Proceedings of the ASME 2010 3rd Joint US-European Fluids Engineering Summer Meeting and 8th International Conference on Nanochannels, Microchannels, and Minichannels
FEDSM-ICNMM2010
August 1-5, 2010, Montreal, Canada

FEDSM-ICNMM2010-30625

DEVELOPMENT OF RAPID SIMULATION METHOD FOR AUTOMOTIVE AERODYNAMICS

Kei Akasaka **Kenji Ono**

Functionality Simulation and Information Team, VCAD System Research Program, Riken
2-1, Hirosawa, Wako-shi, Saitama, Japan

ABSTRACT

Computational fluid dynamics (CFD) is now widely used as an essential tool in the development of automotive aerodynamics. However, the time required for repairing non-watertight geometries has recently become a serious problem in current CFD processes. Therefore, we developed an efficient simulation method that allows the flow around a non-watertight geometry to be computed on a Cartesian grid. This method can substantially reduce the turnaround time and effort required for CFD processes, because the repair work can be eliminated. The proposed method adopts an embedded boundary condition technique to capture arbitrary shapes more accurately on the background Cartesian grid. In addition, a local mesh refinement technique enables higher computational efficiency to be realized, and large-eddy simulation (LES) is used to reproduce high-Reynolds-number turbulent flow. Preliminary tests were performed on an engine bay configuration that had non-watertight geometries and a 1/5-scale model of an automobile configuration. As a result, the proposed method was confirmed to enable rapid grid generation and flow simulation around non-watertight geometries. Moreover, the computed results showed good agreement with experimental data.

INTRODUCTION

Background

At present, computational fluid dynamics (CFD) is widely used as an essential tool in the development of industrial products. However, the time required for repairing non-watertight geometries has recently become a serious problem in current CFD processes. In the case of industrial products such as automobiles and electronic devices, the geometry is composed of several million polygons. In particular, as shown in Figure 1, a geometry including incomplete polygons is called a non-watertight geometry. Since most general CFD software requires a watertight geometry, all incomplete polygons must be repaired before grid generation, as shown in Figure 2. With existing technology, repairing polygons automatically remains difficult, in spite of intensive research [1]. The repair task is time-consuming and can require several days to weeks when the geometry includes many incomplete polygons. This task often accounts for the greater part of the turnaround time of the simulation. If an efficient CFD approach for avoiding the repair of polygons can be developed, the turnaround time is expected to be reduced substantially.

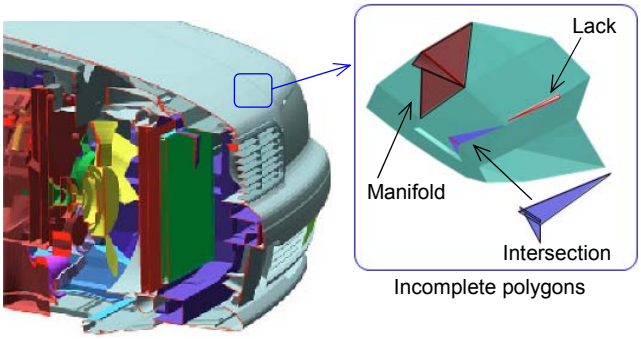


Figure 1: Example of non-watertight geometry including incomplete polygons

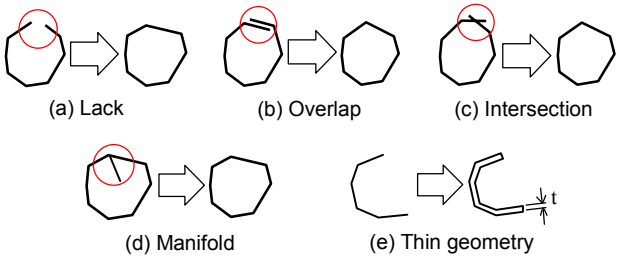


Figure 2: Repair of non-watertight geometries

Here, we propose a CFD approach that tolerates non-watertight geometry (Figure 3). Since the repair of polygons is not required in this approach, the turnaround time can be shortened considerably. Although the flows around the incomplete polygons might be slightly irrelevant (see Figure 3), the proposed method is expected to be useful when an approximate flow (not detailed) around an object is needed without delay. This approach will also reduce the burden of repair work because only incomplete polygons that interfere with the design study need to be repaired.

