AUTOMATIC HANDLING OF DEFECTIVE SURFACE GEOMETRY WITHIN GRID GENERATION FOR NAVIER-STOKES COMPUTATION

Paulus R. Lahur*, Takashi Ishida, Atsushi Hashimoto**, Keiichi Murakami****
Research Center of Computational Mechanics, Inc. (RCCM), Japan
**** Japan Aerospace Exploration Agency (JAXA)**
lahur@rccm.co.jp; ishida.takashi@jaxa.jp; hashimoto.atsushi@jaxa.jp; murakami.keiichi@jaxa.jp

Keywords: *Unstructured, hybrid, hexahedra, prismatic grid*

Abstract

In this study we proposed a robust method to automatically generate grid for Navier-Stokes flow computation, even when the input solid surface geometry contains some defects such as small gaps, overlaps and degenerate triangles. Such defects are often found in surface represented in STL (Stereo Lithography) format, which consists of triangles. This capability means that surface clean-up prior to grid generation is minimized, which is a significant advantage, as it saves a lot of time and manual labor. The grid itself is a hybrid of Cartesian grid (for region far from body surface) and prismatic grid (for boundary layer). Of particular interest is a new technique to handle concave feature of body surface. An approximation technique that minimizes reliance on the original surface geometry (which may contain defects) has been devised. Results have shown that the technique is effective. This opens up the possibility to carry out the whole process of Computational Fluid Dynamics fully automatically.

1 Introduction

This study is a joint effort between Japan Aerospace Exploration Agency (JAXA) and Research Center of Computational Mechanics, Inc. (RCCM). It is a part of JAXA's Hybrid Wind Tunnel project, which aims to carry out fast Computational Fluid dynamics (CFD) simulation of Navier-Stokes flows, in conjunction with wind tunnel experiments [1]. Our research in automatic grid generation is implemented in software called HexaGrid [2]. Previous results suggested that the method has a lot of potentials, as presented in the 4th Drag Prediction Workshop (DPW4) [3]. The results are competitive with that of manual grid generation, which is quite remarkable, considering that the grid is generated automatically [4-6].

Traditionally, the most accurate Navier-Stokes flow simulation uses multi-block structured grid, which takes a long time to generate and requires highly skilled manual labor. The time to generate the grid may be in the order of weeks.

With some compromise on solution accuracy, the turn-around cycle can be reduced to drastically to within days, by employing automatic grid generation. The grid is usually unstructured, which comes in the form of hybrid between either tetrahedral or hexahedral grid in the far-field region (away from solid surface), and prismatic grid in the near-field, to resolve boundary layer. In this study, Cartesian grid was chosen, due to its speed and simplicity in filling the far-field region, as well as the cell geometry that leads to accurate flow solution.

There are three ways to construct a hybrid between Cartesian and prismatic grid, all of which can be automatically generated. In the first approach, the grids are independently generated and then simply overlapped (Fig. 1). This requires the solution to be interpolated back and forth between the grids during flow

solving. This introduces interpolation error, which scales with the grid resolution difference in the overlapping region, as well as the solution gradient.

The second approach is to cut the Cartesian grid cells so that they match the outermost surface of prismatic grid (Fig. 2) [7-9]. This is the extension of cut cell approach [10]. It effectively results in a hybrid of three types of grid: prismatic grid, Cartesian grid, and cut-cell grid. The last grid cell has an arbitrary polyhedron shape, which also varies greatly in size. This property is undesirable in the computation of viscous effects.

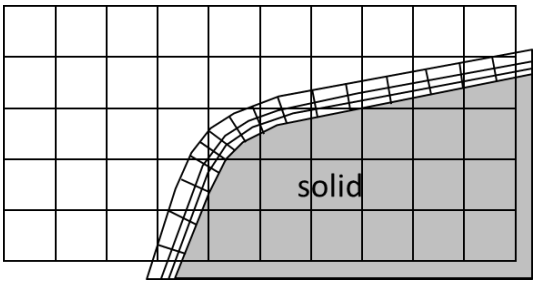

Fig. 1. Overlapping Grids.

