Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 00 (2015) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Polyhedral Mesh Generation and A Treatise on Concave Geometrical Edges

Sang Yong Lee[a],*

[a]*KEPCO International Nuclear Graduate School, 658-91 Haemaji-ro Seosaeng-myeon Ulju-gun, Ulsan 689-882, Republic of Korea*

Abstract

Three methods are investigated to remove the concavity at the boundary edges and/or vertices during polyhedral mesh generation by a dual mesh method. The non-manifold elements insertion method is the first method examined. Concavity can be removed by inserting non-manifold surfaces along the concave edges and using them as an internal boundary before applying Delaunay mesh generation. Conical cell decomposition/bisection is the second method examined. Any concave polyhedral cell is decomposed into polygonal conical cells first, with the polygonal primal edge cut-face as the base and the dual vertex on the boundary as the apex. The bisection of the concave polygonal conical cell along the concave edge is done. Finally the cut-along-concave-edge method is examined. Every concave polyhedral cell is cut along the concave edges. If a cut cell is still concave, then it is cut once more. The first method is very promising but not many mesh generators can handle the non-manifold surfaces especially for complex geometries. The second method is applicable to any concave cell with a rather simple procedure but the decomposed cells are too small compared to neighbor non-decomposed cells. Therefore, it may cause some problems during the numerical analysis application. The cut-along-concave-edge method is the most viable method at this time. In this paper, discussions are presented with the boundary conforming Delaunay primal mesh to reduce the complexity that may be expected at the boundary. Polygonal prism mesh generation at the viscous layer is also investigated.

Keywords: polyhedral mesh generation; dual mesh; concave polyhedral cell; non-manifold elements insertion method; conical cell decomposition/bisection method; cut-along-concave-edge method; polygonal prism mesh in viscous layer;

* Corresponding author. Tel.:+821088055925; fax: +82527127307.
E-mail address: sangleey@kings.ac.kr

1. Introduction

Mesh generation for the solution domain is required for solving partial differential equations. In general, tetrahedral, hexahedral or polyhedral elements are used. Among them, tetrahedral mesh generators are widely used. However, the regular tetrahedral elements are not preferred for non-linear problems because they exhibit lower accuracy and artificial stiffness [7]. Furthermore, the reducing rate of the residual norm during iterations is substantially lower on the tetrahedral mesh than the polyhedral mesh [1]. Several authors have investigated the polyhedral mesh generation methods [1, 2, 8, 14, 15]. Garimella [2] investigated the dual mesh method in detail. In this method, the polyhedral dual mesh is constructed from the tetrahedral primal mesh.

In this study, I used the dual mesh method to generate the polyhedral mesh. The generic issue of concern in the dual mesh method is the concave polyhedrons created along the geometrical edges of the solid model [2]. I am investigating three techniques to eliminate the concavity of polyhedrons along the geometrical concave edges: the non-manifold elements insertion method, the conical cell decomposition/bisection method and the cut-along-concave-edge methods as explained in section 2. The issue of the polygonal prism mesh generation at the viscous layer is also investigated and presented in section 3. Then, conclusions are made in section 4.

At this point, some definitions are presented. A simplex in a mesh is *Delaunay* [9] if it has a circumscribed sphere such that no other point of mesh lies inside it. Moreover, it is *Gabriel* [10] if no other point of the mesh lies inside the diametrical sphere of the simplex. A *boundary conforming Delaunay mesh* of a domain is a mesh with the following properties; (i) every simplex is Delaunay, (ii) every simplex in the boundary of the mesh is Gabriel. The resulting mesh of the dual mesh generation for general tetrahedral meshes is a polyhedral mesh with straight edges but possibly curved faces. For the input tetrahedral mesh of a boundary conforming Delaunay mesh, however, the resulting mesh is a Voronoi [11] mesh with planar faces. Although the procedures presented in this paper are applicable to general input tetrahedral mesh, only the input of the boundary conforming Delaunay primal tetrahedral mesh is used to reduce the complexity that may occur at the boundary.

The dual mesh computation algorithm from a valid primal tetrahedral mesh is well studied [2, 8]. Here, I re-state it as algorithm-I for the following discussions:

Algorithm-I. Algorithm to create dual mesh.

```
1. Create a dual vertex at a central point in each primal mesh region.
2. Create a dual vertex at a central point in each primal mesh face classified on the boundary.
3. Create a dual vertex at mid-point of each primal mesh edge classified on a model edge.
4. Create a dual vertex at each primal mesh vertex classified on a model vertex.
5. Create an interior dual face corresponding to each primal mesh edge classified on the interior of the domain.
6. Create one or more interior dual faces corresponding to each primal mesh edge classified on the boundary.
7. Create one or more boundary dual faces corresponding to each boundary primal vertex.
8. Create one dual region corresponding to each interior primal vertex.
9. Create one or more dual regions corresponding to each boundary vertex.
```

Several open source softwares are used for this study. Salome-Platform [12] is used as a geometrical model constructor and mesh generator. OpenCascade [13] is the built-in solid modeler of the Salome-Platform. TetGen [3, 4] is used to generate a boundary conforming tetrahedral mesh. It can handle only solid models with flat facets. For solid models with curved surfaces, Netgen [5] is used for generating surface meshes. The surface mesh generated by Netgen is usually a boundary conforming Delaunay mesh. Therefore, the primal tetrahedral mesh of the geometrical model with curved surfaces is generated by TetGen with the surface mesh input from Netgen in this study. For the geometrical model with flat surfaces, surface mesh can be generated by either TetGen or Netgen.

2. A treatise on the concave edges

2.1. Non-manifold surface insertion method

This method relies on the capability of the mesh generator and/or the solid modeler to handle non-manifold surfaces. If it is possible to insert a non-manifold surface along the geometrical concave edge, that face can be used to get rid of the concavity problems of the dual mesh. TetGen, for example, has the capability to handle the non-manifold faces and it can generate a boundary conforming Delaunay tetrahedral mesh.

