Update to: Approaches to the Automatic Generation and Control of Finite Element Meshes

Mark S. Shephard

Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180–3590

This paper updates the status of efforts on the development of automatic mesh generation techniques for general three-dimensional domains. The technical areas reviewed include: (i) issues associated with automatic mesh generation of CAD geometric models, (ii) local mesh modification procedures for improving mesh quality, (iii) advances in tetrahedral mesh generators, (iv) generation of anisotropic meshes, (v) hexahedral mesh generators, and (vi) implementation of automatic mesh generators on parallel computers.

1. INTRODUCTION

The development of techniques to improve the level of automation and reliability of finite element methods has been central to its continued growth and use. A 1988 Applied Mechanics Review article discussed the status of methods for the automatic generation and control of finite element meshes. The current article is an update to that article which focuses exclusively on the advances in the techniques for the automatic three-dimensional mesh generation.

The emphasis in the 1988 article on three-dimensional mesh generation was on the basic approaches for the automatic generation of simplex (tetrahedron) elements. Since that time these have been a substantial number of additional papers devoted to this topic (references [1, 2, 19, 48, 52] are three special issues of journals, a proceedings and a book on the topic). Although the same basic approaches dominate tetrahedral mesh generation, there are several areas of development which have been critical to the advancement of these techniques. The first area, considered in section 2, is the ability to generate meshes of three-dimensional models defined in CAD systems. The other areas are procedures to improve the quality of the meshes created (section 3), imporvements to the meshing algorithms (section 4), and the ability to create anisotropic mesh gradations (section 5).

A substantial change since the 1988 paper is the emphasis on the development of automatic hexahedral mesh generation schemes. The more constrained nature of hexahedral mesh generation has forced consideration of new technologies to account for these constraints. Section 6 indicates the current approaches to the development of automatic hexahedral mesh generators.

A final area of development of automatic mesh generation is the development of automatic mesh generation procedures that operate on parallel computers. Efforts in this area are summarized in section 7.

2. AUTOMATIC MESH GENERATION OF MODELS IN CAD SYSTEMS

A key ingredient in the development of a complete automatic mesh generator is the ability to successfully interact with the computerized representation of the geometric domain to create a valid finite element mesh.

Definition: Automatic Mesh Generator — An algorithmic procedure which can create, under program control and without user input or intervention, a valid mesh for geometric domains of arbitrary complexity.

In the case of the curved domains the geometry of entities of the mesh, at least until their relationship to the geometric domain is properly qualified, do not exactly match the geometry of the geometric model. In these cases the application of decision processes based on measures of how closely the mesh approximates the geometric domain are not able to ensure the validity of the mesh. As an example, consider the simple two-dimensional domain in Figure 1(a) which has one curved edge. The mesh in Figure 1(b) is an invalid mesh of that domain since at point **A** the mesh on one portion of the curve incorrectly connects to the mesh on another portion of the curve. On the other hand, the mesh shown in Figure 1(c) is a valid mesh since the mesh did not connect across at point **A**. However, it is clear that the maximum distance from any mesh edge to the boundary of the geometric model for the mesh in Figure 1(c), δ_c, is much larger than that in Figure 1(b), call it δ_b. Therefore, if one defines mesh validity in terms of maintaining a given distance tolerance, δ, and it is selected such that $\delta_b < \delta < \delta_c$, one would incorrectly conclude that the mesh in Figure 1(b) was valid, while the mesh of Figure 1(c) was not.

To address the issues of meshing curved geometries a definition of a triangulation which is valid with respect to curved geometric domains has been defined:

Definition: Geometric Triangulation [46, 51] — *Given a set P of M unique points, each classified with respect to the geometry G, an n-dimensional geometric triangulation, Λ^n is a set of N nondegenerate elements*

$$\Lambda^n = \left\{ s_1^{d_1}, s_2^{d_2}, s_3^{d_3}, \dots, s_N^{d_N} \right\} \tag{1}$$

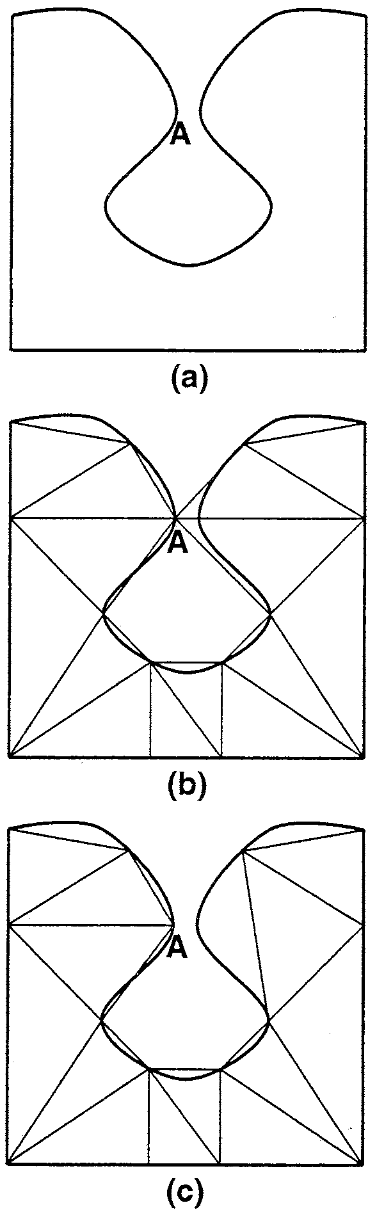


Figure 1. Simple two-dimensional example where "distance" information would not identify the invalid mesh.

where $s_i^{d_i}$ represents the i^{th} element, which is of dimension d_i with $0 \leq d_i \leq n$, of the set. The members of the set satisfy the following properties:

i. *All vertices of each $s_i^{d_i} \in P$*

ii. *For each $i \neq j$, interior $\left(s_i^{d_i}\right) \cap$ interior $\left(s_j^{d_j}\right) = 0$*

iii. *Λ^n is topologically compatible with G*
iv. *Λ^n is geometrically similar to G*

The key differences between the definition of a standard triangulation and a geometric triangulation are the last two items of topological compatibility and geometric similarity. The basic ideal of topological compatibility is the existence of a set of mesh entities which properly cover each of the edges and faces that bound the geometric model, and that fill each of the regions. Figure 2(a) shows a model face covered by a compatible set of mesh faces. Figure 2(b) contains a topological hole characterized by the fact that the three mesh edges are used by only one mesh face on the model face.[1] Figure 2(c) depicts a topological redundancy which is characterized by the four mesh edges each being used by three mesh faces on the model face.

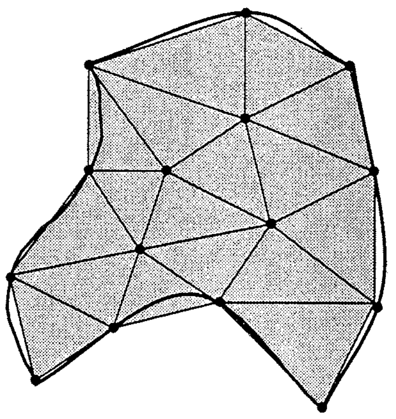
The simplest explanation of a geometrically similar mesh is one that in the limit of refinement will exactly match the geometry of the domain. This simple definition is not a workable one for the development of algorithms to evaluate and correct mesh validity. In all but complex geometric cases, this requirement is satisfied if topological compatibility is satisfied. However, since there is no a priori method to ensure that topological compatibility alone will also ensure geometric similarity, it must be explicitly considered. One method of ensuring geometric similarity employs the concept of parametric intersection [46, 51].

The process of ensuring a set of mesh entities is a valid geometric triangulation of a geometric domain is a function of the mesh generation algorithm. If the initial mesh is created with no concern for its validity; validity must be regained through the application of a mesh assurance algorithm [46, 51].

