A DYNAMIC MESH ADAPTION PROCEDURE FOR UNSTRUCTURED HEXAHEDRAL GRIDS*

Rupak Biswas†
RIACS, Mail Stop T27A-1
NASA Ames Research Center, Moffett Field, CA 94035

Roger C. Strawn‡
US Army AFDD, ATCOM, Mail Stop 258-1
NASA Ames Research Center, Moffett Field, CA 94035

ABSTRACT

Hexahedral elements can be subdivided anisotropically without mesh quality problems that are associated with tetrahedral meshes. Furthermore, hexahedral meshes yield more accurate solutions than their tetrahedral counterparts for the same number of edges. Our adaption procedure uses an edge data structure that facilitates efficient subdivision by allowing individual edges to be marked for refinement or coarsening. Pyramids and prisms are used as buffer elements between refined and unrefined hexahedra to eliminate hanging vertices. Preliminary results indicate that this new adaption procedure is a viable alternative to adaptive tetrahedral schemes.

INTRODUCTION

Anisotropic mesh adaption is a powerful tool for computing steady and unsteady three-dimensional problems that require local grid modifications in order to efficiently resolve solution features. A number of such methods have recently been successfully developed for local refinement and coarsening of unstructured tetrahedral meshes [1-3]. However, repeated anisotropic subdivision can significantly deteriorate the quality of a tetrahedral mesh. Previous work has demonstrated that isotropic subdivision is required if mesh quality is to be controlled effectively for arbitrary refinement levels in tetrahedral meshes. This is a serious limitation when directional flowfield features are present, leading to an inefficient distribution of grid points in the final mesh. In addition, truly anisotropic subdivision is almost impossible to realize for real problems on tetrahedral meshes.

This work builds on the tetrahedral adaption scheme described in [3] and extends the method to unstructured hexahedral meshes. Hexahedral elements are much better suited for anisotropic adaption than their tetrahedral counterparts. Furthermore, they also yield more accurate flowfield solutions for the same number of edges in the mesh.

The remainder of this paper is divided into five sections. The first section briefly describes the edge-based tetrahedral adaption procedure in [3]. The next section highlights the limitations of tetrahedral mesh adaption schemes. The third section presents our extension to hexahedral meshes. The hexahedral adaption scheme shares much of the logic and data structure with the tetrahedral procedure. It, therefore, also shares much of the software. The fourth section contains some preliminary results that describe the current status of the work. Finally, the last section presents improvements that are being made to the basic hexahedral adaption scheme to reduce the excessive amount of propagation that sometimes occur.

EDGE-BASED TETRAHEDRAL ADAPTION SCHEME

Biswas and Strawn [3] describe the tetrahedral mesh adaption scheme, called 3D_TAG, that is used as a framework for the new hexahedral mesh adaption procedure. The 3D_TAG code has its data structure based on edges that connect the vertices of a tetrahedral mesh. This means that each tetrahedral element is defined by its six edges rather than by its four vertices. This edge data structure makes the mesh adaption compatible with Barth's Euler solver [4] as well as facilitates efficient refinement and coarsening. A successful data structure must contain just the right amount of information so as to rapidly reconstruct the mesh connec-

*Paper No. AIAA-96-0027, presented at the 34th AIAA Aerospace Sciences Meeting, Reno, NV, Jan 15-18, 1996.

†Scientist, Member AIAA.
‡Research Scientist, Member AIAA.

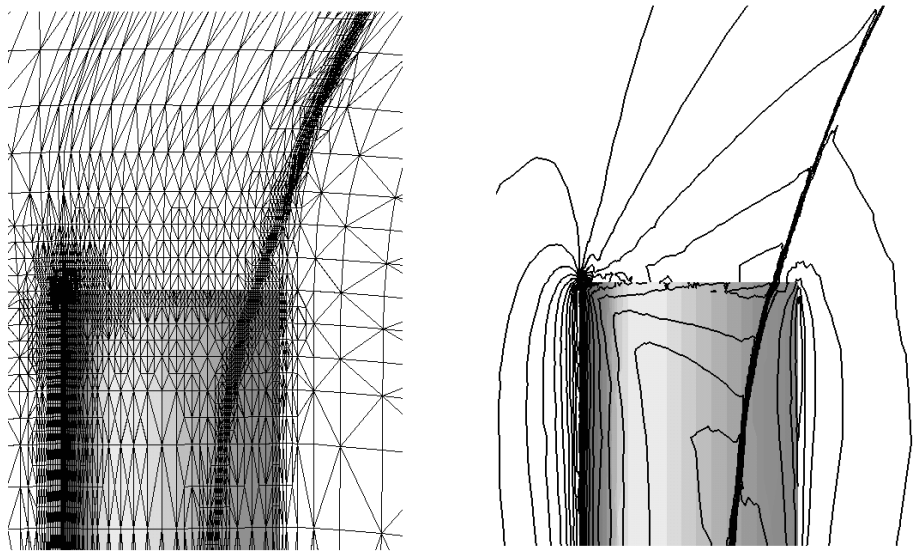

Figure 1: Final mesh and computed pressure contours in the plane of the rotor for a tip Mach number of 0.95.

tivity when vertices are added or deleted but also have a reasonable memory requirement.