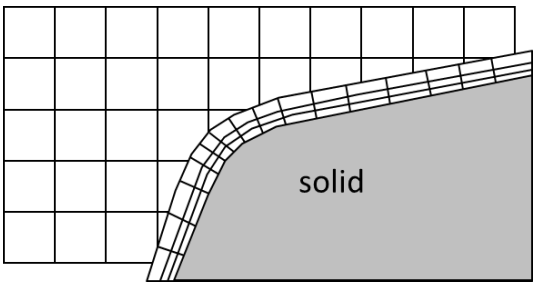

Fig. 2. Cut Cartesian Grid.

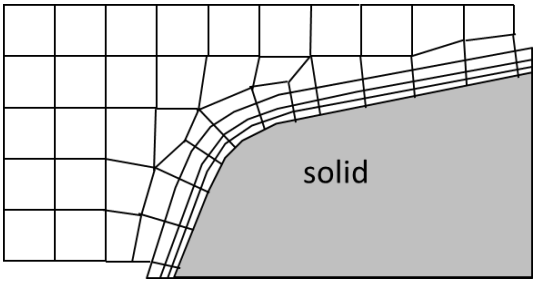

Fig. 3. Deformed Cartesian Grid.

The third approach is to remove solid Cartesian grid cells, deform the cells around solid surface, and generate prismatic grid from them (Fig. 3) [2, 11, 12]. This approach is taken in this study, because the shape and size of the Cartesian and prismatic grid cells blend well at the interface. This is a highly desirable property when computing viscous effects, as has been demonstrated in the previous results.

Now that the grid generation process has become automatic and fast, our focus shifts to the next bottleneck: the manual clean-up of solid surface, which is in STereoLithography (STL) format. This ubiquitous format consists of triangles, and can be produced by virtually any CAD software. However, it is common for this surface to contain defects such as gap and triangle overlap.

The objective in this particular study is thus to improve the grid generation process so that it tolerates surface with defects. This will allow us to skip the time-consuming surface cleaning step. Of particular interest, we address the weakest link in this grid generation approach, namely the capturing of concave features of the solid surface. A new method called approximate concave feature is proposed, which greatly improves tolerance to surface defects.

2 Grid Generation Method

2.1 General Procedure

2.1.1 Cartesian Grid Generation

Cartesian grid (Fig. 4) is generated by means of successive local refinement. This step starts with one cell that covers the whole computational domain, whose size is set by user. In three-dimensional space, each refinement divides a cell isotropically into eight child cells of equal size and shape. At the beginning, the grid is locally refined until the size of cells intersecting solid surface is smaller than a maximum grid size set by user. Then the grid is further refined until the size of cells intersecting the solid surface with large curvature reaches a minimum grid size. In addition, with the help of GUI, we can also explicitly control grid size anywhere using "Refinement Region."

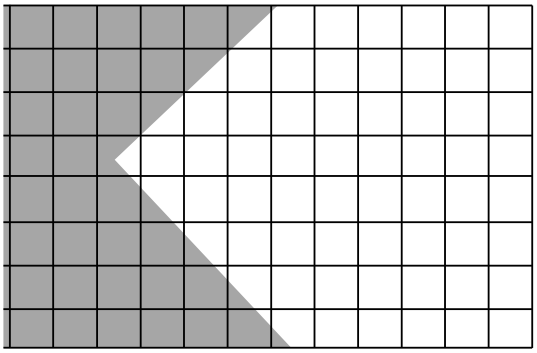

Fig. 4. Cartesian Grid Generation.

2.1.2 Removal of Non-Computational Cells

The purpose of this step is to remove unnecessary grid cells and create sufficient space for prismatic grid around the solid surface (Fig. 5). Removal of cells is carried out for those intersecting the solid object and those around the solid surface.

