The "Hex-Tet" Hex-Dominant Meshing Algorithm as Implemented in CUBIT

Ray J. Meyers (rjmeyer@sandia.gov)
Timothy J. Tautges (tjtautge@sandia.gov)
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM

Dr. Philip M. Tuchinsky (ptuchins@ford.com)
Ford Research Laboratory
Mail Drop 2122 SRL
P. O. Box 2053
Dearborn, MI 48121 USA

Abstract. This is a report of the current status of the Hex-Tet algorithm as implemented in the CUBIT toolset. The Hex-Tet algorithm begins by generating a partial hex mesh using an advancing front "Plastering" algorithm. The boundary of any remaining void is optionally "cleaned-up" to improve its shape and/or other properties. The quad boundary is then converted to a triangular boundary by one of three available methods. Finally, the triangle-bounded void is filled with tetrahedra using the tet-generation capabilities within CUBIT. The result is a mixed-element mesh containing hexahedra, tetrahedra, and optionally pyramids. A set of test problems is described which is used to generate data concerning the robustness and speed of the algorithm, as well as properties of the resulting meshes.

Keywords: Hexahedral Meshing, mixed-element mesh, hex-dominant meshing, unstructured mesh generation, plastering.

Introduction

The desire within the finite element analysis community for an automated all-hexahedral mesh generation algorithm is well documented. At Sandia National Laboratories, one effort to provide this capability is the plastering algorithm (Stephenson:1990, Canann:1992, Blacker:1993, Hipp:1994, Hipp:1995), a 3D extension of the very successful Paving algorithm for all-quadrilateral surface meshes. To date, the all-hexahedral algorithm has proved intractable. In recent years, attention was shifted to the "hex-tet" algorithm, which focuses on creating a hex-dominant mesh by filling the volume with as many hexes as possible, and then filling the remainder with tetrahedra (Dewhirst:1995). Tuchinsky and Clark (Tuchinsky:1997) give an excellent paper on past advancements on this algorithm at Sandia. Our report documents very recent developments with the Hex-Tet algorithm, concentrating on the commercialization of the algorithm for use within CUBIT, and on documenting the current robustness of the algorithm and the properties of resulting hex-dominant meshes. The following portions of the paper will document recent additions to the algorithm to enhance robustness, and describe a test suite of non-trivial problems used to evaluate the algorithm and the results obtained.

Adapting the Hex-Tet algorithm for CUBIT

Much recent work done on the Sandia Hex-Tet algorithm was conducted at Ford Motor Company (Tuchinsky:1997) using Ford software, including the Ford tet-generator. This latest plastering algorithm was adapted for inclusion in the CUBIT system. A quality tet-generation algorithm has been recently incorporated in CUBIT, and these two were combined to provide an automated Hex-Tet algorithm.

To provide a transition from the quad-bounded void left by the hex-generator and the tri-bounded void needed by the tet-generator, three options were provided. The first option, labeled “Two” in the test data below, simply splits the quads into two triangles along the shortest diagonal. The second, labeled “Four” below, creates a center point for each quad and divides the quad into four triangles. The last, called “Pyramid”, inserts pyramid elements between the hexes and tets, resulting in a fully conformal mesh containing three element types.

The pyramid generation algorithm used is essentially that described by Owen, Canann, and Saigal (Owen:1997), with a few additions to handle the very complex void geometries created by the HexToVoid algorithm. The most significant of these additions involved the potential for warpage on the quad faces of the void boundary. When a warped quad is divided into two triangles along the diagonal and the resultant triangle boundary is tetrized, it is possible that both of the triangles of the original face may be owned by a single tetrahedron. In this case, we must find the edge of the tetrahedron which is not part of the original two triangles, and split all tets in the model which share this edge. We can then proceed with pyramid construction as normal.

For most analyses, an ideal mixed-element mesh would contain several layers of all-hex mesh near the boundary, with the tets (and possible pyramids) isolated to relatively small areas in the interior. To that end, special attention was given to the hex generation algorithm to attempt to maximize the percent volume occupied by hexes in the model, and to minimize the number of faces on the original meshing boundary which remain after hex generation.

