9th ANKARA INTERNATIONAL AEROSPACE CONFERENCE AIAC-2017-139
20-22 September 2017 - METU, Ankara TURKEY

A CARTESIAN BASED MESH GENERATOR WITH BODY FITTED BOUNDARY LAYERS

Merve Özkan[1] and M. Haluk Aksel[2]
Middle East Technical University
Ankara, Turkey

Özgür Uğraş Baran [3]
TED University
Ankara, Turkey

ABSTRACT

In this study, a new mesh generator which is capable of producing meshes with body-fitted boundary layers, and composed of mixed types of volume meshes involving pentahedral, tetrahedral, pyramid cells with Cartesian core mesh, is presented. Boundary layer elements are generated from the triangular surface mesh and are composed of pentahedra. The Cartesian core is generated by subdivision from boundary layer mesh intersections and eliminating the cells that are intersecting and outside of the control volume. The core is surrounded by pyramid elements and the gap between the surface boundary layer mesh and the pyramid connection layer is filled with tetrahedra to provide connection between these volume elements. Boundary layer mesh generator is based on the boundary layer library of open source code, SUMO. Another open source code TETGEN is utilized to generate the tetrahedral fill layer. Pentahedral and Cartesian mesh generation modules are developed in this study and all sub-mesh generators are integrated to achieve volume mesh generator.

INTRODUCTION

In the field of Computational Fluid Dynamics (CFD), generation of a well-defined mesh displays an important role on the solutions. Especially, for complex geometries, the meshing process becomes tedious. The Cartesian mesh generator produces very high quality interior cells. However, it is difficult to produce body-fitted volume mesh, with a high quality boundary layer. The generated mesh should be body-fitted and volume conservation should be satisfied, in other words no overlapping cells should exist.

Hexahedral core mesh is preferred over other types of meshes since it provides more accurate calculations [Owen,1998]. It can be generated easily and quickly. Cartesian grids based on hexahedral mesh became popular recently due to feasibility in computer resources on complex geometries. Cartesian mesh can be automatically generated and adapted and have simple flux formulation and simplified data structures. However, one of the most

[1] Msc. Student in Mechanical Engineering Department, Email: ozkan.merve@metu.edu.tr
[2] Prof. in Mechanical Engineering Department, Email: aksel@metu.edu.tr
[3] Asst. Prof. in Mechanical Engineering Department, Email: ozgur.baran@tedu.edu.tr

important drawbacks of the Cartesian mesh is the incapability of the body conforming adaptation in the vicinity of the wall mesh, where the boundary layer exists [Wang, 1998].

Body conforming Cartesian mesh generation is difficult to deal with. In the literature, there are methods which point out to the problem at the boundary layer when Cartesian mesh is used and some methods are found to deal with it. One and the most used of them is the cut cell approach, in which cubic cells at the boundary are cut into polyhedra in order to conform to the body. However, using cut cell approach is not sufficient to achieve accurate and high quality meshing within the boundary layer. Also, in the cut cell approach, many calculations during the generation of the overall mesh are required since various cut cells must be handled in three-dimensional domain. To solve the problems related to cut cells, a new approach for generating a body conforming mesh, is presented. A boundary layer mesh expanded from a conformal surface mesh, which consists of pentahedral volumes that will be inflated from the triangular surface mesh. Using the last pentahedral layer as a new surface mesh, a Cartesian mesh is generated with automatic refinement of cells to the size of the triangle size at the last layer of pentahedral mesh. The intersection with the surface mesh is estimated. The connection between the Cartesian mesh and pentahedral volume mesh is provided by tetrahedral and pyramid elements.

METHOD

The mesh generation procedure starts by using a surface mesh. The test case is shown in Figure 1 is a sphere into a larger computational volume. The triangular surface mesh can be exported in .cgns file format. After importing the .cgns file of triangular surface mesh, a second layer is generated by SUMO. The second layer includes triangle surface mesh which is inflated from the first surface mesh with normal to the surface. Having the new triangular mesh, the region between the new triangular surface mesh and the wall surface mesh is filled with pentahedral mesh layer by SUMO. The connection between the outer volume and the pentahedral layer is not given in this article.

