29th Congress of the International Council of the Aeronautical Sciences
St. Petersburg, Russia; September 7-12, 2014

RAPID LARGE-SCALE CARTESIAN MESHING FOR AERODYNAMIC COMPUTATIONS

Daisuke Sasaki*, Kazuhiro Nakahashi**
***Department of Aeronautics, Kanazawa Institute of Technology, ****JAXA**

Keywords: *Meshing, Cartesian Mesh, CFD, Parallel Computations*

Abstract

Cartesian mesh has a lot of advantages for the CFD, particularly because of the rapid mesh generation capability for complicated real geometries. In this paper, the mesh generation techniques for block-structured Cartesian mesh solver (Building-Cube Method) are described. The target is to generate large-scale (billion-scale) mesh rapidly to reduce the pre-processing time, which will be required to reproduce the curved geometry precisely. In the paper, several examples of the large-scale mesh generation are also presented.

1 Introduction

Great progress has been made in Computational Fluid Dynamics (CFD) in the past several decades and nowadays it is indispensable in the design of fluid mechanics. The demand for the CFD capability is still increasing to handle more complex geometries and detail components. In addition, more precise analysis of complicated flow phenomena is demanded such as various unsteady flows including transition and separation. To satisfy these demands, Cartesian mesh has been recently focused because of the characteristics [1-11]. Cartesian mesh is useful for rapid meshing and it can easily handle moving objects. The higher-order scheme can be easily employed in various solvers compared to unstructured mesh.

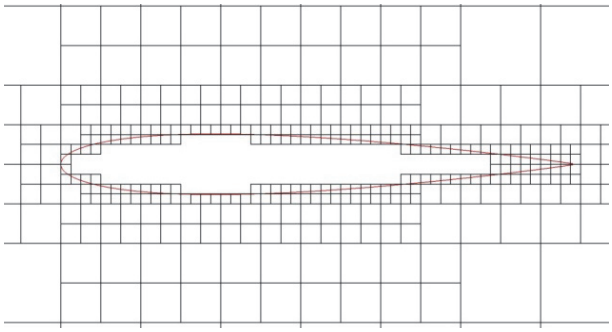
The Cartesian-mesh generally requires large mesh to represent curved-geometry and resolve boundary layer for high-Reynolds number flows. Thus, the parallel computation is indispensable to reduce the computational time for large-scale mesh. To achieve higher parallel efficiency for the massively number of CPUs, Building-Cube Method (BCM [4]) has been proposed, which is a block-structured Cartesian mesh solver. In BCM, three-dimensional flow fields are divided into various sizes of cuboids called 'Cube', and the cubes are subdivided into equally-spaced Cartesian meshes called 'Cell'. As the mesh resolution is determined according to the local cube size, fine meshes near the body and coarse meshes far from the body can be accomplished. Because of the uniform computational load at each cube regardless the cube size, it is expected to achieve the higher parallel performance even with the massive number of processors. Various research has been conducted to develop BCM solvers

Various research of BCM is being conducted to develop efficient and effective solvers using parallel computers [4-11]. The continuous efforts of solver developments with the usage of recent massive parallel computers will reduce the computational time, but efficient mesh generation capability for the large-scale mesh is required. The mesh generation has to be quick and robust to reduce the pre-processing time even if the number of computational mesh is billion. In this paper, the overview of GUI-based mesh generator for the BCM solver is described.

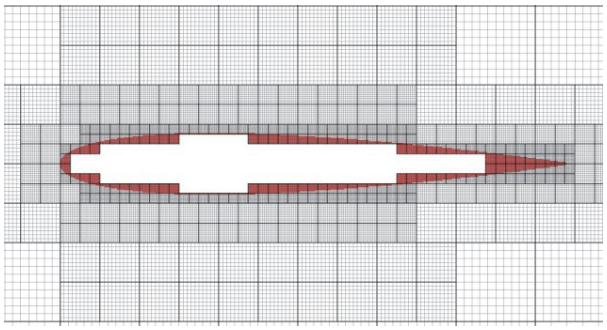
2 Mesh Generation Process

The computational mesh consists of many cuboids which include equally-spaced Cartesian mesh as shown in Fig. 1. It shows Cube and Cell distribution around NACA0012 airfoil for 2D case. These cuboids (blocks) in Fig.1 (a) are called as 'Cube', and Cartesian mesh in each

cube are called as 'Cell' in Fig. 1 (b). Each Cube in this example is composed of 10x10 meshes regardless the Cube size. In this section, the mesh generation process is presented.



