RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

IN-38-CR
38026
P-16

Mesh Quality Control for Multiply-Refined Tetrahedral Grids

Rupak Biswas
Roger Strawn

RIACS Technical Report 94.19 November 1994

To appear in *Applied Numerical Mathematics*

Mesh Quality Control for Multiply-Refined Tetrahedral Grids

Rupak Biswas
Roger Strawn

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 212, Columbia, MD 21044, (410) 730-2656

Work reported herein was supported by NASA via Contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA). Work was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035-1000.

Mesh quality control for multiply-refined tetrahedral grids*

Rupak Biswas[a], Roger C. Strawn[b]

[a]*RIACS, Mail Stop T27A-1, NASA Ames Research Center, Moffett Field, CA 94035*

[b]*US Army AFDD, Mail Stop 258-1, NASA Ames Research Center, Moffett Field, CA 94035*

Abstract

A new algorithm for controlling the quality of multiply-refined tetrahedral meshes is presented in this paper. The basic dynamic mesh adaption procedure allows localized grid refinement and coarsening to efficiently capture aerodynamic flow features in computational fluid dynamics problems; however, repeated application of the procedure may significantly deteriorate the quality of the mesh. Results presented show the effectiveness of this mesh quality algorithm and its potential in the area of helicopter aerodynamics and acoustics.

1. Introduction

Two types of solution-adaptive strategies are typically used with unstructured-grid methods. After an initial solution is obtained on a given base mesh, both schemes use some error indicator to mark regions in the flowfield where the solution is either changing rapidly or remaining relatively constant. In the first strategy, the grid is either globally or locally regenerated with a higher concentration of points in the high-error regions and a lower concentration of points in the low-error regions. Although the resulting grids are usually well-formed with smooth transitions between regions of coarse and fine mesh spacing, such schemes are computationally intensive. This is a major drawback for time-dependent problems where the mesh must be adapted at least every few time steps. The second strategy involves local modifications of the existing grid by individually adding points to the mesh where the error indicator is high, and removing points from regions where the indicator is low. The advantage of this strategy is that relatively few mesh points need to be added (deleted) at each refinement (coarsening) step. This makes it particularly attractive for unsteady problems where the solution usually changes rapidly.

*This work was partially supported by the National Aeronautics and Space Administration under Contract Number NAS 2-13721 with the Universities Space Research Association.

However, complex data structures must be maintained in order to keep a history of the points and/or elements that are added and removed at each adaption step.

The numerical scheme in this paper aims to solve three-dimensional problems in helicopter aerodynamics and acoustics. Because of the importance of flowfield unsteadiness for these problems, we have chosen the local grid modification scheme as the basis for our dynamic mesh adaption, paying particular attention to the quality of the resulting grids.

A number of successful methods have recently been developed for the local refinement and coarsening of unstructured tetrahedral grids [2,3,4,6]. In particular, the mesh adaption algorithm of Biswas and Strawn [2] allows anisotropic mesh refinement and coarsening that results in a more efficient distribution of the grid points to optimally resolve directional flow features. This is possible because the algorithm uses the edges of the unstructured mesh rather than the tetrahedral elements as primitives that are primarily targeted for refinement and coarsening. A brief description of this adaption scheme is given in Section 2 for completeness and to highlight the recent improvements that we have made to the original procedure.

However, one of the problems with anisotropic refinement of tetrahedral meshes is that repeated subdivision can lead to poor element quality. This is one area in unstructured mesh adaption that has not been addressed comprehensively in the past. In order to establish dynamic mesh adaption as a powerful tool for computing steady and unsteady problems, a technique is needed to ensure that the mesh does not deteriorate after several levels of refinement. The strategy that we have developed is presented in Section 3.

Computational results are presented in Section 4. The first test case is a simplified problem that highlights the beneficial effects of our mesh quality logic. The second test case is a more realistic problem where our mesh adaption code is used to model the high-speed impulsive noise for a non-lifting rectangular helicopter rotor blade in hover. Finally, some conclusions drawn from this work are presented in Section 5.

2. Tetrahedral Adaption Procedure

Biswas and Strawn [2] describe the basic tetrahedral mesh adaption scheme that is used in this paper. The code, called 3D_TAG, has its data structure based on edges that connect the vertices of a tetrahedral mesh. This means that the elements and boundary faces are represented by their edges rather than by their vertices. This edge data structure makes the mesh adaption compatible with Barth's Euler solver [1] 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 connectivity 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 and these are shown in Fig. 1. The standard 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. They can either result because the edges of a parent tetrahedron were targeted anisotropically or 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.

