Transactions of the Korean Nuclear Society Spring Meeting
Gyeongju, Korea, October 23-25, 2013

On the Polyhedral Mesh Generation

Sang Yong Lee*, Chan Eok Park, Jong Joo Sohn
Korea Power Engineering Company, Inc., 150 Deokjin-dong, Yuseong-gu, Daejeon, 305-353
**Corresponding author:sanglee@kopec.co.kr*

1. Introduction

A major advantage of polyhedral cells is that they have many neighbors (typically of order 10), so gradients can be much better approximated (using linear shape functions and the information from nearest neighbors only) than is the case with tetrahedral cells. Even along wall edges and at corners, a polyhedral cell is likely to have a couple of neighbors, thus allowing for a reasonable prediction of both gradients and local flow distribution. The fact that more neighbors means more storage and computing operations per cell is more than compensated by a higher accuracy.

Polyhedral cells are also less sensitive to stretching than tetrahedral cells. Smart grid generation and optimization techniques offer limitless possibilities: cells can automatically be joined, split, or modified by introducing additional points, edges and faces. Indeed, substantial improvements in grid quality are expected in the future, benefiting both solver efficiency and accuracy of solutions [1].

Most of the commercial computational fluid dynamics (CFD) codes have the capability to accept the polyhedral mesh. Even the recently developed safety analysis code such as CUPID [2] and SPACE [3] have the same capability. But they can't be fully utilized because there not many robust polyhedral mesh generator available in the world. Only STAR-CD mesh generator provides the capability of generating polyhedral mesh [4].

In principle, any Delaunay tetrahedral mesh generator can be used to generate polyhedral mesh using the duality of the Delaunay cell and the Voronoi region. But there are two practical problems to be resolved before using the duality principle. The genuine Delaunay cells can produce the Voronoi regions but these region may cross the boundary faces. Crossing boundary faces make it difficult to control the shape of the Voronoi regions. Another problem is arisen from the concave boundary faces. Any concave vertex point will produce the concave surface that cover the Voronoi cells.

In this paper, open source program TetGen [5], will be investigated to see whether it can be used to generate polyhedral mesh circumventing the above mentioned problems. A general introduction of TetGen will be made in section 2. The important properties of it will be presented. A detailed procedure to apply TetGen to produce the polyhedral mesh will be described in section 3. Several examples are shown in section 4. Conclusions are made in section 5.

2. A tetrahedral mesh generator TetGen

For a three-dimensional domain, defined by its boundary (such as a surface mesh), TetGen generates the boundary constrained (Delaunay) tetrahedralization, conforming (Delaunay) tetrahedralization, quality (Delaunay) mesh. The latter is nicely graded and the tetrahedra have circum-radius to shortest edge ratio bounded. For a three-dimensional point set, the Delaunay tetrahedralization and convex hull can be generated. The code, written in C++, may be compiled into an executable program or a library for integrating into other applications. All major operating systems, e.g. Unix/Linux, MacOS, Windows, etc, are supported. It is open source code under the General Public License [6].

A boundary conforming Delaunay mesh is defined as a mesh T of domain X such that

(i) every simplex of T is Delaunay, and
(ii) every simplex contained in $|\partial X|$ Gabriel.

The property Gabriel is defined as; a simplex σ is Gabriel, if no other point of σ lies inside the diametrical sphere of σ, i.e., the smallest circumscribed sphere of σ. The Gabriel property ensures that the Voronoi region be inside of the boundaries as shown in Figure-1 for 2-dimensional space..

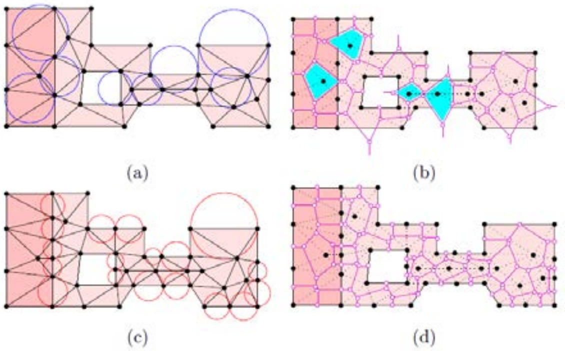
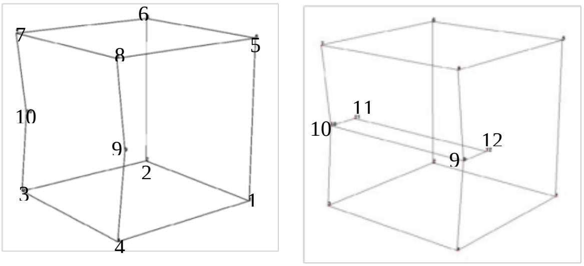


Figure-1: (a) A conforming Delaunay mesh T of a two-dimensional PLS X. X has two regions (shown in different colors) separated by an internal edge. (b) The Voronoi partition of X obtained from taking the dual of T . Some Voronoi edges (highlighted in cyan) cross the internal boundaries are not orthogonal. (c) A boundary conforming Delaunay mesh T' of X. Some diametric spheres of the boundary edges in T' are shown. (d) The corresponding Voronoi partition of X.

3. Applying TetGen to generate polyhedral mesh

In principle, for the convex domain, Voronoi regions generated by TetGen can used as polyhedral mesh if those regions facing boundary is capped by the boundary faces. However, the cap generated at the concave boundary points will not become convex at all. And they can't be treated as cell bounding faces. Therefore, some special arrangements should be made at those points.

