A93-45082

AIAA-93-3391-CP

A Simulation Technique for 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry

Sami A Bayyuk[*] and **Kenneth G Powell**[†] and **Bram van Leer**[‡]
Department of Aerospace Engineering,
The University of Michigan,
Ann Arbor, MI 48109-2140

Abstract

A new technique for the time-accurate numerical simulation of Euler flows around moving and deforming bodies in 2-D is presented. The grid used is of the stationary, Cartesian, unstructured, Quadtree-based type. The use of a stationary grid is made possible by allowing bodies to move across grid-lines. Use is made of the property that the cells in the immediate vicinity of a moving or stationary body can always be merged together to form combined cells that are topologically invariant over a motion step. Without degrading the global order of accuracy, this merging eliminates the CFL number constraint imposed by cut cells of small area, and eliminates the need for special handling of cells that, due to the motion of bodies across grid-lines, are uncovered or swept-over. The flow solver used is based on a finite-volume conservative formulation, and Roe's approximate-Riemann-solver is used for the computation of interface fluxes. 1-exact spatial reconstruction is used in conjunction with a predictor-corrector explicit time-marching scheme to obtain spatial and temporal second-order accuracies. Gradient limiting is used to enforce monotonicity-preservation near discontinuities. In order to resolve flow features economically and controllably the grid is adaptable to both the geometry and and the flow solution. The computational resources required by this technique are shown to be typical of unstructured-grid implementations. Two computations are presented to demonstrate the capabilities and validity of the technique.

1 Introduction and Motivation

1.1 A Perspective

In [1] we described a geometry-adaptive, solution-adaptive technique for solving the steady Euler equations for flows around fixed bodies of (with minor restrictions) arbitrary geometry. In [2] we described an extension of that method to the unsteady Euler equations, still for fixed bodies. This paper describes an extension of the method to the solution of the unsteady Euler equations with arbitrarily moving and deforming bodies of arbitrary geometry.

[*]Doctoral Candidate, Aerospace Engineering
[†]Associate Professor, Aerospace Engineering
[‡]Professor, Aerospace Engineering

1.2 Background, And The Case For Grid-Line Crossing

Algorithm development for the solution of flows around moving and deforming bodies has received much attention lately. Practical applications are numerous and important. They include the improved simulation of fluid-structure interactions (as in aeroelasticity), the simulation of separation and approach problems, and the full (as opposed to the component-by-component) simulation of problems involving continuous periodic relative motion, as in rotor-stator and rotorcraft flows.

Most of the methods developed so far, as in [3], for example, resolve the geometric-complexity issue satisfactorily, typically with unstructured-grid or block-structured-grid techniques. Except in a few cases, however, these methods have a notable geometric limitation: the scale of motion (or deformation) of bodies must be small in comparison with the scales of body-to-body and body-to-outer-grid-boundary separation. Otherwise, relative motion, particularly of the shear type, can distort the grid severely enough to impair flow-solver performance. The origin of this limitation can be traced to the use of grids having a fixed topology between boundaries.

Several strategies have been suggested or attempted to circumvent or counter this limitation. Relying on grid redistribution to relieve unwanted features fails in all but the least severe grid deformations. In addition, the use of a moving grid suffers from the inherent shortcomings that attend continuous global dependence of grid point positions on body positions (primarily updating of the geometric data for the entire grid after every motion step.) Use of local insertion and deletion in structured or block-structured grids, as introduced in [4] can effectively remove unwanted grid distortions but has the disadvantage of possibly leading to grid modification in contra-indication to the flow-features. This would defeat the essential purpose of adaptivity. Therefore, the technique does not hold promise for extension to the general moving body problem. The use of overlaying grids, as in [5], for example, suffers from the usual penalties that attend duplication of data and the need for continuous conservation-violating data interpolation across grid boundaries. The local-remeshing strategy employed in, for example, [6], [7], and [8] is perhaps the wisest in that it directly tackles the cause of the problem. Moreover, since the remeshing is performed only around bodies, economy is inbuilt into the technique. Perhaps the major disadvantages of this class of technique is its use of a moving grid and the difficulty of correcting its violation of conservation.

Allowing bodies to move across grid-lines also attacks the problem at its root by decoupling grid motion from body motion. Consequently, the density distribution of the grid will, as it should, only have locallized dependence on the surface geometry and positions of bodies. Grid modification for geometric changes still requires updating the grid data after every motion step, but only for cells in the immediate vicinity of bodies. In addition, since body and grid motions are decoupled, the use of a stationary grid, and the economic advantages therefrom, become possible. These incentives justify further exploration of this approach.

2 Algorithm Description

2.1 Design Strategy

Finite-volume-based, explicit flow-solvers, such as Roe's approximate-Riemann-solver, have little dependence on the geometric form of the computational cells. In addition, they employ only local operators. While the flow-solver-associated tasks of reconstruction and gradient-limiting significantly increase the required amount of connectivity-information retrieval, it still seems advisable to allow the choice of the data-structural and geometric form of the grid to be dictated primarily by geometric and implementation-efficiency (i.e. computational effort and storage) requirements.

Regarding choices for the data-structural form of the grid, the geometric and implementation-efficiency requirements have traditionally been viewed as favoring opposite strategies. On the one hand, it is becoming generally accepted that complex geometries are better negotiated by unstructured grids, or, in the least, by grids that are not fully-structured. On the other, unstructured grids have lower implementational efficiency. While the geometric difficulties in grid generation are proving largely insurmountable with fully-structured grid techniques, the implementation-efficiency issue for unstructured grids is an area in which much progress can be made by good design and new ideas.