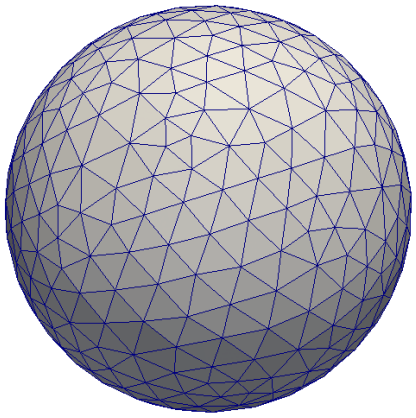

Figure 1: The test case, sphere

Pentahedral Mesh

Prismatic layer in the vicinity of the triangular surface mesh is necessary of the calculating the viscous effects at the wall surface. Also, quadrilateral faces provide good orthogonality [Kallinderis et al., 1998]. Pentahedral mesh is obtained by using a customized mesh generator based on boundary layer mesh generation library of open source code SUMO [Eller, 2009]. In the pentahedral mesh generaton procedure, prismatic envelope layer which will be the second triangulated surface inflated from the surface mesh is generated firstly.

This layer represents the enclosure of the prismatic layer which will be inflated within. Creating the envelope surface includes the stage that provides untangling of the surface normals [Tomac and Eller, 2014].

In the original SUMO, the tetrahedral volume mesh is generated before the generation of pentahedral volume mesh between the far-field boundary and the envelope surface mesh. The SUMO code is modified in this study to have only the pentahedral layer. Therefore, in the modified code, after the envelope layer is generated, the pentahedral mesh is inflated from the wall surface mesh of the solid boundary, whereas in the original code, the tetrahedral volume mesh is generated prior to pentahedral volume mesh.

The pentahedral volume mesh is generated form the wall surface mesh to the envelope surface. The number of pentahedral element layers is given by the user using the configuration file. The configuration file contains the desired properties for the to-be generated boundary layer mesh, such as the number of prismatic element layers, height of the first prismatic cell, maximum ratio of layer growth, maximum layer thickness, etc. The growth of the height of the pentahedral elements are decided by the user, given in the configuration file. The growth starts from the second layer of pentahedral layer, and the height of the pentahedral volume elements increase until reaching the height of the last layer in the pentahaedral mesh. Since the pentahedral mesh generation is in the direction of surface normal, the generation of the penthedral layer is very fast and identical by means of the number of the pentahedral volume elements on different regions of the geometry [Tomac and Eller, 2014].

The envelope surface mesh is exported to be utilized for the refinement for the Cartesian mesh generator in the next step.

The generated pentahedral boundary layer on the surface mesh using the isolated boundary layer mesh generator is shown in Figure 2. The new boundary layer generator has minimum dependencies with the original code to keep the final source of the mesh generator as simple as possible.

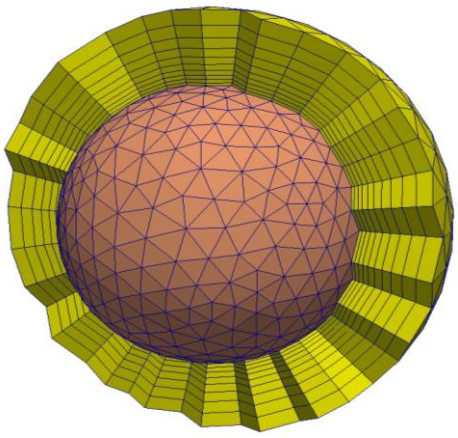

Figure 2: Pentahedral layer on the triangular surface mesh

Cartesian Mesh

The Cartesian mesh generator uses octree algorithm to generate the mesh. The octree algorithm is first developed by Mark Shephard et al. in 1980s. According to the criterion

which specifies the size of the smallest cell, the biggest cube that encapsulates the geometry starts to be divided into eight child cells recursively in 3D domain [Owen, 1998].

