Influence of mesh non-orthogonality on numerical simulation of buoyant jet flows

Masahiro Ishigaki[a,*], Satoshi Abe[a], Yasuteru Sibamoto[a], Taisuke Yonomoto[a]

[a] *Japan Atomic Energy Agency, 2-4, Shirakata, Tokai-mura, Naka-gun, Ibaraki, 319-1195, Japan*

Abstract

In the present research, we discuss the influence of mesh non-orthogonality on numerical solution of a type of buoyant flow. Buoyant jet flows are simulated numerically with hexahedral and prismatic mesh elements in an open source Computational Fluid Dynamics (CFD) code called "OpenFOAM". Buoyant jet instability obtained with the prismatic meshes may be overestimated compared to that obtained with the hexahedral meshes when non-orthogonal correction is not applied in the code. Although the non-orthogonal correction method can improve the instability generated by mesh non-orthogonality, it may increase computation time required to reach a convergent solution. Thus, we propose modified solvers that can reduce the influence of mesh non-orthogonality and reduce the computation time compared to the existing solvers in OpenFOAM. It is demonstrated that calculations for a buoyant jet with a large temperature difference are performed faster by the modified solver.

Keywords: buoyant flow, prismatic and hexahedral mesh, mesh non-orthogonality, finite volume method, OpenFOAM

Highlights

- Influence of mesh non-orthogonality on numerical solution of buoyant jet flows.

*Corresponding author
Email address: ishigaki.masahiro@jaea.go.jp (Masahiro Ishigaki)

- Buoyant jet flows are simulated with hexahedral and prismatic meshes.
- Jet instability with prismatic meshes may be overestimated compared to that with hexahedral meshes.
- Modified solvers that can reduce the influence of mesh non-orthogonality and reduce computation time are proposed.

1. Introduction

After the Fukushima Dai-ichi nuclear power plant accident in March 2011 Japan Atomic Energy Agency (JAEA) started a new project called "ROSA-SA" (Yonomoto et al., 2015), which focuses on thermal-hydraulics behavior in a containment vessel (CV) of a nuclear power plant during severe accidents. This project aims to contribute to new regulatory standards and continuous improvement of safety in nuclear power plants. During a severe accident, water-zirconium reaction occurs in the core, which generates hydrogen gas. Then, hydrogen combustion may arise in a CV. Therefore, it is necessary to understand the gaseous behavior and distribution of hydrogen for reducing the hydrogen risk and improving safety. Since a CV has a large volume in which complex three-dimensional thermal hydraulic behaviors can be seen, Computational Fluid Dynamics (CFD) analysis is used for detailed evaluations.

In 2014, the "International Benchmark Exercise: IBE-3" on CFD analysis of CVs was performed to contribute toward improvement of CFD analysis by using the large CV test facility PANDA at Paul Scherrer Institute (PSI) (Andreani et al., 2014). Jet impingement upon density stratification formed by air and helium gas mixture was investigated as the target experiment in the benchmark test. Blind analyses of this experiment were performed by the participants where they knew only the initial and the boundary conditions. After the experimental results were opened to the participants, post-test analyses were performed. We participated in the blind and the post-test analyses, and employed an open source CFD code called OpenFOAM (OpenFOAM, 2004).

OpenFOAM is based on finite volume method. In our blind analysis, we employed tetrahedral meshes instead of hexahedral meshes for simplicity, because the PANDA facility had complex geometry and mesh generation using tetrahedral meshes was simpler compared to that using hexahedral meshes. The results of our blind analysis were considerably different from the experimental results. To resolve this discrepancy in the numerical results, we used hexahedral meshes in our post-test analysis. Although the same solver and the same turbulence model were used in both analyses, the results obtained by using the hexahedral meshes were drastically different from the results of the blind analysis, and they showed considerably better agreement with the experimental results (Abe et al., 2015).

This problem seems to be well known among the researchers working on containment thermal hydraulic safety. For example, in the appendix of the final summary report of the OECD/SETH-2 project (OECD/NEA, 2012), it is mentioned that "Users reported that unstructured meshes are unusable for simulating stratification".

These results indicate that differences in mesh cell shapes have a large influence on numerical solutions of density stratified flows or buoyant flows. These flows are dominated by buoyancy. Generally, the generation of hexahedral meshes for complex geometry is more difficult than the generation of tetrahedral meshes, and the use of tetrahedral meshes cannot be avoided in some cases. Therefore, it is important to investigate the influence of cell shape on numerical simulations of flows dominated by buoyancy.

Here, we focus on buoyant flow, and studies involving numerical simulations of buoyant flows are reviewed. Generally, a greater number of studies using hexahedral meshes (structured meshes) have been reported than those using tetrahedral meshes (unstructured meshes). For studies of buoyant flows driven by temperature difference in a cavity, many researchers have used hexahedral meshes, e.g., Dixit and Babu (2006), Oliveira and Issa (2001), and de Vahl Davis (1983), and so on. In addition, for analyses of buoyant jets, e.g., Nam and Bill, Jr (1993), Yan and Holmstedt (1999), Kumar and Dewan (2014) and Beccantini

et al. (2008) used hexahedral meshes. In these studies, the influence of cell shape was not focused on. On the other hand, Boivin et al. (2000) developed a numerical scheme of the finite volume method for unstructured meshes and they analyzed buoyant flow in a cavity as a benchmark test. Zitzmann et al. (2005) simulated thermal convection with tetrahedral meshes. However, they did not discuss the sensitivity of numerical solution to mesh cell shape. To the best of the authors' knowledge, few studies focus on the influence of cell shape on buoyant flows.

Accordingly, we analyze buoyant flows with hexahedral and tetrahedral meshes to discuss the influence of cell shapes on numerical simulations of buoyant flows in this study. For the numerical analysis, we use OpenFOAM 2.1.x in the same way as it was used in the IBE-3 test analyses. We focus on the influence of mesh non-orthogonality. Because non-orthogonality of tetrahedral meshes is generally much higher than that of hexahedral meshes, mesh non-orthogonality may have a great influence on the numerical results. Non-orthogonal correction methods for relaxation of the influence of mesh non-orthogonality have been proposed by Muzaferija (1994), Ferziger and Perić (1997), and Jasak (1996). Non-orthogonal correction is implemented in OpenFOAM. The influence of non-orthogonal correction on buoyant flow simulation is also discussed here.

In Section 2 procedure of the non-orthogonal correction implemented in OpenFOAM is described. We investigate two types of the thermal flow solutions; one is based on the Boussinesq approximation and the other is the low Mach number weakly compressible formulation. The Boussinesq approximation is discussed as the basic solution related to thermal flows. Solution methods of the solvers implemented in OpenFOAM for thermal flows are described in Sections 3 and 4. The solver based on the Boussinesq approximation is described in Section 3, and the solver based on the low Mach number weakly compressible formulation is described in Section 4. Increasing computation time to reach a convergent solution is expected generally by non-orthogonal correction, because correction requires face flux to be solved implicitly. (Details are discussed in Section 2.) Therefore, a new solver that can reduce the influence of mesh

non-orthogonality and computation time is desired. Accordingly, modified numerical solutions for thermal flow simulations are proposed in Sections 3 and 4. A few test analyses of buoyant flows are discussed in Sections 5 and 6. In these test analyses, the effect of the non-orthogonal correction and performance of the modified solvers are discussed.

2. Non-orthogonal correction

Mesh non-orthogonality is defined as an angle between a line connecting two cell centers in a mesh and a face normal vector. According to the the OpenFOAM user guide (OpenFOAM, 2016), the use of non-orthogonal correction is generally recommended. On the other hand, non-orthogonal correction is not necessary for a mesh with very low non-orthogonality (e.g., the maximum non-orthogonality lower than 5 degrees). For non-orthogonality greater than 80 degrees, it is generally hard to obtain a convergent solution. Non-orthogonal correction is formulated in OpenFOAM, and this method is called the "over-relaxed approach", as shown by Jasak (1996). In this section, a brief overview of non-orthogonal correction is given according to Jasak (1996).

The non-orthogonal correction is applied to the Laplacian term expressed as (1). Equation (1) also expresses a discretized formulation of the integrated Laplacian term of any scalar ϕ in a given mesh cell. The suffix f refers to the value of the cell interface of the mesh and $\boldsymbol{S}_f$ denotes a face area vector orthogonal to the cell interface. Γ is the diffusion coefficient of the scalar ϕ.

$$\begin{aligned}\int_V \nabla \cdot (\Gamma \nabla \phi) dV &= \int_S \Gamma \nabla \phi \cdot d\boldsymbol{S} \\ &\approx \sum_f \Gamma_f \nabla \phi_f \cdot \boldsymbol{S}_f \end{aligned} \tag{1}$$

Figure 1 shows the vector relationship between the cell centers P and N.[1]

[1]Calculation procedure of the cell center position in OpenFOAM is summarized in Appendix A.