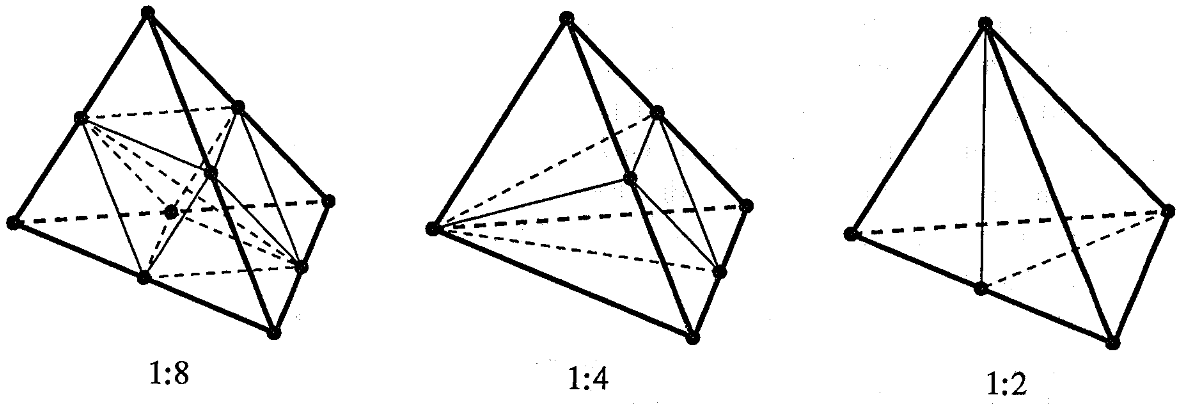
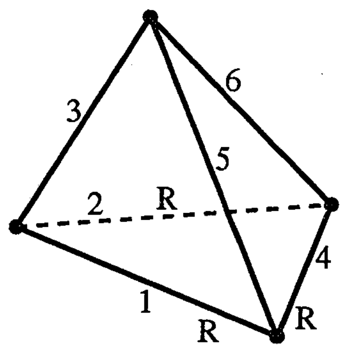


Fig. 1. Three types of subdivision are permitted for a tetrahedral element.

Each tetrahedral element is defined in terms of its six edges. 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 can then be combined to form a binary pattern as shown in Fig. 2 where the edges marked with an R are the ones to be bisected. Once the edge marking is completed, each element can be independently subdivided based on this binary pattern. Special data structures are used in order to ensure that this process is computationally efficient.



6 5 4 3 2 1	Edge number
0 0 1 0 1 1	Pattern = 11

Fig. 2. Sample edge-marking pattern for element subdivision.

Mesh coarsening is also performed using the binary 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 in the data structure at each refinement step so they do not have to be reconstructed. The parents have their edge-marking patterns adjusted to reflect that some edges have been coarsened. The parents are then subdivided based on the new patterns. As a result, the coarsening and refinement procedures share much of the same logic.

There are a couple of constraints for mesh coarsening. First, edges cannot be coarsened beyond the initial mesh. If that were permitted, some mesh regeneration logic would have to be used. Second, edges must be coarsened in an order that is reversed from the one by which they were refined. This is a reasonable restriction that allows only certain edges to coarsen at each mesh adaption step.

A significant feature of this mesh 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. Similarly, 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 [2], the data structure was implemented in C as a series of four dynamically-allocated linked lists for the vertices, edges, elements, and boundary faces of the mesh. 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. Also, a significant amount of CPU time was wasted by using the standard C library functions `malloc` and `free` each time an item was added to or deleted from the data structure. In order to reduce the communication overhead, the linked lists have been replaced with arrays. These sequential arrays are dynamically allocated at the beginning of a mesh adaption step based on the size of the current mesh and an user-estimated fraction of the mesh that is targeted for refinement. A garbage collection algorithm is used to compact free space at the end of every coarsening step.

To provide an estimate of the total memory requirements of our mesh adaption procedure, let us assume that the number of edges is six times and that the number of elements is five times the number of vertices. In that case, the adaption algorithm requires 160 integers per node beyond the storage that is directly required by the Euler flow solver. This number does not include the storage overhead for retaining the parent elements and edges. This overhead increases with the number of refinement levels and is typically about 15% after three adaption steps. The estimate also does not include the storage requirements for the boundary faces because it is negligible for large three-dimensional problems.

3. Mesh Quality

One problem with the anisotropic subdivision of tetrahedral elements is that repeated refinement can lead to poor mesh quality. Poor mesh quality is defined as a grid deficiency that leads to inaccurate flowfield solutions. Poor meshes can have disparate element sizes, large face angles, and high vertex degrees. (The degree of a vertex is defined as the number of edges that are incident upon it.) Examples of triangular elements of inferior quality are shown in Fig. 3. Note that an element that only has a high aspect ratio will not be considered to have poor quality. Such elements occur in boundary layers where the flowfield gradients are highly directional. For those cases, high aspect ratio right tetrahedra are excellent choices for the computational mesh.

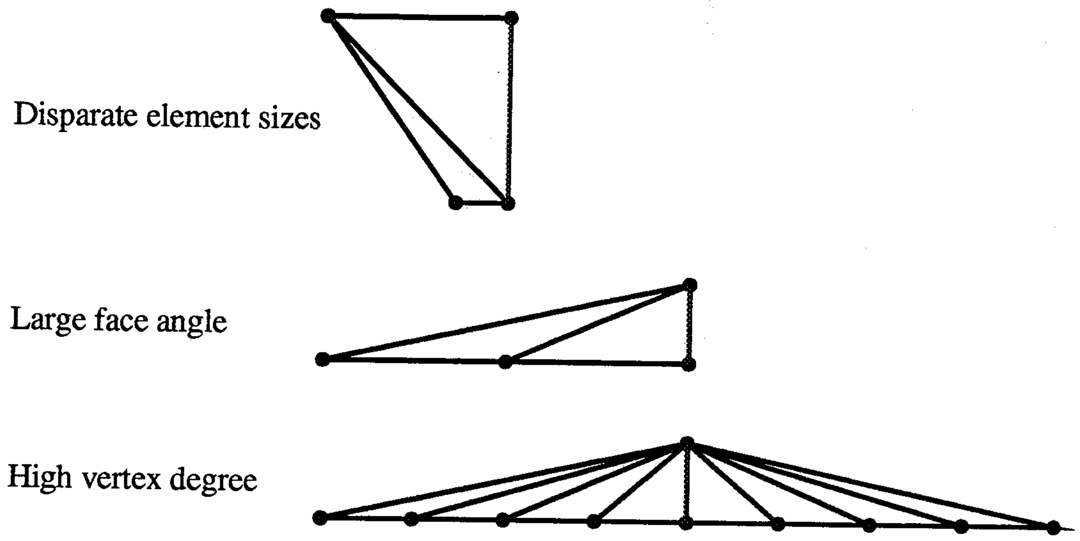


Fig. 3. Examples of two-dimensional elements of poor quality.

