SAND2000-1355J

HEX-DOMINANT MESH GENERATION USING 3D CONSTRAINED TRIANGULATION

Steven J. Owen

Sandia National Laboratories
P.O. Box 5800. M.S. 0847
Albuquerque, NM 87185
Phone: 505-284-6599
sjowen@sandia.gov

ABSTRACT

A method for decomposing a volume with a prescribed quadrilateral surface mesh, into a hexahedral-dominated mesh is proposed. With this method, known as *Hex-Morphing* (H-Morph), an initial tetrahedral mesh is provided. Tetrahedra are transformed and combined starting from the boundary and working towards the interior of the volume. The quadrilateral faces of the hexahedra are treated as internal surfaces, which can be recovered using constrained triangulation techniques. Implementation details of the edge and face recovery process are included. Examples and performance of the H-Morph algorithm are also presented.

Keywords: constrained triangulation, hexahedral-dominant, mesh generation

1. INTRODUCTION

Various techniques have been presented in the literature for decomposing a volume into hexahedral elements. Reference [1] provides an overview of many of these techniques. Most of these methods are applicable only for certain classes of geometry or do not provide sufficient control over the boundary quadrilaterals. Where a prescribed quadrilateral surface mesh is required with an arbitrary geometry, the applicability of most of these techniques is limited. The *Hex-Morphing* (H-Morph) technique, described in this work, is designed to address these issues.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

The H-Morph technique uses some of the fundamental algorithms used for constrained triangulation or *boundary recovery*. Given an initial set of simplices in R^3 comprised of nodes, edges and faces, it is frequently necessary to modify the arrangement of the edges and faces to conform to a particular pattern. For instance, three-dimensional domains are often decomposed into tetrahedra for purposes of developing a computational mesh for finite element analysis. Faces and edges must conform to the boundary of the domain. Where Delaunay mesh generation techniques[2,3] are employed, the tetrahedra can be generated by inserting nodes into the domain and locally retriangulating so as to adhere to the Delaunay criterion. This process generally does not take into account the boundary of the domain, permitting faces and edges to position themselves based only on the empty-sphere property[4]. As a consequence, a separate procedure for transforming the tetrahedra, known as *boundary recovery*, is employed so that resulting faces and edges are constrained to lie on the surface. The resulting mesh is often referred to as a *boundary-constrained* Delaunay mesh, since the process of recovering the boundary may not allow the Delaunay criterion to be satisfied.

Although application for constrained triangulation has long been Delaunay mesh generation, constrained triangulation can equally well be applied where internal boundaries must be honored. In situations where material properties may change within a computational domain, the boundary between materials may need to be enforced. Where the material boundaries are introduced after the initial tetrahedra have been placed, tetrahedral transformations may be applied locally to the simplices.

Another application for constrained triangulation is introduced in this work. With the H-Morph technique, an initial tetrahedral mesh is provided. The quadrilateral faces of the hexahedra are treated as internal surfaces, which must be recovered. Local transformations are performed on the simplices so that the six quadrilateral faces of a hexahedron are recovered from the mesh. All tetrahedra internal to the six faces can subsequently be eliminated and a hexahedron formed in their place

2. OVERVIEW OF H-MORPH

The following discussion, briefly outlines the basic steps involved in the H-Morph technique. A more detailed discussion may be found in References [5] and [6]. With this technique, the surface of the domain is first meshed with quadrilaterals. This can be done using any quadrilateral surface meshing technique. Using a method such as Q-Morph[7] or Paving[8], with its characteristic well-aligned rows of elements, can improve the overall quality of the resulting hexahedra. A quadrilateral mesh may also be extracted from an adjoining volume, where multiple volumes may have been used in an assembly model. With the surface mesh in place, each quadrilateral is split into two triangles and the resulting facets passed to a tetrahedral mesh generator. Any boundary conforming tetrahedral mesh generation scheme [9] may be used, provided the density of internal points roughly matches that of the intended hexahedral mesh. Internal node locations of the tetrahedral mesh need not correspond with precise node locations of the final hexahedral mesh, as subsequent operations will add, delete and adjust node locations. The boundary conforming tetrahedral mesh generator produces a mesh where pairs of triangular surface facets serve as the initial set of fronts for the advancing front hexahedral mesh generation algorithm [10] to be used in the H-Morph process.

The advancing front procedure involves processing individual pairs of triangles by forming hexahedra from the tetrahedra immediately adjacent. Transformations are performed on the tetrahedra in an attempt to provide a local topology matching that of a hexahedron. Once this is accomplished, the local tetrahedra can be deleted and a hexahedron formed in its place. The formation of each new hexahedron, in turn produces new triangle pairs that can themselves be used as the basis for new hexahedra. This technique continues until hexahedra from opposing fronts run into each other or reasonable quality hexahedra can no longer be formed. During this process, smoothing and clean-up techniques [11] are employed on the tetrahedra, to maintain and improve the tetrahedral element quality from which the hexahedra will be formed. Local smoothing operations are also employed on the nodes of the newly constructed hexahedra to improve hexahedra element quality. In cases where tetrahedra remain after the H-Morph process is complete, it is possible to use pyramid elements to maintain compatibility between hexahedra and tetrahedra. Reference [12] provides a description of the pyramid formation algorithm, where pyramids are formed using local tetrahedral transformations.

Figure 1 shows an outline of the process for transforming tetrahedra to hexahedra. Beginning with a triangle pair ABCD, shown in Figure 1 (a), the faces and edges of a hexahedron are essentially carved from the existing tetrahedral mesh. Figure 1 (b) shows an edge, BE, that is selected from the existing topology. Local tetrahedral swaps may be necessary to define this edge. Figure 1 (c) - (d) show the process of recovering individual edges and faces from the tetrahedra. Finally, internal tetrahedra may be deleted and local smoothing techniques employed to improve element quality, as shown in Figure 1 (f).

3. TETRAHEDRAL TRANSFORMATIONS

Throughout the H-Morph process, various transformations must be made to the tetrahedra. These transformations include rearrangement of the local topology in order to arrive at a condition that will better admit a topology to form hexahedra. Some of these transformations have been described in other literature [11,13]. A summary of the transformations used in the boundary recovery process of the H-Morph algorithm are shown in Table 1, where the initial and final tetrahedra are described in terms of their vertices. Application of these transformations will be discussed in subsequent sections of this work.

The *swap23*, *swap22* and *face-split* operations begin with two tetrahedra sharing a single face. Table 1 illustrates how the transformed topology results in three, two and six tetrahedra respectively.

Table 1 also illustrates the *edge-split* and *edge-suppress* operations where the initial configuration is the set of tetrahedra sharing a single internal edge in the mesh, sometimes referred to as a *shell*. For the *edge-split* operation, where the shell contains N tetrahedra, the resulting configuration will have $2N$ tetrahedra. The *edge-suppress* operation attempts to eliminate the edge *ab* by retriangulating the space occupied by the N tetrahedra of the shell. This can be accomplished by triangulating the non-planar polygon $P = \{n_1, n_2, \ldots n_N\}$ and forming tetrahedra $n_k n_j n_i a$ and $n_i n_j n_k b$ for each triangle in polygon P, where n_i, n_j, n_k are vertices in P. The simplest form of the *edge-suppression* operation when, N=3, is the *swap32* operation, the inverse of a *swap23*. For most of the operations described in Table 1, a check must first be made to ensure that the resulting configuration will result in tetrahedra with positive volume.

An additional common operation, which does not involve a change in topology, is the *node-relocation* operation. This involves simply repositioning a single internal node in the mesh. This operation also requires local checks to ensure all neighboring tetrahedra do not become inverted. Where tetrahedra would otherwise become inverted, the node can be incrementally relocated to a position on the vector between the old and new location until all tetrahedra maintain a positive volume.