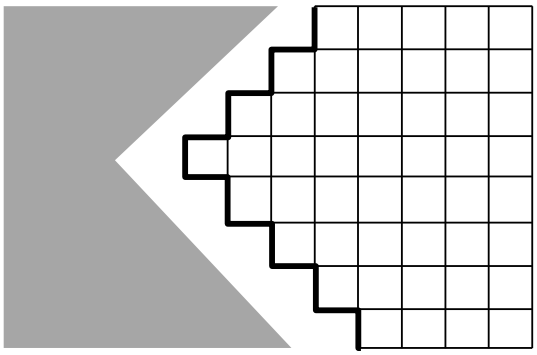

Fig. 5. Removal of Grid Cells.

2.1.3 Solid Surface Capturing

The surface of Cartesian grid is then snapped onto the solid surface (Fig. 6). This is done by moving a node on Cartesian grid surface to the closest location on the solid surface. Note that snapping always finds a unique location closest to the present position. This is a very important advantage, because it means that the method works even when there is a small gap between triangles that form a solid surface, and when the triangles overlap or intersect each other.

However, the weakness of the snapping method is that it cannot capture significantly concave geometry, because it will always move to the closest location. Thus this method is followed by feature capturing method, which forces the snapped Cartesian grid surface to move into the concavity (Fig. 7). For complex geometry, this is a difficult task, especially when the feature curves are close to each other.

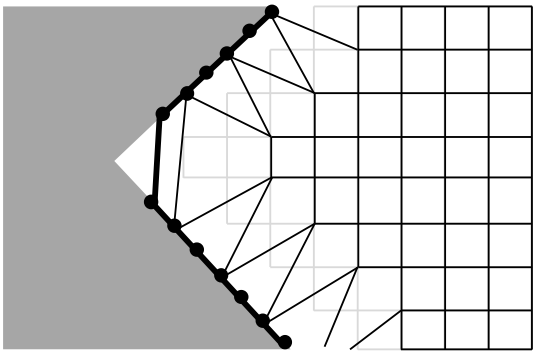

Fig. 6. Solid Surface Capturing.

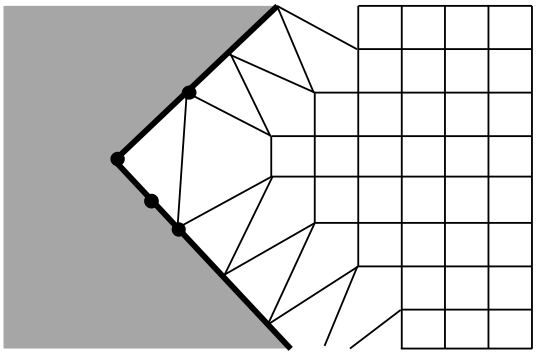

Fig. 7. Feature Capturing.

2.1.4 Prismatic Grid Generation

Prismatic grid layers are constructed on the snapped surface (Fig. 8). The total thickness of prismatic grid is determined by the size of Cartesian grid cell removed. Note that the prismatic grid cells are perpendicular to surface.

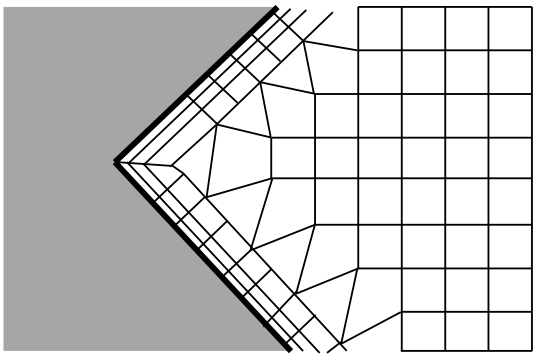

Fig. 8. Prismatic Grid Generation.

2.1.5 Quality Improvement

The grid cells at the interface of Cartesian and prismatic grids are smoothed so they blend well (Fig. 9). A smooth transition of grid cell's size and shape is very important in Navier-Stokes computation.