A two-dimensional example of poor mesh quality is shown in Fig. 4. This is not a contrived case but one that we actually encountered when testing our mesh quality techniques on two-dimensional viscous meshes. Fig. 4(a) shows that all three edges of the bottom two triangles have been marked for refinement. The lowermost edge is on the solid boundary and is not shared by any other element. Fig 4(b) shows the resulting mesh after the edge subdivision and subsequent triangulation. A large angle θ has been created from this refinement. This results from the 1:2 subdivided triangle that acts as a buffer between the 1:4 subdivided triangle and the original mesh. If the element aspect ratio is 50, the angle θ is about 178°. This large angle can lead to inaccuracies in the flow solver if it is located in a region where the flowfield is changing rapidly.

Edge swapping is commonly used to improve the quality of a mesh. It is not a panacea however, and its effects can sometimes propagate with undesirable results. This is particularly true for grids with high aspect ratio elements. When edge swapping is repeatedly used to remove the large angle θ in Fig. 4(b), the results are surprising. Fig. 4(c) shows the mesh after the edge swapping procedure is applied. The large angle has been reduced but the maximum vertex degree has increased and there are large variations in element sizes. This is clearly a poor mesh. All of these problems can be removed by locally regenerating the mesh, but only at the cost of adding more vertices. No straightforward methods exist within the framework of local mesh adaption in order

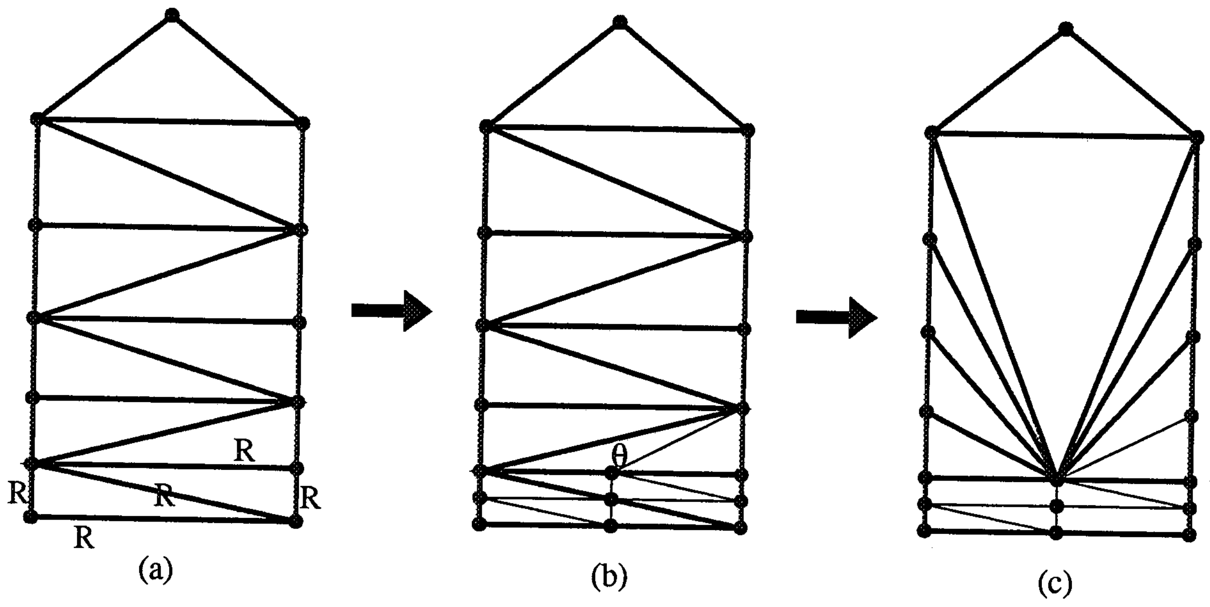


Fig. 4. An example of mesh quality problems for subdivided triangular grids.

to improve the quality of the mesh in Fig. 4(b).

Given these limitations, we tested several strategies to control the quality of the mesh. For all the strategies, it is assumed that the given initial mesh has acceptable element quality and will produce smooth flowfield solutions. This means that the children of elements that are isotropically subdivided (1:8) will have characteristics similar to those of their parents. Thus, the element quality problem is limited only to elements that are anisotropically refined (1:4 and 1:2).

Of all the strategies that we tested, only two are worth reporting. In the first strategy, there are no restrictions when an element with acceptable quality is anisotropically subdivided 1:4 or 1:2. As mentioned earlier, this anisotropic subdivision can cause any one or more of the features of poor meshes shown in Fig. 3 to appear in the child elements. Our algorithm allows these to occur, but marks these child elements as they are created. Child elements that have poor quality are never allowed to be subdivided again, since this could lead to even more problems. If a child element with poor quality has any of its edges marked for refinement, this element and its siblings immediately revert back to their parent. The parent is then isotropically subdivided (1:8), and at this point, all of its children have acceptable element quality. The edges of these eight children can then be marked for refinement in the conventional manner.