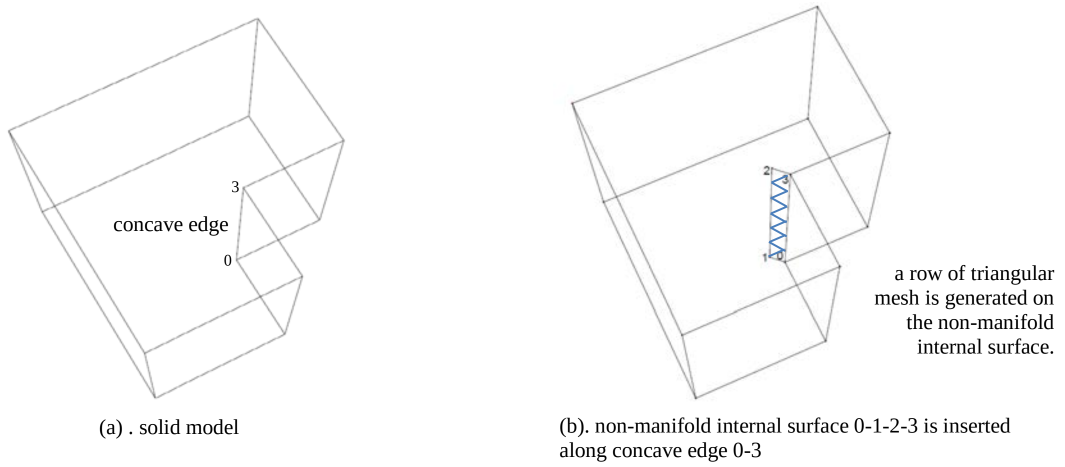

3
concave edge
0
(a) . solid model
2
3
1
0
a row of triangular mesh is generated on the non-manifold internal surface.
(b). non-manifold internal surface 0-1-2-3 is inserted along concave edge 0-3

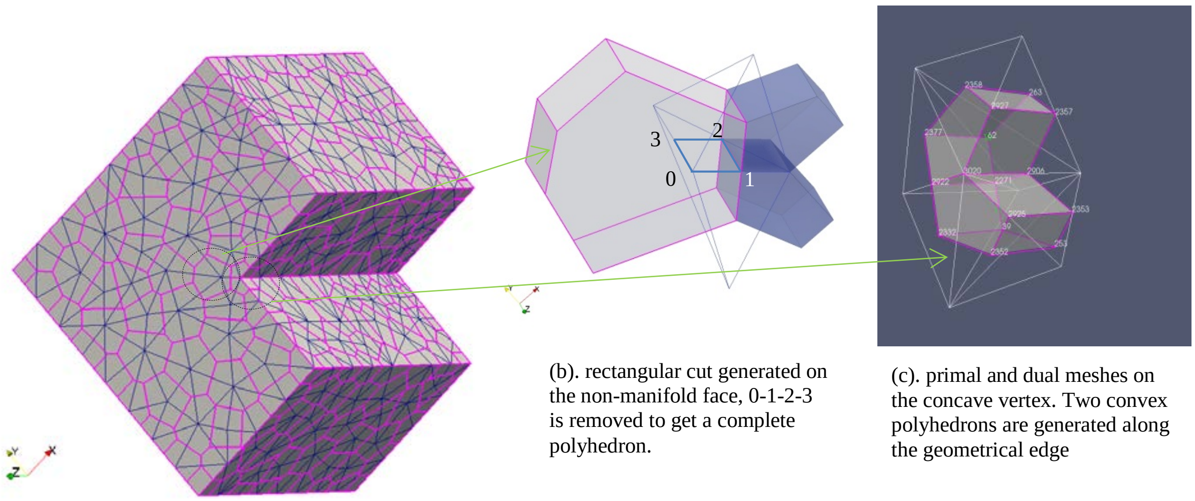
Fig. 1. cavityClipped model with a non-manifold surface.

A solid model, cavityClipped in Fig. 1(a), has a concave edge (0-3). Since the model is bounded by flat surfaces, the model can be handled by TetGen with the planar straight line graph (PSLG) input without any help from solid modeler such as the Salome-Platform. The PSLG input for the geometrical model, cavityClipped, of Fig. 1(a), is easily modified to include a non-manifold facet 0-1-2-3 for the mesh generator TetGen. A non-manifold face 0-1-2-3 is inserted by creating the non-manifold vertices 1 and 2 on the surfaces, by creating non-manifold edges 0-1, 2-3 and 1-2, and, by creating the face 0-1-2-3. The surface mesh generator creates triangular surface meshes on the solid surfaces as well as the internal non-manifold surfaces. Control has to be exercised to generate *a row* of triangles on the non-manifold surface along a concave edge as in Fig. 1(b).

After the volume mesh generation is completed by TetGen, algorithm-I is applied to obtain a polyhedral mesh. Most of the polyhedrons, even those along the concave edge 0-3, are convex except ones along on the internal non-manifold edge, 1-2, in Fig. 1(b). Two convex polyhedrons are produced at the concave vertex such as vertex 3 in Fig. 1(b), one for each side of the non-manifold face as shown in Fig. 2(b) and (c). The polyhedrons of same type are also shown in Fig. 3. But, polyhedrons generated on the non-manifold edge 1-2 in Fig. 1(b) are cut by the non-manifold face. One of the examples is shown in Fig. 2. The polyhedron located at the neighbor of the concave vertex 2 is shown in Fig. 2(b), in which it is cut by a rectangular face, 0-1-2-3. Therefore, it is required to remove this cut face to “heal” the cut polyhedron. The healed polyhedrons are shown also in Fig. 3. Naturally, the algorithm to create a dual mesh with non-manifold internal surface should be modified to add one more step to algorithm-I:

Algorithm-II. Algorithm to create dual mesh with non-manifold internal surface.

Follow algorithm-I for steps 1 to 9.
10. Remove elements that cut the polyhedrons along the internal non-manifold edges.

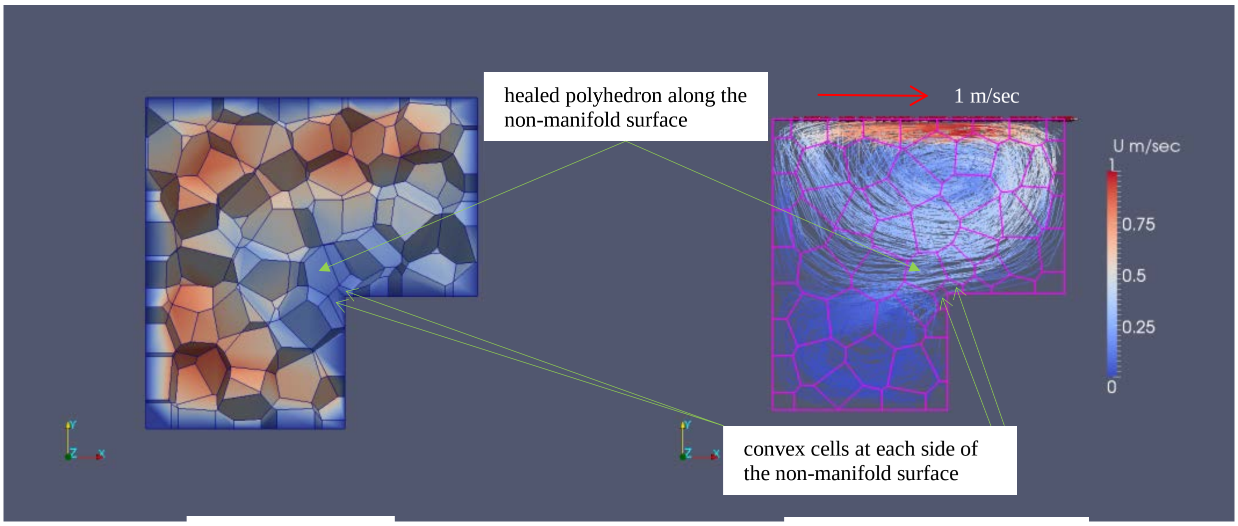


(a). primal and dual meshes

(b). rectangular cut generated on the non-manifold face, 0-1-2-3 is removed to get a complete polyhedron.