(a) Cube allocation



(b) Cell distribution

Fig. 1. Mesh structure

2.1 Overview

The flowchart of mesh generation process is shown in Fig. 2. After CAD data (STL) reading, Cube structure is firstly generated and then Cell is generated in each Cube. Since the Cell is independently generated in each Cube (block), the process is parallelized with OpenMP.

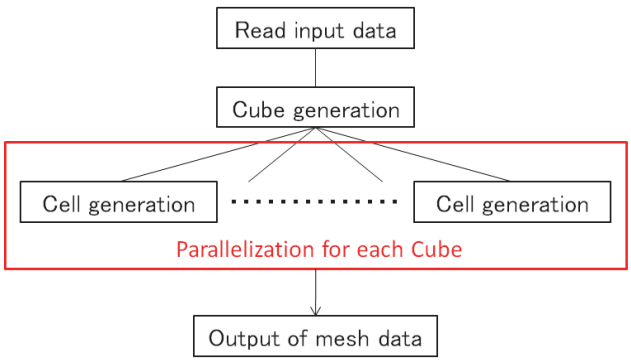


Fig. 2. Flowchart of mesh generation

2.2 Cube Generation

The followings are the process of Cube allocation:

(a) Computational domain size setting
(b) Domain division
(c) Intersection check
(d) Smoothing
(e) Cube classification

The process (a) to (e) is corresponding to Fig. 3. After the setting of outer domain (a), the computational domain is divided into four Cubes (2D) or eight Cubes (3D) (b). The intersection between Cube and objects is checked. If the size of intersecting Cube is larger than the user-defined threshold based on minimum cell size, the Cube is again divided. The process is repeated until all the intersecting Cubes satisfy the threshold (c). After the intersection check, smoothing is conducted to satisfy the condition that the adjacent Cube size has to be double or half (d). Finally, each Cube is flagged as Fluid/Inner/Wall according to the relation with objects (e). In the figure, red is Inner Cube, blue is Wall Cube and Green is Fluid Cube.

2.3 Cell Generation

After the Cube generation, the Cells are equally distributed in each Cube. When Wall Cube is processed, the intersection check is conducted to all the Cells, and thus each Cell is flagged as Fluid/Inner/Wall required for CFD or CAA.

2.4 Rapid and Large-Scale Meshing

Cartesian mesh is usually very efficient in terms of memory usage. However, the amount can be huge when large-scale meshing of billion-cell class is conducted. In the BCM meshing, each Cube information (e.g. Cell flag) is compressed to save the memory usage. During the Cell generation process, only the required data is decompressed for the efficient usage of memory.

One of time-consuming process of mesh generation is intersection check. Thus, the process speed is accelerated using SSE (Streaming SIMD Extensions), which enables the fast vector analysis.