One problem with this strategy is when only some of the children of an anisotropically-subdivided element are of poor quality. If the children that are of acceptable quality are marked for further refinement, they are allowed to be subdivided. If during a later refinement step, one of the child elements of poor quality is targeted for refinement, then the element and its siblings must be coarsened back to their parent. But the problem is that some of the siblings have already been refined and, as mentioned in Section 2, elements must be coarsened in an order that is reversed from the one by which they were created. One way to overcome this is to mark all the children of an anisotropically-subdivided element to be of poor quality if even one child element is of poor quality.

A more serious problem with this algorithm is that the effect of coarsening a child element of poor quality and then refining the parent element isotropically can sometimes propagate throughout the refined region of the mesh. This is because when the parent is refined 1:8, all the neighboring elements that share an edge with this parent element may have to be upgraded. This could potentially lead to disconnecting all the 1:4 and 1:2 elements, isotropically refining the parents of all the poor-quality child elements that have at least one edge marked for refinement, and then connecting up the hanging vertices to create a valid mesh.

All this wasted CPU time could have been saved by never subdividing the 1:4 and the 1:2 elements in the first place. In fact, this is the strategy that we have implemented as a user-selectable option in the 3D_TAG code and is used to control the quality of the meshes for the calculations in this paper. In the context of the first strategy, what this basically means is that all children of anisotropically-subdivided elements are always considered to be of poor quality. No tests are required to determine if a child element is good or bad. To further simplify the algorithm, all buffer elements, irrespective of whether they are marked for refinement or not, revert back to their parent before a refinement step. Edges are then marked for refinement and subsequent 1:8, 1:4, or 1:2 subdivision proceeds as usual. The result of this policy is that some 1:2 and 1:4 elements may have poor quality. This cannot be avoided because buffer elements must exist between regions of the mesh that are at two different refinement levels. However, this is acceptable as long as these poor-quality elements are not further subdivided. With our strategy, none of the buffer elements are ever subdivided. This effectively provides an upper bound on element face angles and controls the growth of the maximum vertex degree.

A two-dimensional example of how our algorithm for controlling the mesh quality works is shown in Fig. 5. Assume that initially there are two triangles and that all three edges of the lower triangle are marked for refinement as shown in Fig. 5(a). After the refinement step is performed and the buffer elements are connected up, the configuration in Fig. 5(b) is obtained. If the edge marked with an R is now refined with our mesh quality logic turned off, the mesh looks as shown in Fig. 5(c). The mesh has obviously deteriorated in quality. If, instead, our mesh quality strategy is used, the buffer elements are first disconnected (cf. Fig. 5(d)), the parent element is then refined isotropically (cf. Fig. 5(e)), and finally the requested refinement is performed (cf. Fig. 5(f)). The mesh does not deteriorate in quality beyond the first level of refinement and an upper bound on face angles is achieved.

Implementation of our mesh quality algorithm does not significantly increase the time required for the refinement portion of our mesh adaption code; however, application of this option limits the refinement procedure to isotropic subdivision. Any possibility for increased efficiency with the anisotropic strategy is lost if the mesh quality logic is in-

