Journal of Marine Science and Technology

Volume 28 | Issue 4 | Article 3

AUTOMATIC POLYHEDRAL MESH GENERATION FOR SHIP RESISTANCE BASED ON THE LOCALLY REFINED CARTESIAN CUT-CELL METHOD

Kwang-Leol Jeong
Research center, NextFoam, Seoul, Rep. of Korea.

Dae-Won Seo
Department of Naval Architecture and Ocean Engineering, Kunsan National University, Kunsan, Republic of Korea., dwseo@kunsan.ac.kr

Recommended Citation

Jeong, Kwang-Leol and Seo, Dae-Won (2020) "AUTOMATIC POLYHEDRAL MESH GENERATION FOR SHIP RESISTANCE BASED ON THE LOCALLY REFINED CARTESIAN CUT-CELL METHOD," *Journal of Marine Science and Technology*: Vol. 28: Iss. 4, Article 3.
DOI: 10.6119/JMST.202008_28(4).0003
Available at: https://jmstt.ntou.edu.tw/journal/vol28/iss4/3

AUTOMATIC POLYHEDRAL MESH GENERATION FOR SHIP RESISTANCE BASED ON THE LOCALLY REFINED CARTESIAN CUT-CELL METHOD

Acknowledgements

This work was supported by the Industrial FundamentalTechnology Development Program (10060329, Development of a numerical simulator for the environmental load on a ship and offshore polar region), funded by the Ministry of Trade, Industry & Energy (MI, Korea) and Kunsan National University

Journal of Marine Science and Technology, Vol. 28, No. 4, pp. 249-257 (2020)
DOI: 10.6119/JMST.202008_28(4).0003

AUTOMATIC POLYHEDRAL MESH GENERATION FOR SHIP RESISTANCE BASED ON THE LOCALLY REFINED CARTESIAN CUT-CELL METHOD

Kwang-Leol Jeong[1] and Dae-Won Seo[2]

Key words: automatic mesh generation, Cartesian cut cell, polyhedral mesh.

ABSTRACT

This paper introduces an automatic grid generation algorithm, based on a locally refined cut-cell method, to calculate ship resistance. Sharp edges and apexes are not accurately expressed in the cut-cell method, and the accuracy of the calculation in the turbulence boundary layer is lower than that of other body-fitted methods. In this study, an automatic mesh generation method is developed to effectively represent edge and apex, and a boundary layer grid generation algorithm is developed to generate meshes that improve the accuracy of ship resistance simulation. To validate the method, the resistances of the KRISO container ship and Japan bulk carrier are calculated using grids generated through the proposed method, and the results are compared with the existing experimental data. The results are qualitatively reasonable, and, therefore, the developed algorithms of the grid generation method can be applied to calculate the resistance of a ship.

I. INTRODUCTION

To calculate highly nonlinear partial differential equations with a numerical method, the computational domain of the problem must be properly discretized. Most of the numerical methods that solve fluid flow equations use a mesh for domain discretization. Even though some mesh-free methods (Monaghan, 1992; Chiu et al., 2011), which use particles or nodes to discretize a domain, have been developed, the mesh-generation method is still faster and produce more accurate results.

Paper submitted 09/07/19; revised 12/26/19; accepted 04/17/20. Corresponding Author: Dae-Won Seo (e-mail: dwseo@kunsan.ac.kr)
[1]*Research center, NextFoam, Seoul, Rep. of Korea.*
[2]*Department of Naval Architecture and Ocean Engineering, Kunsan National University, Kunsan, Republic of Korea.*

Mesh generation may require much time and effort, depending on the complexity of the geometry and the refinement level. As a result, research has been conducted to automatize the grid generation process. Most studies (Bowyer, 1981; Fortune, 1987) focus on the generation of an unstructured tetrahedral mesh using Delaunay triangulation. Some methods of automatic structured grid generation have also been developed (Zhang and Jia, 2018; Sarrate and Matthew, 2014), as the structured grid is much more efficient and produces better results than the unstructured one. The cut-cell method was also adapted for automatic grid generation; however, when considering turbulence problems, the accuracy of the turbulence boundary layer, calculated with a mesh generated using this method, was lower than that of other body-fitted grids, as the grid line is not aligned with the body surface. To solve this problem, Delanaye et al. (1999) and Fujimoto et al. (2009), respectively, proposed a hybrid method and a body-fitted Cartesian grid method, in which a prism layer is employed around the body surface to improve the accuracy of the turbulence boundary layer. The cut-cell method, however, also has problems when sharp edges and apex need to be represented. To mitigate these problems, fully polyhedral data structures have to be employed. Furthermore, Liu et al. (2017) developed an automatic polyhedral mesh generation method in which sharp edges are represented by employing this data structure.

Two types of local refinement methods, the anisotropic (2N-tree), and isotropic (octree) refinement are often employed. In the first one, each cell is split into two cells by cutting it in one direction and a polyhedral data structure is used, as the data structure is too complex (Capizzano, 2018). Meanwhile, in the octree refinement method, each cell is split into eight cells by cutting it in three directions and a hierarchical parent-child data structure is used.

To simulate ship resistance accurately and efficiently, the waves generated by the ship and the turbulent flow around the hull surface have to be accurately calculated. As a result, the grid around the free surface, which may be far from the ship, has to be small enough in the normal direction to reduce the smearing of the volume of fluid (VOF) function. In this case,

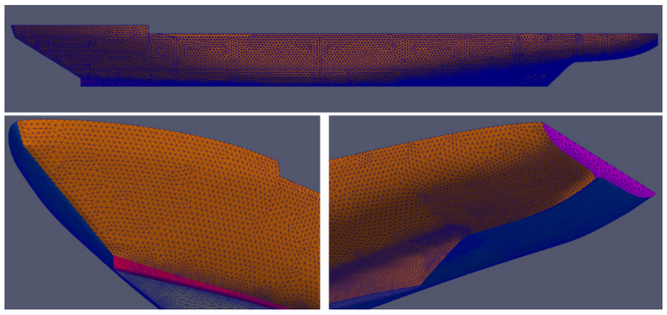
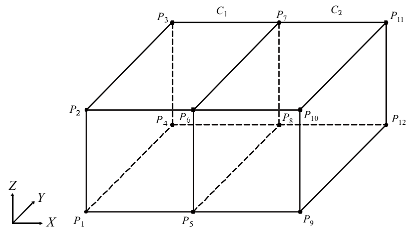


Fig. 1. Example of hull form expressed with triangular elements

therefore, applying anisotropic refinement is much more efficient than octree refinement. Considering the calculation of the turbulence flow around the hull, the boundary layer grid may be generated as in Delanaye et al. (1999) or Fujimoto et al. (2009).