These considerations determine the contemporary framework within which subsequent choices should be made. Within that framework, our final objective gives the arbitrary-geometry capability precedence over the implementation- efficiency criterion and so favors the use of an unstructured grid. The requirements for a geometry-adaptive, flow-adaptive grid imply a continuously changing grid. Thus, one performance criterion for the grid data-structure must be its modification speed. The other performance criterion is the memory requirements, and hence the connectivity-information requirements. The design task can therefore be reduced to the problem of designing a data-structure which jointly satisfies these two criteria as far as possible.

2.2 Grid Data-Structure and Grid Generation

2.2.1 The Basic Quadtree Data-Structure

The data-structural form of the grid is Quadtree-based, as briefly described in, for example, [1]. The Quadtree is a tree type data-structure with a branching ratio of four: except for the leaf (or ultimate) nodes every node is linked to four other nodes, which are viewable as hierchically subordinate. To facilitate tree traversal and connectivity-information retrieval, it is common to also link each node to its hierchical superior. In that case each interior node will have five links, while the root node will have only four, and leaf nodes only one link. This tree structure can be turned into a grid data-structure if, in addition to the connectivity data bourne by the nodes, additional data is attached to them.

2.2.2 Representation Of Data In A Quadtree And Efficiency Issues

In the Quadtree representation of a 2-D grid, the physical space, typically a polygonal region is represented by the root node. The physical space is divided into four pairwise disjoint areas and each is represented by one of the four nodes immediately subordinate to the root node. Each of the quadrants of the original space can now be regarded as a starting point for further subdivision, with the node representing that quadrant being considered as the local root of the descendant tree. Subdivision can now proceed recursively to obtain any refinement level at any desired location in the original space. If the tree is required to represent a grid with varying levels of refinement, then the minimal tree will have branches that terminate at different levels.

For each node of the tree sufficient geometric and connectivity data must be stored to fully define the spatial region represented by the node. The efficiency advantages of the Quadtree structure are best realized, however, if the original space is a square and all subdivisions results only in squares. In that case, for example, no coordinate or geometric data whatsoever needs to be stored except for the root node. The geometric properties of any quadrant would be deducible from the root's and the position in the tree of the quadrant's node. The position in the tree of any node can be retrieved by analyzing the path of an upward-traversal of the tree all the way to the root. This compactness is a property that not even structured body-fitted grids can enjoy, since these (except for simple geometries) must have areas or other geometric data stored. Morever, for such a choice of geometry (i.e. squares) it is possible to choose any level of intermediate (internal) node at which to store geometric data. Such a choice of level corresponds to a choice, regarding the computation of geometric data, between the extremes of:

1. maximum effort and minimum storage, and,
2. minimum storage and maximum effort.

The nodes of the tree can be used to represent any required attribute. Particularly, the gasdynamic properties on which the flow-solver operates can be stored as for the geometric attributes. However, in this case, the attribute need only be assigned at the leafs since only the leafs of the tree are used to represent computational cells.

The resulting structure is one that contains all the required grid data at the leafs with the remaining tree structure used only for connectivity-information retrieval. The exponential nature of this data-structure makes for efficient storage. Since the interior nodes are only used for connectivity-information retrieval, they may be regarded as a memory overhead. The interior-to-leaf node ratio, ν, may be deduced by observing that whenever a node is subdivided the number of interior nodes is increased by 1 while the number of leaf nodes is increased by 3. After, say, m subdivision operations starting with the root node, the total number of interior nodes will be $(0 + m)$ while the number of leaf nodes will be $(1 + 3m)$. Therefore, $\forall m \in \mathrm{N}$,

$$\nu = \frac{m}{1+3m} < \frac{1}{3}.$$

This result holds regardless of the location of the subdivision operations, i.e., it holds for any grid of any density distribution.

The Quadtree is also a very efficient structure in terms of connectivity-information retrieval. In particular, the expected value for the number of tree ascents and descents required for a face-neighbor retrieval is upper-bounded by 2. The corresponding figure for vertex-neighbor retrieval is upper-bounded by 8/3.

In both cases the retrieval procedure consists of ascending the tree from the required node to a node that satisfies certain conditions and then descending from the latter down a path (which is some mirror image of the ascent path) to the sought neighbor. These ascent and descent operations are equivalent to memory access operations, but some computation is necessary to determine the paths. It turns out that the access times are theoretically of the same order as the array-element access times (which require integer arithmetic computation and memory access operations too) for structured grids. Unlike in the structured-grid case, however, these operations do not vectorize.

2.2.3 The Grid Generation Algorithm

The stepwise basic grid generation follows this procedure for fixed bodies:

1. Subdivide the physical domain to a prespecified level (the back-ground refinement level).
2. Examine the resulting leafs for intersection with any bodies in the grid and recursively subdivide each cell that is neither interior nor exterior until one of three conditions is met:
   (a) The resulting leaf cell is exterior to all bodies.
   (b) The resulting leaf cell is interior to a body.
   (c) The resulting leaf cell has reached a prespecified maximum depth (the intersection refinement level) in the tree.
3. Recursively subdivide or coarsen exterior cells according to the current flow-variable gradients. This step may be considered part of the solution-adaptivity.

