

# Automatic Grid Generation Method with Direct Treatment for Defective STL Data

Paulus R. Lahur<sup>1</sup>, Atsushi Hashimoto<sup>2</sup>, and Keiichi Murakami<sup>3</sup>

<sup>1</sup> Research Center of Computational Mechanics, Inc. (RCCM), Shinagawa, Tokyo, 142-004, JAPAN, lahur@oi.rccm.co.jp

<sup>2</sup> Japan Aerospace Exploration Agency (JAXA), Chofu, Tokyo, 182-8522, JAPAN, hashimoto.atsushi@jaxa.jp

<sup>3</sup> Japan Aerospace Exploration Agency (JAXA), Chofu, Tokyo, 182-8522, JAPAN, murakami.keiichi@jaxa.jp

**Summary.** Our objective in this study is to explore an automatic grid generation method for accurate Navier-Stokes flow computation, which can directly use the input of stereolithography (STL) data with defects, that is, without prior clean-up. This approach promises a significant improvement in CFD workflow, because the time and labor required for surface geometry preparation is very significant. Preliminary results suggest that the method is indeed promising. Progress is being made, especially regarding sharp feature capturing, as presented below.

## 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) computations of Navier-Stokes flows, in conjunction with wind tunnel experiments. Our research in automatic grid generation is implemented in software called HexaGrid. Previous results suggested that the method has a lot of potentials in CFD workflow, as presented in the 4th Drag Prediction Workshop (DPW4) [1]. In comparison with the results from manual grid generation, the accuracy of flow solution is very competitive. This is quite remarkable, considering that the method is fully automatic, and it takes only tens of minutes to an hour to run instead of a few weeks of manual labor.

Along with the usual follow-up work such as quality, performance and robustness improvement, a number of interesting extensions are consid-

ered. One of them is direct treatment of surface geometry that contains defects. The surface geometry that we use is a discretized form consisting of triangles, where the input is in STL format, which is a common format for surface definition.

There are two factors that motivate us to explore grid generation that can directly handle dirty STL:

1. Similar to manual grid generation, manual clean up of surface geometry is also time and labor-consuming. Surface geometry preparation and grid generation can easily dominate the timeline of CFD workflow.
2. Our current grid generation method already has some tolerance to defects in surface geometry, so it makes sense to further strengthen this advantage.

Thus, it is tempting to ask whether it is possible to improve the method so that it tolerates most, or ideally, all defects in surface geometry, and thus requires no surface clean up.

## 2 Types of STL Defect

Various types of defect may be found on STL surface, especially when it consists of many components. The types of defect are identified below, more or less in the order of decreasing level of severity.

1. Gap between triangles.
2. Triangles overlap or intersect each other.
3. Useless triangles (e.g. inside solid in a flow computation).
4. Very small triangles that degenerate into lines or points.
5. Inconsistent vertex ordering within triangles, which results in normal vectors of some triangles point into solid, and others into fluid.
6. Distribution of triangles' normal vectors is not smooth.
7. Distribution of triangles' size is not smooth.
8. Lack of resolution.
9. Excessive resolution.

## 3 Selecting Grid Generation Method

Given the present objective, our choice of grid generation method follows these steps of reasoning.

1. Generating boundary grid or interior grid first? Although generating boundary grid first is a very popular approach, it requires cleaning up

the surface prior to grid generation. To avoid this, we chose to generate the interior grid first.

2. To cut or to deform? To construct boundary grid from the interior grid, we have to either cut or deform the cells to fit the solid surface. Proper cutting requires that the solid surface is well defined [2]. Knowing that this may not be the case, we opted for cell deformation instead [3].
3. Exact or approximate sharp feature capturing? The deformation algorithm above can capture most part of the surface, but it does not work well for sharp features, so additional algorithm is needed to capture the features. One possibility is to construct the features from the original STL surface and then capture them. However, this requires cleaning up the parts of surface that contains features. Our experience with this approach has shown that it is difficult to compose an algorithm that is both automatic and reliable for this task. Thus, recently we begin exploring the second possibility, that is, approximate sharp feature capturing using the existing boundary grid.

These steps essentially minimize the amount of information from the STL surface, which in turn, minimize exposure to surface defects.

#### **4 Outline of the Grid Generation Method**

The following is the outline of the grid generation method.

1. Cartesian grid generation. This is the interior grid that encases the whole computational domain. Successive isotropic local refinement is carried out until certain requirements are satisfied.
2. Removal of cells in and near solid body. The purpose of this step is to create sufficient space for prismatic grid around the solid surface for Navier-Stokes computation. The remaining cells form a boundary grid that roughly resembles the solid surface
3. Snap boundary grid (deform cells on boundary) to fit solid surface. This is done by means of snapping all nodes on the boundary grid to the closest location on solid surface. Note that even if the STL surface contains defects, a valid boundary grid can still be constructed.
4. Sharp feature capturing. The resulting boundary grid from the previous step captures most part of the STL surface, except where there are concave features. As mentioned above, the boundary grid itself is used to approximate the features.
5. Prismatic grid generation. This is required to resolve boundary layer flow in a proper Navier-Stokes computation
6. Quality improvement. This is carried out by means of grid smoothing.