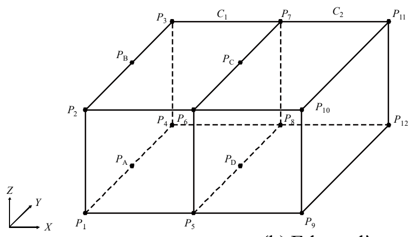
SnappyHexMesh is a standard mesh tool, developed in OpenFOAM, that automatically generates a polyhedral mesh. SnappyHexMesh isotropically refines the grid; therefore, the generated grid is not suitable for the ship-resistance simulations with sharp edges, as described by Kortelainen (2009).

In this paper, an algorithm to generate a fully polyhedral mesh based on the locally refined cut-cell algorithm is proposed. The initial grid is refined with an anisotropic refinement technique, which is more efficient than isotropic refinement, around the free surface. Furthermore, methods that address the issues related to the sharp edges and apexes are also presented. For the turbulence boundary layer, the prism layer is generated by extracting the front of the grid; the grid points around the body boundary are also moved. Each displacement of each grid point is determined by calculating the Laplace equation. To validate the proposed method, the grids for the resistance of KCS and JBC are generated, and the resistance calculations are also conducted with them.



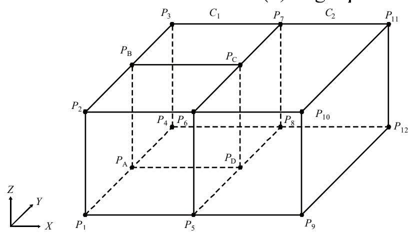
face	point	owner	neighbor
F_1	P_8, P_7, P_6, P_5	C_1	C_2
F_2	P_1, P_2, P_3, P_4	C_1	-
F_3	P_1, P_5, P_6, P_2	C_1	-
F_4	P_2, P_6, P_7, P_3	C_1	-
F_5	P_3, P_7, P_8, P_4	C_1	-
F_6	P_4, P_8, P_5, P_1	C_1	-
F_7	$P_{12}, P_{11}, P_{10}, P_9$	C_2	-
F_8	P_{12}, P_8, P_7, P_{11}	C_2	-
F_9	P_{11}, P_7, P_6, P_{10}	C_2	-
F_{10}	P_{10}, P_6, P_5, P_9	C_2	-
F_{11}	P_9, P_5, P_8, P_2	C_2	-

(a) Initial call



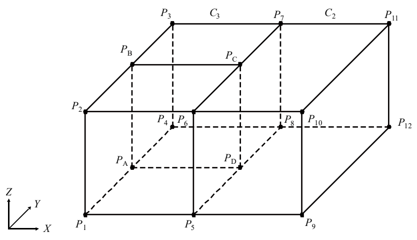
face	point	owner	neighbor
F_1	$P_8, P_7, P_C, P_6, P_5, P_D$	C_1	C_2
F_2	$P_1, P_2, P_B, P_3, P_4, P_A$	C_1	-
F_3	P_1, P_5, P_6, P_2	C_1	-
F_4	$P_2, P_6, P_C, P_7, P_3, P_B$	C_1	-
F_5	P_3, P_7, P_8, P_4	C_1	-
F_6	$P_4, P_8, P_D, P_5, P_1, P_A$	C_1	-
F_7	$P_{12}, P_{11}, P_{10}, P_9$	C_2	-
F_8	P_{12}, P_8, P_7, P_{11}	C_2	-
F_9	P_{11}, P_7, P_6, P_{10}	C_2	-
F_{10}	P_{10}, P_6, P_5, P_9	C_2	-
F_{11}	P_9, P_5, P_8, P_{12}	C_2	-

(b) Edge split



face	point	owner	neighbor
F_1	P_8, P_7, P_C, P_D	C_1	C_2
F_A	P_C, P_6, P_5, P_D	C_1	C_2
F_2	P_1, P_2, P_B, P_A	C_1	-
F_B	P_B, P_3, P_4, P_A	C_1	-
F_3	P_1, P_5, P_6, P_2	C_1	-
F_4	P_2, P_6, P_C, P_B	C_1	-
F_C	P_C, P_7, P_3, P_B	C_1	-
F_5	P_3, P_7, P_8, P_4	C_1	-
F_6	P_4, P_8, P_D, P_A	C_1	-
F_D	P_D, P_5, P_1, P_A	C_1	-
F_7	$P_{12}, P_{11}, P_{10}, P_9$	C_2	-
F_8	P_{12}, P_8, P_7, P_{11}	C_2	-
F_9	P_{11}, P_7, P_6, P_{10}	C_2	-
F_{10}	P_{10}, P_6, P_5, P_9	C_2	-
F_{11}	P_9, P_5, P_8, P_{12}	C_2	-

(c) Face split



face	point	owner	neighbor
F_A	P_C, P_6, P_6, P_D	C_1	C_2
F_E	P_A, P_B, P_C, P_D	C_1	C_3
F_1	P_D, P_C, P_7, P_8	C_2	C_3
F_2	P_1, P_2, P_B, P_A	C_1	-
F_B	P_B, P_3, P_4, P_A	C_3	-
F_3	P_1, P_5, P_6, P_2	C_1	-
F_4	P_2, P_6, P_C, P_B	C_1	-
F_C	P_C, P_7, P_3, P_B	C_3	-
F_5	P_3, P_7, P_8, P_4	C_3	-
F_6	P_4, P_8, P_D, P_A	C_3	-
F_D	P_D, P_5, P_1, P_A	C_1	-
F_7	$P_{12}, P_{11}, P_{10}, P_9$	C_2	-
F_8	P_{12}, P_8, P_7, P_{11}	C_2	-
F_9	P_{11}, P_7, P_6, P_{10}	C_2	-
F_{10}	P_{10}, P_6, P_5, P_9	C_2	-
F_{11}	P_9, P_5, P_8, P_{12}	C_2	-

(d) Cell split

Fig. 2. Schematics of grid refinement.

II. DATA FORMAT

1. Data format for hull shape import

An STL file is considered for the data type of the shape of the hull (Fig. 1). The file contains the number of solids used in the mesh generation process. Furthermore, each solid is composed of a group of triangular elements, which are composed of three nodes and three segments. To measure the force or moment in a part of a ship, the hull surface is represented by several solids.