At each mesh adaption step, individual edges are marked for coarsening, refinement, or no change. Only three subdivision types are allowed for each tetrahedral element. The 1:8 isotropic subdivision is implemented by adding a new vertex at the mid-point of each of the six edges. The 1:4 and 1:2 subdivisions are used in two ways. First, they can result because the edges of a parent tetrahedron are targeted anisotropically. Second, they are used as buffers between the refined elements and the surrounding coarser grid. These buffer elements are required to form a valid connectivity for the new mesh so that there are no "hanging" vertices.

Mesh refinement is performed by first setting a bit flag to one for each edge that is targeted for subdivision. The edge markings for each element are then combined to form a binary pattern. Elements are continuously upgraded to valid patterns corresponding to the three allowed subdivision types until none of the edge patterns show any change. Once this edge-marking is completed, each element is independently subdivided based on its binary pattern. Special data structures are used in order to ensure that this process is computationally efficient.

Mesh coarsening is also performed using the edge-marking patterns. If a child element has any edge marked for coarsening, this element and its siblings are removed and their parent element is reinstated. The parent edges and elements are retained at each refinement step so they do not have to be reconstructed. Reinstated parent elements have their edge-marking patterns adjusted to reflect that some edges have been coarsened. The parents are then subdivided based on their new patterns. As a result, the coarsening and refinement procedures share much of the same logic.

A significant feature of this adaption scheme is using the concept of "sublists." A data structure is maintained where each vertex has a sublist of all the edges that are incident upon it. Also, each edge has a sublist of all the elements that share it. These sublists eliminate extensive searches and are crucial to the efficiency of the overall adaption scheme.

In [3], the data structure was implemented in C as a series of dynamically-allocated linked lists. This facilitated the addition and deletion of mesh points, but the linked lists made it very difficult to pass information directly to the Fortran flow solver. In order to reduce the communication overhead, the linked lists have been replaced with arrays and a garbage collection algorithm is used to compact free space when mesh points are removed.

This mesh adaption scheme and flow solver have been used to successfully model large problems in helicopter aerodynamics and aeroacoustics [6,7]. An example of such calculations is shown in Fig. 1. This figure shows the adapted mesh and solution for a high-speed hovering rotor blade. Here, the mesh adapts to resolve the leading edge compression, the surface shock, and the acoustic wave that propagates to the far field. Computed results show excellent agreement with experimental data for both near and far-field acoustic pressures.

LIMITATIONS OF TETRAHEDRAL SCHEME

One of the problems with anisotropic subdivision of tetrahedral meshes is that repeated refinement can lead to poor mesh quality. Poor mesh quality is defined as a grid deficiency that leads to an inaccurate flowfield solution. Poor meshes can have disparate element sizes, large face angles, and high vertex degrees. This issue was addressed in [5], where it was concluded that isotropic refinement is required if mesh quality is to be controlled effectively for arbitrary refinement levels. This is a serious limitation when directional flowfield features are present, leading to an inefficient distribution of grid points in the final mesh.

A remedy for this problem is to use hexahedral meshes, which do not have these element quality problems. This is because a hexahedron can be subdivided anisotropically in any of the three directions and yield child elements whose face angles are similar to their parent. This ability to anisotropically refine the mesh makes a tremendous difference in the final problem size when directional flow features are present.

Hexahedral meshes also have an advantage in that they yield more accurate flowfield solutions than their tetrahedral counterparts for the same number of edges. Aftosmis et al. [8] have shown that tetrahedral grids require approximately double the storage and CPU time than hexahedral tessellations of the same vertices. This is due to the fact that tetrahedral meshes have more edges. These additional edges, however, do not contribute significantly to an improvement in solution accuracy.

One disadvantage with unstructured hexahedral grids, however, is that grid generation is not as advanced as that for tetrahedral meshes. Some of this is due to the fact that this area has not received much attention till date. We realize that grid generation for hexahedral meshes is an important issue, but feel that the advantages of hexahedral mesh adaption offset the current lack of highly-developed grid generation tools. We have therefore used structured hexahedral grids as our initial meshes for the test problems reported in this paper.

HEXAHEDRAL ADAPTION SCHEME

Our new hexahedral adaption scheme uses the same basic data structure and the binary edge-marking strategy as the tetrahedral scheme. Instead of a tetrahedral element being defined by its six edges, a hexahedral element is defined by its twelve edges. These edges are ordered in a particular manner such that the binary pattern of marked edges in each element determines how it will be refined. However, the fact that there are twelve edges per hexahedron does not mean that there are 2^{12} different ways that an element can be subdivided! The twelve edges are grouped into three sets where each set consists of the four edges that are disjoint from one another. By handling each group of edges in succession, the total number of different ways an element can be refined is reduced to only 10 distinct cases.

Coarsening of hexahedral elements is also performed using the edge-marking patterns. As in the tetrahedral scheme, if a child hexahedron has any of its edges marked for coarsening, this element and its siblings are removed and their parent element is reinstated. The edge-marking patterns for these reinstated parents are adjusted to reflect that some edges have been coarsened. The parents are then subdivided based on their new patterns. As was the case for the tetrahedral adaption scheme, the coarsening and refinement steps share much of the same logic. In fact, the hexahedral and tetrahedral schemes are so similar, that approximately 80% of the software is shared between the two methods.

