The Geode Algorithm: Combining Hex/Tet Plastering, Dicing and Transition Elements for Automatic, All-Hex Mesh Generation

Robert W. Leland[1]
Darryl J. Melander[1]
Ray W. Meyers[1]
Scott A. Mitchell[1]
Timothy J. Tautges[1]

Abstract. *A new all-hexahedral meshing algorithm, referred to as "Geode", is described. This algorithm is the combination of hex/tet plastering, dicing, and a new 26-hex transition element template.The algorithm is described in detail, and examples are given of problems meshed with this algorithm.*

keywords. Geode; hexahedral mesh generation; transition template.

1. Introduction

Finite element analysis is used to model physical phenomena in a wide variety of disciplines, including structural mechanics and dynamics, heat transfer, and computational fluid dynamics (CFD). To perform these analyses, the problem domain must first be discretized into one-, two- or three-dimensional elements. While the finite element method allows many types of elements, tetrahedra and hexahedra are typically used; furthermore, hexahedra are considered more accurate for a given cost for some types of analyses, particularly in the non-linear regime. However, generating all-hexahedral meshes for typical problems has proven quite difficult.

To date, no automatic all-hexahedral meshing algorithm has been found which delivers both complete automation and high-quality meshes, both of which are critical to acceptance of such an algorithm in the FEA field. There have been many research efforts to find a suitable algorithm. Whisker weaving[1] operates in the mesh dual to find the topology of an all-hex mesh, then smooths to locate nodes in physical space. While this algorithm has shown promise, it has not yet achieved the required robustness for typical problems. Other algorithms investigated by Schneiders et. al[2] and by researchers at Cray Research[3] have proven robust, but have generated meshes which tend to be large and which have poorest quality near region boundaries. Recent efforts at Sandia National Laboratories have shown that hex meshes can be obtained by slicing automatically-generated tetrahedral meshes into hexahedra, and that these meshes may be suitable for some analyses, but these meshes have yet to be tested for a wide variety of non-linear analyses[4].

This paper describes the Geode algorithm, which generates all-hexahedral meshes automatically for arbitrary geometries. The algorithm is based on hex-tet plastering, an advancing-front, hex-dominant meshing algorithm which generates hex elements from the boundary inward followed by tetrahedra on the interior[5]. Geode inserts a "necklace" layer of hex elements between the hexes and tets, then dices the ensemble to produce an all-hex mesh. A special transition element is used to dice the necklace layer of hexes [13]. The algorithm is in the early stages of development, and some issues remain regarding the quality of the elements generated in the transition layer in realistic problems. These are being diligently researched, and our feeling is that the algorithm shows much promise.

The remainder of this paper is organized as follows. Section 2 describes the hex-tet plastering algorithm and the dicing techniques in more detail. Section 3 describes the combination of these algorithms and the necessary additions for the

1 Parallel Computing Sciences Dept., Mail Stop 0441, Sandia National Laboratories, Albuquerque, NM 87185. The authors were supported by the Mathematical, Information and Computational Sciences Division of the U.S. Department of Energy, Office of Energy Research, and work at Sandia National Laboratories, operated for the U.S. DOE under contract No. DE-AL04-94AL8500. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. DOE.

Geode algorithm. Section 4 gives examples of the Geode algorithm, and Section 5 gives conclusions and future directions.

2. Key Meshing Technology

The Geode algorithm can be thought of as combining two key meshing algorithms, hex/tet plastering and hex and tet dicing, with a special transition element layer. In this section, we describe each of these pieces.

2.1 Hex-Tet Plastering

In the past, Sandia has researched two very different all-hex meshing algorithms, plastering[6] and Whisker Weaving[1]. After comparing the relative merits of the two, we decided to discontinue research on plastering and devote our efforts to Whisker Weaving. While there has been substantial progress on the Whisker Weaving algorithm since then[7], we still do not have a fully robust and general algorithm. In recognition of this, Sandia recently began pursuing a shorter-term goal of developing an algorithm which would give hex-dominant meshes while retaining the characteristics of suitable (but not optimal) mesh quality and complete automation. This effort resulted in the hex-tet plastering algorithm[5]. The hex-tet plastering algorithm consists of three steps, which are described below.