2. Data format for the fully polyhedral grid

The grid generated by the present algorithm is used to calculate the ship resistance with OpenFOAM; therefore, the format of the grid data is the same as that of the OpenFOAM grid data (fully polyhedral data format). The grid data are composed of the point (position vector of the grid point), face (point list comprising the face), owner cell and neighbor cell entities. A face is also classified as internal or boundary. An internal face belongs to two cells. The cell, whose number is lower than that of another, is defined as owner cell and another cell is defined as neighbor cell. The normal vector of an internal face, which points out the neighbor cell, is defined following the right-hand rule. A boundary face belongs to a cell. The cell is the owner cell of the face and the face does not have neighbor cell. The normal vector of the face points out the outside of the calculation domain. The face with lower owner cell number is assigned lower face number. If some faces have the same owner cell number, the face with lower neighbor cell number is numbered lower

III. LOCAL ANISOTROPIC REFINEMENT

There are two major grid refinement methods for the body boundary in the Cartesian grid. In the isotropic method

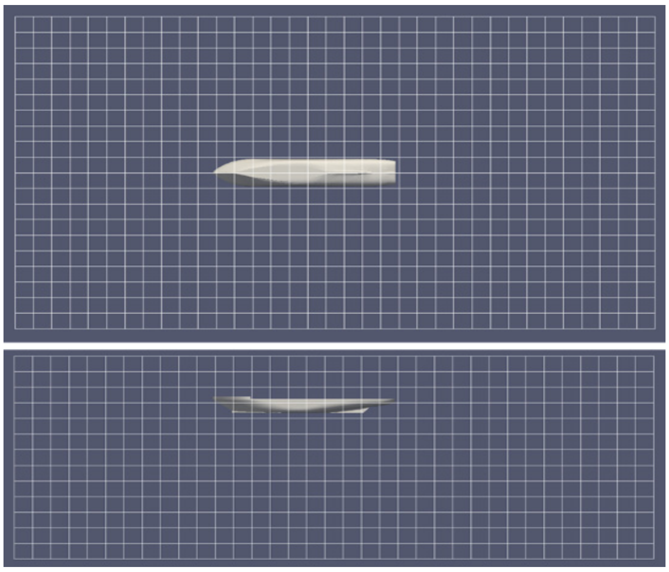


Fig. 3. Calculation domain and initial grid arrangement.

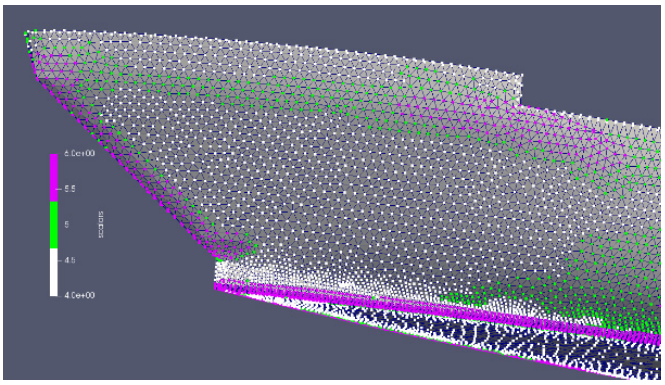


Fig. 4. Z-direction refinement level of each node.

(octree), each cell is split into 8 cells, in the X-, Y-, and Z-directions, concurrently. Meanwhile, in the anisotropic refinement, each cell is split into two cells in one direction.

Furthermore, there are two data structures for the refined Cartesian grid (parent-child structure and fully polyhedral data structure). If the flow solver uses a parent-child data structure, the isotropic refinement results in reduced calculation time and improved accuracy. The fully polyhedral data structure is usually adopted for anisotropic refinement, in which fewer grid cells are required. In the ship resistance calculation, anisotropic refinement significantly reduces the number of cells needed around the free surface, compared to isotropic refinement.

1. Grid refinement process

The grid refinement process is conducted in three stages, which are the edge, face, and cell splits.

Figure 2 shows the configuration of the points composing the faces and the owner and neighbor cells after each stage. In the edge split stage, four new points are added. In the face split stage, 4 new faces are created owing to the cell split. In the cell split stage, a new face composed of 4 points is added to split the cell. Furthermore, a new cell is created and added to the mesh. In some faces, the order of the points that comprise the faces is inversed due to the change in the owner and neighbor numbers.

2. Selection of refinement target cells

The initial grid is generated using regular hexahedron cells in a rectangular parallelepiped domain. The size of a cell is 10% the length of the ship in each direction. Meanwhile, the length, width, and depth of the domain are 3.5, 2.0, and 1.3 times the length of the ship, respectively. Figure 3 shows the calculation domain and the initial grid.

For the ship resistance calculation, the grid must be generated considering the peculiarities of high Reynolds flows and free surface flows. In the first scenario, the grid refinement level around the hull surface is determined by its curvature, which is calculated in each direction, and at each node, with the 2nd order derivative of the hull surface, using triangular elements proposed by Csakany and Wallace (2000). The grid size was smaller than 10% of the radius of curvature. The maximum and minimum refinement levels were set to level 6 and 4, to prevent using too many cells and to proper hull reconstruction, respectively. The refinement levels in three directions are saved at the data structure of each node. Figure 4 shows the refinement level in the Z-direction on the hull surface. In Fig 4, the white color indicates nodes with level 4 of refinement, whose curvature radius in the Z-direction was larger than ten times of the refined cell size. The refined cell size in z-direction is $\Delta z_{initial}/2^n$, where n is the refinement level. Furthermore, the green and purple colors represent nodes with level 5 and 6 of refinement, respectively. The X and Y-direction refinement levels were determined by the curvature radius in the corresponding directions. The refinement target cell is set if the distance from the nearest hull surface node is shorter than the user-defined distance. In this study, the user-defined distance is 5 times the refined cell size.

All cells on the water level were refined 6 times in the Z-direction. In this case, the cell height was approximately 0.15% of the length of the ship. Generally, the wave height around a commercial ship is approximately 1% of the length of the ship; therefore, the cell height was approximately 1/6 the wave height. The cells in the Kelvin wave region were refined 4 times in the X- and Y- directions; however, those cells outside of this region were not refined in these directions. Figure 5 shows an example of the refined Cartesian cell around the ship and free surface.

IV. DEFINITION OF THE HULL SHAPE

Figure 6 shows the process to establish the body shape in a 2D Cartesian mesh. First, the intersection points between the body surfaces and the grid lines were defined. The grid lines

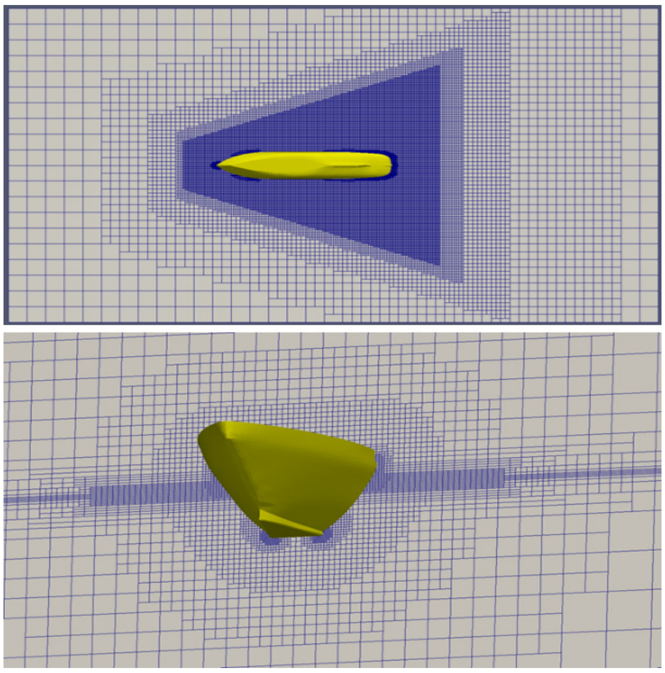


Fig. 5. Sliced planes of the refined grid around the ship.

