Advances in Aerodynamics

RESEARCH Open Access

Moving computational multi-domain method for modelling the flow interaction of multiple moving objects

Momoha Nishimura[1*], Masashi Yamakawa[1], Shinichi Asao[2], Seiichi Takeuchi[2] and Mehdi Badri Ghomizad[1]

*Correspondence:
d7821007@edu.kit.ac.jp
[1]Department of Mechanical and System Engineering, Kyoto Institute of Technology, Kyoto, Japan
Full list of author information is available at the end of the article

Abstract

This study proposes a method for modelling the flow interaction of multiple moving objects where the flow field variables are communicated between multiple separate moving computational domains. Instead of using the conventional approach with a single fixed computational domain covering the whole flow field, this method advances the moving computational domain (MCD) method in which the computational domain itself moves in line with the motions of an object inside. The computational domains created around each object move independently, and the flow fields of each domain interact where the flows cross. This eliminates the spatial restriction for simulating multiple moving objects. Firstly, a shock tube test verifies that the overset implementation and grid movement do not adversely affect the results and that there is communication between the grids. A second test case is conducted in which two spheres are crossed, and the forces exerted on one object due to the other's crossing at a short distance are calculated. The results verify the reliability of this method and show that it is applicable to the flow interaction of multiple moving objects.

Keywords: Crossing, Moving boundary problems, Compressible flow, Moving mesh method

1 Introduction

The need for computational fluid dynamics (CFD) has increased along with technological developments and the expansion of the transportation industry. In particular, moving boundary problems are some of the most important industrial problems. Nowadays, CFD is required for modelling the physics of real-life transport vehicles. CFD is utilised not only for examining the vehicles' individual components but also for fully simulating moving vehicles that travel long distances with complex geometry and configuration, as is the case for the flight simulations of aircraft [1]. Moreover, since drones and flying cars have been a focus of attention, the interaction of flows between multiple moving objects has become of interest. Considering the flow on such a global scale is more challenging due to the limitation of the computational cost. The implication is that CFD requires reasonable and practical methods to tackle problems in modelling multiple objects moving in a large space.

Various numerical methods have been developed for simulating moving boundary problems. One category is non-boundary fitted mesh methods, where the whole computational domain is overlayed by a simple Cartesian mesh without considering the target objects inside the field. Instead, the governing equations, discretised in the Cartesian mesh, are adjusted to consider the presence of the internal boundary associated with those objects by using suitable methods, for example, the immersed boundary method [2, 3]. By virtue of employing a simple mesh, they do not require intricate mesh generation techniques and are able to utilise highly efficient solvers, such as multigrid or FFT-based fast Poisson solvers [4, 5]. On the other hand, non-boundary fitted mesh methods normally have to struggle with the thin boundary layers as well as the external forces exerted on the objects due to the nature of the background Cartesian meshes [6]. In addition, these methods are relatively ineffective in large distance motion problems as their computational domain should represent the whole physical domain, which requires great computational cost [7].

Another approach is the Chimera method [8, 9], which is widely used to represent motions in simulations. This approach addresses moving boundary problems by interpolating the flow field variables between overlapping grids at each time step. This method has been utilised in applications such as the separation of a supersonic aeroplane and rocket booster [9] and the manoeuvring of helicopters in a shipborne environment [10]. However, the Chimera method also constrains the size of the computational domain: the overlapping strategy requires a background grid for the interpolation, but the background cannot be set as an infinitely large grid due to the computational cost. Consequently, it is also difficult to simulate objects moving in a large space (Fig. 1a).

Furthermore, Yamakawa et al. proposed a remeshing technique for unstructured grids, which performs accurate computation without interpolating physical quantities [11]. They simulated the flow interaction of multiple moving objects by placing objects in a single unstructured domain and moving each of them (Fig. 1b). Typically, when objects move in a computational grid, cells that are initially almost equilateral are distorted along with the motion and deformation of the objects, which may eventually lead to calculation failure. Thus, the crossing of two objects was realised by eliminating the distorted cells and splitting the stretched cells in the grid. However, although this technique was shown to be applicable to simulations with a drastic deformation of unstructured cells, a large domain covering the entire flow field was still necessary. Hence the applications of this technique are restricted to objects moving only short distances. That is, for simulation of the whole trajectory of moving objects, the spatial limit is usually the bottleneck.

To address this difficulty, we previously proposed a unique approach: the moving computational domain (MCD) method [12], which is based on the moving-grid finite volume (MGFV) method [13]. The MGFV method is an approach for solving moving boundary problems and automatically meets the geometric conservation law (GCL) condition [14] because it adopts the space and time unified four-dimensional (4D) control volume for the discretisation. Since the MGFV method is one of the moving mesh methods, objects do not necessarily move in a fixed domain. If the computational domain itself moves in line with the motions of an object inside, the spatial restriction for modelling moving objects can be removed; this technique is the MCD method (Fig. 2). It has been able to simulate the flow around a single object moving in a large space, such as the flight simulation of a tilt-rotor aircraft [15] and a swimmer performing the dolphin kick [16].

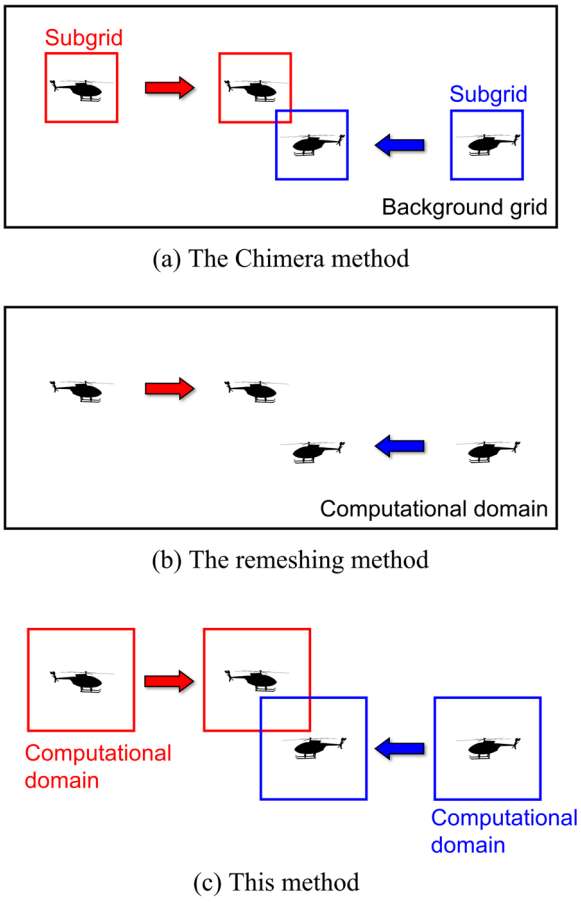


Fig. 1 The concept of (**a**) the Chimera method, (**b**) the remeshing method, and (**c**) this method (illustrated in two-dimensions)

In this paper, we extend the MCD method to multiple computational domains. In the conventional MCD method with a single domain, it was difficult to gain information from outside the computational domain, thus the application was limited to a single object. We address this issue by introducing the mutual communication of independent domains. Physically speaking, in the context of multiple moving objects, the flow around each object does not affect the other objects when they are far away from each other. Accordingly, we essentially need large enough computational domains in the vicinity of

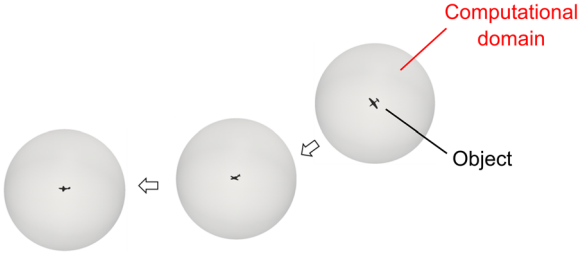


Fig. 2 Moving computational domain of the MCD method

individual objects that follow the objects' motion and capture the fluid flow phenomena there (Fig. 1c), rather than a fixed large domain covering the whole physical field. These domains only need to communicate when they are in close proximity and fluid flow interaction occurs. This approach represents an improved method for simulating multiple vehicles that travel long distances, from the beginning to the end of the trajectories. The approach is particularly suitable for the case of vehicles that are crossed or overtaken by other vehicles at a few points along the trajectories.