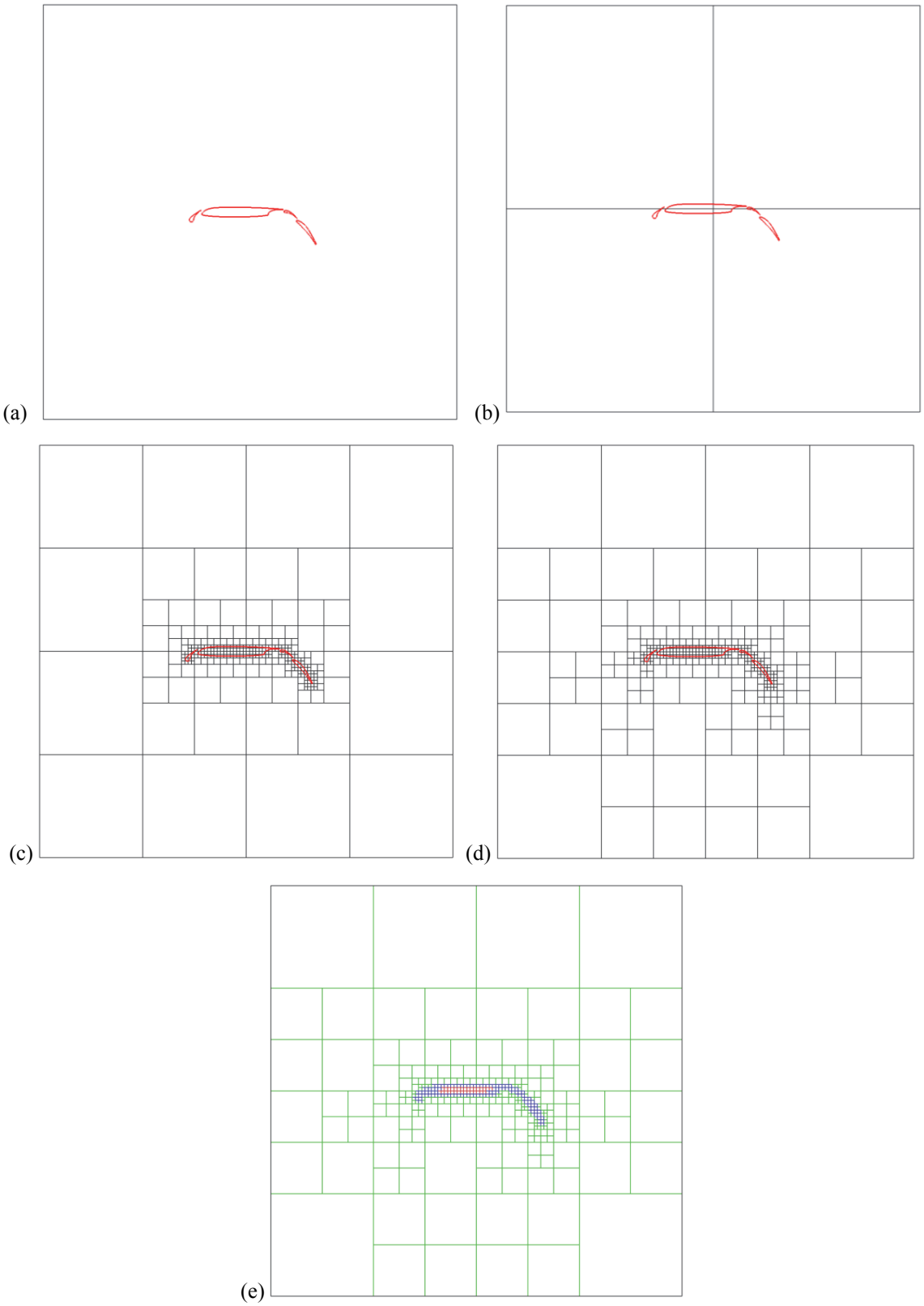
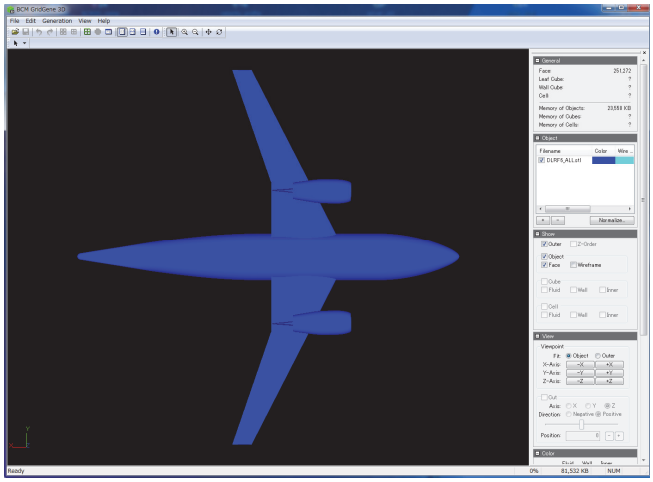


Fig. 3. Flowchart of Cube generation

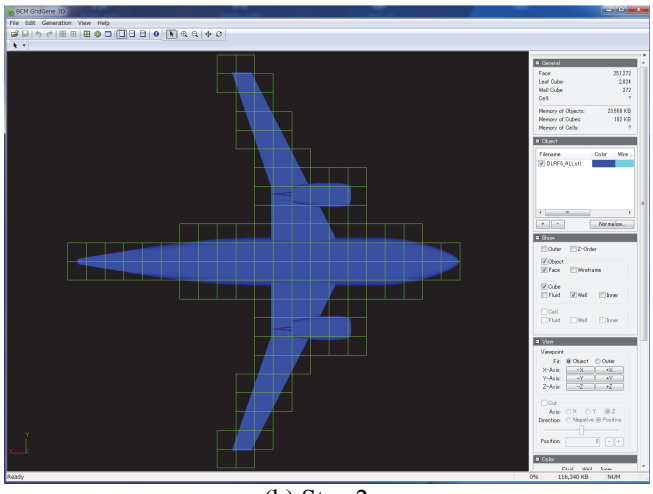
3 Mesh Generation via GUI

The BCM mesh can be interactively generated using GUI (Graphical User Interface). Figure 4 shows 3D meshing for DLR-F6 configuration.

