vFoam: a new set of volume of fluid solvers for turbulent isothermal multiphase flows

Wenyuan Fan\textsuperscript{a,∗}, Henryk Anglart\textsuperscript{a,b}

\textsuperscript{a}Nuclear Engineering Division, Department of Physics, KTH Royal Institute of Technology, 106 91 Stockholm, Sweden
\textsuperscript{b}Institute of Heat Engineering, Warsaw University of Technology, 21/25 Nowowiejska Street, 00-665 Warsaw, Poland

Abstract

The volume of fluid (VOF) method is a popular approach for multiphase flow modeling. The open-source computational fluid dynamics (CFD) software, OpenFOAM, has implemented a variety of VOF-based solvers and provides users a wide range of turbulence models. Since flows under the VOF framework belong to the variable-density incompressible flow category, the isothermal VOF-based solvers in OpenFOAM fail to use the correct turbulence models. vFoam is designed to solve this issue and with the hope to replace all the corresponding existing solvers in the future. With the object-oriented paradigm, vFoam guarantees the usability, reusability and maintainability of the codes. Aside from turbulence modeling, all other features in the original solvers are preserved in vFoam.

Keywords: VOF, CFD, turbulence modeling, variable-density incompressible flow, OpenFOAM

1. Introduction

Multiphase flows, e.g. air-water flows in oceans and gas-oil flows in oil transfer lines, are often encountered in nature and in industrial applications. However, multiphase flow modeling
is challenging due to the existence of the moving interface between different phases. Among various modeling approaches, the volume of fluid (VOF) method [1] is a popular and widely adopted one due to the reasons that will be discussed in Section 2. As a matter of fact, many open-source two-phase computational fluid dynamics (CFD) codes, e.g. OpenFOAM [2] and Gerris [3], have implemented solvers based on VOF.

However, the VOF method also has its own drawbacks which are mostly caused by the existence of the transition zone from one phase to another. This transition zone introduces difficulties in getting a sharp interface and calculating the interface curvature accurately. Therefore, VOF-related investigations mainly focus on sharpening the interface and calculating the curvature accurately [4, 5, 6, 7]. Aside from these issues, the presence of the transition zone also causes problems for turbulent multiphase flow modeling when turbulence models are used. Since turbulence modeling itself is already complicated, the modeling of multiphase flow with turbulence is quite complex [8, 9]. As a result, VOF-related solvers in open-source CFD code struggle with significant difficulties when turbulence models are needed. For instance, the VOF solver in Gerris could not use turbulence models and the isothermal VOF-based solvers in OpenFOAM uses a turbulence modeling approach which is inconsistent with the nature of the VOF method.

In this paper, a new set of VOF based solvers is developed based on the OpenFOAM platform. In the newly designed solvers, corrected governing equations for turbulence quantities are used.

2. Overview of the VOF method

The VOF method [1] was first developed to model immiscible two-phase flows with a simple concept for interface advection:

$$\frac{\partial \alpha}{\partial t} + \vec{u} \cdot \nabla \alpha = 0,$$

where $\alpha$ is the volumetric fraction of the primary phase in a control volume (cell). $\alpha = 1$ means that the cell is entirely occupied by the primary phase, and $\alpha = 0$ implies that the cell is purely filled by the secondary phase. Eq. (1) also indicates that mass conservation is always guaranteed for the two-phase system, which makes it favorable for two-phase flow simulations. However, it is an additional treatment that makes VOF so popular. By substituting the density and viscosity in the single-phase governing equations with the mixture density:

$$\rho_m = \alpha \rho_1 + (1 - \alpha) \rho_2,$$

and the mixture viscosity:

$$\mu_m = \alpha \mu_1 + (1 - \alpha) \mu_2,$$

where subscript 1 and 2 denote the primary and secondary phase respectively, the resultant governing equations could be used to describe the two-phase system.

In order to illustrate an important character of the VOF, a fundamental definition in fluid mechanics, i.e. incompressible flow, is firstly introduced. A flow is incompressible if it satisfies:

$$\nabla \cdot \vec{u} = 0,$$

or the equivalent form:

$$\frac{\partial \rho}{\partial t} + \vec{u} \cdot \nabla \rho = 0.$$
Therefore, a flow with constant density is always incompressible since Eq. (5) is automatically satisfied. This type of flow is referred to as strict incompressible flow. However, for a flow with variable density, as long as Eq. (4) is fulfilled, the flow is still incompressible, and this type of flow is referred to as variable-density incompressible flow. The reason for defining this group of flows separately is that many governing equations could be simplified by using Eq. 4. Consequently, the computing overhead could be reduced.

