AIAA JOURNAL
Vol. 31, No. 8, August 1993

Adaptive Refinement-Coarsening Scheme for Three-Dimensional Unstructured Meshes

Y. Kallinderis* and P. Vijayan†
University of Texas at Austin, Austin, Texas 78712

The paper describes the development and application of an adaptive algorithm for tetrahedral grids. An initial unstructured grid is adapted by employing local division, as well as deletion of the tetrahedral cells. The process is dynamic, and the adapted grid changes follow evolution of the solution. Adaptation of the cells consists of normal division of a tetrahedron into eight subcells, as well as directional division into two or four subcells. A major issue of such adaptive grid algorithms is elimination of *hanging* nodes that appear on the edges in the interface between the embedded and unembedded zones. A novel technique has been developed for treatment of such interface cells that can be applied to eliminate any possible hanging node configuration in a simple way. A special data structure based on octrees has been developed, which is flexible to handle all different types of cell division/deletion. Application cases include a moving source, as well as transonic flow around the ONERA M6 wing.

I. Introduction

CONSIDERABLE progress has been made over the past years on the development and application of computational fluid dynamics for simulations of fluid flow. However, computation of flows around three-dimensional bodies remains a challenging issue due to the complexity of the geometry as well as the flowfields. Generation of a body-conforming grid proves to be a difficult task.[1] In cases of structured meshes, a large number of separate blocks need to be defined, and hexahedral grids are generated within each block. In addition, a special algorithm is required to match the grids of neighboring blocks. A radical alternative is to use unstructured grids wherein no single natural coordinate direction exists. The tetrahedron is the simplest element in three dimensions, and therefore it is the most flexible in covering complex topologies.[1-5]

Resolution of the computational mesh plays a crucial role in the accuracy of computations. However, generation of a grid that both fits the flow geometry and resolves the local flow features is quite difficult and even impossible in some cases. In general, the selection of the grid that is to be used in a numerical simulation is determined a priori before starting the solution procedure, and quite often the grid is modified by the user to improve the results. General algorithms have been developed that are flexible enough to adapt the grid during the solution procedure without intervention by the user. Frequently, the regions that require high resolution are very small compared with the size of the overall computational domain. Grid adaptation consists of adjusting the grid spacing so that the truncation error is relatively small and equally distributed throughout the solution domain. An efficient way of accomplishing this is by local grid embedding. An initial coarse grid is embedded in those local regions where relatively large flow gradients exist. Grid embedding with quadrilateral meshes has been employed for inviscid flows,[6,7] as well as for turbulent flows[8-11] in two dimensions. One of the most serious problems with using quadrilaterals has been the presence of grid interfaces, which require special numerical treatment.[8,12] Such treatments can be quite complicated, and they make the adaptation scheme dependent on the numerical method employed. Avoiding such grid interfaces is a major motivation for employing unstructured grids rather than structured ones. Elimination of interface nodes of adapted triangular meshes is relatively simple.[13,14] However, corresponding elimination for tetrahedra is more complex due to the large number of different configurations of interface nodes in three dimensions.

An essential part of every algorithm that employs unstructured grid representation is the data structure that provides information about the connectivities of the cells, faces, edges, and nodes of the grid. The pointers should require the minimum amount of storage possible, while being sufficient for handling all types of unstructured grid configurations. Missing information from the data structure may be recovered by using searches over the grid entities, which is usually very expensive. Furthermore, the computation time per mesh update should be kept to a minimum, and especially so in cases of frequent mesh adaptations. Up to the present, very few algorithms for adaptive embedding of tetrahedra have been developed.[15]

The present paper addresses the aforementioned issues and describes the development and application of a new three-dimensional dynamic adaptive algorithm for tetrahedral grids. The adaptive grid scheme is based on division/deletion of tetrahedral cells. Adaptation of the cells consists of normal division into eight subcells, as well as directional division into two or four cells. A novel technique has been developed for treatment of the interface cells separating different zones of embedded grid, which can be applied to eliminate any possible configuration of interface nodes in a simple way. A special octree-type data structure has been developed, which can handle all possible types of cell division/deletion. Cases involving both stationary and moving features are considered.

In the following section, the grid adaptation algorithm is presented. Then, the data structures developed are described. Effectiveness of the adaptive algorithm is demonstrated via applications. A moving source, as well as transonic flow around the ONERA M6 wing, are the cases considered.

II. Adaptive Grid Method

Adaptive embedding introduces finer cells in those regions of the field that the algorithm senses need to be resolved, while simultaneously removing cells from previously embedded regions that do not require extra resolution. Different types of

Received June 27, 1992; revision received Jan. 17, 1993; accepted for publication Jan. 30, 1993. Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics. Member AIAA.

†Graduate Research Assistant, Department of Aerospace Engineering and Engineering Mechanics.

cell division/deletion are applied. The resulting grid interfaces are eliminated using a special cell-division technique.

A. Octree Division of a Tetrahedron

Grid embedding essentially involves dividing the cell by inserting new nodes. The new nodes are to be placed within the volume of the original cell in such a way that the flowfield variations are resolved. The child cells resulting from such a division should be topologically similar to the parent cell, and increase in grid stretching and skewness compared with the parent cell should be avoided as much as possible. Furthermore, the number of child cells resulting from the division should be kept to a miniumum to maintain storage efficiency while adequately resolving flow features.

The previous considerations led to a cell-division scheme wherein new nodes are introduced in the middle of the edges of the tetrahedra flagged for refinement.[16] Each edge is divided into two edges and each face into four faces after introducing three edges in the interior of the divided face.