ϕ is defined for each cell center. If a mesh is orthogonal, the vectors $\boldsymbol{d} = \overrightarrow{PN}$ and $\boldsymbol{S}_f$ are parallel. By application of central differencing to the gradient term (Versteeg and Malalasekera (2007), etc), the following expression for cell P is obtained.

$$\nabla\phi_f \cdot \boldsymbol{S}_f = \frac{\phi_{\mathrm{N}} - \phi_{\mathrm{P}}}{|\boldsymbol{d}|}|\boldsymbol{S}_f| \tag{2}$$

This equation also corresponds to the flux expression used in the classical form of the Two-Point Flux Approximation scheme (TPFA scheme). (See Fuhrmann et al. (2014), etc., for example.)

In the case that a mesh is non-orthogonal, as shown in Fig.1, that is, $\boldsymbol{d}$ and $\boldsymbol{S}_f$ are not parallel, $\boldsymbol{S}_f$ is split into two components $\boldsymbol{S}_f = \boldsymbol{\Delta} + \boldsymbol{k}$. Here $\boldsymbol{k}$ is defined so that it satisfies the relationship $\boldsymbol{S}_f \cdot \boldsymbol{k} = 0$. $\boldsymbol{\Delta}$ is defined as below.

$$\boldsymbol{\Delta} = \frac{|\boldsymbol{S}_f|^2}{\boldsymbol{d} \cdot \boldsymbol{S}_f}\boldsymbol{d} \tag{3}$$

From the above relationship, the following equation (4) is obtained.

$$\begin{aligned}\nabla\phi_f \cdot \boldsymbol{S}_f &= \nabla\phi_f \cdot \boldsymbol{\Delta} + \nabla\phi_f \cdot \boldsymbol{k} \\ &= \frac{\phi_{\mathrm{N}} - \phi_{\mathrm{P}}}{|\boldsymbol{d}|}|\boldsymbol{\Delta}| + \nabla\phi_f \cdot \boldsymbol{k}\end{aligned} \tag{4}$$

The second term on the right hand side in (4) is calculated implicitly as the non-orthogonal correction. The value of the gradient at the interface in the second term is obtained by interpolation of the gradient values at cell centers P and N, and it is expressed as follows:

$$\nabla\phi_f = f_x\nabla\phi_{\mathrm{P}} + (1 - f_x)\nabla\phi_{\mathrm{N}} \tag{5}$$

where f_x is the interpolation factor. The gradient at the cell center is calculated by $\nabla\phi_{\mathrm{P}} = \sum_f \boldsymbol{S}_f\phi_f/V_{\mathrm{P}}$, and V_{P} is the cell volume. ϕ_f at each face is calculated by interpolation from the cell center values of the adjacent cells. This correction generally requires iterative calculations because the second term on the right hand side in (4) is calculated implicitly, and it is necessary to update this term iteratively.

Buoyant flow solvers implemented in OpenFOAM are based on the PISO method proposed by Issa (1985), and the details of the solutions are shown in the following sections. Based on the PISO method, the pressure equation includes a Laplacian term of pressure, and pressure is corrected by iterative calculations. The solution flow of pressure correction is shown in Fig.2. The iteration loop of pressure correction includes an iteration loop of the non-orthogonal correction. Generally, the non-orthogonal correction loop is executed once or twice according to the user guide (OpenFOAM, 2016). This means that computation time for convergence may be increased by the non-orthogonal correction, because the correction requires the face flux to be solved implicitly.

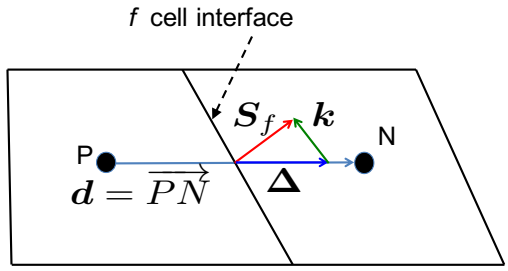

Figure 1: Schematic sketch of non-orthogonal cells P and N

3. Numerical solver using Boussinesq approximation

Two types of solvers for buoyant flow analyses are implemented in OpenFOAM. One is based on the Boussinesq approximation. The other is based on the low Mach number weakly compressible formulation by Zhou et al. (2001). In this section, the solver based on the Boussinesq approximation "buoyantBoussinesqPimpleFoam" is discussed. The source code of this solver is available in reference (Github, 2011).

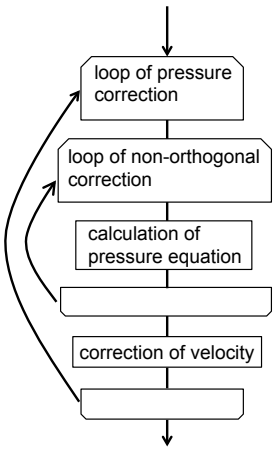

Figure 2: Solution flow of pressure correction

3.1. Numerical solution method of "buoyantBoussinesqPimpleFoam" solver implemented in OpenFOAM

The basic equations are mass and momentum equations, which are (6), (7), and the transport equation of temperature (8).

$$\nabla \cdot \boldsymbol{u} = 0 \tag{6}$$

$$\rho_{ref}\left(\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u})\right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \rho \boldsymbol{g} \tag{7}$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (\boldsymbol{u} T) = \kappa \nabla^2 T \tag{8}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, ρ is the density of gas, ρ_{ref} is the reference density, and T is the gas temperature. μ and κ are the viscosity coefficient and the thermal diffusivity coefficient, respectively. $\boldsymbol{g}$ is gravitational acceleration. Based on the Boussinesq approximation, (7) is modified to obtain the following equation (9).

$$\begin{aligned}\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = &- \nabla\left(\frac{p}{\rho_{ref}}\right) + \nu \nabla^2 \boldsymbol{u} \\ &+ [1 - \beta(T - T_{ref})]\boldsymbol{g}\end{aligned} \tag{9}$$

Here, ρ_k is defined as $\rho_k = 1 - \beta(T - T_{ref})$. ν is the kinematic viscosity coefficient. β is the thermal expansion coefficient. p/ρ_{ref} is rewritten as p. (9)

can be modified as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = -\nabla \left(p - \rho_k \boldsymbol{g} \cdot \boldsymbol{x}\right) + \nu \nabla^2 \boldsymbol{u} - (\boldsymbol{g} \cdot \boldsymbol{x}) \nabla \rho_k \tag{10}$$

where $\boldsymbol{x}$ is the position vector. This modification can improve accuracy of the pressure solution.

The Boussinesq approximation solver is based on the PISO method, and the equations (6) and (10) are solved as the basic equations. Hereafter, the numerical solution method is explained.

The terms related to velocity in (10) are discretized. The equation related to the velocity component u_i is expressed below.

$$A_{\mathrm{P}} u_{i,\mathrm{P}} + \sum_{\mathrm{N}} A_{\mathrm{N}} u_{i,\mathrm{N}} = -\left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}} - (\boldsymbol{g} \cdot \boldsymbol{x}) \left(\frac{\delta \rho_k}{\delta x_i}\right)_{\mathrm{P}} \tag{11}$$

where the suffix i shows the i th component, and $\delta / \delta x_i$ is the discretized operator of the space derivative. A is a coefficient, and its value varies depending on cell position and time step. ρ_k also varies depending on cell position and time step. The suffixes P and N indicate the focused cell and the neighboring cells, respectively. p_{rgh} is equal to $p - \rho_k \boldsymbol{g} \cdot \boldsymbol{x}$. (11) is modified as below.

$$u_{i,\mathrm{P}} = \frac{H}{A_{\mathrm{P}}} - \frac{1}{A_{\mathrm{P}}} \left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}} - \frac{1}{A_{\mathrm{P}}} (\boldsymbol{g} \cdot \boldsymbol{x}) \left(\frac{\delta \rho_k}{\delta x_i}\right)_{\mathrm{P}} \tag{12}$$

where $H = -\sum_{\mathrm{N}} A_{\mathrm{N}} u_{i,\mathrm{N}}$. From (12) and (6), the discretized Poisson equation of pressure expressed as (13) is obtained.

$$\frac{\delta}{\delta x_i} \left[\frac{1}{A_{\mathrm{P}}} \left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}}\right] = \frac{\delta}{\delta x_i} \left[\frac{H}{A_{\mathrm{P}}} - \frac{1}{A_{\mathrm{P}}} (\boldsymbol{g} \cdot \boldsymbol{x}) \left(\frac{\delta \rho_k}{\delta x_i}\right)_{\mathrm{P}}\right] \tag{13}$$

Here, the Einstein notation is applied. The suffix i appears twice in the same term, which means the summation of each component.

In OpenFOAM, predicted velocity u_i^* is calculated by using the values of the previous iteration in (12). Predicted temperature T^* is calculated by (8). Then, $\tilde{u}_{i,\mathrm{P}} = H/A_{\mathrm{P}}$ is obtained by using the predicted velocity u_i^*. Pressure is then calculated by (13) with $\tilde{u}_i$ and T^*. After pressure is determined, corrected velocity is obtained by using (12). These calculations are iterated until a convergent solution is obtained.