The actual procedure differs slightly from that described above. For example, an important deviation from the Quadtree data-structure occurs when interior cells have their representing nodes deleted from the tree (to save memory). Another difference from the above procedure is that any subdivision is not carried out if it violates a neighborhood refinement-level difference of one. Instead, the would-be violated node is recursively divided first. This ensures that the grid maintains three desirable properties: one topological, one geometric, and one implementational. These, respectively, are:

1. The number of neighborhood patterns remains small and hence neighbor accessing remains fast and relatively logic-free.
2. The grid area variation remains boundedly smooth throughout (i.e. the area ratio between any two neighbors is no less than $frac14$ and no more than 4.)
3. The global variations in the grid density maintain some local dependence and hence the motion of a body across a grid does not result in refinements and coarsenings that telescope across several levels of the tree.

Examples of the resulting geometric forms of the grid can be seen in Figures (3)-(10).

For moving bodies, the procedure is expanded as follows:

1. Generate the grid for the current body positions.
2. Coarsen the grid (by one level) only around the bodies.
3. Move the points defining the bodies in accordance with the required trajectories over the coarsened grid and hence obtain the new body shapes.
4. Rebuild the grid down to the required intersection refinement level around the bodies, recomputing all intersection points and updating the local geometric variables.

This procedure makes its strengths and weaknesses self-evident. It is simple and general, making no distinction between body deformation and gross motion. It allows the grid to remain fixed and hence makes for more economical flow-solver and grid-related computations. Only the grid in the immediate vicinity of a moving body is directly affected, leaving the propagation of the local changes to be transmitted to the remainder of the grid by the one-level-refinement-difference rule; no separate checking for global suitability and no global-modification operation are required. This contrasts with, for example, advancing-front grid generation. The major detraction of the strategy is the necessity to recompute all intersection points and geometric variables around bodies after each motion step.

2.3 Representation And Definition Of Bodies

The most general and challenging input for geometric definition of bodies for CFD purposes is the one based on a set of points on the body surfaces. This is the case in both 2- and 3-D. If bodies are given by a set of surface points, parametric cubic spline interpolation is used to reconstruct the surface initially, and, after each motion step. Thus deformation and gross motion are made indistinguishable. If bodies are given by sets of equations, then these equations are used instead of the spline equations to compute any required intersections.

2.4 Motion Control Of Bodies

Pointwise displacement and velocity control options are available for independent control of the trajectory of each control point of each body. This gives total control over the gross motion and the deformation of bodies. The motion of bodies may be coupled to the pressure distribution on the body surfaces for fluid-structure interaction problems provided the density distribution or other equivalent data is given for the bodies.

2.5 Flow-Solver

2.5.1 General

The flow solver is based on a finite-volume discretization of the unsteady Euler System. In conservation form, the Euler System for a time-varying control volume is given by

$$\frac{d(\tilde{\mathbf{U}}\mathbf{V})}{dt} + \oint \tilde{\mathbf{F}}_{\mathbf{n}}.\mathbf{ds} = \mathbf{0}, \tag{1}$$

where $\mathbf{V}$ is the volume of the space, $\tilde{\mathbf{F}}_{\mathbf{n}}$ is the flux tensor, appropriately modified to account for control-surface velocities, and $\tilde{\mathbf{U}}$ the state-vector.

Since the volumes of the boundary cells will in general vary with body motion, the discretization must be applied to the true conserved variables (not their volumetric densities.) The discretization of the above equation for deforming volumes may be expressed in the form

$$\frac{\Delta(\tilde{\mathbf{U}}\mathbf{V})}{\Delta t} = -\sum_{faces} \bar{\tilde{\mathbf{F}}}_{\mathbf{n}}.\mathbf{ds}, \tag{2}$$

where $\bar{\tilde{\mathbf{F}}}_{\mathbf{n}}$ is an appropriate time-average of the flux tensor over the deforming area.

The interface fluxes are computed in accordance with Roe's approximate-Riemann-solver. In order to obtain second-order spatial accuracy away from discontinuities, 1-exact reconstruction is carried out within cells. The reconstruction procedure employs least-squares minimization to obtain the 1-exact cell-centroidal estimates of the reconstructed variable gradients. The variables chosen for reconstruction in the computations presented herein are the primitive variables, namely ρ, u, v, p.

Second-order time-accuracy is obtained by the use of Heun's method, which may be interpreted as a predictor-corrector method. The formulation must be applied to the conserved variables and so takes the form:

$$\begin{aligned}
(\tilde{\mathbf{U}}\mathbf{V})^{(0)} &= (\tilde{\mathbf{U}}\mathbf{V})^{n}, \\
(\tilde{\mathbf{U}}\mathbf{V})^{(1)} &= (\tilde{\mathbf{U}}\mathbf{V})^{(0)} + \frac{\Delta t}{2} Res((\tilde{\mathbf{U}}\mathbf{V})^{(0)}), \\
(\tilde{\mathbf{U}}\mathbf{V})^{(2)} &= (\tilde{\mathbf{U}}\mathbf{V})^{(0)} + \Delta t Res((\tilde{\mathbf{U}}\mathbf{V})^{(1)}), \\
(\tilde{\mathbf{U}}\mathbf{V})^{n+1} &= (\tilde{\mathbf{U}}\mathbf{V})^{(2)}.
\end{aligned}$$

2.5.2 Initial Conditions And Wall And Other Boundary Conditions

Spatially second-order boundary conditions are applied at the moving solid boundaries. This is done by defining the flux vector at a solid boundary ($\frac{\tilde{\mathbf{F}}_n.\mathbf{ds}}{|\Delta \mathbf{s}|}$) by direct use of:

$$\tilde{\mathbf{F}}_n.\mathbf{ds} = \begin{pmatrix} 0 \\ p\Delta y \\ p\Delta x \\ pv_x\Delta y - pv_y\Delta x \end{pmatrix}$$

where p is the extrapolated value of the pressure at the boundary's mid-point as determined from the reconstruction. It should be noted that such an expression for boundary flux computation is quite desirable since it only involves one gasdynamic unknown; the remaining unknowns are all geometric variables.