This paper includes five sections. Section 1 introduced the background and motivation. Section 2 explains the numerical scheme. Section 3 proposes the MCD method and the approach for allowing the communication of multiple domains. The validations are presented in Section 4, which includes a demonstration of crossing two objects as well as a discussion. Finally, Section 5 summarises the study's findings.

2 Numerical scheme

2.1 Governing equations

The three-dimensional (3D) Euler equations for compressible flow are applied to the governing equation:

$$\frac{\partial \boldsymbol{q}}{\partial t}+\frac{\partial \boldsymbol{E}}{\partial x}+\frac{\partial \boldsymbol{F}}{\partial y}+\frac{\partial \boldsymbol{G}}{\partial z}=\boldsymbol{0},$$
$$\boldsymbol{q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ e\end{bmatrix},\ \boldsymbol{E}=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ u(e+p)\end{bmatrix},\ \boldsymbol{F}=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho vw\\ v(e+p)\end{bmatrix},\ \boldsymbol{G}=\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ w(e+p)\end{bmatrix}, \tag{1}$$

where $\boldsymbol{q}$ is a vector of conserved variables, and $\boldsymbol{E}$, $\boldsymbol{F}$, and $\boldsymbol{G}$ represent the inviscid flux vectors. t, ρ, p, and e represent time, density, pressure, and the total energy per unit mass, respectively, and u, v, w represent the velocity components in the x, y, z coordinates. Equation (1) is closed with the following equation by considering the ideal gas law:

$$p=(\gamma-1)\left[e-\frac{1}{2}\rho\left(u^2+v^2+w^2\right)\right],$$

where γ is the ratio of specific heats. Note that Eq. (2) is used for nondimensionalisation in this study:

$$x=\frac{\hat{x}}{\hat{L}},\ y=\frac{\hat{y}}{\hat{L}},\ z=\frac{\hat{z}}{\hat{L}},\ t=\frac{\hat{t}}{\hat{L}/\hat{c}_\infty},\ \rho=\frac{\hat{\rho}}{\hat{\rho}_\infty},$$
$$e=\frac{\hat{e}}{\hat{\rho}_\infty\hat{c}_\infty^2},\ p=\frac{\hat{p}}{\hat{\rho}_\infty\hat{c}_\infty^2},\ u=\frac{\hat{u}}{\hat{c}_\infty},\ v=\frac{\hat{v}}{\hat{c}_\infty},\ w=\frac{\hat{w}}{\hat{c}_\infty}, \tag{2}$$

where "$\hat{}$" indicates values with dimensions, and $\hat{L}$, $\hat{c}_\infty$, and $\hat{\rho}_\infty$ denote the characteristic length, speed of sound, and characteristic density, respectively.

2.2 Flow solver

Equation (1) is integrated over the control volume $\widetilde{\Omega}$ as follows:

$$\int_{\widetilde{\Omega}} \widetilde{\nabla}\widetilde{\boldsymbol{F}}\, d\widetilde{\Omega} = 0$$
$$\widetilde{\nabla} = \left(\frac{\partial}{\partial t}, \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right), \quad \widetilde{\boldsymbol{F}} = (\boldsymbol{q}, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G})\,. \tag{3}$$

In order to solve the governing equations in a moving grid, Eq. (3) is discretised using the unstructured cell-centred approach of the MGFV method. This method uses a unified 4D time-space control volume, as illustrated in Fig. 3a, with a position vector $\widetilde{\boldsymbol{R}}$. As tetrahedra grids are used in this study, the discrete form of the governing equation is as follows:

$$\int_{\widetilde{\Omega}} \widetilde{\nabla}\widetilde{\boldsymbol{F}} d\widetilde{\Omega} = \int_{\partial\widetilde{\Omega}} \widetilde{\boldsymbol{F}} \cdot \tilde{\boldsymbol{n}}\, d\widetilde{S} = \sum_{l=1}^{6} \widetilde{\boldsymbol{F}}_l \cdot \tilde{\boldsymbol{n}}_l = 0,$$

where $\partial\widetilde{\Omega}$ is the 4D boundary surface of the control volume. As depicted in Fig. 3b, the boundary surfaces at $l = 1{-}4$ are equivalent to the 3D solids formed by the corresponding surfaces of a tetrahedron and the surfaces' displacements in the t dimension. For example, the boundary surface at $l = 1$ is equal to the solid composed by connecting the $\widetilde{\boldsymbol{R}}_1^N$-$\widetilde{\boldsymbol{R}}_2^N$-$\widetilde{\boldsymbol{R}}_3^N$ surface and the $\widetilde{\boldsymbol{R}}_1^{N+1}$-$\widetilde{\boldsymbol{R}}_2^{N+1}$-$\widetilde{\boldsymbol{R}}_3^{N+1}$ surface. In contrast, the boundary surfaces at $l = 5$ and $l = 6$ are equivalent to the tetrahedra at the N step and $N+1$ step, respectively. $\tilde{\boldsymbol{n}}$ represents the 4D outward normal vector $\tilde{\boldsymbol{n}} = \left[\tilde{n}_t, \tilde{n}_x, \tilde{n}_y, \tilde{n}_z\right]$. The length of $\tilde{\boldsymbol{n}}$ is equal to the volume of the corresponding solid. It should be stated that while $\tilde{\boldsymbol{n}}_1$ to $\tilde{\boldsymbol{n}}_4$ comprises $\left[\tilde{n}_t, \tilde{n}_x, \tilde{n}_y, \tilde{n}_z\right]$, $\tilde{n}_x, \tilde{n}_y, \tilde{n}_z = 0$ at $\tilde{\boldsymbol{n}}_5$ and $\tilde{\boldsymbol{n}}_6$, which means $\tilde{\boldsymbol{n}}_5$ and $\tilde{\boldsymbol{n}}_6$ only have an $\tilde{n}_t$ component. In addition, at the boundary surface $l = 1-4$, $\widetilde{\boldsymbol{F}}$ is evaluated by combining the values at the N step and $N+1$ step. On the other hand, $\widetilde{\boldsymbol{F}}_5$ is evaluated only by using the value at the N step since $l = 5$ represents the tetrahedron at the N step. Similarly, $\widetilde{\boldsymbol{F}}_6$ is evaluated only by using the value at the $N+1$ step. The details of the MGFV method can also be found in [13].