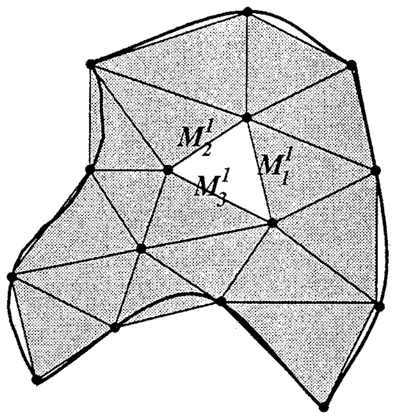
An alternative approach to ensuring mesh validity is to create the mesh by the stepwise discretization of the geometric domain, being sure to maintain the validity of the discretization in each step of the process. The primary mesh validity problem that arises in this approach is due to mesh entities which intersect in three space that were valid in the parametric space of the geometric model entity. This is typically caused by the fact that the geometry of the mesh entities is an approximation to that of the true geometry. Figure 3 shows a simple two-dimensional example with one curved model edge where the mesh entities on the individual model edges are valid, however, one of the mesh edges classified on the curved geometric model edge intersects two other mesh edges.

Automatic mesh generators must interact with the geometric representation of the domain being meshed. One approach to provide the geometric information needed is to provide the meshing algorithm with a data file of the geometric model. Since automatic

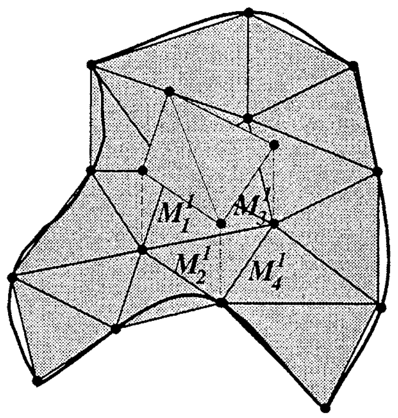
[1] The nomenclature used in this figure, and others, of M_i^d represents the i^{th} mesh entity of dimension d.



(a) compatible mesh



(b) topological hole



(c) topological redundancy

Figure 2. Mesh compatibility.

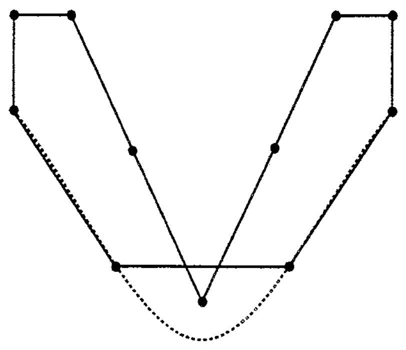


Figure 3. Two-dimensional example showing a self-intersecting boundary mesh.

mesh generators must at least interrogate the geometric representation for information not inherently in the model's data file, this approach necessitates the development of those modeling functionalities within the mesh generator. An alternative approach that more easily maintains consistency with the geometric representation directly employs the functionality of the geometric modeling system in the mesh generation process. An example of a mesh generator using such an approach is one [50] which interacts with the geometric modeling system through a specific set of 21 geometric operators [21]. All the operators are keyed via the topological entities of vertex, edge and face. Approximately half the operators request topological adjacencies. The remaining half are geometric interrogations which are limited to determining pointwise quantities such as the points of intersections and surface normals. This approach has been successfully applied with multiple geometric modeling systems. Figure 4 shows a solid model defined in the Parasolid solid modeler and automatically meshed using this functional approach.

There are several areas of technical complexity that arise when working with general geometric models. Five of the most important are:

1. Accounting for geometric entities which employ periodic or degenerate parameterizations.
2. Accounting for trimmed geometric entities.
3. Near tangencies in the geometric model.
4. Curving finite elements to the model boundary.
5. Account for the adverse effects on the mesh of small geometric features.

An example of the last of these is the small model edge shown in the geometric model of Figure 5(a). This edge was introduced by the solid modeler to close off a face. If this small edge is maintained during the mesh process, poorly shaped elements are obtained. However, by the application of a procedure which locally modifies the mesh to allow a mesh edge to span that edge and part of one of those adjacent to it [17], the element aspect

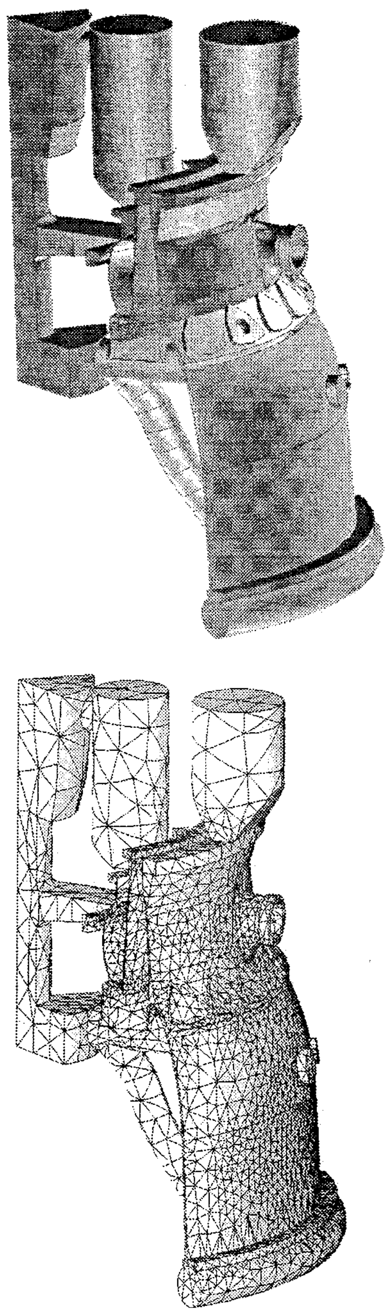


Figure 4. Solid model and automatically generated mesh for casting of engine diffuser with gating.

ratio of the worst element is improved by 1000%, while the smallest dihedral angle is improved from 0.45 degrees to 8.32 degrees.

3. MESH IMPROVEMENT METHODS

Many automatic mesh generators employ local mesh operations to eliminate local geometric invalidities, improve the element shapes, and/or locally make the mesh finer or coarser. The geometric parameters defining the shape of the elements in real space have a specific influence on the ability of that mesh to solve the problem of interest.

There is a limited amount of theoretical information available on the influence of angles of straight sided elements on the convergence of the solution [3]. In addition, a few numerical studies have measured the effects of element shapes on specific element types for specific situations. Although these analyses and studies are limited, a number of useful element shape quality measures have been defined. Commonly applied shape measures for straight sided simplex elements[2] are aspect ratio, ratio of inscribed to circumscribed sphere, smallest dihedral angle and largest dihedral angle. Shape measures in the case of curved boundary elements and elements with four sided faces appear less straight forward. For example, Habraken and Cescotto [22] examined seven distortion measures and found four to be sufficient to detect all types of mesh distortion for eight–noded quadrilaterals in the application they were considering. A general and direct means for gaining a handle on element distortion is to monitor the variation in the Jacobian determinate in the element.

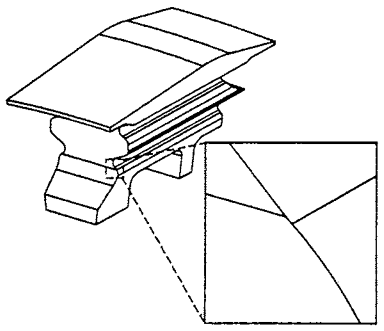
Given a mesh where there are elements of undesirable shape, the hierarchy, with respect to ease of application and ability to ensure shape improvement, of local mesh operations is:

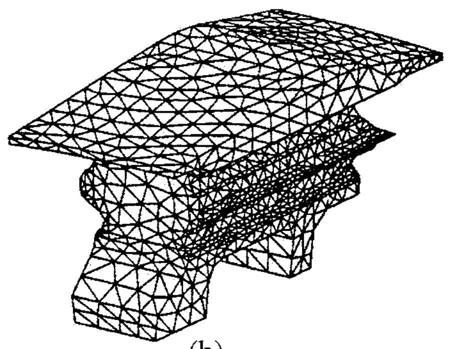
1. vertex repositioning
2. swapping of mesh entities
3. mesh entity split or collapse operations
4. local retriangulations

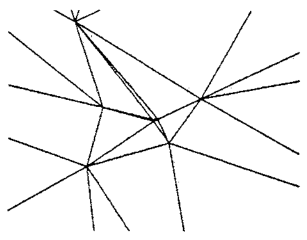
Vertex repositioning can be applied to meshes with elements of any topology. However, with the exception of a small number of specific options, the majority of available swapping, splitting, collapsing and local retriangulation procedures are for the simplex elements of triangles and tetrahedron.

