NASA Technical Memorandum 103966

Three-Dimensional Unstructured Grid Refinement and Optimization Using Edge-Swapping

Amar Gandhi and Timothy Barth, Ames Research Center, Moffett Field, California

March 1993

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

SUMMARY

This paper presents a three-dimensional (3-D) "edge-swapping" method based on local transformations. This method extends Lawson's edge-swapping algorithm into 3-D. The 3-D edge-swapping algorithm is employed for the purpose of refining and optimizing unstructured meshes according to arbitrary mesh-quality measures. Several criteria including Delaunay triangulations are examined. Extensions from two to three dimensions of several known properties of Delaunay triangulations are also discussed.

1 INTRODUCTION

Unstructured grids are becoming popular in computational fluid dynamics (CFD) because of their ability to handle complex geometries with minimal user intervention. In addition, they offer the flexibility of adapting meshes to better resolve flow features and to improve the accuracy of numerical computations. However, there is a need for techniques to automatically refine (adapt) three-dimensional (3-D) unstructured meshes around complex geometries and complex flow features. Experience with flow-solvers also suggests that a certain triangulation of a point set would be preferred over another, owing to considerations of speed and numerical accuracy. To date, much attention has been focused on Delaunay triangulations, principally because it is a well-defined triangulation, and two-dimensional theory indicates that it is optimal in several important measures. However, it may make sense to optimize a 3-D mesh based on other measures, for example, on the tetrahedral face-angle or, perhaps, on the total number of edges and faces in a mesh. These measures have not yet been investigated because there does not exist a general way of refining or optimizing meshes based on arbitrary criteria in order to improve mesh quality.

In this work, a 3-D "edge-swapping" method based on local transformations is used to refine and optimize unstructured meshes according to arbitrary mesh-quality measures. Section 2 presents a review of the Delaunay triangulation, its several characterizations, and Lawson's 2-D edge-swapping algorithm. Extensions from two to three dimensions of several known properties of Delaunay triangulations are also examined. Section 3 outlines our basic algorithm, which extends Lawson's edge-swapping algorithm into 3-D (ref. 1). In section 4, we exploit this idea for the purpose of optimizing meshes and, in section 5, for retriangulating a mesh following insertion of new grid points. The latter idea forms the basis of the mesh-refinement operation. Finally, a brief summary is presented and future work is discussed in section 6, where the time-complexity of the algorithm is also discussed. Though the method we present holds for any arbitrary optimization criteria, the discussion is biased toward Delaunay triangulations.

We thank Marshal Merriam for providing us with the volume triangulations from the surface triangulations. These volume triangulations were then used as the initial triangulations for mesh refinement.

2 DELAUNAY TRIANGULATIONS

The Dirichlet tessellation of a point set is the pattern of convex regions, each being closer to some point P in the point set than to any other point in the set. These Dirichlet regions are also called Voronoi regions. This idea extends naturally to higher dimensions.

DEFINITION: The Delaunay triangulation of a point set is defined as the dual of the Voronoi diagram of the set.

Properties of a 2-D Delaunay Triangulation

Uniqueness. The Delaunay triangulation is unique. This assumes that no four sites are cocircular. The uniqueness follows from the uniqueness of the Dirichlet tessellation.

The circumcircle criterion. A triangulation of $N \geq 2$ sites is Delaunay if and only if the circumcircle of every interior triangle is point-free. If this were not true, the Voronoi regions associated with the dual would not be convex, and the Dirichlet tessellation would be invalid.

Related to the circumcircle criterion is the InCircle test for four points as shown in figure 1.

This test is true if point D lies interior to the circumcircle of $\triangle ABC$, which is equivalent to testing whether $\angle ABC + \angle CDA$ is less than or greater than $\angle BCD + \angle BAD$. More precisely, we have that

$$\angle ABC + \angle CDA \begin{cases} < 180^\circ & \text{InCircle false} \\ = 180^\circ & A, B, C, D \text{ cocircular} \\ > 180^\circ & \text{InCircle true} \end{cases}$$

Since interior angles of the quadrilateral sum to 360°, if the circumcircle of $\triangle ABC$ contains D then swapping the diagonal edge from position $A - C$ into $B - D$ guarantees that the new triangle pair satisfies the circumcircle criterion (hence reversing the outcome of the circumcircle test).

Edge circle property. A triangulation of sites is Delaunay if and only if there exists *some* circle passing through the endpoints of each and every edge that is point-free. This characterization is very useful, because it also provides a mechanism for defining a *constrained* Delaunay triangulation where certain edges are prescribed a priori. A triangulation of sites is a constrained Delaunay triangulation if for each and every edge of the mesh there exists some circle passing through its endpoints containing no other site in the triangulation which is *visible* to the edge. In figure 2, site d is not visible to the segment a-c because of the constrained edge a-b.

Equiangularity property. Delaunay triangulation maximizes the minimum angle of the triangulation. For this reason Delaunay triangulation is often called the MaxMin triangulation.

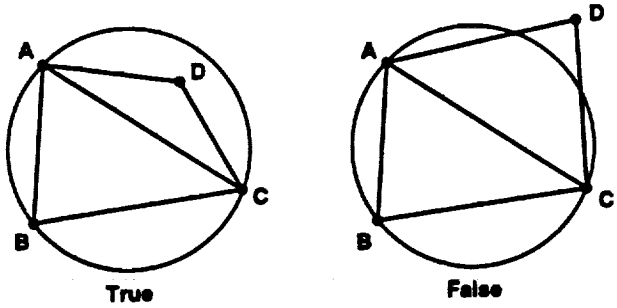

Figure 1. InCircle test for $\triangle ABC$ and D.

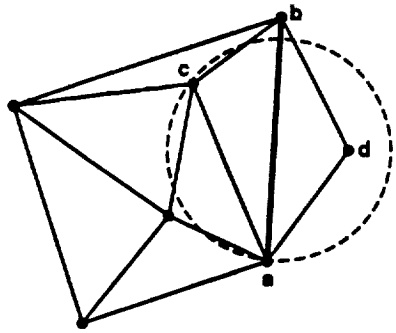

Figure 2. Constrained Delaunay triangulation; site d is not visible to a-c because of constrained segment a-b.

Minimum containment circle. A recent result by Rajan shows that the Delaunay triangulation minimizes the maximum containment circle over the entire triangulation (ref. 2). The containment circle is defined as the smallest circle enclosing the three vertices of a triangle. This is identical to the circumcircle for acute triangles and a circle with diameter equal to the longest side of the triangle for obtuse triangles (see fig. 3). This property extends to k dimensions. Unfortunately, the result does not hold lexicographically.

Nearest neighbor property. An edge formed by joining a vertex to its nearest neighbor is an edge of the Delaunay triangulation. This property makes Delaunay triangulation a powerful tool in solving the closest proximity problem. Note that the nearest neighbor edges do not describe all edges of the Delaunay triangulation, that is, nearest neighbor edges are a subset of the Delaunay triangulation.

Minimal roughness. The Delaunay triangulation is a minimal roughness triangulation for arbitrary sets of scattered data (ref. 3). Given arbitrary data f_i at all vertices of the mesh and a triangulation of these points, a unique piecewise linear interpolating surface can be constructed. The Delaunay triangulation has the property that of all triangulations it minimizes the roughness of this surface as measured by the following Sobolev seminorm:

$$\int_T \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right] dx\, dy$$

This is a interesting result, for it does not depend on the actual form of the data. This also indicates that Delaunay triangulation approximates well those functions that minimize this Sobolev norm. One example

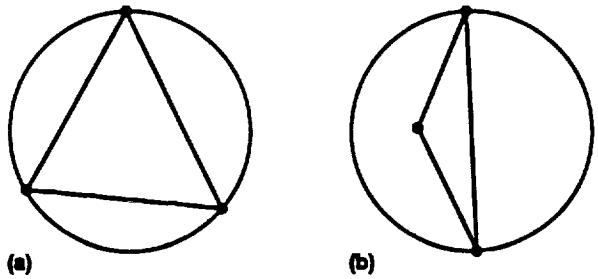

Figure 3. Containment circles.

would be the harmonic functions satisfying Laplace's equation with suitable boundary conditions which minimize exactly this norm. It is also true that a Delaunay triangulation guarantees a maximum principle for the discrete Laplacian approximation (with linear elements).

There exist many algorithms for obtaining the Delaunay triangulation of a point-set in 2-D. The main ones are (1) incremental insertion, (2) divide and conquer, (3) Tanemura/ Merriam (uses advancing fronts), and (4) edge-swapping. Next, we describe the edge-swapping algorithm in detail, for our 3-D work is an extension of the 2-D algorithm.

Edge-Swapping Algorithm in 2-D

Figure 4 shows the three possible non-degenerate arrangements of four points in 2-D. The configurations in cases 1 and 3 have a unique triangulation whereas the configuration in case 2 has two. The 2-D edge-swapping algorithm is based on flipping between the two ways of triangulating the configuration in case 2.

The success of edge-swapping in 2-D hinges upon the following two properties:

1. Global versus local optimization: Lawson observed the relation between local and global properties of the Delaunay triangulation: a triangulation is Delaunay if and only if all adjacent triangle pairs satisfy the InCircle test. This holds for the equiangularity property also; that is, a Delaunay triangulation is the unique triangulation which maximizes the minimum angle locally for all adjacent triangle pairs. Unfortunately, minimizing the containment circle locally for all adjacent triangle pairs does not necessarily yield the Delaunay triangulation.

2. Concavity implies Delaunayhood: If two triangles form a concave shape, then these two triangles also satisfy the circumcircle criteria (see fig. 5). Two triangles can be constructed by joining point D to any two points in triangle ABC. For these two triangles to form a concave shape, point D must lie in the shaded portion of the figure, which shares no points with the interior of the circumcircle of triangle ABC. Hence, concave triangle pairs always satisfy the circumcircle criterion.