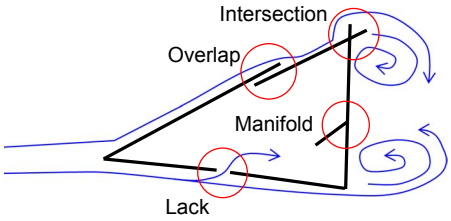


Figure 3: Flow around a non-watertight geometry

Conventional approach and proposed method

The voxel method [2,3] is a conventional approach to the flow simulation around a non-watertight geometry and is among the Cartesian grid methods. Figure 4 shows examples of shape approximation of non-watertight geometries by the voxel method. The voxel method allows uniform grids to be generated automatically in a short time. This feature is highly useful in actual product design. However, in the voxel method, the shape is approximated by cubic elements. In some cases, this shape approximation might be insufficient in terms of accuracy. Therefore the cut-cell technique [4] and the immersed boundary method [5] have been proposed to improve the accuracy of shape approximation on a Cartesian grid.

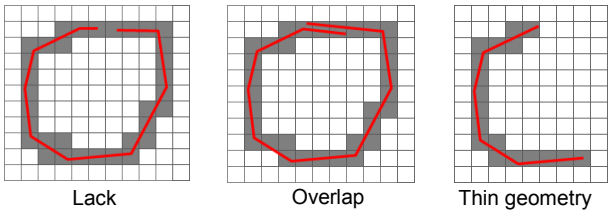


Figure 4: Shape approximation of non-watertight geometries by the voxel method

To address these issues, we also developed a practical simulation method to improve the accuracy of shape approximation by the Cartesian grid method; furthermore, our approach can be applied to non-watertight geometries. In the proposed method, at the cell nearest to a geometric boundary, the governing equation of the fluid is discretized using the distance d_i. As shown in Figure 5, d_i represents the axial distance (d_1, d_2) from a cell center to the geometric boundary. Since the distance information is taken into account in the discretization of the governing equation, the accuracy of shape approximation by this approach is better than that by the voxel method. Moreover, by using the computational technique [6] of the ray tracing in computer graphics, the distance can be calculated with ease. As shown in Figure 5, the distance is also acquired from non-watertight geometries.

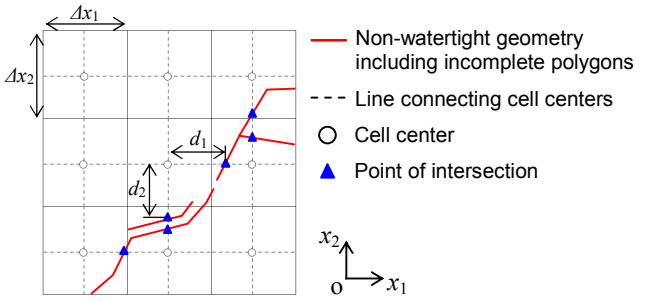


Figure 5: Definition of distance from cell center to geometric boundary in two dimensions

NOMENCLATURE

Variables, Constants

x_i	Cartesian coordinates
d_i	distance from cell center to geometry boundary
u_i	velocity components on the cell center
u_i^*	pseudo velocity components at the cell center
U_i	velocity components on the cell face
p	pressure
ρ	density
t	time
n	time step
ν	kinematic viscosity
Re	Reynolds number
τ_{ij}	Sub grid scale (SGS) Reynolds stress

GOVERNING EQUATIONS AND NUMERICAL METHOD

Governing equations

The governing equations of the incompressible fluid used in this study are the spatially filtered Navier-Stokes equation and a continuity equation:

$$\frac{\partial \tilde{u}_i}{\partial t} + \frac{\partial \tilde{U}_j \tilde{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial \tilde{u}_i}{\partial x_j}\right) - \frac{\partial \tau_{ij}}{\partial x_j}, \quad (1)$$

$$\frac{\partial \tilde{U}_j}{\partial x_j} = 0, \quad (2)$$

where the tilde symbol (~) denotes the spatial filtering operation. Variables are located as shown in Figure 6.