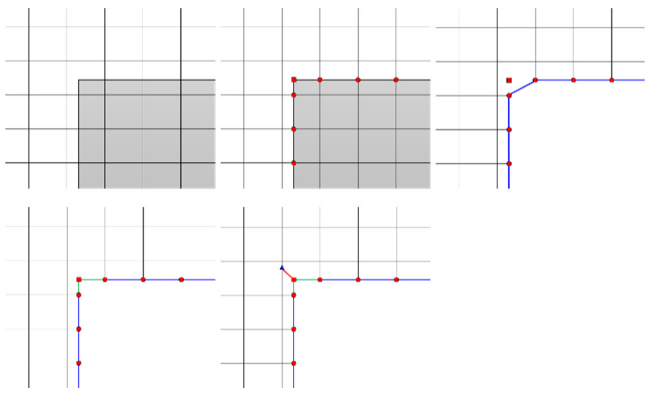
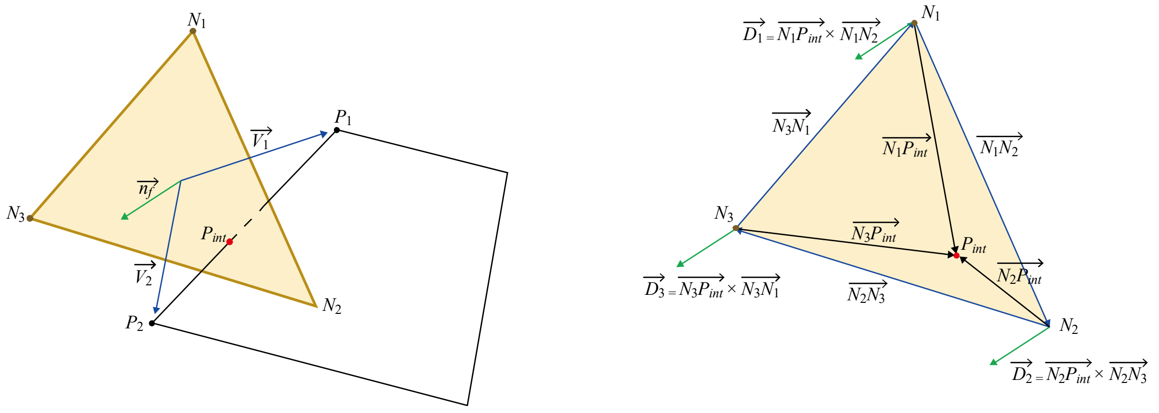


Fig. 6. Body shape procedure using a simple 2D case.


N_1
$\overrightarrow{V_1}$
P_1
$\overrightarrow{n_f}$
N_3
P_{int}
$\overrightarrow{V_2}$
N_2
P_2
$\overrightarrow{D_1} = \overrightarrow{N_1P_{int}} \times \overrightarrow{N_1N_2}$
N_1
$\overrightarrow{N_3N_1}$
$\overrightarrow{N_1P_{int}}$
$\overrightarrow{N_1N_2}$
N_3
$\overrightarrow{N_3P_{int}}$
P_{int}
$\overrightarrow{N_2P_{int}}$
$\overrightarrow{D_3} = \overrightarrow{N_3P_{int}} \times \overrightarrow{N_3N_1}$
$\overrightarrow{N_2N_3}$
N_2
$\overrightarrow{D_2} = \overrightarrow{N_2P_{int}} \times \overrightarrow{N_2N_3}$
Fig. 7. Definition sketch for the intersection point.

were, thereafter, split by the intersection points. The face was also split by connecting them and removing the empty face that was out of the calculation domain. To recover a sharp edge or vertex, the corresponding edge was split by the feature node. To ensure the stability of the solution, concave cells were also split. If there were two or more intersection points in an edge, the surface could not be simply defined. This procedure becomes more complex in three-dimension (3D); therefore, in this chapter, the steps are described in detail.

1. Finding the intersection point

Figure 7 shows a triangular element, composed of points N_1,N_2,N_3, which is intercepted by a grid line $(\overline{P_1, P_2})$ at P_{int}. Furthermore, the triangular element and the grid line are on different planes.

Before calculating the intersection point between a triangular element and a grid line in 3D, two dot products between the normal vector of the triangular element $(\overrightarrow{n_{\text{f}}})$ and the direction vectors $(\overrightarrow{V_1}, \overrightarrow{V_2})$ are calculated to check if the intersection point is between points P_1 and P_2.
$\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ are the directional vectors that connect a point on the triangular element to the endpoints of the grid line. If the intersection point is between these two points, the product of the two dot products will be negative. Moreover, if the point is on the plane, this product will be zero and the displacement of the point in the normal direction of the plane is small. The intersection point in the triangle is based on the direction of three cross products $(\overrightarrow{D_1}, \overrightarrow{D_2}$, and $\overrightarrow{D_3})$. If the intersection point is very close or on the element segment, the grid points P_1 and P_2 are moved very small.

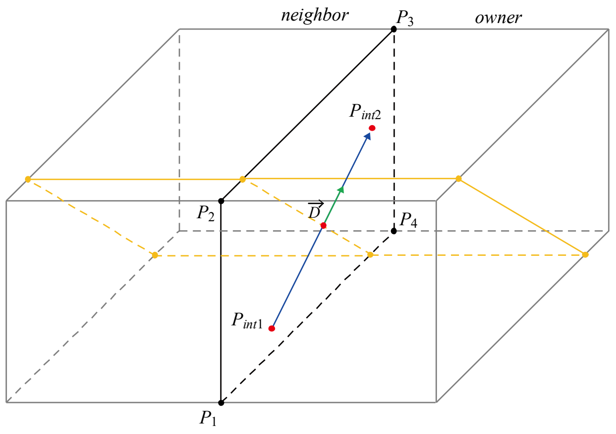


Fig. 8. Schematic sketch for cell split for multiple feature intersection points in a face.

