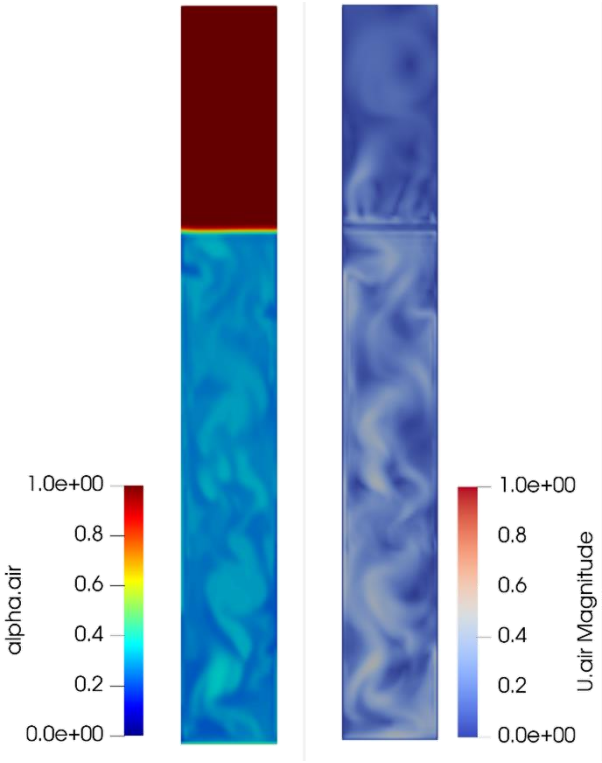
Mastering twoPhaseEulerFoam

Three: Bubble Column



How to Simulate a Bubble Column Using OpenFOAM®

Compatible with	OpenFOAM® 7 OpenFOAM® v1912

Author

Hamidreza Norouzi

Amirkabir University of Technology

Center of Engineering and Multiscale Modeling of Fluid Flow

Extra consideration:

- This document is developed to teach how to use OpenFOAM® software. The document has gone under several reviews to reduce any possible errors, though it may still have some. We will be glad to receive your comments on the content and error reports through this address: h.norouzi@aut.ac.ir

Document history

Revision	Description	Date
Rev1	The text was reviewed and simulation was performed by OpenFOAM v1912.	April 30, 2020
Rev0	The first draft was prepared and run by OpenFOAM 7.	April 23, 2020

Table of Contents

Prerequisites 4
How to get simulation setup files? 4
1. Brief Description of twoPhaseEulerFoam 5
2. Problem definition 5
3. Simulation setup 6
3.1. Physical properties of phases 6
3.2. Defining phases and interphase coupling parameters 8
3.1.1. Phases and phase definition 11
3.1.2. Blending 12
3.1.2. Interphase drag, lift, virtual mass and Heat transfer models 15
3.1.3. Other settings and sub-models 17
3.2. Turbulence properties of phases 19
3.4. Gravitational acceleration 20
3.5. Generating geometry and mesh 20
3.5.1. Mesh generation for 2D simulation 20
3.5.2. Mesh generation for 3D simulation 21
3.6. Boundary and initial conditions 22
4. Running the simulation 23
5. Results 24

Prerequisites

- You need to be familiar with basics of OpenFoam® to start this tutorial.
- You need to first read tutorial One of this series (Mastering twoPhaseEulerFoam, One: Fluidized bed).

How to get simulation setup files?

You have two options to get simulation setup files:

- **Website:** the setup files (a compressed file) are uploaded on www.cemf.ir alongside this tutorial file. You can find the 3D mesh among these files too, so, you can perform 3D simulation.
- **Standard tutorial cases:** execute the following command. It copies a case setup files to your desktop folder. These setup files are very similar to what you simulate here. So, you only need to make the required changes as explained in this tutorial. In this way, you won't have 3D mesh for a 3D simulation.

```
> cp -r $FOAM_TUTORIALS/multiphase/twoPhaseEulerFoam/RAS/bubbleColumn $HOME/Desktop/
```

1. Brief Description of twoPhaseEulerFoam

twoPhaseEulerFoam is a solver for a system of 2 non-reacting compressible fluid phases. One phase in this system is always dispersed. So it is a good candidate for simulating bubble columns in gas/liquid systems or fluidized beds and spouted beds in gas/solid systems. The solver also solves the energy equation (enthalpy or internal energy) for both phases and couple them by the one-film resistance heat transfer model.