The conditions at the outer boundaries of the grid may be chosen arbitrarily, but special precautions must be taken to avoid the unstable interaction, if applicable, of the reconstruction procedure with von Neumann boundary conditions. The initial conditions may be arbitrarily specified.

2.6 Cell Merging

2.6.1 The Small-Cell Problem

Each body cuts out its boundary from the Cartesian cells it intersects. Invariably, some cut cells will have smaller areas than others. Particularly, the area ratio for the smallest cut cell to the smallest uncut cell in the grid may be arbitrarily close to zero. Meanwhile, the CFL stability constraint

$$CFL = \frac{\Delta t(|\vec{q}| + c)}{\sqrt{A}} < 1.0.$$

applies to all cells. It turns out that variations in gasdynamic states are far outweighed by variations in areas and consequently the smaller-area cells are always the ones that determine the global time-step, and hence the overall pace of the computation for a Quadtree-based Cartesian grid. This could imply a code several orders of magnitude slower than a code based on the smallest uncut-cell area.

It must be noted that cut cells do not contribute to the global accuracy of the solution. On the contrary, even if permitted to pace the computation, these cells are invariably numerically decoupled from the rest of the flow field. This is a finite-arithmetic consequence of their residual (a small quantity) being computed from the fluxes (near-cancelling relatively much larger quantities.)

The small-cell problem is one of the major inconveniences of using grids with cut cells.

2.6.2 The Vanishing- And The Newborn-Cell Problems, And More Generally The Moving-Body Problem

Examine cell A in Figure (1). As indicated therein, the motion increment of the body will sweep across the external part of the cell. The usual finite-volume procedure cannot be applied to this cell over the given motion step: there is no guarantee that the final conserved-variable vector in the cell will vanish. If this does not happen, the implication is that mass, momentum, or energy has entered through the body's surface, or, even worse that the cell has negative mass or energy. One possibility is to compute the time at which the state-vector vanishes, and then consider this to be the instant at which the cell is completely swept out, regardless of whether this coincides with the geometric crossing. This approach is not promising. For one it has the defect of requiring special checking for cells experiencing this phenomenon and then special handling. Moreover, all the surrounding cells will require individual computations over subintervals of the global time-step in order to preserve conservation in space.

Now examine cell B. The motion increment exposes part of this previously interior cell, however, no well-founded flux computation across the interface is possible since no state-vector is available on which to base the computations. In addition, it would be necessary to compute the exposure time of this cell from the motion of the body and to break up the gasdynamic computations in the surrounding cells into each subinterval over which the topology of each cell is invariant. Other than the fact that this involves expensive geometric computations, such an approach again introduces exceptional handling for exceptional cells with all the attendant penalties.

As Figure (1) shows, throughout the vicinity of the body there will in general be cells with large variations in area, as well as cells with small initial or final areas or both. These are all undesirable phenomena.

A solution to all the problems outlined above can be achieved simultaneously by cell-merging.

2.6.3 Basic Concept And Some Implementation Considerations

The basic idea of merging is the combination of adjoining cells so that any boundaries between them are ignored for discretization purposes, thus forming new, combined, larger cells.

Under certain restrictions on the curvature of the surface of a body (relative to the dimensions of the intersected cells) and on the body's minimum width, the following can be proved:

All the cells intersected by the surface can be combined with other cells lying within an appropriate envelope around the surface to form new cells such that the following criteria are simultaneously satisfied:

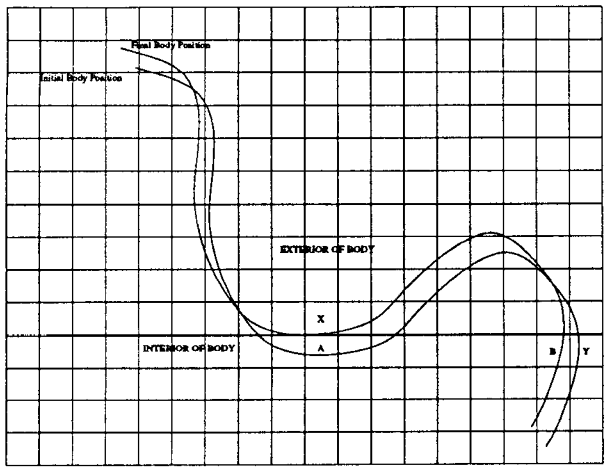

Figure 1: The Vanishing- and Newborn-Cell Problems

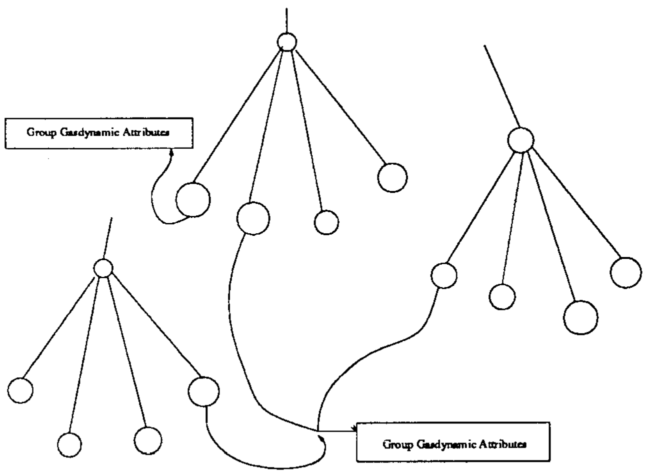

Figure 2: The Data-structural Implementation of Merging

1. Both the initial and final areas of combined cells are no smaller than the smallest uncut cell in the grid or some (less severe) fraction thereof. Note that this criterion constrains, indirectly, the proportional area change within a combined cell.