(c). primal and dual meshes on the concave vertex. Two convex polyhedrons are generated along the geometrical edge

Fig. 2. Construction of convex cells on cavityClipped model.


healed polyhedron along the non-manifold surface
1 m/sec
U m/sec
1
0.75
0.5
0.25
0
convex cells at each side of the non-manifold surface

(a). clipping view

(b). stream lines of icoFoam

Fig. 3. Polyhedral mesh of cavityClipped model.

The X-Y plane clipping view of the polyhedral mesh generated is shown in Fig. 3(a). The validity of the mesh can be confirmed with the application of any computational fluid dynamics solver. The incompressible fluid flow solver, icoFoam [6], is used in this study. The cavityClipped problem is a three dimensional version of the original two-dimensional version of OpenFoam [6]. The fluid at the top lid of the cavity is moved with constant velocity of (1.0, 0.0, 0.0) m/sec. Zero velocities are given to the rest of the faces. The pressure gradients at the surfaces are set to zero. The PISO [6] solver is applied. The calculation reaches the steady state very easily. The stream lines of the two point sources are shown in Fig. 3(b). They look reasonable. It is concluded that the polyhedral mesh generated is valid. The results are not assed quantitatively since the confirmation of mesh validity is the purpose of the calculation.

The above example is for the flat face solid model. For the solid model with curved surfaces, a geometrical modeler such as Salome-Platform may be used to insert a non-manifold surface along the concave edge. If re-meshing capability is available in the mesh generator, it may be used to insert a row of triangular mesh along the concave edge.

2.2. Conical cell decomposition/bisection method

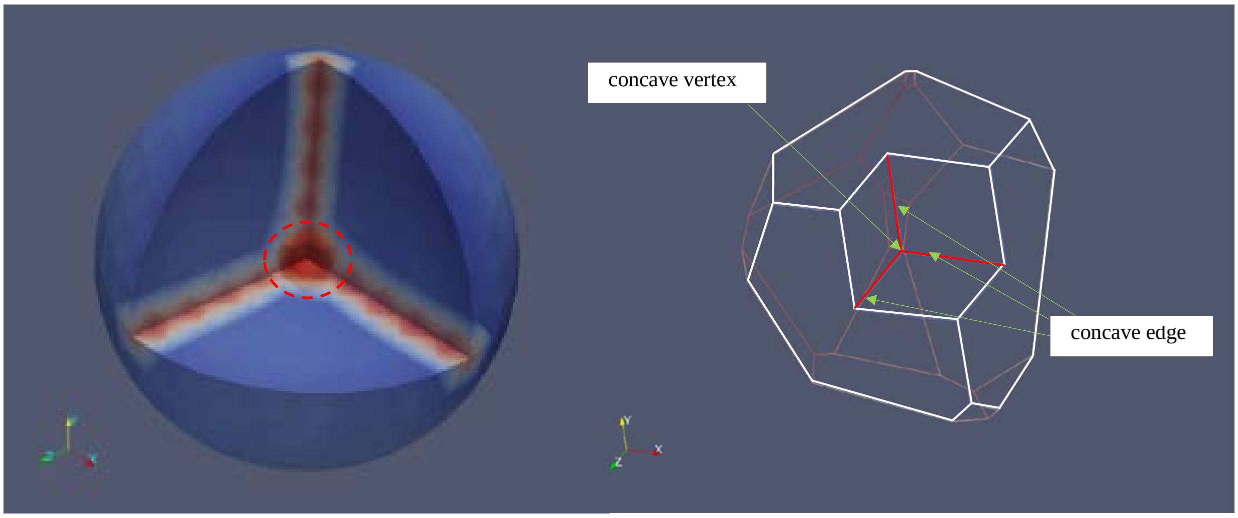
In this method, any concave polyhedral cell is decomposed into polygonal conical cells first, with the polygonal primal edge cut-face as the base and with the dual vertex on the boundary primal vertex as the apex. Then, bisecting the concave polygonal conical cell along the concave edge produces two convex polygonal conical cells. Some definitions of the cell entities are helpful for the following illustrations. All of the following definitions refer to mesh elements in a cell. A concave vertex is a dual vertex on the primal vertex on the concave model edge. A star vertex is a vertex other than the concave vertex. A star edge is an edge that connects the concave vertex to star vertex. A link edge is an edge that connects a star vertex to other star vertex. A star face is a face that is constructed by the concave vertex and two star vertices. In other words, it is a face made by two star edges and a link edge. A bisecting vertex is a vertex created on the internal edge of the concave interior dual face to bisect it. A bisecting edge is an edge that connects a bisecting vertex with star vertex on concave edge. A bisecting star edge is an edge that connect concave vertex to the bisecting vertex. A bisecting face is a face that is constructed by the bisecting edge, the bisecting star edge and the concave edge. A bisected face is a face created by bisecting the base face of a concave conical cell. With these definitions, the cell decomposition/bisection method is described by following the algorithm:

Algorithm-III. Algorithm to decompose the boundary concave cell into the convex conical cells.

1. Create a star edge.
2. Create a star face.
3. Create a conical cell with an interior dual face and star faces.
4. Create a bisecting vertex.
5. Create a bisecting edge.
6. Create a bisecting face.
7. Create a bisected face.
8. Construct a bisected convex conical cell.

The process of the cell decomposition/bisection method is illustrated with the geometrical model, the octant-cut-sphere, in Fig. 4. This model is a simple sphere with an octant of it cut out. The primal and dual mesh generation leads to the polyhedral mesh with concave cells along the concave model edges. The cell generated at the primal vertex that is created by three concave edges is shown in Fig. 4(a) and (b). During the step of star edge creation, edges are constructed from concave vertex to the vertices created on the circumcenter of the primal triangular face on the boundary surface (solid yellow lines in Fig. 5(a)). And star edges are also constructed from the concave vertex to every cell vertex (dashed yellow lines in Fig. 5). Triangular faces are produced with those created edges in step 1 and with the connected link edge. Conical cells can be made with those triangular faces as side faces, with the polygonal primal edge cut-faces as the base faces and with the concave vertex as their apex as shown in Fig. 5(a). Three bisecting vertices are created (Fig. 5(b)). Three bisecting edges are created (solid blue edges in Fig. 5(b)). Three bisecting star edges are created (dashed blue edges in Fig. 5(b)). Three bisecting faces are created with concave edges, bisecting edges and the bisecting star edges. Two bisected faces are created by collecting link edges and a bisected link edge that are on the base polygon of the concave conical cell. Collecting star faces and the bisected face a convex bisected conical cell can be constructed (Fig. 5(b)).