Void Cleanup

Some additional work was required to allow the Hex and Tet portions of the algorithm to work well together. In complex geometries, the void left by the Hex generator is often very complex, with a very high surface to volume ratio, especially for relatively thin-section models (Figure 1).

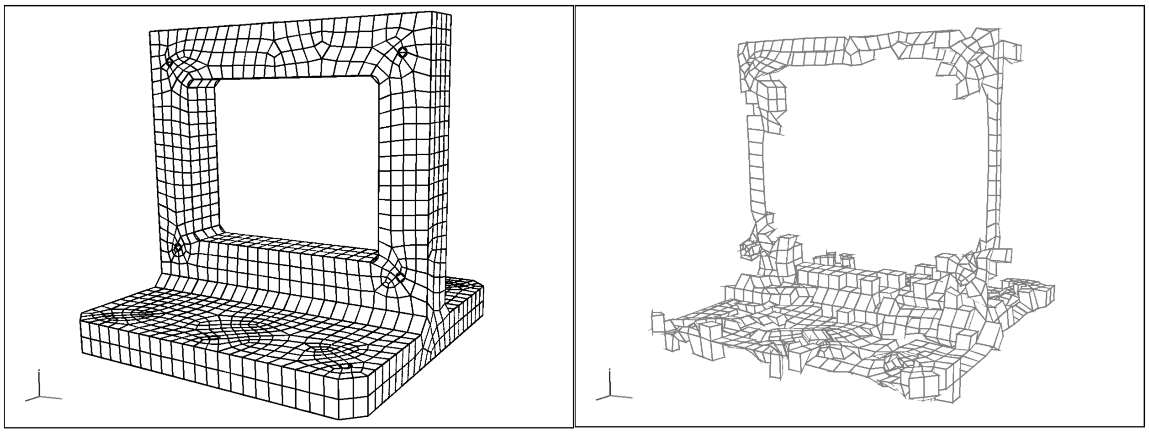

Figure 1: Winblock Model and Void Remaining after Hex Generation

The term “pancake void” was coined to describe the shape of void often seen when two advancing hex fronts would collide. Because of the fixed point density on the surface in addition to the complex geometry, these voids are very difficult for an automated tet-generator to fill with quality elements. To improve both robustness and tet quality, a void cleanup algorithm was devised to condition the void before tet generation.

The void cleanup algorithm attempts to improve the void by removing selected hex elements in such a way that the resulting void has improved properties for mesh generation. A distance tolerance was used to guarantee that any two non-adjacent quads on the void boundary would be separated by at least *d*, where *d* is some user-settable distance. This provides sufficient space for the tet generator to insert better elements. Additional Hex removal may also be performed given any of the following:

1. More than three faces of a single hex are on the void boundary.

2. An edge on the void boundary is connected to more than two faces on the void boundary.
3. The dihedral angle between any two faces on the void boundary is less than some minimum threshold value (typically about 45 degrees) or greater than some maximum value (typically about 315 degrees).
4. One of the interior angles of a boundary face is less than some minimum value (typically 45 degrees) or greater than some maximum value (typically 135 degrees).

Other criteria could also be used depending on the properties of the tet generation algorithm and on the desired results.

Testing the Hex-Tet Algorithm

Although some testing had been conducted on both the hex generator and the tet generator, a quantitive measure of the robustness of the combined algorithm was needed. To this end, a test suite of non-trivial problems was assembled for testing and evaluation. In all, seven models were chosen (see figures 2,3,8-11):

- Hook: This is a model of a real part, named for its general hook shaped appearance. The model contains several small and irregular facets.
- Ugly: Named either for its appearance or its behavior during meshing, this is actually a subset of the Hook model.
- Winblock: A blocky but difficult part with many through holes of widely varying diameter.
- TDDH: This model is best described as a soccer-ball shape with flat panels. It contains many angles between 120 and 150 degrees, which are very difficult for hexing algorithms to handle.
- Knee: A model of an artificial knee joint.
- Throw: A complex machined part, apparently part of a crankshaft assembly.
- Enterprise: A model of the well-known Starship, circa 1964.