For an isothermal immiscible two-phase flow system, the properties of each phase are usually assumed to be constant. Therefore the flow of an individual phase is incompressible. However, by introducing the mixture-property concept, the mixture property is changing with \( \alpha \). Therefore, such two-phase flows belong to variable-density incompressible flow category. This is an important concept which finally causes the issues with current VOF-based solvers in OpenFOAM.

3. Overview of turbulence modeling in OpenFOAM

The turbulence modeling capability is an undeniable outstanding feature of OpenFOAM. Both the Reynolds-Averaged Navier-Stokes (RANS) approach and Large Eddy Simulation (LES) are available for turbulence modeling. Plus, hybrid approaches, e.g. Detached Eddy Simulation (DES), Delayed Detached Eddy Simulation (DDES) and Improved Delayed Detach Eddy Simulation (IDDES), could also be used for turbulence modeling. Despite the diversity of such modeling approaches, the momentum equation could always be written as

\[
\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla p^* + \nabla \cdot \left[ (\mu + \mu_t) \left( \nabla \vec{u} + (\nabla \vec{u})^T - \frac{2}{3}(\nabla \cdot \vec{u}) I \right) \right] + \vec{F}_b, \tag{6}
\]

where \( I \) is the unit second order tensor; \( \vec{F}_b \) includes the gravitational force and other forces, if any; \( \vec{u} \) is the Reynolds-averaged velocity for RANS and Favre-filtered velocity for LES; \( \mu_t \) is the turbulent viscosity for RANS and subgrid-scale viscosity for LES; \( p^* = p + \frac{1}{3} \text{tr}(\tau_t) \) is the modified pressure with \( p \) being the real pressure and \( \tau_t \) being the modeled turbulent stress.

3.1. Turbulence models

Eq. (6) is incomplete due to the occurrence of \( \mu_t \). In order to make it complete, various turbulence models use different additional equation(s) to calculate \( \mu_t \). Among various equations of different turbulent models, we consider the most representative \( k \) equation in RANS modeling:

\[
\frac{\partial \rho k}{\partial t} + \nabla \cdot (\rho \vec{u} k) = \rho P - \rho \varepsilon + \nabla \cdot \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right], \tag{7}
\]

where \( \rho P \) is the production term, \( \rho \varepsilon \) is the dissipation term, and \( \nabla \cdot \left[ \left( \mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right] \) is the diffusion term with \( \sigma_k \) being the turbulent Schmidt number for \( k \).

We refer to Eq. (7) as the full form of \( k \) equation in the sense that the divergence-free condition is not used. Therefore, Eq. (7) is applicable to both compressible and incompressible flows. However, as shown in Fig. 1, the compressible version of turbulence models are constructed as "turbulentFluidThermoModels" which means that the thermal properties of the fluid(s) should always be provided for the compressible version of turbulence models. One underlying reason is that the energy equation is always solved in the solvers where compressible turbulence models are used.
3.2. Turbulence models for incompressible flows

In OpenFOAM, the incompressible version for turbulence models are constructed by assuming that $\rho$ is constant. Thus, incompressible turbulence models in OpenFOAM are actually designed for strict incompressible flows. For instance, the corresponding incompressible version of Eq. (7) reads

$$\frac{\partial k}{\partial t} + \nabla \cdot (\bar{u}k) = P - \epsilon + \nabla \cdot \left[ \left( \nu + \frac{\nu_t}{\sigma_k} \right) \nabla k \right],$$

(8)

where $\nu$ and $\nu_t$ are kinematic viscosities corresponding to $\mu$ and $\mu_t$, respectively.

As shown in Fig. 2, the dependency graph for incompressible turbulence models is much simpler in comparison with the compressible one. One important difference is that the thermal properties of the fluid(s) are no longer needed to construct an incompressible turbulence model.

4. Issues with turbulence modeling in isothermal VOF-based solvers

As stressed in Section 2, under the framework of VOF, the flow is variable-density incompressible due to the fact that the mixture density is always changing with $\alpha$. Obviously, OpenFOAM’s incompressible turbulence models are not suitable for this type of flows. For flows with heat transfer, OpenFOAM gives up the divergence-free condition and uses the compressible form of turbulence models directly. For instance, in diabatic VOF-based solvers, full form of governing equations, like Eq. (7), are always used.
Figure 2: Directory dependency graph for incompressible turbulence models [10]. No thermal property is needed to construct incompressible turbulence models.