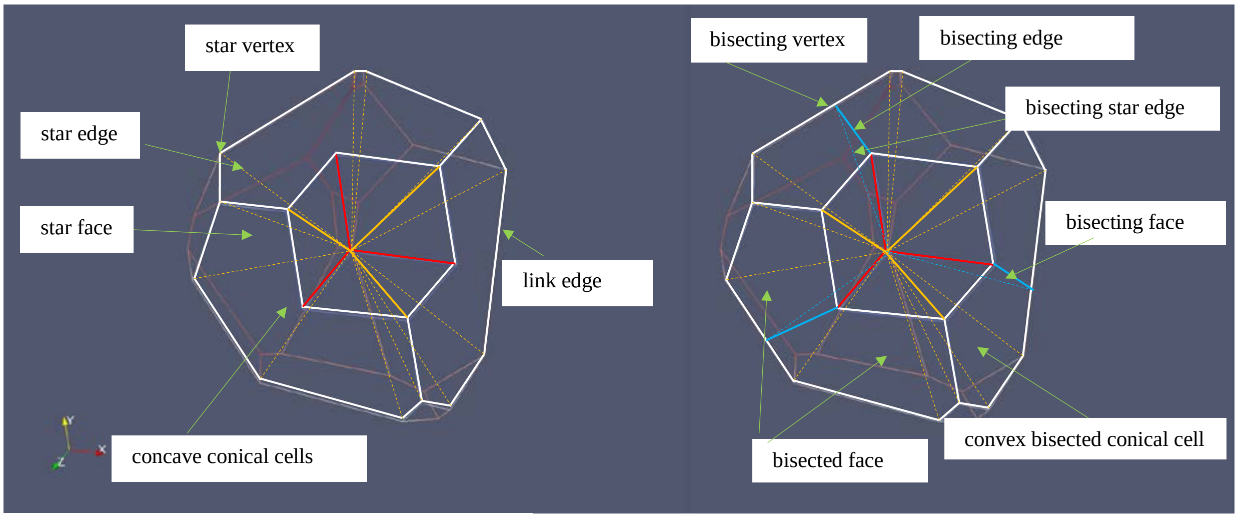
The polyhedral mesh generated is shown in Fig. 6. It is shown that all of the concave polyhedral cells are cut into convex conical cells. Especially, the clipping view (Fig. 6(b)) shows the convex conical cells along the model edges very well. It is noted that this method is very straightforward and can be applicable to any shape of concave polyhedral cell.



(a). cell with three concave edges in the octant-cut-sphere. (in red circle)

(b). concave vertex and edges of a cell with three concave edges.

Fig. 4. The octant-cut-sphere model and the cell with three concave edge.

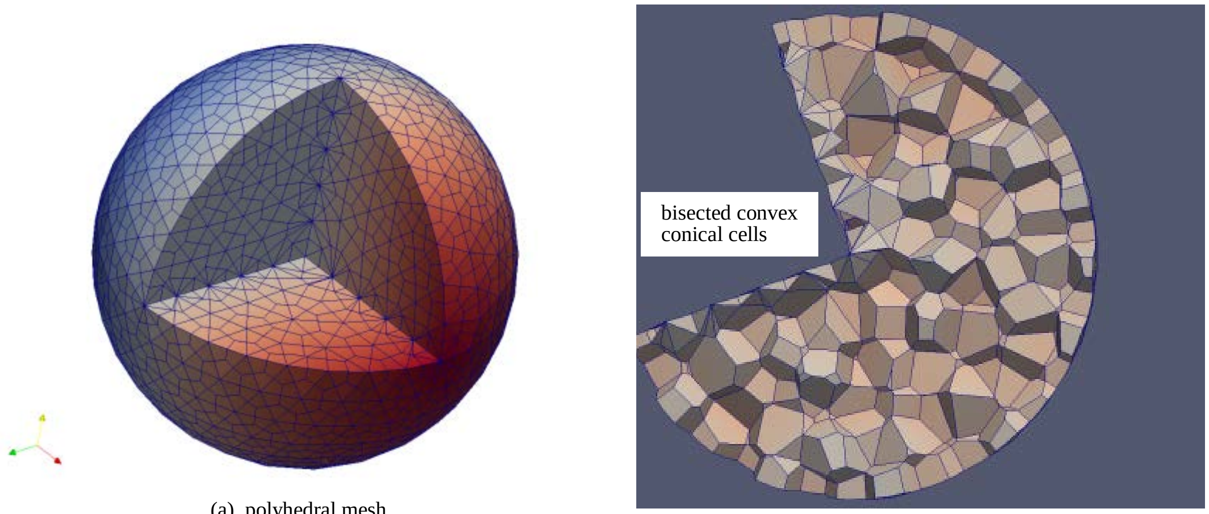


(a). conical cell decomposition by creating edges (yellow).

(b). bisection of concave conical cells (blue).

Fig. 5. Conical cell decomposition and bisection process.

As one can see in Fig. 6(b), the bisected conical cell is much smaller than the neighbor intact convex polyhedral cell. The size of the conical cell is, typically, an order of magnitude smaller than that of the intact convex polyhedral cell. The big difference may be a potential problem in the numerical application of the mesh.



(a). polyhedral mesh (b). clipping view of polyhedral mesh

Fig. 6. Polyhedral mesh of the octant-cut-sphere.

2.3. Cut-along-concave-edge method.

To avoid producing cells that are too small, the concave cell is only cut along concave edges in this method. The cutting method depends on the location of the concave vertex. Some definitions are made for the following discussions. A concave edge vertex is defined as a dual vertex on the primal vertex on the model edge. A middle point vertex is a dual vertex on the middle point of the primal edge.

A concave edge cell is defined to be a concave cell that is located on the model edge. A concave vertex cell is defined as a cell located at the vertex of a model. The angle of the concave edge is an angle made by two edges of the face generated by algorithm-I-6 that meet at a middle point vertex. For example, the angle made by two edges, e1 and e2, is the angle of the concave edge e0 of Fig. 8(a). A bisecting vector is a vector that bisects the angle of the concave edge. Generally, a cutting plane is defined by a bisecting vector and the direction vector of the concave edge with a middle point vertex. A meeting line is the line at which two cutting planes meet. A meeting vertex is a vertex a meeting line meets the boundary surface. With these definitions, the cut-along-concave-edge method is described as following algorithm:

Algorithm-IV. Algorithm of cut-along-concave-edge method.

1. For concave edge cells, cut along concave edges on the model edge.
2. For the concave vertex cell with one concave edge, cut along the concave edge.
3. For the concave vertex cell with two concave edges;
   3.1. Construct cutting planes for two concave edges.
   3.2. Find the meeting line of cutting planes.
   3.3. Cut along a concave edge to meeting line.
   3.4. Cut along next concave edge to meeting line.
   3.5. Construct one convex cell and one concave cell.
   3.6. Construct cutting plane along edge on the meeting line.
   3.7. Cut along the concave cell again.
4. For the concave vertex cell with more than two concave edges;
   4.1. Pick two neighboring concave edges.