The 2-D edge-swapping algorithm (see Algorithm 1) due to Lawson (ref. 4) assumes that a triangulation exists (not Delaunay), then makes it Delaunay through application of edge-swapping such

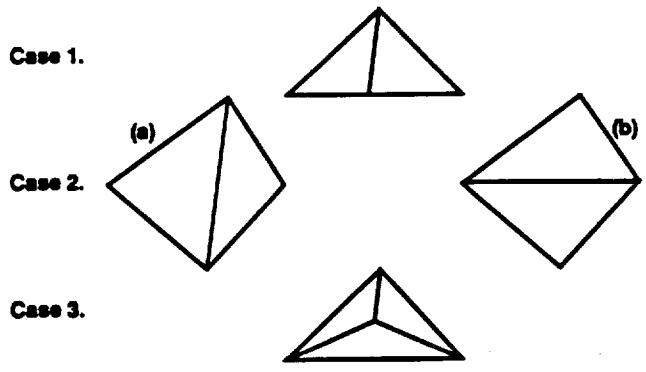

Figure 4. Generic arrangements of four points and possible triangulations of their convex hulls.

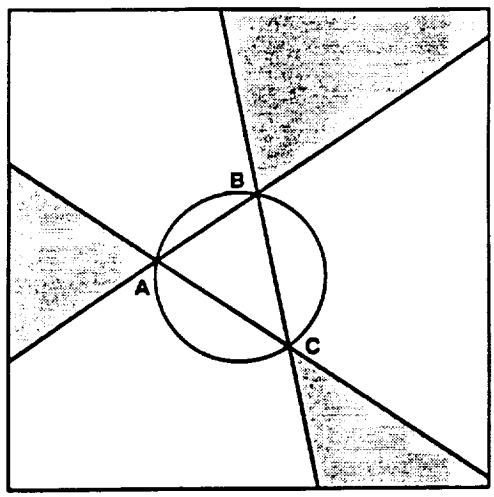

Figure 5. Shaded regions indicate where point D will have to lie to form concave shape with triangle ABC. There is no intersection between shaded regions and circle circumscribing triangle ABC; hence, two triangles that form a concave shape in 2-D are always Delaunay triangulated.

that the equiangularity of the triangulation increases. The equiangularity of a triangulation, $A(T)$, is defined as the ordering of angles $A(T) = [\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{3n(c)_3}]$ such that $\alpha_i \leq \alpha_j$ if $i < j$. We write $A(T^*) < A(T)$ if $\alpha_j^* \leq \alpha_j$ and $\alpha_i^* = \alpha_i$ for $1 \leq i < j$. A triangulation T is globally equiangular if $A(T^*) \leq A(T)$ for all triangulations T^* of the point set. Lawson's algorithm examines all interior edges of the mesh. Each of these edges represents the diagonal of the quadrilateral formed by the union of the two adjacent triangles. In general, one must first check if the quadrilateral is convex, so that a potential diagonal swapping can take place without edge crossing. If the quadrilateral is convex, then the diagonal position is chosen which optimizes a local mesh-quality criterion (in this case the local equiangularity). This amounts to maximizing the minimum angle of the two adjacent triangles. Furthermore, this process of diagonal swapping is local, that is, it does not disrupt the Delaunayhood of any triangles adjacent to the quadrilateral, because any triangle whose circumcircle is point-free will not acquire a point inside its circumcenter as a result of edge-swapping. If an edge satisfies the edge-circle property, then it is part of the final Delaunay triangulation, and edge-swapping will not affect this edge. Lawson's algorithm continues until the mesh is locally optimized and is locally equiangular everywhere. It is easily shown that the condition of local equiangularity is equivalent to satisfaction of the circumcircle criteria described earlier. Therefore, a mesh that is locally equiangular everywhere is a Delaunay triangulation. Note that each new edge-swapping (triangulation T^*) insures that the global equiangularity increases $A(T^*) > A(T)$. Because the triangulation is of finite size this guarantees that the process will terminate in a finite number of steps.

In summary, if two triangles that are not Delaunay triangulated form a convex shape, an edge-swap is always possible to bring them to a Delaunay triangulation. If two triangles form a concave shape, they always have a valid Delaunay triangulation. Hence, termination is a guarantee that the Delaunay triangulation has been achieved. The monotonic increase of the equiangularity measure and the finite-dimensional nature of the triangulation ensure that termination will indeed occur in a finite number of steps. The mesh is a Delaunay triangulation when no more swaps are possible and the algorithm terminates.

Algorithm: Triangulation via Lawson's Algorithm
  While (*swaps occurred in the last cycle over interior edges*)
    For (*all interior edges*)
      If (*triangles adjacent to the edge form convex quad*) then
        Let T = *current triangulation*
        Let T^* = *alternate triangulation formed by swapping diagonal*
        If (*Quality* (T^*) > *Quality* (T)) then
          [*Edge swap* T *into* T^*]
        EndIf
      EndIf
    EndFor
  EndWhile

Algorithm 1. Lawson's 2-D edge-swapping algorithm.

Since the unconstrained Delaunay triangulation is a triangulation of the convex hull of a set of points, edge-swapping will only produce the Delaunay triangulation if the initial triangulation was an arbitrary triangulation of the convex hull.

Note that measures other than the circumcircle or equiangularity criteria can also be employed to optimize meshes. One such triangulation is the local MinMax triangulation which considers convex triangle pairs and chooses a diagonal position that minimizes the maximum interior angle. This triangulation does not possess the properties that are essential for guaranteeing that edge-swapping always produces the unique Delaunay triangulation, namely, local optimization ensures global optimization, and concavity implies Delaunayhood. For this reason, the edge-swapping procedure gets stuck in local optima and fails to reach the globally optimum MinMax triangulation. As we will see, this unfortunate fact is true of 3-D Delaunay triangulations as well.

This concludes the discussion of the 2-D edge-swapping algorithm. Before moving on to the discussion of edge-swapping algorithms in 3-D, some properties of 2-D Delaunay triangulations which extend into 3-D and higher dimensions are examined. Following that, the properties of 2-D Delaunay triangulations that do not extend into 3-D are listed.

Properties of 2-D Delaunay Triangulations That Generalize to 3-D

The properties of 2-D Delaunay triangulations that generalize to 3-D are as follows.

1. Voronoi dual: The Delaunay triangulation extends naturally into three dimensions as the geometric dual of the 3-D Voronoi diagram.

2. Circumsphere criterion: The Delaunay triangulation in 3-D can be characterized as the unique triangulation such that the circumsphere passing through the four vertices of any tetrahedron must not contain any other point in the triangulation.

3. Minimum containment sphere: As in the 2-D case, the 3-D Delaunay triangulation has the property that it minimizes the maximum containment sphere (globally but not locally).

4. Self-centered simplex property: In two dimensions, it can be shown that a mesh entirely composed of acute triangles is automatically Delaunay. To prove this, consider an adjacent triangle pair forming a quadrilateral. By swapping the position of the diagonal it is easily shown that the minimum angle always increases. Rajan (ref. 2) shows the natural extension of this idea to three or more space dimensions (ref. 2). He defines a "self-centered" simplex in $\mathbf{R}^d$ to be a simplex that has the circumcenter of its circumsphere interior to the simplex. In two dimensions, acute triangles are self-centered and obtuse triangles are not. Rajan shows that a triangulation entirely composed of self-centered simplices in $\mathbf{R}^d$ is automatically Delaunay.

5. Global versus local optimization: A triangulation is Delaunay if and only if all adjacent tetrahedra pairs satisfy the circumsphere criteria. This relates the local and global nature of the Delaunay triangulation. There does not seem to be an extension of the equiangular property for Delaunay triangulation to three dimensions.

6. Stereographic projection: The k-dimensional Delaunay triangulation is the stereographic projection of a $(k+1)$-dimensional polyhedra into k-space.

7. $(k+1)$-dimensional convex hulls: Lift the k-dimensional input points onto the $(k+1)$-dimensional paraboloid, that is, in 3-D (x, y, z) is mapped to $(x, y, z, x^2 + y^2 + z^2)$. The convex hull of the $(k+1)$-dimensional points is constructed. The orthographic projection of the lower convex hull onto k-dimensional space is the Delaunay triangulation in k dimensions. The lower convex hull is that part of the convex hull visible from k-dimensional space, that is, in 3-D this would be the part of four-dimensional convex polyhedron visible from $(x, y, z, 0)$ (ref. 5).

Properties of 2-D Delaunay Triangulations That Do Not Generalize to 3-D

The properties of the 2-D Delaunay triangulations that do not generalize to 3-D are as follows.

1. Non-constant number of edges, faces, and tetrahedra during 3-D edge-swapping: The edge-swapping algorithm in 3-D does not preserve the number of edges, faces or tetrahedra. This is clear by observing the two possible triangulations shown in figures 6(a) and 6(b) for the same configuration of five points.

2. 3-D Delaunay triangulation is not a MaxMin triangulation: The Delaunay triangulation does not have a characterization in terms of either MinMax or MaxMin of face-angles or edge-angles. This can be proved by counter example.

3. Concavity does not imply 3-D Delaunayhood: In 2-D, if the triangulation of four points formed a concave shape, the triangulation was guaranteed to be Delaunay. This result does not extend in 3-D. It is possible to have five points triangulated in such a way that their shape is concave and they are not a valid Delaunay triangulation.

Because these properties do not extend to three dimensions, the extension of edge-swapping to 3-D is not straightforward. The circumsphere criterion has no simple characterization in terms of angles, etc. There is no known metric for which a monotonic increase or decrease can be shown in order to guarantee termination of the edge-swapping process. In addition, the fact that the number of edges, faces, and cells is not constant makes any lexicographic ordering of these quantities undefined. The fact that concavity of a shape does not guarantee Delaunayhood means that mere termination of the edge-swapping algorithm does not guarantee that the resulting mesh is Delaunay. These factors conspire to prevent the extension of global edge swapping to 3-D.