Hexahedral adaption schemes generate "hanging" vertices when a hexahedron cannot be split into smaller hexahedra without continuously propagating the mesh refinement into regions where it is not desired. We solve this hanging-vertex problem by using other element types as buffers between refined and unrefined elements. In particular, pyramids and prisms are used to connect up the hanging vertices without unnecessarily propagating the grid refinement. These pyramids and prisms are never subdivided however. If an edge of a pyramid or a prism is marked for subdivision, then the element and its siblings revert back to their parent hexahedron and further refinement is performed directly on the hexahedral element itself. The basic logic for this treatment of buffer elements is identical to the scheme for tetrahedra that is described in [5] when the mesh quality option is turned on.

Sketches of the various types of allowed hexahedral element subdivisions are shown in Fig. 2. The refinement patterns in the top row occur when all four edges in any one or more of the three sets are bisected. This splits the original hexahedron into two, four, or eight smaller hexahedra. Note that all the four disjoint edges in a set are also bisected even if only three edges are marked for refinement. Similar action is taken if two opposite edges in a set are marked.

At most one of the three sets can be marked non-uniformly; that is, only one set is allowed to have one edge or two adjacent edges marked for refinement. The non-uniformly marked set with only one bisected edge generates buffer pyramid elements and these are shown

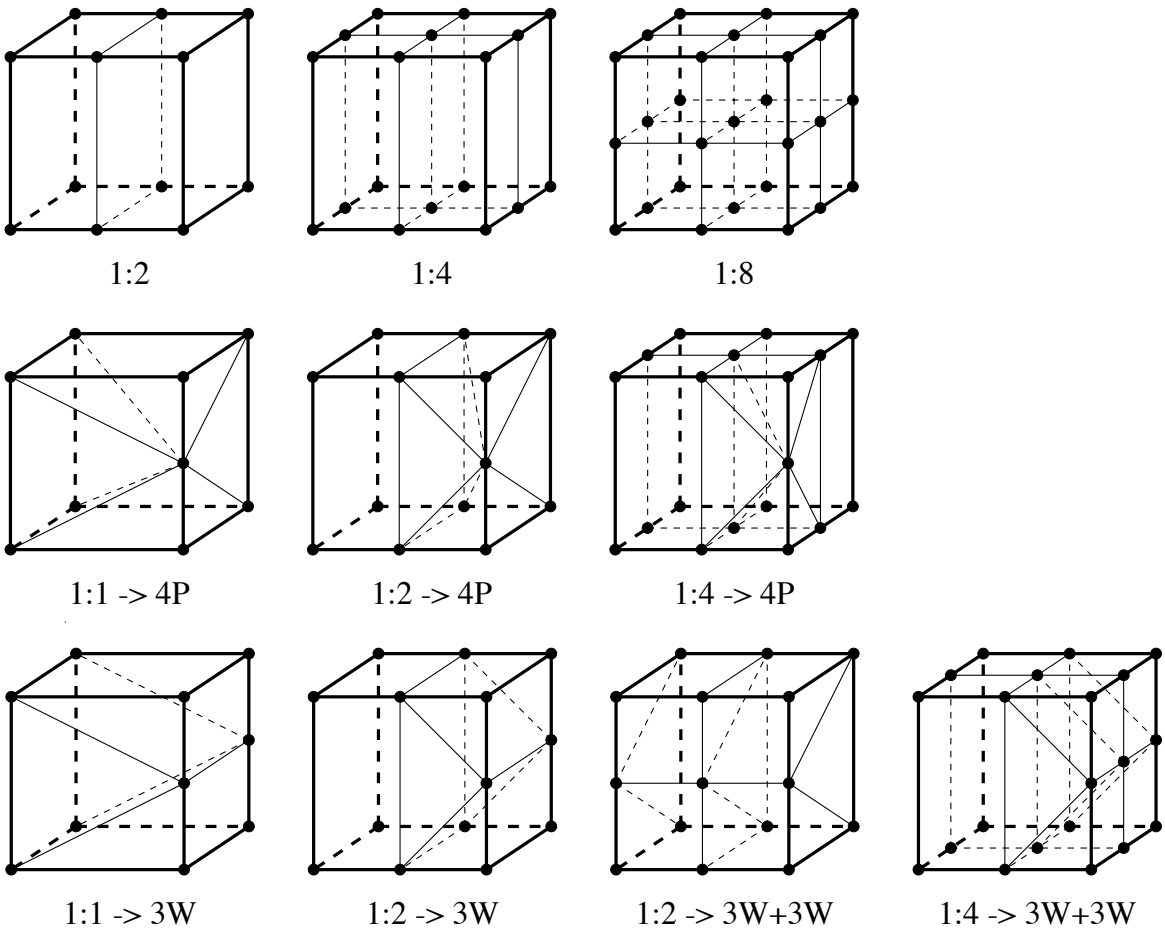

Figure 2: Types of subdivision that are permitted for each hexahedral element.

in the middle row of Fig. 2. Since there is only one set that has exactly one bisected edge, either the parent hexahedron or only one of its children is split into four pyramids.