4.2. Construct cutting planes for neighboring two concave edges.
4.3. Find the meeting line of cutting planes.
4.4. Cut along a concave edge to meeting line.
4.5. Cut along next concave edge to meeting line.
4.6. Construct one convex cell and one concave cell.
4.7. Construct cutting plane with the edge on the meeting line and the next concave edge in the concave cell.
4.8. Cut along the concave cell again.
4.9. If there are more concave edges then repeat steps 4.7 and 4.8.

The application of this algorithm is demonstrated in Fig. 7 and 8 with the geometrical model, octant-cut-sphere-block-merge. The concave vertex cell with two concave edges in Fig. 7(a) is taken as an example of the algorithm-IV. It is shown in Fig. 8(a) before the faces on the same boundary face are merged. The edge e1 and e2 make the angle of the concave edge e0. The sum of the normalized direction vectors for two edges e1 and e2 results in the bisecting vector of the concave edge e0. This vector and the concave edge constitute the cutting plane of the first cut. Similarly, the sum of the normalized direction vectors for two edges e4 and e5 results in the bisecting vector of the concave edge e3. This vector and the concave edge constitute the cutting plane of the second cut. Cutting by two cutting planes produces two cells, a convex cell (convex cell-0 of Fig. 8(b)) and a concave cell that has to be bisected again.

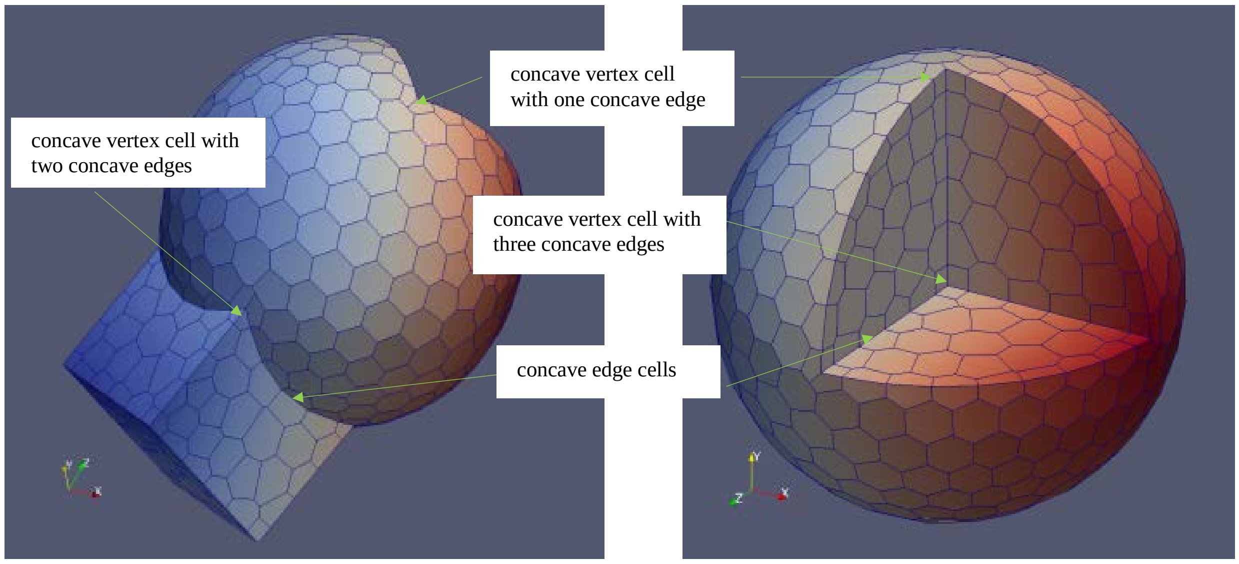
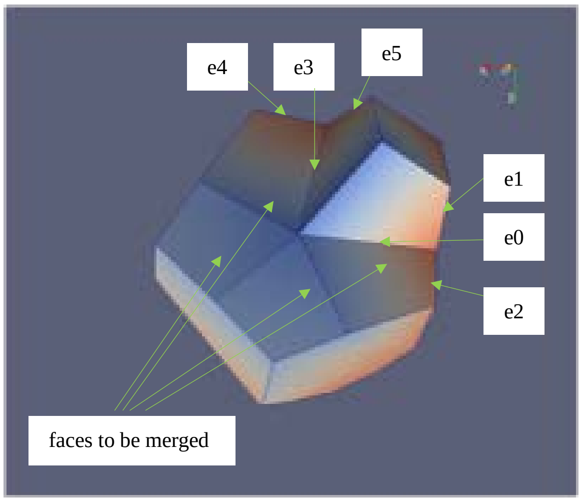
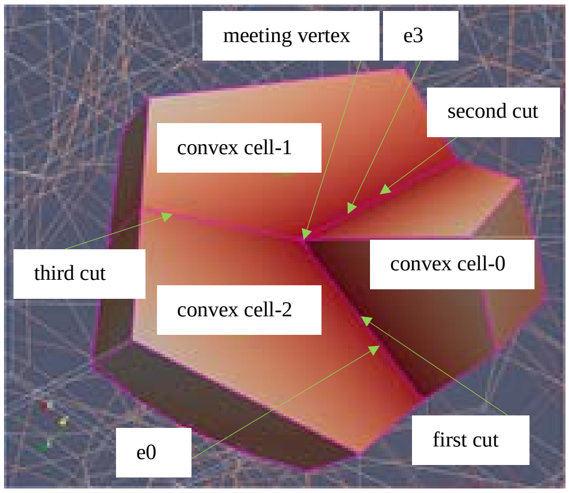

concave vertex cell with one concave edge
concave vertex cell with two concave edges
concave vertex cell with three concave edges
concave edge cells
(a). polyhedral mesh view 1. (b). polyhedral mesh view 2.

Fig. 7. Polyhedral mesh of the octant-cut-sphere-block-merge.

Two planes meet at a meeting line that passes the meeting vertex. The sum of the normalized direction vectors for two concave edges e0 and e3 results in the bisecting vector of the concave edge on the meeting line. This vector and the concave edge on the meeting line constitute the cutting plane of the third cut. The third cut produces two convex cells, convex cell-1 and convex cell-2. The concave vertex polyhedral cell with two concave edges is cut into three convex polyhedral cells (Fig. 8(b)). As one can see on the Fig. 7, all of the concave vertex cells and the concave edge cells are properly cut into convex polyhedral cells.