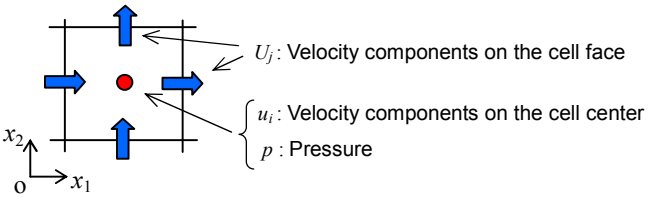


Figure 6: Location of variables on collocated grid in two dimensions

Numerical method

Both convective and viscous terms of the Navier-Stokes equation are discretized using a central difference scheme with second-order accuracy. In some cases, an upwind scheme with second-order accuracy and a central difference scheme with fourth-order accuracy are also used for the convective term. For the time integration, the Adams-Bashforth method with second-order accuracy is adopted. The fractional step algorithm [7] is used for the pressure-velocity coupling. The SGS Reynolds stress must be modeled; in this study, the Smagorinsky model [8] is used. In addition, the Cartesian grid is adopted and combined with the local mesh refinement technique using a nested grid method [9]. This technique allows efficient deployment of finer grids close to the geometry in order to improve computational accuracy.

Discretization of Navier-Stokes equation

The discretization method is described for the one-dimensional case for ease of presentation. The viscous term is discretized by using a central difference scheme with second-order accuracy:

$$\nu \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)\bigg|_i = \nu\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}\right), \tag{3}$$

where the index i denotes position on the x-axis, and the tilde symbol (~) is omitted. In the proposed method, when the geometric boundary is located in the right side of cell i as shown in Figure 7, the linearly extrapolated velocity $\hat{u}_{i+1}$ is used in place of u_{i+1} of equation (3):

$$\hat{u}_{i+1} = \left(1 - \frac{3\Delta x}{\Delta x + 2d}\right)\overline{u}_{i-1/2}. \tag{4}$$

Here, hat (^) and bar (¯) symbols denote extrapolating and interpolating operations, respectively; Δx is the grid size, and d is the distance shown in Figure 5. This extrapolated velocity $\hat{u}_{i+1}$ is calculated under the following two assumptions [10] : firstly, the gradient of velocity in the vicinity of the geometric boundary is linear; secondly, the velocity at the geometric boundary satisfies the no-slip condition. Meanwhile, $\overline{u}_{i-1/2}$ denotes the linearly interpolated velocity on a cell face position calculated from u_i and u_{i-1}.

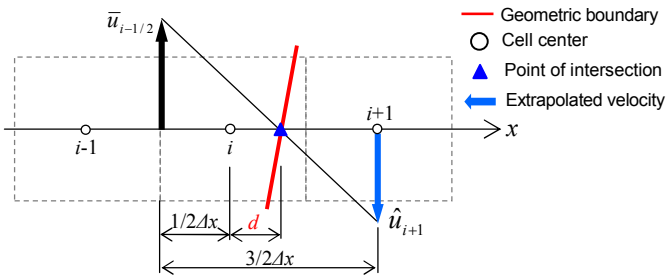


Figure 7: Schematic of velocity extrapolation assuming a linear profile in one dimension

The convective term is discretized by using a central difference scheme with second-order accuracy:

$$\left.\frac{\partial U\,u}{\partial x}\right|_i = \frac{f_{i+1/2} - f_{i-1/2}}{\Delta x},$$
$$f_{i-1/2} = \frac{1}{2}U_{i-1/2}(u_i + u_{i-1}), \tag{5}$$
$$f_{i+1/2} = \frac{1}{2}U_{i+1/2}(u_{i+1} + u_i).$$

Similar to the viscous term, when the geometric boundary is located in the right side of cell i as shown in Figure 7, the extrapolated velocity $\hat{u}_{i+1}$ is used in place of u_{i+1} of equation (5). The convective and viscous terms of the other axes (x_2 and x_3) are also discretized in the same manner.

It should be noted that the extrapolated velocity can be calculated by other methods. For example, it can be also determined by using u_i as shown in equation (6) [11]:

$$\hat{u}_{i+1} = \left(1 - \frac{\Delta x}{d}\right)u_i. \tag{6}$$

However, $1/d$ is included in equation (6); the $1/d$ term might result in computational instability if the distance d is extremely small. On the other hand, equation (4) is able to maintain computational stability when the distance d is almost zero.