In this code, the octree data structure consists of a root cell and the child cells that are formed by dividing the root cell into child cells recursively. In octree algorithm, a root cell that encapsulates the geometry is generated as the first step. Root cell should enclose the entire computational domain to have a valid Cartesian mesh. Then, this is divided into eight child cells in three-dimensional domain. The division process is repeated by dividing child cells into eight cells at the prescribed locations. The criterion that is sought through all child cells is obtaining the size of the smallest child cell by the size of the triangular volume element on the outermost layer of the generated pentahedral mesh, the envelope surface mesh. The length of one side of the finest cube cell is equal to the size of the side of the triangle on the envelope surface mesh. The decision, whether a child cell would undergo division or not, is made by an algorithm that compares the size of the triangle of the envelope surface mesh and the finest child cell of the root cell by using a bitwise shift operator. Therefore, the number of steps is computed by the algorithm to generate desired number of child cells to obtain the finest cell which satisfies the size considerations, so the required refinement level of root cell can be determined.

Then, a seed cell is defined. This seed cell marks its neighbors recursively if the Cartesian cell intersects the marked boundary faces of the surface mesh. When the marking of seed cell reaches the envelope mesh of pentahedra and recursive algorithm cannot proceed any further, the marking stops. The remaining cells, which are not marked are omitted from the final mesh starting with seed cell because they are outside of or intersecting with the computational domain. A central cut from the Cartesian mesh for the test case is shown in Figure 3.

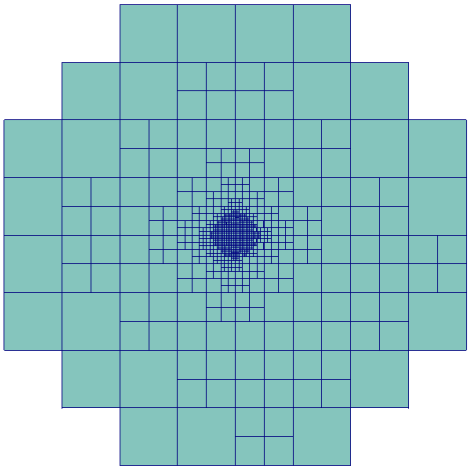

Figure 3: A cut through Cartesian mesh around sphere surface mesh in *YZ* Plane

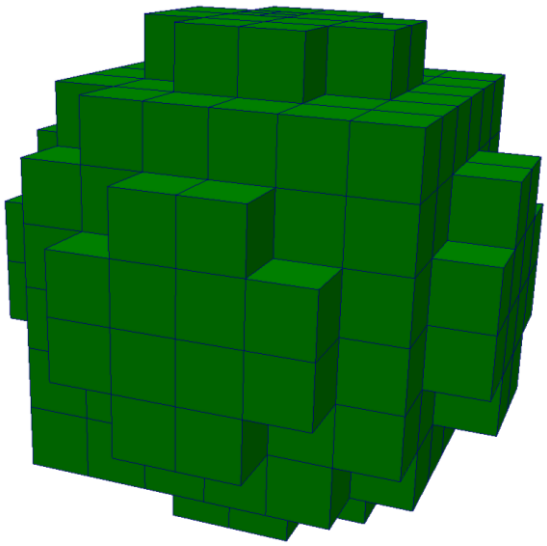

Figure 4: 3D Cartesian mesh after removing cells which are marked by seed cell

Data structure of the octree Cartesian mesh is formed with object oriented in C++ code. Each cell has property data. For a compact memory footprint, very few data are stored inside cells. In the cells, the refinement level of the current cell from the root cell, the binary coordinates and information of their children (if there exists) is stored. The *x, y* and *z* coordinates can be reached from the root cell, because as in Equation (1), the coordinates are estimated stating from the coordinate of the root cell. However, some data such as nodal coordinates, status of the cell (whether the cell is in the computational domain) and the information about parent and neighbor cells are not stored at the cell center.