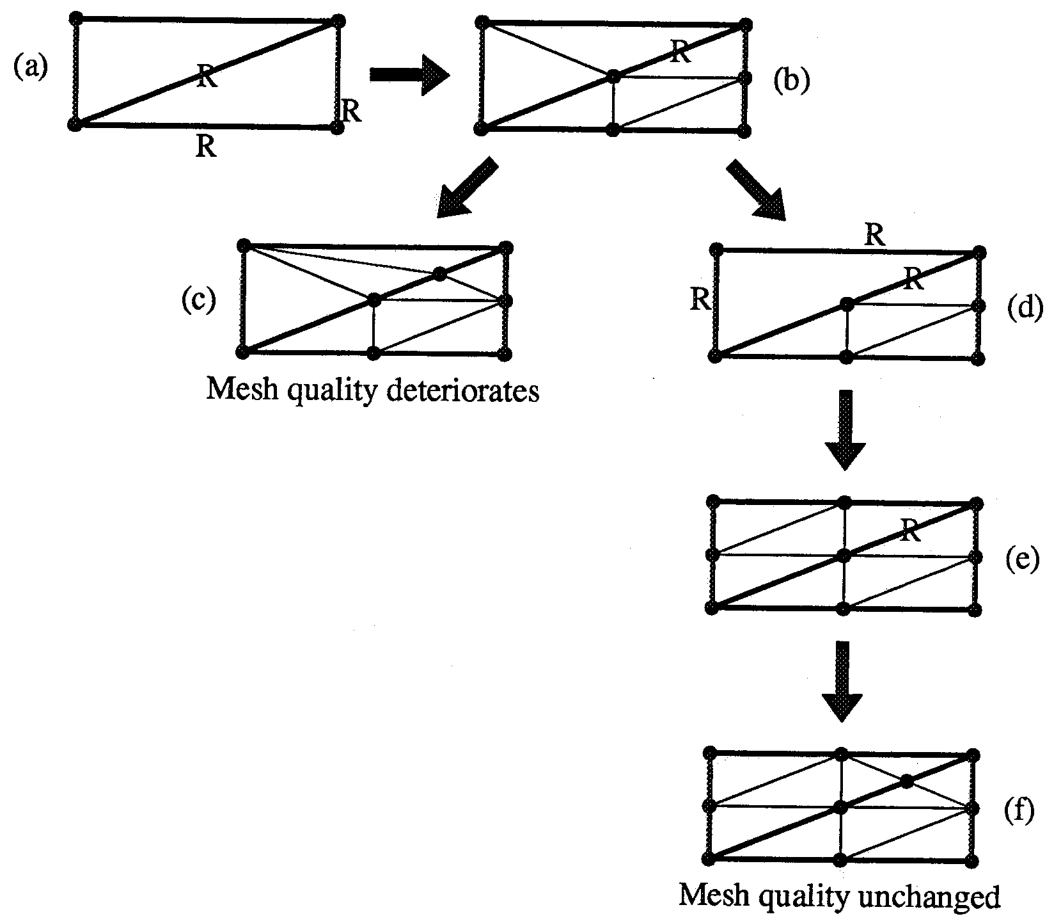


Fig. 5. A two-dimensional schematic that highlights the beneficial effects of our mesh quality algorithm.

voked. However, for real fluid dynamics problems on tetrahedral meshes, truly anisotropic subdivision is almost impossible to realize.

4. Computational Results

The effects of the mesh quality logic are analyzed on a simplified test problem. The initial mesh, illustrated in Fig. 6, is $11 \times 11 \times 6$ with some stretching in the vertical direction. A geometric error indicator targets every element up to the second layer for uniform refinement. This subdivision strategy is a demanding test for grid quality since 1:4 and 1:2 buffer elements will be continuously targeted for further refinement.

Fig. 7 shows two-dimensional sections of the resulting meshes after four levels of refinement. The grid in Fig. 7(a) did not employ the mesh quality logic described in Section 3, while the grid in Fig. 7(b) did. The two meshes have significant differences that can be explained with the help of Table 1. The largest face angle for the grid without mesh quality continues to increase with each refinement. However, when the mesh quality

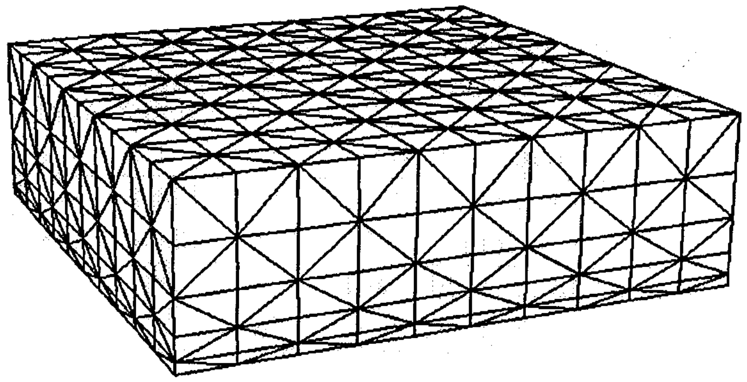


Fig. 6. Initial mesh for the first test case.

logic is used, the largest face angle is bounded after two refinement levels. The actual upper bound of the face angle depends on the properties of the initial mesh. Similarly, the maximum vertex degree for the grid without mesh quality increases exponentially whereas the vertex degree with mesh quality increases only slightly with each refinement. High vertex degrees result in flow solver inaccuracies and poor efficiency on vector and parallel computers.

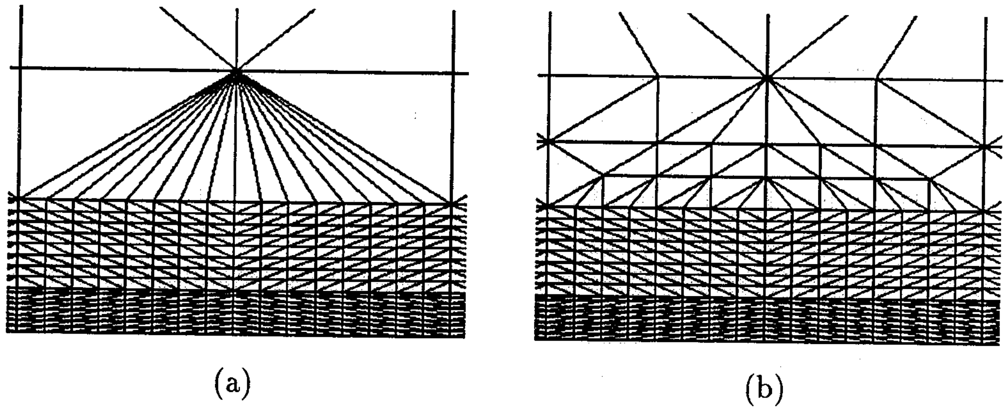


Fig. 7. Final mesh after four levels of refinement (a) without mesh quality, and (b) with mesh quality.

In addition to the improvements reported in Table 1, the mesh quality logic has had two other beneficial effects on the mesh. First, the mesh in Fig. 7(a) shows a large disparity in the sizes of adjacent elements. A much smoother transition, as shown in Fig. 7(b), is obtained using the mesh quality logic. Finally, in addition to a lower overall maximum angle, the mesh in Fig. 7(b) has a smaller percentage (0.086%) of elements with large angles (> 130°) than its counterpart (0.175%) in Fig. 7(a).