Discretization of Poisson equation

The Poisson equation of pressure, which is given in equation (7), is discretized by using a central difference scheme with second-order accuracy. The discretized Poisson equation is solved iteratively by the successive over-relaxation (SOR) method:

$$\frac{\partial}{\partial x_j}\left(\frac{\partial p^{n+1}}{\partial x_j}\right) = \frac{1}{\Delta t}\frac{\partial u_j^*}{\partial x_j}. \tag{7}$$

In this study, equation (8) is taken as the boundary condition of pressure at the geometric boundary, except for extremely small Reynolds numbers:

$$\frac{\partial p}{\partial n} = 0. \tag{8}$$

Equation (8) shows that the pressure gradient in the normal direction $\boldsymbol{n}$ is zero at the geometric boundary. In this study, the difference in the pressure gradients between the normal and axial directions is assumed to be very small since the grid resolution is extremely fine. Under this assumption, the pressure gradient in the normal direction is approximated by the pressure gradient in the axial direction, as shown in Figure 8 and equation (9):

$$\frac{\partial p}{\partial n} \approx \frac{\partial p}{\partial x_1} = 0 \quad \text{(at point of intersection A in Figure 8)},$$
$$\frac{\partial p}{\partial n} \approx \frac{\partial p}{\partial x_2} = 0 \quad \text{(at point of intersection B in Figure 8)}. \tag{9}$$

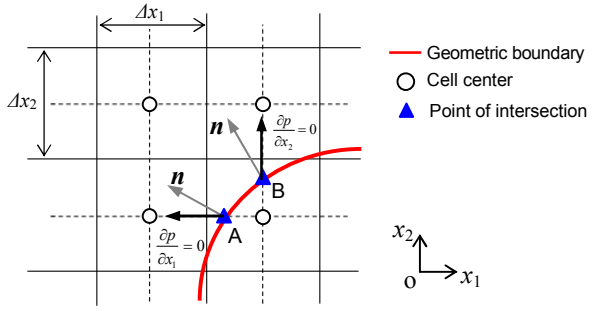


Figure 8: Schematic of pressure boundary condition at geometric boundary in two dimensions

Efficiently solving the Poisson equation is critical, taking into account the boundary condition given in equation (9). In this study, the left-hand side of the Poisson equation is discretized as shown in equation (10):

$$\frac{\partial^2 p^{n+1}}{\partial x_1^{\,2}} + \frac{\partial^2 p^{n+1}}{\partial x_2^{\,2}} = \frac{1}{\Delta x_1}\left(\psi_{i+1/2,j}\left.\frac{\partial p^{n+1}}{\partial x_1}\right|_{i+1/2} - \psi_{i-1/2,j}\left.\frac{\partial p^{n+1}}{\partial x_1}\right|_{i-1/2}\right), \\ + \frac{1}{\Delta x_2}\left(\psi_{i,j+1/2}\left.\frac{\partial p^{n+1}}{\partial x_2}\right|_{j+1/2} - \psi_{i,j-1/2}\left.\frac{\partial p^{n+1}}{\partial x_2}\right|_{j-1/2}\right) \tag{10}$$

where

$$\psi_{i\pm1/2,j\pm1/2} = \begin{cases} 0 : \text{Geometric boundary} \\ 1 : \text{Fluid.} \end{cases} \tag{11}$$

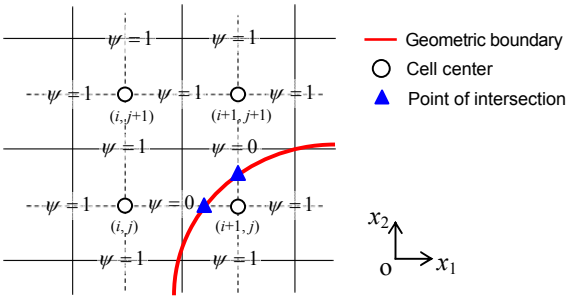


Figure 9: Function for installing pressure boundary condition in two dimensions

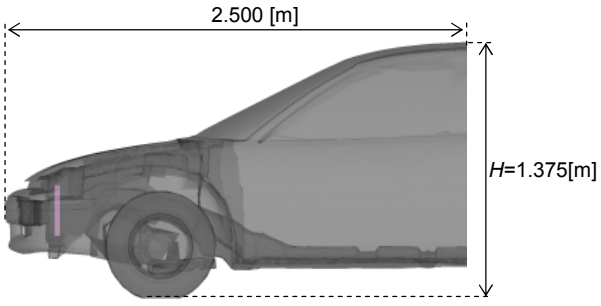
Here, i and j represent positions in the x_1 and x_2 directions, respectively, and ψ is a function denoting the existence of the geometric boundary. The function ψ is defined on the cell face as shown in Figure 9 and equation (11). The function can be calculated at the same time as the distance d_i. Discretization using this function allows efficient implementation [12] of the boundary condition when programming.