Vertex repositioning algorithms employ one or more criterion to control node motion. The classic criterion is related to a central difference operator for the Laplacian operator on a structured quadrilateral mesh. This approach, when applied iteratively to unstructured meshes is referred to as centroidal smoothing, since it reduces to placing the vertex at the

[2] Straight sided simplex elements have a constant Jacobian determinate over the element.


(a)


(b)


(c)

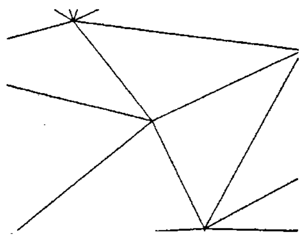

(d)

Figure 5. Example of small feature elimination; (a) geometric model with small feature, (b) final mesh with influence of small feature removed, (c) and (d) mesh in the vicinity of small feature without and with the elimination of adverse influence respectively.

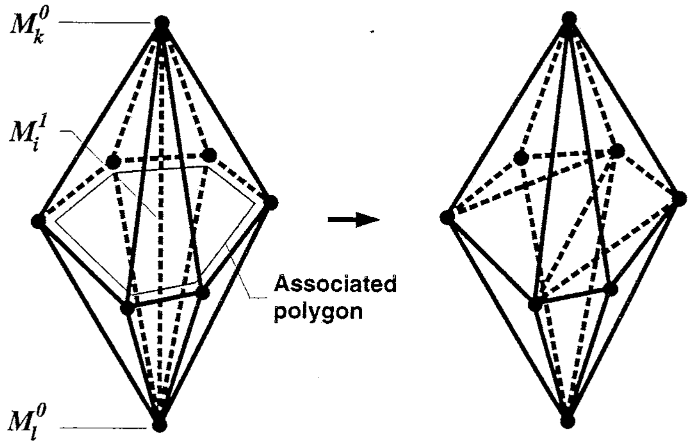


Figure 6. Edge removal.

centroid of the nodes to which it is connected. Improvements to the process are possible in which one conceptually employs spring stiffnesses to further control the node point motion [27]. Since these methods improve the mesh based on an indirect criterion they are not assured to improve the mesh quality. However, their computational efficiency and ability to produce well shaped elements without abrupt changes in the mesh make them popular. Since, in the case of concave domains, specific implementations can take a valid finite element mesh and make it invalid, the more reliable implementations of these procedures employ motion limiters which are based on the direct measurement of element shapes.

Additional vertex repositioning procedures have been proposed based on the satisfaction of specific variational operators and local applications of optimization of the shape of connected elements [36].

The most local of swapping procedures for simplex elements relies on the basic property that a set of $n + 2$ points in $\Re^n$ may be triangulated in, at most, two ways [29]. In three dimensions these swaps are commonly referred to as 2–to–3 and 3–to–2 swaps. Although it has been shown that these two options are sufficient to convert a general triangulation into a Delaunay triangulation [24], more powerful swapping operations are desired when other mesh optimization criteria are being locally applied.

Brière de l'Isle and George [16] have proposed an edge removal technique as part of an algorithm. In edge removal a mesh edge M_i^1 inside a model region which is bounded by vertices M_k^0 and M_l^0 can be eliminated by retriangulating the polyhedron of all connected tetrahedrons. The polyhedron is retriangulated by: (i) triangulating the polygon of all mesh vertices of the polyhedron which are neither M_k^0 nor M_l^0, and (ii) connecting the new mesh faces to M_k^0 and M_l^0 (Fig. 6).

Multi-face removal is a procedure that reverses edge removal, in other words, it considers a configuration that could have resulted from edge removal and obtains the starting configuration. Methods for the valid application of multi-face removal are given in reference [13].

In three dimensions, a mesh region can be split into four new regions, a mesh face into three new faces, and a mesh edge into two new edges. Mesh entity splitting adds a vertex connecting the boundary vertices of the polyhedron to the new vertex. In the case of a mesh region, the polyhedron is the mesh region itself. In the case of a mesh face or a mesh edge, the polyhedron is built from the union of all mesh regions connected to the face or edge, respectively. Figure 7 displays the three types of splits in three dimensions and indicates for each one of them the change in number of mesh regions. If the mesh entity to be split is in a model region, all boundary faces of the polyhedron are fully visible from the split vertex and splitting is therefore always possible. If the mesh entity is on the model boundary and the new vertex has to be snapped to the model entity it is classified on, splitting may not be geometrically possible.

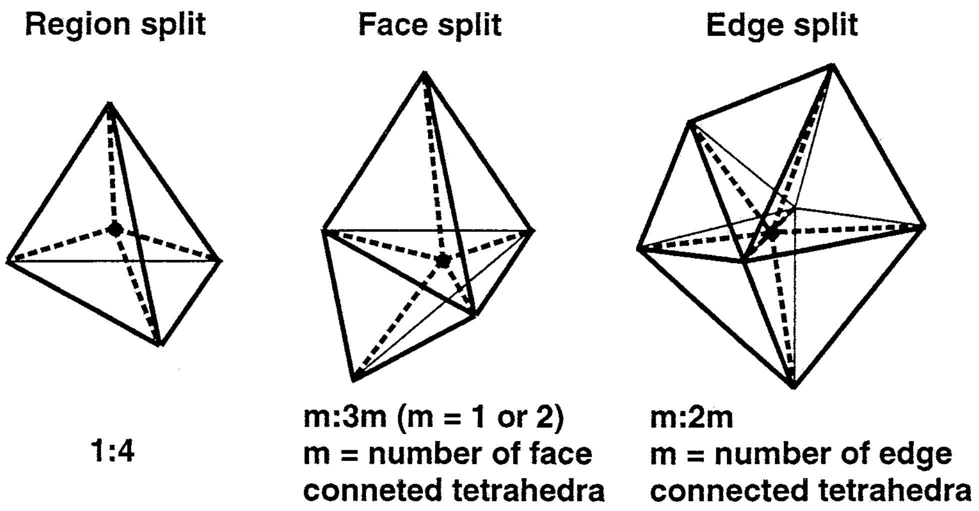
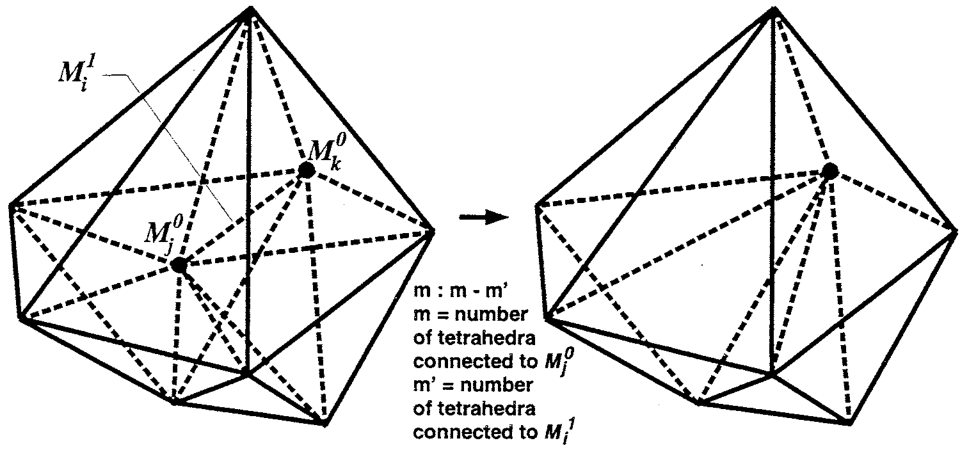


Figure 7. Mesh entity splitting in three dimensions.