Following division of all four faces of the tetrahedron, there are several ways according to which the interior of the cell can be divided into tetrahedra.[16] The main considerations for such a division are the number of child cells, as well as the quality of the resulting grid in terms of skewness and stretching. In the present work, the midface edges on the faces of the original cell are connected, as shown in Fig. 1, to form four faces that are interior to the original cell, namely, the faces constituted by the nodes 5-6-7, 5-8-9, 6-8-10, and 7-9-10. This results in four corner child cells, namely, the cells constituted by the nodes 1-5-6-7, 2-5-8-9, 3-6-8-10, and 4-7-9-10. The interior octahedron, which is constituted by the nodes 5-6-7-8-9-10, is divided into four tetrahedra by the proper choice of the shortest diagonal of the octahedron, which results in the formation of the lowest aspect ratio cells. Furthermore, it also insures that the solid angles of the cells will not become degenerate if the cells resulting from each refinement process are henceforth embedded ad infinitum. This scheme results in forming eight child cells, thus constituting the octree cell structure.

B. Elimination of Interface Nodes

Following division of a portion of the grid cells, the resulting grid contains a number of cells that are left with midedge nodes on some of their six edges due to refinement of the neighboring cells. These *interface* cells constitute the border between the divided and the undivided cells. Numerical schemes employ normal tetrahedral cells with four corner nodes, and significant changes are necessary for the scheme to be applied to interface cells with additional midedge nodes.[12] This is not desired, since then the adaptive algorithm becomes dependent on the specific numerical scheme that is employed.

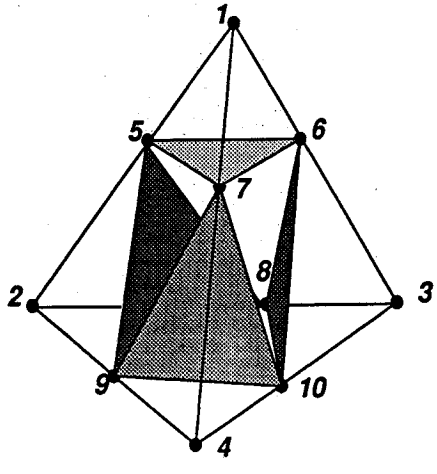

Fig. 1 Typical division of a tetrahedron into eight subcells.

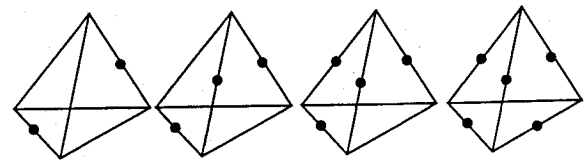

Fig. 2 Representative cases of the 56 possible configurations of hanging nodes.

Hence, a special method of cell division has been incorporated in the adaptive algorithm that eliminates such interface cells.

There are 56 possible configurations of interface nodes, the representative cases of which are shown in Fig. 2. The existence of so many different configurations of cells in which these hanging nodes appear makes any method that treats each configuration separately impractical.

1. Centroidal Node Division

One way to eliminate the interface cells is to introduce additional nodes on the undivided edges of the cell so as to result in a configuration wherein there are midedge nodes on all of the six edges. The interface cell could henceforth be embedded by the standard octree division described earlier. However, such a scheme would result in propagating the interface zone due to the additional grid points introduced. The embedding zone propagation might result in grid refinement over regions where it is not needed and hence makes the scheme inefficient. A new simple method that treats the different cases in a unique general way has been developed in the present work.[17] The method introduces a node at the center of the cell. Then, all of the corner nodes and the midedge nodes are connected to it. The interface cell is now divided as illustrated in Fig. 3, which shows a typical case of a cell with two hanging nodes. The children cells corresponding to face 2,3,4 are C-2-3-6 and C-3-4-6.

2. Directional Division

Two interface node configurations appear frequently, and especially so in cases of directional flowfield variations. Those two cases are treated separately with a simple technique. In the case in which all of the hanging nodes are appearing on the edges of the same face, the interface cell is directionally divided into four children as shown in Fig. 4. Face 2,3,4 is divided into four faces, each corresponding to one of the children cells. In the case of a hanging node appearing on only one of the six edges, the interface cell is henceforth divided into two children as shown in Fig. 5, where face 2,3,4 is divided into two faces. Directional cell division results in a significant reduction in the number of interface cells. The same type of division can be applied to noninterface cells in cases of flowfields with a directional component.

C. Cell Unrefinement

An important function of the developed grid adaptation algorithm has been the deletion of previously divided tetrahedra. Flowfields with evolving flow features are very common, and an adaptive scheme, which eliminates resolution at a region in which additional nodes are not needed any more, is essential. In principle, the unrefinement process is the inverse process of the refinement scheme. The midedge nodes on the "parent" cell edges are removed, resulting in deletion of the corresponding child edges, faces, and cells. Cells that have been flagged to be deleted are eliminated along with their sibling cells that were formed from the same parent cell. In this way, the parent cells are recovered after the deletion of their child cells.

The parent cells are recovered by employing a special pointer. This pointer gives the number of the parent cell for each child cell. Similar pointers give the parent faces and edges. In this way expensive searches through the data structure are avoided. The memory penalty for storing these data

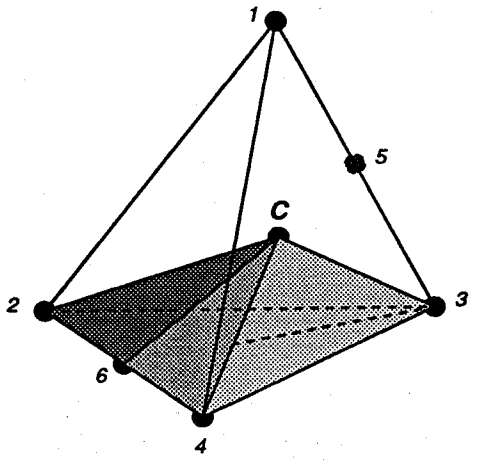

Fig. 3 General interface cell division with centroidal node (two hanging nodes on different faces).