RESULTS AND DISCUSSION

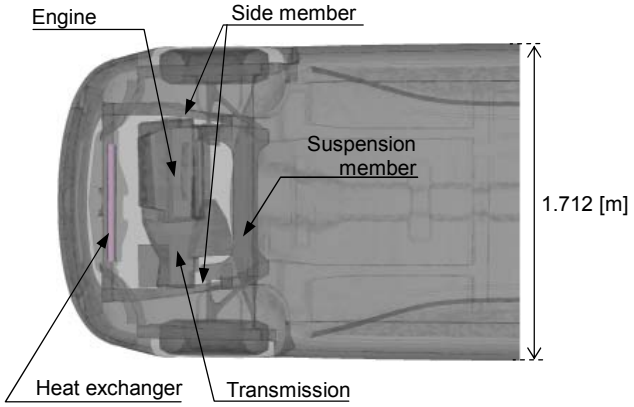
As validation and verification of the proposed method, two cases were calculated. First, the flow around the non-watertight geometry of an engine bay configuration was calculated as verification. Second, as validation, the flow around a 1/5-scale model of an automobile configuration was calculated for comparison with experimental data.

Case 1: Flow simulation for engine bay model including incomplete polygons

The engine bay model of Case 1 is shown in Figure 10. This model was used for benchmark testing of commercial grid generation software by the Society of Automotive Engineers of Japan (JSAE) in 2003 [13]. The model has a non-watertight geometry that is composed of 0.27 million polygons including many incomplete polygons, such as those shown in Figure 1.



(a) Side view

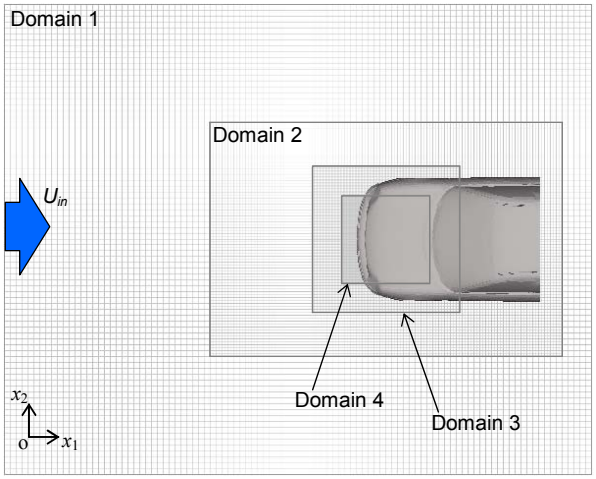


(b) Top view

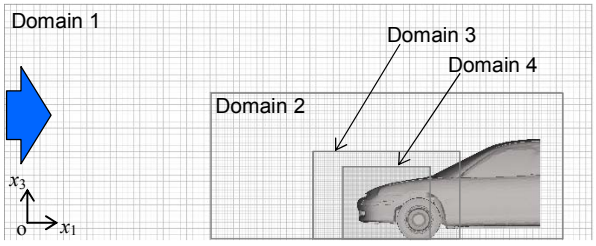
Figure 10: Geometry of automotive engine bay [13]

The computational domains and grid of Case 1 are shown in Figure 11 and Table 1. The nested grid technique was adopted for the computational grid. This nested grid was composed of four domains. The total number of grids was approximately 3 million. At the inlet of Domain 1, a uniform velocity condition was imposed, while at the exit, free-outlet condition was adopted. The no-slip condition was imposed on the ground,

ceiling and sidewall of Domain 1. In this case, the Reynolds number was 1.8×10^6, based on the automobile height H and the inlet velocity U_{in}. For the convective term, an upwind scheme with second-order accuracy was adopted.



(a) Top view



(b) Side view

Figure 11: Computational domains and grid for Case 1

Table 1: Grid size and dimensions of each domain for Case 1

	Grid size (Δx_1, Δx_2, Δx_3)	Dimension (Length × Width × Height)
Domain 1	0.0800H	8.0H × 6.4H × 3.2H
Domain 2	0.0400H	4.8H × 3.2H × 2.0H
Domain 3	0.0200H	2.0H × 2.0H × 1.2H
Domain 4	0.0100H	1.2H × 1.2H × 1.0H