2. No corner vertex of a combined cell is crossed by the surface during the motion step, if there is motion of the surface.

3. No combined cell is formed from the merging of more than eight cells.

The first criterion alone eliminates the small cell problem for both moving and stationary boundaries. For a moving boundary, the second criterion implies that the combined cells have an invariant intersection topology over the given motion step. Thus, the criterion ensures that combined cells have their intersection status preserved: cut cells remain cut, interior cells remain interior, and exterior cells remain exterior. Consequently, both the vanishing and newborn-cell problems are eliminated. These two criteria are essential for the viability of merging. The second criterion has an important consequence for the temporal order of accuracy of the overall technique. It can be shown that length-averaging of the interface fluxes for cells with varying area does not, of itself, lead to loss of second-order time-accuracy in equation (2) if this criterion is satisfied. Finally, the last criterion, while not affecting the order of accuracy, places a limit on the loss of resolution due to merging.

As an example, referring to Figure 1, the cells marked A and B will be merged with the cells marked X and Y respectively, eliminating a vanishing-cell and newborn-cell problem respectively.

From the data-structural point of view, merging cells is very elegant and simple. Before a merging operation, each participating cell will have its individual gasdynamic and geometric attributes. A single new set of gasdynamic and geometric attributes is created for the merged group and has its variables computed from the individual attributes. The new set is then assigned to each participating cell. The individual attributes are thereafter discarded (to release memory for future use.) In computing the group attribute values, adherence to conservation is the only restriction. For example, the group state-vector must be obtained by volume-averaging of the participants' state-vectors.

Figure (2) shows how the merging data-structure is attached to the Quadtree backbone. The particular figure shown demonstrates how merged cells (which must be neighbors) could lie far apart in terms of the tree structure.

In the actual implementation, all cells in the grid are merged into merging groups (which will have only one cell for exterior cells.) This arrangement gives a uniform representation that allows merging to remain invisible to most flow-solver operators.

2.6.4 The Merging Algorithm

Given that merging will solve all the problems listed above by *combining cells together till all undesirable features are eliminated*, the remaining problem can now be reduced to identifying the cells that are to be merged together as efficiently as possible. The algorithm developed for this purpose follows the following procedure:

1. For each body in the grid, a suitable initiation point for merging is found. Starting at that point, the surface is traversed in one direction visiting each cut cell along the way. For each such cell, the intersection pattern of the grid lines in the neighborhood is analyzed. Depending on the outcome, cells are merged into groups of up to eight cells. The merged cells at the end of this step satisfy the second and third criteria above.

2. For each merged group the initial and final areas are computed. If the first merging criterion fails, additional unmerged cells chosen along a direction as perpendicular as possible to the local body surface are merged with the group. At the end of this step all merged groups satisfy all three criteria.

3. All remaining unmerged cells in the grid are self-merged. This is done for the reason explained above. At the end of this procedure all cells in the grid satisfy all three merging criteria and the data-structure is ready for the gasdynamic operators.

2.6.5 Unmerging

After the merging is complete and the gasdynamic states have been updated and the bodies have moved through their latest motion increments, the cells around the bodies are unmerged. This is the reverse operation of merging. Thus cells must be reidentified as separate entities and their individual state-vectors must be recomputed from the merged group state-vectors and the final areas of the individual cells. The loss of resolution resulting from this operation when applied to groups of more than one cell is minimized by use of the gradients already computed for the merged cells. Clearly, interior cells must be assigned zero state vectors, in upholdance of conservation.

Unmerging must be carried out because the criterion of topological invariance for merged cells has a life-cycle of only one motion step in general; the intersection pattern around any moving body must be reanalyzed to determine the new merging groups.

2.6.6 An Accuracy Consideration

Having established that coarsening, whether due to direct coarsening or to merging, does not change the order of accuracy, the effect of merging on accuracy can be looked at from another point of view. Since the ultimate purpose of grid adaptation is economic optimization, it is more meaningful to express the loss of resolution in terms of a cost-benefit analysis. In that context, it can be stated that if the resolution level at the body surfaces corresponds to what is available at the penultimate nodes, the extra refinement level (which typically accounts for a small proportion of the total number of cells in the grid) can be viewed as an extra overhead of sacrificial cells required to make the algorithm work. In actuality, the resolution is higher than that afforded by the penultimate nodes since the gradient information at the leafs is averaged and stored at the penultimate nodes and subsequently used in the prolongation to the rebuilt leafs.

2.6.7 An Undesirable Constraint

The current merging algorithm indirectly introduces a restriction on the motion-step of the body: the merging algorithm requires the cells to be highly refined around sharp corners and at the same time requires the motion step magnitude to be less than the smallest uncut-cell length. These two requirements together imply that the permissible motion step for a solid body must decrease with the smallest radius of curvature on the body. This constraint can be eliminated.