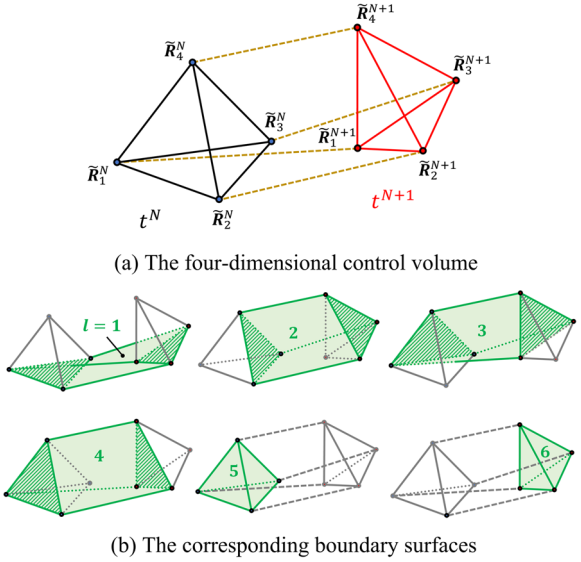


Fig. 3 (**a**) The 4D control volume and (**b**) the corresponding boundary surfaces in the MGFV method

From the procedure above, Eq. (4) can be obtained:

$$\begin{aligned} \boldsymbol{q}^{N+1}(\tilde{n}_t)_6 + \boldsymbol{q}^{N}(\tilde{n}_t)_5 + \sum_{l=1}^{4} \left\{ \boldsymbol{q}^{N+\frac{1}{2}}\, \tilde{n}_t + \boldsymbol{H}^{N+\frac{1}{2}} \right\}_l &= 0 \\ \boldsymbol{H} = \boldsymbol{E}\tilde{n}_x + \boldsymbol{F}\tilde{n}_y + \boldsymbol{G}\tilde{n}_z & \\ \boldsymbol{q}^{N+\frac{1}{2}} = \frac{1}{2}\left(\boldsymbol{q}^{N} + \boldsymbol{q}^{N+1}\right), \quad \boldsymbol{H}^{N+\frac{1}{2}} &= \frac{1}{2}\left(\boldsymbol{H}^{N} + \boldsymbol{H}^{N+1}\right), \end{aligned} \tag{4}$$

where the superscript N indicates the index of the time steps. The inviscid flux vector $\boldsymbol{H}_l$ is calculated by Roe's flux difference splitting [17]. To provide second-order accuracy, the MUSCL (Monotonic Upstream-centred Scheme for Conservation Laws) scheme is applied. The primitive variables of $\boldsymbol{q}$ are linearly reconstructed using the gradient computed with the Green-Gauss method and Hishida's limiter [18]. $\boldsymbol{q}\tilde{n}_t$ in Eq. (4) is estimated by the upwind scheme:

$$\boldsymbol{q}\tilde{n}_t = \frac{1}{2}\left[\boldsymbol{q}^{+}\tilde{n}_t + \boldsymbol{q}^{-}\tilde{n}_t - |\tilde{n}_t|\left(\boldsymbol{q}^{+} - \boldsymbol{q}^{-}\right)\right].$$

In the MGFV method, time stepping is represented by the following equation:

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\frac{1}{\widetilde{\Omega}}\left[\boldsymbol{q}^{N+1}(\tilde{n}_t)_6 + \boldsymbol{q}^{N}(\tilde{n}_t)_5 + \sum_{l=1}^{4}\left\{\boldsymbol{q}^{N+\frac{1}{2}}\, \tilde{n}_t + \boldsymbol{H}^{N+\frac{1}{2}}\right\}_l\right]. \tag{5}$$

For the 3D MGFV method, $\widetilde{\Omega}$ represents the 4D control volume, and the value can be approximated with the corresponding 3D sixth volume V_6:

$$\widetilde{\Omega} \approx \Delta t V_6.$$

Here, V_6 has the same value as the absolute value of the normal vector to the sixth volume $\tilde{\boldsymbol{n}}_6$, but $\tilde{\boldsymbol{n}}_6$ comprises only the $\tilde{n}_t$ component, then

$$V_6 = (\tilde{n}_t)_6.$$

Therefore, Eq. (5) finally becomes:

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\frac{1}{\Delta t(\tilde{n}_t)_6}\left[\boldsymbol{q}^{N+1}(\tilde{n}_t)_6 + \boldsymbol{q}^{N}(\tilde{n}_t)_5 + \sum_{l=1}^{4}\left\{\boldsymbol{q}^{N+\frac{1}{2}}\, \tilde{n}_t + \boldsymbol{H}^{N+\frac{1}{2}}\right\}_l\right].$$

A pseudo-time approach based on the Newton-iteration scheme is employed to solve unsteady flows, and the two-stage rational Runge-Kutta (RRK) scheme [19] is applied to pseudo-time stepping. Here, the two-stage RRK scheme can be written with pseudo-time t^* and pseudo-time step ν:

$$\begin{aligned} \boldsymbol{g}_1 &= -\Delta t^* \mathfrak{L}\left(\boldsymbol{q}^{N+1\langle\nu\rangle}\right) \\ \boldsymbol{g}_2 &= -\Delta t^* \mathfrak{L}\left(\boldsymbol{q}^{N+1\langle\nu\rangle} + c_2\boldsymbol{g}_1\right) \\ \boldsymbol{g}_3 &= b_1\boldsymbol{g}_1 + b_2\boldsymbol{g}_2 \\ \boldsymbol{q}^{N+1\langle\nu+1\rangle} &= \boldsymbol{q}^{N+1\langle\nu\rangle} + \frac{2\boldsymbol{g}_1(\boldsymbol{g}_1\cdot\boldsymbol{g}_3) - \boldsymbol{g}_3(\boldsymbol{g}_1\cdot\boldsymbol{g}_1)}{(\boldsymbol{g}_3\cdot\boldsymbol{g}_3)}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} \mathfrak{L}\left(\boldsymbol{q}^{N+1}\right) = \frac{1}{\Delta t\,(\tilde{n}_t)_6}&\left[\boldsymbol{q}^{N+1}\,(\tilde{n}_t)_6 + \boldsymbol{q}^{N}\,(\tilde{n}_t)_5 + \sum_{l=1}^{4}\left\{\boldsymbol{q}^{N+\frac{1}{2}}\,\tilde{n}_t + \boldsymbol{H}^{N+\frac{1}{2}}\right\}_l\right] \\ \boldsymbol{q}^{N+1\langle 0\rangle} &= \boldsymbol{q}^{N}. \end{aligned}$$

$\boldsymbol{g}_i \cdot \boldsymbol{g}_j$ represents the summation of the scalar product of vectors $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$ in all the cells. b_1, b_2, and c_2 are coefficients which satisfy the following relations:

$$b_1 + b_2 = 2.0, \quad b_2 c_2 \leq -0.5.$$

This study uses $b_1 = 1.0$, $b_2 = -1.0$, $c_2 = 2.0$ to provide second-order accuracy.