In the case of gas-liquid systems, laminar and turbulence models (RAS and LES) models can be selected for both phases. In this case of gas-solid or liquid solid systems, these models can only be applied for the fluid phase. The solid phase momentum equation incorporates a model for the stress tensor. Two main approaches are possible: inviscid solid phase with a pressure model and a solid phase with stress tensor that is obtained by KTGF theory and frictional models.

Various sub-models for interphase coupling are also provided that make it possible to select a wide range of physical models for the system. It is also possible to extend the standard solver to include new sub-models in the simulation.

2. Problem definition

The bubble column in this tutorial is a cylinder with internal diameter of 0.26 m and height of 2 m. One meter of the column is initially filled with water at 300 K and atmospheric pressure. Air (at 300 K) enters from the bottom of the column through a perforated distributor (here, a uniform gas is assumed for simplicity). You simulate this column to find gas holdup distribution and other field variables. All properties and operating conditions are given in Table 1. This simulation is performed based on the experiments of Kumar et al. [Kumar, Moslemian, and Dudukovic, 1997. AIChE Journal, 43, 1414-1425] with some slight changes.

Table 1: Physical properties and operating conditions

Water viscosity [Pa.s]	3.645×10^{-4}	Air viscosity [Pa.s]	1.84×10^{-8}
Water density [kg/m^3]	~1000	Air density	Ideal gas
Water heat capacity [J/kg/K]	4195	Air heat Capacity [J/kg/K]	1007
Water Prandtl number	2.289	Air Prandtl number	0.7
Water temperature [K]	300	Inlet air temperature [K]	300
Air-water surface tension [N/m]	0.072	Superficial gas velocity [m/s]	0.05
Column ID [m]	0.26	Bubble diameter [m]	0.003
Column height [m]	2		

3. Simulation setup

Two alternative simulations are followed in this tutorial. The first one is a two-dimensional simulation for a column with dimensions 0.26×2 m^2. The mesh for this simulation is generated by blockMesh utility. The second one is a three-dimensional simulation for a column with the ID of 0.26 and height of 2 m. The mesh for this simulation is generated using a meshing software which is converted to OpenFOAM® mesh.

3.1. Physical properties of phases

Physical properties of phases are defined separately in thermophysicalProperties.air and thermophysicalProperties.water in constant folder.

In constant/thermophysicalProperties.air the physical properties of air are specified:

constant/thermophysicalProperties.air

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight   28.9;
    }
```

```
    thermodynamics
    {
        Cp          1007;   // J/kg/K
        Hf          0;
    }
    transport
    {
        mu          1.84e-05;    // Pa.s
        Pr          0.7;
    }
}
```

In `thermoType` sub-dictionary, you can the type of physical model you want to consider for this phase, air. Thermo-physical model type is `heRhoThermo`, which is a density-based model. This is the only option here, since `phaseModel` class (the class used to define phase properties in this solver) in the `twoPhaseSystem` is a wrapper around `heRhoThermo` class. Air is considered as a `pureMixture` whose properties are defined in `mixture` sub-dictionary. And finally, `energy` defines the type of energy equation you want to be solved for this phase. Two options are available: `sensibleInternalEnergy` and `sensibleEnthalpy`. The first option tells the solver to solve energy equation based on the internal energy of the mixture and the second, based on the sensible enthalpy of the mixture. Based on the settings in this file, ideal gas EOS is selected for density, constant heat capacity model (`hConst`) for enthalpy, and constant transport properties for dynamic viscosity and heat conductivity (it is calculated using Prandtl number). For more details on `heRhoThermo` model, see the Appendix A of part One in this tutorial series.

The properties of water phase are defined as follows:

constant/thermophysicalProperties.water

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          eConst;
    equationOfState perfectFluid;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
```

```
        molWeight       18;
    }
    equationOfState
    {
        R               3000;
        rho0            1027;
    }
    thermodynamics
    {
        Cv              4195;  // J/kg/K
        Hf              0;
    }
    transport
    {
        mu              3.645e-4;  // Pa.s
        Pr              2.289;
    }
}
```

Almost the same model is selected for water. Except that the density of water (`equationOfState`) is obtained by `perfectFluid` model:

$$\rho = \frac{1}{RT}p + \rho_0$$

where ρ_0 is density at 0 K and R is a constant that should be supplied by user. Higher values of R result in lower compressibility of the fluid.