3 3-D EDGE-SWAPPING ALGORITHMS

The algorithms of Bowyer (ref. 6) and Watson (ref. 7) extend naturally to three dimensions with estimated complexities of $O(N^{5/3})$ and $O(N^{4/3})$ for N randomly distributed vertices. They do not give worst-case estimates. It should be noted that in three dimensions, Klee shows that the maximum number of tetrahedra that can be generated from N vertices is $O(N^2)$ (ref. 8). Thus, an optimal worst-case complexity would be at least $O(N^2)$. Under normal conditions this worst-case scenario is rarely encountered. Baker reports more realistic actual run times for Watson's algorithm (refs. 9 and 10).

Until most recently, the algorithm of Green and Sibson based on edge-swapping was thought not to be extendable to three dimensions because it was unclear how to generalize the concept of edge-swapping to three or more dimensions. In 1986, Lawson published a paper in which he proved the fundamental combinatorial result (ref. 4).

THEOREM. The convex hull of $d+2$ points in $\mathbf{R}^d$ can be triangulated in at most two ways.

Joe (refs. 11 and 12) and Rajan (ref. 2) have constructed algorithms based on this theorem for Delaunay triangulations. Independently, we have devised an edge-swapping method based on Lawson's theorem to optimize and refine unstructured meshes. The remainder of this section will review our algorithm and the basic ideas behind 3-D edge-swapping algorithms.

It is useful to develop a taxonomy of possible configurations addressed by Lawson's theorem. Figure 7 shows configurations in 3-D of five points which can be triangulated uniquely, and hence no change is possible. We call these arrangements "unswappable." Figure 6 shows configurations that allow two ways of triangulation. It is possible to flip between the two possible triangulations, and we call these arrangements "swappable."

There are two arrangements that allow two triangulations. Figures 7(a) and 7(b) show the subclass of companion triangulations that can be transformed from one type to another, thereby changing the number of tetrahedra from two to three or vice versa. Figures 6(c) and 6(d) show the other subclass of configurations that can be transformed from one type to another while keeping constant the number of tetrahedra (i.e., 2). These figures reveal an important difference between the two- and three-dimensional algorithms. *The number of tetrahedra involved in the swapping operation need not be constant.*

The 3-D edge-swapping algorithm is based on flipping between the two ways of triangulating the configurations shown in figure 6. One good way of finding all sets of five points in the mesh

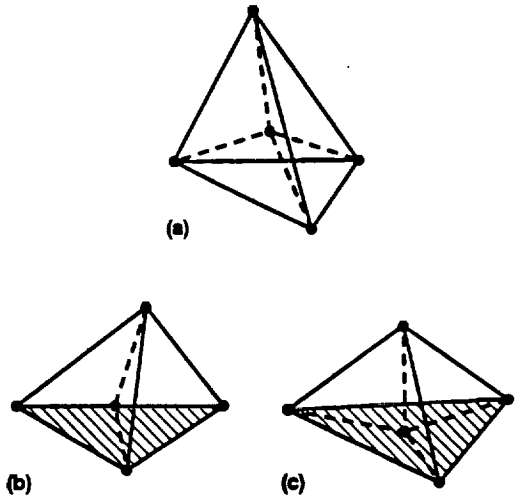

Figure 6. Generic swappable configurations of five points; shaded regions denote planar surfaces.

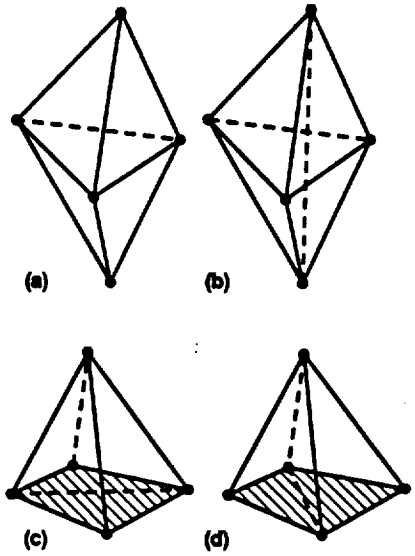

Figure 7. Generic nonswappable configurations of five points; shaded regions denote planar surfaces.

is to loop through all the faces in the mesh and then consider the five points that make up the two adjoining tetrahedra for that face. In this sense, faces are an extension of edges to 3-D. The face-wise edge-swapping primitive is presented in Algorithm 2.

The first step is to find *all* the tetrahedra that are described by the five nodes of the two tetrahedra adjacent to the face in question. A maximum of four tetrahedra can be built from five nodes. Since any two tetrahedra made from the same five points will have to share three points (i.e., a face) it is sufficient to look only at the four neighboring tetrahedra of any of the two tetrahedra adjacent to the face in question. This constitutes a linear time-algorithm for finding all the non-overlapping tetrahedra made from the five points. This operation was not needed in 2-D, because the swappable configurations in 2-D have only two triangles, and these are found by considering the triangles adjacent to all the interior edges.

Primitive: EDGE_SWAP(face)
Let C = { *Set of tetrahedra made from the 5 nodes of the two adjoining tetrahedra* }
If (*shape(C) = convex*) then
Let T = *current triangulation*
Let T^* = *alternate triangulation (if it exists)*
If (*Quality*(T^*) > *Quality*(T))then
[*Edge Swap T into T^**]
EndIf
EndIf

Algorithm 2. 3-D edge-swapping primitive.

If these tetrahedra form a convex shape, then the configuration is described by one of the configurations in figures 6 and 7 (configurations of the convex hull), and edge-swapping is permitted if there exists an alternative triangulation for that configuration. If only two of the three tetrahedra in figure 6(b) were present, the two tetrahedra would form a concave shape. Obviously, edge-swapping of concave shapes cannot be done without possibly creating overlapping tetrahedra in the mesh. For swappable configurations, a check is performed to see if the local mesh quality measure (discussed further below) will improve by edge-swapping into the alternative triangulation. If it does, the swap is performed; otherwise, the triangulation is left unchanged. Some of the computational details of the 3-D edge-swapping algorithm are discussed in appendix A.

4 3-D MESH OPTIMIZATION

The 3-D edge-swapping algorithm can be used to optimize existing triangulations (see Algorithm 3). In fact, there is no known way to triangulate a given set of points based on the MinMax or MaxMin of the face angles directly. An alternative is to start with an existing triangulation and optimize it. This requires that we cycle through all the faces in a mesh and apply the edge-swapping procedure at each step. This process is continued until no more swaps are possible.

Algorithm: Three-dimensional mesh optimization.
While (*swaps occurred in the last cycle over interior faces*)
For (*all interior faces*)
EDGE_SWAP (*face*)
EndFor
EndWhile

Algorithm 3. 3-D mesh optmization using edge-swapping.

In the following subsections, a few swap criteria and the meshes they produce are examined.

3-D Delaunay Triangulations

The InSphere criterion is binary in the sense that either the triangulation of a set of five points satisfies the criterion or it does not. It can also be shown that if one of the two ways to triangulate a set of five points (if there exist two ways) fails the InSphere criterion, then the other one will pass and vice versa. Cases (a), (b), and (c) in figure 6 will always pass the InSphere criterion.

Joe has shown, however, that processing faces in an arbitrary way may result in getting stuck in local optima (ref. 11). This is an important difference between two- and three-dimensional combinatorial edge-swapping.

The Delaunay triangulation of a set of points is bounded by their convex hull and, since edge-swapping simply retriangulates a configuration, only triangulations that also include the convex hull will be considered.

In figure 8, it can be seen that the number of edge-swaps taking place in order to convert an arbitrary triangulation into the Delaunay triangulation goes down with each cycle. The convergence appears to be independent of the mesh-size and appears to be exponential in nature. The total number of edge-swaps naturally depends on how far the mesh is from being optimum.

Next, an algorithm is presented that, we conjecture, yields *globally* Delaunay meshes in an order-independent method. This is more desirable than algorithms presented by Rajan (ref. 2), which require

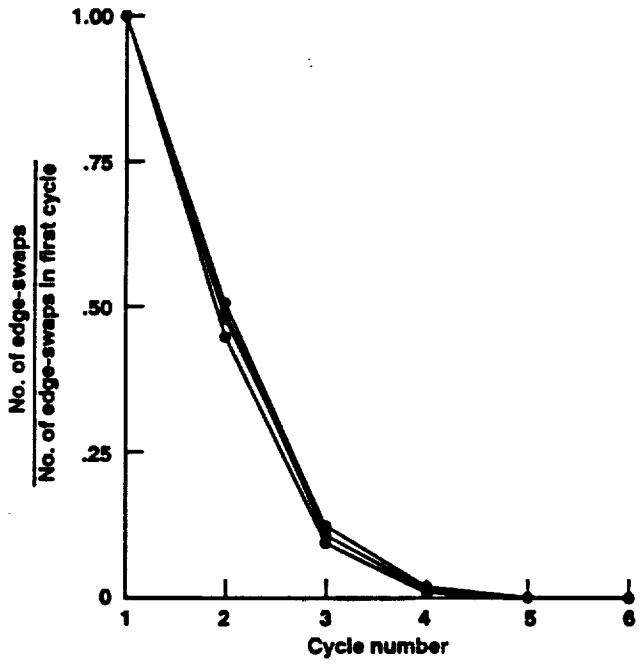

Figure 8. Curves show that number of edge-swaps taking place to convert an arbitrary triangulation into Delaunay triangulation goes down with each cycle. Curves are derived from meshes containing 500, 1,000, 8,000, and 10,000 random points in a unit cube and have been normalized to have the same maximum. Observe that the convergence history does not seem to depend on total size of the mesh.

a certain sequence in which points must be inserted and edge-swaps performed in order to guarantee that a Delaunay triangulation is produced.