2.7 Solution Adaptation

As explained previously, this procedure amounts to refining or coarsening cells in accordance with the resolution sought at the cell's location.

Much freedom exists in choosing a criterion to quantify the desired level of resolution. Unfortunately, however, only the vaguest guidelines, e.g., the concept of equidistribution, have been developed to guide such choice. In [1] a particular choice was described, and in [2] another. Currently, the code is using a criterion based on a weighted average of the velocity divergence and the magnitude of the velocity-curl. The weights may be varied to impart different relative sensitivities to shear- and shock- type features.

3 Computational Requirements

The storage requirements of the technique with second-order accuracy are approximately thirty words per cell for moving bodies and twenty words per cell for stationary bodies. These requirements can be reduced to about half the quoted values but only at the expense of processing time. There are additional storage requirements, most notably for postprocessing. The maximum storage requirement at any time during a run does not exceed thirty-five words (i.e. 140 bytes) per cell for moving bodies.

The total processing time with second-order accuracy is currently about three milliseconds per cell per iteration on a DECstation 5000/200. However, the code is not configured for maximum speed. For the examples shown herein more than 70% of the computational effort was consumed by the flow-solver, while the geometric computations' share was about 15% during body motion. The remainder is consumed mainly in the data-structural modifications and grid adaptation. Since geometric computations are only necessary for intersected cells, the relative effort required for geometry-related operations decreases with the ratio of intersected to unintersected cells.

4 Demonstrative Results

The attached figures show a sequence of successive solutions to two unsteady problems. In the first, an ellipse instantaneously attains a constant horizontal speed, contracts as it passes between two stationary ellipses, then expands to its original shape as it leaves the ellipses. The example demonstrates that the technique can be used to simulate an approach problem, a separation problem, and a problem in which the dominant relative motion across a small separation is of the shear type.

In the second problem, a cylinder imparts, in rapid succession, by purely radial deformation, a compressive and an expansive wave to the surrounding gas. This example was chosen to give the reader a graphical sense of the extent to which the strong directional allignment of a Cartesian grid distorts an axisymmetric problem.

5 Concluding Remarks

We have presented an approach for the simulation of Euler flows around arbitrarily moving and deforming bodies of arbitrary geometry. The approach differs from other approaches developed so far primarily in that a stationary grid is used and body motion across grid lines is allowed. We have shown how grid-line crossing in conjunction with a Quadtree-based Cartesian grid can eliminate the grid distortion problem simply and neatly and can intrinsically ensure the global smoothness of the grid. We have shown how the flow-solver-related problems of small cells, vanishing cells and swept-over cells arising from cell cutting and grid-line crossing can be solved by exploiting the flexibility of the data-structure and without sacrificing the order of accuracy, as verified in [9]. Finally, we have demonstrated that the technique has computational requirements consistent with an unstructured-grid technique.

From the long-term perspective, the technique has demonstrated that Cartesian grids can be used to solve the moving and deforming body problem in 2-D. Moreover, unlike Delaunay triangulation, the technique presented here generalizes to 3-D without the introduction of additional grid-generation difficulties.

Regarding extension to unsteady viscous flows, the technique promises some potential but many important issues await more study, particularly with regard to the efficient handling of length scale disparities of flow-features in a Quadtree-based data-structure.

Acknowledgements

This work was funded in part by a grant from the McDonnell Aircraft Company, monitored by Dr. August Verhoff.

References

[1] D. D. Zeeuw and K. G. Powell, "An adaptively-refined cartesian mesh solver for the euler equations," AIAA Paper 90-0000, 1990.

[2] Y. L. Chiang, B. van Leer, and K. G. Powell, "Simulation of unsteady inviscid flow on an adaptively refined cartesian grid," AIAA Paper 92-0443, 1992.

[3] J. T. Batina, "Unsteady euler algorithm with unstructured dynamic mesh for complex aircraft aeroelastic analysis," AIAA Paper 89-1189., 1989.

[4] A. Evans, M. Marchant, J. Szmelter, and N. Weatherill, "Adaptivity for compressible flow computations using point embedding on 2-d structured mutli-block meshes," *International Journal for Nummerical Methods in Engineering*, vol. 32, pp. 895–919, 1991.

[5] J. Benek, P. Buning, and J.L.Steger, "A 3-d chimera grid-embedding technique," AIAA Paper 85-1523-CP, 1985.

[6] R. Lohner, "Adaptive remeshing for transient problems," *Computational Methods in Applied Mechanical Engineering*, vol. 75, pp. 195–214, 1989.

[7] L. Formaggia, J. Peraire, and K. Morgan, "Simulation of store separation by the finite element method," *Applied Mathemetics Modelling*, vol. 12, pp. 175–181, 1988.

[8] E. Probert, O. Hassan, K. Morgan, and J. Peraire, "An adaptive finite element method for transient compressible flows with moving boundaries," *International Journal for Nummerical Methods in Engineering*, vol. 32, pp. 751–765, 1991.

[9] W. Coirier, K. Powell, and M. Berger, "An accuracy assessment of cartesian mesh approaches for the euler equations," AIAA Paper 93-3335, 1993.