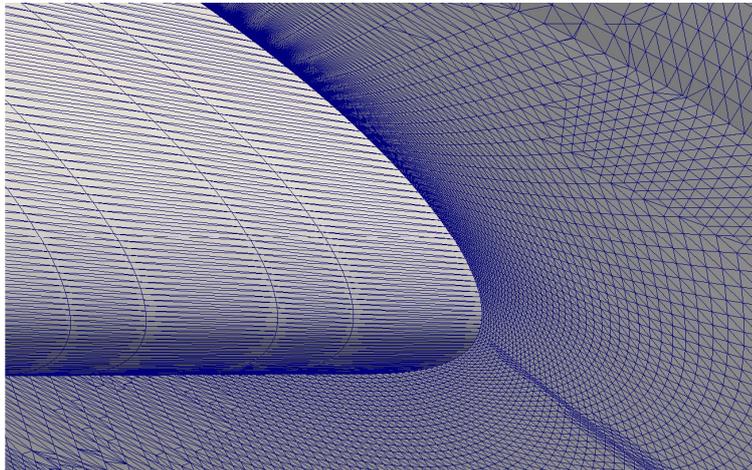
## 5 Results

To demonstrate the effectiveness of the method, the STL data of an aircraft model is used (Fig. 1). The vertices of the unconnected triangles are randomly perturbed by a maximum magnitude of 0.0001 of the length of the aircraft. As shown in Fig. 2, this is a huge value compared to the size of the triangles. In fact, forming a topologically valid surface from this data would be extremely difficult, to say the least.

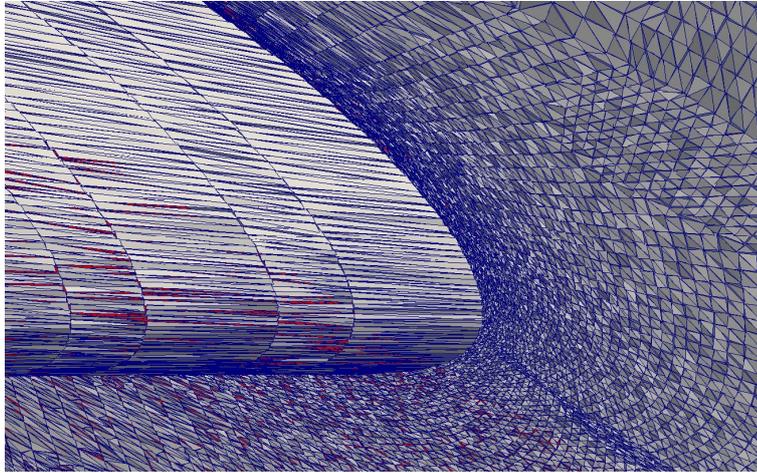
The perturbed grid is then used as the input for the grid generation. The result is shown in Fig. 3. Although the boundary grid becomes uneven due to the large perturbation, the algorithm still manages to reconstruct the surface, including the sharp feature. This clearly demonstrates the potential of the feature approximation method.

## References

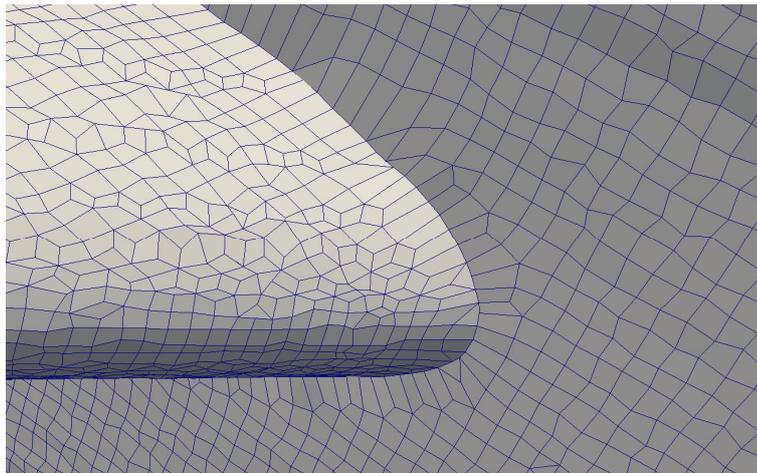
- 1 Hashimoto A, Murakami K, Aoyama T, Lahur P, “Lift and Drag Prediction Using Automatic Hexahedra Grid Generation Method,” AIAA paper 2009-1365
- 2 Aftosmis, M.J., “Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries,” VKI Lecture Series, 1997-02, 1997
- 3 Tchon, K.F., Hirsch, C., and Schneiders, R., “Octree-based Hexahedral Mesh Generation for Viscous Flow Simulations,” AIAA paper 97-1980, 1997



**Fig. 1.** Original STL data (wing – body junction of an aircraft model).



**Fig. 2.** STL data after random perturbation (maximum magnitude of 0.0001 of the aircraft length).



**Fig. 3.** Boundary grid generated using perturbed STL data.