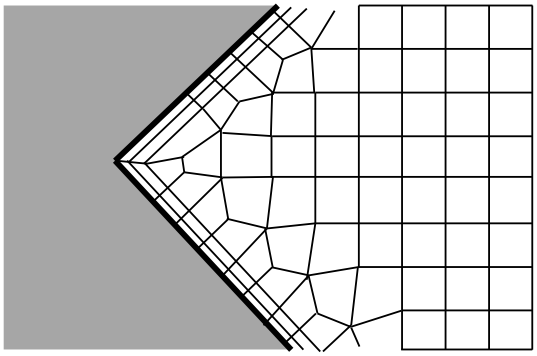

Fig. 9. Quality Improvement.

2.2 Geometry Defects

Due to various factors, defective or "dirty" STL data may be produced. At this stage, we need to specify which conditions are considered defective.

1) Gap between triangles.
2) Triangles overlapping or intersecting with each other.
3) Useless triangle (inside a solid body).
4) Very small triangle that degenerate into line or point.
5) Inconsistent vertex ordering within triangle, which results in some normal vectors point into solid, and others into fluid.
6) Irregular normal vector distribution of triangles.
7) Irregular size distribution of triangles.
8) Lack of resolution.
9) Excessive resolution.

The top three are the most serious, because they prevent us from constructing a valid boundary.

Because the grid is generated from the Cartesian grid first, and then snapped to solid surface, defects in the surface such as gap and triangle overlap are ignored. Thus the resulting surface grid is a valid surface. The snapping algorithm is very simple and powerful in solving the problem of defective solid surface. A grid node will never step into a gap between triangles (as long as the gap is reasonably small). Even if triangles overlap or intersect each other, snapping will always get the closest location. Even if an unused triangle exists inside the solid, it will be completely ignored. Moreover, the resulting surface grid already has a smoothly distributed face size and orientation.

However, snapping algorithm has an inherent weakness, that is, it fails to capture (sharp) concave features of solid surface.

In our previous study, this is dealt by first constructing the concave feature directly from the STL data. The simplest method is to compute the angle between neighboring triangles. If the angle exceeds certain threshold, then the line segment shared by the two triangles is declared as feature. Connecting the line segments will result in a discretized form of feature curve. Having done this, we then move the appropriate surface grid nodes onto the feature.

The problem with this approach is the implementation is highly prone to surface defect. The root of the problem lies in the first step. When the STL data is dirty, extracting a feature is prone to failures. As in the case of the body surface, a dirty feature curve consisting of line segments also has similar traits.

Additionally, we depend on the threshold angle as a parameter. If the threshold is small, a large number of line segments will be produced, resulting in "fake" features. If the threshold is large, only a small line segments will be produced, resulting in loss of important features.

Thus we need an algorithm that, just like the snapping algorithm, provides a powerful yet simple way to conveniently sidestep the dirty STL problem. In this study we propose the following method.

2.3 Procedure to Capture Concave Feature

Let us first note that by the time the Cartesian grid is snapped to body surface, we already have a very good approximation of the body surface. Thus, instead of trying to construct body feature directly from the original surface data (which may contain defects), we can make use of the

grid surface (which does not contain any defect).

The idea is to use the data during the snapping of grid surface around the concave feature to reconstruct the line segments that form the feature. A bi-planar assumption is a natural choice due to its simplicity. As the grid resolution increases, so the quality of approximation also increases. The main idea of the algorithm is illustrated in Fig. 10. The method is as follows. For each grid face:

1) Detect whether concave feature capturing is necessary (if the distance between the face of surface grid and solid surface is large).
2) If feature capturing is necessary, using the information from surface grid snapping, group the nodes of the face into two sides.
3) Construct two planes that approximate the local body surface at either side of the feature.
4) Intersect the two planes to form approximate feature line.
5) Move the appropriate nodes to the feature line. Note that at this stage, we need to synchronize the movement of the nodes of this face with those of neighboring face.