4. RECOVERY

Various approaches to the edge and face recovery process have been proposed, most notably that of Weatherill and Hassan[14] and George and Bourachaki [9,13]. These are invaluable resources and serve as the basis for much of the boundary recovery process as implemented in this research. Within the context of the H-Morph algorithm, the edge and face recovery process is used for delimiting each individual edge and face of the hexahedra. Since the recovery process is applicable only for triangular facets, two facets per hexahedral face must be recovered.

The two-dimensional edge recovery process, described in the context of the Q-Morph algorithm[7], can be effectively modified for the three-dimensional condition. Unlike the two-dimensional case, the additional process of recovering a face is required. This is by virtue of the fact that even after enforcing the edges of a triangle facet in the tetrahedral mesh, there is no guarantee that a face will occupy the plane inside the edges. It is conceivable that any number of edges and faces may penetrate the plane of the face under consideration. The face recovery process is a set of procedures for eliminating these penetrating faces and edges to recover the triangle facet between three existing edges.

4.1 Edge Recovery

It is convenient to consider the edge and facet recovery procedures separately. First considered is edge recovery. The objective of the edge recovery is essentially to recover the top edge and diagonal edge of the quadrilateral as illustrated by the dashed lines in Figure 2. The quadrilateral to be recovered will be described by nodes $n_1n_2n_3n_4$. Algorithm 1 is a description of the edge recovery process. A *pipe, $\Lambda(S)$*, is first constructed containing the set of all triangle faces through which the intended edge passes. Figure 3 shows an example pipe through which segment *S* passes from node A to B. Also shown is an exploded view of the tetrahedra involved in the pipe. After forming $\Lambda(S)$, each face is processed by attempting to perform the *swap23* operation, described in Table 1. For each face, a set of three tetrahedra $T^{-1}(F_i)$ can be defined, resulting from the *swap23*. If the volume of any tetrahedra in $T^{-1}(F_i) < 0$, then the face, F_i is placed back on the queue, $\Lambda(S)$ to be processed again later.

Unlike the two-dimensional case that is guaranteed to recover the edge, the three-dimensional case has no such guarantee. When any single face, F_i, fails the *swap23* operation, George and Borouchaki [13] propose the insertion of one or more nodes into the mesh to resolve the issue. This proposed resolution, although implemented and tested in the current research, in practice, resulted in an excessive number of new nodes introduced into the mesh. Additional nodes result in disproportionate numbers of tetrahedra, generally of low quality, on the interior of the mesh, tending to over-constrain the problem. In the traditional boundary recovery problem, the nodes are constrained to be stationary. Conversely, in the H-Morph process, since the node locations on the interior of the mesh can be readjusted, the edge recovery process can take advantage of this characteristic. As a result, after an initial *n* attempts to swap any face F_i in the pipe, the node locations for faces remaining in the pipe not lying on the current front are adjusted or smoothed. The smoothing operation is simply Laplacian [15], which readjusts the node locations to the centroid of the nodes attached by an edge. Once the node locations are adjusted, the edge recovery algorithm can be restarted, forming a new pipe, $\Lambda(S)$ that must be reduced in the same manner.

In cases where the smooth operation fails to reduce $\Lambda(S)$, the edge and face recovery process is aborted. Aborting the edge and face recovery process simply indicates that the current quadrilateral face under construction could not be completed with the current topology. When this occurs, the construction of the current hexahedron is also aborted and the front placed back on a list to be processed later. In most cases, when the same front is selected for processing again, the local topology will have changed enough as to result in a successful edge and face recovery. A limited number of failures for any given front are permitted. After exceeding the permissible number of failures, the front is simply left with its adjacent tetrahedra, to be processed later for insertion of pyramid elements.

Algorithm 2 is an illustration of the procedure used for defining the pipe, $\Lambda(S)$, referred to in step 2 of Algorithm 1. The procedure begins at node A and marches through the local topology along segment *S*, collecting faces in the pipe as it goes, until finally arriving at node B where it terminates.

The first phase of Algorithm 2 is an attempt to determine through which of node A's adjacent tetrahedra, segment *S* passes. Let F_j be the set of 3 faces on tetrahedron T_i that have node A as a vertex, and V_j be their corresponding outward pointing normalized normal vectors. If V_S is the vector defined by *S*, then if $\mathbf{V}_S \cdot \mathbf{V}_j < 0$ for all 3 faces, F_j, then T_i contains segment *S*. This case is illustrated in Figure 4. Once the appropriate tetrahedron T_i is identified through which segment *S* passes, the opposite face on the tetrahedron, F_i from A is determined and passed to the next phase of the algorithm.

The algorithm continues by adding face F_i to the pipe, *Λ(S)* and the next tetrahedron, T_{i+1}, adjacent F_i is determined. The next candidate face for the pipe will be one of the three faces, F_j on T_{i+1}, where $F_i \neq F_j$. Algorithm 2 proposes a straightforward and efficient manner for determining which of the three faces, $F_{i+1} = F_j$, segment *S* intersects. Once face F_{i+1} is determined, it is added to *Λ(S)*. Setting $F_i = F_{i+1}$ and $T_i = T_{i+1}$, the process loops until arriving at node B.

Algorithm 2 also describes the case where $\mathbf{V}_S \cdot \mathbf{V}_j = 0$ for any one face, F_j, indicating that segment *S* is coplanar with F_j. In the case where $\mathbf{V}_S \cdot \mathbf{V}_j = 0$ for two faces, F_j, segment *S* is colinear with the edge shared by the two faces, F_j where $\mathbf{V}_S \cdot \mathbf{V}_j = 0$. It should be noted that in Algorithm 2, when segment *S* directly intersects either an edge or a node $\neq$ B, rather than directly handling the case, an attempt is made to modify the local topology. When *S* intersects an edge, the *edge-suppression* operation described in Table 1 is first attempted. If this is not successful, then the end nodes of the intersected edge are smoothed and repositioned using the *node-relocation* operation. When *S* intersects a node, the node is simply moved out of the way, again using smoothing and *node-relocation.* In the event the local topology is changed as a result of an edge or node intersection with *S*, Algorithm 2 must be restarted. On occasions, there are cases where the edge-suppression and smoothing operations will fail. When this occurs, the edge/face recovery procedure fails with the same consequences as addressed previously.

4.2 Face Recovery

Once the required edges have been recovered in the tetrahedral mesh, the two triangular facets that will form the quadrilateral face of the hexahedron must be recovered. Similar to the edge recovery process, this is done by a series of tetrahedral transformations.

Figure 5 is an example of a series of edges, E_0,E_1,E_2, that have been recovered, where a triangular facet, F_r, defined by nodes N_0,N_1,N_2, does not exist in the mesh. In this example, two edges, $\alpha_0\beta_0$ and $\alpha_1\beta_1$, *penetrate* F_r. The set of tetrahedral elements surrounding a single edge in the mesh is referred to as a *shell*. If E is an edge in the tetrahedral mesh, then *shell*(E) is the set of all tetrahedra sharing the common edge, E. George and Borouchaki [13] refer to the set of shells whose defining edge penetrates F_r as a *pebble*. The objective of the face recovery process is to eliminate the pebble, resulting in the recovered facet, F_r.

Elimination of the pebble involves, first determining the edges, E_p which penetrate F_r, and second, transforming the tetrahedra in *shell*(E_p) to eliminate E_p. Let E_i, $\{i = 0,1,2\}$ be the set of edges on F_r and N_i be the corresponding set of nodes on F_r where N_i is opposite E_i as in Figure 6. The facet recovery algorithm proceeds by interrogating the tetrahedra in each *shell*(E_i), to locate a penetrating edge, E_p. The process continues, locating and eliminating E_p until all edges, E_p, have been eradicated, consequently recovering facet F_r.