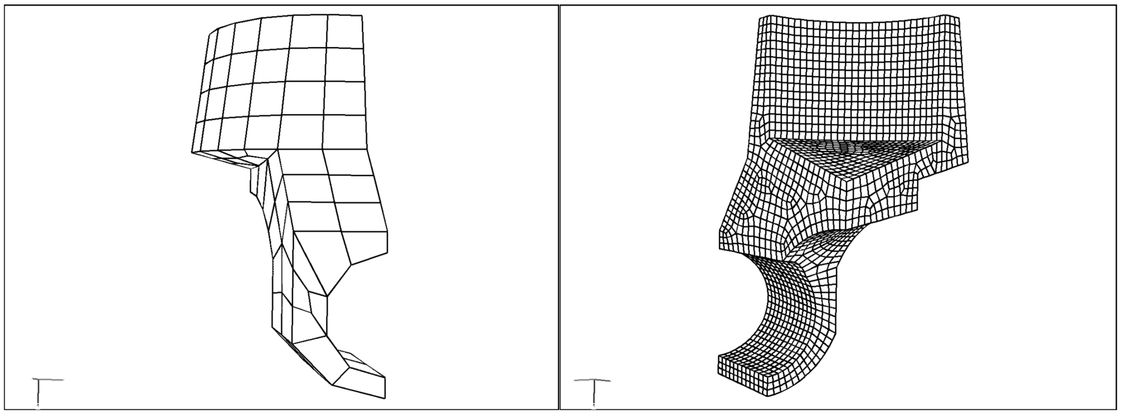

Figure 2: Hook Model, Low and High Resolutions

The models were chosen as examples of complex volumes for which current all-hex algorithms do not work well. Each model was tested over as wide a range of interval sizes as possible (from a maximum of 11 sizes for the Ugly model, to only two for the Enterprise). In general, the coarse mesh limit was governed by the coarsest mesh the current cubit surface meshing algorithms could produce for a given model with little or no human intervention. In general, the coarse meshes were too coarse to capture all of the detail in the model. The fine mesh limit was generally determined by the available dynamic memory limits of the computer used for these tests. Finer tests could be run on larger machines, but the range of intervals used for the models seems sufficient to draw meaningful conclusions on the capabilities of the Hex-Tet algorithm.

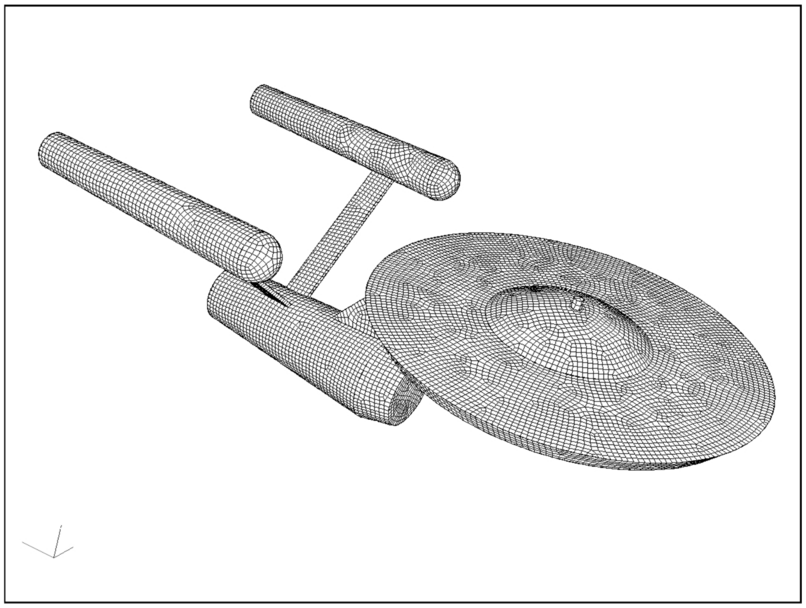
Figure 3: The Enterprise Model

Tests were performed for a total of 43 model/interval combinations with three transition methods for each. The table below summarizes the results for the test suite.