3.2. Defining phases and interphase coupling parameters

The simulation contains two phases: air and water. Phase definition and interphase coupling can be found in constant/phaseProperties file. Various settings can be fixed in this file. Since this file contains various details, the whole content of the file is first presented as a reference and later in the following subsections details are expalined.

constant/phaseProperties

```
phases (air water);

air
{
    diameterModel   isothermal;
    isothermalCoeffs
    {
        d0              3e-3;
        p0              101325;
    }
```

```
    residualAlpha   1e-6;
}

water
{
    diameterModel   constant;
    constantCoeffs
    {
        d               1e-4;
    }

    residualAlpha   1e-6;
}

blending
{
    default
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.5;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.5;
    }

    heatTransfer
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.7;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.7;
    }
}

sigma
(
    (air and water)     0.072
);

aspectRatio
(
    (air in water)
    {
        type            Wellek;
    }

    (water in air)
    {
        type            constant;
        E0              1.0;
    }
);

drag
(
    (air in water)
    {
        type            IshiiZuber;
```

```
        residualRe      1e-3;
        swarmCorrection
        {
            type            none;
            residualAlpha   Tomiyama;
            l               1.75;
        }
    }

    (water in air)
    {
        type            SchillerNaumann;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }

    (air and water)
    {
        type            segregated;
        m               0.5;
        n               8;
        swarmCorrection
        {
            type        none;
        }
    }
);

virtualMass
(
    (air in water)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }

    (water in air)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }
);

heatTransfer
(
    (air in water)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
    }

    (water in air)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
    }
```

```
);


lift
(
    (air in water)
    {
        type constantCoefficient;
        Cl   0.25;
    }

    (water in air)
    {
        type none;
    }
);

wallLubrication
(
);

turbulentDispersion
(
);
```

3.1.1. Phases and phase definition

`phases` entry defines two phase names for the simulation: `air` and `water`. In `air` sub-dictionary, the `diameterModel` and `residualAlpha` are defined. Here, `isothermal` diameter model is selected for air (bubbles) with bubble diameter of 0.003 m at reference pressure 101325 Pa. In `water` sub-dictionary, `constant` diameter model with diameter of 0.0001 m is selected for water droplets. More details on the available diameter models can be found in the following table.

Table 2: diameter models in twoPhaseEulerFoam

Diameter model	Description
constant	Assumes a constant diameter for the dispersed phase which can be defined in constantCoeffs dictionary. This can be used for both solid and fluid (gas/liquid) dispersed phases.
isothermal	Assumes an isothermal model for bubble diameter that changes directly with pressure. In isothermalCoeffs dictionary, two keywords d_0 and p_0 should be defined. d_0 represents mean bubble diameter at p_0 (in Pa). The mean bubble diameter at pressure p is given by: $d = d_0 (p_0/p)^{1/3}$
IATE	Interfacial Area Transport Equation for bubble diameter: It solves for the interfacial curvature per unit volume of the phase rather than interfacial area per unit volume. This model can only be used for bubbles as dispersed phase. More information can be found in [Ishii, M., Kim, S. and Kelly, J., Nuclear Engineering and Technology, 2005:37 (6)].

Note

This solver does not have any population balance model for tracking the bubble size changes during the operation of the column. reactingTwoPhaseEulerFoam solver provides a population balance model for tracking bubble size changes.

3.1.2. Blending

In this sub-dictionary you can specify for the solver how to distinguish the continuous and dispersed phases in the two phase system. There are three types for blending: `none`, `linear` and `hyperbolic`. Blending type `none` is used when one phase is always dispersed into another. This used when simulating gas-solid systems (like fluidized beds); spraying systems where droplets are always dispersed in the gas; or liquid-solid systems. See part one in this tutorial series (Mastering twoPhaseEulerFoam, One: Fluidized bed) for more details. The other two types are good for more complicated systems where dispersed and continuous phases may interchange.

Sub-dictionary `blending` is as follows:

```
blending
{
    default
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.5;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.5;
    }

    heatTransfer
    {
        type            linear;
        maxFullyDispersedAlpha.air 0.3;
        maxPartlyDispersedAlpha.air 0.7;
        maxFullyDispersedAlpha.water 0.3;
        maxPartlyDispersedAlpha.water 0.7;
    }
}
```

Two blending criteria are defined here: `default` and `heatTransfer`. The `default` blending is used for all interphase models (here, `drag`, `virtualMass`, `lift` and etc.) and the `heatTransfer` blending is explicitly used for `heatTrasnfer` model.