3.2. Modification of solver

According to our preliminary analysis, the influence of mesh non-orthogonality on numerical solution of jet flows is hardly observed when buoyancy is non-existent. This point is discussed in Section 5 in detail. Thus, we focus on the buoyancy term included in the equations of the solution for buoyant flows.

On the right hand side of (13), the Laplacian of density is included. The buoyant force term can strongly affect buoyant flow. Therefore, it is necessary to calculate the Laplacian of density precisely. In the case that the maximum mesh non-orthogonality is higher than, for example, 5 degrees, the computation accuracy of this Laplacian term may decrease (OpenFOAM, 2016). Non-orthogonal correction for the Laplacian term has been implemented in OpenFOAM, as shown in Section 2. However, non-orthogonal correction may increase the computation time. A solver that can reduce the influence of mesh non-orthogonality without increasing computation time is desired. We show that the modified solver eliminates the influence of the Laplacian of density while solving the Poisson equation of pressure more accurately even when a mesh with the maximum mesh non-orthogonality greater than 5 degrees is employed.

The reference position and dynamic pressure are defined as $\boldsymbol{x}_{ref}$ and p'. Pressure p is rewritten as below.

$$p = p' + \rho_{ref}\boldsymbol{g} \cdot (\boldsymbol{x} - \boldsymbol{x}_{ref}) \tag{14}$$

$$\nabla p = \nabla p' + \rho_{ref}\boldsymbol{g} \tag{15}$$

From (15) and (7), (16) is obtained.

$$\rho_{ref}\left(\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u})\right) = -\nabla p' + \mu\nabla^2\boldsymbol{u} + (\rho - \rho_{ref})\boldsymbol{g} \tag{16}$$

To this equation, the Boussinesq approximation is applied.

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = -\nabla p' + \nu\nabla^2\boldsymbol{u} - \beta(T - T_{ref})\boldsymbol{g} \tag{17}$$

where $p' \equiv p'/\rho_{ref}$. In a similar way as the described formulation, the new discretized Poisson equation for dynamic pressure is obtained in (18).

$$\frac{\delta}{\delta x_i}\left[\frac{1}{A_{\mathrm{P}}}\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}\right] = \frac{\delta}{\delta x_i}\left[\frac{(\rho_k g_i)_{\mathrm{P}} - \sum_{\mathrm{N}} A_{\mathrm{N}} u_{i,\mathrm{N}}}{A_{\mathrm{P}}}\right] \tag{18}$$

where $\rho_k = -\beta(T - T_{ref})$. In this equation, the Laplacian term of density is not included. We construct a new solver by using this pressure equation, and compare the numerical solutions with those obtained by using the existing solver in OpenFOAM.

4. Numerical solver using low Mach number weakly compressible formulation

The Boussinesq approximation cannot be applied when the change in density is large. Then, the low Mach number weakly compressible formulation by Zhou et al. (2001) is applied when the change in density is large and the characteristic speed is very small compared to the acoustic speed. Here, the solver using this formulation "buoyantPimpleFoam" is discussed. The summary of this solver is explained by Kumar and Dewan (2014).

4.1. Numerical solution method of "buoyantPimpleFoam" solver in OpenFOAM

The basic equations are the mass, momentum and energy equations (19)–(21), and the equation of state of the fluid.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{19}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \frac{1}{3}\mu \nabla D + \rho \boldsymbol{g} \tag{20}$$

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho h \boldsymbol{u}) = \nabla \cdot \left(\frac{\mu}{Pr} \nabla h \right) + \frac{\partial p}{\partial t} \tag{21}$$

where h is enthalpy of the fluid and D is $D = \nabla \cdot \boldsymbol{u}$. Pr is Prandtl number. In a similar way as (10), (20) is modified as (22).

$$\begin{aligned} \frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = & -\nabla (p - \rho \boldsymbol{g} \cdot \boldsymbol{x}) + \mu \nabla^2 \boldsymbol{u} \\ & + \frac{1}{3}\mu \nabla D - (\boldsymbol{g} \cdot \boldsymbol{x}) \nabla \rho \end{aligned} \tag{22}$$

The solver using the low Mach number weakly compressible formulation is also based on the PISO method. The terms related to velocity are discretized.

$$\frac{A_{\mathrm{P}}}{\rho_{\mathrm{P}}}\rho_{\mathrm{P}}u_{i,\mathrm{P}} + \sum_{\mathrm{N}} A_{\mathrm{N}}u_{i,\mathrm{N}} = -\left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}} - (\boldsymbol{g}\cdot\boldsymbol{x})\left(\frac{\delta \rho}{\delta x_i}\right)_{\mathrm{P}}$$
$$\Leftrightarrow \rho_{\mathrm{P}}u_{i,\mathrm{P}} = \rho_{\mathrm{P}}\frac{H}{A_{\mathrm{P}}} - \frac{\rho_{\mathrm{P}}}{A_{\mathrm{P}}}\left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}} - \frac{\rho_{\mathrm{P}}}{A_{\mathrm{P}}}(\boldsymbol{g}\cdot\boldsymbol{x})\left(\frac{\delta \rho}{\delta x_i}\right)_{\mathrm{P}} \tag{23}$$

where $H = -\sum_{\mathrm{N}} A_{\mathrm{N}}u_{i,\mathrm{N}}$ and $p_{rgh} = p - \rho\boldsymbol{g}\cdot\boldsymbol{x}$. A is a coefficient and it is variable in the same manner as (11). From (19) and (23), the discretized pressure equation (24) is obtained.

$$\frac{\delta}{\delta x_i}\left[\frac{\rho_{\mathrm{P}}}{A_P}\left(\frac{\delta p_{rgh}}{\delta x_i}\right)_{\mathrm{P}}\right] = \frac{\delta}{\delta x_i}\left[\rho_{\mathrm{P}}\frac{H}{A_{\mathrm{P}}} - \frac{\rho_{\mathrm{P}}}{A_{\mathrm{P}}}(\boldsymbol{g}\cdot\boldsymbol{x})\left(\frac{\delta \rho}{\delta x_i}\right)_{\mathrm{P}}\right] + \frac{\partial \rho_{\mathrm{P}}}{\partial t} \tag{24}$$

Predicted velocity u_i^* is calculated by using the values of the previous iteration in (23). Predicted enthalpy h^* is calculated by using (21), and predicted density ρ^* is calculated with the equation of state of a gas. $\tilde{u}_{i,\mathrm{P}} = H/A_{\mathrm{P}}$ is calculated by using the predicted velocity u_i^*. Pressure is obtained by using $\tilde{u}_i$, ρ^* and the known values of density in (24). After pressure is obtained, corrected velocity is calculated by using (23). The calculations are iterated until a convergent solution is obtained. In addition, density is corrected by the equation of state.

4.2. Modification of solver

On the right hand side of (24), the Laplacian of density is also included in the solver based on the low Mach number weakly compressible formulation. Here, we also consider elimination of the Laplacian of density from the pressure equation. Pressure p is expressed in terms of dynamic pressure p' and spatially homogeneous density ρ_c.

$$p = p' + \rho_c\boldsymbol{g}\cdot\boldsymbol{x} \tag{25}$$

The value of ρ_c relates to the boundary condition of pressure at inlet. Considering the sign of the pressure gradient at the inlet, it is necessary to determine the value of ρ_c. The minimum value of density of the fluid is used as the value

of ρ_c in this study. Detail pertaining to the determination of ρ_c is discussed in Section 6. (25) is inserted into (20).

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p' + \mu \nabla^2 \boldsymbol{u} + \frac{1}{3}\mu \nabla D + (\rho - \rho_c)\boldsymbol{g} \tag{26}$$

The velocity term is discretized in the above equation.

$$\begin{aligned} &\frac{A_{\mathrm{P}}}{\rho_{\mathrm{P}}}\rho_{\mathrm{P}} u_{i,\mathrm{P}} + \sum_{\mathrm{N}} A_{\mathrm{N}} u_{i,\mathrm{N}} = -\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}} + (\rho_{\mathrm{P}} - \rho_c) g_i \\ &\Leftrightarrow \rho_{\mathrm{P}} u_{i,\mathrm{P}} = \rho_{\mathrm{P}} \frac{H}{A_{\mathrm{P}}} - \frac{\rho_{\mathrm{P}}}{A_{\mathrm{P}}} \left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}} \end{aligned} \tag{27}$$

where $H = (\rho_{\mathrm{P}} - \rho_c) g_i - \sum_{\mathrm{N}} A_{\mathrm{N}} u_{i,\mathrm{N}}$. From this equation, a new discretized pressure equation expressed as (28) is obtained. We construct a new solver using this pressure equation.

$$\frac{\delta}{\delta x_i}\left[\frac{\rho_{\mathrm{P}}}{A_{\mathrm{P}}}\left(\frac{\delta p'}{\delta x_i}\right)_{\mathrm{P}}\right] = \frac{\delta}{\delta x_i}\left[\rho_{\mathrm{P}} \frac{H}{A_{\mathrm{P}}}\right] + \frac{\partial \rho_{\mathrm{P}}}{\partial t} \tag{28}$$

In this study, OpenFOAM version 2.1.x is used. OpenFOAM version 4.0 has been released. The fundamental process of solution of buoyantPimpleFoam in version 4.0 is the same as that in version 2.1.x. Therefore, the above modification to the solver can be applied to the newer version of OpenFOAM. In the same manner, modification of buoyantBoussinesqPimpleFoam can be applied to the newer version of OpenFOAM.