	Hook	Ugly	Winblock	Enterprise	Knee	TDDH	Throw	Totals
Surface Mesh Success	9/9	11/11	2/5	2/2	5/6	6/6	4/4	39/43
Hexing Success	9/9	11/11	2/2	2/2	2/5	6/6	3/4	35/39
Tet Success, trans: Two	9/9	11/11	2/2	2/2	2/2	6/6	2/3	34/35
Tet Success, trans: Four	8/9	10/11	0/2	0/2	0/2	5/6	2/3	25/36
Tet Success, trans: Pyramid	9/9	11/11	2/2	2/2	2/2	6/6	2/3	34/35

Figure 4: Results of Test Runs

Robustness

The success rate of the HexToVoid algorithm on a wide range of difficult models was surprising. For the Hex generation portion of the algorithm, the success rate was 35/39, or 90%. The failures were breakout problems, which showed up three times on the Knee problem and once on the Throw model.

The success rate for tetrizing the remaining voids was dependent on the transition method used. The success rate using the two-triangle transition was a very respectable 34/35 or 97%. This is excellent considering the very difficult voids the tet generator is asked to fill. Using the four-triangle transition, the rate was 25/36, or 70%. The large difference in success is probably due to the fact two good triangles can be formed from a skewed quad, whereas the four triangle transition on the same quad results in very thin triangles. The pyramid transition actually uses the two-triangle transition and tet generation, and then inserts the pyramid transition layer as a post-processing step. The success rate for the pyramid transition was identical to the two-transition at 97%.

For the models shown, the overall success rate for the Hex-Tet algorithm using the two-triangle or pyramid transition is 87%, while using the four-triangle transition results in 64% success for the tests performed. Interestingly, the HexToVoid algorithm performed remarkably well for large interval sizes (very coarse meshes).

Algorithm Effectiveness and Predictability

In addition to robustness, the testing included metrics which might indicate whether the algorithm is behaving sensibly. For example, we would expect that as resolution is increased, the percentage of the volume filled with hexes should increase, and the percentage of original boundary quads remaining after hex generation should decrease. Three of the models (Hook, Ugly, and TDDH) had enough data points for meaningful graphs. The graph for the Ugly model is shown below. The summary indication is that for very coarse models, the graphs jump around somewhat randomly. When the resolution becomes sufficient to accurately capture the details of the model, the behaviors then become monotonic and predictable. From this viewpoint, it appears that the hex generation portion of the algorithm behaves correctly and predictably as mesh size is scaled.

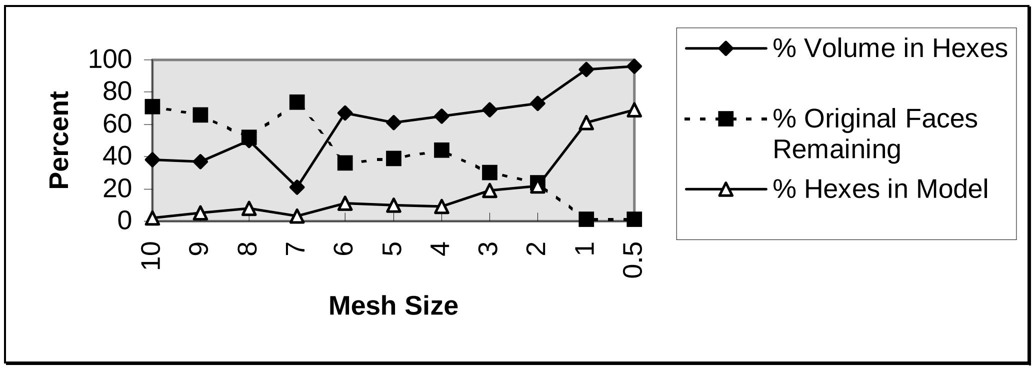

Figure 5: Algorithm Effectiveness For The Ugly Model

Algorithm Speed

Currently, the Hex-Tet algorithm is still very slow. The Hex generator, which takes most of the time, has been constructed for robustness, but as yet never modified for efficiency. The table below summarizes the average speed of the Hex and Tet generation portions of the algorithm, as well as overall speed.