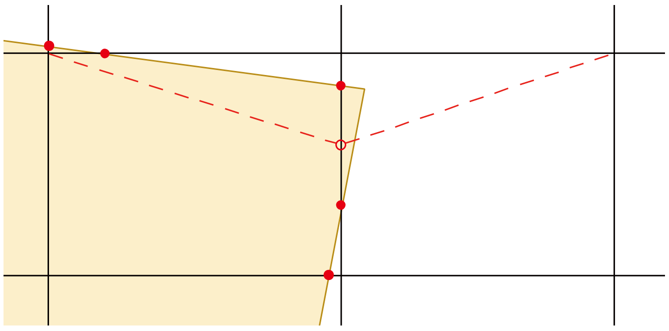


Fig. 9. Schematic of splitting the faces that have an edge with two or more intersection points.

A long computational time is required to execute this procedure. To mitigate this problem, bucket sort (Liu et al., 2017) and bounding box check (Hafex and Oshima, 1998) are employed.

2. Cell split considering a sharp edge and apex

Before the cells are cut, the intersection points between the feature segment and grid face must be defined when a sharp edge is present. If two or more intersection points in a grid face are present, the feature lines may be miss calculated. If a face (P1, P2, P3, and P4, in Fig. 8) has three or more intersection points, the owner and neighbor cells are also split by the plane that includes the mid-point of the two intersection points, as shown in Fig. 8.

Cells that include a feature node are also defined before cutting the cells. Furthermore, if a cell contains two or more feature nodes, the cell is also split by the plane that includes the mid-point of the two nodes, and that has a normal vector aligned with the directional vector connecting both nodes.

3. Definition of a smooth surface

The body boundary is defined by cutting the grid line and face using intersection points between the grid lines and triangular elements. If, however, two or more intersections were present on a grid line, the body boundary could not be accurately defined. Even when thick and smooth bodies are considered, an edge may be intercepted by two different points.

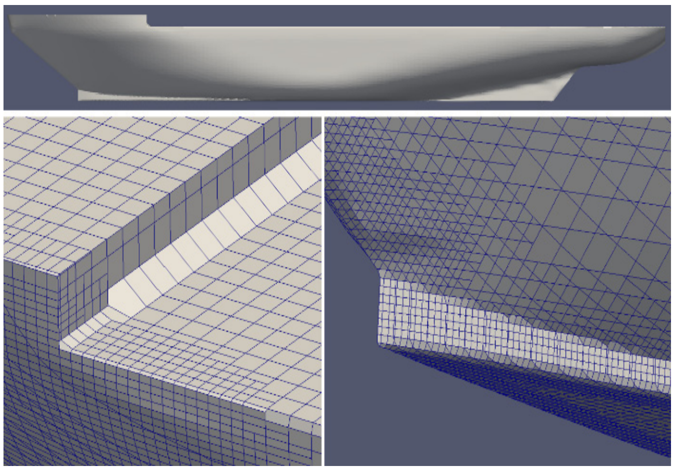
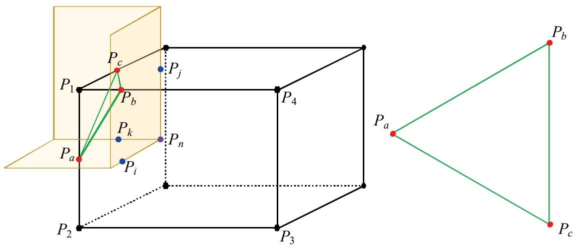
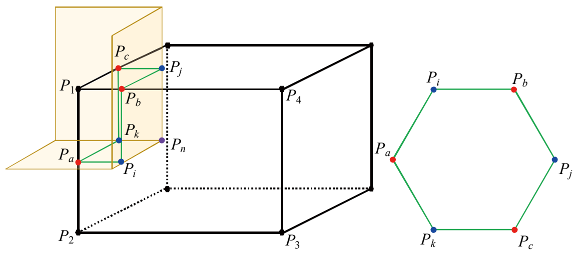


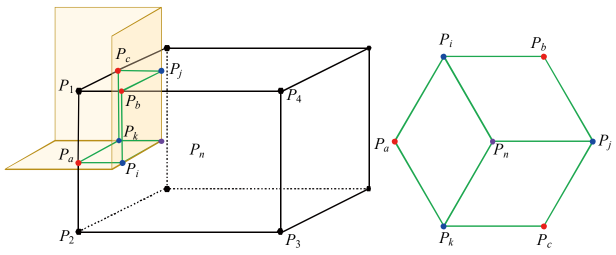
Fig. 10. Hull surface generated by cut-cell.



(a) Reconstructed body boundary using cut cell



(b) Adding feature intersection points for sharp edge



(b) Spilt of body boundary face with feature node for apex

Fig. 11. Schematic of the reconstruction of a sharp edge and an apex.

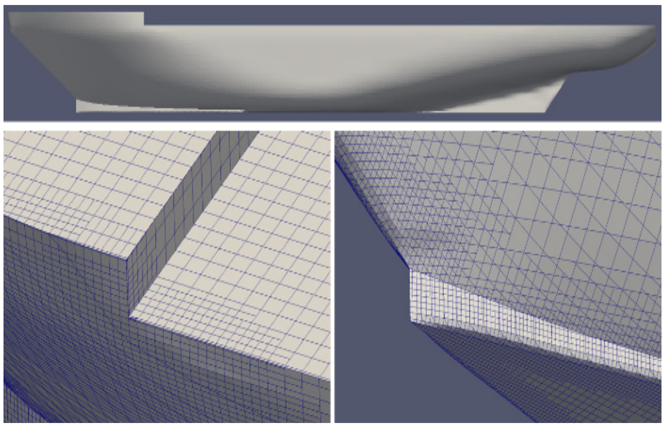


Fig. 12. Generated hull surface after the feature line and feature point are defined.

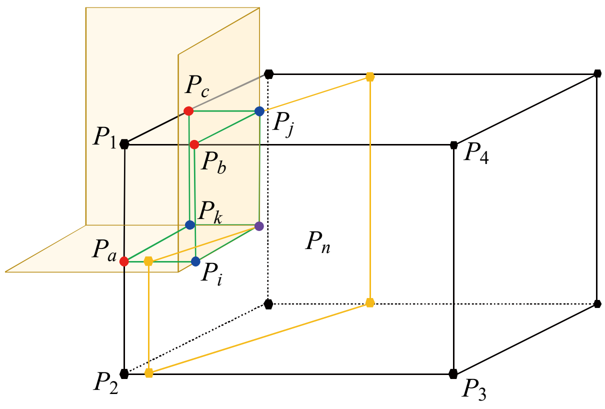
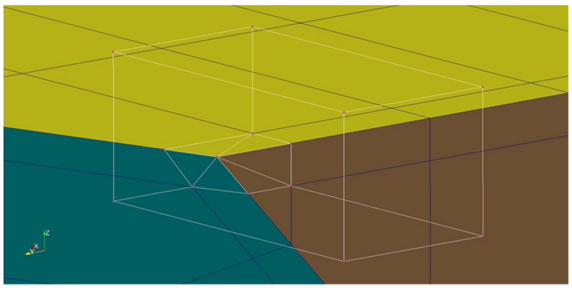


Fig. 13. Schematic of the split procedure of a concave cell.