The 3-D edge-swapping algorithm gets stuck in local optima (unlike 2-D) because in 3-D it is possible to have a concave set of tetrahedra which are non-Delaunay triangulated. Typically, this does not happen, because one of the tetrahedra that make up the concave shape forms a convex shape with another neighbor, and this convex pair is not Delaunay triangulated; hence, they edge-swap, allowing the process to continue. This does not work when all the tetrahedra that form the concave shape are Delaunay triangulated with all the neighbors with which they form convex shapes. Joe reports an example of triangulations of this kind (ref. 11).

Recognizing that it is edge-swapping with the neighbors that pushes the triangulation out of its local optimum, we propose a scheme where the tetrahedra which form a concave shape and are not Delaunay triangulated will edge-swap with the neighbors with which they form convex shapes, regardless of whether the convex shape is Delaunay triangulated or not. Thus, an edge-swap under the new scheme takes place under two conditions: (1) as before when the tetrahedra that form a convex shape are not Delaunay triangulated, and (2) when one of the tetrahedra that form the convex shape forms a concave shape with some other tetrahedron with which it is not Delaunay triangulated.

This is termed the "helper strategy" because by edge-swapping convex non-Delaunay triangulations, some other configuration is being helped out of a local optimum toward a global one. The result is a substantially higher number of edge-swaps because configurations that are not necessarily non-Delaunay are also being edge-swapped. The edge-swap primitive modified with the helper-strategy idea is presented in Algorithm 4. The algorithm for mesh optimization remains unchanged.

Primitive: EDGE_SWAP_H(face)
  Let C = { *Set of tetrahedra made from the 5 nodes of the two adjoining tetrahedra* }
  If (*shape(C) is not Delaunay*) then
    [*Set tag on all tetrahedra in C*]
  EndIf
  If (*shape(C) = convex*) then
    Let T = *current triangulation*
    Let T^* = *alternate triangulation (if it exists)*
    If (*tag is set on any tetrahedra in C*)
      [*Edge Swap T into* T^*]
    EndIf
  EndIf

Algorithm 4. 3-D edge-swapping primitive using the helper strategy.

The significance of Algorithm 4 is that it seems to successfully retriangulate any arbitrary triangulation into its corresponding Delaunay triangulation without getting stuck in a local optimum. We do not have a proof that this is the case, but we conjecture that it is. It should also be pointed out that this algorithm does not assume any ordering in which the pairs of tetrahedra are to be encountered. Depending on the order of traversal of faces, the algorithm will have a different history, but the claim

is that the final triangulation will always be the same, namely, the Delaunay triangulation. In figure 9, it can be seen that the general nature of convergence is similar to that of edge-swapping without the helper-strategy idea, although with substantially more edge-swaps and more cycles.

Of course, this idea only makes sense for tests that are binary in nature, like the Delaunay InSphere test, which determine if the triangulation is correct or not. It is unclear when to tag the tetrahedra while working to MinMax or MaxMin face angles or when using swap criteria that involve comparisons (and not a simple true/false condition for the triangulation).

3-D MinMax and MaxMin Triangulations

Babuška and Aziz show that from the point of view of finite elements, accuracy demands that no angle in a 2-D mesh be too close to 180° (ref. 13). In other words, triangulations that minimize the maximum angle are desirable. These triangulations are referred to as MinMax triangulations. The edge-swapping algorithm can be applied locally to produce a triangulation that minimizes the maximum face angle. In 2-D, the edge-swapping algorithm (working with edge angles) gets stuck in local minima and, depending on the order in which the edges are traversed, different local minima are reached. In practice, the local minima all seem very close to the global minimum, which makes edge-swapping

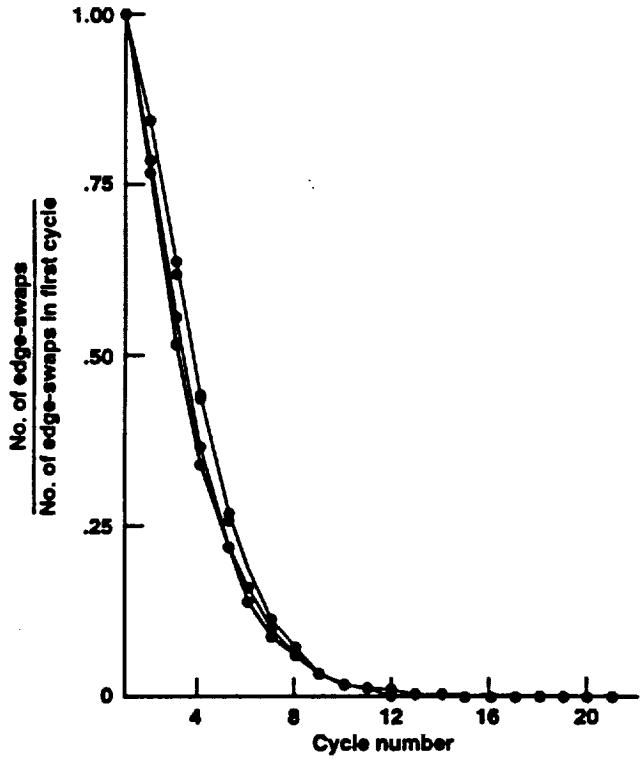

Figure 9. Curves show that number of edge-swaps taking place to convert an arbitrary triangulation into Delaunay triangulation using helper-strategy idea goes down with each cycle. Curves are derived from meshes containing 500, 1,000, 8,000, and 10,000 random points in a unit cube and have been normalized to have the same maximum. Observe that the convergence history does not seem to depend on the total size of the mesh and looks very similar to the one in figure 8.

a practical way to get a nearly optimal MinMax triangulation. We observe that in 3-D as well, there are many local minima, and the order of face traversal determines which one is found. It is hard to determine how far these local minima are from the global minimum, but we believe that edge-swapping is a practical way to get nearly optimal MinMax meshes.

Lawson has shown that in 2-D, Delaunay triangulations have the property that the minimum edge angle is maximized (i.e., MaxMin triangulation). So in 2-D, the MaxMin triangulation is unique, and the edge-swapping algorithm will converge to it. In 3-D, however, the Delaunay triangulation is not the same as in MaxMin triangulation, and the edge-swapping algorithm working with the MaxMin criteria has the same property of getting stuck in local minima as the MinMax. Again, it is hard to judge how close the local minima are from the global minimum, but we still conclude that edge-swapping is a fairly efficient technique for the construction of MaxMin triangulations.

Figure 10 shows the histogram of face-angle distributions before and after mesh optimization using the MinMax swap criteria. The number of face-pairs before and after is not the same and so the distributions have been normalized to yield the same area under the histogram. Observe that the number of very large and very small angles has decreased. This improves the quality of the mesh from the finite-element standpoint, for there are less extremely acute or obtuse face-angles in the mesh. Also the peak in the histogram has shifted from about 30° to about 50° and now there are more angles in the range 40° – 120°. This implies that there are more "well-shaped" elements in the mesh.

Histogram showing improvement in face angle using edge-swapping procedure on triangulation of 500 random sites in unit cube.

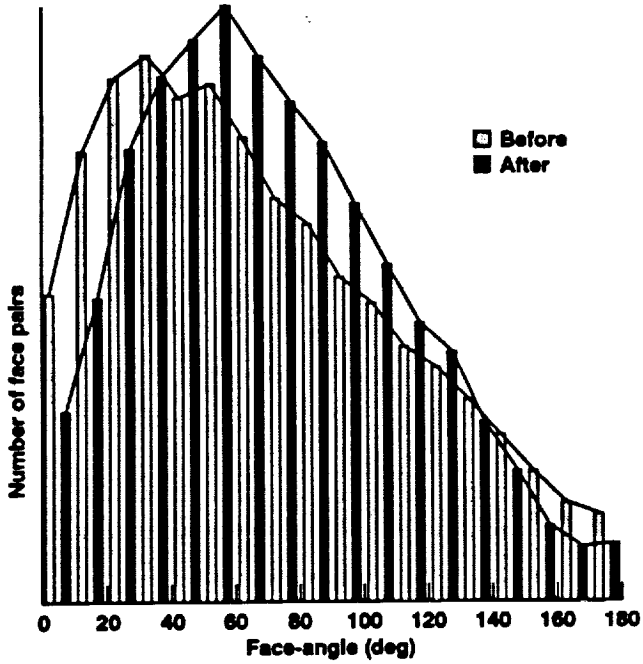

Figure 10. InCircle test for $\triangle ABC$ and D.

Figure 11 plots the convergence history of the optimization procedure based on MinMax and MaxMin swap criteria. Statistically, it was observed that most of the swaps were from the three-tetrahedra configuration in figure 6(b) to the two-tetrahedra configuration in figure 6(c). This is because, statistically, the face-angles are more regular in the two-tetrahedra configuration. This means that the resulting mesh promises "better" answers at lower cost.

3-D Minimum Edge and Face Triangulations

Another mesh of interest is the minimum edge/face triangulation. Since finite-volume flow solvers work edge-wise (in cell-vertex schemes) or face-wise (in cell-centered schemes), it is beneficial to reduce the number of edges and faces in a mesh. This is easily accomplished by edge-swapping such that the configuration in figure 6(a) is always edge-swapped to that in figure 6(b). Each time this operation is performed, one edge, one face, and one tetrahedron are removed from the mesh. Again, different meshes will be produced depending on how faces are traversed, and the final mesh may only be at a local minimum.

The following table illustrates the effect of using the minimum-edge swap criteria on a mesh of 8,000 random sites in a unit cube.