A. Plastering

The plastering algorithm can be considered a volume analogy to the paving algorithm[8]. Starting with a closed boundary of quadrilateral elements, each element is projected into the volume to form a hexahedron. Elements are projected layer by layer to maintain an advancing-front mesh, which has the desirable characteristic of placing highest quality elements near volume boundaries. As the meshing front proceeds inward, special merging operations are done to stitch the front(s) together (see [6] for more details of this process.)

At some point in this process, the plastering algorithm is unable to proceed further without producing intersections with another part of the front or forming hexes whose quality is below a prescribed minimum. At this point, the plastering algorithm exits and the hex-tet plastering algorithm proceeds to the next stage.

B. Hex-Tet Transition

There are several possible methods to transition between a hex and tet mesh; these methods can be classified by whether they are conformal or non-conformal[2]. We have implemented one conformal transition method and two non-conformal methods; the choice of which method to use can be made at runtime in our implementation.

A non-conforming interface between hexes and tets can be obtained simply by converting each quadrilateral bounding the void to several triangles; we have implemented both a two- and four-triangle transition. These transitions differ in their robustness and in the number of tetrahedra ultimately generated [5], but leave a triangle-bounded void on which to start the tet mesher.

To transition between hex and tet meshes in a conformal manner, additional element types are necessary. The two most common elements used in this case are triangular prisms and pyramids. We choose to use only pyramids for the transition in Hex-Tet Plastering (the Geode algorithm uses a different transition method described later), since they are simpler to use in this context. In particular, we have implemented the method proposed by Canann et. al [9]. In this method, tetrahedral meshing is performed from a two-triangle non-conforming interface; afterwards the tetrahedra next to the interface are converted to pyramids using a combination of transition primitives and tetrahedra recombinations.

2 A conformal mesh is one where elements sharing more than one node also share all the edges and faces comprised by the shared nodes. Such meshes are also said to be contiguous. A non-conformal mesh is one where these conditions do not hold.

C. Tetrahedral meshing

Following the transition region generation, the remaining region is meshed with tetrahedra, using the tetrahedral meshing engine in CUBIT[10].

Studies have shown the hex-tet plastering algorithm to be quite robust while generating hex and tet elements of high quality[5].

2.2 Hex and Tet Dicing

The hex dicing algorithm was implemented in CUBIT to allow the generation of large (multi-million element) meshes[11]. This algorithm simply divides each hexahedral element in a mesh into smaller elements, optionally projecting the newly generated nodes onto the geometric boundary where appropriate. If the coarse mesh is conformal and all-hexahedral, the fine mesh will have the same characteristics.

Tetrahedral meshes can be subdivided into hexahedra as well by using a simple primitive. Each triangle is subdivided into three quadrilaterals, a node is inserted in the interior of the tetrahedron, and edges are made which connect that node to the four mid-face nodes (see Figure 1). This splits the tetrahedron into four hexahedra. If the initial mesh is all-tetrahedral and fully conformal, the fine mesh will be conformal as well, but will consist only of hexahedral elements.

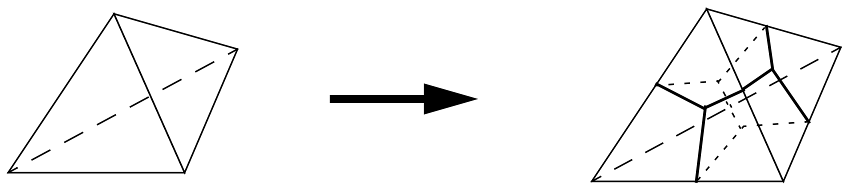

Figure 1. Tetrahedral dicing or subdivision.

It is interesting to note that tet dicing could be considered as an alternative method for generating all-hexahedral meshes (when combined with an automatic tetrahedral meshing algorithm). Conventional wisdom has in the past held that these meshes would have insufficient quality for finite element analysis; however, recent studies have shown otherwise for selected analysis types[4].

2.3 Transition Layer

Hex-tet plastering has been shown to yield hex-dominant meshes of sufficient quality, and we described earlier how hex and tet elements can each be split into finer, all-hexahedral elements. A natural question is, then, can these methods be combined to yield an automatic, all-hexahedral meshing algorithm that generates fully conformal meshes? We emphasize that this method would generate high-quality, well fitted hexes along the boundary, a very desirable property that has eluded previous attempts to produce an automatic hex meshing algorithm.