Algorithm	Elements per Second
Hex Generation	9.84
Tet Generation	221.84
Overall	61.40

Figure 6: Average Timing Statistics for 43 Test Models

Element Quality

Idea of what constitutes a model of acceptable quality varies greatly from discipline to discipline and from individual to individual. For the purposes of this study, we chose to use a normalized average element jacobian as a reasonable indication of the quality of elements created. Of course, this does not mean the models would be acceptable for any given analysis problem. The jacobian was calculated as an average of the jacobian at each node, and was then normalized by dividing by the cube of the average edge length to give a normalized range of

$$-1.0 <= \text{jacobian} <= 1.0$$

As a computational minimum, all elements should have a jacobian greater than zero. Both the hex and tet generators currently prevent any element from being created which has a non-positive jacobian. However, global smoothing, which in general will improve the quality of elements, can sometimes also create negative jacobians. The table below shows the ranges of jacobians found for the test suite. The absolute minimum and maximum reflect, respectively, the lowest minimum and highest maximum found over all 43 models.

Parameter	Hex Elements	Tet Elements
Average Minimum Jacobian	0.306	0.062
Average Maximum Jacobian	0.983	0.655
Average Jacobian	**0.843**	**0.330**
Absolute Minimum Jacobian	0.007	0.001
Absolute Maximum Jacobian	1.000	0.710

Figure 7: Quality Metrics for The 43 Model Test Suite

The quality of hex elements generated overall was excellent, although there were isolated hexes in some models which would not be acceptable. The quality of the tet elements generated was not as good. This is partly attributable to the fact that the tet improvement algorithms planned for CUBIT (Joe:1995) have not yet been implemented.

Conclusions

The Hex-Tet algorithm implemented in CUBIT appears to be a useful addition to the CUBIT toolset. The algorithm handled a wide range of non-trivial models with robustness, producing meshes of acceptable quality in most cases.

Although many of the models used to test the algorithm are large and complex, we do not recommend using the Hex-Tet algorithm as a silver bullet to mesh an entire assembly with the push of a button. Rather, it is a useful tool to be used in the overall context of the CUBIT package, where models can be easily decomposed as necessary, and the best algorithm for each volume or sub-volume can be invoked as necessary. We do see it as highly useful, however, for those geometries which cannot be decomposed or meshed using other current algorithms, provided the intented solver can handle the resultant mixed element mesh.

Further work still needs to be done to improve the speed and efficiency of the hex generation portion of Hex-Tet to make it a faster and easier tool to use.