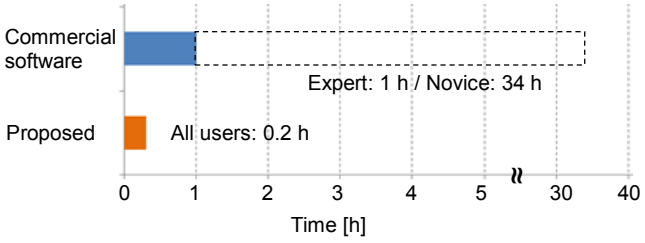
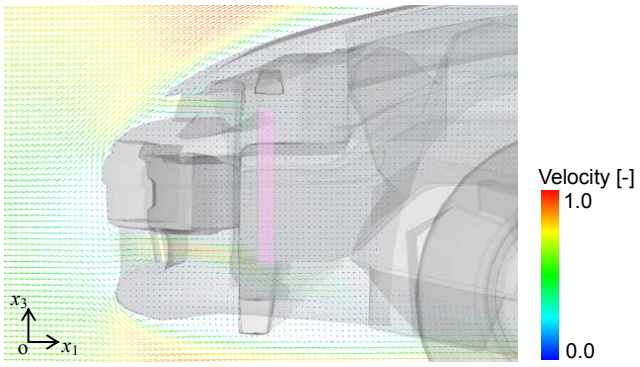


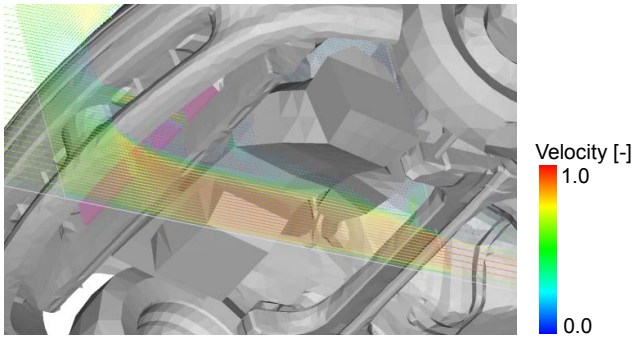
Figure 12: Comparison of grid generation time between proposed method and commercial software

The time required for grid generation was only 10 minutes. This time included not only grid generation but also importation of the geometry and adjustment of the position of the computational domain. A comparison of the grid generation time is shown in Figure 12. In this study, the proposed method was compared with the commercial software that was found to be fastest among the four commercial software packages evaluated in the JSAE benchmark test. In these commercial ones, the unstructured polyhedral grid was used and the number of grids was approximately 3 million. Figure 12 shows that the proposed method can generate the grid much faster than the commercial software. Since the commercial software must repair all incomplete polygons, this repair time can be considered as a major reason for its longer grid generation time. Moreover, in the JSAE benchmark test, it was found that the grid generation time of the commercial software varied depending on the skill and experience of the user. Meanwhile, in the case of the proposed method, it was found that all users could generate the grid in the same time and quality regardless of expertise, which represents a highly useful feature for CFD users.

Figure 13 shows velocity vectors inside the engine bay around the center section of the model. This figure indicates that the proposed method enables grid generation and stable simulation even if the non-watertight geometry has incomplete polygons including lack, overlap, manifold, and intersection.



(a) Side view (transparent view)

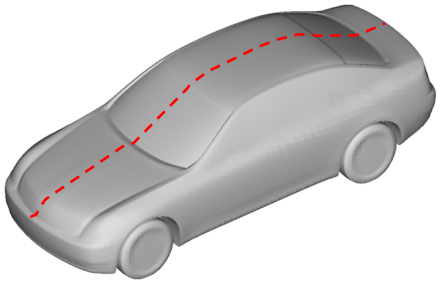


(b) Bottom view

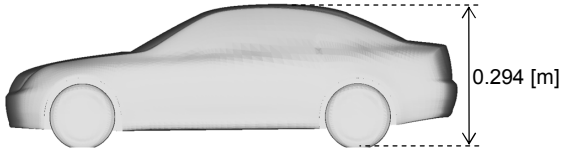
Figure 13: Distribution of velocity vectors in automotive engine bay

Case 2: Flow simulation around 1/5-scale model of automobile configuration

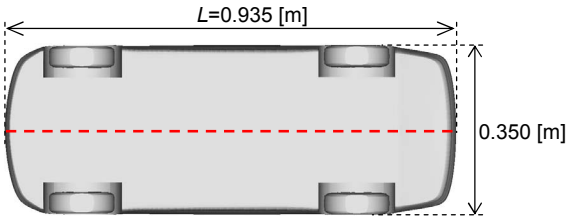
The 1/5-scale model of Case 2 is shown in Figure 14. This model is composed of 83 thousand polygons. In this study, the pressure distribution along the red dashed line was compared between the present results and experimental data.