The non-uniformly marked set with two adjacent bisected edges generates buffer prism elements and these are shown in the bottom row of Fig. 2. Depending on the actual marking pattern, either the parent hexahedron or at most two of its children are split into three prisms each. As mentioned earlier, the prisms and pyramids are never subdivided. They revert to their parent hexahedra if additional subdivision is required.

The Euler flow solver that is used with the resulting mixed-element meshes was developed by Barth [4]. This finite-volume upwind code solves for solution variables at the vertices of the mesh and satisfies the integral conservation laws on non-overlapping polyhedral control volumes surrounding these vertices. Improved accuracy is achieved by using a piecewise linear reconstruction of the solution in each control volume. Computation of the fluxes across each face of the control volume is carried out by summing the contributions from each edge in the mesh. These edges can make up arbitrary polyhedral elements. This means that flowfields on mixed element meshes can be solved just as easily as those on purely tetrahedral grids.

COMPUTED RESULTS

The new hexahedral mesh adaption procedure has been applied to a pair of subsonic and transonic flow problems. Even though both these cases represent steady-state problems, this algorithm is applicable to unsteady flows as well. Steady-flow examples have been chosen as the simplest test cases that fully exercise all aspects of the new mesh refinement/coarsening scheme in three dimensions.

Both test problems consider a NACA 0012 wing that is mounted between two inviscid sidewalls. This is shown in the left portion of Fig. 3. The advantage of these cases is that although the problems are three-dimensional, the results can be easily visualized along

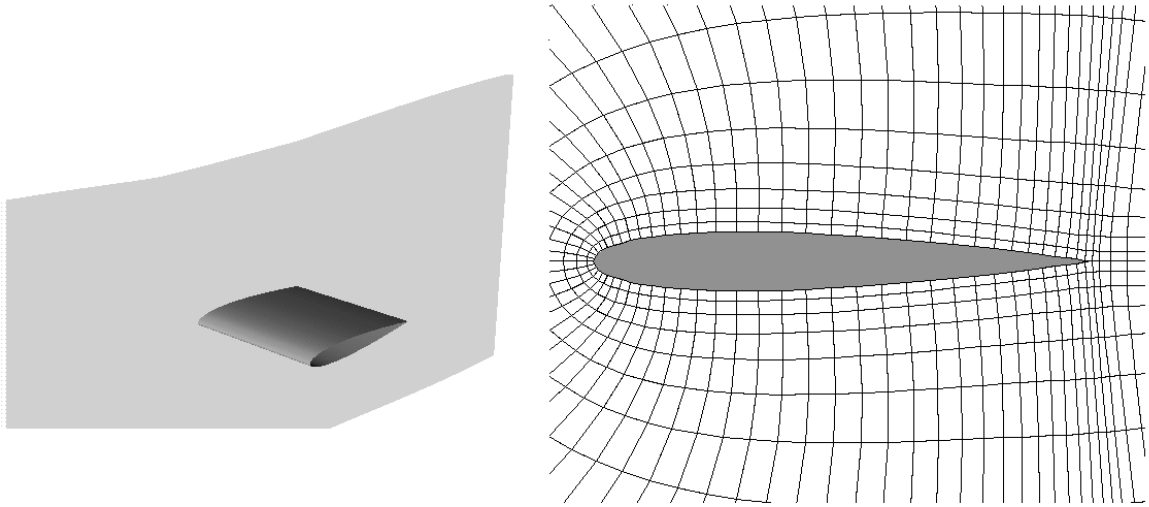

Figure 3: A view of the initial mesh for a NACA 0012 wing mounted between two inviscid sidewalls.

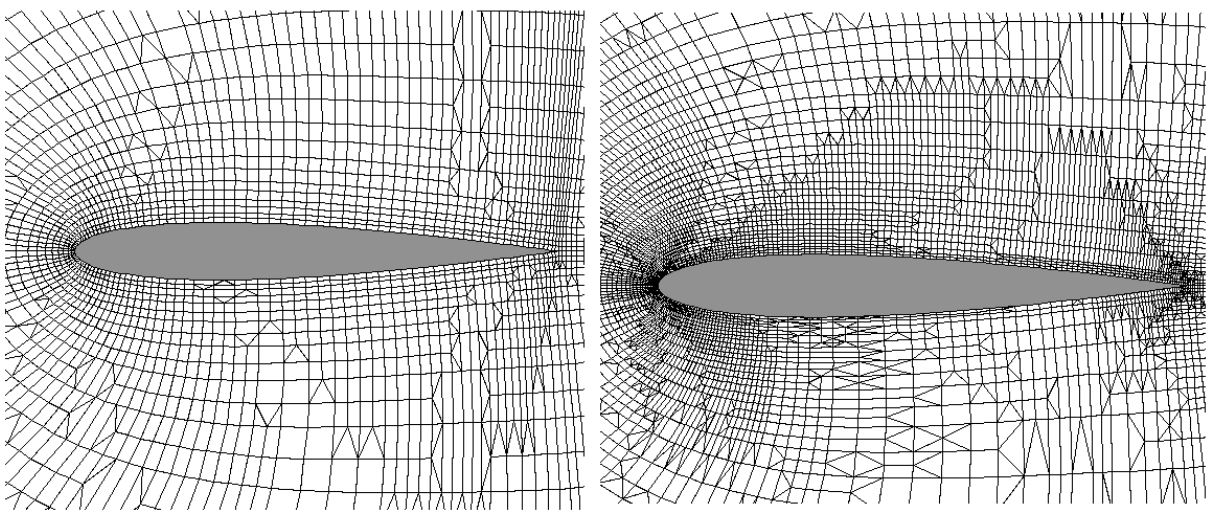

Figure 4: Two mesh adaption stages for the inviscid NACA 0012 wing, $M_\infty = 0.5$, $\alpha = 3^o$.

the sidewalls and compared to existing high-resolution two-dimensional computations.