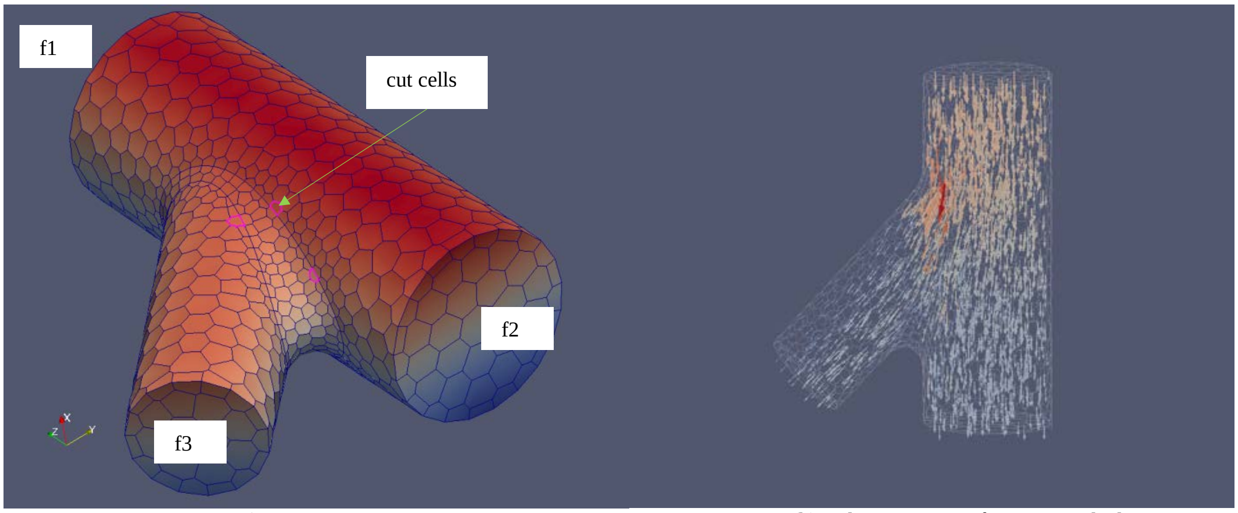
(a). merge faces on the same model face.



(b). three cuts for the cell with two concave edges

Fig. 8. Cutting process of the concave vertex cell with two concave edges.

As shown in Fig. 9, the mergedPipes model is constructed by merging one smaller pipe into the large pipe and by adding a fillet to smooth the pipe junction. There are two concave edges along the fillet but no concave vertices are in this model. The polyhedral mesh of the model is shown in Fig. 9(a). Some of the cut cells are shown in red along the concave edges in the same figure.



(a). surface with edge view.

(b). velocity vectors of icoFoam calculation

Fig. 9. Polyhedral mesh of mergedPipes model with fillet.

The validity of the polyhedral mesh generated by cut-along-concave-edge method can be confirmed with the application of any computational fluid dynamics solver. The incompressible fluid flow solver, icoFoam[6] is chosen. The boundary condition for this problem is simple. A constant pressure boundary condition is given to faces f2 and f3. Constant velocity condition is set to f1. Zero velocities are given to rest of the faces. The pressure gradients at the surfaces are set to zero. The PISO [6] solver is applied. The calculation reaches to the steady state very easily. It is therefore concluded that the polyhedral mesh generated is valid. The vector plots of velocity are shown in Fig. 9(b) and they look reasonable. No additional efforts are needed to investigate the results in detail since the confirmation of mesh validity is the purpose of the calculation.

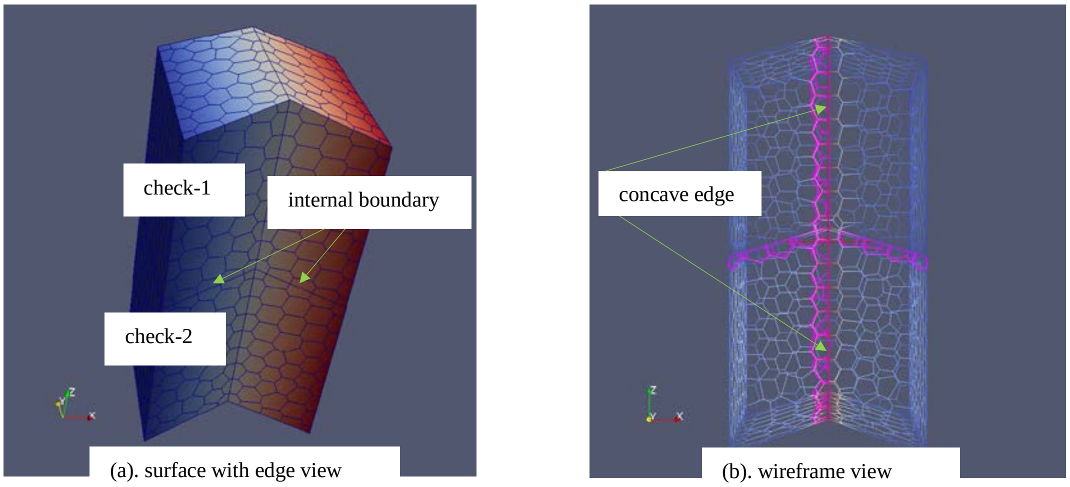


Fig. 10. Polyhedral mesh of the mergedChecks – internal boundary test.

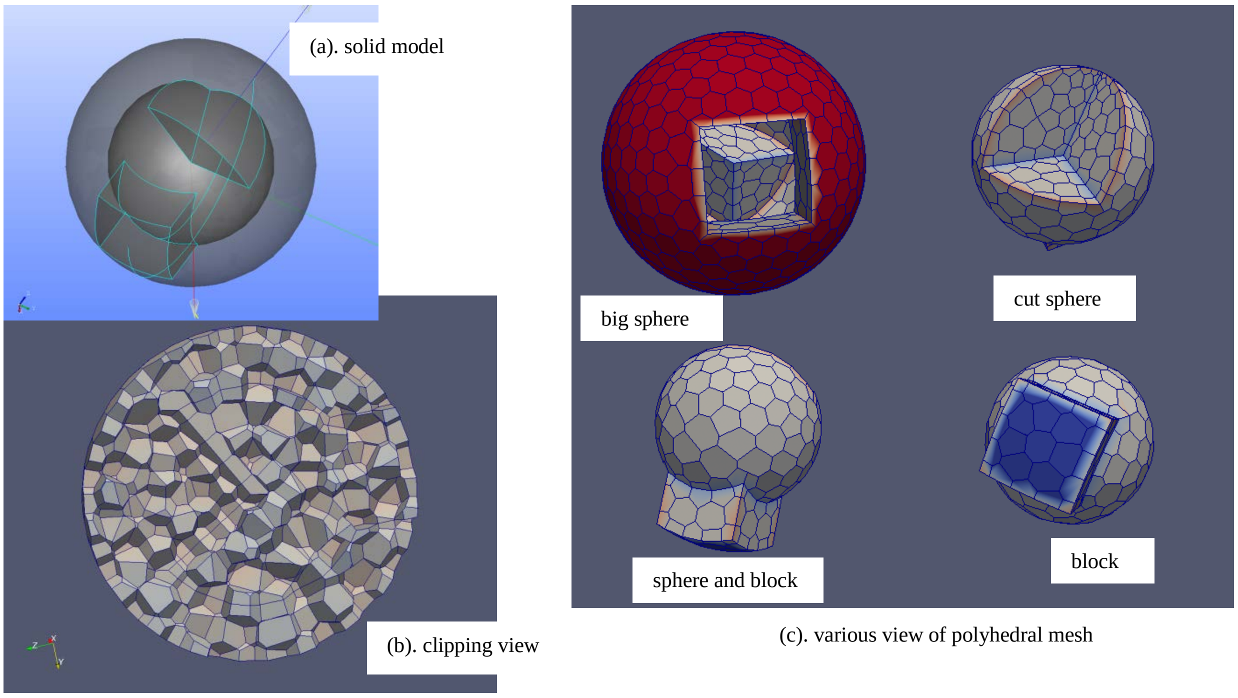


Fig. 11. Polyhedral mesh of the model A-Shape.