Currently we are working on other model of approximation to handle other types of concave features, such as feature point at concave-convex saddle point and concave corner. Such models will require more than two planes, and perhaps a combination between line and plane.

Note that in certain cases, the approximation may result in a grid node moving far away from its original location, such as in deep concavity. The approximation may result in grid surface intersecting itself. This condition is thus checked and prevented from occurring.

Furthermore, higher degree of approximation (quadratic) is also worth exploring in our future work.

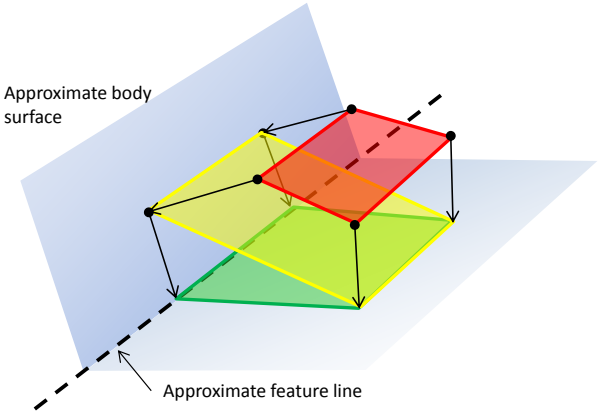

Fig. 10. Approximate Concave Feature

3 Sample Cases

A couple of samples that help with qualitative assessment are shown below.

3.1 Aircraft Model

This case is chosen because of the proximity of its feature curves provides us with an insight on how the method behaves in such situation (Fig. 11). Fuselage and horizontal tail junction has two concave feature curves close to each other.

Result in Fig. 12 shows that the coarse grid, with cell size comparable to distance between features, cannot capture the feature. However, this problem is rectified as soon as the grid resolution is increased (Fig. 13). It seems that, at least in this case, two or three cells between feature lines are sufficient for a reasonable capture. As the resolution is increased further, naturally the approximation quality becomes better (Fig. 14 – 16).

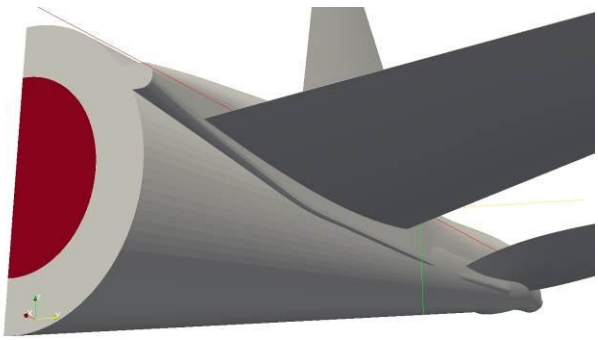

Fig. 11. Body surface geometry (STL)