Edge collapsing removes one vertex from the triangulation (one of the two end vertices of the edge). Edge collapsing retriangulates the polyhedron of all mesh regions connected to one end vertex (collapsed vertex) by connecting the boundary faces of the polyhedron to the other end vertex (collapsing vertex) (Fig. 8).



M_i^1
M_k^0
M_j^0
m : m - m'
m = number of tetrahedra connected to M_j^0
m' = number of tetrahedra connected to M_i^1

Figure 8. Edge collapsing in three dimensions.

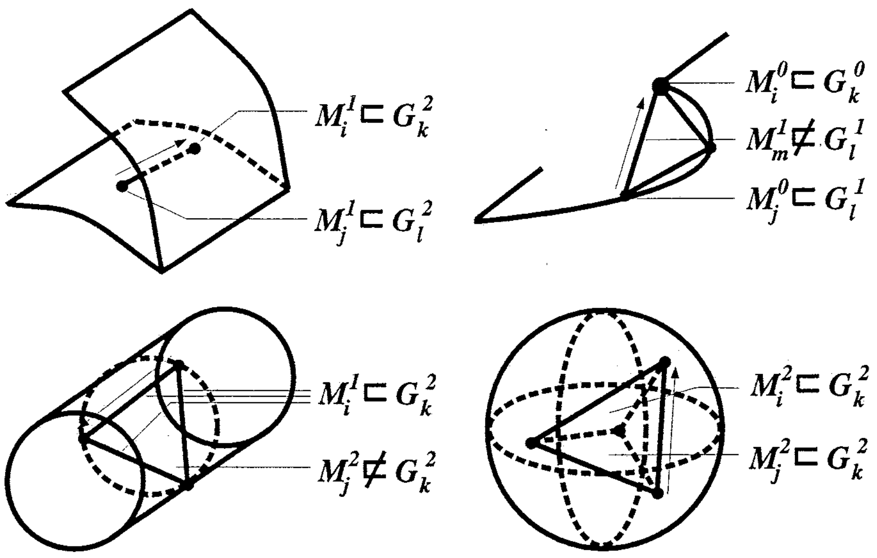


Figure 9. Some cases where edge collapsing in three dimensions is not possible due to topology.

Edge collapsing cannot be performed if at least one of the new mesh regions is geometrically invalid. Figure 9 graphically illustrates some of the cases where edge collapsing is not possible.[3]

[3] The nomenclature used in this figure of $M_i^{d_i} \sqsubset G_j^{d_j}$ indicates that $M_i^{d_i}$, which is the ith mesh entity of dimension d_i, lies on, and is uniquely associated with $G_j^{d_j}$, which is the jth geometric model entity entity of dimension d_j. Note that mesh entities are associated with the lowest order geometric model entity on which they lie and that $d_i \leq d_j$.

The last option for locally improving the mesh is the application of a local retriangulation procedure, where a local retriangulation procedure is any procedure which eliminates a set of mesh entities in a local region and retriangulate that region to obtain an improved mesh.

The application of any of the combination of local mesh modification procedures can be used to improve a given triangulation. Issues on which operations to apply and when for the purposes of reaching a given goal are not necessarily straight forward. The following list indicates some of the basic issues on what is known, and not known, on the application of the mesh local improvement techniques just considered to three-dimensional triangulations:

1. Mesh motion is the most highly constrained of the improvement techniques, thus it will not be able to improve the mesh in situations where other procedures can.
2. Properly applied mesh entity swapping algorithms can be used to convert a triangulation into a Delaunay triangulation. The application of another criteria to drive the process only leads to local optimum. However, in three dimensions Delaunay triangulation can produce poorer meshes than other criteria. For example, starting from a Delaunay triangulation, a "superior mesh" can be obtained by performing swapping based on locally maximizing the minimum angle [25]. An alternative strategy that is likely to be better, particularly in the case of anisotropic refinement, is to locally minimize the maximum dihedral angles on an element-by-element basis.
3. The mesh entity splitting and collapsing, and the local retriangulation procedures are the most flexible of the local mesh modification procedures. However, these procedures do locally alter the mesh gradation and must be applied with care.

One strategy that has been found to be reliable and more powerful than other strategies [13] has two major steps. The first step is to apply a straight forward vertex repositioning technique with the constraint that a vertex is moved in the direction indicated only so long as the shape of the worst shaped element connected to it is improving. The second step applies the more powerful tools of swapping followed by splitting and collapsing to eliminate the small number of elements which have a shape measure below a desired threshold. Although this strategy lacks any proof of convergence to a given goal, it has been found to produce superior mesh improvements, particularly when the criteria used to drive the second step is minimizing the maximum dihedral angle.

4. ADVANCES IN TETRAHEDRAL MESH GENERATION

The development of automatic three-dimensional tetrahedral element mesh generators has a long history (see [2, 19, 47, 48, 52] for recent general information on the topic).

The basic strategies which dominate the creation of elements in these mesh generators are Delaunay triangulation and element removal by the advancing front method.

The simplicity of developing a mesh generator using the circumsphere containment properties of a Delaunay triangulation has been the major attraction of this approach. However, there are a number of critical issues that must be addressed in the development of a Delaunay-based automatic mesh generator. The most obvious issue in the case of general domains is to recover a valid representation of the surface of the domain. A second important issue is dealing with the finite accuracy of real calculations in the application of the circumsphere test. The third issue is ensuring the computational efficiency of the method, particularly the computational growth rate for meshes with large numbers of elements. Current implementations employ a number of devices to deal with these issues. Boundary recovery procedures include forcing point insertion such that the surface triangles are not disturbed, introducing new points on the boundary to recover missing portions of the boundary, to applying local mesh modification operations to recover the missing surface triangles without concern for maintaining a Delaunay triangulation. Devices to deal with finite precision calculations include employing circumspheres ϵ larger or smaller than the exact values such that all errors are one sided and can be handled, and converting all locations to an integer coordinate system and performing only integer calculations. Devices to control computational growth rate range from using secondary searching structures, to first creating a coarse triangulation and performing intelligent point insertion into known entities. References [4, 7, 8, 10, 20, 45, 55] discuss the various issues associated with three-dimensional Delaunay triangulation.

Advancing front mesh generators [23, 30, 32, 34, 37, 38] are element removal mesh generators which employ a specific set of heuristics to create elements working from the boundary, referred to as the front, into the interior. The heuristics used in these algorithms address the order in which faces are removed from the front, where new points are placed on the interior, and when to connect to existing entities on the front to remove a face. The computational efficiency of these procedures is dictated by how well the validity and nearness checks of a proposed removal are performed. The computational growth rate is controlled by using secondary structures, such as a background grid, octree, alternating digital tree, etc., to limit the number of mesh entities on the front that must be checked. The computational effort of a specific set of checks is a strong function of the ordering of the operations and the ability to employ the most efficient geometric operations.

A secondary structure which has proven popular for use in the development of mesh generators is the octree structure. Although there are major differences in the various octree-based meshing approaches published [9, 26, 39, 45, 50], the consistent aspects of these procedures are:

1. The octree structure is used to localize many of the meshing processes.
2. Those octants containing portions of the object's boundary receive specific consideration to deal with the actual boundary of the object.
3. The corners of the octants as well as points defined by the interaction of the boundary of the object being meshed with the octants' boundaries are used as nodes in the mesh.
4. The mesh gradation is controlled by varying the level of the octants within the octree through the domain occupied by the object.