However, for isothermal VOF-based solvers, an inconsistency arises. Using the chain rule, Eq. (7) could be rewritten as

\[
\left( \frac{\partial k}{\partial t} + \nabla \cdot (\bar{u}k) \right) + \frac{k}{\rho_m} \left( \frac{\partial \rho_m}{\partial t} + \bar{u} \cdot \nabla \rho_m \right) = P - \epsilon + \nabla \cdot \left[ \left( \nu_m + \frac{\nu_t}{\sigma_k} \right) \nabla k \right] + \frac{\nabla \rho_m}{\rho_m} \cdot \left[ \left( \nu_m + \frac{\nu_t}{\sigma_k} \right) \nabla k \right].
\]

(9)

In comparison with Eq. (8), there is an additional term on the l.h.s. of Eq. (9). According to the incompressible flow condition described by Eq. (5), this term should vanish. Therefore, the l.h.s. of Eq. (9) is equivalent to that of Eq. (8). On the r.h.s. of Eq. (9), an extra term, which contains \( \nabla \rho_m \), arises. As long as \( \rho_1 \neq \rho_2 \), this extra term is not zero for the transition zone. Therefore, the strict incompressible form of \( k \) equation deviates from the original \( k \) equation when it is applied to VOF simulations.

It is clear that this deviation is caused by the diffusion term where \( \rho \) is inside the divergence operator. As a result, not only Eq. (7), any equation that has this form of diffusion term will get an inconsistent strict incompressible version as long as \( \nabla \rho_m \neq (0, 0, 0) \). A list of available turbulence models in the official release of OpenFOAM v1706, which are related to isothermal VOF simulations, is shown in Table 1. There are 6 models which are only available in strict incompressible form. Among all the other 24 models, which could be used in both compressible and strict incompressible forms, only 2 could avoid the deviation issue.

Therefore, the strict incompressible turbulence models should not be applied to isothermal VOF simulations. However, these strict incompressible models are actually used in the corresponding solvers, e.g. interFoam, interIsoFoam and multiphaseInterFoam. The reason is that
Table 1: Turbulence models in OpenFOAM v1706

| Turbulence models          | Type       | Available for compressible flows | Available for strict incompressible flows | Correct forms available for variable-density incompressible flows |
|----------------------------|------------|----------------------------------|------------------------------------------|---------------------------------------------------------------|
| SpalartAllmaras            | RANS       | ✓                                | ✓                                        | ×                                                            |
| kEpsilon                   | RANS       | ✓                                | ✓                                        | ×                                                            |
| RNGkEpsilon                | RANS       | ✓                                | ✓                                        | ×                                                            |
| realizableKE                | RANS       | ✓                                | ✓                                        | ×                                                            |
| LaunderSharmaKE            | RANS       | ✓                                | ✓                                        | ×                                                            |
| kOmega                     | RANS       | ✓                                | ✓                                        | ×                                                            |
| kOmegaSST                  | RANS       | ✓                                | ✓                                        | ×                                                            |
| kOmegaSSTSAS               | RANS       | ✓                                | ✓                                        | ×                                                            |
| kOmegaSSTLM                | RANS       | ✓                                | ✓                                        | ×                                                            |
| v2f                        | RANS       | ✓                                | ✓                                        | ×                                                            |
| LRR                        | RANS       | ✓                                | ✓                                        | ×                                                            |
| SSG                        | RANS       | ✓                                | ✓                                        | ×                                                            |
| qZeta                      | RANS       | ×                                | ✓                                        | -                                                            |
| kkLOmega                   | RANS       | ×                                | ✓                                        | -                                                            |
| LamBremhorstKE             | RANS       | ×                                | ✓                                        | -                                                            |
| LienLeschziner             | RANS       | ×                                | ✓                                        | -                                                            |
| ShihQuadraticKE            | RANS       | ×                                | ✓                                        | -                                                            |
| LienCubicKE                | RANS       | ×                                | ✓                                        | -                                                            |
| Smagorinsky                | LES        | ✓                                | ✓                                        | ✓                                                            |
| WALE                       | LES        | ✓                                | ✓                                        | ✓                                                            |
| kEqn                       | LES        | ✓                                | ✓                                        | ✓                                                            |
| dynamicKEqn                | LES        | ✓                                | ✓                                        | ✓                                                            |
| dynamicLagrangian          | LES        | ✓                                | ✓                                        | ×                                                            |
| DeardorffDiffStress        | LES        | ✓                                | ✓                                        | ×                                                            |
| SpalartAllmarasDES         | DES        | ✓                                | ✓                                        | ×                                                            |
| SpalartAllmarasDDES        | DDES       | ✓                                | ✓                                        | ×                                                            |
| SpalartAllmarasIDDES       | IDDES      | ✓                                | ✓                                        | ×                                                            |
| kOmegaSSTDES               | DES        | ✓                                | ✓                                        | ×                                                            |
| kOmegaSSTDDES              | DDES       | ✓                                | ✓                                        | ×                                                            |
| kOmegaSSTIDDES             | IDDES      | ✓                                | ✓                                        | ×                                                            |