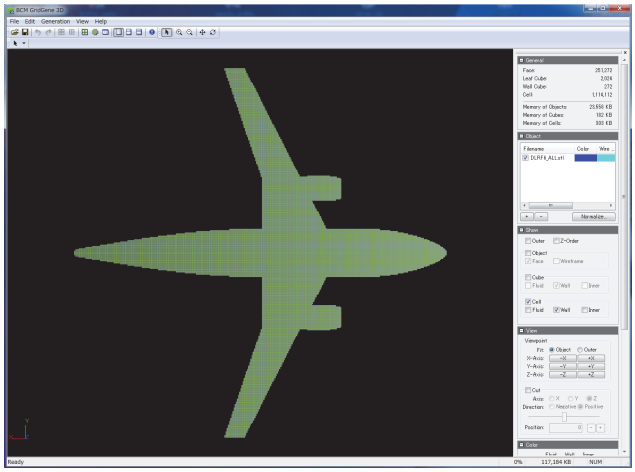
Step1: read STL data and input meshing parameters (# of Cell division, minimum Cell size, etc.)
Step2: generate Cube
Step3: generate Cell
Step4: check mesh data



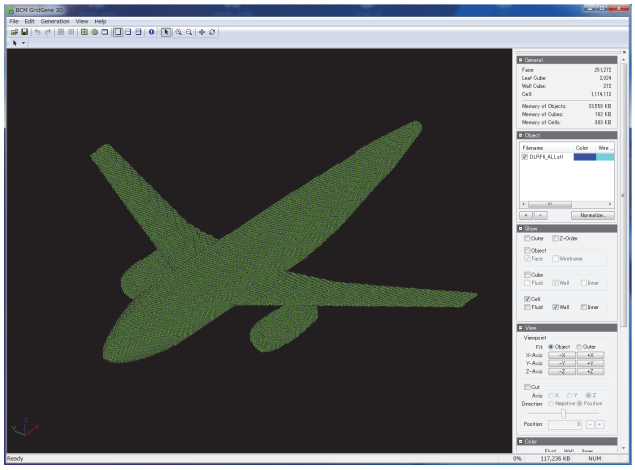
(a) Step 1



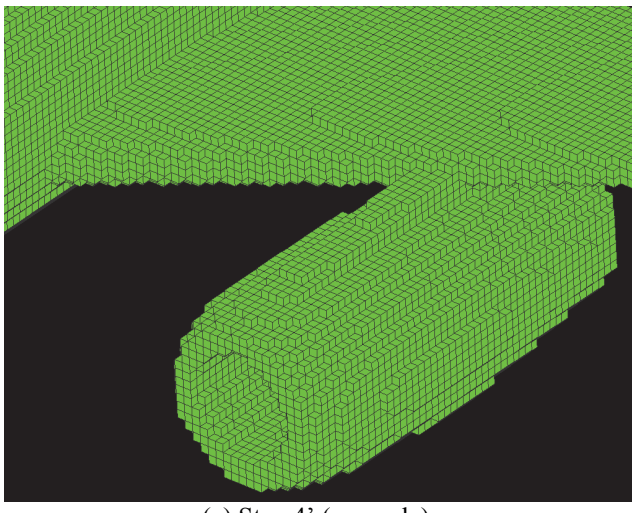
(b) Step 2



(c) Step 3



(d) Step 4



(e) Step 4' (example)