The 3D_TAG code has been recently used to model the high-speed impulsive noise for a non-lifting rectangular helicopter rotor blade in hover [8]. It simulates the model-rotor acoustics experiment of Purcell [5] where a 1/7th scale model of a UH-1H rotor blade

Table 1
Largest face angle and maximum vertex degree versus the number of refinement levels

		Number of refinement levels				
	Adaption strategy	0	1	2	3	4
Max. face angle	Mesh quality OFF	90.00	129.81	141.34	145.56	147.38
	Mesh quality ON	90.00	129.81	139.68	139.68	139.68
Max. vertex degree	Mesh quality OFF	18	26	54	158	558
	Mesh quality ON	18	26	24	26	34

was tested over a range of hover-tip Mach numbers, M_{tip}, from 0.85 to 0.95. These rotor speeds produce strong aerodynamic shocks at the blade tips and an annoying pattern of high-speed impulsive noise. Since the model-rotor blades were untwisted and run in a non-lifting configuration, the problem was symmetric about the plane of the rotor and eliminated the need to model the wake system. Numerical results in this paper will only be presented for the high-speed hovering rotor case ($M_{\text{tip}} = 0.95$).

The computational mesh only needs to cover the upper half plane because of the symmetry condition of the non-lifting configuration. A conventional C-H structured grid extending out to two rotor radii in the spanwise direction is first generated. Grid points off the blade tip are concentrated along the expected path of the acoustic wave. An unstructured grid is then created by splitting each hexahedron of this structured grid into five tetrahedra. The resulting tetrahedral grid serves as a base mesh for the initial solution and subsequent local mesh adaptions. A view of the initial tetrahedral mesh boundaries for the $M_{\text{tip}} = 0.95$ case is shown in Fig. 8. The rotor blade has an aspect ratio of 13.71. The initial mesh has 13,967 vertices, 60,986 tetrahedra, and 6,818 triangular boundary faces and extends 1.5 rotor radii above the plane of the rotor.

The rotary-wing version [7] of the basic Euler solver [1] was used in all calculations. It is an explicit finite-volume upwind scheme that solves for the unknowns 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. The Euler equations are written in an inertial reference frame so that the rotor blade and grid move through stationary air at the specified rotational and translational speeds. Fluxes across each computational control volume are computed using the relative velocities between the moving grid and the stationary far field.

An initial solution is first obtained on the grid shown in Fig. 8. This required 1,000 time steps and approximately 20 CPU minutes on a Cray C-90 computer. An error indicator based on acoustic pressure is then used to target regions for adaption. A detailed description of our error indicator is given in [8]. A total of three refinement levels and two coarsening steps are performed with the mesh quality option turned on. The flow solver

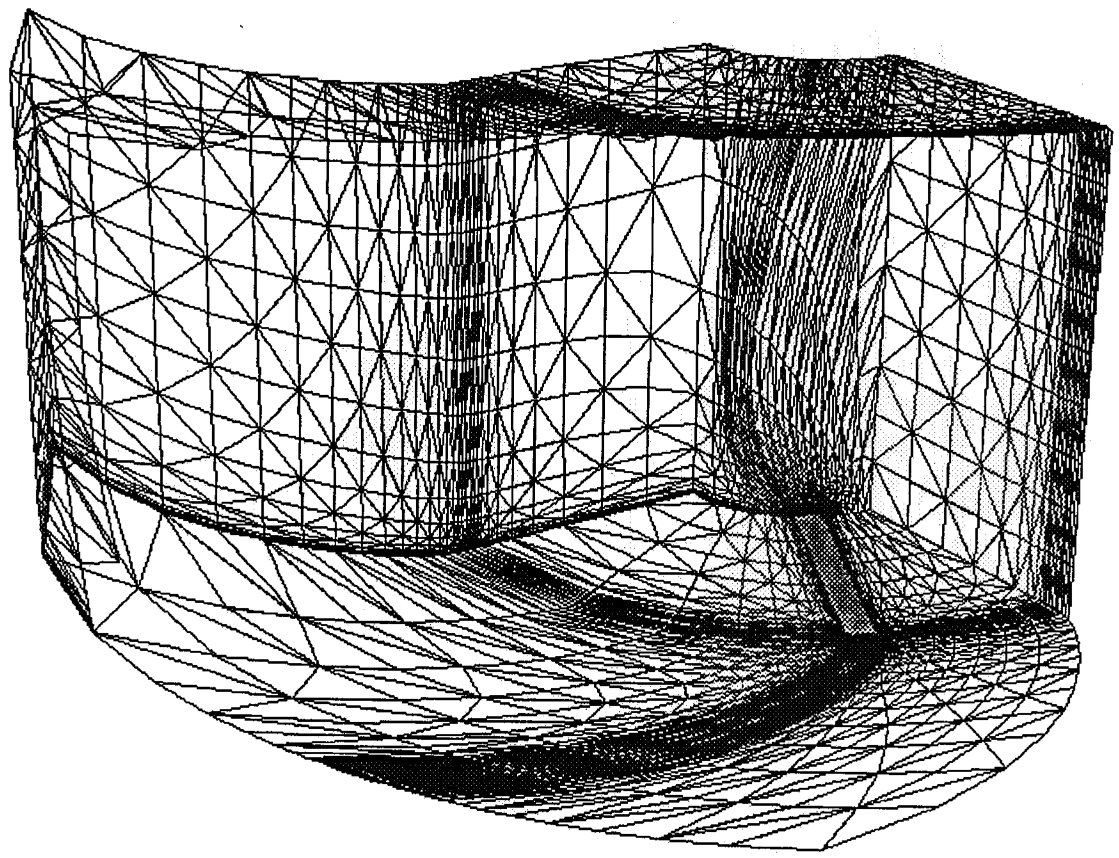


Fig. 8. Boundaries of the initial tetrahedral mesh for $M_{\text{tip}} = 0.95$.