3 Moving computational multi-domain method

3.1 Moving computational domain (MCD) method

The MCD method [12] is applied when objects move. In order to simulate movement over a large space, an object is located in a computational domain which moves in line with the object inside, as shown in Fig. 2.

3.2 Classification of the cells and their interpolation

3.2.1 Categorising vertices and cells

Inter-grid communication can be achieved by applying the overset approach [9] to implement the interpolation. Each cell is categorised as either an active cell, a nonactive cell, or an interpolation cell, based on the positional relationship between one grid and the other overlapping grid. Active cells are the cells where the flow field variables are calculated; nonactive cells are the cells which are outside the calculating area; and the interpolation cells are the cells where the variables are interpolated.

The cells are classified by following the steps below. First, whether the nodes of each grid are included in the overlapping partner grid is determined. In 3D cases, when the number of owner cell searches for all nodes is massive, the searching algorithm plays a pivotal role in the efficiency of the methods. Thus, the KD-tree-based algorithm that serves the purpose is employed. After that, all the node points are designated as either active or nonactive based on the node classification method described in the study by Nakahashi [9]. Since objects are represented as holes in the grids, if a node is situated inside the object in the partner grid, the node is designated as an in-hole node. Then, a tetrahedron cell is assigned as an active cell if all the nodes of the cell are active; as a nonactive cell if all the nodes of the cell are nonactive or if one or more nodes of the cell are in-hole nodes; and the remaining cells are the interpolation cells which are responsible for inter-grid communication. If the layer of the detected interpolation cells is insufficiently thin, the cells around the interpolation cells are also reassigned as interpolation cells, ensuring a sufficient number of interpolated cells.

3.2.2 Finding donor cells

After the cells are classified, the donor cells are identified. The variables of the donor cells are used for the interpolation to obtain those of the interpolation cells. First, the nodes of each tetrahedron interpolation cell are checked to determine which overlapping partner cells contain them; Fig. 4a and b illustrate this in two-dimensions. In the next step, the distances between the centre of an interpolation cell and the centres of its detected partner cells are calculated, and in turn, the partner cell that has the largest distance is found, where the distance is determined as a reference length (Fig. 4c). Then, the distances between the centre of an interpolation cell and the centres of the other cells in the overlapping partner grid are calculated. The partner cells with smaller centre-to-centre distances than the reference length are identified as donor cells (Fig. 4d and e).

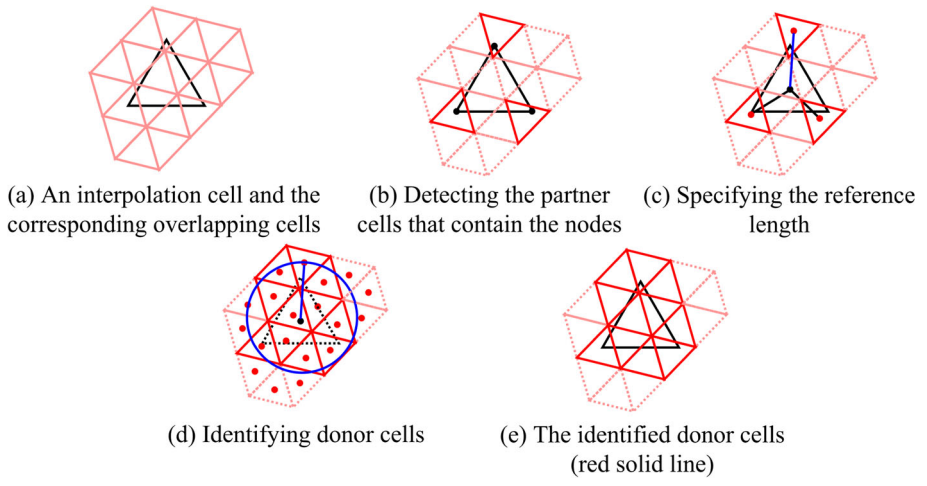


Fig. 4 Donor cell identification procedure (illustrated in two-dimensions)

It is important to note that if any of the donor cells detected for an interpolation cell is not active at the current or previous time step, the interpolation cell can no longer be an interpolation cell and must be changed to an active cell. This process ensures that variables are interpolated only from cells which have reliable values.

3.2.3 *Inverse distance weighted interpolation*

Inverse distance weighting is employed for the interpolation. $\boldsymbol{q}$ at the interpolation cells $\boldsymbol{q}^{\mathrm{ip}}$ is computed from the variables of the donor cells $\boldsymbol{q}^{\mathrm{donor}}$:

$$\boldsymbol{q}^{\mathrm{ip}} = \frac{\sum_{i=1}^{m} w_i \boldsymbol{q}_i^{\mathrm{donor}}}{\sum_{i=1}^{m} w_i}$$
$$w_i = \frac{1}{d\left(\mathcal{X}^{\mathrm{ip}}, \mathcal{X}_i^{\mathrm{donor}}\right)^p}.$$

Here, $d\left(\mathcal{X}^{\mathrm{ip}}, \mathcal{X}_i^{\mathrm{donor}}\right)$ is the distance between the centre of an interpolation cell $\mathcal{X}^{\mathrm{ip}}$ and the centres of the corresponding donor cells $\mathcal{X}_i^{\mathrm{donor}}$, and m represents the total number of donor cells found for an interpolation cell. The power parameter $p = 1$ is used in this study.

3.2.4 *Modification of the RRK scheme for inter-grid communication*

The cells which are outside the flow field must be excluded in the process of calculating flow. Therefore, each cell i has the following value depending on whether the cell is a blanked cell or not:

$$\mathrm{iblank}(i) = \begin{cases} 1, & \text{if a cell is not blanked} \\ 0, & \text{if a cell is blanked.} \end{cases}$$

Multiplying this iblank(i) to the formula of the RRK scheme ensures that the solutions are not updated in blanked cells. Since active cells are the cells that calculate the flow field variables, interpolation cells are the cells that only perform interpolation, and nonactive cells are the cells outside the calculating area, iblank(i) takes the form of the following equation:

$$\text{iblank}(i) = \begin{cases} 1, & \text{if a cell is active} \\ 0, & \text{else.} \end{cases}$$

Therefore, Eq. (6) is modified with the preceding iblank:

$$\boldsymbol{q}_i^{N+1\langle\nu+1\rangle} = \boldsymbol{q}_i^{N+1\langle\nu\rangle} + \text{iblank}(i)\ \frac{2\boldsymbol{g}_{1i}(\boldsymbol{g}_1 \cdot \boldsymbol{g}_3) - \boldsymbol{g}_{3i}(\boldsymbol{g}_1 \cdot \boldsymbol{g}_1)}{(\boldsymbol{g}_3 \cdot \boldsymbol{g}_3)}. \tag{7}$$

In the actual coding, blanked cells should be skipped for the sake of efficiency.

4 Validations

Two cases were tested to validate the reliability of this method. One is a shock tube test including a moving grid inside. An empty grid (subgrid) was prepared in the shock tube (main grid), and the subgrid was moved. Since the subgrid does not contain any object inside, the purpose of this test is to confirm that the main flow is not affected by grid movement. In addition, the results can be compared with the exact solution to verify that the variables are accurately interpolated between the overlapping grids.

