodic Euler and Navier-Stokes Solutions About Oscillating Profiles," *Canadian Aeronautics and Space Journal,* Vol. 38, No. 2, 1992, pp. 71–75.

[19]Paraschivoiu, M., "Unsteady Euler Solution for Oscillating Airfoil and Oscillating Flap," AIAA Paper 92-0131, Jan. 1992.

[20]Cook, C. H., and Fansler, K. S., "Comparison with Experiments for TVD Calculations of Blast Waves from a Shock Tube," *International Journal for Numerical Methods in Fluids,* Vol. 9, No. 1, 1989, pp. 9–22.