The coordinates of the cells in the root cell is given by Equation (1).

$$Coordinate\ of\ Cell = Coordinate\ of\ Root\ Cell + \frac{Location\ of\ Cell}{2^{Refinement\ Level}} \times (Side\ Length\ of\ Root\ Cell) \quad (1)$$

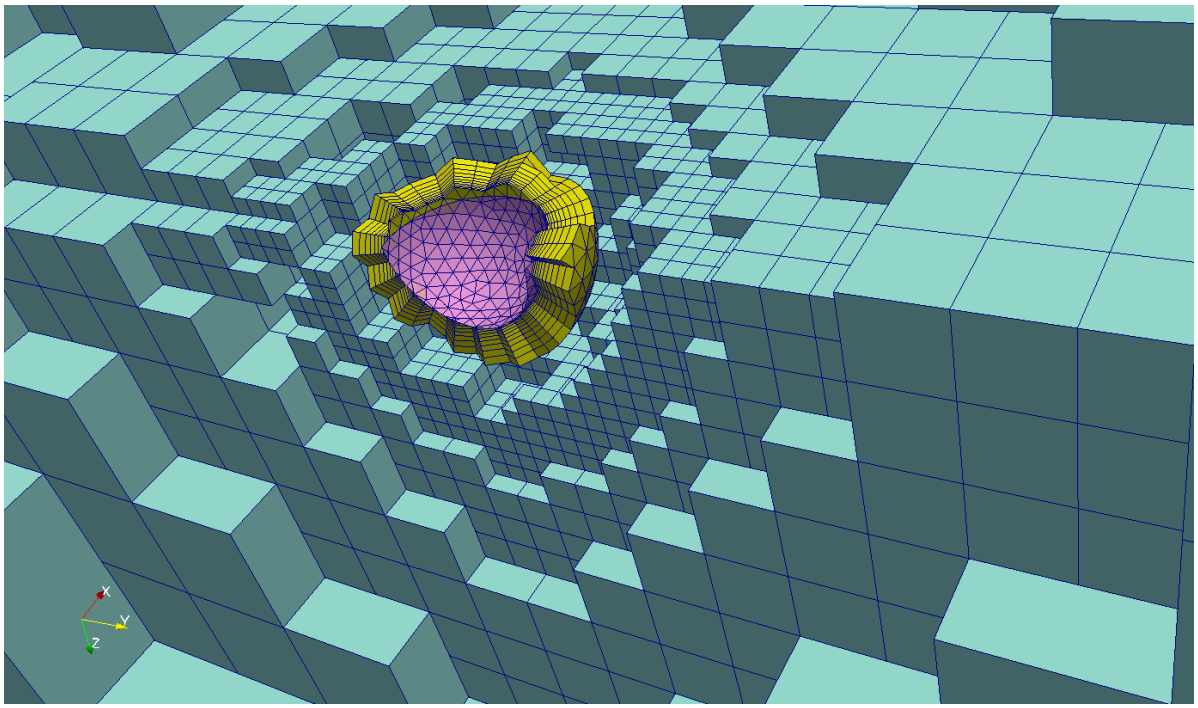

Figure 5: Cartesian mesh and pentahedral mesh

Pyramid Elements

After generating the Cartesian mesh and the pentahedral layer on the geometry, it can be seen that there is a gap between them, as shown in Figure 5. This gap is to be filled with tetrahedral and pyramid volume elements. In Figure 5, the inner-most layer, designated as pink, is the surface mesh of the sphere test case. The yellow part is the pentahedral mesh and blue mesh is Cartesian Mesh. A buffer layer, which consists of pyramid cells, on top of the quadrilateral faces of the Cartesian mesh is generated in order to obtain a triangular surface mesh shell.