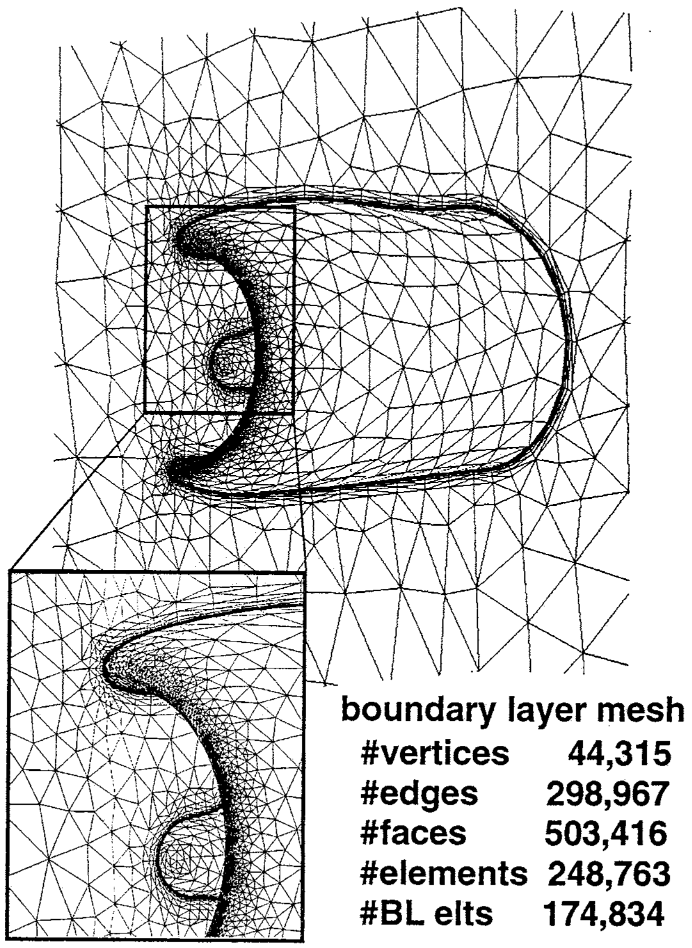
The most critical of these issues is dealing with the interactions of the object's boundaries and the octree. Techniques used here range from allowing only specific interactions, to basing the process on accurately determined intersections coupled with procedures to ensure a valid geometric triangulation. Although straight forward, allowing only specific interactions severely limits the complexity of domains that can be properly handled. Dealing with accurate intersections leads to geometric approximation complications and difficulties in dealing with close object-to-octant boundary interactions that can lead to poorly shaped elements. By the proper selection of algorithmic steps, these issues can be addressed providing meshing procedures capable of meshing general objects (Figs. 4 and 5(b) were generated with an octree-based mesh generator).

5. ANISOTROPIC MESH GENERATION

There are a number of important physical problems where the variations of the solution parameters are strongly directionally dependent. In these cases it is desirable to employ a mesh with high aspect ratio elements such that the short dimension of the element is aligned with the direction of high variation of solution parameters. The problem is that most of the automatic tetrahedral meshing procedures are best suited for the creation of isotropically refined meshes. Therefore, it is desirable to have specific procedures that can produce the appropriate types of anisotropic refinements.

One possibility is to simply provide directionally based mesh control information (which in general must be a second order tensor) and to employ a meshing procedure which can create elements using that information. Such approaches have been used in conjunction with advancing front mesh generators and have proved useful in creating some degree of anisotropy in the mesh. A similar approach is to mesh in a distorted coordinate system using an isotropic mesh generator and to map the mesh back to create an anisotropic mesh. Such procedures have been used with Delaunay mesh generators when the mapping could be easily defined, and with octree meshes when the anisotropy could be aligned with the axes of the octree.

In the special case of the strongly anisotropic refinement desired in the problems with boundary layers, more specialized techniques have been developed. In these problems the strongly anisotropic refinement is needed in the immediate vicinity of the boundary and it needs to be well structured. This need can be met in that small region by the appropriate growing of the surface triangulation into the domain on the boundary layer surfaces in a direction more-or-less normal to the boundary [11, 28, 33, 40]. Figure 10 shows a boundary layer mesh generated in the vicinity of an engine nacelle.



boundary layer mesh

#vertices	**44,315**
#edges	**298,967**
#faces	**503,416**
#elements	**248,763**
#BL elts	**174,834**

Figure 10. Boundary layer mesh for engine nacelle.

Other procedures can be employed in different situations. For example, there are a number of simulations where portions of the geometric domain are thin compared to the mesh size required in the other directions. However, a few elements in any direction through the model, including the thickness direction, are needed to obtain the desired solution accuracy. If the domain is uniformly thin, the desired anisotropic mesh can be obtained by repeating the surface mesh on one side to the other side one time for each required layer. If the model contains variations in the thickness, including possessing thick sections where an isotropic mesh is desired, an alternative approach is needed. One possibility is to (i) apply a mesh generator that produces an isotropic mesh through the domain, (ii) determine the thin portions of the domain, and (iii) apply a specially constructed set of local mesh modifications to produce the desired anisotropic mesh refinement [18]. Figure 11(a) shows a mesh created by an isotropic mesh generator in a geometric model with thin sections. Figure 11(b) shows the mesh with anisotropic refinement in the thin sections that was created by applying an appropriate set of local mesh modifications to the original isotropic mesh [18].

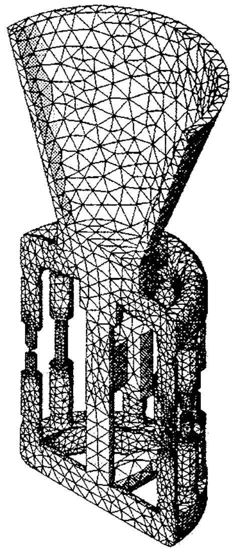
6. HEXAHEDRAL MESH GENERATION

The topology of the elements to be generated has a major influence on the type of mesh generation approach possible. Most of the literature in the area of automatic mesh generation has focused on the generation of simplex elements of triangles and tetrahedra because of their more fundamental geometric properties. Since many analysis procedures require, or prefer, quadrilaterals in two dimensions and hexahedron in three dimensions, there is a desire for automatic mesh generators that can effectively generate meshes of these element topologies.

In two dimensions it is reasonably straight forward to automatically generate all, or, if preferred, nearly all, quadrilateral meshes, while the automatic generation of hexahedral elements in three dimensions continues to be a challenge which has not yet been satisfactorily resolved. There is a fundamental difference between the properties of polyhedra decomposition in two and three dimensions which is at the heart of the difficulty. In two dimensions any n-sided, simply connected, convex polygon can be decomposed into n convex quadrilaterals by (i) introducing a point on the interior of the polygon, (ii) splitting each original edge of the polygon into two, and (iii) introducing edges between each of the edge split points and the point introduced on the interior (Fig. 12).

In three dimensions it is possible to decompose convex polyhedron with n-trivarient vertices into n convex hexahedra, where a trivarient vertex is defined as one with exactly three edges emanating from it. On such polyhedron, each face is subdivided as above, an additional point is introduced on the interior and edges are created from that point to the points introduced on each of the faces of the polyhedron (Fig. 13).

However, the same is not true if the polyhedra has non-trivarient vertices such as a pyramid. To match-up with the tetrahedron, wedge, and hexahedron shown in


(a) initial mesh

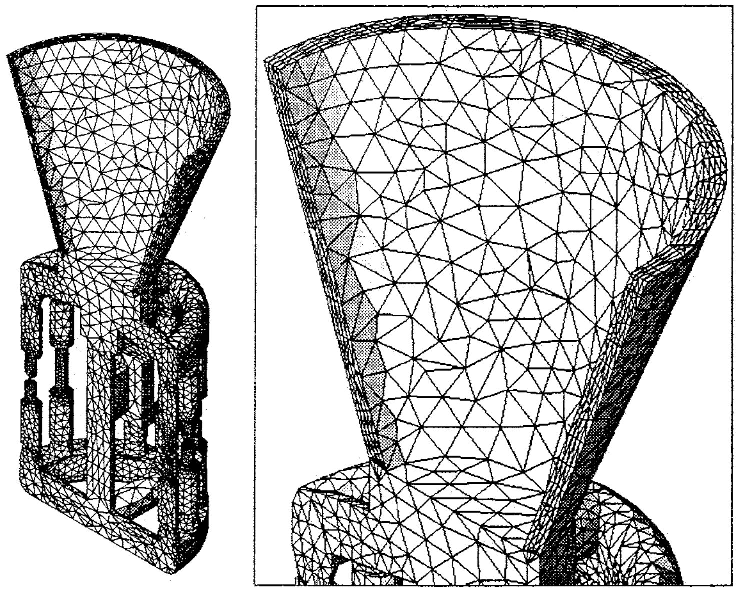

(b) multiple element through the thickness

Figure 11. The application of local mesh modifications to create anisotropic mesh refinement in thin sections.