A solution to this problem is to insert internal boundary faces along the concave points so that the concave regions become locally convex. Figure-2 shows this procedure. Original block has a concave line along the points 9 and 10 (a). This concave line can be treated as two separated (locally) convex domains by inserting an internal boundary face (9-10-11-12) along the concave line (b). The Gabriel property of TetGen at the inserted internal boundary should make it sure that well separated Voronoi regions are created at both side of the internal boundary.



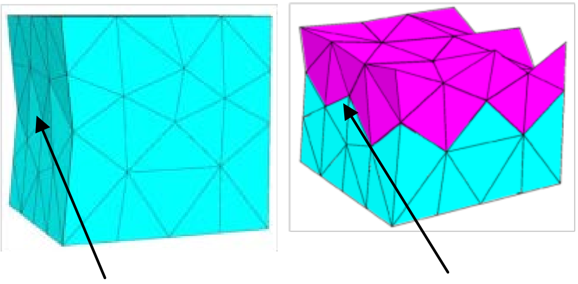
a) Concave line 9-10

b) Inserted internal boundary 9-10-11-12

Figure-2 Making locally convex region of along concave line

4. Generated Voronoi cells

Figure-3 shows the Delaunay mesh for the domain with concave surface, i.e., Figure-2.a without inserting internal boundary surface. As one can see in the Figure-3.a, the boundary shape is not fully respected. As a result of this, Voronoi cell generated at this point will have concave cap, which is not a proper cell.

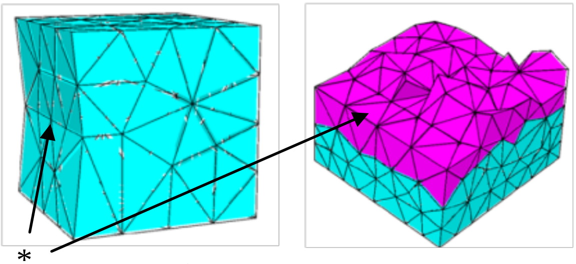


a) Boundary non-conformity

b) Non-convex Voronoi cell

Figure-3 Mesh for the concave face without internal BD face

Figure-4 shows the boundary conforming Delaunay mesh for the locally convex block presented in Figure-2.b. Segment 9-10 that constitutes the inserted internal boundary face force the TetGen to construct tetrahedrons along with it (see *).



a) Boundary conformity

b) Convex Voronoi cells

Figure-5 Boundary conformed Delaunay mesh for the box.

Figure-4.b shows the cut view around the internal boundary surface. As already suggested, a row of tetrahedral cells are generated on both top and bottom of the internal boundary face. As a result of this, Voronoi cells around this area will have convex caps. Therefore, proper polyhedral cells can be generated.

One more problem with handling polyhedral mesh system is the fact that not many post processors are available for them. Most commercial mesh handler support only regular cells that have fixed connectivity such as tetrahedron, hexahedron, prism and pyramid.

Fortunately, it was found that ParaFoam from openFoam [7] can handle the polyhedral cell. Figure-7 shows the two polyhedra depicted by ParaForm.

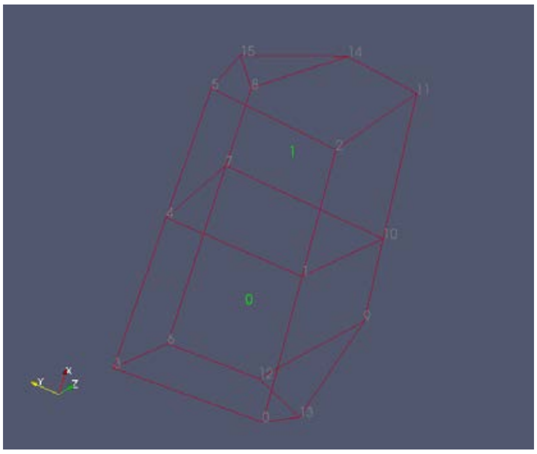


Figure-7 Two polyhedra processed by ParaFoam..

5. Conclusion

A procedure to generate polyhedral mesh is devised for the TetGen code using the internal boundaries. The Gabiel property of TetGen code that Voronoi regions generated with this procedure can be used as polyhedral cells. ParaFoam will be a candidate for the post-processing tool. An interfacing program between TetGen and ParaFoam is being developed.

REFERENCES

[1] http://www.plmmarketplace.com/upload/Temp/The_Advantage_of_polyhedral.pdf, "The advantage of polyhedral meshes", M. Peric and S. Ferguson, CD-adapco.

[2] J. J. Jeong, et. al., "The Cupid code development and assessment strategy", Nuclear Engineering and Technology, vol. 42 no.6 December 2010.

[3] S. J. Ha, C. E. Park, K. D. Kim, and C. H. Ban, "Development of the SPACE Code for Nuclear Power Plants", Nuclear Technology, Vol. 43, No. 1, 2011.

[4] Meshing user guide, STAR-CD Version 4.02, CD-adapco.

[5] http://tetgen.berlios.de/, TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator, Version 1.4, User's Manual, January 16, 2006, H. Si.

[6] http://en.wikipedia.org/wiki/GNU_General_Public_License

[7] http://www.openfoam.com/