You use `linear` as the blending type. In linear blending type, you need to define two values: `maxFlullyDispersedAlpha` and `maxPartlyDispersedAlha`. These two values define the boundaries for the dispersed, mixed and continuous phase. For example, consider the settings in `default` sub-dictionary.

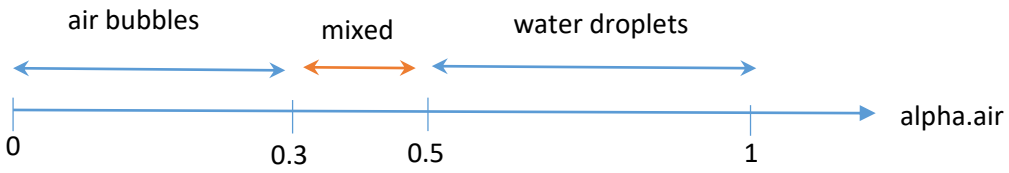


Figure 1: Phase boundaries defined for air in default sub-dictionary

The mechanism for the calculation of an interphase coupling parameter K is as follows. There are two factors f1 and f2 that are calculated in any blending model. f1 is the factor for phase#1 and f2 is the factor for phase#2. Suppose that you defined two models for `(air in water)` and `(water in air)`. In pair `(air in water)`, air is considered as dispersed phase and water as continuous phase and vice versa. The interphase parameter for `(air in water)` is K12 and that for `(water in air)` is K21. The interphase coupling parameter is then calculated as:

$$K = (1-f1) \times K12 + f2 \times K12 + (f1-f2) \times K_{mixed} \qquad \text{Eq. (1)}$$

where K_{mixed} is the selected model for the conditions in which no continuous phase is defined, for example settings `(air and water)` in `drag` sub-dictionary. The linear blending gives the following graph for f1 and f2:

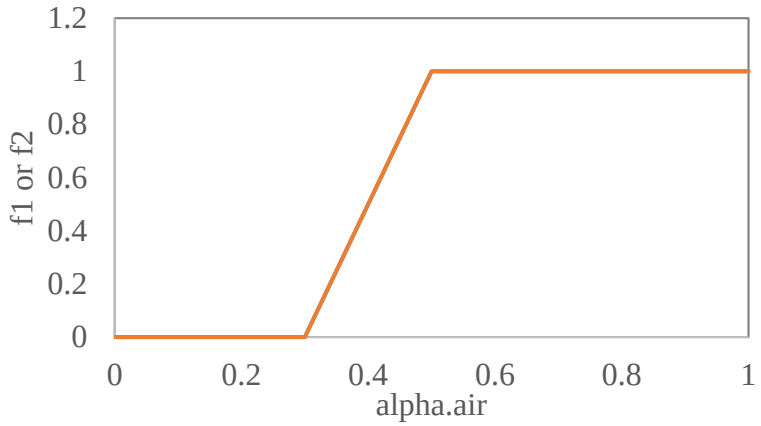


Figure 2: Factors f1 and f2 for phase air with the settings defined in default sub-dictionary

and the hyperbolic blending gives the following graphs for f1 and f2:

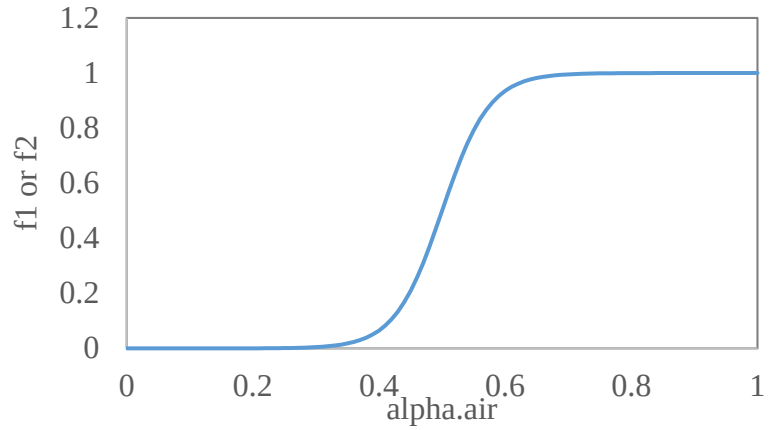


Figure 3: Factors f1 and f2 for phase air if the blending type is hyperbolic with the following settings:

```
{
    type            hyperbolic;
    transitionAlphaScale    0.3;
    maxDispersedAlpha.air   0.5;
    maxDispersedAlpha.water 0.5;
}
```

Comparing these graphs with Eq. (1), you will understand how interphase transfer coefficient is calculated.