Another validation test involves crossing two spheres. This context is close to the practical application of this method. However, in general, the simulation of real-life vehicles is expensive and intricated in terms of modelling the configuration and condition details. Therefore, to demonstrate the capability of the proposed method, we opt for a more straightforward test case which resembles real-life problems, such as vehicles passing each other. In addition, a sphere-crossing simulation has been reported so far in [11]. For this reason, the current study uses sphere crossing as a validation test to allow the results to be compared with the values calculated using the method reported by Yamakawa [11].

4.1 Shock tube test

4.1.1 Computational conditions

A gas at high pressure and a gas at low pressure are separated in a tube by the diaphragm, which serves as a barrier. A subgrid was placed in the low-pressure region of a shock tube (main grid), and the grid was moved in the x and z directions. The initial positions of the main grid and subgrid are depicted in Fig. 5. Since a shock wave passes through the low-pressure region of the main grid, the subgrid was positioned in the region in order to verify that the interpolation does not change the shape of the shock wave. The movement in the y direction was fixed, allowing the flow and the movement of the subgrid to be examined at a cross section perpendicular to the y-axis in a recognisable manner. Thus, the subgrid was moved in Eq. (8) from rest at $t = 0$. The initial conditions are shown in Table 1. The boundary conditions at the wall of the main grid were the slip conditions

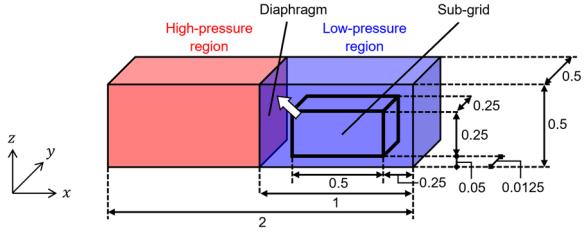


Fig. 5 Shock tube test

Table 1 Initial conditions (shock tube test)

High-pressure region (subscript *H*)		Low-pressure region (subscript *L*)	
ρ_H	1.0	ρ_L	0.1
u_H	0.0	u_L	0.0
v_H	0.0	v_L	0.0
w_H	0.0	w_L	0.0
p_H	ρ_H/γ	p_L	ρ_L/γ

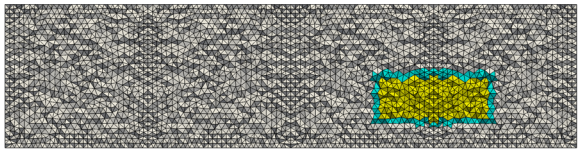
for an inviscid fluid. All the cells at the wall of the subgrid were interpolation cells. The variables in the subgrid were calculated using the values interpolated from the outer main grid. Conversely, the information from the subgrid was communicated to the main grid through the interpolation cells of the main grid. The test was carried out until the shock wave reached the right end of the main grid.

$$\begin{aligned} \Delta x &= -0.0375 - 0.0375 \sin(2\pi t - 0.5\pi) \\ \Delta z &= 0.075 + 0.075 \sin(\pi t - 0.5\pi) \end{aligned} \tag{8}$$

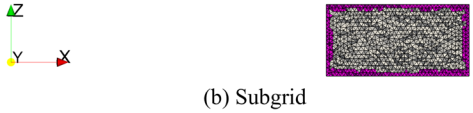
The two unstructured grids for the main grid with 512152 cells and for the subgrid with 80928 cells were generated by *MEGG3D* (an unstructured mesh generator [20, 21]). Figure 6 shows the cross section at the middle of the length in the y direction of (a) the main grid and (b) the subgrid, with colouring according to the cell type. The white cells represent active cells, the cyan cells are the interpolation cells of the main grid, the magenta cells are the interpolation cells of the subgrid, and the yellow cells are nonactive cells. Figure 6 implies that the region of the main grid which overlaps the subgrid was occupied with nonactive cells, and the subgrid performed the calculation instead.

4.1.2 Results and discussion

The graph in Fig. 7 provides the density distribution at the centre of the length in the y and z directions of the grids and compares it with the exact solution. Figure 8 presents the flow field in the cross section at the centre of the length in the y direction. The upper side illustrates the density distribution while the lower side illustrates the grids. The nonactive cells are omitted, and only the active and interpolation cells are shown. The colouring rule is the same as in Fig. 6, but the active cells of the subgrid are shown in grey tones to emphasise the fact that they belong to the subgrid.



(a) Shock tube (main grid)



(b) Subgrid

Fig. 6 Computational grids for the shock tube test

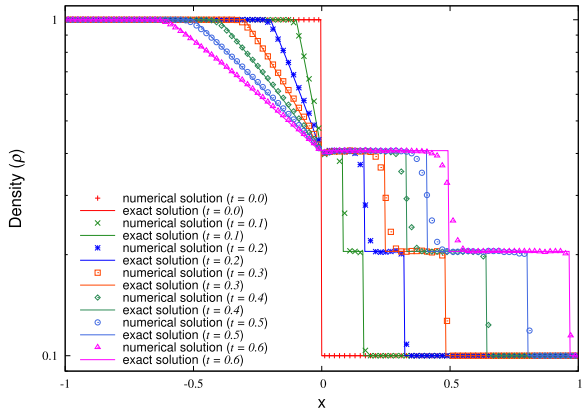


Fig. 7 Comparison of density with the exact solution (shock tube test)

The grids in Fig. 8 show that the subgrid moved in x and z directions. In addition, the interpolation cells of each grid did not overlap but were slightly shifted to ensure that the donor cells were active cells, as mentioned in Section 3.2.2. It can also be seen that the classification of each cell was dynamically updated at each time step as the subgrid moved. The density distribution in Fig. 8 shows that the shock propagated after the diaphragm was removed. The shape of the shock remained after it passed through the

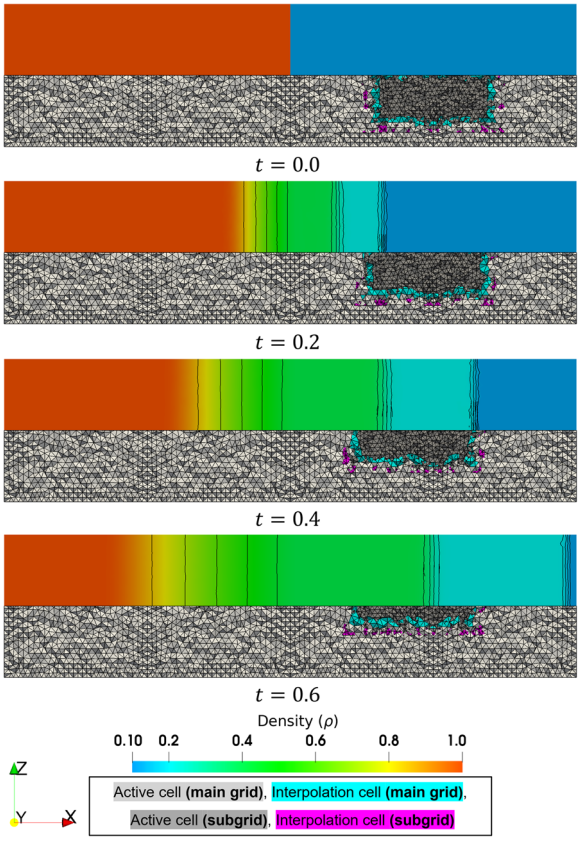


Fig. 8 Flow field at each time step (shock tube test)