Figure 13, the quadrilateralization of the surface would need to be as indicated in Figure 14(a). However, no method of generating a set of valid hexahedra with that surface

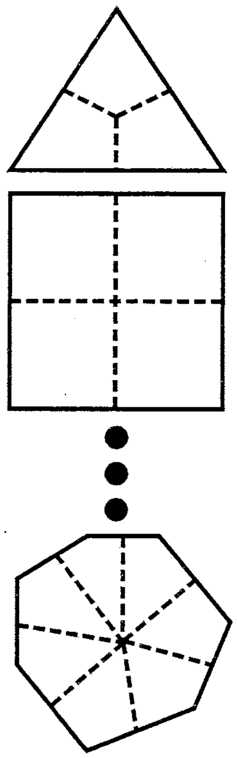


Figure 12. Decomposition of convex polygons into quadrilaterals through edge-splitting and connecting to an interior point.

quadrilateralization is known. On the other hand, the pyramid can be split into two tetrahedron, which can each be decomposed into four hexahedron using the trivarient polyhedra decomposition procedure (Fig. 14(b)). However, this is not acceptable since the six quadrilateral configuration resulting on the original quadrilateral face of the original pyramid will not match the desired four quadrilateral subdivision that is desired to match the previous subdivision. The only clear solution is to first decomposed the original domain into tetrahedra and subdivide each of them. However, the elements created in this manner are highly distorted.

Another manifestation of the difficulty of all hexahedral mesh generation is demonstrated by the construction of dual graphic of such meshes. The specific dual construction, referred to as the spatial twist continuum [35], clearly demonstrates that for a given quadrilateralization of the surface of an object, there must be specific paths through the mesh of hexahedra which share common faces, and that in fact, the constraints on the surface quadrilateralization are such that it is not in general possible to generate an all hexahedral mesh for just any given surface quadrilateralization.

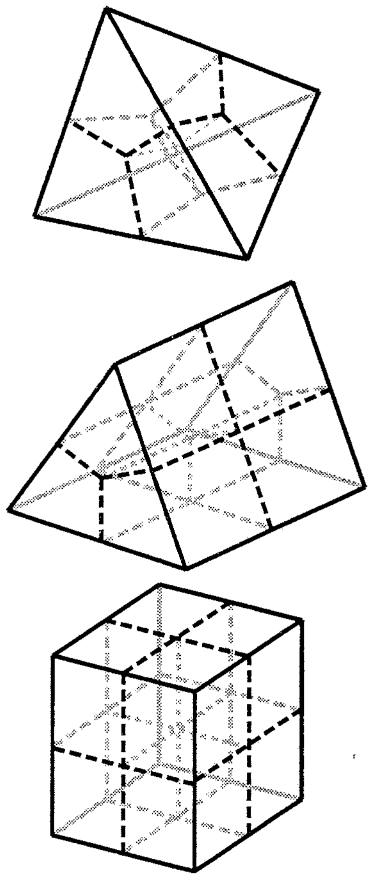


Figure 13. Decomposition of three common convex trivarient polyhedra into hexahedra.

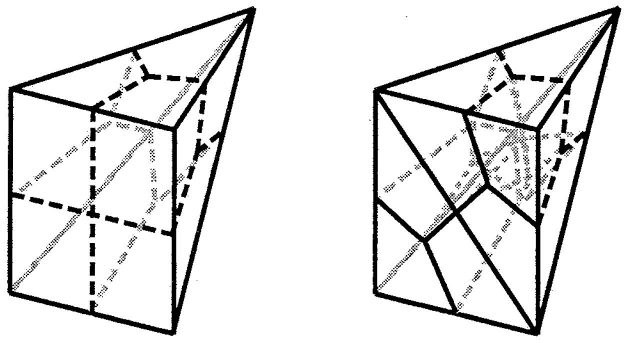

(a) unworkable surface quadrilateralization
(b) 2 tetrahedron to 8 hexahedron

Figure 14. Decomposition of a pyramid.

The approaches being taken to develop automatic hexahedral mesh generators are attempting to address the added constraints of hexahedral mesh generation by various means. Three alternative approaches being taken are:

1. Grid based techniques.
2. Medial axis with mapping in the subdomains.
3. Element removal with consideration of the constraints of the hexahedral meshes.

The basic concept of the grid based approach is to fill the interior of the domain with a uniform grid of hexahedra such that the exterior boundary of the cells is approximately a distance of one cell length from the boundary. The mesh is then completed by connecting the interior cells to the boundary in such a manner that only hexahedral elements are added. A key aspect of such an approach to creating an essentially uniform mesh, is ensuring that the interior cells can be connected to the boundary such that only hexahedra are introduced. Such a procedure has been developed [44] that deals with a specific set of situations that arise.

The basic assumption in the medial axis based methods is that the medial axis will decompose the domain of interest into simple enough regions that specific mapping templates can be applied to each resulting subregions in such a manner that hexahedral elements are formed. The medial surface of a three-dimensional object is the locus of the center of an inscribed sphere of maximal diameter as it rolls around the interior of the object. Maximal inscribed spheres are always in contact with two or more points on the boundary of the object. By its definition, the medial surface is always equidistance from multiple surface entities, defining, in some sense, a decomposition by middle surfaces. Figure 15 shows a simple two-dimensional object and its medial axis. Issues that have to be addressed in the development of this hexahedral mesh generation approach include:

1. Generation of a medial axis of the domain.
2. Conversion of the medial axis information into a set of regions which can be filled with hexahedra by a mapping procedure.
3. Ensuring that the mapped meshes of each of the generated regions can be properly matched.

A key unknown in the development of such approaches is the second step of being able to convert the medial axis information for general domains into a set of regions that can be filled with a valid set of hexahedra of acceptable shape and configuration to match to neighboring regions. References [41, 42] discuss some of the issues and progress made on the development of these approaches.

The application of element removal procedures in the generation of hexahedral elements is far more complex than that of tetrahedral mesh generation due to the constraints of

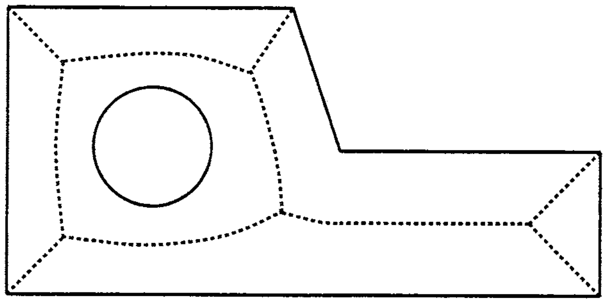


Figure 15. Two-dimensional example showing its medial axis.

hexahedral meshes. One tool introduced to deal with some of the complications that arise are wedges and seams [6] which close-up gaps in some cases, and drive problem situations to the boundary, altering the faces quadrilateralization to eliminate the problem. Even with these tools, deriving a set of heuristics to apply hexahedral element removal using only local information has proven difficult. To more directly address the constraints of hexahedral mesh generation, specific consideration on how to employ the dual representation of the spatial twist continuum [35] has received attention. Since a surface quadrilateralization already dictates a specific set of constraints on the mesh, procedures employing the spatial twist continuum information already dictated by the surface quadrilaterals are under consideration [5, 54].

7. PARALLEL MESH GENERATION

Scalable parallel processing techniques are becoming central to the solution of large scale simulations consisting of tens of millions of finite elements. Therefore consideration must be given to the parallel generation of meshes to ensure mesh generation does not become a serial computational bottleneck. Another reason for parallel mesh generation is the shortage of memory on a single sequential machine when dealing with very large meshes. On a parallel machine, the memory problem is addressed by distributing the mesh over a number of processors, each of which stores its own portion of the mesh.

Efficient parallel algorithms require a balance of work load among the processors while maintaining interprocessor communication at a minimum. Key to determining and distributing the work load and controlling communications is knowledge of the structure of the calculations and communications. Parallel mesh generation is difficult to effectively control since the only structure known at the start of the process is that of the geometric model which has no discernible relationship to the work load needed to generate the mesh. On the other hand, the more useful structure to discern work load and control communications is the mesh which is only fully known at the end of the process.