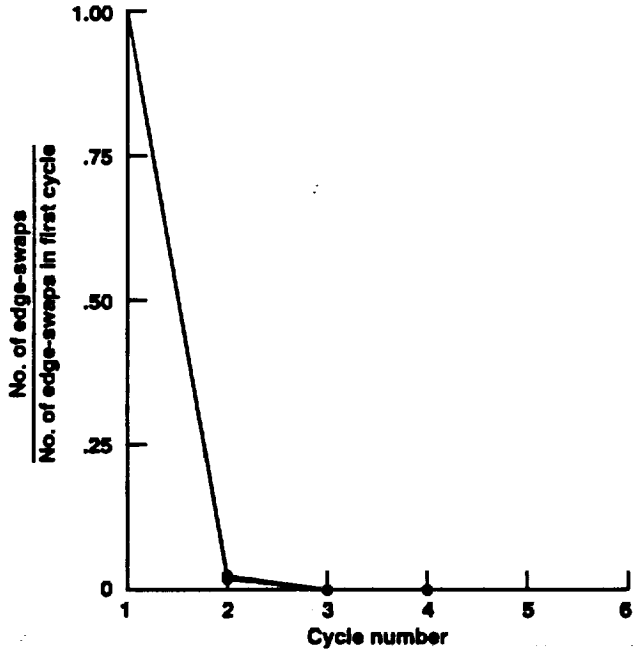

Figure 11. Curves show convergence history of optimization procedure based on MinMax and MaxMin swap criteria as a function of the number of cycles. Curves are derived from meshes containing 1,000 and 10,000 random points in a unit cube and have been normalized to have the same maximum. Observe that the convergence is fairly rapid and is not dependent on the size of the mesh.

	Before	After
Nodes	8,000	8,000
Edges	60,928	54,653
Faces	105,729	93,179
Cells	52,800	46,525

Observe that the number of edges, faces, and cells goes down quite dramatically. These reductions directly translate into reduced CPU time per iteration. Because most of the swaps in MinMax/MaxMin optimization are from the three-tetrahedra configuration to the two-tetrahedra configuration, the convergence history for the Minimum Edge/Face optimization looks essentially the same as in figure 11.

5 MESH REFINEMENT AND INCREMENTAL MESH GENERATION

In this section we discuss improving the quality of a mesh by inserting new sites into the mesh and retriangulating. This procedure can be used to adaptively refine meshes and for generating meshes incrementally. The only difference lies in how the new sites to be inserted are chosen. In the former case, the sites to be inserted are chosen so as to better resolve some flow features whereas in the latter case, the sites are chosen so as to improve the overall quality of the mesh. We discuss both of these in detail below.

The layout of this section is as follows. First, we address the issue of obtaining the initial volume triangulation from a surface definition for incremental mesh generation to work with. Then, we describe the operation of inserting a site and retriangulating the mesh following the site insertion. Finally, we look at inserting sites based on the solution (mesh refinement) and based on improving mesh-quality measures (incremental mesh generation).

Obtaining the Initial Volume Triangulation

Given the surface definition of the objects around which the grid is to be constructed and the far-field boundary, we need an initial volume triangulation. This is the starting point for the incremental mesh-generation and refinement process. Necessarily, this is a constrained triangulation since some of the faces in the triangulation are already specified, and we are not permitted to alter these. Constrained Delaunay triangulation or constrained triangulations using other measures are not well-defined in 3-D. At present, constrained Delaunay triangulations are obtained using the Tanemura/Merriam algorithm. This algorithm works by propagating a front from the defined surfaces such that the tetrahedra formed satisfy the circumsphere criteria.

There are two difficulties with this procedure. First, since a 3-D-constrained Delaunay triangulation is not well-defined, the algorithm produces different meshes depending on the order of traversal through the surface faces. Second, there might be some volumes created during the front propagation that are non-tetrahedralizable. Sprinkling random points in the domain serves to avoid the latter problem and makes the algorithm considerably more robust. We do not know a cure for the former difficulty.

Mesh Refinement Operation

The edge-swapping algorithm provides an effective way of inserting a point into an existing triangulation. Simply find the tetrahedra into which the point is to be inserted, connect all four faces to the inserted point, and test the faces according to the swap criteria to determine if edge-swapping should take place. If a set of five points corresponding to a face is retriangulated, we proceed to test all the outer faces of the new triangulation for swappability and so on. This propagates a front that retriangulates the mesh. It is known that any new face created during the retriangulation of a Delaunay mesh is indeed a part of the final mesh as well, and so back-propagation of the front is not required; that is, it is sufficient for the front to move only outward, and there is no need to propagate information about a local transformation in the reverse direction. This may not be true for other mesh-quality measures, and back propagation then becomes necessary. In the discussion that follows, when we speak about edge-swapping for Delaunay triangulations we mean the edge-swapping algorithm without the helper-strategy idea.

Rajan proves that it is possible to find a certain sequence of edge-swaps that will guarantee that Delaunay triangulation is recovered when a site is added to an existing Delaunay triangulation (ref. 2). In practice, however, it is found that this ordering of edge-swaps does not seem to be necessary in order to recover the Delaunay triangulation. This has been recently proven by Joe (ref. 12). This may not be true of other mesh qualities. In fact, for MinMax and MaxMin triangulations, since even the initial triangulation is most likely not globally optimum, it does not make sense to talk about recovering the MinMax or MaxMin triangulation. It is only hoped that the resulting triangulation is not too far away from the globally optimum triangulation.

It is useful to recognize that once a point has been inserted, one is free to retriangulate the mesh using any swap criteria. In this way, this procedure is superior to Bowyer's algorithm, because unlike Bowyer's algorithm alternative swap criteria can be employed to obtain triangulations other than Delaunay triangulations.

Now, we address the issue of how the site to be inserted is chosen. Since our work involves finite-volume calculations, refining cells (tetrahedra) makes sense. We insert a site at the circumcenter of the cell marked for refinement. This makes sense for Delaunay triangulations, for the inserted point is equidistant from the four nodes of the tetrahedron marked for refinement (this is easier to see by considering the corresponding Voronoi diagram). It is not clear how well this idea works with other mesh-quality measures.

To find the cell in which the new site lies, a walking algorithm is employed. Starting at an arbitrary cell T_i, barycentrics are computed to determine which face of T_i the new site lies behind. The next step is to traverse to the cell behind that face. This procedure is applied recursively until the cell in which the new site falls is found. When doing cellwise refinement, the cell T_i which is marked for refinement can be used as the seed for the walking algorithm, for the insertion site does not always lie within T_i.

It would be desirable to refine the boundaries as we refine the volume. An analytic representation of the boundary surfaces in the form of splines, NURBS, etc. can be constructed from the initial data. When needed, sites can be added on the surface, and the surface can be retriangulated to improve its quality. The boundary surfaces are two-manifolds in three-space, and this retriangulation can be

done using 2-D edge-swapping by considering the triangulation in an appropriate two-parametric space. The volume triangulation is a function of the surface triangulation, and we do not know a way based on edge-swapping (i.e., local transformations) to obtain the volume triangulation corresponding to a modified surface triangulation, short of starting from scratch. More elaborate algorithms that entail retriangulating small cavities can be employed to carry out this operation, but there is no guarantee that the entire mesh will not have to be retriangulated. This is a topic for further research.

The insertion algorithm can be used to adaptively refine meshes. Following, we describe some criteria for mesh-refinement and examine the resulting meshes.

Adaptive Mesh Refinement

Sites may be added where large gradients occur in the solution in order to better resolve discontinuities and sharp flow features. Refining by gradients in velocity, entropy, and density yields good results, though other flow variables may also be used, depending on the flow characteristics. The solution variables are defined at the nodes of the mesh, and we define the gradient of quantity q cell-wise as the largest difference in q along the six edges of the tetrahedra multiplied by the cube root of the volume of the tetrahedra (in order to preferentially refine larger cells). A heap (sorted list) of cells is constructed according to the cell-wise gradients. The cell on the top of the heap is refined by inserting a site at its circumcenter. The mesh is retriangulated by using the swap-criteria following each site insertion and the heap is updated. This process is repeated until a prescribed number of sites has been added.

We studied the case of a Mach 1.5 flow around two spheres in which the streamwise direction is along the line connecting the spheres. We started with a 5,000-node mesh. The solution on this mesh was computed and 2,000 sites were added in regions of large cell-wise density gradients. The refinement was repeated by using cell-wise density gradients to add 2,000 sites, and finally cell-wise velocity gradients were used to add an additional 8,000 sites in three refinements (2,000, 4,000, and 5,000). We used the Delaunay InSphere test for edge-swapping. A slice of the grid and the corresponding solution for the 5,000-node mesh and the final 20,000-node mesh are shown in figure 12. We see that this algorithm works to sharpen the shocks and the regions in the wakes of the spheres. The standoff distance between the leading sphere and the bow shock is in good agreement with experiment. Notice also that the final solution is considerably more symmetrical than the initial solution.

Incremental Mesh Generation

It is generally accepted that for Euler calculations, a *well-shaped* element is a uniform tetrahedron. In order to optimize a mesh for Euler calculations, sites can be added so that the resulting mesh has elements of small aspect ratio. The idea of adding sites at the circumcenters of tetrahedra with large aspect ratios works well here. When working with the Delaunay circumsphere criteria, this produces high-quality meshes in 2-D and seems to work well in 3-D also. It is not obvious how well this site-insertion scheme works with other mesh-quality measures. A heap (sorted list) is maintained according to the cell's aspect ratios. Cells are retrieved from this heap in order of aspect ratio until the target maximum aspect ratio is met. The mesh is retriangulated by using the procedure mentioned above, following each site-insertion in order to recover the quality of the triangulation. The heap is also updated following each site-insertion.