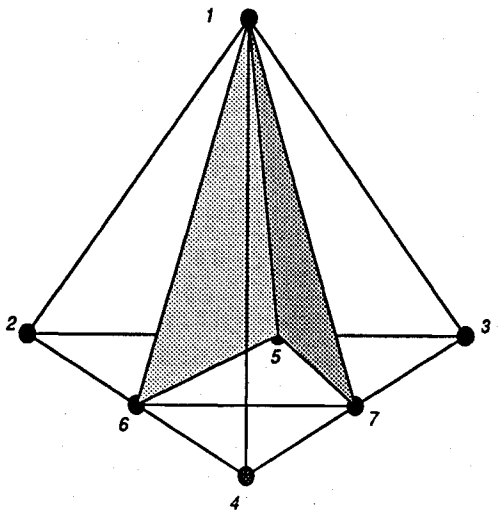

Fig. 4 Directional cell division into four subcells when all three hanging nodes are on the same face.

structures for the parents is not significant, since the number of parent cells is one-eighth of the number of child cells for each embedding level.

The unrefinement process starts at the highest embedding level (finest cells) and henceforth proceeds by visiting the cells marked for deletion in decreasing order of their embedding levels. Cells flagged for unrefinement with siblings flagged for refinement are not deleted. If at least one sibling is flagged for coarsening, whereas the rest are not flagged, then all of them are flagged for deletion. An embedding level difference higher than one is not allowed between two cells that share a common face or edge to avoid multiple hanging nodes at the edges. If this is violated, then the cell that is flagged for deletion is not removed. It should be emphasized that the previous constraint is not restrictive to the extent of hindering the adaptive algorithm from resolving the prominent flow features.

During the unrefinement of any group of sibling cells, the following steps are implemented:

1) A global sweep through all of the *unflagged cells* is done, and the faces, edges, and nodes of these cells are flagged *not to be deleted*.

2) A global sweep through all of the *cells flagged for deletion* is done, and their siblings, as well as their parents, are traced using the pointers.

3) A sweep through all of the traced *parent cells* is done. The faces, edges, and nodes that were introduced *interior* to each parent cell at a prior division of the parent cell are now deleted.

4) This leaves the parent cell with its faces and edges still divided. If these sibling faces and sibling edges are *not* the ones that were flagged in the first step (i.e., they do not belong to another cell that is not marked for deletion), then they are unrefined to form the original parent cell.

5) If these sibling faces and sibling edges had been flagged *not* to be deleted in the first step, then the parent cell cannot be fully reformed, since not all of the midedge nodes can be deleted. Thus the parent cell is marked as an interface cell.

D. Adaptation Procedure

The criterion for division of a cell was based on monitoring the values of the velocity differences and gradients across the cell edges (*detection parameters*). A threshold value for the detection parameters is employed that is set by using the mean and standard deviation of the distribution of each parameter. More specifically, $\Phi_{\text{thre.ref.}} = \Phi_{\text{ave}} + \alpha\Phi_{\text{sd}}$, where Φ_{ave} and Φ_{sd} are the average and standard deviation values of the detection parameter Φ. If the detection parameter is above the threshold for refinement, the edge is flagged for division. Similarly, if the detection parameter is lower than $\Phi_{\text{thre.unref.}} = \Phi_{\text{ave}} - \alpha\Phi_{\text{sd}}$, then the edge is flagged for deletion. Here, α is an empirically chosen value, typically 0.4. This method follows previous work in Ref. 9. An upper bound in number of grid cells is imposed due to computer memory limitations. If this bound is exceeded, then the algorithm increases the value of α. The tetrahedral cells are visited, and if five or six of their edges are flagged for either division or deletion, then the cells are flagged accordingly. An additional layer of cells is added to surround the region to be embedded to insure that the flow feature lies within the refined region. To avoid increase in skewness of the interface cells, those cells are not flagged for further refinement. This results in a gradual decrease in the size of the cells as the finer embedded grids "focus" more on the regions with the highest flow gradients (e.g., shock waves).

The grid-adaptive algorithm consists of the following basic steps:

1) Apply the solver on the current grid.
2) The feature detector identifies the cells to be refined, as well as the cells to be deleted.
3) Delete cells that are flagged for deletion.
4) Divide cells that are flagged for refinement.
5) Update the pointers for the new grid.

The previous procedure can be repeated a number of times until a specified maximum embedding level is achieved. In cases of unsteady flowfields, the adaptation can be invoked at every step.

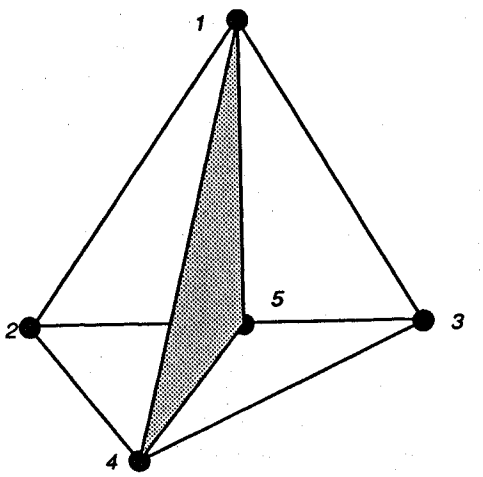

Fig. 5 Directional cell division into two subcells when there is only one hanging node.

III. Data Structure

Implementation of data structures for a dynamic adaptive algorithm is a relatively complicated task. The complexity of the algorithm depends primarily on the amount of grid information available from a previous adaptation pass. This information is made available to the algorithm by means of the data structures that provide the connectivities between the cells, faces, edges, and nodes. In the present work, a new set of data structures has been developed with storage efficiency as the main consideration. Hence, only the connectivity information that is deemed absolutely essential has been included in the data structure.

A. Pointer Arrays

Cell arrays contain the connectivity information that links the cells to their faces, edges, and nodes:

$$ICELNOD\ (i,\ icell),\ i = 1, \ldots, 4$$
$$ICELEDG\ (i,\ icell),\ i = 1, \ldots, 6$$
$$ICELFAC\ (i,\ icell),\ i = 1, \ldots, 4$$

are the pointers that give the nodes, edges, and faces of any element *icell*.