Löhner, et al. [31] have parallelized a two-dimensional advancing front procedure. The approach taken is to subdivide (partition) the domain (with the help of a background grid) and distribute the sub-domains to different processors for triangulation. The interior of subdomains are meshed independently. Then, the inter-subdomain regions are meshed using a coloring technique to avoid conflicts. Finally, the "corners" between more than two processors are meshed following the same basic strategy. A "one master-many slaves" paradigm has been chosen to drive the parallel procedures. This approach has been extended to three dimensions with some modifications [53]. A load balancing phase follows the initial domain splitting (at the background grid level). The interface gridding incorporates mechanisms (i) to avoid degradation of performance by using fine grain parallelism, and (ii) to reduce the number of processors when there is too much communication overhead. Results show scalability of the method.

Saxena and Perruchio [43] describe a parallel Recursive Spatial Decomposition (RSD) scheme which discretizes the model into a set of octree cells. Interior and boundary cells are meshed by either templates or element removal in parallel at the octant level meshes. This requires no communication between octants. The main difficulty for this meshing approach is to guarantee that a boundary octant can always be meshed regardless of the complexity of the model. Parallel results have been simulated on a sequential machine.

An octree-based parallel three-dimensional mesh generator has been developed by de Cougny, et al. [14, 15] in which the parallel computations are controlled through the use of the octree structure. The lack of initial structure and ability to accurately predict work load during the meshing process underlies the selection of algorithmic procedures in this parallel mesh generator. The octree structure supports the distribution and redistribution of computational effort to processors.

In this procedure the surface triangulation is first generated on each of the model faces using a Delaunay-based mesh generator. When there are not a sufficient number of model faces for effective parallelization based on the number of processors being employed, faces are split with the portions meshed on separate processors.

Figure 16 gives a two-dimensional graphical depiction of the region meshing portion of this mesh generator. First a variable level octree which is consistent with the surface triangulation, and other mesh control information, is generated. The octants are classified as interior, outside, or boundary. Those classified as outside receive no further consideration. Interior octants too close to the surface triangulation are reclassified boundary. This reclassification avoids the complexities that may lead to poorly shaped elements in that neighborhood. Interior octants are meshed using templates. Face removal procedures are used to connect the boundary triangulation to the interior octants.

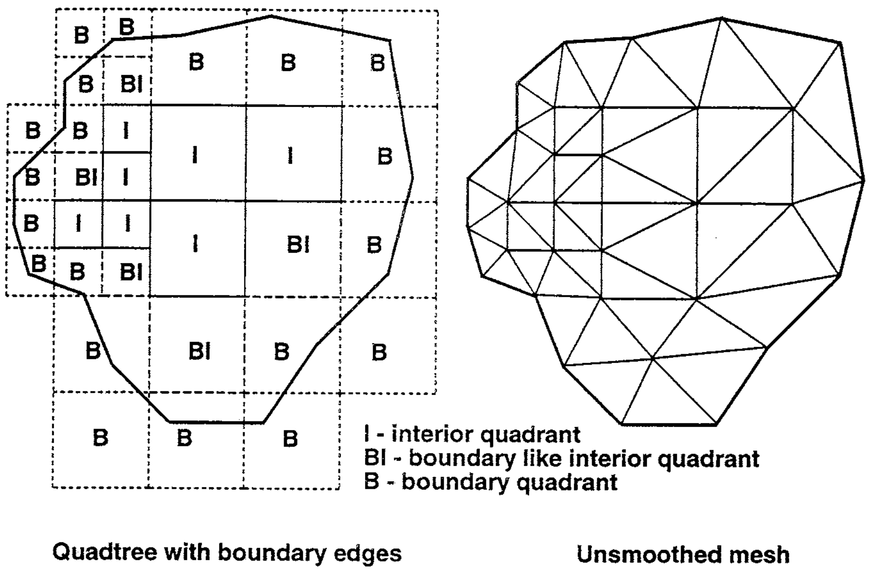

I - interior quadrant
BI - boundary like interior quadrant
B - boundary quadrant
Quadtree with boundary edges
Unsmoothed mesh

Figure 16. Graphical depiction of the basics of the presented mesh generator.

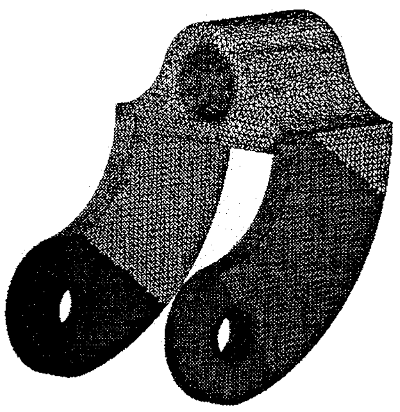


Figure 17. Final repartitioned mesh for parallel mesh generation example (the gray scale indicates which of the 8 processors contain the mesh entities).

The key to parallelization of the meshing process is a parallel distributed octree structure and maintaining load balance during the meshing process by the repartitioning of the portions of the domain remaining to be meshed. Dynamic repartitioning enables redistribution of the load among processors as evenly as possible at key stages of the mesh generation process [12, 49]. Figure 17 shows a mesh generated in parallel on 8 processors. The speed-up on the 8 processors was just over 6.

8. CLOSING REMARKS

This paper has provided an update to the 1988 review article on the automatic generation and control of finite element meshes. In the past seven years there has been substantial progress on the development of automatic mesh generation capabilities for three-dimensional domains. Today there are a number of procedures available to automatically generate such meshes when tetrahedral elements are used. Although automatic mesh generators for hexahedral elements are not yet fully developed, investigations on the development of such procedures is underway. Procedures to automatically generate meshes on parallel computers have been developed.

9. ACKNOWLEDGMENTS

The author would like to acknowledge the NASA Ames Research Center, under grants NAG 2–832 and NCC 2–9000, the National Science Foundation under grant DMS-93–18184, and the SCOREC Industrial Affiliates for their support of our work on automatic mesh generation and its parallelization.

The input of Hugues L. de Cougny, Saikat Dey, Pascal Frey, Rao Garimella to material presented in this document is gratefully acknowledged.

10. REFERENCES

[1] In *Proceedings of the 4th International Meshing Roundtable*, Albuquerque, NM, 1995. Sandia National Laboratory.

[2] C. G. Armstrong, editor. *Advances in Engng. Software*, volume 13:5/6. Computational Mechanics Publ., Southhampton, England, 1991.

[3] I. Babuska and A. K. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Anal.*, 13(12), 1976.

[4] T. J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5:161–175, 1989.

[5] T. Blacker, S. A. Mitchell, T. J. Tautges, P. Murdoch, and S. Benzley. Forming and resolving wedges in the spatial twist continuum. *Engineering with Computers*, 1996. to appear.

[6] T. D. Blacker and M. B. Stephenson. Seams and wedges in plastering: A 3D hexahedral mesh generation algorithm. *Engineering with Computers*, 9:83–93, 1993.

[7] H. Borouchaki, F. Hecht, E. Saltel, and P. L. George. Reasonably efficient Delaunay based mesh generation in 3 dimensions. In *Proceedings of the 4th International Meshing Roundtable*, pages 3–14, Albuquerque, NM, 1995. Sandia National Laboratory.

[8] H. Borouchaki and S. H. Lo. Fast Delaunay triangulation in three dimensions. *Comp. Meth. Appl. Mech. Engng.*, 39:493–500, 1991.

[9] E. K. Buratynski. A fully automatic three-dimensional mesh generator for complex geometries. *Int. J. Numer. Meth. Engng.*, 30:931–952, 1990.

[10]J. C. Cavendish, D. A. Field, and W. H. Frey. An approach to automatic three-dimensional mesh generation. *Int. J. Numer. Meth. Engng.*, 21:329–347, 1985.