Algorithm 3 defines the method used for detecting a penetrating edge E_p, with Figure 6 illustrating the local topology involved. Line 9 of Algorithm 3 addresses the case where a node, N_{opp} lies exactly on the plane of F_r. If this occurs, N_{opp} can simply be perturbed by a small amount, ε out of the plane. Once the perturbation has been accomplished, the determination of E_p must be restarted, as the local face normal vectors will have changed.

Once determination of the penetrating edge has been accomplished, in order to eliminate edge E_p, the *edge-suppress* operation described in Table 1 is used. For the simple case where there are exactly three tetrahedra in *shell*(E_p), the *edge-suppression* involves a *swap32* operation, the simplest form of *edge-suppression*. This is the inverse of the *swap23* operation previously mentioned and also described in Table 1. If the pebble consists of a single shell containing exactly three tetrahedra, the *swap32* operation is guaranteed to be successful, resulting in the recovery of the facet F_r. Where there exists more than one shell in the pebble or where the number of tetrahedra in *shell*(Ep) is greater than three, there is no such guarantee that the edge suppression operation will be effective. The Current implementation of *edge-suppression* provides for cases up to a maximum of seven elements adjacent E_p. The *edge-suppression* attempts to determine an alternate configuration of the tetrahedra in *shell*(E_p) where the volume of all resulting tetrahedra is positive. If no such configuration exists, the *edge-suppression* operation fails.

Upon failing the *edge-suppression*, an alternative method is used to attempt to reduce the number of tetrahedra in *shell*(E_p). Frequently if the number of tetrahedra at E_p can be reduced, the *edge-suppress* operation will be successful. This reduction can be done by performing a *swap23* operation on tetrahedra that share a common face on *shell*(E_p). Each *swap23* operation performed, effectively reduces the number of tetrahedra surrounding E_p by one. The reduction of a shell must be done judiciously, in order to maintain reasonably shaped tetrahedra. Each set of two adjacent tetrahedra in *shell*(E_p) is examined and the quality of the potential three tetrahedra formed from a *swap23* operation is determined. The set, which results in the greatest minimum quality for any single potential tetrahedra, is selected to perform the *swap23* operation.

On occasion, where the reduction can be performed on a set of tetrahedra whose faces form a single quadrilateral face on the front, the *swap22* operation, described in Table 1 can be used instead. This results in the diagonal edge at the front being swapped. In traditional face recovery applications, swapping a boundary edge would not be permitted, as all edges at the boundary would be considered, *constrained.* In the context of H-Morph, this is a harmless operation, as the diagonal edges on the quadrilateral faces are free to be swapped as needed.

Once the *shell*(E_p) has been reduced, the *edge-suppress* operation can once again be invoked. The *reduction* and *edge-sppress* operations can be iteratively effected until either the edge E_p has been eliminated or *shell*(E_p) can not be further reduced. Figure 7 shows a simple example where failure to reduce the shell can occur. Although the penetrating edge, E_p, has only three surrounding tetrahedra, a *swap32* operation cannot be invoked without inverting one or more elements.

George and Borouchaki [13] refer to the configuration in Figure 7 as a *firtree*. In the case where an irreducible shell is encountered, rather than inserting additional nodes, the nodes on the pebble are first smoothed. This smoothing operation often results in the successful execution of the face recovery process without having to insert additional nodes. In the rare occasions where the smoothing still fails to effect the face recovery, the current front is aborted and placed at the back of its state list to be processed later. Frequently, a subsequent operation will change the local topology enough that the resulting face recovery will be successful.

As illustrated by the preceding pages, the edge and face recovery operations can be reasonably complex. They are, however, the key to the success of the H-Morph algorithm. Once two triangle facets have been recovered from the tetrahedra, a quadrilateral can be formed and a new front defined. This process is repeated for each of the quadrilateral faces of the hexahedron.

5. EXAMPLES

A number of examples were solved using the H-Morph procedure to demonstrate its validity in generating hexahedral-dominant meshes. A simple blocky-type configuration as shown in Figure 8 is first considered. This type of model is most often handled using a mapped meshing technique [16] after manually decomposing the geometry into *mappable* regions. An automatic geometry decomposition technique [17,18] or the sub-mapping [19] method can also be used with this class of geometry. The H-Morph algorithm is able to mesh this geometry without the need for decompostion. Figure 8(a) shows the initial boundary constrained tetrahedral mesh where a total of 4197 tetrahedra have been generated. Figure 8(b) and Figure 8(c) show intermediate stages of the H-Morph algorithm, where the tetrahedra are systematically converted into hexahedra. Finally, Figure 8(d) shows the completed mesh with a total of 756 hexahedral elements. The H-Morph algorithm is able to resolve this model completely into regular hexahedral elements.

Figure 9 is a non-trivial example showing a cube with two intersecting cylinders. Also shown is a sectioned view of the finite element model after completing the H-Morph process. Figure 9 shows tetrahedra comprising approximately 3 percent of the model volume remaining on the interior of the volume. The final example in Figure 10 represents a more complex surface mesh which cannot be resolved my mapping or sweeping methods. The geometry used in Figure 10 is a standard model previously used by Meyers *et al.* [20] to illustrate the Hex-Tet algorithm, which utilizes a similar premise to H-Morph. The H-Morph technique was able to resolve approximately 93 percent of this example into hexahedra.

6. PERFORMANCE

All elements in the previous examples have positive Jacobian Ratios. This indicates that H-Morph is, at a minimum, able to generate elements that are non-inverted. No mathematical proof, however, exists that guarantees positive Jacobian Ratios. The current research has focussed primarily on obtaining good quality hexahedral elements within the volume. In most cases, the hexahedral elements are of sufficient quality for finite element analysis as defined by standard shape checking procedures [21]. An open problem remaining in the H-Morph algorithm is the development of a general smoothing method for a mixed element mesh. While methods have been presented in the literature for all-tetrahedral or all-hexahedral element meshes, a robust method for mixed element shapes has not yet been provided. As a result, in the current implementation, the quality of the tetrahedra and pyramids can be less than desirable.

In its present implementation, H-Morph generates approximately 10-20 hexahedral elements per second on a standard NT workstation. This is slow by most standards, but tends to be about par when compared to the published performance of the Hex-Tet [20] algorithm. Optimization of the H-Morph algorithms must be addressed as they gain maturity.

7. CONCLUSION

A novel new application for constrained triangulation has been introduced in this work. The hex-dominant mesh generation problem has long been a difficult problem. The proposed method satisfies many of the industrial requirements of finite element applications. The proposed method will conform to a prescribed quadrilateral surface mesh, is general purpose, and is able to mesh without the need to decompose or recognize special classes of geometry. The approach proposed for the generation of hexahedral elements begins with an initial tetrahedral mesh and systematically transforms the tetrahedra into a topology appropriate for the formation of well-shaped hexahedra. The process, built on the ideas initially developed in the Q-Morph algorithm [7] for two-dimensional quad meshes, utilizes an advancing front approach. Beginning at the boundary surface mesh, quadrilateral fronts are classified and processed, replacing tetrahedra with hexahedra as the algorithm proceeds. The proposed method is also able to maintain a valid mixed hexahedral-tetrahedral mesh throughout the entire procedure. This eliminates the need for meshing complex internal voids that would otherwise remain after meshing using other methods [20]. The H-Morph algorithm has currently been implemented and tested on a limited number of models sufficient to demonstrate its validity in generating hex-dominant meshes. Further development of the proposed method is on-going and performance and robustness will inevitably improve as the algorithms gain maturity.

ACKNOWLEDGEMENT

The author wishes to acknowledge Ansys Inc. for providing the funding and resources for the development of this work.