The "depth" of the adapted grid is indicated by the embedding level pointer $LEVEMCEL\ (i)$, $i = 1, \ldots, ncells$:

$LEVEMCEL\ (i) = 0, \ldots,$ cell belongs to the initial grid
$LEVEMCEL\ (i) = n, \ldots,$ cell belongs to the grid of nth embedding level

The adaptive algorithm recognizes a locally fine cell as being flagged for refinement or unrefinement along with its sibling cells depending on the value of the cell flag pointer $IEMB\ (i)$, $i = 1, \ldots, ncells$:

$IEMB\ (i) = 0, \ldots,$ cell is not flagged for division/deletion
$IEMB\ (i) = -1, \ldots,$ cell is flagged for deletion
$IEMB\ (i) = 1, \ldots,$ cell is flagged for division

Face arrays contain the connectivity information that links the faces to edges/nodes and cells. The pointers that indicate the three edges and the two neighboring cells, respectively, of any face are

$$IFACEDG\ (i, face),\ i = 1, \ldots, 3$$
$$IFACCEL\ (i, face),\ i = 1, \ldots, 2$$

The edge arrays provide connectivity information that links the edges to the nodes. The pointer that gives the two nodes of any edge is $IEDGNOD\ (i, edge)$, $i = 1, 2$.

B. Trees

The number of child cells that are formed after the division of the cell are varied. Eight cells are created following normal cell division, whereas four or two are formed in the case of directional cell division. As a consequence, a special tree structure that does not lay any restrictions on the number of child cells that are formed by dividing a parent cell is employed. Four components are identified for each cell to constitute the cell tree.

The first component is the *parent cell*, which is the coarse cell upon the division of which the particular cell is formed. When any parent cell is refined, the first of the group of new fine cells that are formed is its *first child*. The group of cells that are formed from the same parent constitute the *sibling cells*. For any cell, the *right sibling* cell is the one following it and the *left sibling* cell is the one preceding it at the time of forming the respective group of sibling cells. The four components for any cell in the cell tree are indicated by the pointer $IBROCEL\ (i,\ cell)\ i = 1, \ldots, 4$:

$IBROCEL\ (1,\ CELL)$ = parent cell
$IBROCEL\ (2,\ CELL)$ = first child
$IBROCEL\ (3,\ CELL)$ = left sibling
$IBROCEL\ (4,\ CELL)$ = right sibling

An example of the tree corresponding to an adapted mesh is shown in Fig. 6. The shaded circles denote embedded cells. Use of pointer *IBROCEL* is also illustrated in the figure.

At the time of dividing a parent cell, all of the siblings are formed in sequential order. However, in cases in which a refined grid at a later time is partially coarsened, *holes* are created. The existing cells are *swapped* into the locations of the holes to insure that the numbers of all cells in the grid are in sequential order. The swapping is done by simply renumbering the last cell in the list. The new number is the number of the hole. This swapping may disturb the sequential order of the sibling cells of the same group. The cell tree that is employed is flexible enough to handle this, and it does not entail any requirements as regards the sequential order of sibling cells or the number of sibling cells that belong to the same parent.

After a face is divided, it is marked as a coarse face, and the group of four sibling faces (two in case of directional division) that are formed are assigned to the face tree pointer $IBROFAC\ (i,\ face)$, $i = 1, \ldots, 4$. Similarly, a divided edge is marked as a coarse edge, and the two new sibling edges that are formed are assigned to the edge tree pointer, $IBROEDG\ (i,\ edge)$, $i = 1, 2$.

The required storage M for the data structures used is

$$M = 23 \times C + 13 \times F + 7 \times E$$

where C, F, and E are the numbers of cells, faces, and edges, respectively. Approximate relations between the number of cells C, edges E, faces F, and nodes N of a tetrahedral mesh are $E \simeq C + N$, $F \simeq 2C$, and $C \simeq 5N$. Based on these relations, the total memory requirement for the adaptive algorithm, accounting for all of the pointers is *56 integer words per cell*.

IV. Applications

The developed adaptive algorithm is tested through a number of cases designed to evaluate its effectiveness. The first case of a moving point source in a channel tests the dynamic grid adaptation scheme with grid refinement implemented in conjunction with the grid coarsening. Then, the case of transonic inviscid flow around the ONERA M6 wing evaluates accuracy of the adaptive algorithm via comparisons with the experiment. The measured computing time for refinement is 0.0016 CPU seconds per cell division and 0.0085 CPU seconds per cell deletion on a Sun Sparc 330 workstation.

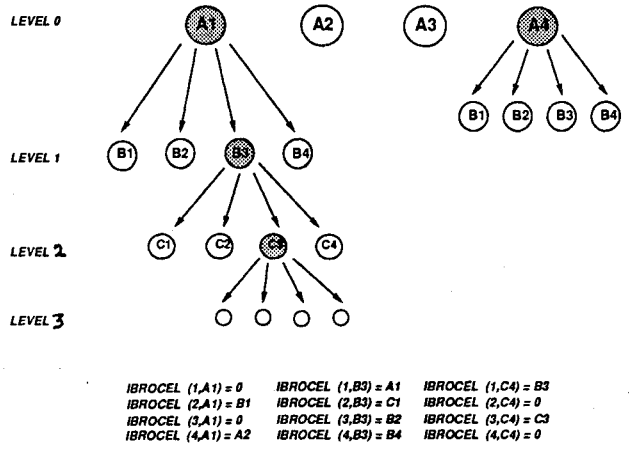

Fig. 6 Cell tree of an embedded grid and corresponding pointer.

The flow solver employed is an explicit, Lax-Wendroff-type, vertex-based finite volume scheme.[17] The coupling of the solver with the adapter is quite simple. The adapter reads the current solution from the solver and generates the new embedded grid. This new grid is read by the solver, which interpolates the solution to the newly introduced nodes and continues the flow simulation.