The capability to handle the non-manifold surface such as at the internal boundary is tested with two geometrical models. In the mergedChecks model (Fig. 10(a)), 2 “check” shape blocks are merged to produce internal surfaces.

Each block has two concave edges. Therefore, after merging, this model has two internal boundary surfaces, and four concave edges. One of the concave edges is an internal concave edge. The polyhedral mesh of the mergedChecks is shown in Fig. 10(a) and (b). All the cells on the concave edges are cut in two smaller cells and aligned along the concave edges. The wireframe view (Fig. 10(b)) shows them very well.

Another test of the polyhedral mesh generation is done with the more complex geometrical model, A-shape (Fig. 11(a)). This model is constructed with the model octant-cut-sphere-block-merge and a large sphere. It has various internal edges and surfaces. Many of the internal edges are concave. The clipping view of the mesh is shown in Fig. 11(b). It shows how cells are arranged along the internal boundaries. Also, various views of the polyhedral mesh generated for this model are shown in Fig. 11(c). It is shown that concave polyhedral cells are properly cut into convex polyhedral cells.

3. Polygonal prism mesh generation at the viscous layer

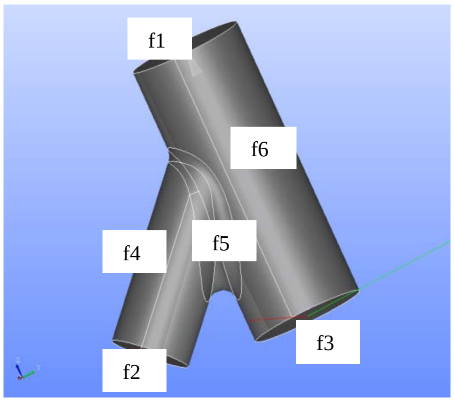
The procedure to generate a tetrahedral mesh with a triangular prism viscous layer mesh is started with a surface triangular mesh for whole surfaces. Then, the faces with viscous layers and the in-out faces are identified. Faces, f1, f2 and f3 are in-out faces for the merged pipe model in Fig. 12(a). The rest of the faces, f4, f5 and f6 are viscous faces. Triangular meshes on the viscous faces are extruded into the pipe body up to predetermined layer thickness yielding triangular prism cell layers. The triangular meshes on the in-out faces are contracted radially to accommodate the extrusion of the viscous layers. The tetrahedral mesh generated for the merged pipe model and its viscous layers are shown in Fig. 12(b).

Direct extension of this procedure to polyhedral mesh generation causes several problems. As noted before, the polyhedral mesh generation assumes the Delaunay and Gabriel properties for the boundary cells and faces. But, the extrusion and/or contraction process destroys that property. One way to get around this problem may be to adjust (or reduce) the geometrical shape first; meshing the modified geometry; and then, extruding the polygon up to the original viscous surface. Another method is to use the conventional surface triangular meshing and extrusion processes, as they are. But the next step is to use the extruded surface mesh, which most probably does not have Delaunay and Gabriel properties, as geometry data instead of using it for volume mesh generation. This technique is a kind of re-meshing method.

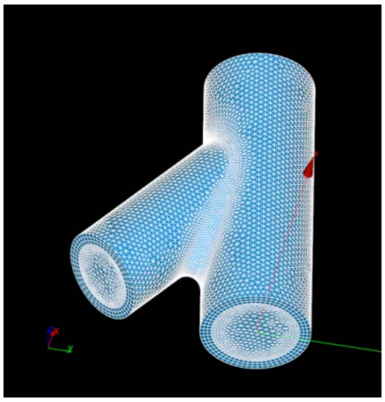
A triangular surface mesh can be converted to geometric data through STL (Stereo-lithography) data format. Then, the STL data are used to generate the surface mesh with Delaunay and Gabriel properties. The newly generated surface mesh is then used to generate a polyhedral volume mesh and polygonal surface mesh. Finally, the generated polygonal surface meshes on the viscous faces are extruded up to the original geometrical surfaces. Fig.4 shows the polyhedral mesh for the STL data of the merged pipe. A challenging issue is to find the best direction for the extrusion of the polygons. Best solution for this issue is to use STL triangles as coordinate references. Every vertex of the polygons on the model surfaces is projected to a triangle on the STL data. The barycentre coordinate of the projected point on the triangle can be used to locate the corresponding points on the respective corresponding triangles in the outer layers. Therefore, the procedure is summarised:

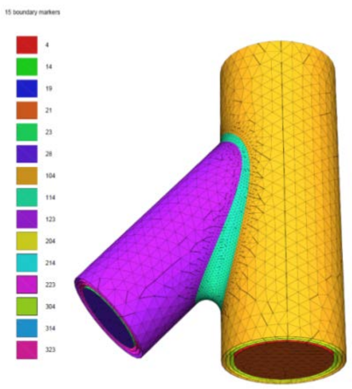
Procedure-I. Procedure to generate polygonal prismatic viscous layers.

1. Generate fine surface triangular mesh.
2. Generate triangular prism viscous layers by conventional extrusion process.
3. Make STL pseudo geometrical data.
4. Regenerate triangular surface mesh with proper size using the STL data.
5. Generate tetrahedral volume mesh with triangular mesh of innermost layer and in-out surfaces.
6. Generate polyhedral mesh with tetrahedral volume mesh.
7. Make any concave cell convex by the proper algorithm.
8. Project the vertices of the surface polygons to innermost layer of STL data.
9. Calculate the barycenteric coordinates of the projected points.
10. Extrude out layer by layer using the barycenter coordinates
11. Construct polygonal prism cells.

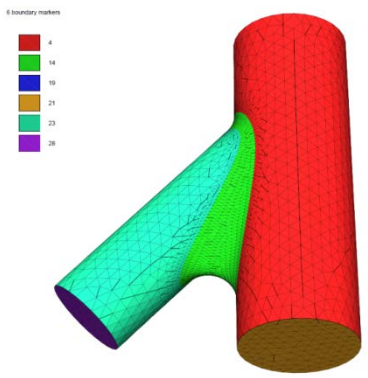


(a). solid model.

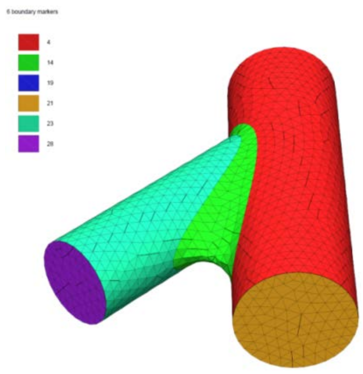

(b). tetrahedral mesh with viscous layer.