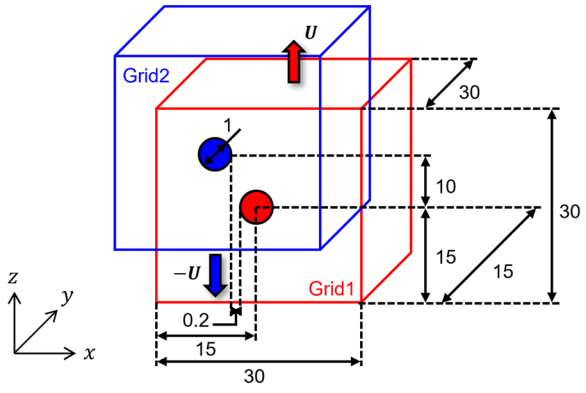


Fig. 9 Crossing test

subgrid. Figure 7 shows the position of the shock that was captured at each time, compared with the exact solution. Therefore, these results verify that grid movement does not affect the flow and that the flow field variables are accurately communicated between the two grids.

4.2 Crossing test

4.2.1 Computational conditions

Two identical grids which include a sphere in the centre were prepared in the configuration depicted in Fig. 9. The grids are not shifted in the y direction. To compare with the values provided using the method described in Section 2 and 3 in [11], the spheres were placed in exactly the same initial positions and were moved at exactly the same speed as the ones in the literature. The comparative data were calculated using remeshing, adding and eliminating the cells as the spheres moved, with a single grid including the two spheres, as depicted in Fig. 7 of the literature. Please refer to the literature for the detailed methodology for the comparative data.

In the current study, the two spheres started to approach each other at speed U in the z direction from the position where the centres were 10 apart and crossed at a distance of 0.2. The moving speed U is shown in Eq. (9). Since the velocity is nondimensionalised by the speed of sound in this study, U was equal to the speed of sound when $U = 1.0$. The initial conditions are shown in Table 2. The boundary conditions were the slip conditions at the wall of the spheres and the Riemann invariant boundary conditions at the outer boundaries of each grid.

$$U = \begin{cases} 10t, & t < 0.1 \\ 1.0, & t \geq 0.1 \end{cases} \tag{9}$$

Table 2 Initial conditions (crossing test)

Whole region	
ρ	1.0
u	0.0
v	0.0
w	0.0
p	ρ/γ

The two identical unstructured grids with 502506 cells were generated by *MEGG3D*. The fineness of the computational grids was as close as possible to that of the literature. Figure 10 illustrates the cross section of the grids at the centre of the spheres in the y direction, with (a) showing the overlapping grids without the nonactive cells, with (b) and (c) showing Grid 1 and Grid 2, respectively, and with (d) showing the cells around the sphere. The cells are coloured according to the cell type. The white cells represent active cells, the magenta cells are the interpolation cells of Grid 1, the cyan cells are the interpolation cells of Grid 2, and the yellow cells are nonactive cells. At the region where Grid 1 and Grid 2 overlap, only one grid or the other has nonactive cells.

4.2.2 Scalability of overlapping cell search

We investigated the scalability of the KD-tree-based algorithm for searching for the owner cell for each node, and we depicted the results in Fig. 11. The figure also includes the $y = A_0 x \log(x)$ graph (A_0 is an arbitrary number) as a reference. This graph indicates that the performance of our numerical implementation of the KD-tree search is consistent with the theoretical prediction $\mathcal{O}(n \log n)$.

4.2.3 Results and discussion

Figure 12 shows the cross section of the grids at the centre of the spheres in the y direction, with each time step of the spheres (a) approaching each other ($t = 1.6$), (b) crossing

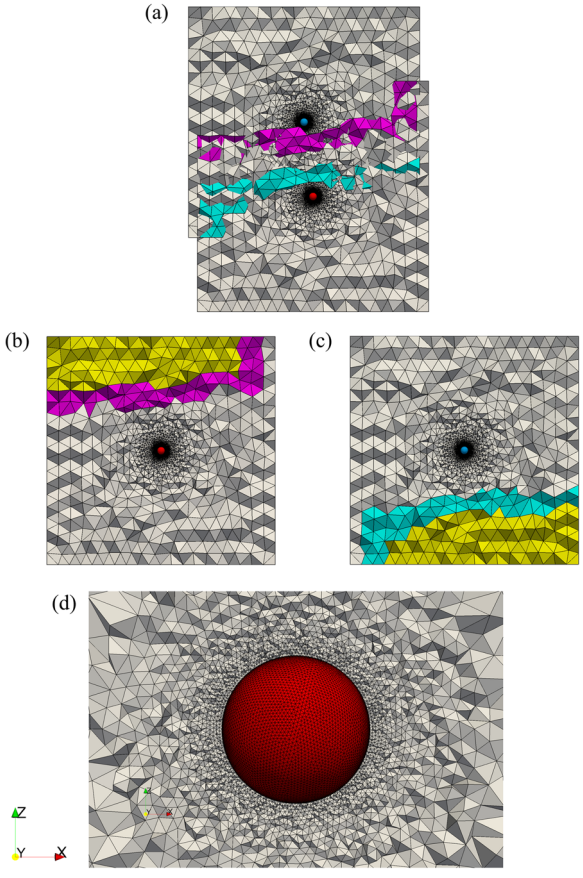


Fig. 10 Computational grids for the crossing test

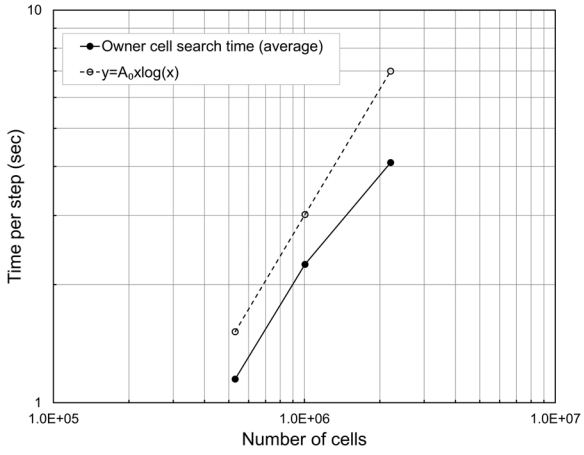


Fig. 11 Scalability test (crossing test)