A. Moving Source

The adaptive coarsening capability is an essential part of the algorithm developed. Frequently, flowfields contain features that move with time. An algorithm, which is capable of refining only, would yield a globally embedded mesh that is prohibitively expensive to use. Therefore, grid coarsening plays a crucial role in keeping the high resolution locally within the region of the moving feature. To demonstrate the cell-deletion feature of the present scheme, a source in a channel is considered to be moving in the x direction. The source induces velocities V that are given by the formula $V = 1/r^2$. The flowfield, as shown in Fig. 7, has a very sharp velocity gradient in the immediate vicinity of the source, and the gradients diminish rapidly far away from the source. The isometric view of the initial grid employed is shown in Fig. 8. The source is moved through the channel in incremental steps. The computational domain is bound by the coordinates $x = 0$, $y = 0$, $z = 0$ and $x = 14.5$, $y = 4.5$, $z = 4.5$. The size of the grid is $30 \times 10 \times 10$ grid points and 11,745 tetrahedral cells. The source is initially located at the point $x = 1.25$, $y = 2.25$, $z = 2.25$ and is moved, parallel to the x axis, in incremental steps of $\Delta x = 3.0$, $\Delta y = 0$, $\Delta z = 0$.

The first grid adaptation yields 4499 nodes and 21,935 cells, whereas the second adaptation yields 6594 nodes and 37,001 cells. Then, the third adaptation yields 8345 nodes and 49,111 cells, whereas 9870 nodes and 59,791 cells appear in the grid after the final adaptation. In the present case, use of grid embedding only would have resulted in a globally refined mesh composed of 132,600 cells. This represents a saving factor in number of cells of 2.25. The number will be considerably larger for multilevel embedding cases. The results of the grid adaptation are shown in Fig. 9. The figure shows three sectional views of the adapted grid. At each section the adapted grid is shown at four locations of the source. The grid sections are taken parallel to the x-y plane and at progressively smaller distances from the source positioned at the midspan. Increasing degree of refinement observed in the planes that are closer to the source show the three-dimensionality of the flowfield. The source is moved in small increments so as to result

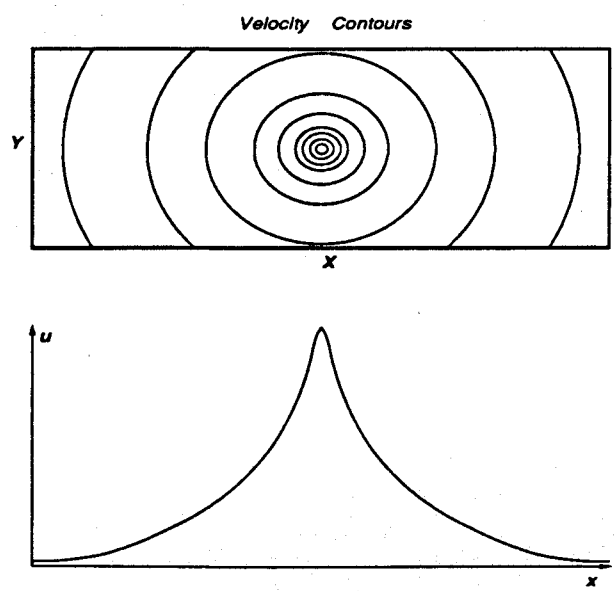

Fig. 7 Velocity field due to a source, at the midspan of the channel.

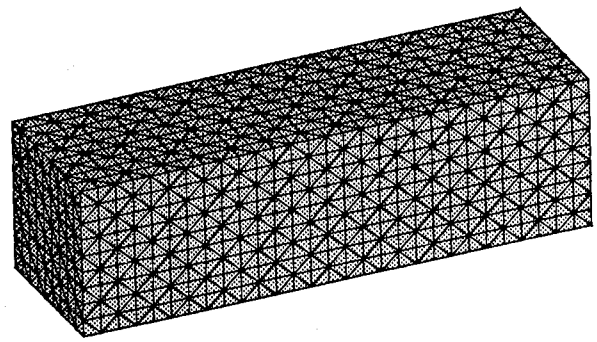

Fig. 8 Isometric view of the channel used to demonstrate the dynamic grid adaptation around a moving source.

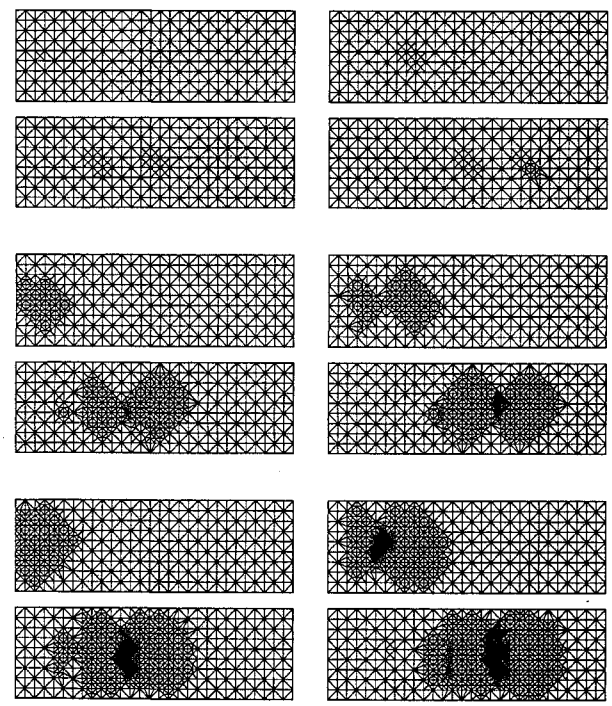

Fig. 9 The x-y planes of the embedded grid at the four different locations of the source: a) section through $z = 0$, b) section through $z = 1$, c) section through $z = 2$.

in an overlap zone between the regions that are marked for embedding in successive adaptations. This results in two levels of embedding in some regions of the grid. When the same region of the grid is embedded up to two levels, even if in an ensuing pass of adaptation the features are sensed to have moved sufficiently far away from that region, the unrefinement scheme coarsens the grid at these regions by only one level in that pass of adaptation. Then, in the succeeding pass it fully unrefines that region. This feature of the unrefinement process is brought forth in Fig. 9. Therefore, the number of adaptation passes required to unrefine a multiply embedded region of the grid equals the number of embedding levels.