The initial three-dimensional computational mesh was created from two structured C-H meshes placed one chordlength apart in the spanwise direction. The initial mesh consisted of 4,006 points and 1,900 hexahedral elements. A total of 69 points were located on the surface of the wing at each two-dimensional plane. The outer computational boundaries were located at approximately 20 chords above and below the wing. A view of the initial mesh along a sidewall is shown in the right portion of Fig. 3.

Flow conditions for the first test problem were a free-stream Mach number of 0.5 and a three-degree angle of attack. This subsonic case features no shocks either on the top or the bottom surface of the airfoil.

Figure 4 shows the sequence of mesh refinement and coarsening steps for this problem. The absolute difference in density across each edge of the mesh was used as the adaption indicator. The first mesh refinement marked 4,084 edges for refinement. This resulted in the addition of 5,382 points and 14,179 edges to the mesh. An additional coarsening and refinement step was then performed. We targeted 5,048 edges for coarsening and 4,984 edges for refinement. These values were chosen only to obtain a reasonable solution for the problem. No attempt was made to optimize the adapted mesh for efficiency or the accuracy of the final computed results.

The flow solver was run for approximately 500 it-

erations on each intermediate mesh. The fact that an adapted mesh starts out with the interpolated coarse-grid solution means that it converges rapidly for steady-state calculations, even on fine meshes.

	Initial Mesh	First Adaption	Second Adaption
Points	4,006	9,388	14,056
Edges	9,809	23,988	36,756
Hex elements	1,900	4,161	5,841
Pyr elements	0	0	0
Prm elements	0	792	1,995
Quad bdy faces	4,006	8,596	12,061
Tri bdy faces	0	1,584	3,990

Table 1: Measures of problem size for $M_\infty = 0.5$, $\alpha = 3^o$.

Table 1 lists the number of points, edges, elements, and boundary faces for the initial and the two adapted meshes. Note that there is a very high percentage of boundary faces because of the way the problem has been set up. Every point in the initial mesh is a boundary point. One other interesting feature is that there are no pyramid elements in the adapted meshes. This is due to the fact that this is essentially a two-dimensional problem that has been set up in three dimensions in order to compare the results with existing, high-quality structured-grid solutions.

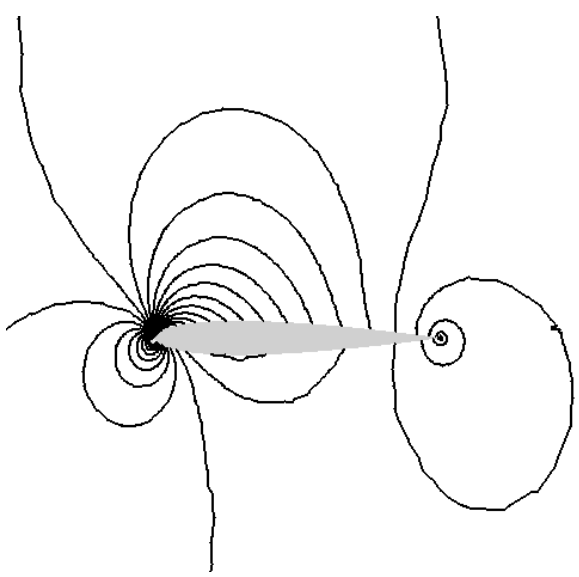

Figure 5: Computed pressure contours at the sidewalls for the solution-adaptive inviscid wing mesh, $M_\infty = 0.5$, $\alpha = 3^o$.

Pressure contours for this case are shown in Fig. 5. The stagnation point at the leading edge is captured with high resolution. Results for the computed pressure coefficient on the final mesh are presented in Fig. 6.

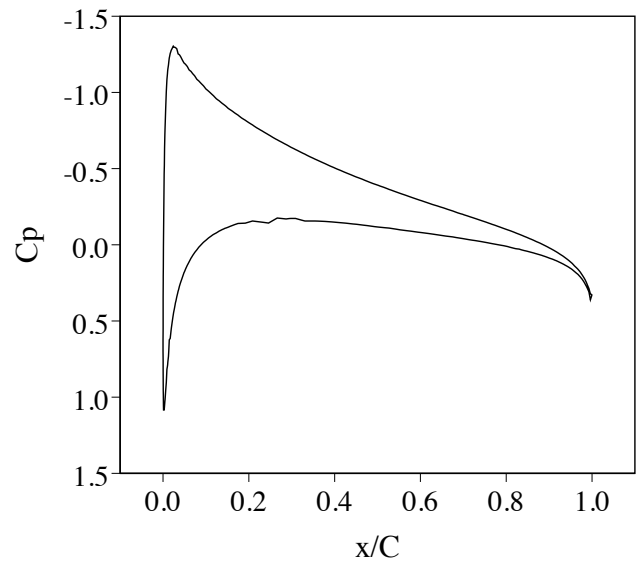

Figure 6: Computed surface pressures for the solution-adaptive inviscid wing mesh, $M_\infty = 0.5$, $\alpha = 3^o$.