Compressible turbulence models cannot be used in adiabatic flows, as illustrated in Fig. 1.

5. **Code implementation**

5.1. **Coding strategy**

The goal of the present study is to construct turbulence models for isothermal incompressible with varying density. Therefore, adiabatic VOF solvers could use the correct turbulence models.
Several aspects should be considered for developing and implementing the desired models and solvers.

5.1.1. Usability

The new solvers are developed with the hope to replace corresponding existing solvers in OpenFOAM. Therefore, they are designed for common users who are unnecessarily able to write their solvers or even conscious of the issues stressed in Section 4. To the users, the usage of the new solvers should be as similar to the existing solvers as possible. Ideally, input files used for the new solvers should be exactly the same to those for the existing solvers.

5.1.2. Object-oriented programming

From the point of view of object-oriented programming, the new solvers should reuse as many existing codes as possible and make the minimum changes to where it is really needed. Therefore, even though the final goal is to design new isothermal VOF-based solvers for turbulent variable-density incompressible flows, the key point is to construct the corresponding turbulence models which are suitable for the desired solvers.

As mentioned in Section 2, the reason for assuming that the flow is incompressible is to utilize a simplified form of governing equations and to enhance the solver performance. Therefore, it is tempting to construct the variable-density incompressible turbulence models using the divergence-free condition. Actually, all the 6 models, as listed in Table 1, which are only available in strict incompressible forms, are constructed based on this consideration. However, the disadvantage is also quite obvious that these simplified models are no longer valid for compressible flows for all the other 24 turbulence models listed in Table 1, the strict incompressible versions are constructed based on the full versions only using the constant-density assumption and without using the divergence-free condition. Even though some computing time is sacrificed, this treatment significantly increases the reusability and maintainability of the codes. Therefore, we also opt the latter approach in this study. For the 24 models in Table 1 that have full governing equations like Eq. (7). They will be available in the new solvers as well. For the other 6 models which only have strict incompressible form, they will become unavailable in the new solvers.

It should be noted that for any customized turbulence model, as long as the full-form governing equations are employed, it could be adapted to the new solver as well.

5.2. New class for incompressible turbulence models with varying density

A new class, \texttt{vIncompressibleTurbulenceModel}, is created for the isothermal turbulence models where the density, \texttt{"rho"}, is explicitly referenced in its constructor, as shown in Listing 1. We use the prefix \texttt{v} to denote that this class is designed for variable-density flows. The same prefix will be used for the newly designed solvers as well. It should be emphasized that, with the new class, all the 24 full-form governing equations listed in Table 1 could be constructed without reading thermal properties from the input files. This allows the users to use the same input files for fluid properties.

\begin{verbatim}
Listing 1: Constructor for \texttt{vIncompressibleTurbulenceModel}.
Foam::vIncompressibleTurbulenceModel::vIncompressibleTurbulenceModel
\end{verbatim}

\texttt{Foam::vIncompressibleTurbulenceModel::vIncompressibleTurbulenceModel} (}
const volScalarField& rho,
const volVectorField& U,
const surfaceScalarField& alphaRhoPhi,
const surfaceScalarField& phi,
const word& propertiesName
);

turbulenceModel
(
U,
alphaRhoPhi,
phi,
propertiesName
),
rho_ (rho)
{}

5.3. New solvers for isothermal VOF-based flows

In vFoam, several isothermal VOF-related solvers are provided with which the full-form turbulence models are employed. All these solvers are modified based on the corresponding solvers in the official release of OpenFOAM. Since almost the same changes are made to each of these solvers, only one example is given here to illustrate how to change the existing interIsoFoam solver to the newly designed vInterIsoFoam. It should be mentioned that such modifications also apply to isothermal VOF-based solvers in other versions of OpenFOAM and any user-customized isothermal VOF-based solvers.