REFERENCES

1 Steven J. Owen, "A Survey of Unstructured Mesh Generation Technology", *Proceedings, 7th International Meshing Roundtable*, (1998)

2 David F. Watson, "Computing the Delaunay Tessellation with Application to Voronoi Polytopes", *The Computer Journal*, Vol. 24(2) pp.167-172 (1981)

3 Timothy J. Baker, "Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation", *Engineering with Computers*, Vol. 5, pp.161-175 (1989)

4 Boris, N. Delaunay, "Sur la Sphere" Vide. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk Vol 7 pp.793-800 (1934)

5 Steven J. Owen and Sunil Saigal, "H-Morph, An Indirect Approach to Advancing Front Hex Meshing", *International Journal for Numerical Methods in Engineering*, Vol. 49 (2000)

6 Steven J. Owen, "Non-Simplicial Unstructured Mesh Generation," Ph.D. Dissertation, Carnegie Mellon University (1999)

7 Steven J. Owen, Matthew L. Staten, Scott A. Canann and Sunil Saigal, "Q-Morph: An Indirect Approach to Advancing Front Quad Meshing," *International Journal for Numerical Methods in Engineering*. Vol. 44, pp. 1317-1340 (1999)

8 Ted D. Blacker, "Paving: A New Approach To Automated Quadrilateral Mesh Generation", *International Journal For Numerical Methods in Engineering*, John Wiley, Num 32, pp.811-847, 1991

9 P.L. George, F. Hecht and E. Saltel, "Automatic Mesh Generator with Specified Boundary", *Computer Methods in Applied Mechanics and Engineering*, Vol 92, pp.269-288 (1991)

10 T.D. Blacker and Meyers R.J. "Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm", *Engineering with Computers*, Vol. 2, No. 9, pp.83-93 (1993)

11 Barry Joe, "Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations", Siam J. Sci. Comput., Vol 16, pp.1292-1307, 1995

12 Owen, Steven J., Scott A. Canann and Sunil Saigal "Pyramid Elements for Maintaining Tetrahedra to Hexahedra Conformability", *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ASME, pp.123-129, July 1997

13 Paul-Louis George and Houman Borouchaki, "Delaunay Triangulation and Meshing, Application to Finite Elements", Hermes, Paris (1998)

14 Nigel P. Weatherill and O. Hassan, "Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints", *International Journal for Numerical Methods in Engineering*, Vol 37, pp.2005-2039 (1994)

15 David A. Field, "Laplacian Smoothing and Delaunay Triangulations," *Communications in Applied Numerical Methods*, Wiley, Vol 4, pp.709-712 (1988)

16 W.A. Cook, and W.R. Oakes "Mapping Methods for Generating Three-Dimensional Meshes", *Computers in Mechanical Engineering*, August 1982, pp. 67-72

17 Timothy J. Tautges, Shang-sheng Liu, Yong Lu, Jason Kraftcheck, Rajit Gadh, "Feature Recognition Applications in Mesh Generation", AMD-Vol. 220 *Trends in Unstructured Mesh Generation*, ASME, pp.117-121 (1997)

18 Shang-Sheng Liu, and Rajit Gadh, "Basic LOgical Bulk Shapes (BLOBS) for Finite Element Hexahedral Mesh Generation", *Proceedings, 5th International Meshing Roundtable*, pp.291-306 (1996)

19 David R. White, "Automated Hexahedral Mesh Generation by Virtual Decomposition", *Proceedings, 4th International Meshing Roundtable*, pp.165-176(1995)

20 Ray J. Meyers, Timothy J. Tautges and Phillip M. Tuchinsky, "The Hex-Tet Hex-Dominant Meshing Algorithm as Implemented in Cubit", *Proceedings, 7th International Meshing Roundtable*, pp. 151-158 (1998)

21 Stan Kelley, 'Element Shape Checking', *Ansys Theory Manual*, chapter **13**, (1998).

LIST OF FIGURES

Figure 1. Construction of a hexahedron with the H-Morph algorithm. 15
Figure 2. Face $n_1n_2n_3n_4$ to be recovered from tetrahedra 16
Figure 3. Set of faces comprising the *pipe* for edge recovery 17
Figure 4. Definition of adjacent tetrahedra to node A through which segment *S* passes. 18
Figure 5. Pebble 19
Figure 6. Determination of penetrating edge, E_p. 20
Figure 7. The "firtree" as an example of an irreducible shell. 21
Figure 8. Progression of H-Morph algorithm on simple blocky model 22
Figure 9. Cube with intersecting cylinders 23
Figure 10. Throw model (Courtesy of Sandia National Labs) 24

LIST OF TABLES

Table 1. Tetrahedral transformations used in edge and face recovery 25

LIST OF ALGORITHMS

Algorithm 1. Edge recovery process 26
Algorithm 2. Formation of *Λ(S)* 27
Algorithm 3. Determination of penetrating edge, E_p. 29

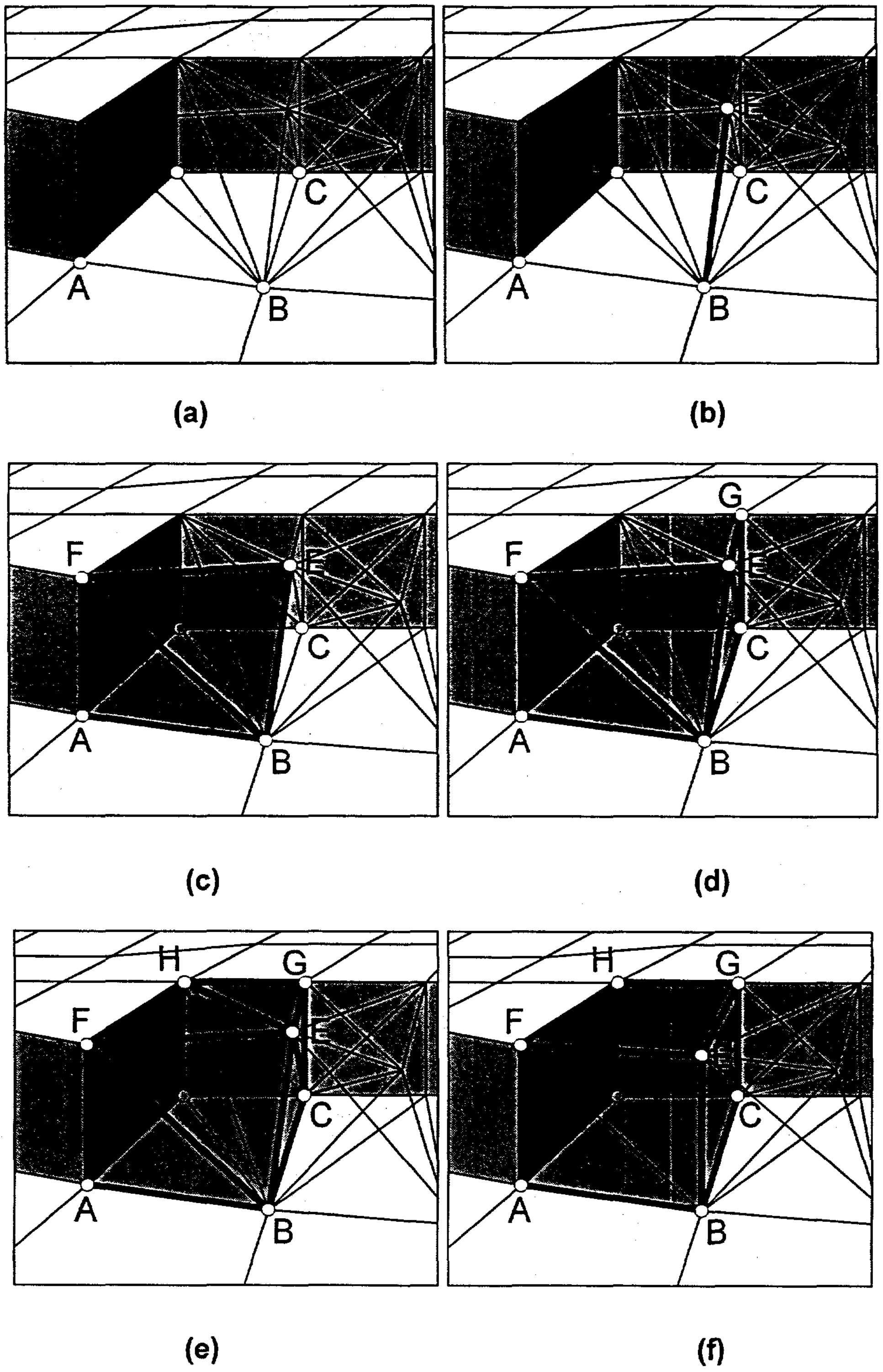


Figure 1. Construction of a hexahedron with the H-Morph algorithm.