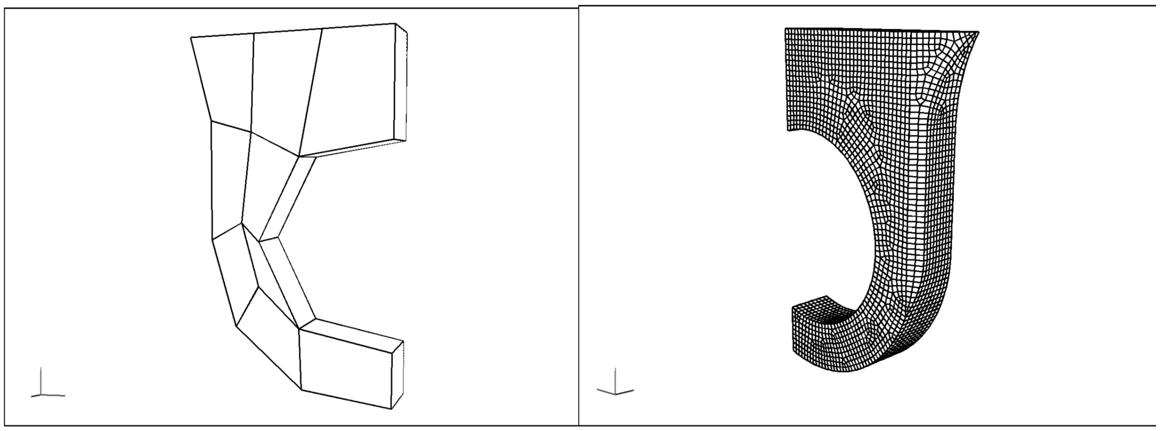

Figure 8: Ugly Model, Low and High Resolutions

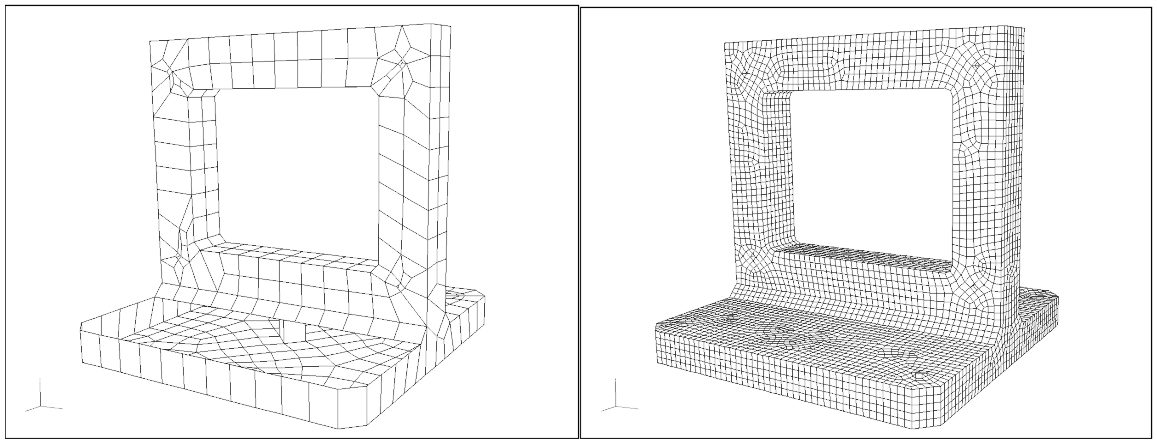

Figure 9: Winblock Model, Low and High Resolutions

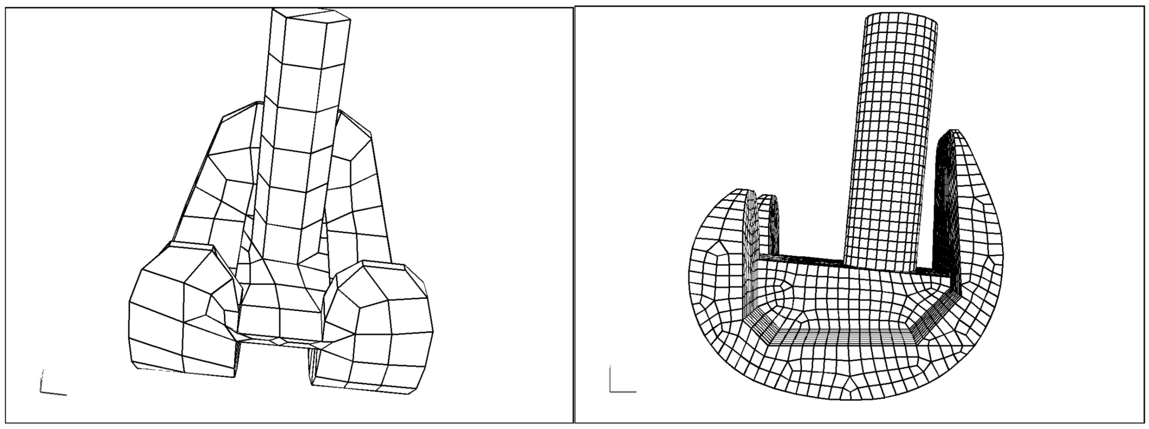

Figure 10: Knee Model, Low and High Resolutions

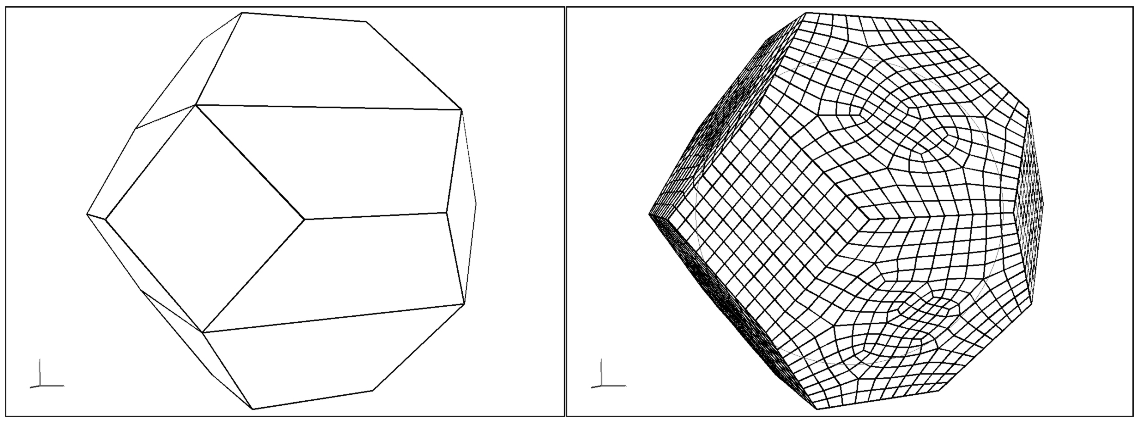

Figure 11: TDDH Model, Low and High Resolutions

Bibliography

Blacker, T. D. and Meyers, R. J., 1993, “Seams and Wedges in Plastering: A 3-D Hexahedral Mesh Generation Algorithm.” , Engineering With Computers, Vol 9, pp. 83-93.

Canann, S. A., “Plastering: A New Approach to Automated, 3-D Hexahedral mesh Generation”, American Institute of Aeronautics and Astronics, 1992.

Dewhirst, D. L., Vangavolu, S., and Wattrick, H., “The Combination of Hexahedral and Tetrahedral Meshing Algorithms”, Proceedings, 4th International Meshing Roundtable, Sandia National Laboratories, October, 1995, pp. 291-304.

Hipp, J. R. and Lober, R., “Plastering: All-Hexahedral Mesh Generation Through Connectivity Resolution”, Proceedings 3rd International Meshing Roundtable, Oct. 24-25, 1994, Albuquerque, New Mexico.

Hipp, J. R. and Clark, B., “Plastering Algorithm and New Knife Embedding and Face Insertion Implementation”, Proceedings 4th International Meshing Roundtable, Oct. 16-17, 1995, Albuquerque, New Mexico.

Joe, B., “Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations”, Siam Journal of Scientific Computing, Vol. 16, pp. 1292-1307, 1995.

Owen, S. J., Canann, S. A., and Sunil, S., “Pyramid Elements For Maintaining Tetrahedra to Hexahedra Conformability”, Trends in Unstructured Mesh Generation, AMD Vol 220, ASME 1997, pp. 123-129.

Stephenson, M. B., Canann, S. B., and Blacker, T. D., “Plastering: a New Approach to Automated, 3-D Hexahedral Mesh Generation - Progress Report 1”, Report SAND89-2192, Sandia National Laboratories, Albuquerque, New Mexico, 1990.

Tuchinsky, P. M., and Clark, B. W., 1997, “The ‘Hex-Tet’ Hex-Dominant Automesher: An Interim Progress Report” , Proceedings 6th International Meshing Roundtable, Sandia National Laboratories, October 1997, pp. 183-193.