B. ONERA M6 Wing

The ONERA M6 wing is considered for evaluating the accuracy of the adaptive flow solution. This configuration has been used as a benchmark case for evaluating the accuracy of several Euler methods. The wing has a leading-edge sweep of 30 deg, an aspect ratio of 3.8, a taper ratio of 0.56, and symmetrical airfoil sections. The wing has a root chord of 0.67 and a semispan of 1.0 with a rounded tip. The computational

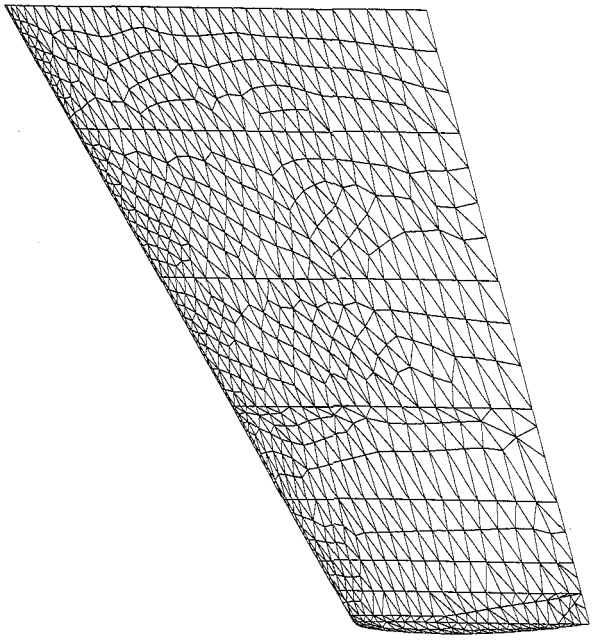

Fig. 10 Triangulation on the ONERA M6 wing upper surface corresponding to the initial coarse grid.

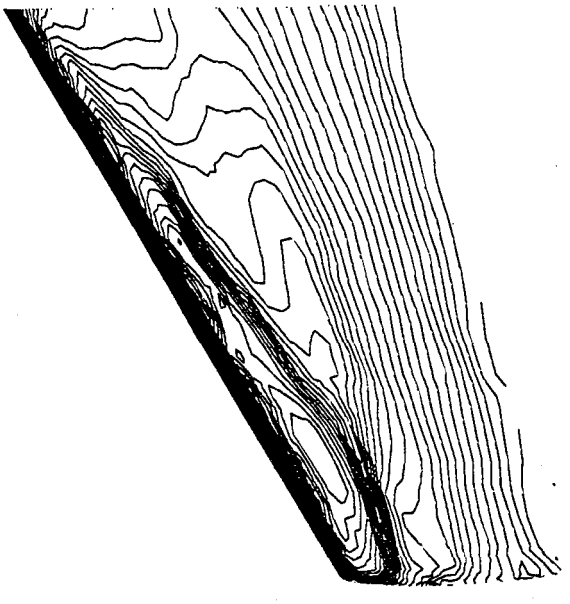

Fig. 11 Mach number contour lines on the wing upper surface; solution obtained on the initial coarse grid ($\Delta M = 0.02$).

domain is bounded by a rectangular box with boundaries at $-6.5 \le x \le 11.0$, $0.0 \le y \le 2.5$, and $-6.5 \le z \le 6.5$. An inviscid, transonic flow solution was computed at $M_\infty = 0.84$ and an angle of attack $\alpha = 3.06$ deg.

The adaptive solution procedure is tested using a relatively coarse initial grid, which was generated by the advancing front method.[18] The grid is adapted twice based on the flow features captured. The initial grid is composed of 35,008 cells and only 6910 nodes. The surface triangulation corresponding to the initial grid is shown in Fig. 10. Figure 11 shows the Mach number contour lines on the wing upper surface, plotted using an increment of $\Delta M = 0.02$. It is observed that the solution features the foreshock only. The shock is quite strong near the wing tip and is diffused toward the symmetry plane. The aft shock is not captured on the initial coarse grid.

The grid is now adapted. The once-adapted grid has 37,123 nodes and 206,577 cells. Figure 12 shows that the first level embedding (light shaded areas) covers a predominant portion of the wing surface. To improve the solution further, the grid is adapted again, and the resulting second level embedded grid on the wing surface is shown in Fig. 12 (darker shaded areas). The second level covers the foreshock and the aft shock, as well as the leading-edge region. The twice-adapted grid has 144,548 nodes and 833,701 cells. It is observed that the number of cells is relatively large. This is due to the fact that the number of tetrahedra is five to six times larger than the number of nodes, and each cell division leads to the creation of eight new cells.

Mach number line contours in Fig. 13 show that the foreshock is a lot sharper and is very well captured in comparison with Fig. 11. The aft shock is now captured well up to span-

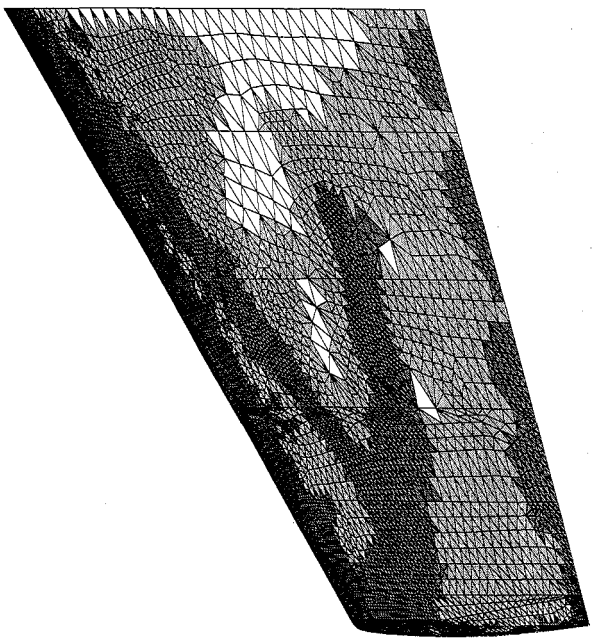

Fig. 12 Triangulation on the wing upper surface corresponding to the twice adapted grid. Light shaded area denotes the first level and the dark shaded area denotes the second level.