If a hex-tet plastered mesh which uses a two-triangle transition is diced using the methods described above, the result is a non-conformal interface. Dicing a hex-tet plastered mesh which uses pyramids to get a conformal interface gets us closer to the goal, yielding diced hex and tet regions separated by pyramids whose faces are diced into quadrilaterals. We initially thought this would lead to the desired algorithm; the only missing piece was a transition element corresponding to this quad-bounded pyramid. Unfortunately, obtaining an all-hexahedral mesh for this geometry and topology has proven problematic. Currently, the minimal all-hexahedral mesh for this arrangement consists of approximately 128 hex elements. In fact, this has been posed as an “open problem” to the meshing community for some time[12], and has not been solved to date.

We therefore developed an alternative approach to using pyramids on the hex-tet interface. We use a “necklace” layer of hexes consisting of a single hex for each quadrilateral on the initial void boundary. Each necklace has one face on the original hex boundary, and the opposite face on the new, tet-facing boundary. The remaining four faces are shared by other necklace hexes only. The transition problem then becomes one of transitioning between the diced hex-facing quad and the two-triangle-then-diced tet-facing quad (see Figure 2, left). The solution to this problem is described in another paper[13], however, the result is the transition element shown in Figure 2, center, right.

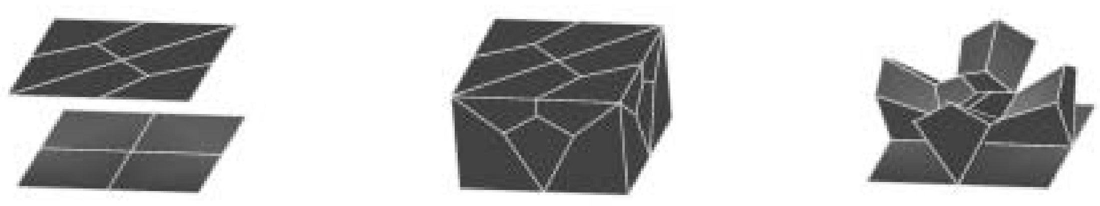
Figure 2. Constraints on transition element (left); proposed transition element meeting those constraints (center, right); from [13].

3. The Geode Algorithm

Using the algorithms described earlier, we can now construct the Geode algorithm. The required steps, as implemented in CUBIT, are:

1. Plaster

2. Generate necklace layer of hexes

3. Convert void-bounding quads to two triangles

4. Tet mesh remaining void

5. Dice hexes and tets

6. Insert transition template into each necklace hex and smooth locally

The sequencing of these steps can be modified; for example, we experimented with generating the necklace layer either before or after the generation of tets. In addition, we have explored the generation of the necklace by pulling faces back from the void instead of projecting the faces into the void. We have found that the procedure described above, using projection-based necklace generation, works best with the plastering algorithm available in CUBIT[14].

4. Examples and Applications

Examples of meshes generated by the Geode algorithm are shown in Figure 3 and Figure 4. These are early results meant to characterize the sort of geometries we can presently mesh automatically and with reasonable quality using the Geode algorithm. The current limitation is that we have difficulty ensuring that all elements in the transition layer have positive Jacobians. We are working on advanced smoothing algorithms to remedy this situation. Since the underlying components of the Geode algorithm are already capable of automatically meshing substantially more complex geometries than those shown, we are optimistic that, with improved smoothing, the Geode algorithm will have practical impact on difficult meshing problems.

It has been shown by Meyers et. al[5] that hex-tet plastering generates an increasing volume fraction of hexes as initial bounding mesh size is decreased, as expected; this characteristic also benefits the Geode algorithm, by minimizing the number of transition elements and concentrating them in region interiors, away from principal loads and/or wall effects.

There are many potential applications of the Geode algorithm which we intend to explore. The most obvious is as an automatic, all-hexahedral meshing algorithm for arbitrary geometries. This algorithm would be appropriate for generating hex meshes for optimization loops or for parallel meshing. Previously only tet-based hex meshes could be used in these settings because of the level of automation required.