5. Test analysis 1: buoyant jet with small temperature difference

5.1. Problem description and numerical simulation

The first test case is a vertical buoyant jet with a small temperature difference injecting into a 2D rectangular region. Figure 3 depicts a schematic sketch of the problem for test analysis. The x and y directions indicate the horizontal and the vertical directions, respectively. Gravitational acceleration $\boldsymbol{g}$ is along the $-y$ direction and $|\boldsymbol{g}| = 9.81\ \mathrm{m/s^2}$. The width and height of the rectangular region are $L = 7$ m and $H = 14$ m, respectively. The inlet and the outlets are located at the center and both sides of the container bottom, and the widths of the inlet and the outlets are $l = 1$ m.

Initial temperature T_0 was homogeneous, and its value was 300 K. Initial state was quiescent. The fluid was injected homogeneously. Injection velocity was $U_{in} = 0.1$ m/s, and temperature of the injected fluid was $T_{in} = 310$ K. The kinematic viscosity coefficient ν was set to $\nu = 10^{-4}$ m^2/s. The thermal expansion coefficient β was set to $\beta = 3 \times 10^{-3}$ K^{-1}. Prandtl number was 0.7. The fluid was assumed to be an ideal gas. Reynolds number defined by the injection velocity and the inlet width was $Re = U_{in} l/\nu = 1000$. Richardson number Ri was defined by the temperature difference $\Delta T = T_{in} - T_0$, the injection velocity and the region height, as expressed below.

$$Ri = \frac{-g\beta\Delta T H}{U_{in}^2} \approx -410 \tag{29}$$

The absolute value of Ri was much higher than unity, and its sign was negative. This flow was dominated by buoyancy, and it was unstable.

Non-slip and adiabatic conditions were applied on the wall as the boundary conditions. Considering buoyant force, the pressure boundary condition on the wall was $\nabla p \cdot \boldsymbol{n} = \rho_k \boldsymbol{g} \cdot \boldsymbol{n}$, where $\boldsymbol{n}$ was a unit vector normal to the wall. At the inlet boundary, velocity was equal to U_{in}, and the boundary condition of pressure was $\nabla p \cdot \boldsymbol{n} = \rho_k \boldsymbol{g} \cdot \boldsymbol{n}$. The pressure was fixed, and the velocity gradient was fixed to zero at the outlet boundary.

The numerical simulations were performed by using the solver based on the Boussinesq approximation because the temperature difference in this test problem was small. No turbulence model was applied. In this study, we focused on the influence of mesh non-orthogonality rather than that of the numerical scheme. Therefore, we applied the numerical schemes commonly used for simulation in OpenFOAM (OpenFOAM, 2016). The Euler implicit method was employed for time marching. The gradient term was discretized by using the Gauss linear scheme. For the gradient term, standard finite volume discretization was used for the integrated term with the Gauss divergence theorem, which is based on summing values on cell faces and requires interpolation of the values from the cell centers to the face center. We employed linear interpolation (central differencing). The divergent terms of velocity and temperature equations

were discretized by using the Gauss limited linear scheme. For the divergent terms, standard finite volume discretization was used, and interpolation was performed by using the limited linear scheme. The limited linear scheme is one of the Total Variation Diminishing (TVD) schemes implemented in OpenFOAM (The Open CAE Society of Japan, 2016), and the flux limiter function is defined as $\psi(r) = \max(0, \min(\frac{2}{k}, 1))$, where r is the consecutive gradient and k is the input parameter ($0 \leq k \leq 1$). In this simulation, $k = 1$ was applied. Here, application of the first order upwind scheme was avoid to decrease the influence of numerical viscosity. The linear interpolation scheme was used to interpolate the other terms.

The iteration number of the pressure corrector loop was set to 2, and the time step was controlled to keep the maximum Courant number ≤ 0.3. We also tested cases in which the iteration number of the pressure corrector loop was set to 3 and the value of the maximum Courant number was less than 0.15, and we obtained almost the same results as the results obtained using the original parameters. In addition, a limited scheme for the gradient term was tested. However, a difference from the result obtained using the original gradient scheme (Gauss linear scheme) was not observed.

For 2D simulations using OpenFOAM, the mesh was constructed from multiple cells in the width and height directions and one cell in the depth direction. Then, the actual created meshes with rectangular elements and triangular elements were composed of hexahedral and prismatic elements with one cell in the depth direction (z direction), respectively. Four types of meshes, namely, coarse hexahedral mesh, coarse prismatic mesh, fine hexahedral mesh, and fine prismatic mesh were created by using GAMBIT developed by ANSYS and Gmsh (Geuzaine and Remacle, 2009). Typical cell size of the coarse meshes was 0.02m, and that of the fine meshes was 0.01m. In the prismatic meshes, the hexahedral cells were constructed on the boundaries to avoid flow disturbance at the inlet. Pictures of the coarse meshes around the inlet are shown in Figs. 4 and 5. The numbers of mesh cells and the values of mesh non-orthogonality are given in Table 1. To check mesh quality, the values of non-orthogonality were

calculated with the utility code implemented in OpenFOAM. The value of non-orthogonality is defined as the angle between the line connecting two cell centers and the normal of their common face. A non-orthogonality of 0.0 means the mesh is perfectly orthogonal.

Table 1: Mesh characteristics

Mesh type	Number of cells	Non-orthogonality	
		maximum	average
coarse hexahedral mesh	245k (350×700)	0.0	0.0
coarse prismatic mesh	545k	12.8	3.0
fine hexahedral mesh	980k (700×1400)	0.0	0.0
fine prismatic mesh	2570k	41.2	7.1

5.2. Results and discussion

First, numerical results pertaining to non-buoyant jet flows ($g = 0$) are discussed to investigate the influence of buoyancy. Temperature fields obtained using the coarse hexahedral mesh and the coarse prismatic mesh without non-orthogonal correction (NO correction) at 200 s from the start of jet injection are shown in Fig. 6 and 7, respectively. The unit of temperature is Kelvin (K). The inlet conditions are the same as the conditions shown in 5.1. Although NO correction was not applied, no difference in the temperature fields was observed. Qualitatively the same tendency was observed even when the injection velocity was increased to 13 times of the original injection velocity.

Second, we discuss the numerical results pertaining to buoyant jet flows. Figure 8 shows the temperature distributions obtained using the coarse hexahedral mesh at 10 s and 20 s from start of jet injection. This simulation was performed by using the existing solver buoyantBoussinesqPimpleFoam. Symmetrical temperature distribution is observed at 20 s from the start of jet injection. Around downstream of the jet edge, symmetrical vortices are observed. We compared the results obtained using the fine hexahedral mesh with those obtained using

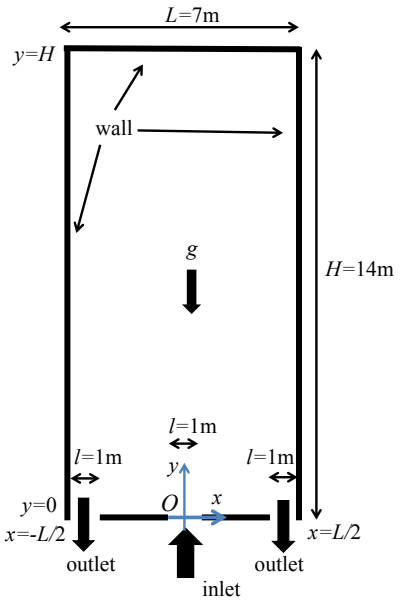

Figure 3: Schematic sketch of geometry of buoyant jet with small temperature difference

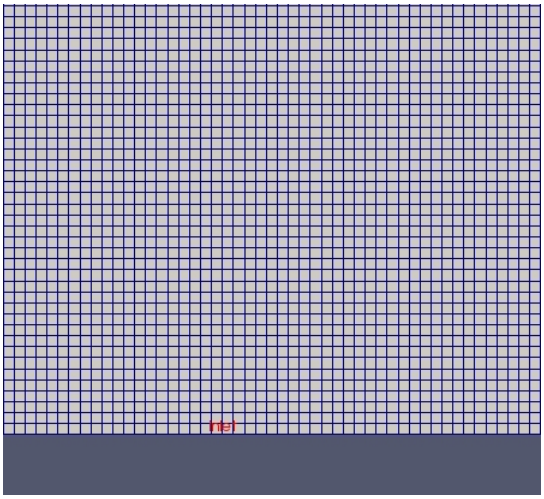

Figure 4: Coarse hexahedral mesh around inlet

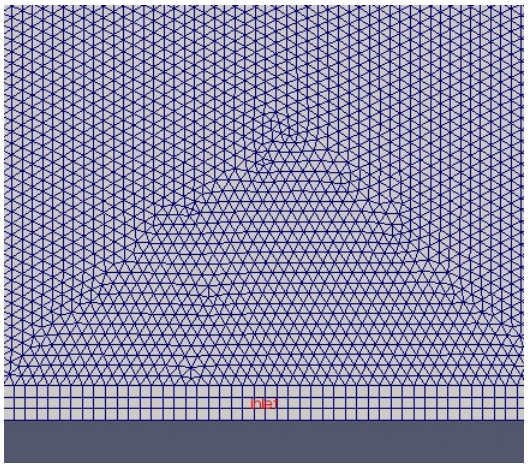

Figure 5: Coarse prismatic mesh around inlet

the coarse hexahedral mesh. Figure 9 shows the temperature distributions simulated using the fine hexahedral mesh. The temperature distributions obtained with the coarse hexahedral mesh by using the existing solver shows good agreement with those obtained with the fine hexahedral mesh. The coarse hexahedral mesh has enough mesh resolution in this simulation.