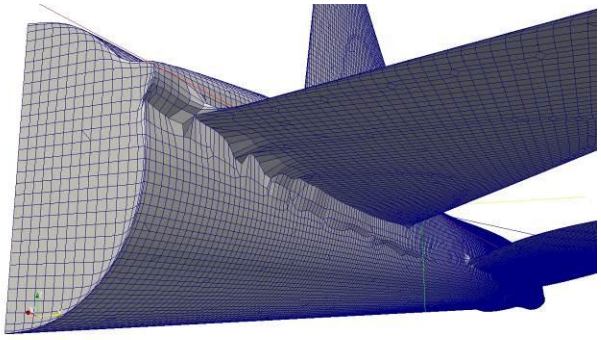

Fig. 12. Grid with minimum cell size = 2.0

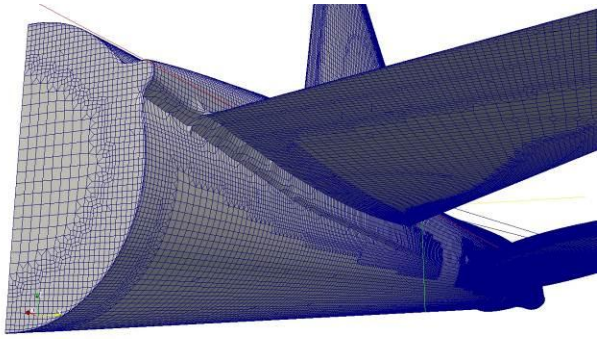

Fig. 13. Grid with minimum cell size = 1.0

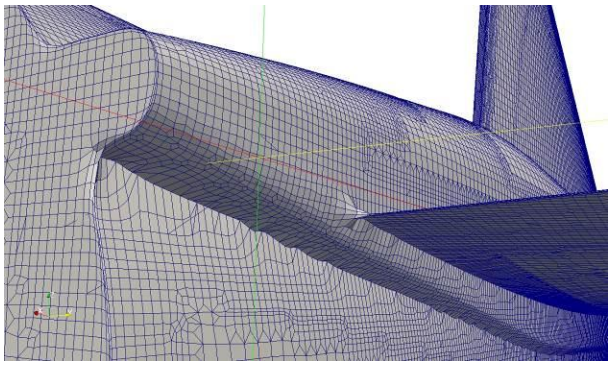

Fig. 14. Grid with minimum cell size = 0.5

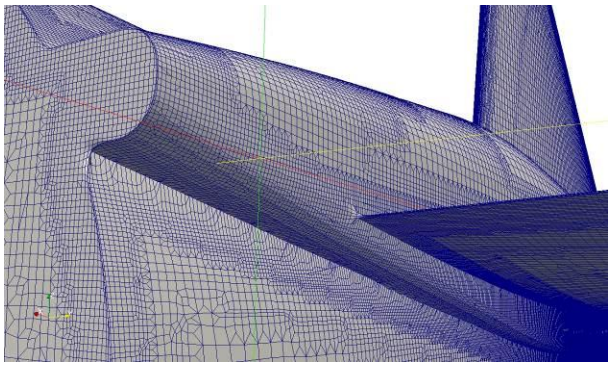

Fig. 15. Grid with minimum cell size = 0.25

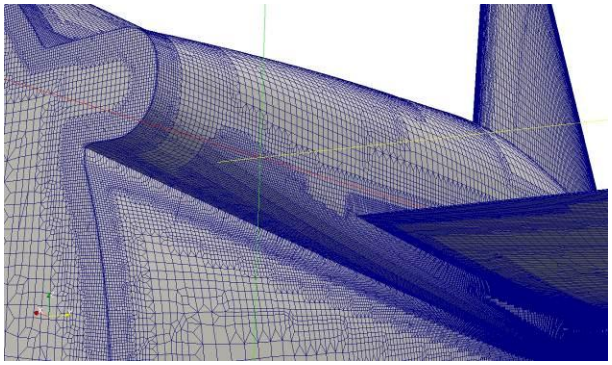

Fig. 16. Grid with minimum cell size = 0.125

3.2 Aircraft Model in Wind Tunnel

The end target of this project is the CFD simulation of an aircraft model in wind tunnel testing environment, complete with the support system, as shown in Fig. 17 and 18. Note that the components are not trimmed against each other. Each component (shown in different color) is discretized without any regard to its neighbors. Furthermore, it is clear that the sting support components protrude into the aircraft fuselage and vertical support component (Fig. 18). The grid generator simply ignores these "defects" and generate grid straight from this input. Clearly, the time saved by not having to clean up this surface is significant.

The grids are shown in subsequent figures. Reasonable grid has been obtained, although more work is still necessary at this stage to improve its quality. Concave geometry features are well captured.