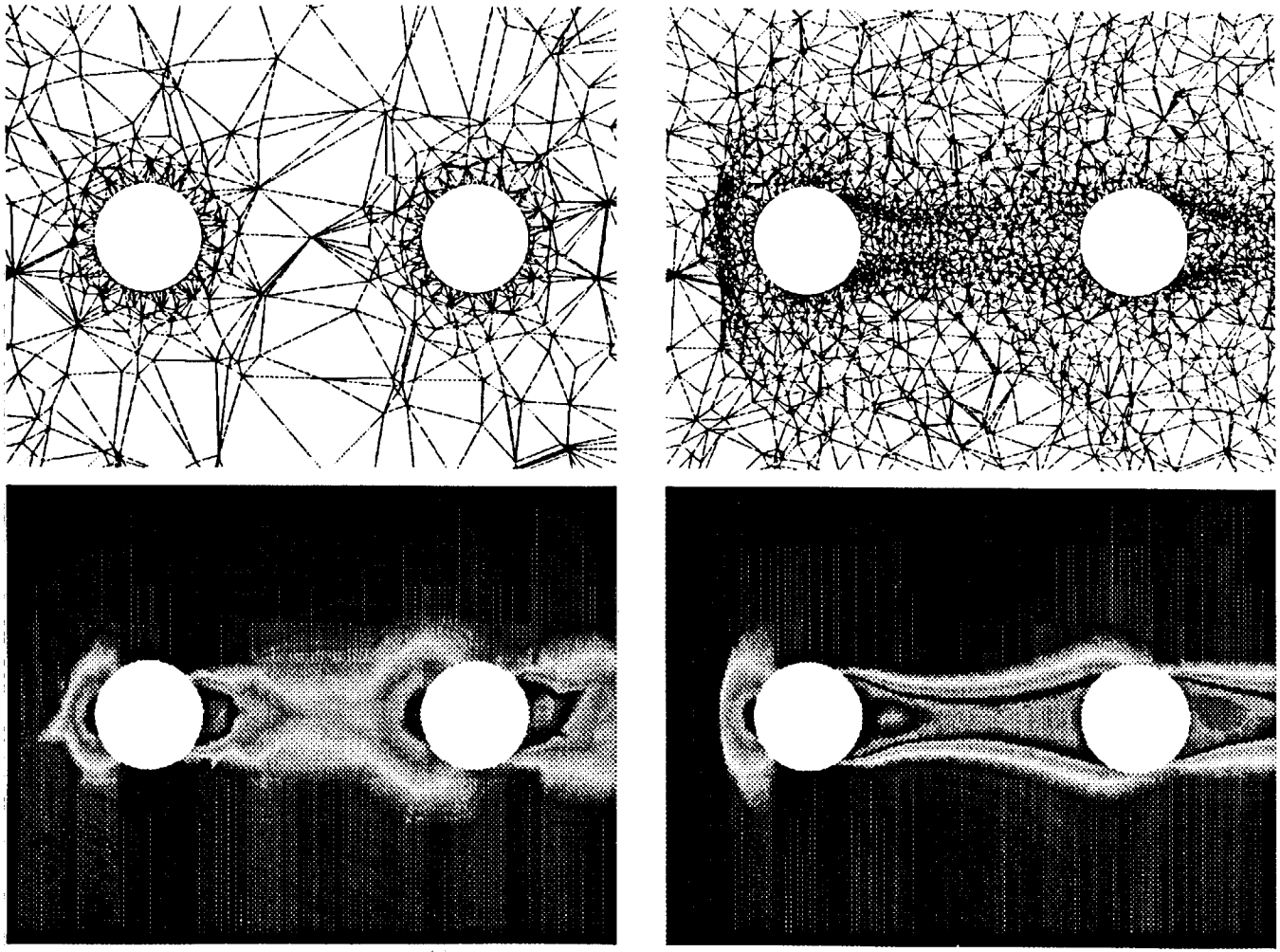

Figure 12. An illustration of the effect of refining the mesh based on the solution. The top left figure shows the intersection of a 5,000-node mesh around two spheres with a plane passing through the equatorial plane of the spheres. Notice that the elements are fairly well-shaped. The bottom left figure shows the velocity distribution on the cutting plane from a solution computed on this mesh for a free-stream velocity of Mach 1.5 where the streamwise direction is in the line connecting the two spheres. The bow shock cannot be resolved well because the grid points are not clustered enough at the shock location. On the top right is the same mesh after 15,000 nodes have been added at regions of high density and velocity gradients. Notice the clustering of points in the bow shock region and in the wake behind the spheres where there is stagnation. On the bottom right is the solution computed using the refined mesh. The features are considerably better resolved.

There are several possible definitions for the aspect ratio of a tetrahedron. One is the ratio of lengths of the longest and the shortest edge. Another is the ratio of radii of the circumsphere and insphere of the tetrahedron (see appendix C). Figure 13 shows the decrease in aspect ratio (as defined by the former definition) as sites are added at the circumcenters of refinable tetrahedra with large aspect ratios. Refinable tetrahedra are those whose circumcenters lie within the mesh. The Delaunay InSphere test is used as the swap criteria.

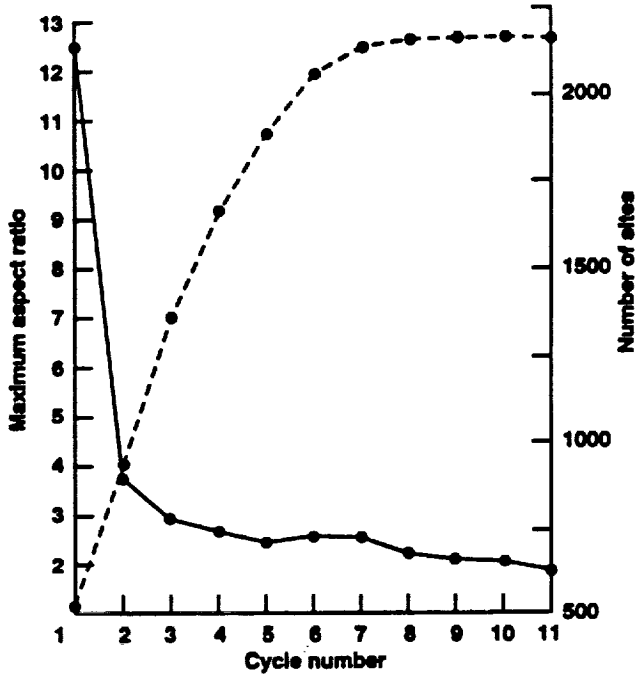

Figure 13. Solid curve indicates the dropping maximum aspect ratio of refinable cells as a function of how many times all the cells in the mesh are cycled over. The dotted line indicates the total number of sites in the mesh. There is more than a fourfold increase in the number of sites in the mesh to reduce the maximum aspect ratio from 12 to 2.

6 CONCLUDING REMARKS AND FUTURE RESEARCH

In this paper, techniques are presented based on an edge-swapping method for 3-D unstructured meshes. Mesh adaptation, incremental mesh refinement, and mesh optimization are explored using this method. The techniques described offer an advantage over existing methods in that arbitrary criteria for mesh quality can be employed.

The important conjectures relating to Delaunay triangulations are as follows:

1. Edge-swaps with the helper-strategy idea need not be performed in any particular sequence in order to transform an arbitrary triangulation into the Delaunay triangulation (mesh optimization).

2. Edge-swaps (without the helper-strategy idea) need not be performed in any particular sequence in order to recover the Delaunay triangulation following a site insertion (mesh refinement).

All the searching is done locally, so the time required to do an operation is independent of the total number of points.

Some future directions for this work follow:

1. Investigating mesh-quality measures: With the aid of our edge-swapping primitive, different mesh-quality measures can be attempted. Clearly, more work is possible in investigating meshes produced by using new mesh-quality measures.

2. The helper strategy: We think that there is more to the helper strategy for Delaunay triangulations than meets the eye. We suspect that (1) it is possible to actually *prove* why the helper strategy yields Delaunay triangulations without getting stuck in local optima, and (2) there is some characterization of the helper strategy which makes it possible to obtain globally optimum MinMax and MaxMin triangulations also.

3. Surface refinement: As we mentioned in section 5, obtaining a volume triangulation corresponding to a modified surface triangulation appears problematic. Having this capability would improve the quality of refined meshes dramatically and, hence, we consider this an important problem on which to focus future research.

APPENDIX A

COMPUTATIONAL ASPECTS OF 3-D EDGE-SWAPPING

Here we discuss the computational aspects of some of the operations needed for the edge-swapping algorithm.

Determining Convexity

This operation tests whether the shape formed by the tetrahedra $\mathcal{T}_1$ and $\mathcal{T}_2$ is convex. Let the vertices of the tetrahedra be numbered $\mathcal{T}_1 = (1,2,3,4)$ and $\mathcal{T}_2 = (1,2,3,5)$, that is, $(1,2,3)$ is the face shared by $\mathcal{T}_1$ and $\mathcal{T}_2$ and nodes 4 and 5 are at the two ends of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Barycentric coordinates are employed to perform the convexity test. The $b_{1,2,3,4}$ satisfying

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} 1 \\ x_5 \\ y_5 \\ z_5 \end{bmatrix}$$

are called the barycentric coordinates of node 5. They indicate the position of 5 in relation to the nodes of tetrahedron $\mathcal{T}_1$. For each s, the sign of b_s indicates the position of 5 relative to the plane H_s passing through the triangular face opposite node s. Thus, $b_s = 0$ when 5 is in H_s; $b_s > 0$ when 5 is on the same side of H_s as node s; and $b_s < 0$ when 5 is on the opposite side of H_s from node s. Clearly, $b_{1,2,3,4} > 0$ if 5 lies inside tetrahedron $(1,2,3,4)$. If the cone formed by planes $H_{1,2,3}$ of $\mathcal{T}_1$ is considered, then $\mathcal{T}_1$ and $\mathcal{T}_2$ would form a convex shape if and only if node 5 lies in the cone on the side opposite from node 4 (fig. A-1).