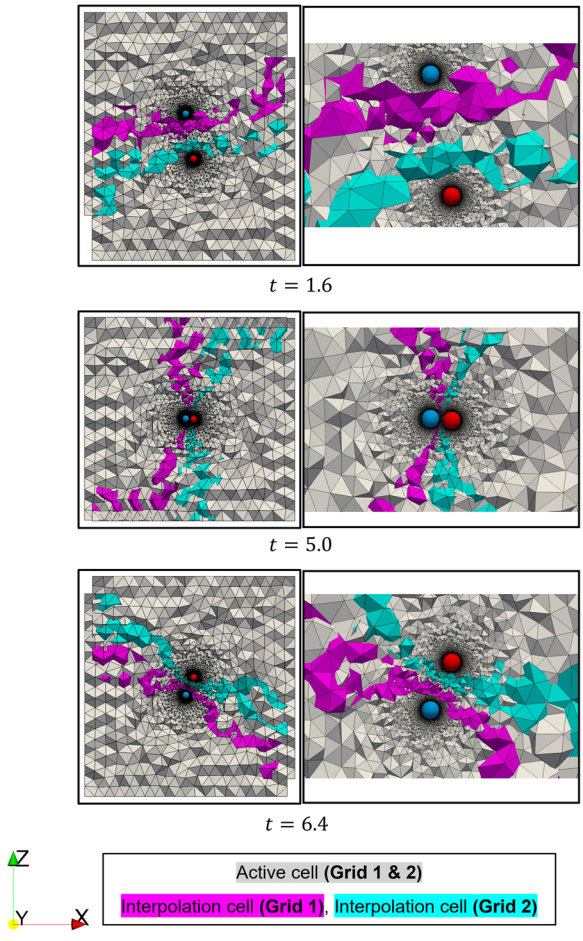


Fig. 12 Computational grids at each time step (crossing test)

($t = 5.0$), and (c) moving away from each other ($t = 6.4$). The left side depicts the whole view, and the right side illustrates the enlarged view around the spheres. The nonactive cells are omitted, and only the active and interpolation cells are shown. The colouring rule is the same as in Fig. 10. Figure 13 shows the pressure distribution on the surface of the

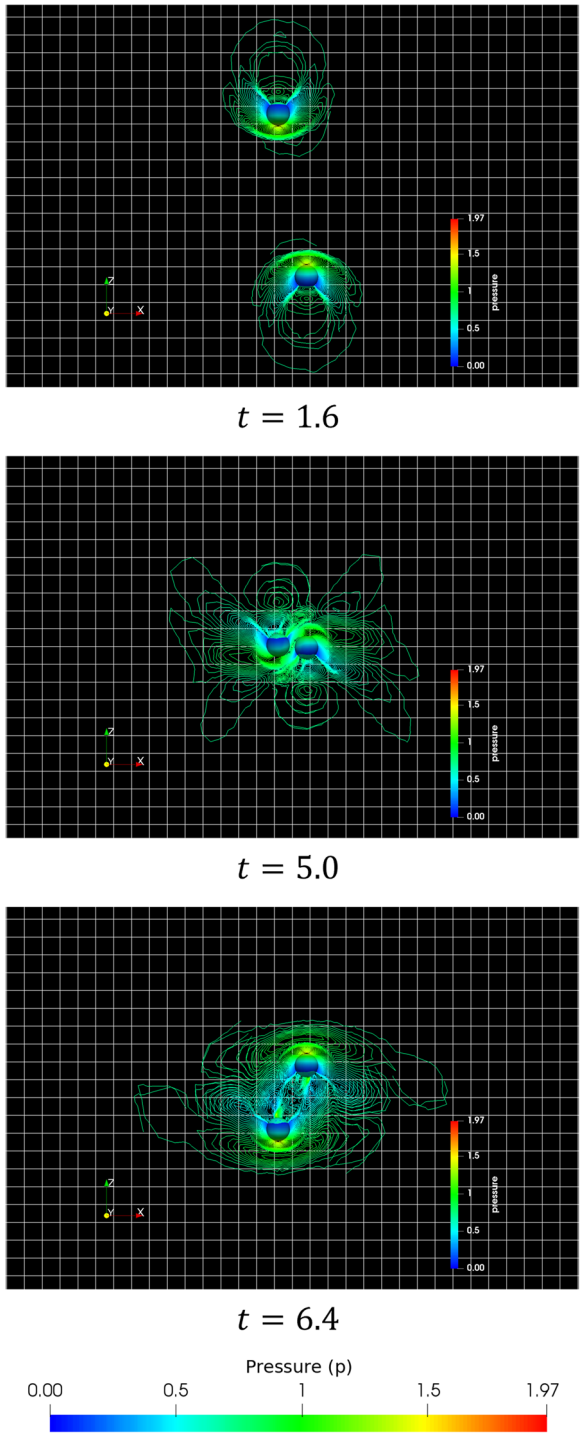


Fig. 13 Pressure distribution on the sphere surface and pressure contour at each time step (crossing test)

spheres and the contour plot of the pressure in the flow field at each time step. The graph in Fig. 14 provides the time histories of the drag coefficient C_D, the lift coefficient C_L, and the side-force coefficient C_S for one sphere and compares them with the values calculated using the method of the literature [11]. These coefficients are calculated by Eq. (10):

$$C_D = \frac{2F_D}{\rho V^2 S},\ C_L = \frac{2F_L}{\rho V^2 S},\ C_S = \frac{2F_S}{\rho V^2 S}, \tag{10}$$

where F_D, F_L, and F_S represent the drag, lift, and side-force, respectively, which are the forces exerted on the sphere in the z-, y-, and x-axes, respectively. V indicates the maximum speed of the sphere, and S denotes the frontal area of the sphere.

The grids in Fig. 12 show that the computational domains followed the movement of the corresponding spheres as they moved. Each domain moved independently, and the classification of each cell was updated as the steps progress. Figure 13 implies that a bow shock was generated in front of the spheres, which travel at the speed of sound, and that another shock was generated behind the spheres at $t = 1.6$. When crossing at $t = 5.0$, the shock in front of one sphere and the shock behind the other sphere interacted. After crossing at $t = 6.4$, the shock behind each sphere affected the rear of each other's sphere. Similar phenomena can be observed in the literature by Yamakawa [11], and therefore it can be said that the flow field variables were successfully communicated between the domains in this study. Figure 14 shows that C_D made a dramatic increase just before crossing and then fell drastically as the crossing was over. These phenomena were caused by the mutual influence of the high-pressure fields in front of the two spheres and the low-pressure fields behind the spheres. In addition, fluctuation can also be observed in C_S, and this implies that the forces which made the spheres waver were exerted on the spheres when crossing. The comparative data (the dashed lines) in Fig. 14 suggests that the same trends can be seen in all the drag, lift, and the side-force coefficients. In particular, these coefficients were in good agreement with the comparative data when crossing, which is a noticeable feature in this simulation.

These results show that the flow interaction of multiple moving objects can be simulated using computational domains created around each object that communicate with each other. On the other hand, those multiple-object simulations using a single fixed computational domain containing all the objects normally require remeshing the inside of

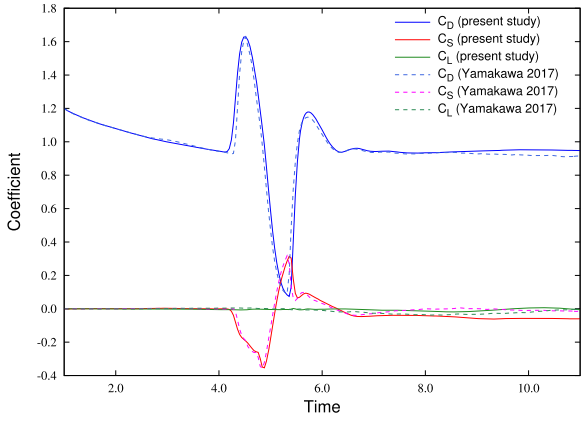


Fig. 14 Comparison with the values calculated using the method proposed by the literature [11] (crossing test)

the domain as the objects move. Such remeshing needs to introduce complex techniques, such as adding and eliminating cells as represented in Yamakawa [11]. In addition, the limitation in the size of the computational domain constrains the space in which objects can move. In contrast, remeshing is unnecessary with the method introduced in the current study. Moreover, the computational domains themselves can be moved freely, which makes it possible to move each object along with the corresponding domain, without spatial limitations.