(a) before

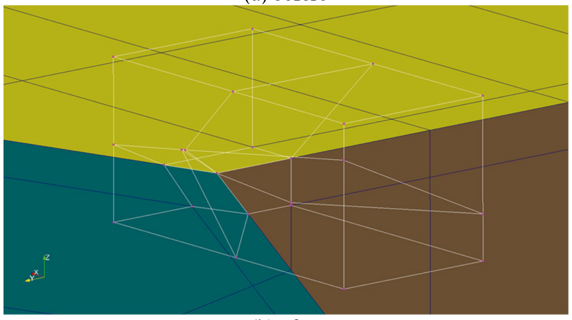

(b) after

Fig. 14. Example of concave cell split

The edges and face are split using the mid-point and the edge connecting the mid-points, respectively, as shown in Fig. 9.

If there grid lines that have two or more intersection points are absent, the grid lines are cut by the intersection points. The body boundary faces are, thereafter, reconstructed by connecting them. Figure 10 shows an example of a generated hull surface, in which the smooth surface is accurately defined but does not have sharp edges and apex.

4. Definition of sharp edge and apex

The body boundary generated when the rectangular parallelepiped created with the intersection points (P_a, P_b, P_c) between the triangular elements and the grid lines are shown in Fig. 11 (a). The grid face {P_1, P_2, P_3, P_4} is, thereafter, redefined as {P_a, P_2, P_3, P_4, P_b} by adding P_a and P_b, and removing P_1. To improve the body boundary, the feature intersection points (P_i, P_j, P_k) between the feature segments and grid faces are added to the grid faces and the previous grid face {P_a, P_2, P_3, P_4, P_b} is reconstructed as {P_a, P_2, P_3, P_4, P_b, P_i}, as shown in Fig. 11 (b). At this stage, the body boundary face consists of {P_a, P_i, P_b, P_j, P_c, P_k}. To represent the apex, the body boundary faces are decomposed by connecting the feature intersection points with the feature node (P_n), as shown in Fig. 11 (c). Figure 12 shows an indicates this reconstruction for the hull surface.

V. IMPROVEMENT OF GRID QUALITY

1. Split of Concave Cells

After reconstructing the sharp edge and apex, the shape of some body boundary cells may be concave, which hinders the stability and accuracy of the solution, as shown in Fig. 11 (c). To address this issue, the concave cells were split by a plane that contains P_j and P_n and whose normal vector is the sum of the normal vectors of faces {P_i, P_b, P_j, P_n} and { P_j, P_c, P_k, P_n}, as shown in Fig. 13. This process is conducted iteratively until all cells have a convex shape. Figure 14 shows an example of a concave cell that was split into four convex cells.

2. Elimination of small grids

Very small cells were generated after a cut cull. To ensure the stability of the solution, the time step of the solution has to be small enough to result in a Courant–Friedrichs–Lewy number lower than one. To increase the minimum time step and, therefore, reduce the time needed to calculate the solution, small cells have to be merged, enlarged, or removed. To solve this problem, Powell (1998) and other researchers suggested merging each small cell with one of their neighbors, by removing the longest Cartesian edge of the small cell.

In this study, however, some cells were split owing to their feature line or concavity. As a result, the merging process suggested by Powell (1998) could not be employed. Each small cell was, thereafter, merged by removing the smallest edge

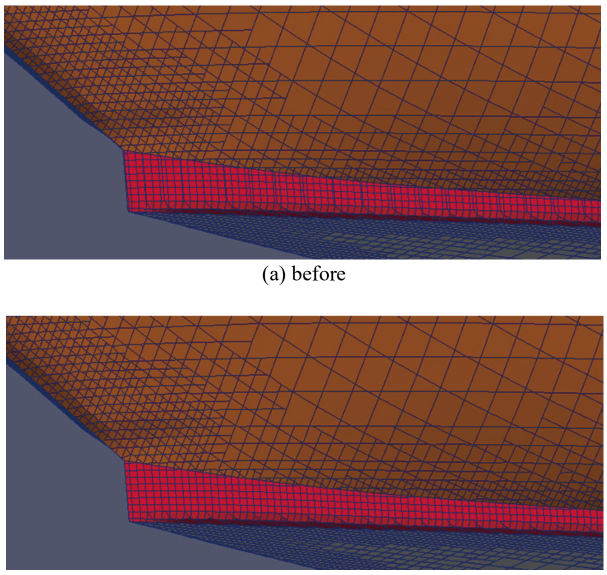


Fig. 15.Comparison of hull surface grids before and after small grids removal.

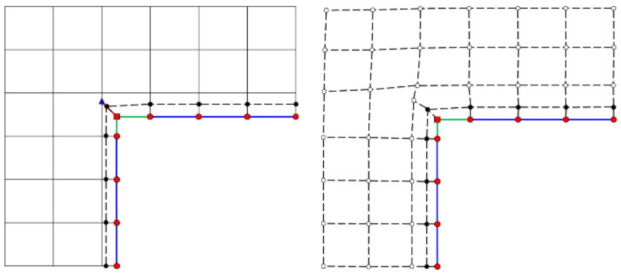


Fig. 16. Schematic sketch for boundary layer cell generation.

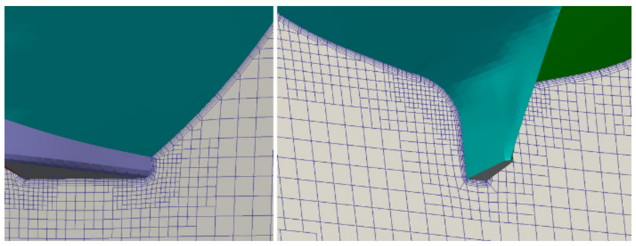


Fig. 17. Boundary layer cells around the hull surface.

through an iteratively merging procedure of its two points until all the small cells were removed. Figure 15 shows the hull surface grid before and after the removal of cells that are smaller than 5% of a regular cell.