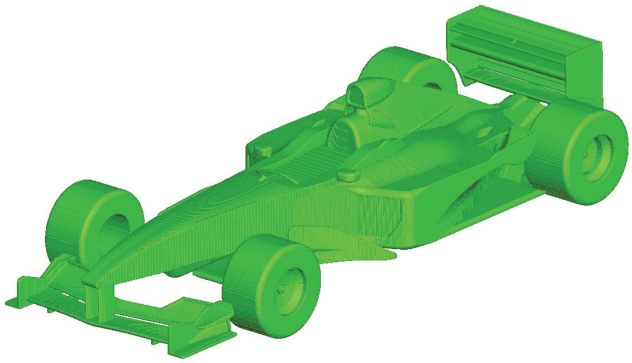
Fig. 4. BCM Meshing Using GUI

4 Large-Scale Mesh Generation

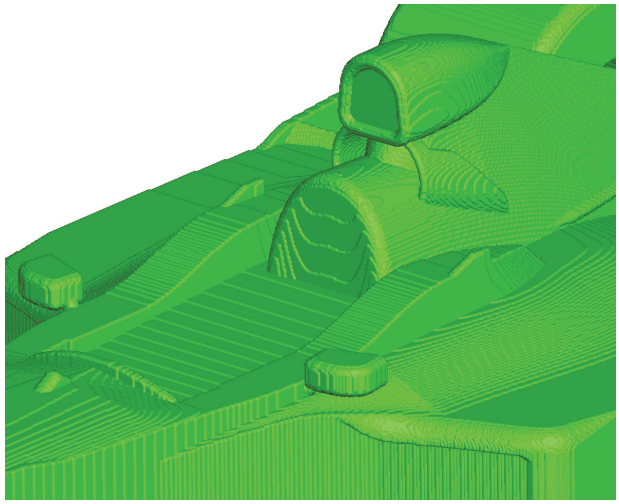
In the section, several examples of large-scale meshing are shown. The mesh is generated using Windows machine (Windows 7 64bit, Intel Xeon X5482 2.2GHz x2, 32GB memory). The objects are Formula one (F1) car model and DLR-F6.

4.1 F1 Model

The original STL data and generated mesh information for F1 model is described in Table 1. As described in the table, 800 million cells are generated in 3 minutes. The minimum cell size is 6.1e-4, which corresponds to 2.8 mm for 4.6m F1 body length. As shown in Fig. 5, the detail of the F1 model is represented.



(a) Overview



(b) Close-up view

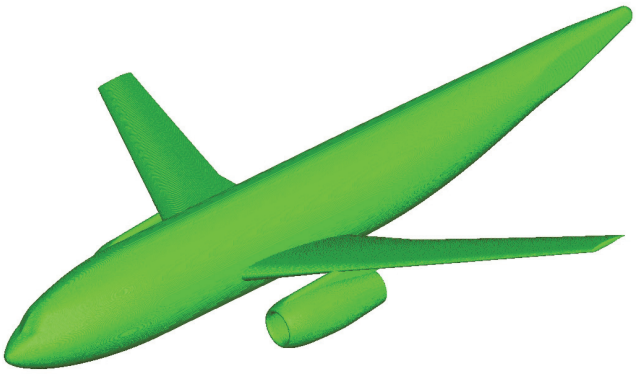
Fig. 5. Generated cells for F1 model

Table 1. Meshing information

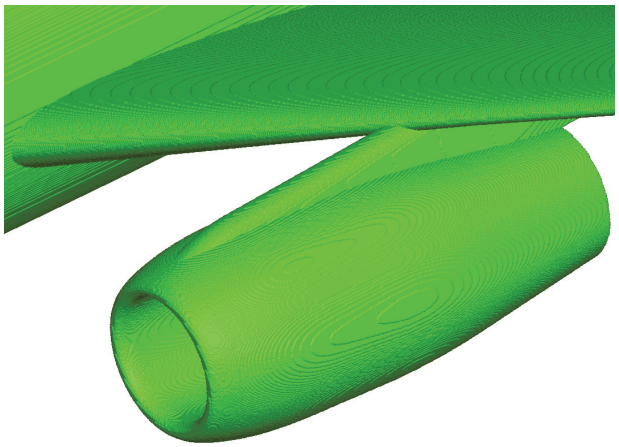
#facet	1,189,898
#cubes	3,385
#cells	887,357,440
#cells in a cube	64×64×64
min. grid spacing	6.1e-4
time[min]	3.4
Memory Usage[MB]	254
Output data size[MB]	11.1

4.2 DLR F6 Wing-Fuselage-Nacelle Model

The mesh information for an aircraft model of DLR F6 is described in Table 2 and the generated mesh is shown in Fig. 6. The meshing time is 3 minutes for 1.8 billon cells and the memory usage is very low. Although the staircase representation, the curved surface at wing camber, nacelle and fuselage are well represented. Since the minimum cell size is 3.1e-4 against the fuselage length, much finer mesh is required when the boundary layer is resolved for high-Reynolds number flow.