Flow conditions for the second test problem using the same wing configuration were a free-stream Mach number of 0.85 and a one-degree angle of attack. Figure 7 shows two mesh adaption steps for this problem. Approximately 4,000 edges were targeted for refinement the first time. A total of 5,038 edges were then coarsened and 4,972 edges were refined. This increased the mesh size to 14,120 points, 36,886 edges, 5,942 hexahedral elements, and 1,911 buffer prism elements.

There is a lack of smoothness in the final mesh resulting from the anisotropic mesh refinement and the fact that relatively few mesh points are used. A smoother mesh can be realized by adjusting the error indicator and/or adding more points to the grid. Additional coarsening and refinement steps can provide a more efficient distribution of points. More effective adaption indicators can also help to smooth out the mesh and target flow regions near the trailing edge.

Pressure contours for this case are shown in Fig. 8. These contours show good resolution of the shock on the upper surface out to the far field. Additional mesh adaption steps will help to sharpen up the shock on the lower surface. Results for the computed pressure coefficient on the final mesh are presented in Fig. 9. These computed results are compared to the AGARD Euler solution No. 9 taken from [9]. Their structured-grid solution used an O-mesh consisting of 320 points on the airfoil surface and 64 points normal to the surface.

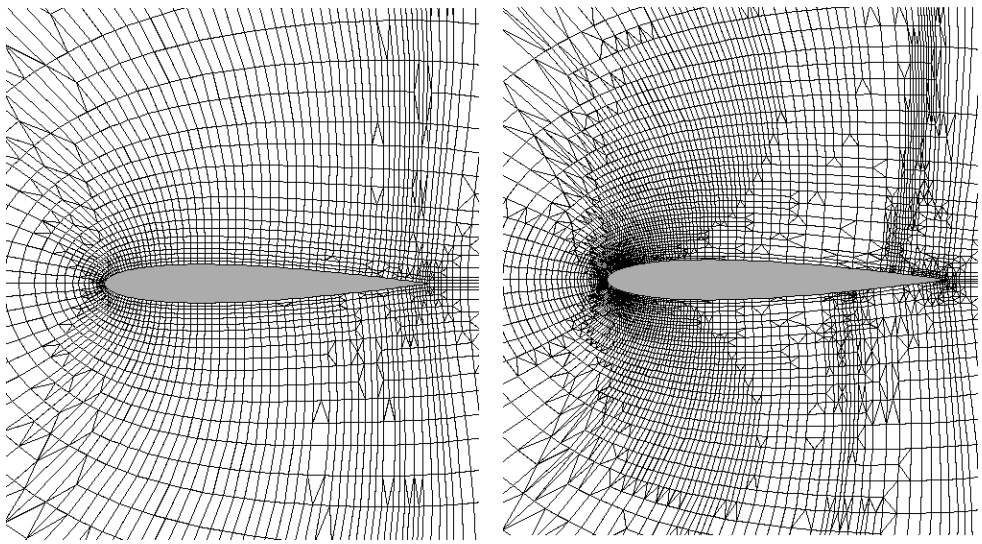

Figure 7: Two mesh adaption stages for the inviscid NACA 0012 wing, $M_\infty = 0.85$, $\alpha = 1^\circ$.

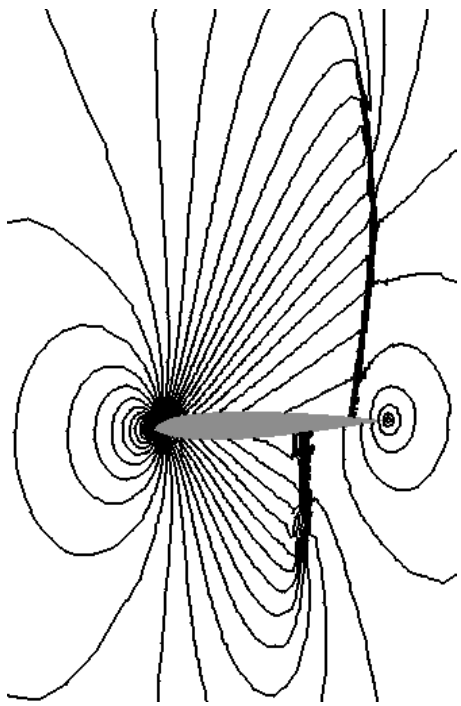

Figure 8: Computed pressure contours at the sidewalls for the solution-adaptive inviscid wing mesh, $M_\infty = 0.85$, $\alpha = 1^\circ$.