5 Conclusions

This study introduces moving computational domains that communicate with each other as a means of simulating the full trajectory of multiple moving objects. The MCD method is applied to modelling the movement of objects. In this method, computational domains are moved in line with the trajectory of each object, thereby removing the spatial constraints for simulating moving objects. The interaction of each domain's flow with the other's flow is achieved by applying the overset approach, which allows the communication of the flow field variables. A shock tube test is conducted to verify that the basic requirements for this method are satisfied. The comparison of the results to the exact solution confirms that the flow variables are accurately interpolated between the domains. Then, the crossing of two spheres is simulated. This test demonstrates that the shock wave generated around each sphere interacts with the other, and that fluctuation can be observed in the drag coefficient of one sphere. Also, fluctuation observed in the side-force coefficient of the sphere suggests that the forces that cause the spheres to waver are exerted on them when crossing. These results verify that this method can reliably model the flows around multiple objects and their interaction.

Abbreviations

3D: Three-Dimensional; 4D: Four-Dimensional; CFD: Computational Fluid Dynamics; MCD: Moving Computational Domain; MGFV: Moving-Grid Finite Volume; GCL: Geometric Conservation Law; MUSCL: Monotonic Upstream-centred Scheme for Conservation Laws; RRK: Rational Runge-Kutta; MEGG3D: Mixed-Element Grid Generator in 3 Dimensions

Acknowledgements

N/A.

Authors' contributions

MN is the corresponding author of this paper, who was responsible for conceptualisation, program development, and writing. MY contributed to conducting the validation tests, interpreting data, and writing the manuscript. SA and ST participated in the discussion and contributed to interpreting data and writing the manuscript. MBG participated in the discussion and contributed to developing the software and writing the manuscript.. All the authors read and approved the final manuscript.

Funding

This publication was subsidised by JKA through its promotion funds from KEIRIN RACE and by JSPS KAKENHI Grant Number 21K03856.

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Department of Mechanical and System Engineering, Kyoto Institute of Technology, Kyoto, Japan. [2] Department of Mechanical Engineering, College of Industrial Technology, Hyogo, Japan.

Received: 21 September 2021 Accepted: 13 December 2021
Published online: 24 January 2022

References

1. Takii A, Yamakawa M, Asao S, Tajiri K (2020) Turning flight simulation of tilt-rotor plane with fluid-rigid body interaction. J Therm Sci Technol 15(2):JTST0021. https://doi.org/10.1299/jtst.2020jtst0021
2. Peskin CS (1972) Flow patterns around heart valves: A numerical method. J Comput Phys 10(2):252–271. https://doi.org/10.1016/0021-9991(72)90065-4
3. Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37(1):239–261. https://doi.org/10.1146/annurev.fluid.37.061903.175743
4. Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852. https://doi.org/10.1016/j.jcp.2008.01.028
5. Ghomizad MB, Kor H, Fukagata K (2021) A structured adaptive mesh refinement strategy with a sharp interface direct-forcing immersed boundary method for moving boundary problems. J Fluid Sci Tech 16(2):JFST0014. https://doi.org/10.1299/jfst.2021jfst0014
6. Posa A, Balaras E (2014) Model-based near-wall reconstructions for immersed-boundary methods. Theor Comput Fluid Dyn 28:473–483. https://doi.org/10.1007/s00162-014-0326-5
7. Kor H, Ghomizad MB, Fukagata K (2018) Extension of the unified interpolation stencil for immersed boundary method for moving boundary problems. J Fluid Sci Tech 13(2):JFST0008. https://doi.org/10.1299/jfst.2018jfst0008
8. Steger JL, Benek JA (1987) On the use of composite grid schemes in computational aerodynamics. Comput Methods Appl Mech Eng 64(1–3):301–320. https://doi.org/10.1016/0045-7825(87)90045-4
9. Nakahashi K, Togashi F, Sharov D (2000) Intergrid-boundary definition method for overset unstructured grid approach. AIAA J 38(11):2077–2084. https://doi.org/10.2514/2.869
10. Crozon C, Steijl R, Barakos GN (2018) Coupled flight dynamics and CFD — demonstration for helicopters in shipborne environment. Aeronaut J 122(1247):42–82. https://doi.org/10.1017/aer.2017.112
11. Yamakawa M, Mitsunari N, Asao S (2017) Numerical simulation of rotation of intermeshing rotors using added and eliminated mesh method. Procedia Comput Sci 108:1883–1892. https://doi.org/10.1016/j.procs.2017.05.061. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland
12. Watanabe K, Matsuno K (2009) Moving computational domain method and its application to flow around a high-speed car passing through a hairpin curve. J Comput Sci Technol 3(2):449–459. https://doi.org/10.1299/jcst.3.449
13. Matsuno K (2010) Development and applications of a moving grid finite volume method. In: Topping BHV, Adam JM, Pallarés FJ, Bru R, Romero ML (eds). Developments and applications in engineering computational technology. Chapter 5. Saxe-Coburg Publications, Stirlingshire. pp 103–129. https://doi.org/10.4203/csets.26.5
14. Thomas PD, Lombard CK (1979) Geometric conservation law and its application to flow computations on moving grids. AIAA J 17(10):1030–1037. https://doi.org/10.2514/3.61273
15. Takii A, Yamakawa M, Asao S, Tajiri K (2020) Six degrees of freedom flight simulation of tilt-rotor aircraft with nacelle conversion. J Comput Sci 44:101164. https://doi.org/10.1016/j.jocs.2020.101164
16. Yamakawa M, Mizuno N, Asao S, Tanaka M, Tajiri K (2020) Optimization of knee joint maximum angle on dolphin kick. Phys Fluids 32(6):067105. https://doi.org/10.1063/1.5142422
17. Roe PL (1981) Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 43(2):357–372. https://doi.org/10.1016/0021-9991(81)90128-5
18. Hishida M, Hashimoto A, Murakami K, Aoyama T (2011) A new slope limiter for fast unstructured CFD solver FaSTAR (*in Japanese*). In: JAXA Special Publication: Proceedings of 42nd Fluid Dynamics Conference / Aerospace Numerical Simulation Symposium 2010, JAXA-SP-10-012. https://ci.nii.ac.jp/naid/120006828212/
19. Wambecq A (1978) Rational Runge-Kutta methods for solving systems of ordinary differential equations. Computing 20(4):333–342. https://doi.org/10.1007/BF02252381
20. Ito Y, Nakahashi K (2002) Surface triangulation for polygonal models based on CAD data. Int J Numer Methods Fluids 39:75–96. https://doi.org/10.1002/fld.281
21. Ito Y (2013) Challenges in unstructured mesh generation for practical and efficient computational fluid dynamics simulations. Comput Fluids 85:47–52. https://doi.org/10.1016/j.compfluid.2012.09.031

Publisher's Note