Here, we discuss the numerical results obtained with the prismatic meshes. The calculation conditions, used solver, and the number of applications of NO correction in the cases using the coarse prismatic mesh are summarized in Table 2. These simulations were performed by using the existing solver buoyantBoussinesqPimpleFoam and the modified solver. Figure 10 shows closeup views of the temperature fields obtained with the coarse prismatic mesh. The temperature fields obtained by using the existing solver without NO correction (condition 1a) in Fig.10(1a) shows asymmetrical temperature distribution, and the flow field is more unstable than that obtained using the coarse hexahedral mesh. This result indicates that the simulation using the prismatic mesh without NO correction may overestimate flow instability compared to the simulation using the hexahedral mesh. Moreover, the result obtained using the fine prismatic mesh without NO correction shows more disturbed distribution than that obtained

using the coarse prismatic mesh. This may be ascribed to the higher mesh non-orthogonality of the fine prismatic mesh than that of the coarse prismatic mesh.

Table 2: Calculation conditions using coarse prismatic mesh

Condition	Solver	Number of applications of NO correction
1a	existing solver	none
1b	existing solver	1
1c	existing solver	2
1d	modified solver	none
1e	modified solver	1

Figures 10(1b) and (1c) show the temperature distributions obtained using the coarse prismatic mesh by the existing solver with one and two applications of NO correction (conditions 1b and 1c), respectively. NO correction should be applied multiple times, as described in Section 2. Then, we discuss the number of applications of NO correction. Asymmetrical temperature distributions are observed under condition 1b. By contrast, the results obtained under condition 1c show good agreement with the results obtained using the hexahedral mesh. These findings indicate that mesh non-orthogonality can strongly influence numerical solution of buoyant jets, and the numerical results are corrected appropriately by NO correction even when the prismatic mesh is used.

Figures 10(1d) and (1e) show the temperature distributions calculated with the modified solver using the coarse prismatic mesh without NO correction and those obtained with one application of NO correction (conditions 1d and 1e), respectively. The temperature distributions obtained under condition 1d show more symmetrical distributions than those obtained by using the existing solver without NO correction, and the results obtained under condition 1e show good agreement with those obtained using the coarse hexahedral mesh. The solution of buoyant flow based on the PISO method is affected strongly by

the Laplacian of density included in the pressure equation. The influence of mesh non-orthogonality is reduced by using the modified solver, and the correct solution can be obtained by fewer applications of NO correction.

Vertical temperature distributions at the center of the inlet ($x = 0$ m) are plotted in Fig.11. The numerical results obtained using the coarse hexahedral and the coarse prismatic meshes are shown. The time is 20 s from the start of jet injection. The results obtained under conditions 1c and 1e show good agreement with the results obtained using the coarse hexahedral mesh. Large differences are observed between the results obtained using the hexahedral mesh and the other results obtained using the coarse prismatic mesh (conditions 1a, 1b, and 1d).

Table 3 shows the averaged computation times (wall clock times) and standard deviations of the computation time in simulating flow fields until 20 s from the beginning of jet injection with the coarse prismatic mesh. The computation times under conditions 1a–1e are compared. For these calculations, 2 Intel Xeon E5-2690v2 3.0GHz CPUs with 10 cores/CPU were used, and the number of parallel calculations was 20. The OpenFOAM code was built by the Intel compiler bundled with C++ version 14.0.0. Intel MPI version 4.1.1.036 was used for the parallel computations. The computation times were measured eight times for each solver. Large differences in computation times were not observed between conditions (1a) and (1b) and between (1d) and (1e). The computation times under conditions (1b) and (1c) were almost comparable, even after increasing the number of applications of NO correction. Although the number of times that the pressure equation was solved was increased by the NO correction, the total computation time might not increase considerably because the time taken to solve the pressure equation once might decrease. Because the standard deviations in computation time were relatively large compared with difference between the averaged computation times, the cases that the calculation under the condition (1b) was much faster than that under the condition (1c) were observed. The averaged computation time with the modified solver (1e) was about 14% longer than that with the existing solver (1c). This result indicates

that further improvement of the solver based on the Boussinesq approximation is necessary.

Table 3: Comparison of computation times

Solver (condition)	Computation time [s]	Standard deviation [s]
existing solver (1a)	3010.0	195.1
existing solver (1b)	3022.5	218.0
existing solver (1c)	3050.8	11.0
modified solver (1d)	3645.4	118.3
modified solver (1e)	3587.9	187.7

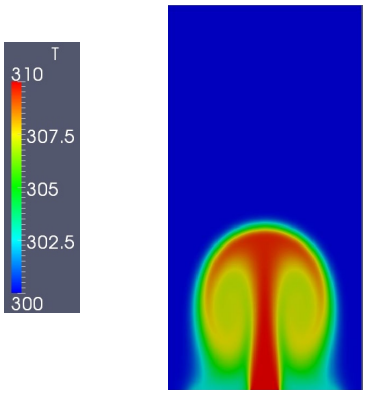

Figure 6: Temperature field of jet $g = 0$ obtained using coarse hexahedral mesh without non-orthogonal correction. The field at 200s from the start of jet injection is shown.

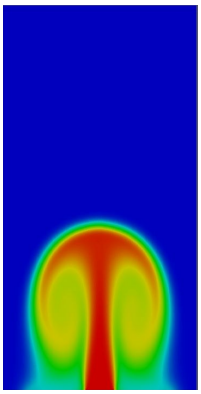

Figure 7: Temperature field of jet $g = 0$ obtained using coarse prismatic mesh without non-orthogonal correction. The field at 200s from the start of jet injection is shown.

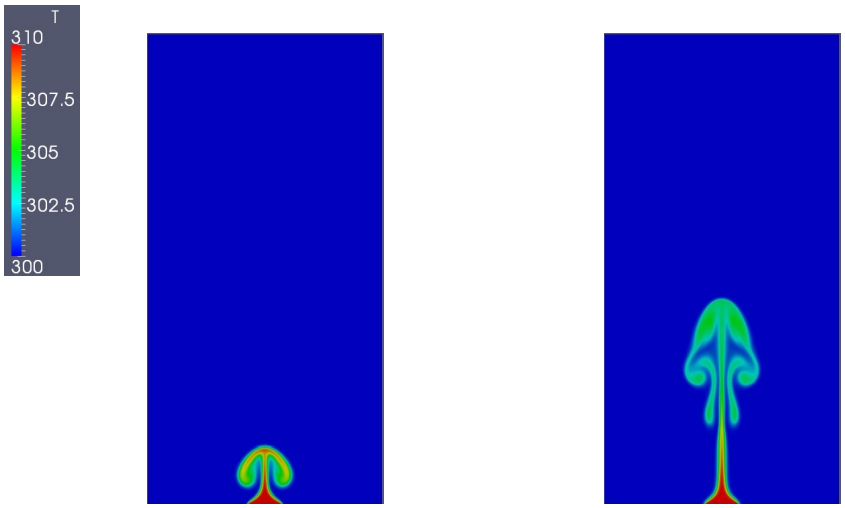

Figure 8: Temperature field obtained using coarse hexahedral mesh by existing solver without non-orthogonal correction. Left: field at 10s from the start of jet injection, right: field at 20s from the start of jet injection.

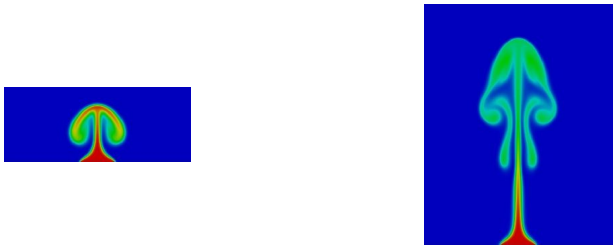

Figure 9: Closeup view of temperature field obtained using fine hexahedral mesh by existing solver without non-orthogonal correction. Left: field at 10s from the start of jet injection, right: field at 20s from the start of jet injection. The color scale is the same as that in Fig.8.