is run for 500 time steps between mesh adaption stages. A close-up of the final adapted grid and pressure contours in the plane of the rotor are shown in Fig. 9. The mesh has been refined to adequately resolve the leading edge compression and capture both the surface shock and the resulting acoustic wave that propagates to the far field. The final mesh contains 140,419 vertices, 783,164 tetrahedra, and 20,546 triangular boundary faces. The third refinement stage almost doubled the mesh size from 483,212 edges to 933,855 edges and required about three times the CPU time needed for one iteration of the Euler flow solver on the final mesh on a Cray C-90. This is acceptable considering that the adaption procedure does not vectorize and that the mesh was essentially doubled in size.

Computed results show excellent agreement with experimental microphone data for far-field acoustic pressures. The result at 1.053 rotor radii in Fig. 10(a) shows good agreement between computation and experiment for the general wave shape and duration of the acoustic signal. The peak negative pressure is somewhat overpredicted, but the impulsive shock is well captured by the computation. The result at the 1.78 radial location in Fig. 10(b) shows that the computation agrees extremely well with experiment except for a small region near the beginning of the acoustic signal. This discrepancy is probably due to a lack of grid resolution in this region. The error indicator that we

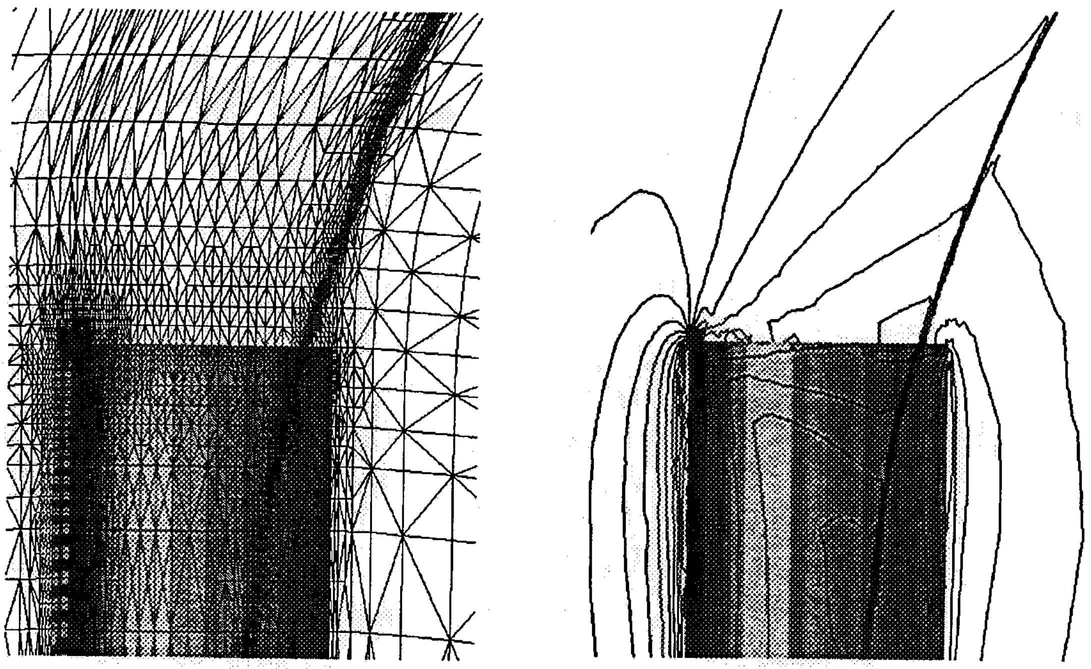


Fig. 9. Final mesh and computed pressure contours for $M_{\text{tip}} = 0.95$.

have used is based on pressure differences across edges. This strategy typically refines the edges that are in the middle of the acoustic wave, but not so much at the beginning or at the end. An error indicator based on second derivatives of pressure might be more appropriate for these regions.

This helicopter acoustics problem was also run without mesh quality in the 3D_TAG code. The mesh shows a slightly higher maximum vertex degree and largest face angle, but these differences have no significant effects on either the flow solver efficiency or computed solution. Clearly, the mesh quality logic has a much smaller effect here than for the earlier test case. This is probably because the first test case presented a worst-case mesh refinement scenario that does not usually occur in practical problems. Typical refinement strategies for steady problems target smaller geometrical regions at each refinement step. This means that the anisotropically-refined elements are rarely targeted for further refinement. As long as this is the case, the mesh quality logic is not required. This is not true, however, for mesh refinement in unsteady flows.