The first step is to modify the preprocessor directives in the main file of the solver. In interIsoFoam, "turbulentTransportModel.H" is included for the construction of the strict incompressible turbulence models, as shown in Listing 2. This should be substituted with "vTurbulentTransportModel.H" in vInterIsoFoam such that the full form of turbulence models, where the density is explicitly included, could be used, as shown in Listing 3.

Listing 2: Preprocessor directives in "interIsoFoam.C".

#include "isoAdvection.H"
#include "fvCFD.H"
#include "subCycle.H"
#include "immiscibleIncompressibleTwoPhaseMixture.H"
#include "turbulentTransportModel.H"
#include "pimpleControl.H"
#include "fvOptions.H"
#include "CorrectPhi.H"

Listing 3: Preprocessor directives in "vInterIsoFoam.C".

#include "isoAdvection.H"
The second step is to change the part which actually constructs the turbulence model. As shown in Listing 4, *interIsoFoam* only needs the velocity field "U" and the flux field "phi" to construct the turbulence field, and the density "rho" is not included in "phi". As for *vInterIsoFoam*, two additional fields, i.e. "rho" and "rhoPhi", are necessary for turbulence model construction, as shown in Listing 5. It should be noted that both "rhoPhi" and "phi" are available in the adiabatic VOF solvers, explicitly taking "phi" as an argument enhances the performance of the turbulence models.

Listing 4: Turbulence model construction in *interIsoFoam*.

```cpp
autoPtr<incompressible::turbulenceModel> turbulence
(
incompressible::turbulenceModel::New(U, phi, mixture)
);
```

Listing 5: Turbulence model construction in *vInterIsoFoam*.

```cpp
autoPtr<incompressible::turbulenceModel> turbulence
(
incompressible::turbulenceModel::New(rho, U, rhoPhi, phi, mixture)
);
```

Only these two changes are needed to enable the new solver to use the full-form turbulence models. Therefore, all the other features of the existing solvers are preserved.

5.4. Verification

The correctness of the code implementation has been verified by testing OpenFOAM tutorials, examples could be found in Appendix Appendix A and Appendix B.

6. Usage

The source code is provided in [11]. In order to use the code, one needs to load the environment variable for OpenFOAM v1706 first and then run "./compile.sh" to compile the code. As for the usage for a specific solver, e.g. *vInterFoam*, it is almost the same with the corresponding existing solver *interFoam*. For instance, it could be executed on 1024 processors by simply typing "mpirun -np 1024 vInterFoam -parallel" in the terminal.

Since the full-form turbulence models are used in the new solvers, the corresponding discretization schemes should be provided to solve the governing equations numerically. Other than this, the users could reuse all their *interFoam* input files for *vInterFoam*.
7. Performance evaluation

With both the new and original solvers available, we conduct a simple performance evaluation based on the experiment conducted by [12]. The comparison will justify our motivation for developing the new solvers.

The experiments were carried out in a rectangular flow channel with a 0.1% downward slope. The channel was 12.6 m long, 20 cm wide and 10 cm high. Three co-current air-water stratified flows were investigated and the corresponding flow configurations are listed in Table 2.

| Run reference | Water flow rate [L/s] | Air flow rate [L/s] |
|---------------|-----------------------|---------------------|
| 250           | 3.0                   | 45.4                |
| 400           | 3.0                   | 75.4                |
| 600           | 3.0                   | 118.7               |

A 2D computational domain is constructed as shown in Fig. 3. Similarly to the experiment, air and water are supplied via corresponding inlets. These two inlets are assumed to be separated by a zero-thickness 100 mm-long baffle. One reason for making such assumption is that details of the baffle are not provided in the paper. Another reason is that the measuring zone is quite far away from the inlets indicating that the detailed inlet configurations of the inlet region should only have minor effects on the results of the measuring zone.

7.1. Boundary conditions

Boundary conditions for all the flow variables are listed in Table 3.
Table 3: Boundary conditions.

|        | airInlet | waterInlet | outlet | upperWall | lowerWall | baffleAir | baffleWater |
|--------|----------|------------|--------|-----------|-----------|-----------|-------------|
| $\alpha$ | $\alpha = 0$ | $\alpha = 1$ | $\nabla \alpha = 0$ | $\alpha = 0$ | $\alpha = 1$ | $\alpha = 0$ | $\alpha = 1$ |
| $U$    | mappedC * | mappedC | advective | no slip | no slip | no slip | no slip |
| $p_{rgh}$ | fixed flux | fixed flux | fixed total pressure | fixed flux | fixed flux | fixed flux | fixed flux |
| $k$    | mappedN ** | mappedN | $\nabla k = 0$ | $k = 0$ | $k = 0$ | wall function | wall function |
| $\omega$ | mappedN | mappedN | $\nabla \omega = 0$ | wall function | wall function | wall function | wall function |

* mapped condition with the constraint on the average value.
** mapped condition without constraints.