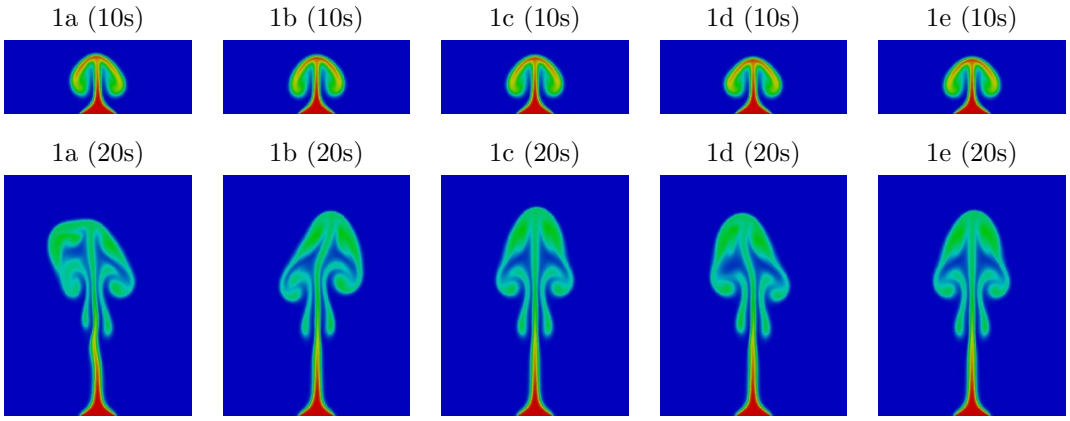

Figure 10: Closeup views of temperature fields obtained with coarse prismatic mesh under conditions (1a)-(1e). Upper: fields at 10s from the start of jet injection, lower: fields at 20s from the start of jet injection. The color scale is the same as that in Fig.8.

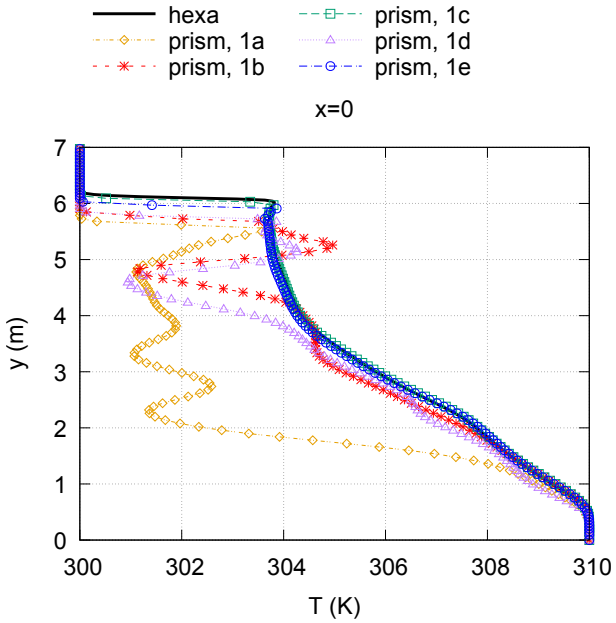

Figure 11: Vertical distributions of temperature at center of inlet ($x = 0$ m). The time is 20s from the start of injection. Solid line: result obtained with coarse hexahedral mesh, dashed lines and symbols: results obtained with coarse prismatic mesh (1a)–(1e).

6. Test analysis 2: buoyant jet with large temperature difference

6.1. Problem description and numerical simulation

Buoyant jets simulated by Beccantini et al. (2008) have been analyzed as a test for buoyant flows with large temperature difference. They simulated a vertical buoyant jet injecting into a 2D rectangular region. A schematic sketch of the geometry is shown in Fig.12. Definitions of the axes are the same as those in test analysis 1 in Section 5. The width and height of the region are $L = 3$ m and $H = 7$ m, respectively. The inlet is located at the center of the container bottom, and its width is $l = 0.2$ m. This container has no outlet. The origin of the coordinate system is located at the center of the inlet.

The initial condition was the same as that in test analysis 1. Initial temperature was $T_0 = 300$ K and initial pressure was $p_0 = 10^5$ Pa. Injection temperature was $T_{in} = 600$ K. Injection velocity U_{in} was given by a parabolic profile $U_{in} = 6U_m(l^2/4 - x^2)/l^2$. The injection mass flow rate was $\dot{m}_{in} = 1.0$ kg/m^2·s, and the velocity U_m at the center was about 1.7 m/s. Reynolds number Re was defined by the injection mass flow rate, injection width and viscosity coefficient: $Re = \dot{m}_{in} l/\mu$. Re by Beccantini et al. was equal to 40. The viscosity coefficient was $\mu = 5 \times 10^{-3}$ Pa·s in this simulation. Richardson number was estimated to be $Ri \approx -24$. Temperature dependency of material properties was not considered by Beccantini et al. Therefore, material properties were assumed to be constant in this simulation.

The temperature difference in this test problem was large. Therefore, the numerical solver based on the low Mach number weakly compressible formulation was applied. Non-slip and adiabatic conditions were applied as the boundary conditions on the wall. Considering buoyant force, the pressure boundary condition on the wall was $\nabla p \cdot \boldsymbol{n} = \rho \boldsymbol{g} \cdot \boldsymbol{n}$. At the inlet boundary, the velocity was fixed to U_{in} and the boundary condition of pressure was $\nabla p \cdot \boldsymbol{n} = \rho \boldsymbol{g} \cdot \boldsymbol{n}$. This boundary condition was rewritten by using the dynamic pressure p' in (25) of the modified solver, and $\nabla p' \cdot \boldsymbol{n} = (\rho - \rho_c)\boldsymbol{g} \cdot \boldsymbol{n}$. In this analysis, the inlet was located on the bottom of the container. Then, the boundary condition on p'

was obtained as $\frac{\partial p'}{\partial y} = -(\rho - \rho_c)|\boldsymbol{g}|$. It is necessary to satisfy $\rho - \rho_c \geq 0$ so that the gradient of p' is zero or negative along the flow direction for smooth injection. Therefore, the minimum value of density, namely, density at the maximum temperature position, was used as ρ_c in this study.

In this simulation, no turbulence model was applied. We also applied the commonly used numerical schemes in the same way as that in test analysis 1. The Euler implicit method was used for time marching method. The gradient term was discretized by the Gauss linear scheme. Standard finite volume discretization was used, and the gradient term was interpolated by using the linear interpolation scheme. The divergent terms in the velocity and the enthalpy equations were discretized by using the Gauss linear scheme. Standard finite volume discretization was used, and the divergent terms were interpolated by using the linear scheme. The linear interpolation scheme was used for interpolating the other terms.

The iteration number of the pressure corrector loop was set to 2, and the time step was controlled to keep the maximum Courant number ≤ 0.3.

Beccantini et al. used a hexahedral mesh with 120×120 cells. In this simulation, the same mesh size was used. A prismatic mesh was also created, and the characteristic cell size was 3m/120=0.025m. The hexahedral cells were constructed on the boundaries of the prismatic mesh in the same way as that in test analysis 1. The numbers of mesh cells and the values of mesh non-orthogonality are given in Table 4.

Table 4: Mesh characteristics for buoyant jet with large temperature difference

Mesh type	Number of cells	Non-orthogonality	
		maximum	average
hexahedral mesh	14.4k (120×120)	0.0	0.0
prismatic mesh	72.8k	12.1	3.2

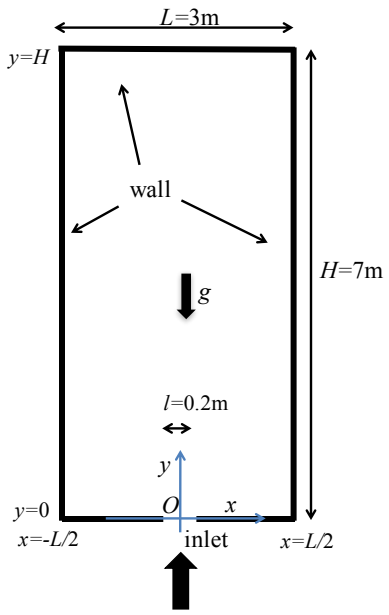

Figure 12: Schematic sketch of geometry of buoyant jet with large temperature difference

6.2. Results and discussion

Temperature distributions and contours after 6 s from the start of jet injection are shown in Figs. 13 and 14. These figures show the results obtained with the hexahedral mesh by the existing solver buoyantPimpleFoam and by the modified solver, respectively. The temperature range is 329–582 K. Increments of the contours are 18 K. The results obtained in these simulations are very similar and show good agreement with the results of Beccantini et al. The solutions obtained by using the existing solver and the modified solver are valid for buoyant jet flows with large temperature difference.