VI. BOUNDARY LAYER GRID GENERATION

The thickness of the first boundary layer cell is y+ = 50. To calculate the boundary layer thickness, the viscous stress on the hull surface was estimated. The frictional resistance and frictional stress were calculated with the formulation Eq. 1 suggested by Schlichting and Kesin (1979) and Eq. 2, respectively. Meanwhile, the frictional velocity and thickness of the first boundary layer (y) were calculated by Eqs. 3 and 4, respectively.

$$C_f = \left[2\log_{10}(Rn) - 0.65\right]^{-2.3} \tag{1}$$

$$\tau = \frac{1}{2} C_f \rho U^2 \tag{2}$$

$$U^* = \sqrt{\tau/\rho} \tag{3}$$

$$y = \frac{y^+(=50)\nu}{U^*} \tag{4}$$

The boundary grid is generated by copying the surface grid, as shown at the left of Fig. 16. Afterward, the location of the center of the cells around the hull surface was calculated by obtaining their displacement through the Laplace equation Eq. 5.

$$\nabla \cdot \left(\Gamma \nabla \vec{R}\right) = 0 \tag{5}$$

The boundary condition is, thereafter, imposed to the nearest copied boundary layer cells of the hull, and the diffusivity is set constant. The displacement of the grid point was calculated by the inverse-distance weighted average of the neighboring cells. Figure 17 shows the generated boundary layer around the hull surface. The number of boundary layers was 5 and the expansion ratio was 1.3.

VII. RESISTANCE CALCULATION WITH THE GENERATED GRIDS

To evaluate the proposed method, grids were generated to calculate the resistances of the KRISO container ship (KCS) and Japan bulk carrier (JBC). The domain size was proportionally equal to that discussed in Section III.2. The number of the boundary layer was equal to 5 and the expansion ratio was 1.3. KCS is a relatively high-speed commercial ship, while JBC is a slow speed ship. Both calculations were conducted considering model scales of the KCS (scale: 31.599) and JBC (scale: 40.0). The draft of KCS and JBC were, 19.0 m and 25.0 m in full scale, respectively, and their speed was, respectively, 24.0 knots and 14.5 knots, that is, Fr=0.26, and Fr=0.142. More details about the KCS have been discussed in Van et al. (1997) and Kim et al. (2011). Meanwhile, additional information about JBC can be found at http://www.t2015.nmri.go.jp/jbc.html. The resistance of both ships was calculated with ESPER, an OpenFOAM solver developed by NEXTfoam Co., Ltd. The k-OmegaSST with the Spalding wall function was selected as the turbulence model and wall function, respectively, and the VOF method was also

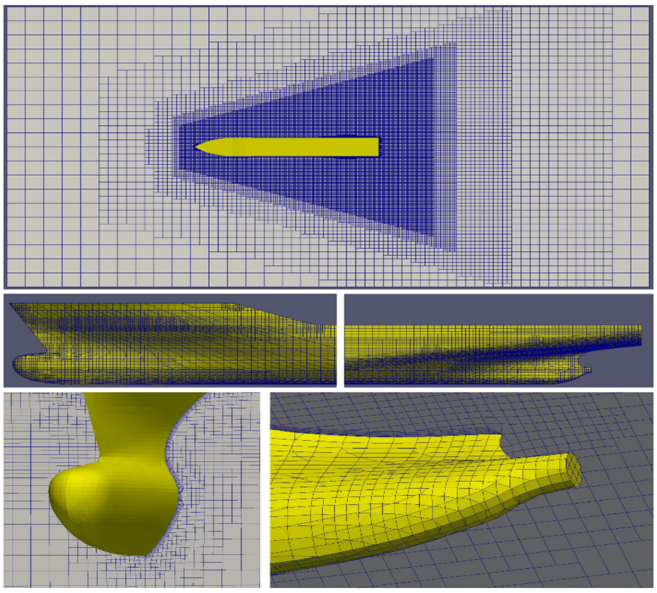


Fig. 18. Grid generated for the KCS resistance calculation.

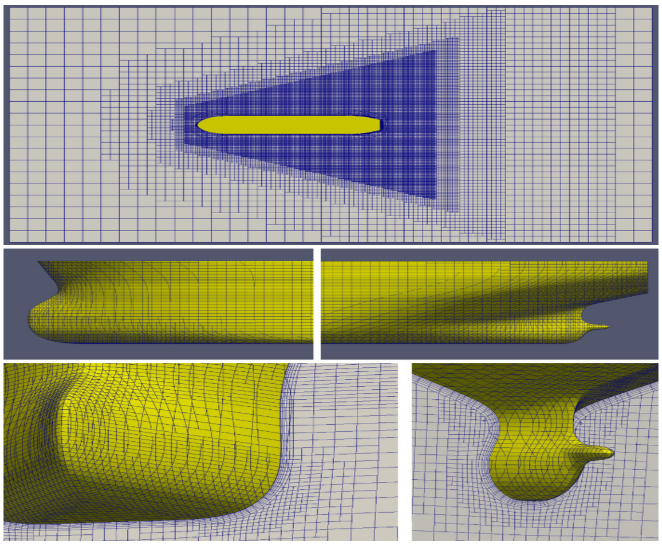


Fig. 19. Grid generated for the JBC resistance calculation

used for the free surface. The Von Karman constant, C_μ, E, and β_1 were set to 0.4187, 0.09, 9.793, and 0.075, which are the same used in the FLUENT commercial software. The running attitude is considered using the dynamic mesh method of OpenFOAM.

Figure 18 shows the grid used to calculate the KCS resistance. The number of cells was 1,113,525 and the grid was generated in approximately 10 min. Grid refinement, cut grid, and add layer took approximately 2, 4, and 4 min, respectively. The grid generation time depended on the number of triangular elements. Meanwhile, Figure 19 shows the grid used to calculate the JBC resistance. The number of cells was 1,025,838 and the mesh was also generated in approximately 10 min.

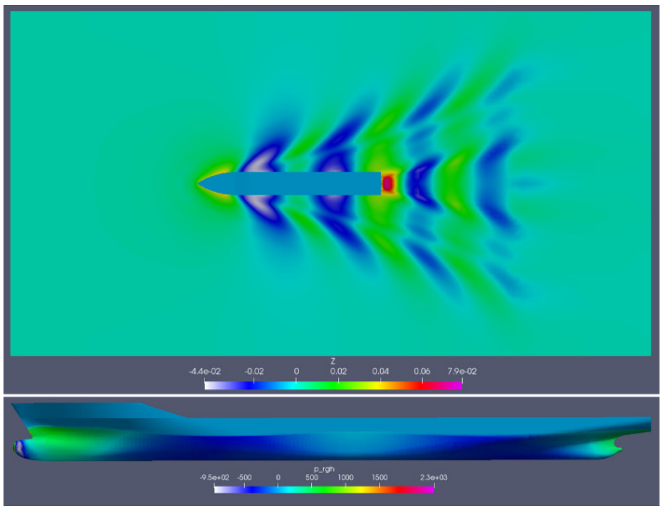

Z
-4.4e-02 -0.02 0 0.02 0.04 0.06 7.9e-02
p_rgh
-9.5e+02 -500 0 500 1000 1500 2.3e+03

Fig. 20. Calculation results of the KCS resistance with the generated grid.