7.2. Results

Three meshes with $\Delta y = 2$ mm, 1 mm, and 0.5 mm are constructed for the simulations, where $\Delta y$ denotes the mesh size (around the interface) in the vertical direction. For each mesh, both the strict incompressible and variable-density incompressible version of kOmegaSST model are used for turbulence modeling. All the simulations are run in transient modes. After the initial-condition effects die out, the sampling was carried out for 50 s to consider the variations caused by the wavy interface.

In Fig. 4-6, the pressure drop and $k$ profiles in the fully developed regions are compared with the experimental values. Two important conclusions could be easily made from these figures. One is that the strict incompressible version is much more sensitive to mesh refinement. This makes it impractical to conduct sensitivity studies on the mesh size. On the other hand, the variable-density incompressible version could capture the abrupt change in $k$ around the interface. However, the strict incompressible version totally misses this abrupt change.
We are aware that the variable-density incompressible version still fails to match the experimental data on a quantitative level, and that turbulence damping proposed by [8] might help to give a better prediction. However, they are both out of the scope of the current study. By conducting this performance evaluation, we are intending to show that the variable-density version could give a better prediction in comparison with the strict incompressible version. Therefore, the variable-density incompressible version of turbulence models should be used in VOF-based...
solvers.

8. Conclusions and outlooks

Due to the limitations of OpenFOAM’s turbulence models, isothermal VOF-based solvers could only use the strict incompressible form of turbulence models, which is inconsistent with the fact that such solvers are intended to solve variable-density flows.

With the object-oriented paradigm, by making the minimal changes to the existing codes, the developed solvers could construct the correct turbulence models and, at the same time, preserve all the other features of existing solvers.

All the newly designed solvers benefit from the new class for turbulence models. However, the newly designed class for isothermal variable-density turbulence models could also be applied to other flows that are not described by the VOF frame work.

Acknowledgement

The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at PDC. One of the authors (Wenyuan Fan) is also grateful for the support of China Scholarship Council (CSC).

References

References

[1] C.W Hirt and B.D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, jan 1981.
Appendix A. Consistency verification

All the newly designed solvers should use different governing equations for turbulence quantities in comparison with the corresponding existing solvers. However, when the flow is laminar, i.e. no turbulence model is used, both new and original solvers should provide exactly the same result. The damBreak4Phase case for multiphaseInterFoam is used for the consistency verification. The initial condition is given in Fig. A.7, water, oil, mercury and air will change their positions as time proceeds due to the density differences. multiphaseInterFoam and vMultiphaseInterFoam are used to run this case and the simulations last for 6 s. At the end of the simulation, both solvers give the same prediction for phase distributions, as shown in Fig. A.8. This proves that the newly designed solvers have the same behavior with their corresponding existing solvers if the flow is laminar.
Figure A.7: Initial condition for test case *damBreak4Phase* where 0, 1, 2, 3 are used to denote water, oil, mercury and air, respectively.

Figure A.8: Phase distribution at the end of simulations. Exactly the same result is predicted by both solvers with reasonable stratification caused by the density difference.
Appendix B. Modification verification

The goal of the modification verification is to show that the full-form turbulence models are constructed in the new solvers. The motorBike case for *interDyMFoam* is used for this test. In the simulation, a motorbike runs on the ground which is covered by water. The flow is turbulent and a turbulence model (kOmegaSST in Table 1) is used in the simulations. As shown in Fig. B.9, two solvers give quite different predictions for the shape of water-air interface and the distribution of velocity magnitude. This proves that *vInterDyMFoam* does use a different turbulence model when compared with *interDyMFoam*. However, we could neither conclude that Fig. B.9a is wrong nor claim that Fig. B.9b is correct.

![Result comparison for interDyMFoam and vInterDyMFoam](image)

Figure B.9: Result comparison for *interDyMFoam* and *vInterDyMFoam*: water-air interface (denoted by the gray surface); velocity magnitude in the middle plane of the domain.