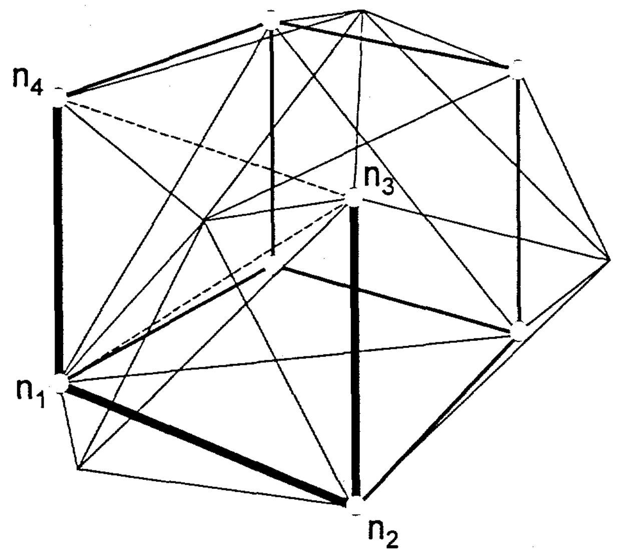


Figure 2. Face $n_1n_2n_3n_4$ to be recovered from tetrahedra

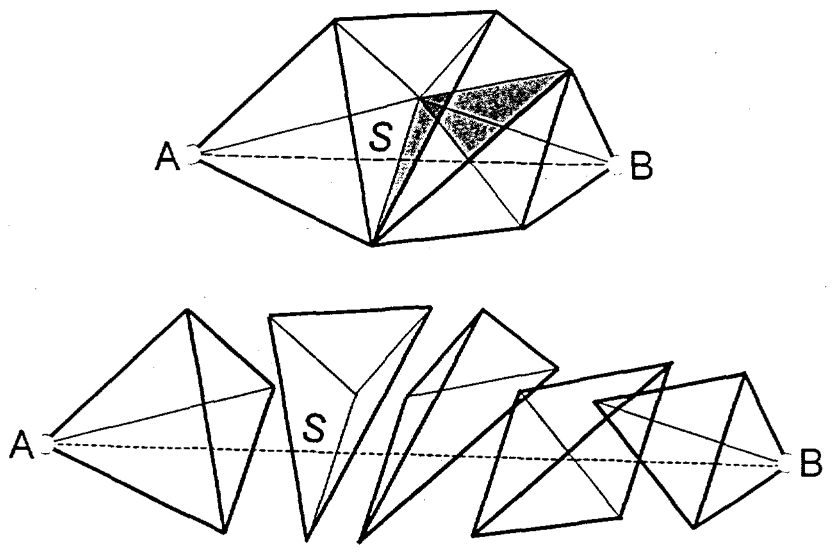


Figure 3. Set of faces comprising the *pipe* for edge recovery

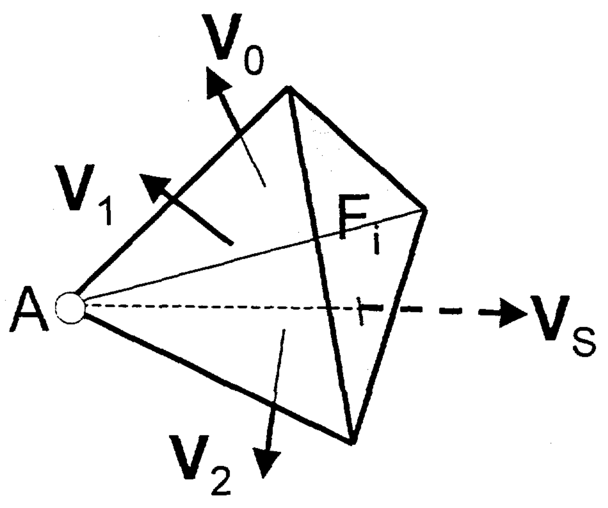


Figure 4. Definition of adjacent tetrahedra to node A through which segment *S* passes.

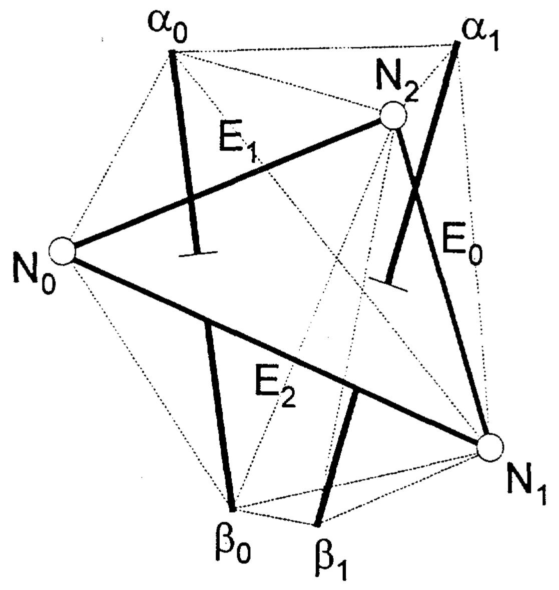


Figure 5. Pebble

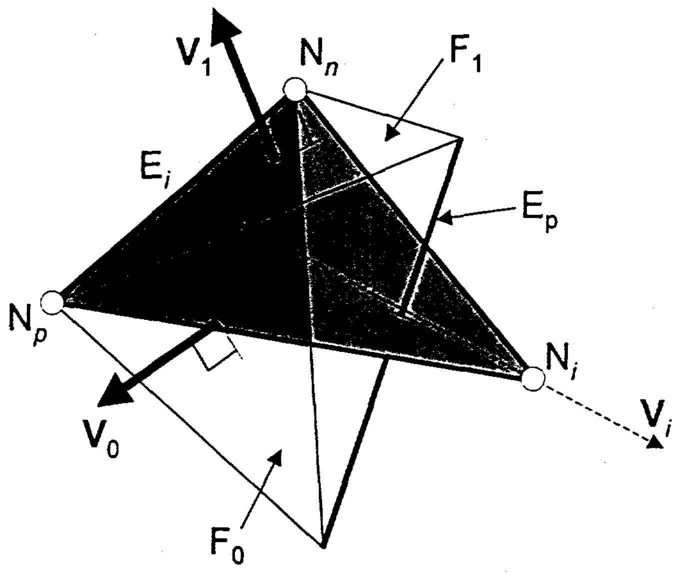


Figure 6. Determination of penetrating edge, E_p.