(a) Overview



(b) Close-up view

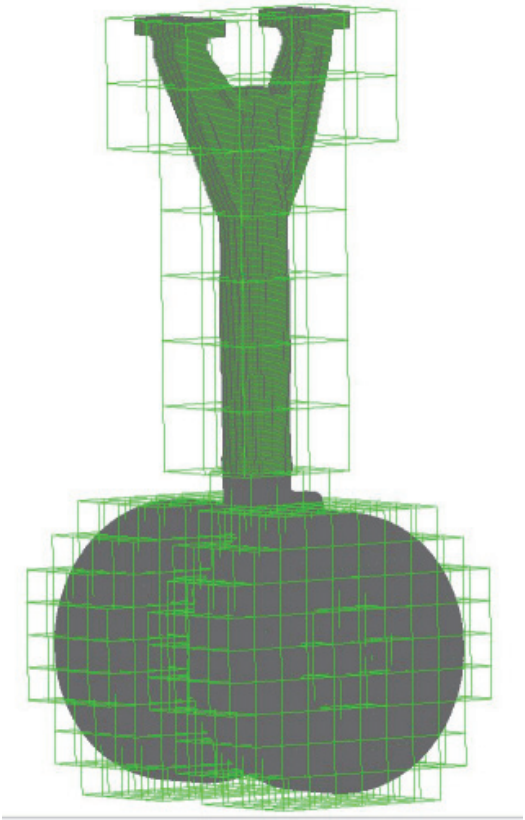
Fig. 6. Generated cells for DLR-F6

Table 2. Mesh information

#facet	251,272
#cubes	6,994
#cells	1,833,435,136
#cells in a cube	64×64×64
min. grid spacing	3.1e-4
time[min]	3.2
Memory Usage[MB]	220
Output data size[MB]	8.63

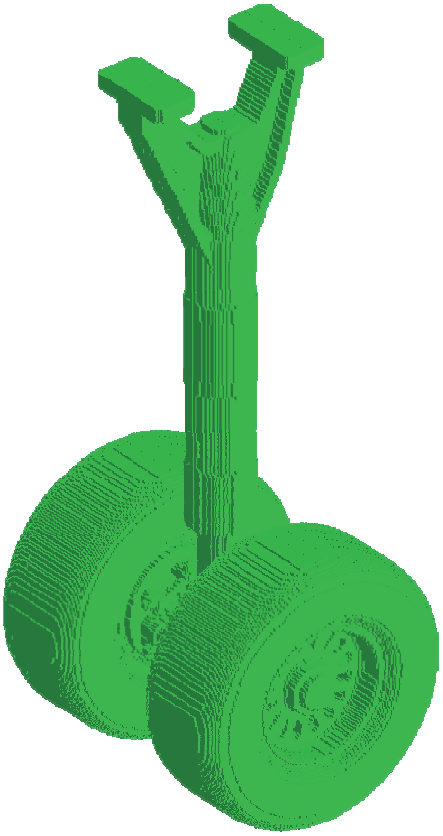
4.3 Aircraft Landing Gear

Another example of large-scale meshing of aircraft landing gear [10] is shown in Fig. 7. As shown in the figure, the complicated geometry of landing gear is well-reproduced. The cooling holes at the wheel are enough represented with the minimum mesh spacing of 1.0×10-3 D (based on the wheel diameter). The amount of mesh is 0.9 billion and the meshing time is a few minutes.

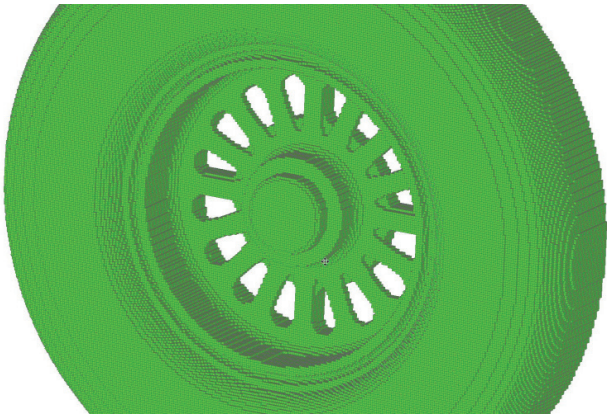


(a) Cube allocation

Fig. 7. Example of aircraft landing gear mesh (Con'd)



(b) Surface mesh



(c) Close-up view of wheel with cooling holes

Fig. 7. Example of aircraft landing gear mesh

4.4 F1 Model (50B Cells)

Much larger-scale meshing capability has to be considered for the expecting Peta/Exa scale computers. Table 6 describes the generated 50 billion cells of F1 model used in Sec. 4.1.This proves that the present meshing technique can generate 50 billion cells in 2 hours using 3GB memory. However, there are still many issues to handle large-scale mesh such as visualization problem.