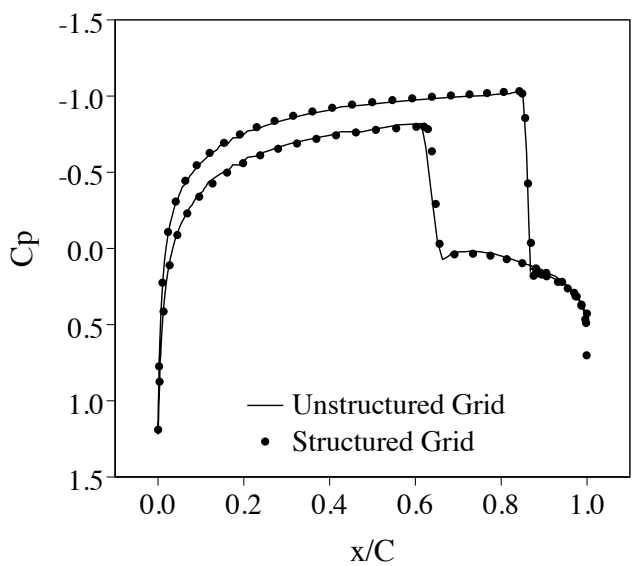

Figure 9: Computed surface pressures from the solution-adaptive unstructured-grid scheme are compared to those from a structured-grid method.

The outer computational boundary was located 25 chords from the airfoil surface. The results in Fig. 9 show excellent agreement between the two solutions. We expect that the slight difference in the shock location on the lower surface will disappear with a better error indicator and more levels of adaption. The purpose of this test case is to show that the solution-adaptive unstructured hexahedral grid calculation has a more efficient mesh distribution than its structured-grid counterpart.

IMPROVEMENTS TO BASIC HEXAHEDRAL SCHEME

One serious problem with this hexahedral adaption scheme is the phenomenon of excessive propagation. As mentioned earlier, at most one of the three sets of edges can be marked non-uniformly. This restriction

requires that elements be continuously upgraded if they have more than one set of edges that are marked non-uniformly. This causes mesh refinement to be propagated into regions where it is not desired.

Figure 10 depicts a scenario where the refinement is propagated because the element edge-markings need to be upgraded to allowable patterns. The top row shows three contiguous elements. The leftmost element has two edges marked for refinement, while the remaining two elements each have only one edge marked. Since only one set of edges can be marked non-uniformly, the leftmost element must have all four edges in one of the two sets marked. This is shown in the second row. However, using the same argument as before, the middle element must now be upgraded. This is shown in the third row. At the next iteration, the rightmost element will have to be upgraded. It is thus obvious that the refinement is being propagated unnecessarily due to the restriction that we have imposed.

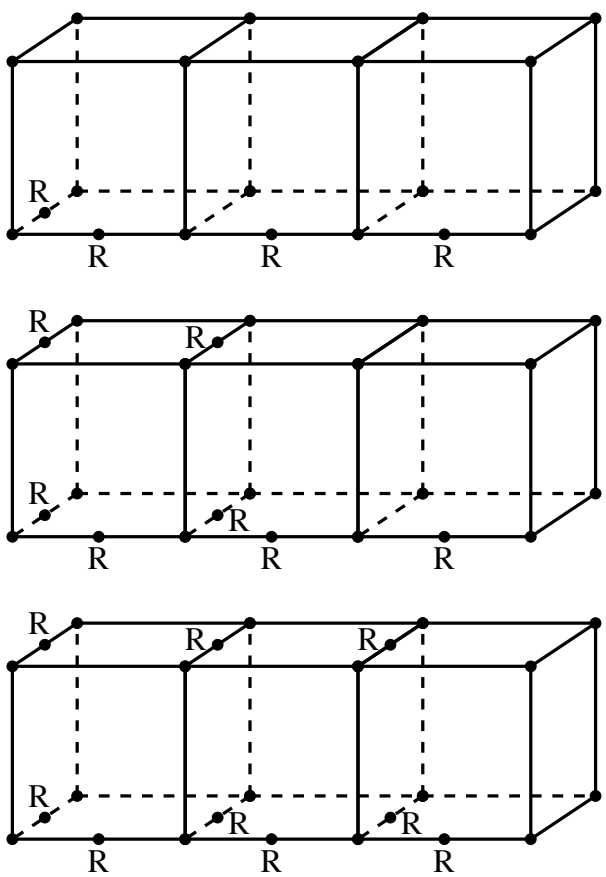

Figure 10: Schematic showing the problem of refinement propagation in one direction.

We solve this problem by removing this restriction altogether. We first refine only those hexahedral elements that will generate smaller hexahedra as dictated by their edge-marking patterns. Any partially-marked element is not upgraded but instead have a new vertex inserted at its center. This center vertex is then joined by edges to the eight vertices of the hexahedron, thereby creating six pyramids as shown in Fig. 11.

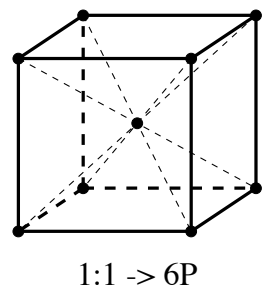

Figure 11: Six pyramid elements are initially created by inserting a vertex at the center of a hexahedron.

Each of these six pyramids are then examined sequentially. Each pyramid has a face of the original hexahedron as its base. The four edges that constitute the face can have one of six distinct refinement patterns. This causes the special subdivisions that are shown in Fig. 12. If none of the edges on the face are marked, we stay with the one pyramid element. If only one edge is marked, three tetrahedral elements are generated. If two adjacent edges are marked, we obtain four tetrahedral elements. However, if two opposite edges are marked, then two pyramid buffer elements are created. If three edges are marked for refinement, then we generate one pyramid and three tetrahedral elements. Finally, if all four face edges are bisected, we obtain four smaller pyramids.