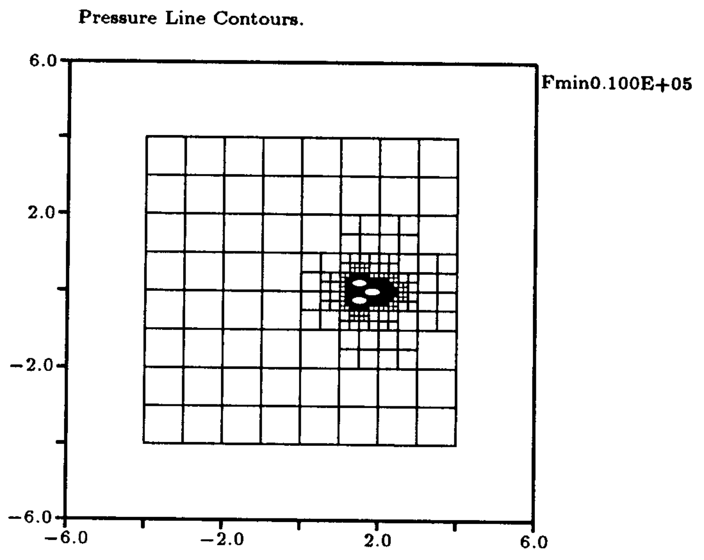

Figure 3: Initial Positions and Geometry.

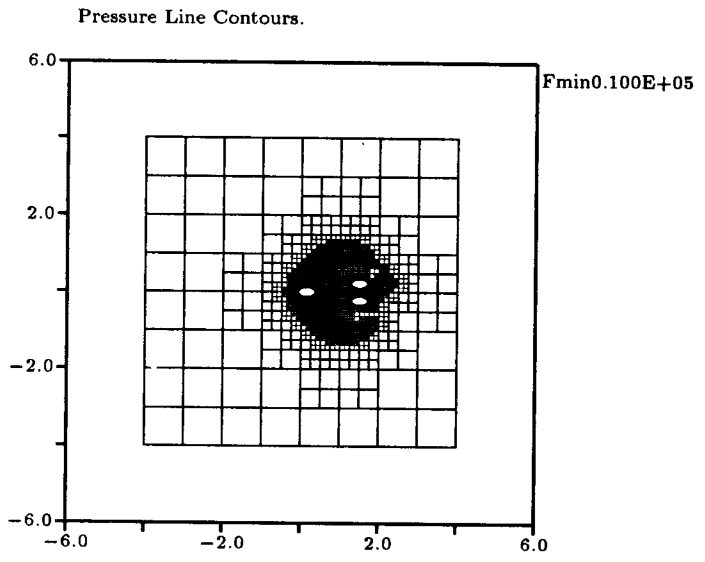

Figure 4: Final Positions and Geometry.