3.1.2. Interphase drag, lift, virtual mass and Heat transfer models

A number of interphase momentum transfer mechanisms can be defined between two phases. The drag sub-dictionary is as follows:

```
drag
(
    (air in water)
    {
        type            IshiiZuber;
        residualRe      1e-3;
        swarmCorrection
        {
            type            none;
            residualAlpha   Tomiyama;
            l               1.75;
        }
    }

    (water in air)
    {
        type            SchillerNaumann;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }

    (air and water)
    {
        type            segregated;
        m               0.5;
        n               8;
        swarmCorrection
        {
            type        none;
        }
    }
);
```

For conditions where air is dispersed in water (air bubbles), `IshiiZuber` model with swarm correction model of `Tomiyama` is used [Tomiyama et al., Trans. Jpn. Soc. Mech. Eng., B 1995, 61 (588), 2810–2817]. The swarm correction is a factor which is multiplied by drag force coefficient to account for the bubble interaction effects at high gas holdup. It reads as:

$$Cs = \left(max(\alpha_{water}, \alpha_{residual})\right)^{3-2l}$$

For conditions where water is dispersed (droplets), the correlation of `SchillerNaumann` with no swarm effect is selected. Various types of drag models that are implemented for this solvers are

discussed in tutorial One of this series (Mastering twoPhaseEulerFoam, One: Fluidized bed). The `segregated` drag model gives K_{mixed} for the conditions where no distinct dispersed phase is identified.

Virtual mass is the force induced by the acceleration of one phase relative to another. This effect is commonly used in bubble column simulations. Virtual mass models for both pairs are selected in `virtualMass` sub-dictionary:

```
virtualMass
(
    (air in water)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }

    (water in air)
    {
        type            constantCoefficient;
        Cvm             0.5;
    }
);
```

Two models for virtual mass are implemented: `constantCoefficient` and `Lamb`. In constant coefficient model, virtual mass coefficient is taken constant (Here, it is equal to 0.5). Lamb's model is more complicated than constant model and accounts for the aspect ratio of bubbles in the calculation of this coefficient [Lamb, Hydrodynamics, Cambridge University Press, 1895].

In `heatTransfer` entry, `RanzMarshall` model is selected for calculating heat transfer coefficient between air bubbles in water as the continuous phase and vice versa. As it was explained earlier, the solver uses one-file resistance model for heat transfer. If for example you want to consider the heat transfer resistance to be in the other phase in each pair settings, you need to define `spherical` heat transfer model. In `spherical` model, Nu is constant and equal to 10.

```
heatTransfer
(
    (air in water)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
```

```
    }

    (water in air)
    {
        type            RanzMarshall;
        residualAlpha   1e-4;
    }
);
```

In `lift` entry, constantCoefficient lift model is selected for air bubbles (air in water) and no lift model is selected for water droplets (water in air).

```
lift
(
    (air in water)
    {
        type constantCoefficient;
        Cl   0.25;
    }

    (water in air)
    {
        type none;
    }
);
```

Other lift models that are available in this solver are listed in Table 3.

Table 3: Lift models implemented in twoPhaseEulerFoam solver.

Model name	Reference
`constantCoefficient`	
`LegendreMagnaudet`	[Legendre, D. and Magnaudet, J., Journal of Fluid Mechanics, 1998 (368), 81-126.]
`Moraga`	[Moraga, F.J. et al., International Journal of Multiphase Flow, 1999 (25), 1321-1372.]
`Tomiyama`	[Tomiyama, A. et al., Chemical Engineering Science, 2002 (57), 1849-1858.]

3.1.3. Other settings and sub-models

In `sigma` entry, you can define the interfacial surface tension between air and water.

```
sigma
(
    (air and water)      0.072    // N/m
);
```

This value is used in various models that has sigma as their input property (mostly in Eötvös and Morgan dimensionless numbers).