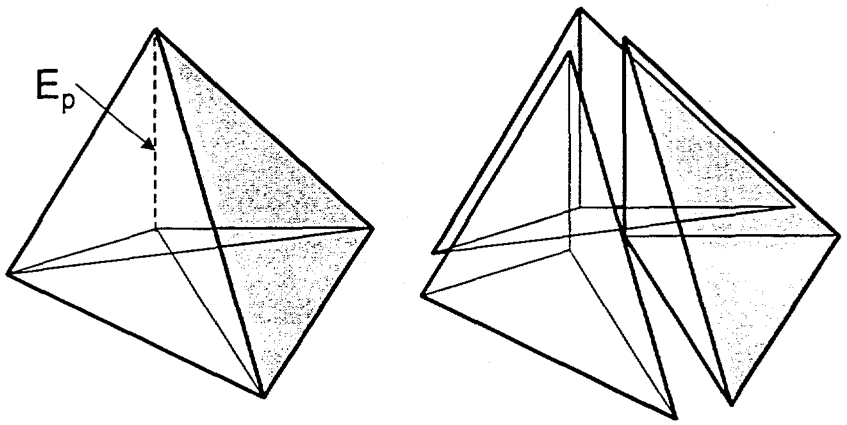


Figure 7. The "firtree" as an example of an irreducible shell.

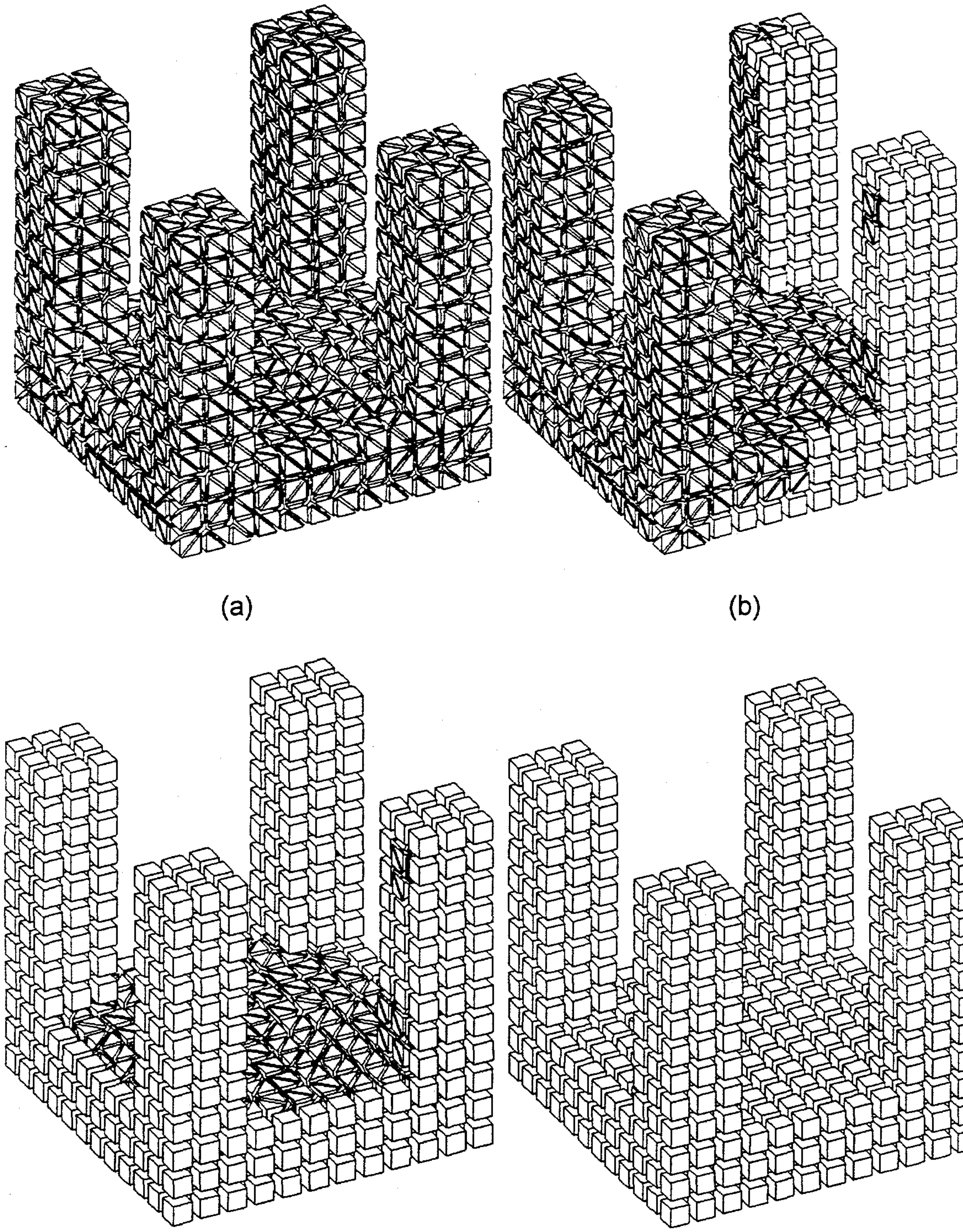
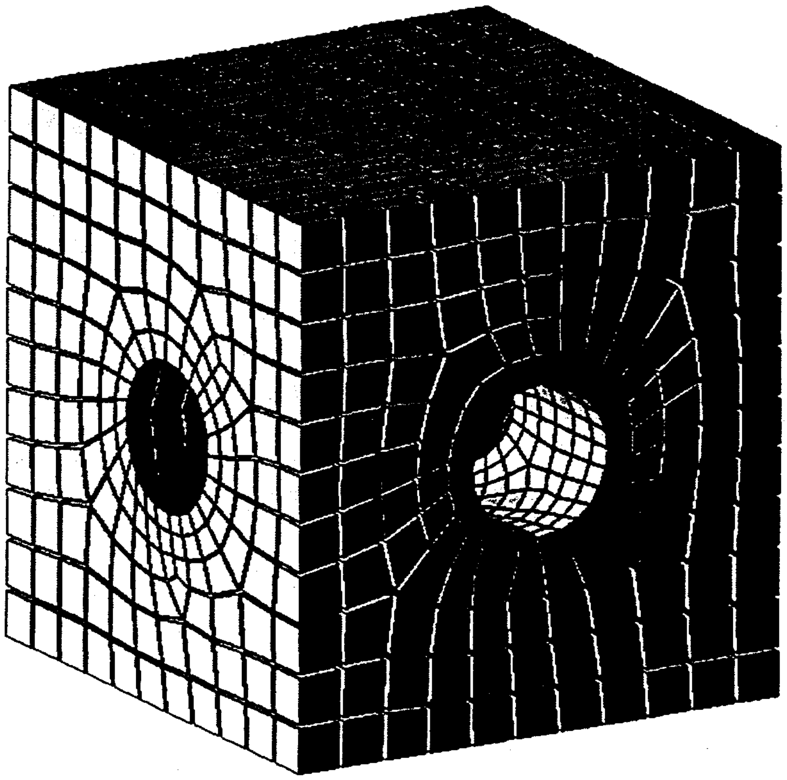
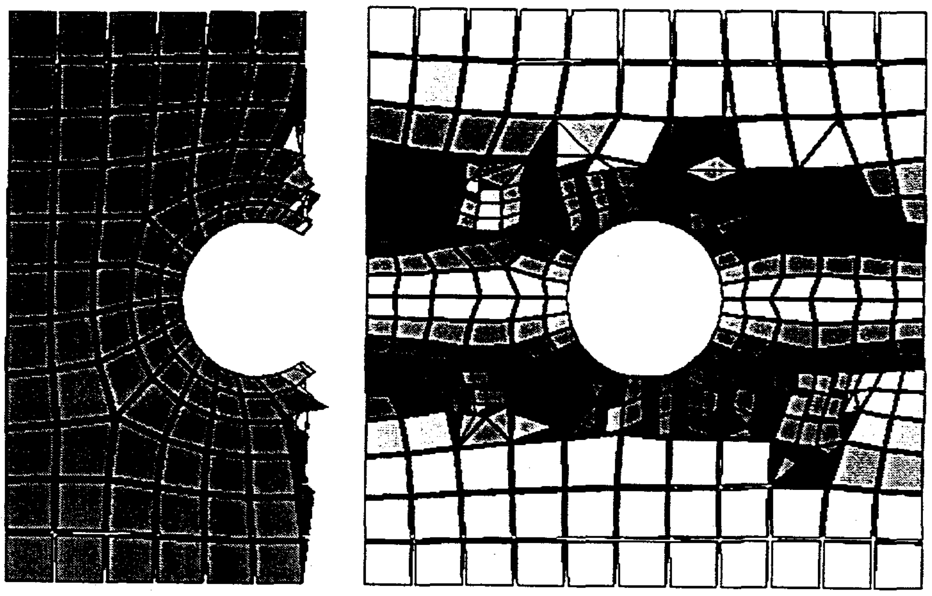