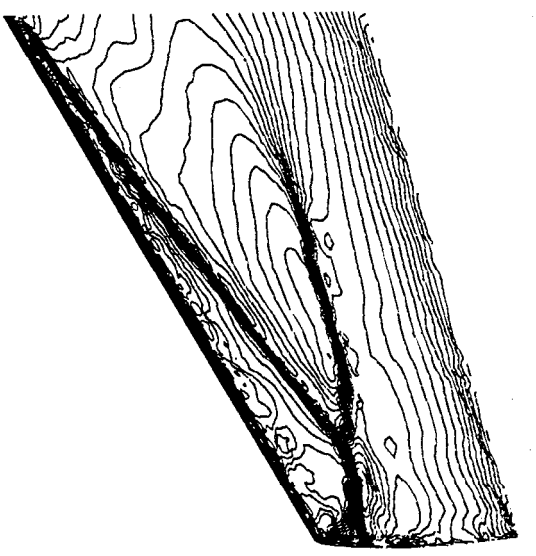

Fig. 13 Mach number contour lines on the wing upper surface; solution obtained on the twice-adapted grid ($\Delta M = 0.02$).

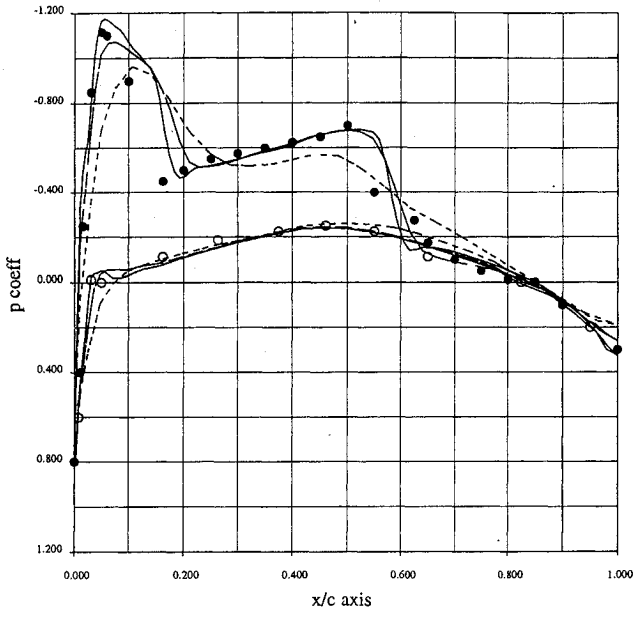

Fig. 14 Pressure coefficients comparison on the wing surface at $\eta = 0.44$ spanwise location: •, experimental values (upper surface); o, experimental values (lower surface); - - -, initial grid solution; - - -, one-level adapted grid; ——, two-level adapted grid.

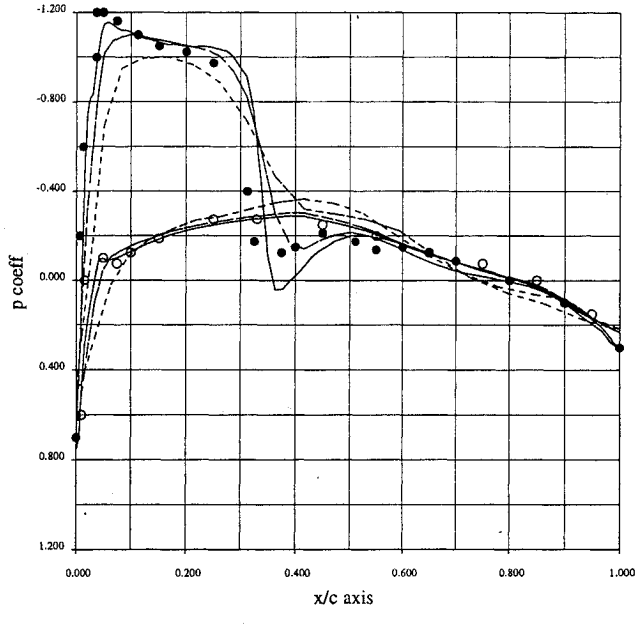

Fig. 15 Pressure coefficients comparison on the wing surface at $\eta = 0.90$ spanwise location: •, experimental values (upper surface); o, experimental values (lower surface); - - -, initial grid solution; - - -, one-level adapted grid; ——, two-level adapted grid.

wise location $\eta = 15\%$. Improvement in the error indicator is needed to place embedded grids over the relatively weak aft shock close to the symmetry plane. Figures 14 and 15 present the comparison of the pressure coefficients on the wing surface obtained by experiments and by the current scheme, at two different spanwise locations. The values corresponding to the initial coarse grid calculation are shown in the short dashed lines. It is seen that the computed values are quite different from the experimental data at both spanwise locations and that there is no indication of a clear aft shock. The numerical results corresponding to the once-adapted grid are shown with long dashed lines, whereas the ones corresponding to the twice-adapted mesh are illustrated with solid lines. It is observed that the shocks become sharper with increasing embedding levels and that the numerical results are getting quite close to the experimental data after two adaptations.