[11]S. D. Connell and M. E. Braaten. Semistructured mesh generation for three-dimensional navier-stokes calculations. *AIAA J.*, 33(6):1017–1024, 1974.

[12]H. L. de Cougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Ozturan, and M. S. Shephard. Load balancing for the parallel solution of partial differential equations. *Applied Numerical Mathematics*, 16:157–182, 1994.

[13]H. L. de Cougny and M. S. Shephard. Parallel mesh adaptation by local mesh modification. Technical Report 21-1995, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1995. submitted to Computer Methods in Applied Mechanics and Engineering.

[14]H. L. de Cougny, M. S. Shephard, and C. Ozturan. Parallel three-dimensional mesh generation. *Computing Systems in Engineering*, 5(4-6):311–323, 1994.

[15]H. L. de Cougny, M. S. Shephard, and C. Ozturan. Parallel three-dimensional mesh generation on distributed memory MIMD computers. *Eng. with Computers*, 1995. to appear.

[16]B. E. de l'Isle and P. L. George. Optimization of tetrahedral meshes. In I. Babuska, J. E. Flaherty, J. E. Hopcroft, W. D. Henshaw, J. E. Oliger, and T. Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, pages 97–128. Springer-Verlag, New York, 1995.

[17]S. Dey, M. S. Shephard, and M. K. Georges. Elimination of the adverse effects of small model features by the local modification of automatically generated meshes. Technical Report 30-1995, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1995. submitted to Engineering with Computers.

[18]R. Garimella and M. S. Shephard. Tetrahedral mesh generation with multiple elements through the thickness. In *Proceedings of the 4th International Meshing Roundtable*, pages 321–333, Albuquerque, NM, 1995. Sandia National Laboratory.

[19]P. L. George. *Automatic Mesh Generation*. John Wiley and Sons, Ltd, Chichester, 1991.

[20]P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundaries. *Comp. Meth. Appl. Mech. Engng.*, 92:269–288, 1991.

[21]M. K. Georges and R. Ramamoorthy. Geometric operators for the Finite Octree mesh generator. Technical Report SCOREC Report # 13-1991, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1990.

[22]A. Habraken and S. Cescotto. An automatic remeshing technique for finite element simulation of forming processes. *Int. J. Numer. Meth. Engng.*, 30:1503–1525, 1990.

[23]H. Jin and R. I. Tanner. Generation of unstructured tetrahedra by advancing front technique. *Int. J. Numer. Meth. Engng.*, 36:1805–1823, 1993.

[24]B. Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comp.*, 10(4):718–741, 1989.

[25]B. Joe. Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation. *Int. J. Numer. Meth. Engng.*, 31:987–997, 1991.

[26]A. Kela. Exact octree approximations from geometric (csg/brep) models: Their derivation and application in finite element mesh generation. *submitted to IEEE Comp. Graphics and Applications*, 1989.

[27]S. R. Kennon and D. A. Aderson. Unstructured grid generation for non-convex domains. In S. Sengupta, J. Hauser, P. R. Eiseman, and J. F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Mechanics'88*, pages 599–609. Pineridge Press, 1988.

[28]A. Khawaja, H. McMorris, and Y. Kallinderis. Hybrid grids for viscous flows around complex 3-D geometries including multiple bodies. Technical Report AIAA-95-1685-CP, Washington, D.C., 1995.

[29]C. L. Lawson. Properties of n-dimensional triangulations. *Computer Aided Geometric Design*, 3(4):231–246, 1986.

[30]S. H. Lo. Volume discretization into tetrahedron - II. 3D triangulation by advancing front approach. *Computers and Structures*, 39:493–500, 1991.

[31]R. Löhner, J. Camberos, and M. Merriam. Parallel unstructured grid generation. *Comp. Meth. Appl. Mech. Engng.*, 95:343–357, 1992.

[32]R. Löhner and P. Parilch. Three-dimensional grid generation by the advancing front method. *Int. J. Num. Meths. Fluids*, 8:1135–1149, 1988.

[33]D. L. Marcum, N. P. Weatherill, M. J. Marchant, and F. Beaven. Adaptive unstructured grid generation for viscous flow applications. Technical Report AIAA-95-1726-CP, Washington, D.C., 1995.

[34]P. Moller and P. Hansbo. On advancing front mesh generation in three dimensions. *Int. J. Numer. Meth. Engng.*, 38:3551–3569, 1995.

[35]P. Murdoch, S. Benzley, T. Blacker, and S. A. Mitchell. The spatial twist continuum: A connectivity based method for representing all hexahedral finite element meshes. *Int. J. Numer. Meth. Engng.*, 1996. submitted.

[36]V. N. Partasarathy and S. Kodiyalam. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design*, 1992.

[37]J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz. Finite element Euler computations in three dimensions. *Int. J. Numer. Meth. Engng.*, 26:2135–2159, 1988.

[38]J. Peraire, J. Peiro, and K. Morgan. Adaptive remeshing for three-dimensional compressible flow computations. *J. Comp. Phys.*, 103:269–285, 1992.

[39]R. Perucchio, M. Saxena, and A. Kela. Automatic mesh generation from solid models based on recursive spatial decompositions. *Int. J. Numer. Meth. Engng.*, 28:2469–2501, 1989.

[40]S. Pirzadeh. Viscous unstructured three-dimensional grids by advancing layer method. Technical Report AIAA-94-0417, Washington, D.C., 1994.

[41]M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: II. solids with flat and concave edges. *Int. J. Numer. Meth. Engng.*, 1996.

[42]M. A. Price, C. G. Armstrong, and M. A. Sabin. Hexahedral mesh generation by medial surface subdivision: I. solids with convex edges. *Int. J. Numer. Meth. Engng.*, 1996.

[43]M. Saxena and R. Perucchio. Parallel fem algorithms based on recursive spatial decompositions - I. automatic mesh generation. *Computers and Structures*, 45:817–831, 1992.

[44]R. Schneiders and R. Bunten. Automatic mesh generation of hexahedral finite element meshes. *Computer Aided Geometric Design*, 12:693–707, 1995.

[45]W. J. Schroeder and M. S. Shephard. A combined octree/Delaunay method for fully automatic 3-D mesh generation. *Int. J. Numer. Meth. Engng.*, 29:37–55, 1990.

[46]W. J. Schroeder and M. S. Shephard. On rigorous conditions for automatically generated finite element meshes. In J. Turner, J. Pegna, and M. Wozny, editors, *Product Modeling for Computer-Aided Design and Manufacturing*, pages 267–281. North Holland, 1991.

[47]M. S. Shephard. Approaches to the automatic generation and control of finite element meshes. *Applied Mechanics Review*, 41(4):169–185, 1988.

[48]M. S. Shephard, editor. *Engineering with Computers*, volume 5. Springer-Verlag, New York, 1989.

[49]M. S. Shephard, J. E. Flaherty, H. L. de Cougny, C. Ozturan, C. L. Bottasso, and M. W. Beall. Parallel automated adaptive procedures for unstructured meshes. In *Parallel Computing in CFD*, volume R-807, pages 6.1–6.49. AGARD, Neuilly-Sur-Seine, France, 1995.

[50]M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generation by the Finite Octree technique. *Int. J. Numer. Meth. Engng.*, 32(4):709–749, 1991.

[51]M. S. Shephard and M. K. Georges. Reliability of automatic 3-D mesh generation. *Comp. Meth. Appl. Mech. Engng.*, 101:443–462, 1992.

[52]M. S. Shephard and N. P. Weatherill, editors. *Int. J. Numer. Meth. Engng.*, volume 32. Wiley-Interscience, Chichester, England, 1991.

[53]A. Shostko and R. Löhner. Three-dimensional parallel unstructured grid generation. *Int. J. Numer. Meth. Engng.*, 38:905–925, 1995.

[54]T. J. Tautges, T. Blacker, and S. A. Mitchell. The whisker weaving algorithm: A connectivity based method for constructing all-hexahedral finite element meshes. *Int. J. Numer. Meth. Engng.*, 1996. submitted.

[55]N. P. Weatherill and O. Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *Int. J. Numer. Meth. Engng.*, 37:2005–2039, 1994.