Aspect ratio is used in a series of interphase models. Here a constant aspect ratio is defined for both bubbles and droplets. It assumed to be 1 (spherical).

```
aspectRatio
(
    (air in water)
    {
        type            constant;
        E0              1.0;
    }

    (water in air)
    {
        type            constant;
        E0              1.0;
    }
);
```

Other aspect ratio models are also implemented which are listed in the table below.

Table 4: Aspect ratio models implemented in twoPhaseEulerFoam solver

Model name	Reference
constant	
VakhrushevEfremov	Vakhrushev, I.A. & Efremov, G.I., Chemistry and Technology of Fuels and Oils, 1970 (6), 376-379.
Wellek	Wellek, R.M.et al., International Journal of Multiphase Flow, 1966 (12), 854-862.

Other phase interaction models are wall lubrication and turbulent dispersion force (bubble induced turbulent dispersion), which are not included in this simulation. Here is a list of bubble induced turbulent dissipation models is given in Table 5 and a list of wall lubrication models in Table 6.

Table 5: Bubble induced turbulent dispersion models in twoPhaseEulerFoam

Model	Reference
Burns	Burns, A.D. et al., 5th international conference on multiphase flow, 2004.
constantCoefficient	
Gosman	Gosman, A.D. et al., AIChE Journal, 1992 (38), 1946-1956.
LopezDeBertodano	Burns, A.D. et al., 5th international conference on multiphase flow, 2004.

Table 6: wall lubrication models models in twoPhaseEulerFoam

Model	Reference
Antal	Antal, S.P. et al., International Journal of Multiphase Flow, 1991 (17), 635 – 652.
Frank	Frank, T., NAFEMS Seminar: Simulation of Complex Flows (CFD), 2005.
Tomiyama	Tomiyama, A., Multiphase Science and Technology, 1998 (10), 369-405.

3.2. Turbulence properties of phases

Turbulence properties of air phase and water phase are separately defined in files turbuleceProperties.air and turbylenceProperties.water, respectively. Here, the mixtureKEpsilon turbulence model with default model coefficient is selected for the phase air and the same model for the phase water.

constant/turbulenceProperties.air

```
simulationType  RAS;

RAS
{
    RASModel mixtureKEpsilon;

    turbulence      on;
    printCoeffs     on;

}
```

constant/turbulenceProperties.water

```
simulationType  RAS;

RAS
{
    RASModel mixtureKEpsilon;

    turbulence      on;
    printCoeffs     on;
}
```

Note

In tutorial Two of this series (Two: Extending the Solver), the method of implementing a new turbulent model for this solver is explained.

3.4. Gravitational acceleration

The gravitational acceleration can be defined in constant/g file as follows:

constant/g

```
dimensions      [0 1 -2 0 0 0 0];
value           (0 -9.81 0);
```

3.5. Generating geometry and mesh

Depending on whether you want to perform 2D or 3D simulation, you need to follow one of the following steps.

3.5.1. Mesh generation for 2D simulation

The content of system/blockMeshDict is as follows:

```
convertToMeters 1;

vertices
(
    (-0.13 0 0)
    ( 0.13 0 0)
    ( 0.13 2 0)
    (-0.13 2 0)
    (-0.13 0 0.1)
    ( 0.13 0 0.1)
    ( 0.13 2 0.1)
    (-0.13 2 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (26 200 1) simpleGrading (1 1 1)
);

edges
(
);

patches
(
    patch inlet
    (
        (1 5 4 0)
    )
    patch outlet
    (
        (3 7 6 2)
    )
    wall walls
```

```
    (
        (0 4 7 3)
        (2 6 5 1)
    )
);
```

Eight vertices of this column are defined first. Note that the 3rd dimension of this column is taken equal to 0.1 m. however, since the number of cells in this direction is one, the simulation is considered 2D and hence this length does not have any effect on the results. Three patches are defined: inlet as the gas inlet, outlet at the top, and walls at both sides of the column. Execute the following command to generate the 2D mesh:

> blockMesh

Note

You can define all the boundaries as patch, and then by applying no-slip condition (for example, for velocity field) on those patches, satisfy the wall condition. However, this has a drawback when the solver needs to know which boundary should be considered as a wall. This is especially important in this solver when wall lubrication models are active, since these models need to know the actual distance between each cell and wall.