The calculation conditions associated with the prismatic mesh are given in Table 5. The temperature contour obtained by the existing solver without NO correction (condition 2a) is shown in Fig.15. Disturbed temperature distribution is observed, and the jet bends to large extent. The temperature contour obtained by the existing solver with one application of NO correction (condition 2b) is shown in Fig.16. It shows almost symmetrical profile owing to NO correction.

Table 5: Calculation conditions with prismatic mesh

Condition	Solver	Number of applications of NO correction
2a	existing solver	none
2b	existing solver	1
2c	modified solver	none
2d	modified solver	1

The temperature field obtained by the modified solver without NO correction (condition 2c) is shown in Fig.17. The numerical result is considerably better than the result obtained by the existing solver without NO correction, as shown in Fig.15, but it becomes slightly disturbed and shows asymmetrical distribution. The temperature field obtained by the modified solver with one application of NO correction (condition 2d) is shown in Fig.18. The result is improved by the application of NO correction.

Figures 19 and 20 show the vertical distributions of vertical velocity u_y at $x = 0$ and $-L/4$. The symbols show the results of the simulation by Beccantini et al. The red and green lines show the results obtained with the hexahedral mesh by the modified solver and the existing solver, respectively. The orange and blue lines show the results obtained with the prismatic mesh by the modified solver and the existing solver with one application of NO correction, respectively. The vertical velocity distributions obtained using the hexahedral mesh and the prismatic mesh show good agreement with the results of Beccantini et al.

A comparison of the computation times and the standard deviations of the computation times for different parallel numbers is given in Table 6. The computation times under conditions 2b and 2d are compared. The computation environment was the same as that in the previous section. The calculation times were measured three times for each case. The calculations by the modified solver were about 20–30% faster than those by the existing solver for each parallel number. This might be ascribe to improvement of convergence of the

pressure equation. Moreover, these calculations show relatively good parallel efficiency. More detailed analysis of the computation speed of these solvers is necessary, for example, using a larger mesh.

We also analyzed 3D buoyant jet flows with large temperature difference using tetrahedral mesh roughly. The same tendency as the numerical solutions of 2D buoyant jet flows was observed in terms of the influence of non-orthogonal correction. In the future, more detailed investigation of the influence on 3D buoyant flows is necessary.

Table 6: Comparison of computation times and standard deviations for different parallel numbers

Parallel number	Existing solver (2b)		Modified solver (2d)	
	Computation time [sec]	Standard deviation [sec]	Computation time [sec]	Standard deviation [sec]
1	3293.7	203.9	2493.0	59.8
10	426.0	1.4	336.3	5.7
20	200.7	2.5	168.3	0.9

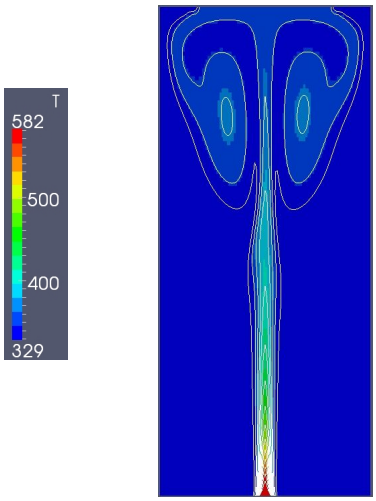

Figure 13: Temperature contour and field obtained using hexahedral mesh by existing solver without non-orthogonal correction

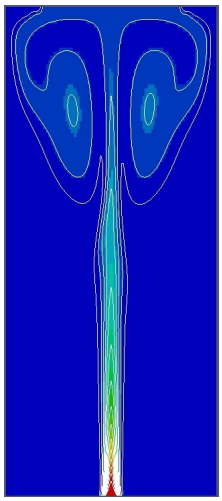

Figure 14: Temperature contour and field obtained using hexahedral mesh by modified solver without non-orthogonal correction

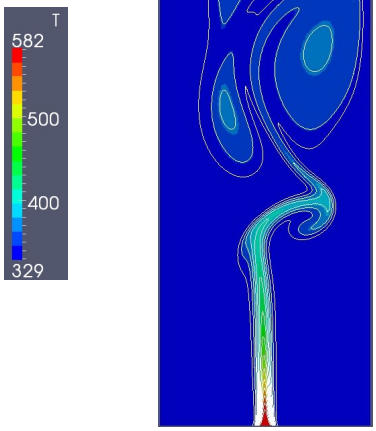

Figure 15: Temperature contour and field obtained using prismatic mesh by existing solver without non-orthogonal correction (condition 2a)

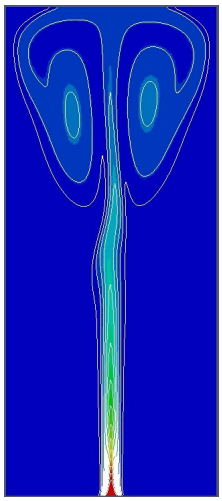

Figure 16: Temperature contour and field obtained using prismatic mesh by existing solver with one application of non-orthogonal correction (condition 2b)

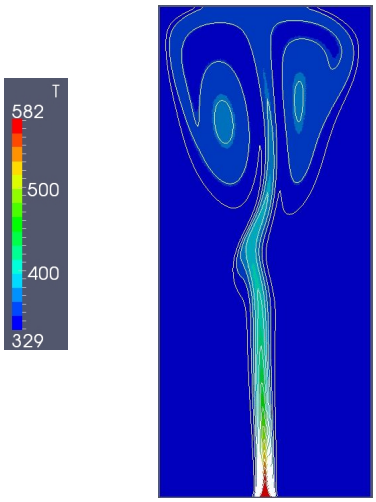

Figure 17: Temperature contour and field obtained using prismatic mesh by modified solver without non-orthogonal correction (condition 2c)

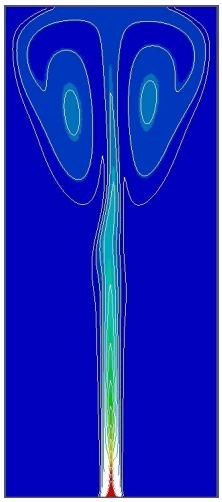

Figure 18: Temperature contour and field obtained using prismatic mesh by modified solver with one application of non-orthogonal correction (condition 2d)

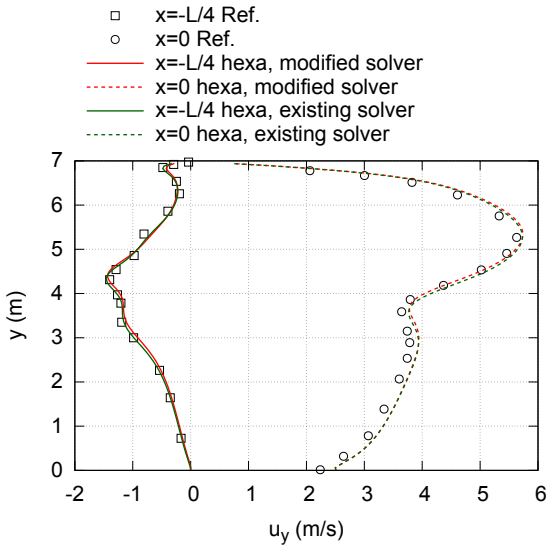

Figure 19: Vertical distributions of vertical velocity obtained using hexahedral mesh

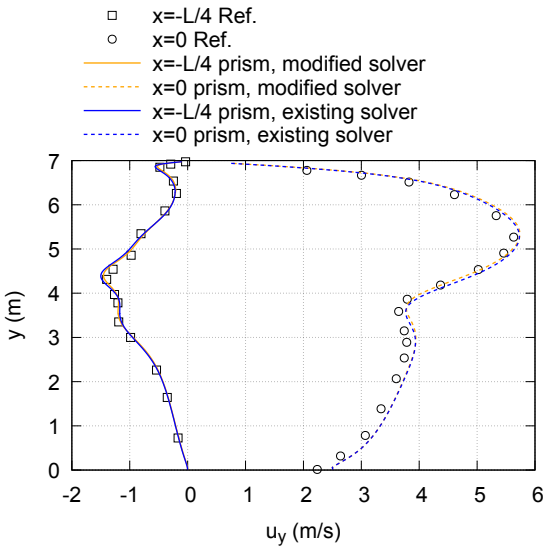

Figure 20: Vertical distributions of vertical velocity obtained using prismatic mesh with one application of non-orthogonal correction (conditions 2b and 2d)

7. Conclusion

The influence of mesh non-orthogonality on the numerical solutions of 2D buoyant jet flows with small and large temperature differences was analyzed by using OpenFOAM in this study. Buoyant jet was affected strongly by mesh non-orthogonality, while large influence occurred by the difference in mesh type was hardly observed in the case of non-buoyant jets. The numerical result obtained with the prismatic mesh indicated a tendency to overestimate flow instability compared to that obtained with the hexahedral mesh when non-orthogonal correction was not applied. For accurate numerical simulation of buoyant jets, it is necessary to focus on mesh non-orthogonality and non-orthogonal correction.