(a) Quarter view

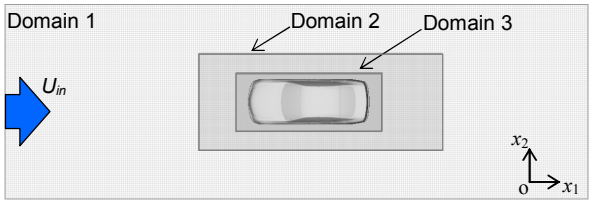


(b) Side view

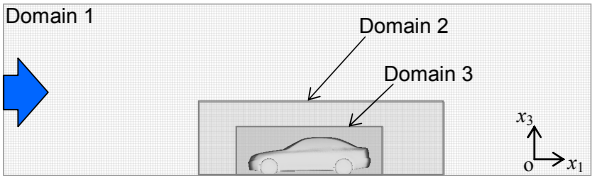


(c) Bottom view

Figure 14: Geometry of 1/5-scale model of automobile



(a) Top view



(b) Side view

Figure 15: Computational domains and grid for Case 2

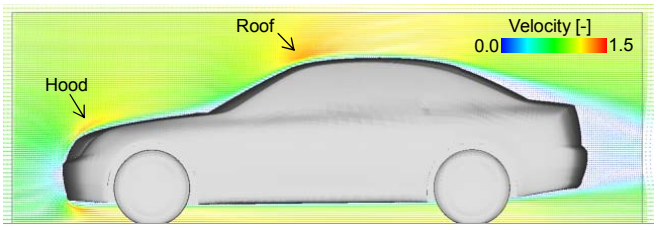
Table 2: Grid size and dimensions of each domain for Case 2

	Grid size (Δx_1, Δx_2, Δx_3)	Dimensions (Length × Width × Height)
Domain 1	0.016*L*	4.80*L* × 1.60*L* × 1.28*L*
Domain 2	0.008*L*	2.00*L* × 0.80*L* × 0.64*L*
Domain 3	0.004*L*	1.20*L* × 0.48*L* × 0.40*L*

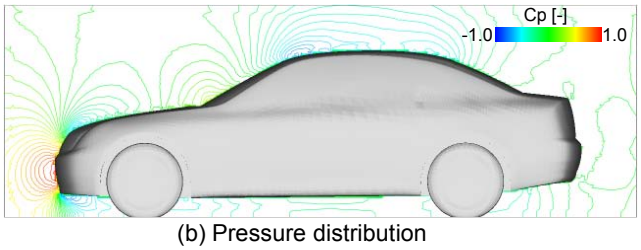
The computational domains and grid of Case 2 are shown in Figure 15 and Table 2. The nested grid was composed of three computational domains. The total number of grids was approximately 8 million. Except for the ground, the boundary conditions of the inlet, exit, ceiling, and side wall were the same as in Case 1. For the ground, a moving ground condition was adopted. In this case, the Reynolds number was 1.5×10^6, based on the automobile length L and the inlet velocity U_{in}. For the convective term, a central difference scheme with fourth-order accuracy was adopted. The Smagorinsky model was used for the turbulence model. The coefficient of the Smagorinsky model was taken as Cs=0.15 in this study.

Figure 16 shows velocity vectors and the pressure distribution at the center section of the automobile body (x_2=0). Figure 16(a) shows that around curves and corners on the hood, the roof, and underbody, the velocity is higher. Meanwhile, Figure 16(b) shows that the pressure around these curves and corners is extremely low.

The comparison of the pressure coefficient (Cp) on the automobile body between the present results and experimental data is shown in Figure 17. The present results are in reasonably good agreement with experimental data. Around the curves and corners on the upperbody in particular, the negative pressure was predicted quantitatively. These results indicate that the proposed method enables the approximate aerodynamic characteristics of an automobile to be predicted satisfactorily.