3.5.2. Mesh generation for 3D simulation

For a 3D simulation, a compressed mesh file is supplied with this tutorial. Execute the following commands to generate the mesh:

```
> tar xvf CylinderBubbleColumn.tar.xz
> fluentMeshToFoam CylinderBubbleColumn.msh
> rotateMesh "(0 0 1)" "(0 1 0)" -noZero -constant
```

The first command extracts the compressed mesh file. The second command converts the mesh to foam mesh. The last command rotates the mesh 90 degrees around x-axis. The axis of column is aligned along z-axis and you need to have a column with its axis aligned along y-axis.

3.6. Boundary and initial conditions

Initial and boundary conditions of the filed variable are defined in 0 folder. In table below, you can see a brief overview of the important boundary conditions. Note that the exact syntax for boundary conditions can be found in the simulation setup files.

Table 7: List of boundary and initial conditions

Field name	internalField	Inlet condition	Outlet condition	walls condition
alpha.air	0	fixedValue value = 0.5	inletOutlet value = 1	zeroGradient
epsilon.air & epsilon.water	1.5e-4	fixedValue value = $internalField	inletOutlet value = $internalField	epsilonWallFunction value = $internalField
k.air & k.water	3.75e-5	fixedValue value = $internalField	inletOutlet value = $internalField	kqRWallFunction value = $internalField
nut.air & nut.water	1e-8	calculated value = $internalField	calculated value = $internalField	nutkWallFunction; value $internalField
p_rgh	101325	fixedFluxPressure value = $internalField	prghPressure p = $internalField value = $internalField	fixedFluxPressure value = $internalField
U.air	(0 0.1 0)	fixedValue value = $internalField	pressureInletOutletVelocity phi = phi.air value = $internalField	fixedValue value = (0 0 0)
U.water	(0 0 0)	fixedValue value = $internalField	pressureInletOutletVelocity phi = phi.water value = $internalField	fixedValue value = (0 0 0)

T.air	300	fixedValue;
		value = $internalField;
			inletOutlet;
			phi = phi.air;
			value = $internalField;
				zeroGradient;

T.water	300	fixedValue;
		value = $internalField;
			inletOutlet;
			phi = phi.water;
			value = $internalField;
				zeroGradient;

Based on the values given above, the air volume fraction in inlet is 0.5 and upward local air velocity is 0.1 m/s. This is equivalent to superficial gas velocity of 0.05 m/s. The mass flux of air at inlet boundary is calculated as $\rho\alpha(\boldsymbol{U}.\boldsymbol{n})A$. In the calculation of mass flux, volume fraction appears. Superficial gas velocity is defined for an empty bed in which the mass flux for inlet boundary is calculated as $\rho\alpha(\boldsymbol{U}.\boldsymbol{n})A$.

For scalar variables, `inletOutlet` condition is applied to prevent reverse flow into the column. Since the mesh is not resolved near the walls, wall functions are specified for turbulence

variables. For volume fraction fields, `zeroGradient` is applied at walls and for p_rgh field, `fixedPressureFlux`.

Based on the problem definition, the bed is initially filled by water particles up to 1 m. This condition can be specified by using `setFields` utility. The settings can be found in system/setFieldsDict. The `default` settings are first applied for the whole domain (column) and then the values specified in the `regions` entry overwrite the `default` values. Here, the default setting is $\alpha_{air} = 1$ (and hence $\alpha_{water} = 0$), which creates and an empty column. Then, volume fraction of the cells reside in the cuboid `box` defined by two points `(-0.13 0 -0.13)` and `(0.13 1.0 0.13)` is set to $\alpha_{air} = 0.0$ (and hence $\alpha_{water} = 1.0$). Execute the `setFields` command from case directory to apply these initial conditions.

```
> setFields
```

system/setFieldsDict

```
defaultFieldValues
(
    volScalarFieldValue alpha.air 1
);

regions
(
    boxToCell
    {
        box (-0.13 0 -0.13) (0.13 1.0 0.13);
        fieldValues
        (
            volScalarFieldValue alpha.air 0
        );
    }
);
```

Note

If you are using OpenFoam v1912 and before executing `setFields`, you must first copy and rename alpha.air.orig to alpha.air. In other words, the folder 0 should contain alpha.air file.