(c). STL data for layers.



(d). STL data for innermost layer plus in-out faces.



(e). re-mesh with STL data.

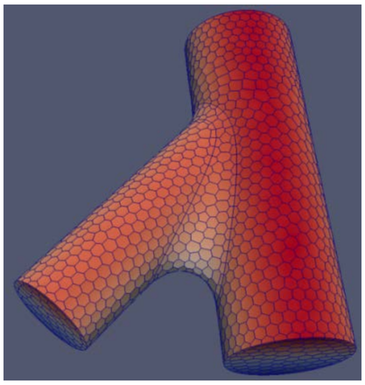
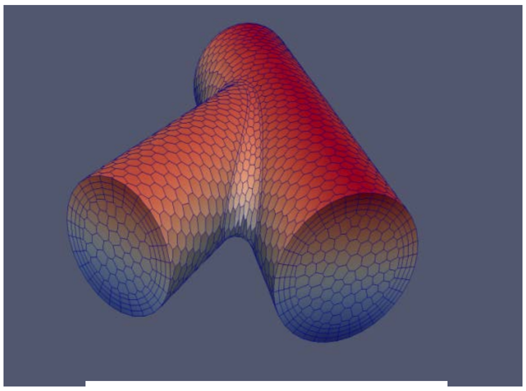

(f). polyhedral mesh.

Fig. 12. mergedPipes model and tetrahedral mesh generation.


(a). extrusion of polygons for layers.

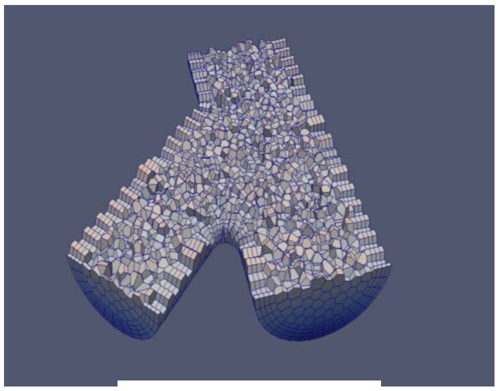

(b). clipping view.

Fig. 13. Polyhedral mesh of the mergedPipes with polygonal prism viscous layers.

The whole procedure to generate polygonal prismatic viscous layer is illustrated with the model, mergedPipes again. Step-1 and step-2 of the procedure-I is shown in Fig. 12(b). A finer triangular mesh is better for the correct extrusion of the polygons. All of the triangles are collected to generate STL data with layers as shown in Fig. 12(c) (step-3). Triangles on the innermost layer and in-out faces are shown in a separate figure (Fig. 12(d)). STL data are virtually the reduced geometrical model of the mergedPipes. The next step (step-4) is to generate a triangular mesh with the reduced model (Fig. 12(e)). Through step-5, step-6 and step-7, polyhedral mesh can be constructed by the dual mesh approach with the help of the cut-along-concave-edge method (Fig. 12(f)). The vertices of the polygons on the reduced model are not on the STL data. Instead, they are on the re-meshed triangles.

Therefore, it is necessary to project them to the innermost STL data layer, i.e. to the reduced model (step-8). Each of the triangles in the innermost layer has the corresponding triangle in outer layers. Projected points of vertices can be extruded to corresponding triangle in the outer layers. The barycenter coordinate [16] of the projected point in the projected triangle is calculated (step-9). It is used to locate the corresponding point in triangles of outer layers. The extrusion process locates those points in outer layers (Fig. 13(a)). Quadrangular side faces should be added to create proper prism cells (step-10 and step-11). The clipping view shows the regular polyhedral cells inside and polygonal prisms at the surfaces as expected (Fig. 13(b)).

4. Conclusions

The polyhedral mesh generation procedure is successfully developed using open source softwares, Tetgen, Netgen and SALOME Platform. Concave cells can be made convex by three methods, the non-manifold surface insertion method, the conical cell decomposition/bisection method and the cut-along-concave-edge method. Test runs with the simple incompressible fluid flow code, icoFoam, confirm the validity of the generated polyhedral meshes. A procedure to generate the polygonal prism mesh in the viscous layers is successfully designed by employing a re-meshing technique.

Even though the algorithms to generate the convex polyhedral mesh in this paper assume the boundary conforming Delaunay mesh, they can be applicable to the general tetrahedral mesh if the dual elements are properly defined.

References

[1] M. Peric, Simulation of Flows in Complex Geometries: New Meshing and Solution Methods, NAFEMS Seminar: May 3-4, 2004, Niedernhausen/ Wiesbaden, Germany

[2] R. V. Garimella, J. Kim, and M. Berndt, Polyhedral mesh generation and optimization for non-manifold domains, In Josep Sarrate and Matthew Staten, editors, Proceedings of the 22nd International Meshing Roundtable, pages 313-330, Springer International Publishing, 2014.

[3] H. Si, Three Dimensional Boundary Conforming Delaunay Mesh Generation, PhD Dissertation, TU Berlin, 2008.

[4] http://wias-berlin.de/software/tetgen/

[5] J. Schroberl, Netgen an advancing front 2d/3d-mesh generator based on abstract rules, Computing and visualization in science, 1(1): 41-52, 1997. http://sourceforge.net/projects/netgen-mesher/

[6] OpenFOAM, The open source computational fluid dynamics toolbox; http://www.openfoam.com/.

[7] E. Wang, Thomas Nelson, Rainer Rauch, Back to Elements - Tetrahedra vs. Hexahedra, CAD-FEM GmbH, Munich, Germany.

[8] G. Balafas, Polyhedral Mesh Generation for CFD-Analysis of Complex Structures, Master Thesis, Technische Universitat Munichen, 2014.

[9] B. N. Delaunay. Sur la sphere vide. Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7:793–800, 1934.

[10] K.R. Gabriel and R.R. Sokal, A new statistical approach to geographic analysis. Systematic Zoology, 18(3):259–278, 1969.

[11] G. Voronoi, Nouvelles applications des parametres continus a la theorie de formas quadratiques. Reine Angew. Math., 133:97–178, 1907.

[12] SALOME, Open source integration platform for numerical simulation; http://www.salome-platform.org/

[13] Open CASCADE Technology, 3D modeling and numerical simulation; http://www.opencascade.org/.

[14] C. B. Barber, D. P. Dobkin, H. T. Huhdanpapp, The Quickhull algorithm for convex hulls, ACM Trans. on Mathematical Software 22(4), 469–483 (1996); http://www.qhull.org.

[15] C. H. Rycroft, Voro++: A three-dimensional Voronoi cell library in C++, Chaos 19, 041111 (2009).

[16] I. Amidror, Scattered data interpolation methods for electronic imaging systems: a survey, 11(2), 157-176, 2002.