The conditions to satisfy this requirement are $b_4 < 0$ and $b_{1,2,3} > 0$. In order to test if three-cells form a convex shape, the nodes are renumbered as if the three-cell configuration were edge-swapped into the corresponding two-cell configuration for purposes of the barycentric test. For example, consider a three-cell configuration with numbering $(1,2,3,4)$, $(2,3,4,5)$, and $(1,2,4,5)$; then the corresponding two-cell configuration would have $(1,3,5)$ as the common face and nodes 2 and 4 at the ends of the two tetrahedra. Notice that if the three tetrahedra formed a convex shape, then it would be possible

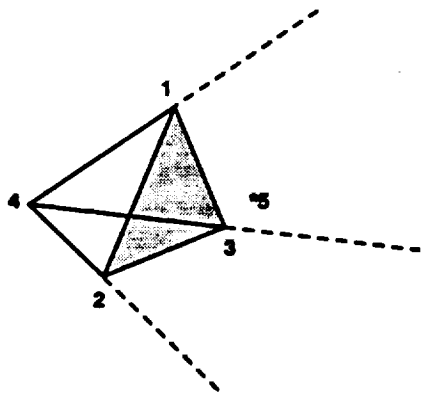

Figure A-1. Convexity cone for node 4.

to edge-swap it to a convex two-tetrahedra configuration. If the three cells formed a concave shape, however, the transformed two-cell triangulation would contain overlapping tetrahedra, which the test above would label as concave.

Using Cramer's rule to solve the $Ax = b$ problem posed above requires computing determinants of the form

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{vmatrix}$$

five times. An optimization is possible by exploiting the property of determinants that subtracting one row or column from another leaves the determinant unchanged. If the first column is subtracted from the rest, the above 4×4 determinant simplifies into a 3×3 one.

$$\overset{C_{2,3,4}-C_1}{\longrightarrow} \begin{vmatrix} 1 & 0 & 0 & 0 \\ a_{11} & a_{12}-a_{11} & a_{13}-a_{11} & a_{14}-a_{11} \\ a_{21} & a_{22}-a_{21} & a_{23}-a_{21} & a_{24}-a_{21} \\ a_{31} & a_{32}-a_{31} & a_{33}-a_{31} & a_{34}-a_{31} \end{vmatrix} = \begin{vmatrix} a_{12}-a_{11} & a_{13}-a_{11} & a_{14}-a_{11} \\ a_{22}-a_{21} & a_{23}-a_{21} & a_{24}-a_{21} \\ a_{32}-a_{31} & a_{33}-a_{31} & a_{34}-a_{31} \end{vmatrix}$$

Delaunay Circumsphere Test

The 3-D Delaunay triangulation is defined as the unique triangulation such that the circumsphere of any tetrahedron contains no other point in the mesh. To determine where point E lies in relation to the circumsphere of tetrahedron (A, B, C, D), denoted by $(\bigcirc\ ABCD)$, the following InSphere primitive is employed:

$$\text{InSphere}\ (A,B,C,D,E) = \begin{cases} < 0 & \text{if } E \text{ is inside } \bigcirc\ ABCD \\ = 0 & \text{if } E \text{ is on } \bigcirc\ ABCD \\ > 0 & \text{if } E \text{ is outside } \bigcirc\ ABCD \end{cases}$$

where InSphere is computed from the following determinant:

$$\begin{aligned} &\text{InSphere}\ (A,B,C,D,E) \\ &= \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ x_A & x_B & x_C & x_D & x_E \\ y_A & y_B & y_C & y_D & y_E \\ z_A & z_B & z_C & z_D & z_E \\ w_A^2 & w_B^2 & w_C^2 & w_D^2 & w_E^2 \end{vmatrix} \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_A & x_B & x_C & x_D \\ y_A & y_B & y_C & y_D \\ z_A & z_B & z_C & z_D \end{vmatrix} \\ &= \begin{vmatrix} x_B-x_A & x_C-x_A & x_D-x_A & x_E-x_A \\ y_B-y_A & y_C-y_A & y_D-y_A & y_E-y_A \\ z_B-z_A & z_C-z_A & z_D-z_A & z_E-z_A \\ w_B^2-w_A^2 & w_C^2-w_A^2 & w_D^2-w_A^2 & w_E^2-w_A^2 \end{vmatrix} \begin{vmatrix} x_B-x_A & x_C-x_A & x_D-x_A \\ y_B-y_A & y_C-y_A & y_D-y_A \\ z_B-z_A & z_C-z_A & z_D-z_A \end{vmatrix} \end{aligned}$$

and $w_P^2 = x_P^2 + y_P^2 + z_P^2$

The first determinant is the 3-D extension of Guibas's InCircle primitive (ref. 14). It represents the volume of a pentatope whose vertices are the points A, B, C, D, E projected onto the 4-D paraboloid

$(x^2+y^2+z^2)$. (A pentatope is the simplest polytope in 4-D just as a tetrahedron is the simplest polytope in 3-D and a triangle in 2-D. A pentatope can be constructed by joining the tetrahedron to a fifth point outside its three-space (ref. 15)). The coordinates in three-space of these five points remain unchanged; they simply acquire a value in their fourth coordinate equal to the square of their distances from the origin. The volume of this polytope is positive if point E lies outside $\bigcirc\ ABCD$ and negative if point E lies inside $\bigcirc\ ABCD$, provided that tetrahedron (A, B, C, D) has a positive volume (as given by the second determinant). The determinant is degenerate if point E lies exactly on $\bigcirc\ ABCD$.

This test is motivated by the observation in 3-D that the intersection of a cylinder and a unit paraboloid is an ellipse lying in a plane (fig. A-2). So, any four co-circular points in 2-D will project to four co-planar points, and the volume of the tetrahedra made from these four co-planar points will be zero. The paraboloid is a surface that is convex upward; the points interior to a circle get projected to the paraboloid below the intersection plane, and the points exterior to it get projected above the intersection plane. If a point lies outside the circumcircle of three other points, the tetrahedron made from these four points will have positive volume, provided the three points were ordered in a counterclockwise fashion. The volume will be negative if the point lies inside the circumcircle.

The second determinant is the 3-D extension of Guibas's *CCW* (counterclockwise) primitive (ref. 14), which computes the volume of tetrahedron (A, B, C, D). Thus, the InSphere primitive works irrespective of how the points A, B, C, and D are ordered. If it can be guaranteed that all the tetrahedra in a mesh have their vertices ordered to have positive volumes then the need to compute the second determinant is eliminated. The InSphere primitive becomes ill-behaved when the points A, B, C, and D all lie nearly on a plane, because the position of the circumsphere with respect to the points (i.e., whether the circumsphere is above or below the plane) becomes very sensitive to small perturbations in the coordinates of the five points.

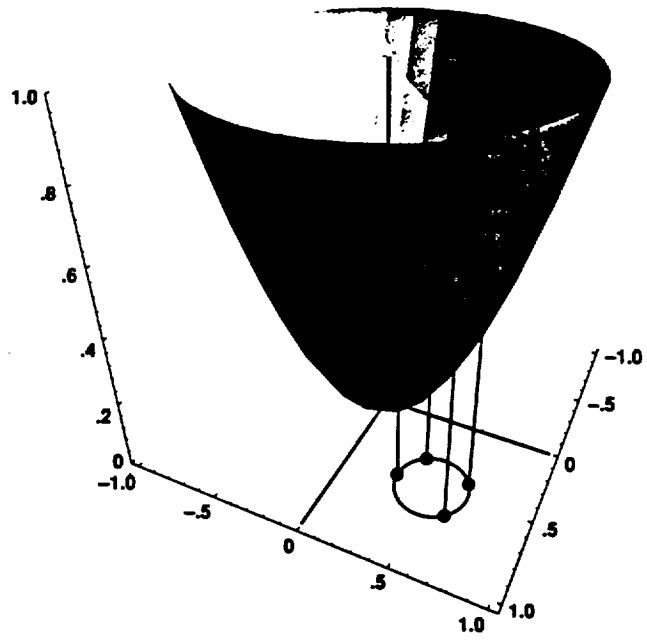

Figure A-2. Projection of cocircular points onto unit paraboloid.

APPENDIX B

RELATIONSHIP BETWEEN BARYCENTRICS AND THE INSPHERE TEST

Some remarks are made here about barycentric coordinates of two adjacent tetrahedra and the InSphere test. The convention that nodes 1, 2, and 3 represent the common face between two tetrahedra and that nodes 4 and 5 are at the ends of these tetrahedra is used. Here, only the sign of the barycentrics is considered and the cases in which any of the barycentric coordinates take on the value zero are ignored.

1. $b_4 > 0$: This means that nodes 5 and 4 are on the same side of face 123, which they share. This would imply overlapping cells and so this should never happen.

2. $b_4 < 0$ and $b_{1,2,3} < 0$: This can never happen, for there is no place where node 5 can be put so as to have these barycentric coordinates.

3. $b_4 < 0$ and two of $b_{1,2,3} < 0$ and one of $b_{1,2,3} > 0$: All the six pairs of tetrahedra in figure 7(a) have these barycentric coordinates. The shape formed by any pair is concave. The same is true of any group of three tetrahedra from this set of four. All the four tetrahedra together form a convex shape. Let us say for instance that the barycentric coordinates for a pair of tetrahedra were $b4, 1, 2 < 0$, and $b_3 > 0$. There is no point that is shared between the cone formed by $H_{4,1,2}$ and the interior of ◯ 1234 on the other side of tetrahedra 1234. Hence, the triangulation of these two tetrahedra is Delaunay. For this result, only the barycentrics are considered, and the fact that these tetrahedra came from a four-tetrahedra configuration is not used. So, if any two tetrahedra have these barycentrics they will form a concave shape and would be Delaunay triangulated. If all the four tetrahedra in case 5 were present, all the pairs of tetrahedra would be Delaunay triangulated, and hence the entire configuration would also be Delaunay triangulated. Since this is the only configuration from five points that admits four tetrahedra we can say that *four tetrahedra made from five points are always Delaunay triangulated.* If only three of the four tetrahedra were present, the three pairs of tetrahedra would again have these barycentrics and would thus be Delaunay triangulated. This happens to be the only way to produce a concave shape formed of three cells from five points and so we can conclude that *a concave configuration of three tetrahedra made from five points is always Delaunay.*

4. $b_4 < 0$ and one of $b_{1,2,3} < 0$ and two of $b_{1,2,3} > 0$: The three pairs of tetrahedra in figure 6(b) have these barycentric coordinates. The shape formed by any pair is concave, whereas the shape formed by all the three tetrahedra is convex. Whether this configuration satisfies the InSphere test depends on how the points are placed. If the configuration is non-Delaunay and all the three tetrahedra are present, then we can edge-swap to the configuration in figure 6(a). If there are only two of the three tetrahedra present, then we would not be able to edge-swap this configuration. This happens to be the only case in which we cannot edge-swap to make the triangulation Delaunay.