5. Summary and Conclusions

Dynamic mesh adaption in three dimensions provides a powerful framework for computing steady and unsteady problems that require localized mesh refinement. However, one of the problems with repeated anisotropic subdivision on tetrahedral meshes is that it can lead to poor mesh quality. A grid quality algorithm has been presented in this paper that ensures the mesh does not deteriorate after several levels of refinement; however,

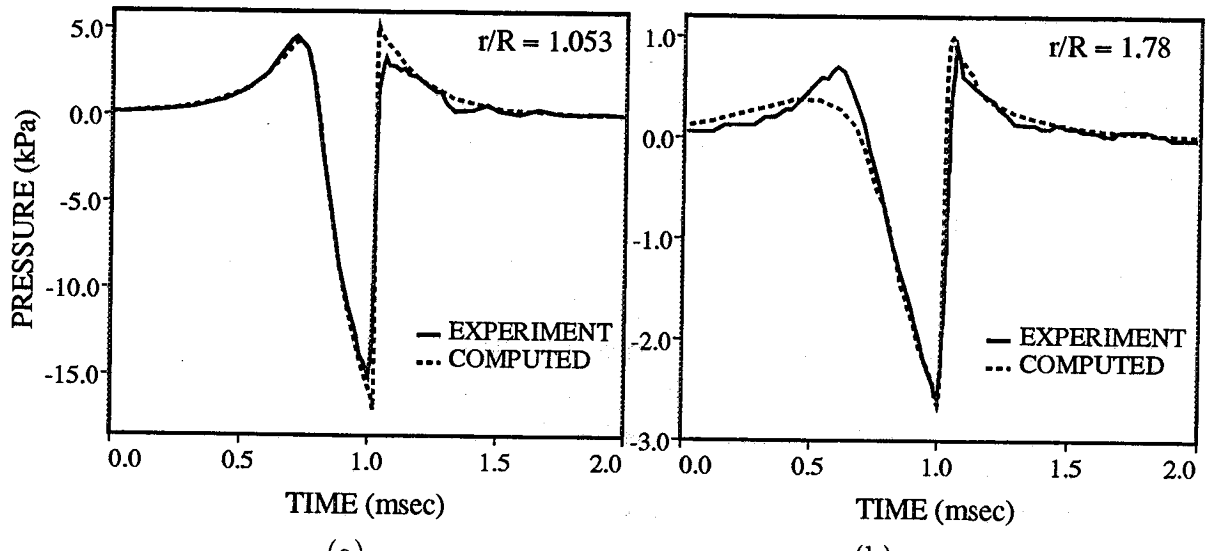


Fig. 10. Computed and experimental acoustic pressures at (a) 1.053 and (b) 1.78 rotor radii in the rotor plane for $M_{\text{tip}} = 0.95$.

exercising this option generally results in an isotropically-refined mesh. The algorithm assumes that the initial mesh has acceptable quality, and that as long as the tetrahedral elements are subdivided isotropically, subsequent refined meshes will retain this quality. This is ensured by never refining a tetrahedron that was previously subdivided anisotropically. If such a tetrahedron needs to be refined, it is first coarsened, along with its siblings, back to its parent. The parent is then refined isotropically and its children can be subdivided as required. Experiments conducted on a test problem presented in Section 4 demonstrate that the mesh quality algorithm significantly helps to prevent deterioration of the mesh. Results are also presented for a practical problem in helicopter acoustics that show this method can be applied efficiently to large problems in computational fluid dynamics. Computer resource requirements for the dynamic mesh adaption scheme in terms of CPU time and in-core memory are extremely reasonable and compare favorably with those for the Euler flow solver.

The disadvantage of using the mesh quality strategy is that meshes that are isotropically refined do not use grid points as efficiently as anisotropic meshes when directional flow features are present. Although it is not required for all steady-state problems, this guarantee of a high quality mesh is important, particularly for unsteady problems with multiple levels of refinement. One important aspect to consider is that true anisotropic refinement on tetrahedral elements is almost impossible to realize for real problems in computational fluid dynamics. A remedy for this drawback is to use hexahedral meshes which can be subdivided anisotropically without any element quality problems. Furthermore, the ability to anisotropically refine the mesh makes a tremendous difference in the final problem size when directional flow features are present.

6. References

[1] T. J. Barth, A 3-D upwind Euler solver for unstructured meshes, AIAA-91-1548, *AIAA 10th Computational Fluid Dynamics Conference* (1991).

[2] R. Biswas and R. C. Strawn, A new procedure for dynamic adaption of three-dimensional unstructured grids, *Appl. Numer. Math.* 13 (1994) 437–452.

[3] Y. Kallinderis and P. Vijayan, An adaptive refinement/coarsening scheme for 3-D unstructured meshes, *AIAA J.* 31 (1993) 1440–1447.

[4] R. Löhner and J. D. Baum, Numerical simulation of shock interaction with complex geometry three-dimensional structures using a new adaptive h-refinement scheme on unstructured grids, AIAA-90-0700, *AIAA 28th Aerospace Sciences Meeting* (1990).

[5] T. W. Purcell, CFD and transonic helicopter sound, Paper No. 2, *14th European Rotorcraft Forum* (1988).

[6] R. Rausch, J. Batina, and H. Yang, Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations, AIAA-93-0670, *AIAA 31st Aerospace Sciences Meeting* (1993).

[7] R. C. Strawn and T. J. Barth, A finite-volume Euler solver for computing rotary-wing aerodynamics on unstructured meshes, *AHS J.* 38 (1993) 61–67.

[8] R. C. Strawn, M. Garceau, and R. Biswas, Unstructured adaptive mesh computations of rotorcraft high-speed impulsive noise, AIAA-93-4359, *AIAA 15th Aeroacoustics Conference* (1993).