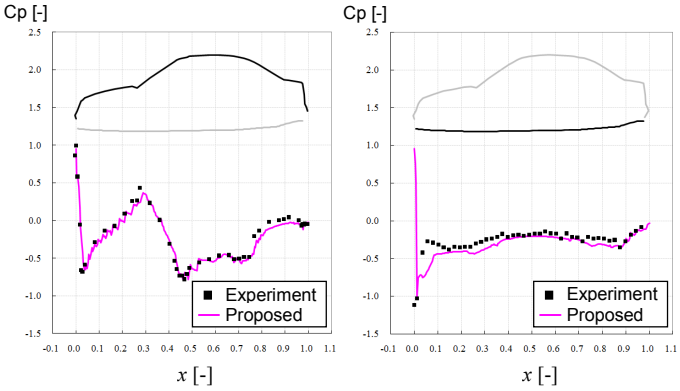


(a) Distribution of velocity vectors



(b) Pressure distribution

Figure 16: Distribution of velocity vectors and pressure around 1/5-scale model



(a) Upperbody surface (b) Underbody surface

Figure 17: Distribution of pressure coeffiecient on surface of 1/5-scale model

CONCLUSIONS

1. The proposed method enables rapid grid generation and stable simulation of non-watertight objects. This method can contribute greatly to the reduction of time and effort required for repairing incomplete polygons, which is the most time-consuming task in CFD processes.
2. By using this method, all users can generate the grid in the same time and quality regardless of expertise, which represents a highly useful feature for CFD users.
3. The pressure distribution predicted by the proposed method showed good agreement with experimental data. This approach was confirmed to enable rapid simulation of the approximate aerodynamic characteristics of an automobile.

ACKNOWLEDGMENTS

The calculations were performed on the RIKEN Integrated Cluster of Clusters (RICC) facility.

REFERENCES

[1] Z. J. Wang and K. Srinivasan, An Adaptive Cartesian Grid Generation Method for Dirty Geometry, *International Journal for Numerical Methods in Fluids*, Vol.39, pp. 703-717 (2002).

[2] K. Tsuboi, K. Miyakoshi and K. Kuwahara, Incompressible Flow Simulation of Complicated Boundary Problems with Rectangular Grid System, *Theoretical and Applied Mech.*, Vol. 40, pp.297-309 (1991).

[3] K. Ono, K. Fujitani and H. Fujita, Applications of CFD Using Voxel Modeling to Vehicle Development, *In Proceedings of the 3rd ASME/JSME Joint Fluids Engineering Conference*, FEDSM99-7323 (1999).

[4] J. J. Quirk, An Alternative to Unstructured Grids for Computing Gas Dynamic Flows around Arbitrarily Complex Two-Dimensional Bodies, Computers and Fluids, vol.23, pp.125–142 (1994).

[5] E. A. Fadlun, R. Verizicco, P. Orlandi and J. Mohd-Yusof, Combined Immersed -Boundary Finite -Difference Methods for Three-Dimensional Complex Flow Simulations, Journal of Computational Physics, Vol.161, pp.35-60 (2000).

[6] T. Möller and B. Trumbore, Fast, Minimum Storage Ray-Triangle Intersection, *Journal of Graphics Tools*, No.2, pp.21-28 (1997).

[7] A. J. Chorin, Numerical Solution of the Navier Stokes Equations, *Mathematics of Computation*, No.22, pp.745-762 (1968).

[8] J. Smagorinsky, General circulation experiments with the primitive equations I. the basic experiment, *Monthly Weather Review*, Vol.91-3, pp.99-165 (1963).

[9] K. Ono, K. Akabane, H. Shiozawa and K. Fujitani, Prediction of cooling flow rate through the front grille using flow analysis with a multi-level mesh, *Soul 2000 FISITA world automotive congress*, Paper No. F2000H201 (2000).

[10] K.Akasaka and K. Ono, Simulation of Incompressible Viscous Flow on Cartesian Grid for Arbitrary Geometries Composed of Non-Watertight Polygon Elements, *Transactions of the Japan Society of Mechanical Engineers,* Series B, vol.76, No.764, pp.536-545 (2010).

[11] O. Ichikawa and K. Fujii, Study of Boundary Conditions on Solving Flow Field around Arbitrary Shape by Using Cartesian Grid, *Proceeding of 13th CFD symposium*, F03-3, p.248 (1999).

[12] K. Akasaka and K. Ono, An Implementation of Boundary Condition of Incompressible Flow Solver for Complex Geometries on Voxel Method, *Transactions of JSCES*, Vol. 2006, 20060024 (2006).

[13] K. Ono, K. Ikeda, S. Nakamura, N. Hamamoto, R. Isomura, T. Hasegawa, and H. Toya, Benchmark test of CFD software for a simulation of a flow inside an automotive engine bay, *Proceeding of JSAE symposium,* No.11-03, pp.38-69 (2003).