Table 3. Meshing information for 50billion cell case

#facet	1,189,898
#cubes	23,416
#cells	49,106,911,232
#cells in a cube	128×128×128
min. grid spacing	9.54e-5
time[min]	103.0
Memory Usage[GB]	3.053
Output data size[MB]	442

5 Conclusion

In this paper, rapid mesh generation for block-structured Cartesian mesh, Building-Cube Method, is briefly introduced. The introduced technique helps to save the required memory for the large-scale meshing and can rapidly generate large-scale meshes. Large-scale meshing for the three cases was demonstrated, and all the meshing cases were completed in a few minutes. The meshing tool was capable to generate 50 billion cells in a few hours, however, more research has to be conducted for the efficient handling of such billion-scale meshes.

6 Contact Author Email Address

The contact author is Daisuke Sasaki at Kanazawa Institute of Technology, and the corresponding email address is dsasaki@neptune.kanazawa-it.ac.jp

Acknowledgements

This research was supported by 21226018. This work was partially supported by “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” and “High Performance Computing Infrastructure” in Japan. Part of the computations were obtained using supercomputing resources at Cyberscience Center, Tohoku University.

References

[1] Aftosmis M J, Nemec M and Cliff S E. Adjoint-Based Low-Boom Design with Cart3D. 29th AIAA Aerodynamics Conference, Honolulu, Hawaii, AIAA Paper 2011-3500, 2011.

[2] Aftosms M J, Berger M J and Alonso J J. pplications of a Cartesian Mesh Boundary-Layer Approach for Complex Configurations. 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, USA, AIAA Paper 2006-652, 2006.

[3] Takahashi S, Nonomura T and Fukuda K. A Numerical Scheme Based on an Immersed Boundary Method for Compressible Turbulent Flows with Shocks: Application to Two-Dimensional Flows around Cylinders. *Journal of Applied Mathematics*, Article ID 252478, 2014.

[4] Nakahashi K and Kim L S. Building-Cube Method for Large-Scale, High Resolution Flow Computations. AIAA Paper 2004-0423, 2004.

[5] Nakahashi K. High-Density Mesh Flow Computations with Pre-/Post-Data Compressions. AIAA paper 2005-4876, 2005.

[6] Nakahashi K, Kitoh A and Sakurai Y. Three-Dimensional Flow Computations around an Airfoil by Building-Cube Method. AIAA paper 2006-1104, 2006.

[7] Takahashi S, Ishida, T, Nakahashi K, Kobayashi H, Okabe K, Shimomura Y, Soga, T and Musa A. Study of High Resolution Incompressible Flow Simulation based on Cartesian Mesh. 47th AIAA Aerospace Sciences Meeting, AIAA Paper 2009-0563, 2009.

[8] Takahashi S. Study of Large Scale Simulation for Unsteady Flows. Ph.D. Dissertation, Department of Aerospace Engineering, Tohoku University, 2009.

[9] Ishida T, Takahashi S and Nakahashi K. Efficient and Robust Cartesian Mesh Generation for Building-Cube Method. *Journal of Computational Science and Technology*, Vol. 2, No. 4, pp. 435-446, 2008.

[10] Sasaki D, Deguchi A, Onda H and Nakahashi K. Landing Gear Aerodynamic Noise Prediction Using Building-Cube Method. Modelling and Simulation in Engineering, Vol. 2012, Article ID 632387, 2012.

[11] Fukushima Y, Misaka T, Obayashi S, Jeong S, Sasaki D and Nakahashi K. CFD-CAA Coupled Computation of Fan Noise Propagation from Engine Nacelle Based on Cartesian Mesh Method. 19th AIAA/CEAS Aeroacoustics Conference, Berlin, Germany, AIAA Paper 2013-2020, 2013.

Copyright Statement