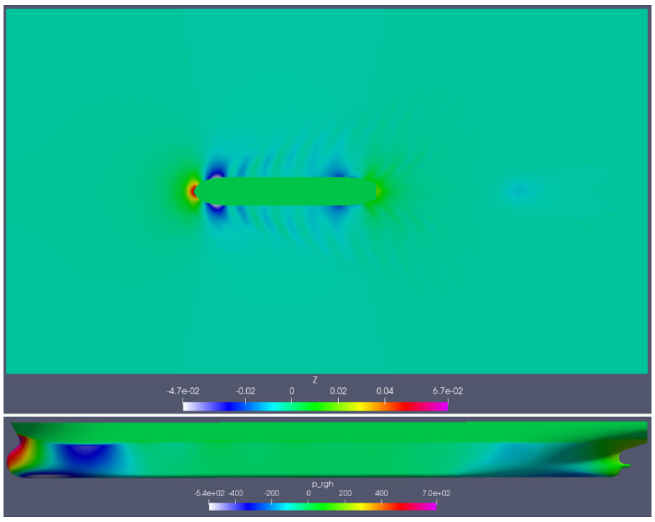

Z
-4.7e-02 -0.02 0 0.02 0.04 6.7e-02
p_rgh
-5.4e+02 -400 -200 0 200 400 7.0e+02

Fig. 21. Calculation results of the JBC resistance with the generated grid.

The thickness of the boundary layer grid used in the JBC case was thicker than that in the KCS, due to the slow ship speed.

The wave height contours and pressure distribution on the hull surfaces of KCS and JBC are shown in Figs. 20 and 21, respectively. The results are qualitatively reasonable, and the calculated resistances of KCS and JBC (85.33 N and 36.95 N, respectively) are 3.6% and 2.1% larger than experimental data, respectively. The present calculations and corresponding experiments were conducted with bare hulls without a rudder. The detailed experimental data of KCS and JBC were explained by Kim et al., (2011) and https://t2015.nmri.go.jp, respectively. The trim and sinkage considering the KCS were 0.154 deg. and 12.26 mm, which were 8.88% and 12.05% smaller than experimental data, respectively. Meanwhile, the calculated trim value of JBC was 0.110 deg, which was 6.66% larger than experimental data, and the calculated sinkage was 5.904 mm, which was 1.93% smaller than experimental data.

The developed algorithms of grid generation based on the cut-cell method were also applicable to the resistance calculation.

VIII. CONCLUSIONS

This paper presented a method to automatically generate a fully polyhedral grid to calculate the ship resistance. The method precisely reconstructs the hull surface, considering sharp edges and apexes, by defining the feature intersection point before the cell cut. The anisotropic grid refinement and curvature-based refinement level in the method were able to effectively and efficiently arrange the grid, using a relatively small number of cells. The quality of the boundary layer grid was also acceptable.

The developed algorithms of the grid generation based on the cut-cell method were validated by using generated grids for simulations of the KCS and JBC ships. The resistance results calculated with the grids were compared with the corresponding experimental data. The errors of KCS and JBC were approximately 3.6% and 2.1%, respectively. The results show that the grid quality of the meshes generated by this method is good enough; therefore, it can be applied to practical problems.

In the future, the present grid generation algorithm will be developed for POW and propulsion, including various appendages, such as strut, shaft, duct, and fin.

ACKNOWLEDGMENTS

This work was supported by the Industrial Fundamental-Technology Development Program (10060329, Development of a numerical simulator for the environmental load on a ship and offshore polar region), funded by the Ministry of Trade, Industry & Energy (MI, Korea) and Kunsan National University.

REFERENCES

Bowyer, A. (1981). Computing dirichlet tesselations, The Computer Journal 24(2), 162-166.

Capizzano, F. (2018). Automatic generation of locally refined Cartesian meshes: data management and algorithms. Numerical Methods in Engineering 113, 789-813.

Chiu, E.K., Q. Wang and A. Jameson (2011). A conservative meshless scheme: general order formulation and application to Euler Equations. 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.

Csakany, P. and A.M. Wallance (2000). Computation of local differential parameters on irregular meshes. In Proc. of IMA Conference on Mathematics of Surfaces, 19-33.

Delanaye, M., M. Aftosmis, M. Berger, Y. Liu and T. Pulliam (1999). Automatic Hybrid-Cartesian Grid generation for high-Reynolds number flow around complex geometries. AIAA paper 99-0777.

Fortune, S. (1987). A sweepline algorithm for voronoi diagrams. Algorithica 2, 153-174.

Fujimoto, K., K. Fujii and Z.J. Wang (2009). Improvements in the reliability and efficiency of body-fitted Cartesian Grid Method. AIAA 2009-1173.

Hafez, M. and K. Oshima (1998). Solution of the euler equations on solution-adaptive cartesian grids. Computational Fluid Dynamics Review 1998.

Kim, J., I.R. Park, K.S. Kim and Y.C. Kim (2011). Development of a numerical method for the evaluation of ship resistance and self-propulsion performances. J. the Society of Naval Architects of Korea 48(2), 147-157.

Kortelainen, J. (2009). Meshing Tools for Open Source CFD – A Practical Point of View. Research Report VTT-R-02440-09.

Liu, Y., A.A. Saputra, J. Wang, T. Francis and C. Song (2017). Automatic polyhedral mesh generation and scaled boundary finite element analysis of STL models. Computer methods in applied mechanics and engineering 313, 106-132.

Monaghan, J.J. (1992). Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics 30, 543-574.

Powell, K.G. (1998). Solution of the Euler on Solution-Adaptive Cartesian Grids. VKI Lectures Series.

Sarrate, J. and S. Matthew (2014). Multiblock structured mesh generation for turbomachinery flows. Proceedings of the 22nd International Meshing Roundtable.

Schlichting, H and J. Kestin (1979). Boundary Layer Theory. McGraw-Hill.

Van, S.H., W.J. Kim, D.H. Kim, G.T. Yim, C.J. Lee and J.Y. Eom (1997). Measurement of flows around a 3600TEU container ship model. Proceedings of the Annual Autumn Meeting, SNAK, Seoul, 300-304.

Zhang, Y. and Y. Jia (2018). 2D automatic body-fitted structured mesh generation using advancing extraction method. Journal of Computational Physics 353, 316-335.