4. Running the simulation

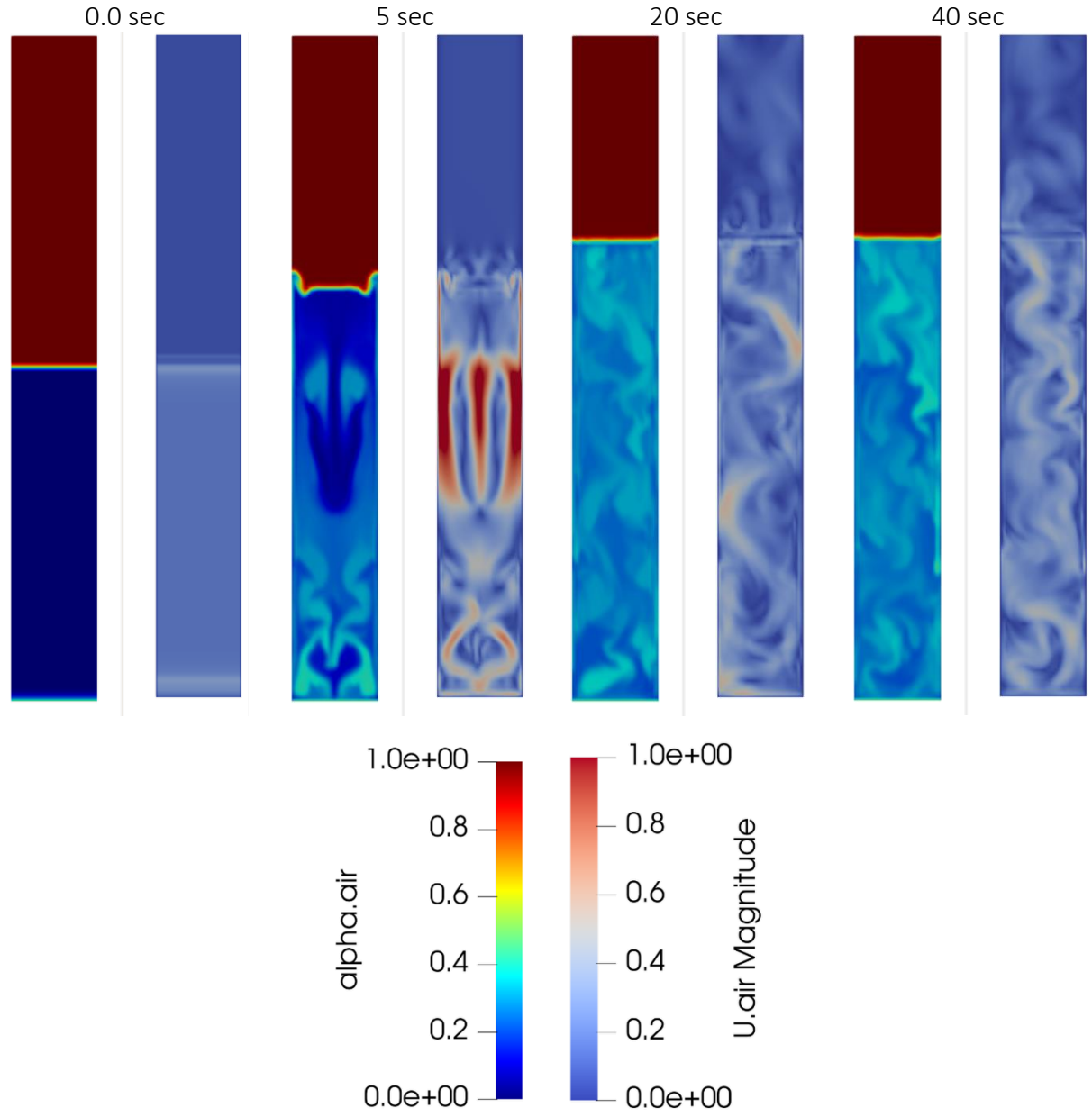
You need to execute the following commands in the case directory:

```
> twoPhaseEulerFoam
```

Depending on the computational power of your computer, it may take several minutes to complete the simulation.

5. Results

Some snapshots of the bubble column simulation are presented here.


0.0 sec
5 sec
20 sec
40 sec
1.0e+00
0.8
0.6
0.4
0.2
0.0e+00
alpha.air
1.0e+00
0.8
0.6
0.4
0.2
0.0e+00
U.air Magnitude