Tetrahedral Mesh

Before the last step, Cartesian mesh is covered with triangular surface mesh and envelope surface mesh is also triangular. Between them a gap still exists. Therefore, the gap is easily filled with tetrahedral volume elements.

The tetrahedral filling is provided by TETGEN which is a very robust Delaunay triangulation based on open source mesh generator. In 3D domain, where the boundaries are input to the code by the user, the tetrahedral mesh is created in between these boundaries. Tetrahedral volume mesh is generated by using Delaunay tetrahedralization algorithm. The code with this algorithm generates points that will be gathered to create boundary-constrained conforming quality Delaunay mesh [Si, 2013]. The tetrahedral volume mesh on the pentahedral layer can be seen in Figure 5.

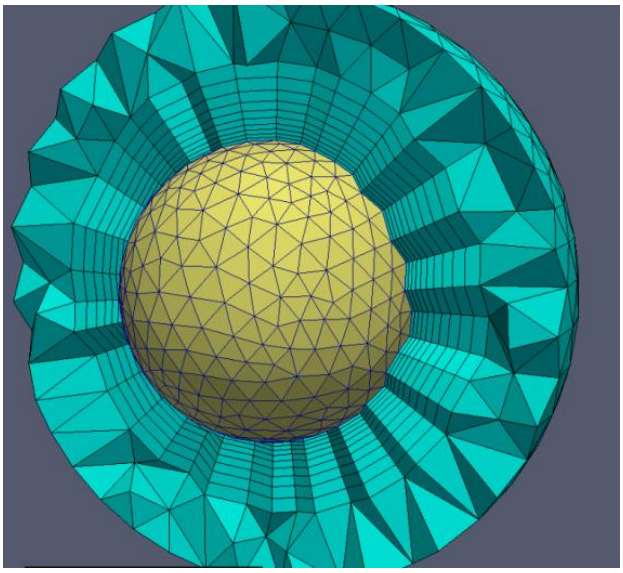

Figure 6: Tetrahedral volume mesh on the pentahedral layer

CONCLUSION

In this study, three separate C++ codes are integrated to obtain body conforming, hex dominant mesh generator. These codes are a pentahedral boundary layer mesh generator, an octree-based Cartesian mesh generator and a Delaunay based tetrahedral mesh generator. In CFD, since Cartesian meshes are handled easily and it is generated fast, it is preferable over other types of meshes. The study presented in this paper can be an alternative to the cut cell approach for Cartesian mesh. In cut cell method, due to irregular cells at the wall surface of the geometry, a number of calculations are carried out at the intersection of the geometry with the Cartesian Mesh [Owen, 1998]. With this new hybrid mesh, while consuming less time to generate the mesh, the boundary layer is generated to confirm more accurately with the surface geometry. After generating all mesh, three discrete mesh is combined, and this hybrid mesh generator will be used to implement the new high quality mesh to the open-source solver SU2.

ACKNOWLEDGEMENT

The authors acknowledge the support of this research by TÜBİTAK (The Scientific and Technological Research Council of Turkey) Project No. 114M590.

References

Eller, D. (2009) Sumo – aircraft geometry and surface modelling tool, https://www.larosterna.com/sumo.html

Kallinderis, Y., Khawaja, A., & Mcmorris, H. (1996). Hybrid prismatic/tetrahedral grid generation for viscous flows around complex geometries. *AIAA Journal, 34*(2), 291-298. doi:10.2514/3.13063

Owen, S. J. (1998), “A Survey of Unstructured Mesh Generation Technology”, Proceedings, 7’th International Meshing Roundtable.

Si, H. (2013) TetGen A Quality Tetrahedral Mesh Generator and 3D Delaunay Triangular User’s Manual.

Tomac, M. & Eller, D. (2014). Towards Automated Hybrid-prismatic Mesh Generation. *Procedia Engineering, 82*, 377-389.

Wang, Z. J. (1998). A Quadtree-based adaptive Cartesian/Quad grid flow solver for Navier-Stokes equations. *Computers & Fluids, 27*(4), 529-549.