There is another application of Geode which is not as obvious, but which we feel is perhaps more significant, and this is as another tool in the hex-meshing toolbox. Since Geode generates hex meshes which have suitable but not optimal quality, it should be used only when necessary. Given the availability of simpler algorithms which generate meshes of higher quality, e.g. mapping and sweeping, along with techniques and tools for decomposing geometry into sub-regions, Geode could be used only on the difficult sub-regions, leaving other regions to be meshed with simpler algorithms. We believe that this combination of algorithms will substantially reduce the amount of geometry decomposition (and hence time) required to mesh complex assemblies, while maintaining the high overall mesh quality which motivates the need for hex meshes in the first place. We note that a hex meshing algorithm based on tet-dicing could not be used in this type of application.

A key capability for this application is that of generating an all-hexahedral mesh starting from a fixed, all-quadrilateral boundary. In this context, volumes meshed with Geode would need to be meshed first, or a method to de-refine a quadrilateral mesh would be necessary, since Geode dices the initial quadrilaterals bounding the void.

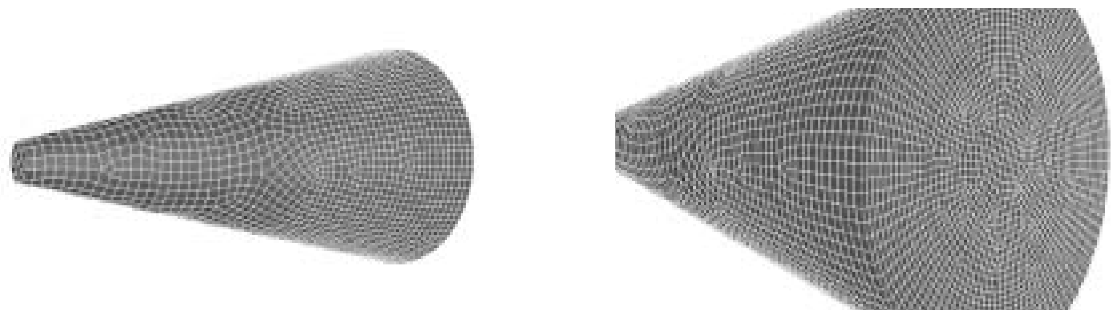

Figure 3. Two views of a conical geometry automatically hex meshed by the Geode algorithm.

Figure 3 demonstrates application of the Geode algorithm to a conical geometry; two views of the same geometry are shown. Axial sweeping is possible but not desirable here because the high aspect ratio between the source and target surfaces would result in elements of inappropriate size on one of these surfaces. Azimuthal sweeping is also possible but leads to wedge shaped elements also considered undesirable for the intended analysis. The Geode algorithm was able to generate a valid all-hex mesh with fairly uniform size starting from a Paved surface mesh. In this instance no Plastering was used, so the surface mesh displays one face from each of the elements in the transition layer. When Plastering was employed, the surface to which the transition layer is attached becomes less regular, and construction of valid transition elements becomes more difficult, resulting in the generation of small fraction with negative Jacobians.

On the left in Figure 4 a three dimensional duct geometry is shown. This could, after minimal decomposition, be meshed with traditional sweeping, so this is merely a demonstration problem. The Geode algorithm was able to generate an all-hex mesh of about 24k elements for this geometry. Two of these elements had negative Jacobians in the case where no plastering was used, and four did when Plastering was used. In the latter case, the worst Jacobian was 10e-3, and 16 elements had Jacobians less than 10e-2.

On the right in Figure 4, we show a cube geometry with circular imprints. The imprints prevent straight-forward sweeping and make this a difficult problem requiring substantial decomposition with the standard approach. The Geode algorithm was able to automatically generate a valid, all-hex mesh with no negative Jacobians for this geometry.

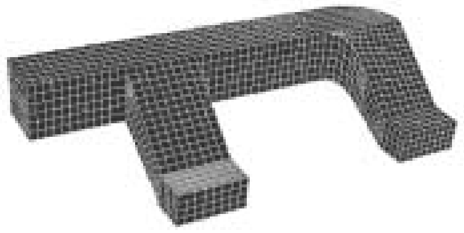
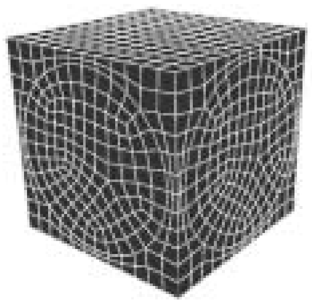

Figure 4. Examples of all-hex meshes generated automatically be the Geode algorithm.

Here again it was necessary to turn off Plastering to generate a fully valid mesh. With Plastering, one of the 11.5k hex elements generated had a negative Jacobian.