Figure 8. Progression of H-Morph algorithm on simple blocky model

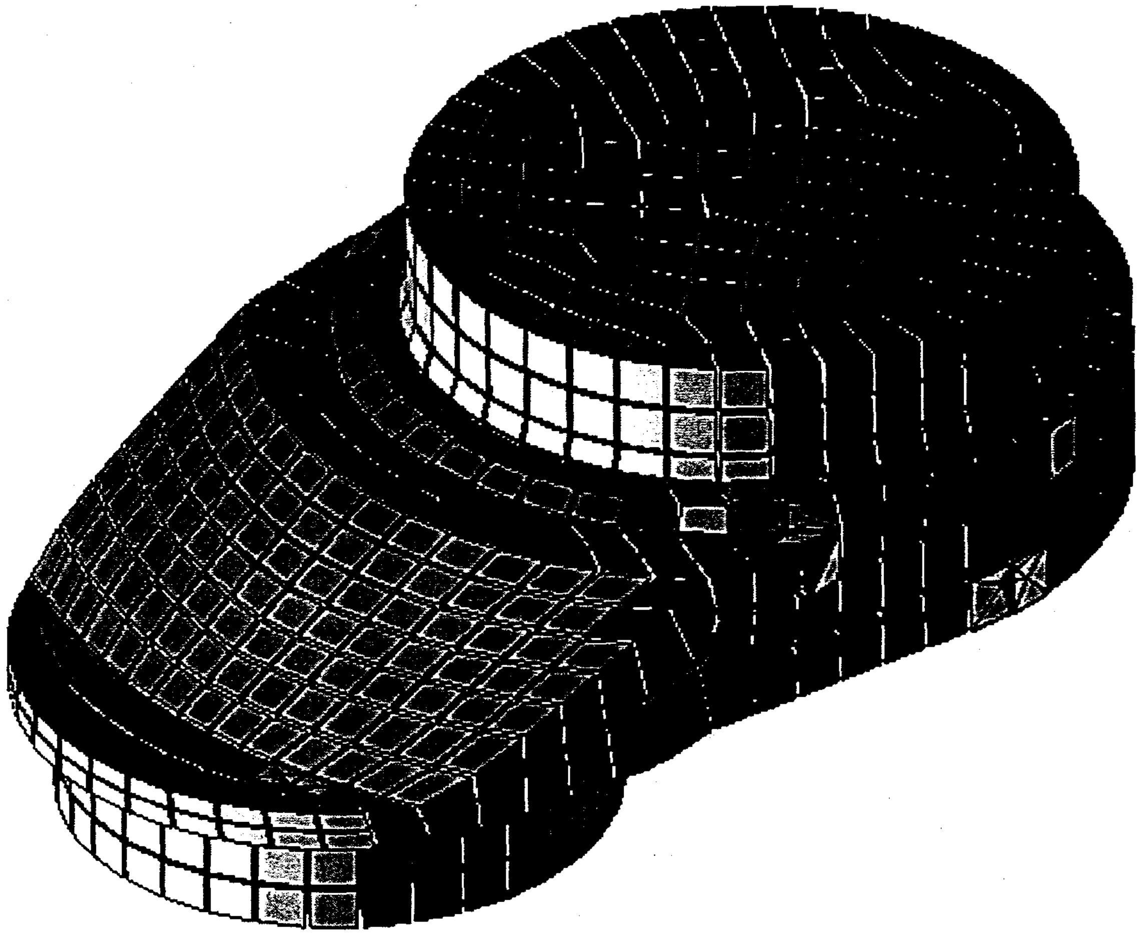


(a) Finite Element Mesh



(b) Cut-away views of Finite Element Mesh

Figure 9. Cube with intersecting cylinders



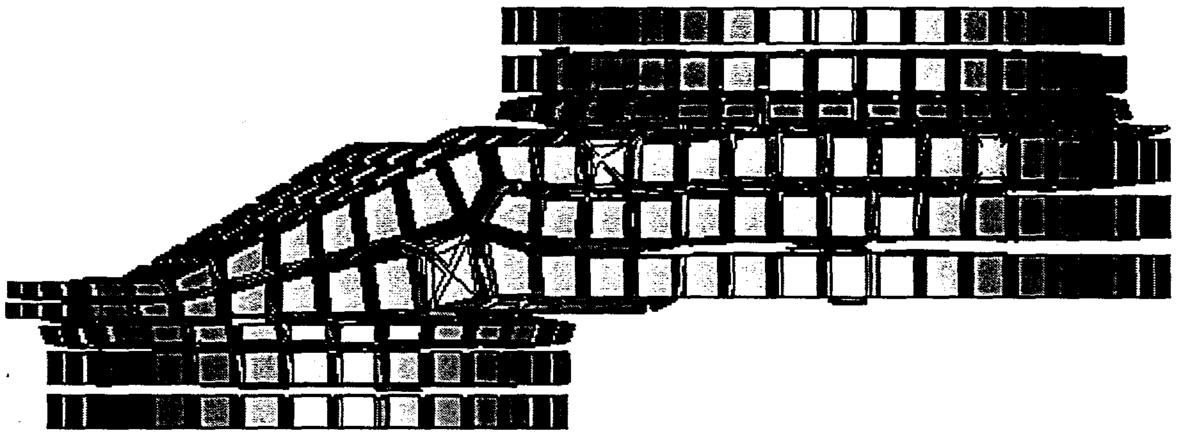
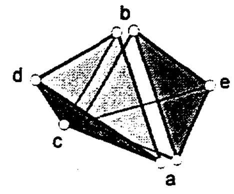
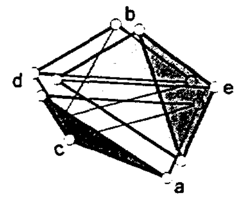
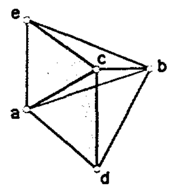
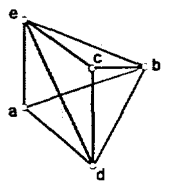
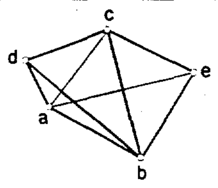
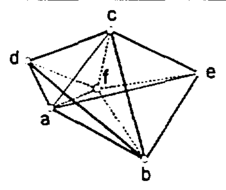
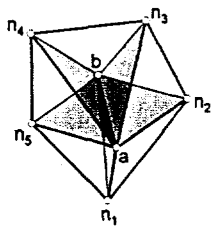
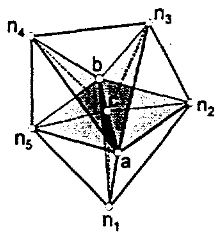
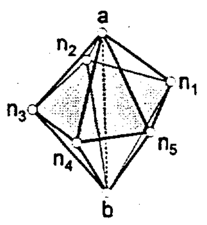
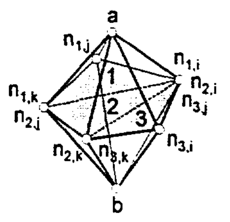


Figure 10. Throw model (Courtesy of Sandia National Labs)