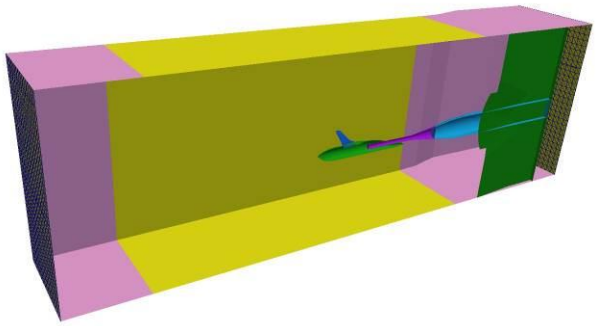

Fig. 17. Wind Tunnel Geometry.

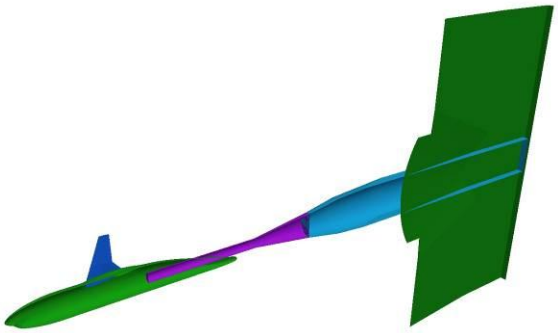

Fig. 18. Model and Support System Geometry.

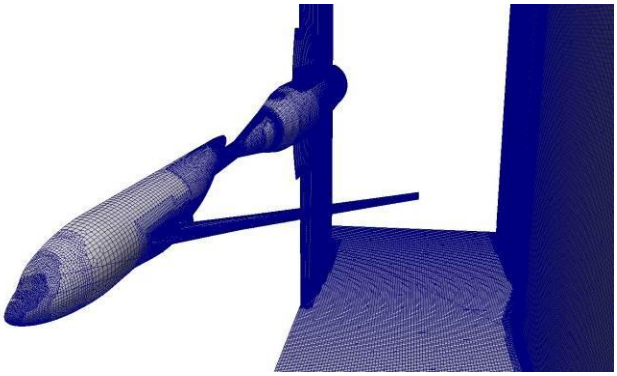

Fig. 19. Grid around Model.

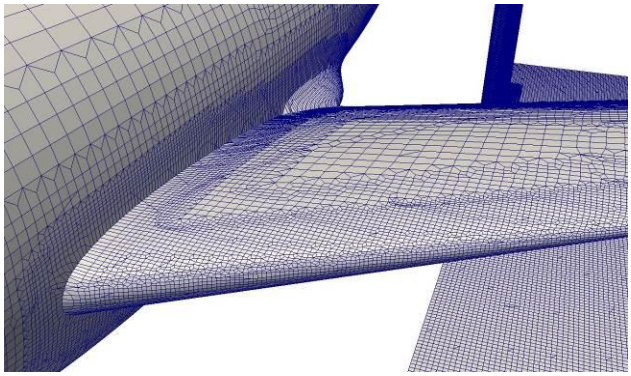

Fig. 20. Grid around Fuselage and Wing Junction (Top).

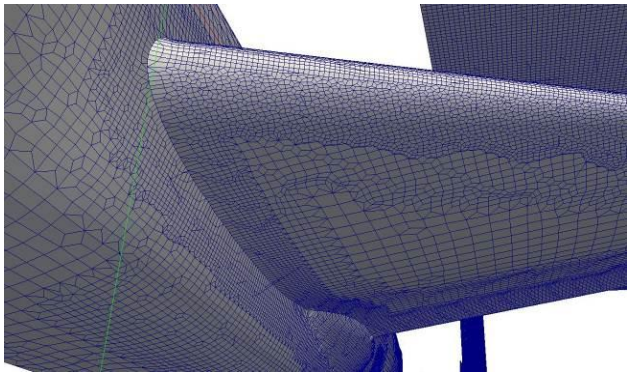

Fig. 21. Grid around Fuselage and Wing Junction (Bottom).