Modified solvers to reduce the influence of mesh non-orthogonality were developed based on the Boussinesq approximation and the low Mach number weakly compressible formulation. The influence of mesh non-orthogonality was relaxed by the modified solvers. The modified solvers could calculate flow fields with fewer or the same number of iterations of non-orthogonal correction as the existing solvers. Especially, calculations by the modified solver based on the low Mach number weakly compressible formulation were faster than those by the existing solver.

In this study, we analyzed 2D buoyant jet flows. In the future, we would discuss the detailed effect of 3D geometry on thermal flows, and analyze thermal flows by using different types of meshes, for instance, meshes with higher non-orthogonality or meshes with the same number of elements but different non-orthogonality (e.g., parallelogram-type meshes), and so on. We discussed the influence of mesh non-orthogonality by the non-orthogonal correction implemented in OpenFOAM (over-relaxed approach), and it is necessary to study alternative ways to correct the influence of mesh non-orthogonality. It would be necessary to study the influence of mesh characteristics other than mesh non-orthogonality, for instance, skewness or cell size for flows dominated by buoyancy.

Appendix A

Details of the computation procedure of the cell center in OpenFOAM were described by Moukalled et al. (2015). A brief summary of this procedure is given here. First, the location of the geometric center of the cell $\boldsymbol{x}_G$ is computed, and the cell element is decomposed into a number of polygonal pyramids. Figure 21 shows the cell element and the decomposed pyramid. Each polygonal pyramid is formed of a geometric center $\boldsymbol{x}_G$ as the apex and a polygonal face of the cell element as the base.

$$\boldsymbol{x}_G = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{x}_i \tag{30}$$

where k is the number of the apex of the cell element, and $\boldsymbol{x}_i$ is the position vector of the apex.

Second, the center of the decomposed pyramid $(\boldsymbol{x}_{CE})_{pyramid}$ is calculated. The vector connecting $\boldsymbol{x}_G$ and the center of the base face f: $(\boldsymbol{x}_{CE})_f$ is defined as $\boldsymbol{d}_{Gf} = (\boldsymbol{x}_{CE})_f - \boldsymbol{x}_G$. The center of the face f: $(\boldsymbol{x}_{CE})_f$ is calculated by decomposing the base face into a number of triangles. $(\boldsymbol{x}_{CE})_{pyramid}$ is calculated as below.

$$(\boldsymbol{x}_{CE})_{pyramid} = \frac{3(\boldsymbol{x}_{CE})_f + \boldsymbol{x}_G}{4} \tag{31}$$

The center of the cell element $(\boldsymbol{x}_{CE})_C$ is computed as the volume-weighted average of the centers of the decomposed pyramids.

$$V_{pyramid} = \frac{\boldsymbol{d}_{Gf} \cdot \boldsymbol{S}_f}{3},\ V_C = \sum_{sub-pyramids} V_{pyramid} \tag{32}$$

$$(\boldsymbol{x}_{CE})_C = \frac{\sum_{sub-pyramids} (\boldsymbol{x}_{CE})_{pyramid} V_{pyramid}}{V_C} \tag{33}$$

References

Abe, S., Ishigaki, M., Sibamoto, Y., Yonomoto, T., 2015. RANS analyses on erosion behavior of density stratification consisted of helium-air mixture gas by a low momentum vertical buoyant jet in the PANDA test facility, the third international benchmark exercise (IBE-3). Nucl. Eng. Des. 289, 231–239.

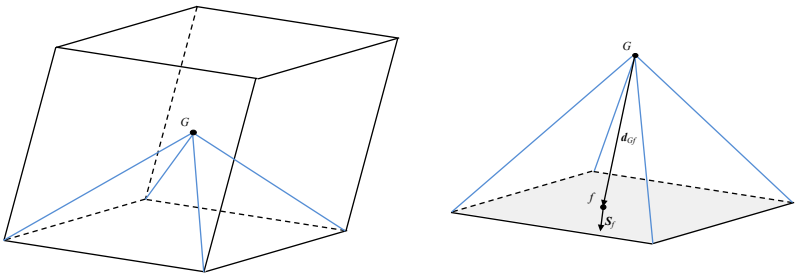

Figure 21: Cell element and decomposed pyramid.

Andreani, M., Badillo, A., Kapulla, R., 2014. Synthesis of the OECD/NEA-PSI CFD benchmark exercise, in: Proceedings of CFD4NRS-5 OECD/NEA & IAEA Workshop, Zurich, Switzerland.

Beccantini, A., Studer, E., Gounand, S., Magnaud, J.P., Kloczko, T., Corre, C., Kudriakov, S., 2008. Numerical simulations of a transient injection flow at low Mach number regime. J. Numer. Meth. Engng 76, 662–696.

Boivin, S., Cayré, F., Hérard, J.M., 2000. A finite volume method to solve the Navier-Stokes equations for incompressible flows on unstructured meshes. Int. J. Therm. Sci. 39, 806–825.

Dixit, H.N., Babu, V., 2006. Simulation of high Rayleigh number natural convection in a square cavity using the lattice Boltzmann method. Int. J. Heat Mass Transfer 49, 727–739.

Ferziger, J.H., Perić, M., 1997. Computational methods for fluid dynamics. Springer. 2nd edition.

Fuhrmann, J., Ohlberger, M., Rohde, C., 2014. Finite volumes for complex applications VII-elliptic, parabolic and hyperbolic problems: FVCA 7, Berlin, June 2014. Springer.

Geuzaine, C., Remacle, J.F., 2009. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Meth. Eng. 79, 1309–1331.

Github, 2011. Official OpenFOAM repository. https://github.com/OpenFOAM/OpenFOAM-2.1.x/tree/master/applications/solvers/heatTransfer/buoyantBoussinesqPimpleFoam, (accessed 29 September 2016).

Issa, R.I., 1985. Solution of the implicitly discretised fluid flow equations by operator-splitting. J. Comp. Phys. 62, 40–65.

Jasak, H., 1996. Error analysis and estimation for the finite volume method with applications to fluid flows. Ph.D. thesis. the University of London.

Kumar, R., Dewan, A., 2014. URANS computations with buoyancy corrected turbulence models for turbulent thermal plume. Int. J. Heat Mass Transfer 72, 680–689.

Moukalled, F., Mangani, L., Darwish, M., 2015. The finite volume method in computational fluid dynamics: An advanced introduction with OpenFOAM and Matlab. Springer.

Muzaferija, S., 1994. Adaptive finite volume method for flow prediction using unstructured meshes and multigrid approach. Ph.D. thesis. University of London.

Nam, S., Bill, Jr, R.G., 1993. Numerical simulation of thermal plumes. Fire Saf. J. 21, 231–256.

OECD/NEA, 2012. OECD/SETH-2 project - investigation of key issues for the simulation of thermal-hydraulic conditions in water reactor containment - final summary report. NEA/CSNI/R(2012)5.

Oliveira, P.J., Issa, R.I., 2001. An improved PISO algorithm for the computation buoyancy-driven flows. Numeric. Heat Transfer, Part B 40, 473–493.

OpenFOAM, 2004. http://www.openfoam.com, (accessed 17 May 2016).

OpenFOAM, 2016. OpenFOAM user guide version 4.0. https://raw.githubusercontent.com/OpenFOAM/OpenFOAM-dev/master/doc/Guides/OpenFOAMUserGuide-A4.pdf, (accessed 29 September 2016).

The Open CAE Society of Japan, 2016. Numerical heat transfer and fluid flow using OpenFOAM. Morikita Publishing. (in Japanese).

de Vahl Davis, G., 1983. Natural convection of air in a square cavity: a benchmark numerical solution. Int. J. Numer. Meth. Fluids 3, 249–264.

Versteeg, H.K., Malalasekera, W., 2007. An introduction to computational fluid dynamics: the finite volume method. Prentice Hall. 2nd edition.

Yan, Z., Holmstedt, G., 1999. A two-equation turbulence model and its application to a buoyant diffusion flame. Int. J. Heat Mass Transfer 42, 1305–1315.

Yonomoto, T., Sibamoto, Y., Ishigaki, M., Abe, S., 2015. The ROSA-SA project on containment thermal hydraulics. Presented at the International Experts Meeting on Strengthening Research and Development Effectiveness in the Light of the Accident at the Fukushima Daiichi Nuclear Power Plant. http://www-pub.iaea.org/iaeameetings/cn235p/Session3/S3-5-Taisuke-Yonomoto.pdf, (accessed 17 May 2016).

Zhou, X., Luo, K.H., Williams, J.J.R., 2001. Large-eddy simulation of a turbulent forced plume. Eur. J. Meth. B Fluids 20, 233–254.

Zitzmann, T., Malcolm, C., Pfrommer, P., Rees, S., Marjanovic, L., 2005. Simulation of steady-state natural convection using CFD, in: 9th International IBPSA Conference, pp. 1449–1456.

List of Figure captions

List of Table captions