Table 1. Tetrahedral transformations used in edge and face recovery

swap23	 abce, acbd	 abde, bcde, cade
swap22	 aceb, adcb	 adeb, edcb
face-split	 abce, acbd	 abfe, bcfe, cafe, bafd, cbfd, acfd
edge-split	 abn_1n_2, abn_2n_3, $abn_in_{i+1}, \ldots abn_Nn_1$ N = no. adjacent tets at edge ab	 acn_1n_2, acn_2n_3, $acn_in_{i+1}, \ldots acn_Nn_1$, cbn_1n_2, cbn_2n_3, $cbn_in_{i+1}, \ldots cbn_Nn_1$
edge-suppress	 abn_1n_2, abn_2n_3, $abn_in_{i+1}, \ldots abn_Nn_1$ N = no. adjacent tets at edge ab	 a $n_{1,j}$ $n_{1,i}$ $n_{2,i}$ $n_{1,k}$ $n_{3,j}$ 1 2 3 $n_{2,j}$ $n_{2,k}$ $n_{3,k}$ $n_{3,i}$ b $n_{1,k}n_{1,j}n_{1,i}a$, $n_{1,i}n_{1,j}n_{1,k}b$, $n_{2,k}n_{2,j}n_{2,i}a$, $n_{2,i}n_{2,j}n_{2,k}b, \ldots$ $n_{M,k}n_{M,j}n_{M,i}a$, $n_{M,i}n_{M,j}n_{M,k}b$ M = no. unique trias in polygon $P=\{n_1,n_2,\ldots n_N\}$

Algorithm 1. Edge recovery process

```
1.  LET $S$ be the line segment from A to B
2.  LET $\Lambda(S)$ be a list of faces $F_i$ that are intersected by $S$ (see Algorithm 2)
3.  LET $nfail(F_i)$ be the number of attempts to swap $F_i$; $nfail(F_i)=0$
4.  FOR EACH $F_i \in \Lambda(S)$
5.      LET $T(F_i)$ be the set of 2 tetrahedra adjacent $F_i$
6.      LET $T^{-1}(F_i)$ be the set of 3 tetrahedra where a swap23 operation (Table 1) has been
            performed removing face $F_i$.
7.      IF volume of 3 tetrahedra in $T^{-1}(F_i) > 0$ THEN
8.          Form $T^{-1}(F_i)$
9.          Delete $F_i$ from $\Lambda(S)$
10.         FOR EACH of the 3 common faces, $F_j$ in $T^{-1}(F_i)$
11.             IF $F_j$ intersects $S$ THEN add $F_j$ last on $\Lambda(S)$,
12.     ELSE,
13.         IF $nfail(F_i) < 4$ THEN Place $F_i$ last on $\Lambda(S)$
14.         ELSE smooth($\Lambda(S)$); goto 2
15.     END IF
16. NEXT $F_i$ on $\Lambda(S)$
```

Algorithm 2. Formation of $\Lambda(S)$

```
1.  LET $\mathbf{V}_S$ be the vector from node A to node B
2.  LET topocase = *atbegin*
3.  LOOP
4.      IF topocase = *atbegin*
5.          LET T(A) be the set of tetrahedra adjacent A, $T_k(A) \in T(A)$
6.          FOR EACH $T_k(A) \in T(A)$
7.              LET $F(T_k)$ be the set of faces on $T_k(A)$ with node A as a vertex, $F_j(T_k) \in F(T_k)$
8.              LET $\mathbf{V}_j$ be the outward pointing normalized normal to face $F_j(T_k)$ on $T_k(A)$
9.              IF $\mathbf{V}_S \cdot \mathbf{V}_j < 0 \; \forall \; F_j(T_k)$ THEN LET $F_i$ be the opposite face on $T_k(A)$ from A; $T_i$ =
                    tetrahedra adjacent $F_i$, $T_i \neq T_k(A)$ ; topocase = *atface*
10.             ELSE IF $\mathbf{V}_S \cdot \mathbf{V}_j = 0$ for one $F_j(T_k)$ THEN LET $E_i$ be the opposite edge on $F_j(T_k)$
                    from A; topocase = *atedge*
11.             ELSE IF $\mathbf{V}_S \cdot \mathbf{V}_j = 0$ for two $F_j(T_k)$ THEN LET $N_i$ be the opposite node from A on the
                    common edge for both $F_j(T_k)$ where $\mathbf{V}_S \cdot \mathbf{V}_j = 0$ ; topocase = *atnode*
12.             ELSE NEXT $T_k(A)$
13.             END IF
14.         END FOR
15.     ELSE IF topocase = *atnode*
16.         IF $N_i$=B THEN topocase = *atend*
17.         ELSE smooth($N_i$) using Laplacian smoothing and node-relocation operation; topocase
                = *atbegin*
18.     ELSE IF topocase = *atedge*
19.         Perform edge-supression operation on edge $E_i$
20.         IF edge-supression successful THEN topocase = *atbegin*
```

```
21.         ELSE smooth endnodes of edge $E_i$ using Laplacian smoothing and node-relocation
              operation; topocase = *atbegin*
22.     ELSE IF topocase = *atface*
23.         IF $F_i$ is a face on the front THEN fail ELSE Add face $F_i$ into pipe $\Lambda(S)$
24.         LET $N_j$ be the nodes on $F_i$, $j = 0,1,2$; $N_3$ = opposite node on $T_i$ from $F_i$
25.         LET $d_j = V_s \cdot \left(\overrightarrow{N_jN_3} \times \overrightarrow{N_jA}\right), j = 0,1,2$
26.         IF $d_j = 0 \; \forall \; j$ THEN $N_i = N_3$; topocase = *atnode*
27.         ELSE IF $d_j = 0$ for $j = k$ THEN $E_i = \overline{N_kN_3}$; topocase = *atedge*
28.         ELSE IF $d_j < 0$ AND $d_k > 0$, $k = (j+1)\%3$ THEN $F_i = \Delta N_jN_kN_4$; $T_{i+1}$ = adjacent
              tetrahedra at $F_i$, $T_{i+1} \neq T_i$; $T_i = T_{i+1}$; topocase = *atface*
29.         END IF
30.     END IF
31. UNTIL topocase = *atend*
```

Algorithm 3. Determination of penetrating edge, E_p.

```
1.  LET $E_p$ = NULL
2.  LET $T_j$ be a tetrahedra where $T_j \in shell(E_i)$
3.  FOR EACH $T_j \in shell(E_i)$ AND $E_p$ = NULL
4.      LET $F_k$, $\{k = 0,1\}$ be the faces on $T_j$ sharing edge $E_i$
5.      LET $\mathbf{V}_k$ = outward pointing normal of $F_k$ from $T_j$
6.      LET $\mathbf{V}_i = \left( \overrightarrow{N_i} - \frac{\overrightarrow{N_n} + \overrightarrow{N_p}}{2} \right)$, where $n = (i+1)\%3$,  $p = (i+2)\%3$
7.      IF $\mathbf{V}_i \cdot \mathbf{V}_k < 0$, $\{k = 0,1\}$, THEN
8.          LET $E_p$ = opposite edge on $T_j$ from $E_i$
9.      ELSE IF $\mathbf{V}_i \cdot \mathbf{V}_k = 0$, THEN
10.         LET $N_{opp}$ = opposite node on $F_k$ from $E_i$
11.         $N_{opp} = N_{opp} + \varepsilon$
12.         GO TO 1
13.     ENDIF
14. NEXT $T_j$
```

BIOGRAPHY

Steve Owen is currently a Senior Member of Technical Staff in the Parallel Computing Sciences Department at Sandia National Laboratories in Albuquerque, New Mexico. He also chairs the 9th International Meshing Roundtable Steering Committee. He is working on research and development in finite element mesh generation and geometry related issues. Steve recently completed his Ph.D. from Carnegie Mellon University (1999) while working at ANSYS Inc. in Pittsburgh Pennsylvania. He received his Bachelor's and Master's degrees from Brigham Young University.