V. Summary

A three-dimensional grid-adaptive algorithm for unstructured tetrahedral meshes was presented. Cell division, as well as cell deletion, was employed to adapt the mesh to flow solutions. The refinement algorithm was general and treated "normal" cells that were divided into eight subcells, as well as cells that contained hanging nodes. Directional division was also applied. The cell-division method prevented unnecessary extension of the embedded region, which would have resulted in considerable deterioration of efficiency of the algorithm. The developed data structure handled all different types of cell division/deletion and required memory storage of 56 integer words per cell. The computing time required for refinement and for coarsening on a Sun Sparc 330 station was 0.0016 CPU seconds per cell division and 0.0085 CPU seconds per cell deletion. Applications considered both stationary and moving features. An adaptive grid was created for transonic flow around a wing. Solution accuracy was improved following each adaptation of the mesh. For the case of a moving feature, the dynamic adaptation scheme was shown to be very effective in restricting the refinement to the vicinity of the local flow features. The adaptive algorithm is independent of the specific geometry and flow solver that is employed. Future work will focus on development of error indicators that are capable of detecting both strong and weak flow features.

Acknowledgment

This work was supported by the Air Force Office of Scientific Research Grant 91-0022, which was monitored by L. Sakell.

References

[1]Baker, T. J., "Developments and Trends in Three Dimensional Mesh Generation," *Applied Numerical Mathematics*, Vol. 5, March 1989, pp. 275–304.

[2]Halt, D. W., and Agarwal, R. K., "A Compact Higher Order Euler Solver for Unstructured Grids with Curved Boundaries," AIAA Paper 92-2696, 1992.

[3]Marcum, D. L., and Agarwal, R. K., "Finite-Element Navier-Stokes Solver for Unstructured Grids," *AIAA Journal*, Vol. 30, No. 3, 1992, pp. 648–654.

[4]Mavriplis, D., "Three Dimensional Unstructured Multigrid for the Euler Equations," AIAA Paper 91-1549-CP, June 1992.

[5]Peraire, J., Peiro, J., Formaggia, L., Morgan, K., and Zienkiewicz, O. C., "Finite Element Euler Computations in Three Dimensions," AIAA 26th Aerospace Sciences Meeting, AIAA Paper 88-0032, Reno, NV, Jan. 1988.

[6]Dannenhoffer, J. F., III, and Baron, J. R., "Grid Adaptation for the 2-D Euler Equations," AIAA Paper 85-0484, 1985.

[7]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, Nov. 1987, pp. 1211–1228.

[8]Kallinderis, Y., "Adaptation Methods for Viscous Flows," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, CFDL-TR-89-5, Cambridge, MA, May 1989.

[9]Kallinderis, Y., and Baron, J. R., "Adaptation Methods for a New Navier-Stokes Algorithm," *AIAA Journal*, Vol. 27, No. 1, 1989, pp. 37–43.

[10]Kallinderis, Y., and Baron, J. R., "A New Adaptive Algorithm for Turbulent Flows," *Computers and Fluids Journal*, Vol. 21, No. 1, 1992, pp. 77–96.

[11]Kallinderis, Y., "Algebraic Turbulence Modeling for Adaptive Unstructured Grids," *AIAA Journal*, Vol. 30, No. 3, 1992, pp. 631–639.

[12]Kallinderis, Y., "Numerical Treatment of Grid Interfaces for Viscous Flows," *Journal of Computational Physics*, Vol. 98, No. 1,

1992, pp. 129–144.
[13]Holmes, D. G., and Connell, S. D., "Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids," AIAA Paper 89-1932-CP, Buffalo, NY, 1989.
[14]Lohner, R., "The Efficient Simulation of Strongly Unsteady Flows by the Finite-Element Method," AIAA Paper 87-0555, 1987.
[15]Lohner, R., and Baum, J., "Numerical Simulation of Shock Interaction with Complex Geometry Three-Dimensional Structures Using a New Adaptive H-Refinement Scheme on Unstructured Grids," AIAA Paper 90-0700, 1990.
[16]Parthasarathy, V., "A Dynamic Grid Adaptation Scheme for 3-D Unstructured Grids," M.S. Thesis, Dept. of Aerospace Engineering and Engineering Mechanics, Univ. of Texas at Austin, Rept. CAR 92-6, Austin, TX, May 1992.
[17]Kallinderis, Y., Parthasarathy, V., and Wu, J., "A New Euler Scheme and Adaptive Refinement/Coarsening Algorithm for Tetrahedra Grids," AIAA Paper 92-0446, Reno, NV, 1992.
[18]Frink, N. T., Parikh, P., and Pirzadeh, S., "A Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes," AIAA Paper 91-0102, 1991.

Recommended Reading from the AIAA Education Series

Gasdynamics: Theory and Applications

George Emanuel

This unique text moves from an introductory discussion of compressible flow to a graduate/practitioner level of background material concerning both transonic or hypersonic flow and computational fluid dynamics. Applications include steady and unsteady flows with shock waves, minimum length nozzles, aerowindows, and waveriders. Over 250 illustrations are included, along with problems and references. An answer sheet is available from the author.
1986, 450 pp, illus, Hardback, ISBN 0-930403-12-6, AIAA Members $42.95, Nonmembers $52.95, Order #: 12-6 (830)

Advanced Classical Thermodynamics

George Emanuel

This graduate-level text begins with basic concepts of thermodynamics and continues through the study of Jacobian theory, Maxwell equations, stability, theory of real gases, critical-point theory, and chemical thermodynamics.
1988, 234 pp, illus, Hardback, ISBN 0-930403-28-2, AIAA Members $39.95, Nonmembers $49.95, Order #: 28-2 (830)

Place your order today! Call 1-800/682-AIAA

AIAA
American Institute of Aeronautics and Astronautics
Publications Customer Service, 9 Jay Gould Ct., P.O. Box 753, Waldorf, MD 20604
FAX 301/843-0159 Phone 1-800/682-2422 9 a.m. - 5 p.m. Eastern

Sales Tax: CA residents, 8.25%; DC, 6%. For shipping and handling add $4.75 for 1-4 books (call for rates for higher quantities). Orders under $100.00 must be prepaid. Foreign orders must be prepaid and include a $20.00 postal surcharge. Please allow 4 weeks for delivery. Prices are subject to change without notice. Returns will be accepted within 30 days. Non-U.S. residents are responsible for payment of any taxes required by their government.