Insertion of this center vertex obviously eliminates any possibility of refinement propagation. No additional edges are marked for subdivision and none of the neighboring hexahedral elements are affected. The entire refinement procedure is contained within a single hexahedron which is properly split into a combination of pyramid and tetrahedral elements to obtain a valid mesh connectivity.

SUMMARY AND CONCLUSIONS

This paper has presented a new method for the dynamic adaption of three-dimensional unstructured hexahedral meshes. The algorithm uses edge-based data structures in order to allow anisotropic refinement and coarsening. Pyramids and prisms are used as buffer elements to interface regions of refined and unrefined hexahedra.

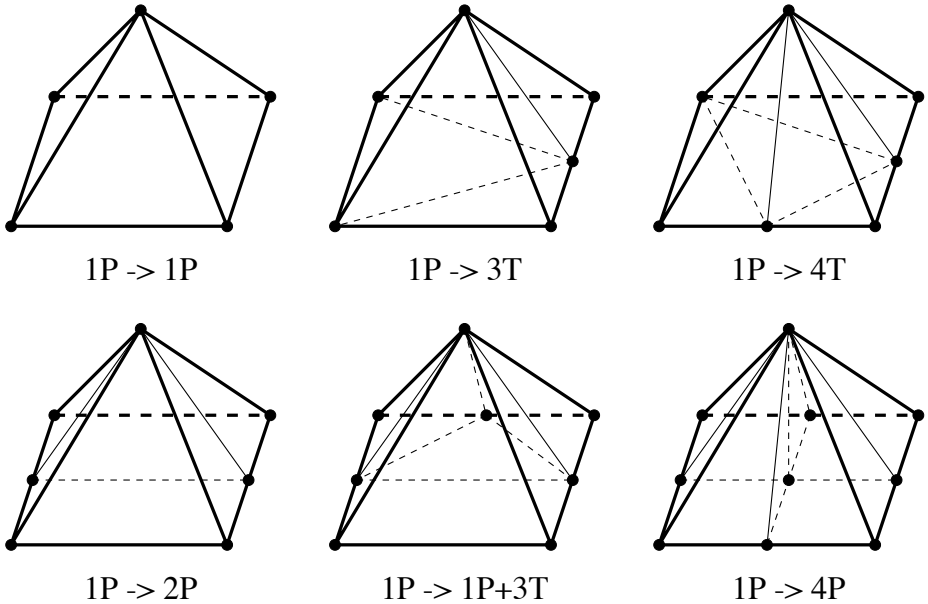

Figure 12: Special subdivision types when inserting a vertex at the center of a hexahedron.

The solution-adaptive scheme has been demonstrated for two sample cases. Computed solutions for the Euler equations show good agreement with results from conventional structured-grid methods.

A problem with excessive propagation of the refined region is eliminated by not upgrading the element edge-marking patterns but instead by inserting a vertex at the center of the hexahedron. The hexahedron is then locally split into a combination of pyramids and tetrahedra to generate a valid mesh. This removes the requirement of having to adjust the refinement strategy for any neighboring hexahedra.

REFERENCES

[1] Kallinderis, Y., and Vijayan, P., "An Adaptive Refinement/Coarsening Scheme for 3-D Unstructured Meshes," *AIAA Journal*, Vol. 31, 1993, pp. 1440–1447.

[2] Rausch, R., Batina, J., and Yang, Y., "Spatial Adaptation Procedures on Tetrahedral Meshes for Unsteady Aerodynamic Flow Calculations," AIAA-93-0670, *AIAA 31st Aerospace Sciences Meeting*, 1993.

[3] Biswas, R., and Strawn, R. C., "A New Procedure for Dynamic Adaption of Three-Dimensional Unstructured Grids," *Applied Numerical Mathematics*, Vol. 13, 1994, pp. 437–452.

[4] Barth, T. J.,"A 3-D Upwind Euler Solver for Unstructured Meshes," AIAA-91-1548, *AIAA 10th Computational Fluid Dynamics Conference*, 1991.

[5] Biswas, R., and Strawn, R. C., "Mesh Quality Control for Multiply-Refined Tetrahedral Grids," *Applied Numerical Mathematics*, to appear.

[6] Strawn, R. C., Biswas, R., and Garceau, M., "Unstructured Adaptive Mesh Computations of Rotorcraft High-Speed Impulsive Noise," *Journal of Aircraft*, Vol. 32, 1995, pp. 754–760.

[7] Duque, E. P. N., Biswas, R., and Strawn, R. C., "A Solution Adaptive Structured/Unstructured Overset Grid Flow Solver with Applications to Helicopter Rotor Flows," AIAA-95-1766, *AIAA 13th Applied Aerodynamics Conference*, 1995.

[8] Aftosmis, M., Gaitonde, D., and Tavares, T. S., "On the Accuracy, Stability, and Monotonicity of Various Reconstruction Algorithms for Unstructured Meshes," AIAA-94-0415, *AIAA 32nd Aerospace Sciences Meeting*, 1994.

[9] Test cases for inviscid flow field methods — report of Fluid Dynamics Panel Working Group 07, AGARD-AR-211, 1985.