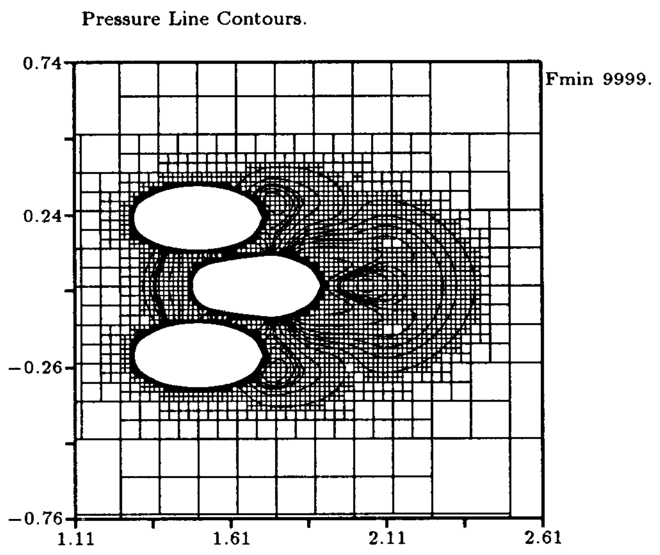

Figure 5: After 100 Time Units

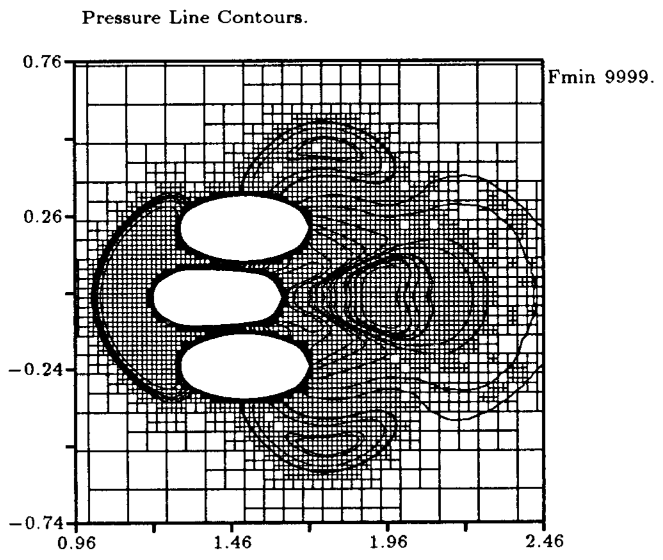

Figure 6: After 200 Time Units

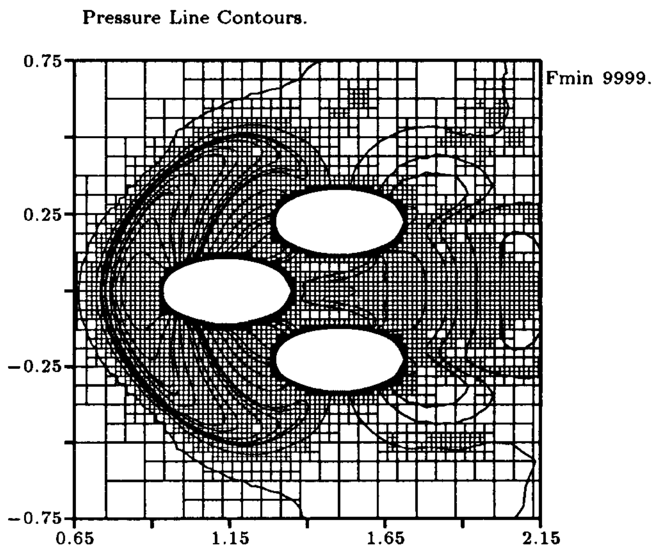

Figure 7: After 300 Time Units

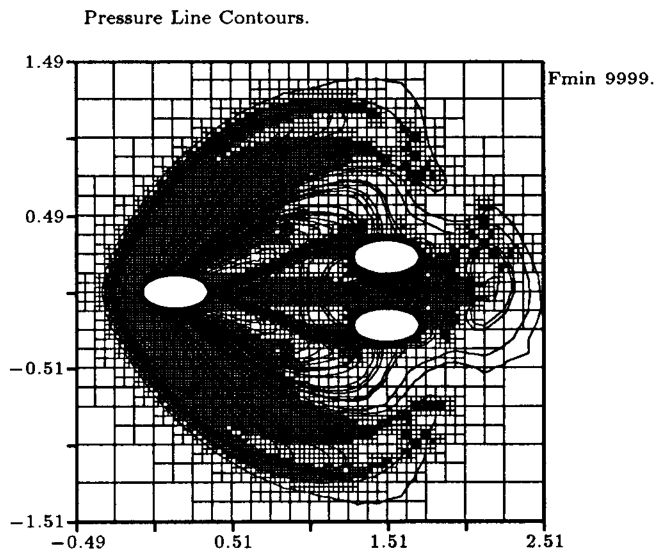

Figure 8: After 650 Time Units

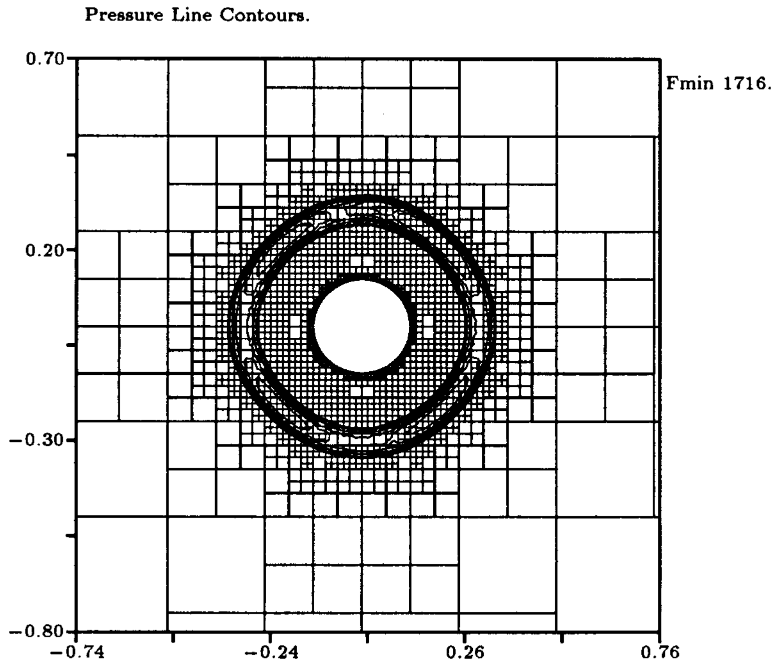

Figure 9: After 50 Time Units.

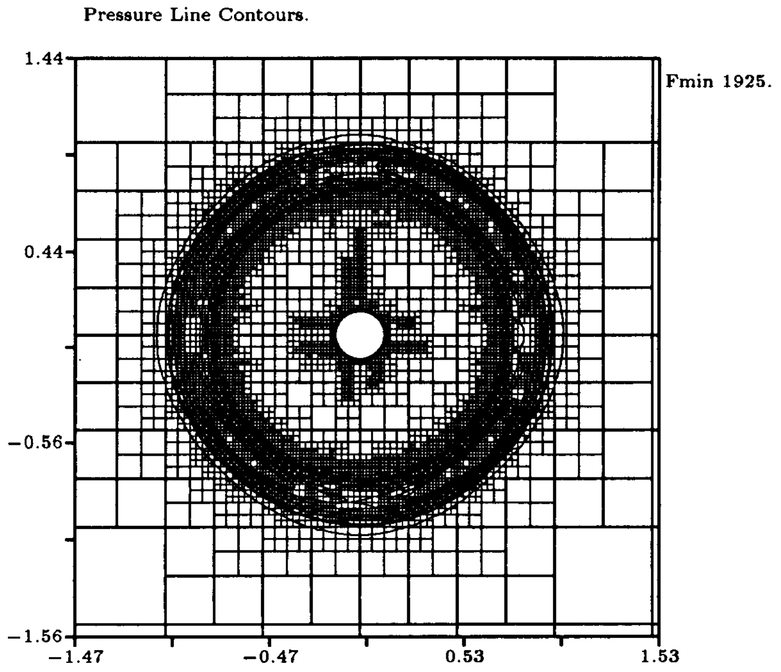

Figure 10: After 150 Time Units.