5. $b_4 < 0$ and $b_{1,2,3} > 0$: This occurs for the pair of tetrahedra arranged as shown in figure 6(a). The barycentrics of this configuration guarantee convexity, and whether it satisfies the InSphere test depends on how the points are placed. If the InSphere test is not satisfied, we can edge-swap to go into the configuration shown in figure 7(b).

The configurations in figures 6(c) and 6(d) and in figures 7(b) and 7(c) all have $b_4 < 0$. Figure 7(b) has one of $b_{1,2,3} > 0$ and two of $b_{1,2,3} = 0$; figures 6(b) and 6(c) have two of $b_{1,2,3} > 0$ and one of $b_{1,2,3} = 0$; and figure 7(c) has one of $b_{1,2,3} < 0$, one of $b_{1,2,3} > 0$, and one of $b_{1,2,3} = 0$.

Whether a face is part of the Delaunay triangulation of a mesh depends only on whether the cells on the two sides of that face pass the InSphere test. So when we say that five points are Delaunay triangulated, we mean only that the interior faces of this triangulation will be part of the Delaunay triangulation of the mesh regardless of the other points in the mesh. The outer faces may or may not survive, depending on how the other points in the mesh are distributed.

APPENDIX C

FORMULAS FOR TETRAHEDRAL IN-SPHERES AND CIRCUMSPHERES

In-sphere (Sphere Inscribed within a Tetrahedron)

A sphere inscribed within a tetrahedron is defined as the largest sphere that is completely contained within a tetrahedron. This sphere is tangent to the four faces of the tetrahedron, that is, the perpendicular distance between the center of the in-sphere and the four faces is the same (not to be confused with the InSphere test).

The perpendicular distance between a point (x_p, y_p, z_p) and the plane $ax + by + cz + d = 0$ is given by

$$\frac{ax_p + by_p + cz_p + d}{s_i\sqrt{a^2 + b^2 + c^2}}$$

where s_i is either $+1$ or -1, chosen to have the same sign as the numerator, in order to make the distance a positive quantity. The points satisfying $ax+by+cz+d = 0$ form the dividing plane between regions where points satisfy $ax + by + cz + d < 0$ and where they satisfy $ax + by + cz + d > 0$, and so the numerator will be either positive or negative, depending on which side of the plane the point (x_p, y_p, z_p) lies.

The equation of a plane passing through points $(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)$ is given by

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0$$

The determinant can be expanded in x, y, and z to give an expression of the plane in the form $ax + by + cz + d = 0$. Let a_i, b_i, c_i, and d_i for $i = 1, \ldots, 4$ be the coefficients in the equations of the plane passing through the four faces. Then, the center (x_c, y_c, z_c) and radius r of the in-sphere satisfy the equations

$$a_i x_c + b_i y_c + c_i z_c + d_i = s_i\sqrt{{a_i}^2 + {b_i}^2 + {c_i}^2}r, \quad i = 1, \ldots, 4$$

The center of the in-sphere always lies within the tetrahedron and so any other point in the interior of the tetrahedron (the centroid for instance) will have the same s_i's as the center of the in-sphere. Hence, s_i's for (x_c, y_c, z_c) can be determined before knowing the exact location of (x_c, y_c, z_c). We have the following system:

$$\begin{bmatrix} a_1 & b_1 & c_1 & -s_1\sqrt{{a_1}^2 + {b_1}^2 + {c_1}^2} \\ a_2 & b_2 & c_2 & -s_2\sqrt{{a_2}^2 + {b_2}^2 + {c_2}^2} \\ a_3 & b_3 & c_3 & -s_3\sqrt{{a_3}^2 + {b_3}^2 + {c_3}^2} \\ a_4 & b_4 & c_4 & -s_4\sqrt{{a_4}^2 + {b_4}^2 + {c_4}^2} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ r \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_2 \\ -d_3 \\ -d_4 \end{bmatrix}$$

Circumsphere (Sphere Circumscribing a Tetrahedron)

A sphere circumscribing a tetrahedron is defined as the smallest sphere that completely contains the tetrahedron. This sphere passes through the four nodes of the tetrahedron; that is, the center of the sphere is equidistant from these four nodes. This yields the following four equations:

$$(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2 = r^2, \quad i = 1, \ldots, 4$$

where (x_i, y_i, z_i) are the coordinates of the ith node and where (x_c, y_c, z_c) represents the center of the circumsphere and r the radius. Expanding and rearranging the terms in the above equations yields

$$-2x_i x_c - 2y_i y_c - 2z_i z_c + r^2 + {x_c}^2 + {y_c}^2 + {z_c}^2 = {x_i}^2 + {y_i}^2 + {z_i}^2, \quad i = 1, \ldots, 4$$

Thus, (x_c, y_c, z_c) can be found by solving the following system:

$$\begin{bmatrix} -2x_1 & -2y_1 & -2z_1 & 1 \\ -2x_2 & -2y_2 & -2z_2 & 1 \\ -2x_3 & -2y_3 & -2z_3 & 1 \\ -2x_4 & -2y_4 & -2z_4 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ u \end{bmatrix} = \begin{bmatrix} -({x_1}^2 + {y_1}^2 + {z_1}^2) \\ -({x_2}^2 + {y_2}^2 + {z_2}^2) \\ -({x_3}^2 + {y_3}^2 + {z_3}^2) \\ -({x_4}^2 + {y_4}^2 + {z_4}^2) \end{bmatrix}$$

where $u = {x_c}^2 + {y_c}^2 + {z_c}^2 - r^2$. Even though u is written in terms of x_c, y_c, and z_c, it does represent an independent degree of freedom. The radius r can be computed more easily as the distance between the center of the circumsphere and any of the four nodes of the tetrahedron.

REFERENCES

1. Lawson, C. L.: Software for C^1 Surface Interpolation. Mathematical Software III, John R. Rice, ed., Academic Press, New York, 1977.

2. Rajan, V. T.: Optimality of the Delaunay Triangulation in R^d. Proceedings of the 7th ACM Symposium on Computational Geometry, 1991, pp. 357–363.

3. Rippa, S.: Minimal Roughness Property of the Delaunay Triangulation. Computer Aided Geometric Design, vol. 7, no. 6, 1990, pp. 489–497.

4. Lawson, C. L.: Properties of n-Dimensional Triangulations. Computer Aided Geometric Design, vol. 4, 1986, pp. 231–246.

5. Edelsbrunner, H.: Algorithms in Computational Geometry. Springer-Verlag, Berlin, 1987.

6. Bowyer, A.: Computing Dirichlet Tessellations. Comput. J., vol. 24, 1981, pp. 162–166.

7. Watson, D. F.: Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes. Comput. J., vol. 24, no. 2, 1981, pp. 167–171.

8. Klee, V.: On the Complexity of d-Dimensional Voronoi Diagrams. Arkhiv der Mathematik, vol. 34, 1980.

9. Baker, T. J.: Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation. Engineering with Computers, vol. 5, 1989, pp. 161–175.

10. Baker, T. J.: Unstructured Mesh and Surface Fidelity for Complex Shapes. Proceedings of the 10th AIAA Computational Fluid Dynamics Conference, Hawaii, 1991.

11. Joe, B.: Three-Dimensional Delaunay Triangulations from Local Transformations. SIAM J. Sci. Stat. Comput., vol. 10, 1989, pp. 718–741.

12. Joe, B.: Construction of Three-Dimensional Delaunay Triangulations Using Local Transformations. Computer Aided Geometric Design, vol. 8, 1991, pp. 123–142.

13. Babuška, I.; and Aziz, A.: On the Angle Condition in the Finite Element Method. SIAM J. Numer. Anal., vol. 13, no. 2, 1976.

14. Guibas, L. J.; and Stolfi, J.: Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. ACM Trans. Graph., vol. 4, 1985, pp. 74-123.

15. Coxeter, H.: Regular Polytopes. Dover Publications, New York, 1973.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	Technical Memorandum

4. TITLE AND SUBTITLE

Three-Dimensional Unstructured Grid Refinement and Optimization Using Edge-Swapping

5. FUNDING NUMBERS

505-10-11

6. AUTHOR(S)

Amar Gandhi and Timothy Barth

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A-92169

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-103966

11. SUPPLEMENTARY NOTES

Point of Contact: Timothy Barth, Ames Research Center, MS 202A-1, Moffett Field, CA 94035-1000
(415) 604-6740

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified-Unlimited
Subject Category – 59

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This paper presents a three-dimensional (3-D) "edge-swapping" method based on local transformations. This method extends Lawson's edge-swapping algorithm into 3-D. The 3-D edge-swapping algorithm is employed for the purpose of refining and optimizing unstructured meshes according to arbitrary mesh-quality measures. Several criteria including Delaunay triangulations are examined. Extensions from two to three dimensions of several known properties of Delaunay triangulations are also discussed.

14. SUBJECT TERMS

Unstructured grid refinement, Unstructured grid optimization, Edge-swapping, Local transformations, Delaunay triangulations

15. NUMBER OF PAGES

34

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		