5. Conclusions and Future Work

The hex-tet plastering algorithm has been shown to be robust and to generate mixed-element meshes of suitable quality for a variety of example geometries. Combining this tool with a diced-hex to diced-tet transition template results in an automatic all-hexahedral meshing algorithm. After investigating a number of transition template options, including pyramids and necklace layers of hexes, we have used a transition template based on a necklace layer of hex elements. The resulting Geode algorithm has been implemented in CUBIT, and preliminary results are encouraging. We believe that with additional work on associated smoothing techniques, the Geode algorithm will be capable of automatically generating all-hex meshes of quality suitable for FEM analysis for a meaningful class of geometries.

The Geode algorithm can be used where automatic hex-meshing is required, for example in optimization loops and for parallel mesh generation. Another powerful application is in combination with geometry decomposition and other well-known meshing algorithms; this toolbox approach shows potential in substantially reducing the time to mesh for complicated assemblies.

Going forward, we intend to refine the combination of algorithms in Geode in order to increase the quality of the resulting meshes. In particular, the methods used to generate the necklace transition layer need further investigation. We also intend to use Geode meshes for typical analysis problems at Sandia in order to explore their suitability for engineering analyses of various types. Finally, we will also explore the criteria used to determine when to use Geode versus performing further geometry decomposition to get more mappable and sweepable sub-regions.

References

[1] Timothy J. Tautges, Ted D. Blacker, Scott Mitchell, “The Whisker Weaving Algorithm: a Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes”, Int. J. Numer. Methods Eng., 39:3327-3349 (1996).

[2] R. Schneiders, R. Schindler, F. Weiler, “Octree-based Generation of Hexahedral Element Meshes”, Proceedings of the 5th International Meshing Roundtable, Sandia National Laboratories report SAND96-2301, Oct. 10-11, 1996, Pittsburgh, Pennsylvania.

[3] R. Taghavi, “Automatic, Parallel and Fault Tolerant Mesh Generation from CAD”, Eng. with Comp. (1996) 12:178-185.

[4] Ed Boucheron, Randy Weatherby, “ALEGRA - Meshing Approaches”, http://www.sandia.gov/1431/mesh_showcase.html.

[5] Ray J. Meyers, Timothy J. Tautges, Philip M. Tuchinsky, “The ‘Hex-Tet’ Hex-Dominant Meshing Algorithm as Implemented in CUBIT”, to appear in 7th International Meshing Roundtable, Detroit, MI, October 1998.

[6] Stephenson, M. B., S. A. Canann, T. D. Blacker, “Plastering: A New Approach to Automated 3D Hexahedral Mesh Generation -- Progress Report I”, SAND89-2192, Sandia National Laboratories, 1990.

[7] N. Folwell, S. Mitchell, “Reliable Whisker Weaving via Curve Contraction”, submitted to 7th International Meshing Roundtable, Detroit, MI, October 1998.

[8] T. D. Blacker, M. B. Stephenson, “Paving: A New Approach to Automated Quadrilateral Mesh Generation”, Int. J. Numer. Methods Eng., 32:811-847 (1991).

[9] Owen, S. J., Canann, S. A., Saigal, S., “Formation of Pyramid Elements as a Means of Maintaining Tetrahedra to Hexahedra Conformability,” AMD-Vol. 220 Trends in Unstructured Mesh Generation, ASME, p. 123-129, 1997.

[10] D. R. White, Timothy J. Tautges, “Tetrahedral Meshing Using CUBIT”, CU-TIPS #3 (to be written), Sandia National Laboratories, 1998.

[11] Darryl J. Melander, Timothy J. Tautges, Steven E. Benzley, “Generation of Multi-Million Element Meshes for Solid Model-Based Geometries: The Dicer Algorithm,” AMD-Vol. 220 Trends in Unstructured Mesh Generation, ASME, p. 131-135, 1997.

[12] R. Schneiders, “An interesting open problem”, http://www-users.informatik.rwth-aachen.de/~roberts/open.html.

[13] Scott A. Mitchell, “The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to any Diced Hexahedral Mesh”, to appear in 7th International Meshing Roundtable, Detroit, MI, October 1998.

[14] T. D. Blacker et al., “CUBIT Mesh Generation Environment, Volume 1: User’s Manual”, SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, May 1994.