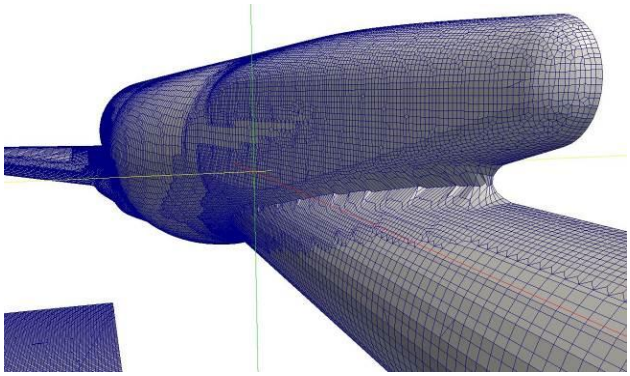

Fig. 22. Grid around Fuselage and Sting Support Junction.

4 Concluding Remarks

In this study we proposed a robust method to automatically generate grid for Navier-Stokes flow computation, even when the input solid surface geometry contains some defects such as small gaps, overlaps and degenerate triangles. Such defects are often found in surface represented in STL (Stereo Lithography) format, which consists of triangles. This capability means that surface clean-up prior to grid generation is minimized, which is a significant advantage, as it saves a lot of time and manual labor. The grid itself is a hybrid of Cartesian grid (for region far from body surface) and prismatic grid (for boundary layer). Of particular interest is a new technique to handle concave feature of body surface. An approximation technique that minimizes reliance on the original surface geometry (which may contain defects) has been devised. Results have shown that the technique is effective.

Our future work includes adding more approximation model to capture other types of feature, as well as improving the quality of the grid. Higher degree of approximation is also being considered.

We would like to conclude that the capability to generate grid without cleaning up the input surface opens up the possibility of fully automated CFD simulations.

References

[1] Watanabe, S., Kuchi-ishi, S., Aoyama, T., "A Prototype System towards EFD/CFD Integration: Digital/Analog-Hybrid Wind Tunnel", Proceedings of 27th Congress of International Council of the Aeronautical Sciences, 2010.

[2] Lahur, P. R., "Automatic Hexahedra Grid Generation Method for Component-based Surface Geometry," AIAA paper 2005-5242, 2005.

[3] Hashimoto, A., Murakami, K., Aoyama, T., Lahur P., "Lift and Drag Prediction Using Automatic Hexahedra Grid Generation Method," AIAA paper 2009-1365.

[4] Hashimoto, A., et al., "Drag Prediction on NASA CRM using Automatic Hexahedra Grid Generation," AIAA Paper 2010-1417, 2010.

[5] Vassberg JC, Tinoco EN, Mani M, Rider B, Zickuhr T, Levy DW, Brodersenk OP, Eisfeldk B, Crippak S, Wahls RA, Morrison JH, Mavriplis DJ, Murayama M, "Summary of the Fourth AIAA CFD Drag Prediction Workshop," AIAA 2010-4547, 2010

[6] Vassberg, J.C., DeHaan, M.A., Rivers, S.M., Wahls, R.A., "Development of a Common Research Model for Applied CFD Validation Studies," AIAA paper 2008-6919.

[7] Deister, F. and Hirschel, E.H., "Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometries," AIAA paper 99-0782, 1999.

[8] Leatham, M., Stokes, S., Shaw, J.A., Cooper, J., Appa, J., and Blaylock, T.A., "Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations," AIAA paper 2000-2247, 2000.

[9] Karman, S.L.Jr., "SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries," AIAA paper 95-0343, 1995.

[10] Aftosmis, M.J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," VKI Lecture Series, 1997-02, 1997.

[11] Tchon, K.F., Hirsch, C., and Schneiders, R., "Octree-based Hexahedral Mesh Generation for Viscous Flow Simulations," AIAA paper 97-1980, 1997.

[12] Wang, Z.J. and Chen, R.F., "Anisotropic Solution-Adaptive Viscous Cartesian Grid Method for Turbulent Flow Simulations," AIAA Journal, Vol. 40, No. 10, 2002, pp. 1969-1978.

Copyright Statement