Rising Bubble with Mass Transfer using interFoam

Mano Prithvi Raj R

Velammal Engineering College, Chennai, Tamil Nadu

Abstract

The main focus of this case study is to extend interFoam, a two-phase simulation solver in OpenFOAM, such that it accepts a user-defined mass transfer rate between the said phases and allows mass transfer to occur between the two phases. Validation of the solver is done using the sucking interface problem to validate phase change, and is then executed to simulate a bubble rising in a column of a miscible fluid, where a vapour bubble undergoes condensation.

1. Introduction

The rising bubble simulation is a benchmark case study used to study interfacial flows and to validate the interface capturing ability of multiphase solvers. One of these solvers is OpenFOAM's *interFoam*. However, majority of these studies assume flow to be immiscible. However, there are a number of case studies, especially in the nuclear and energy industries, where condensing and evaporating flows need to be validated. In these flows, the fluids are miscible and there is a transfer of mass between the phases.

On its own, the interFoam solver does not take the mass transfer between the fluids in a simulation into account, that is, the solver can only be used to solve problems that are immiscible in nature. A modification to the governing can be made so that the interFoam solver can more versatile and can be applied to a number of multi-phase problems that involve immiscible phases and interfacial mass transfer. Appropriate source terms are added to the necessary equations and the new solver's ability to simulate mass transfer between two phases by solving the one-dimensional sucking interface problem.

The new solver is named and compiled as "interMassFoam".

2. Problem Statement

The simulation starts with a bubble of one phase submerged in a column filled with another phase. During the simulation, the bubble should rise upwards with time. Isothermal conditions and miscible components are assumed. A constant mass transfer rate has been defined explicitly, whose value needs to be provided by the user. It is expected that the bubble should mix with the fluid surrounding it during the movement. But before the bubble case, the working of the modified solver is validated using the sucking interface problem. The solver works on the assumptions that flow is incompressible, isothermal and the fluid is Newtonian.

3. Governing Equations:

3.1 interFoam Governing Equations:

3.1.1. Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \nabla . (\rho \boldsymbol{U}) = 0 \tag{1}$$

This is the global continuity equation. Mass is conserved throughout the simulation.

3.1.2. Momentum Equation:

$$\partial(\rho \boldsymbol{U}) \partial t + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \nabla \cdot \left(\mu(\nabla \mathbf{U}\, \mathrm{T} + \nabla \boldsymbol{U})\right) = -\nabla P + \rho \boldsymbol{g} + \boldsymbol{\sigma}\, \kappa \nabla \alpha_L \tag{2}$$

The last term on the right-hand side of the momentum equation indicates the surface tension between two phases. The surface tension is computed using the Continuum Surface Tension (CSF) model. $\boldsymbol{\sigma}$ is the surface tension coefficient and κ is the curvature.

3.1.3. Volume Fraction Transport Equation:

The volume fraction derivation begins with how thermophysical properties, in this case the density, are defined. The average density ρ in a cell is calculated as:

$$\rho = \alpha \rho_L + (1 - \alpha)\rho_V \tag{3}$$

where α_L is the volume fraction, whose value distinguishes the two phases at position x and time t:

$$\alpha(x, t) = \begin{cases} 1, & x \in \Omega_L \\ 0, & x \in \Omega_V \end{cases} \tag{4}$$

where Ω_L and Ω_G are the domains pertaining to phases L (liquid) and V (vapour).

On substituting the average density equation into the continuity equation (Eqn. 1), we get:

$$\frac{\partial}{\partial t}(\alpha\rho_L + (1-\alpha)\rho_V) + \nabla \cdot (\alpha\rho_L + (1-\alpha)\rho_V)\boldsymbol{U} = 0 \quad (5)$$

This is where the mass transfer rate is introduced. Separate equations are written for liquid and vapour phases' continuity equations.

$$\begin{cases} \frac{\partial}{\partial t}(\alpha\rho_L) + \nabla \cdot (\alpha\rho_L)\boldsymbol{U} = -\dot{m} \\ \frac{\partial}{\partial t}(\alpha\rho_V) + \nabla \cdot \big((1-\alpha)\rho_V\big)\boldsymbol{U} = \dot{m} \end{cases} \quad (6)$$

where $\dot{m}$ is the mass transfer rate and the unit is kg/m^{3}s. The signage, in theory, indicates the direction of mass transfer relative to the phases. The value of mass transfer can be determined by using additional transport equations, such as a temperature or an energy equation and in that case, the positive value of $\dot{m}$ will indicate boiling, and a negative value will indicate condensation. On further simplification:

$$\begin{cases} \frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha\boldsymbol{U}) = -\dot{m}\frac{1}{\rho_L} \\ -\frac{\partial \alpha}{\partial t} - \nabla \cdot (\alpha\boldsymbol{U}) + \nabla \cdot \boldsymbol{U} = -\dot{m}\frac{1}{\rho_L} \end{cases} \quad (7)$$

The first equation is used as the governing equation for the volume fraction equation.

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha\boldsymbol{U}) = -\dot{m}\frac{1}{\rho_L} \quad (8)$$

The value of volume fraction needs to be in the range of 0 to 1. Out of bound values would contradict the practicality of the problem's physics. OpenFOAM solves the volume fraction equation explicitly, using the MULES algorithm, which in principle restricts the undershooting and overshooting of the volume fraction value.

4. Simulation Procedure

4.1 Validation of Phase Change and Interfacial Mass Transfer

The Sucking Interface problem is used to test if the modifications made to interFoam are applied and the solver does indeed allow mass transfer and in turn, phase-change. For the sake of testing the solver and validating the phase change, the sucking problem is taken as a one-dimensional case. The left and right sides of the computational domain are a wall and flow

outlet, respectively. The vapor phase occupies the space between the wall and the interface with the liquid phase, which is assumed to be superheated.

4.1.1 Geometry and Mesh

A simple 1 x 0.05 m^2 rectangular block is taken as the geometry and meshing is done using the blockMesh command. The mesh has 50 hexahedral cells. The phases are defined using the setFields command.

4.1.2 Initial and Boundary Conditions:

Since the isothermal condition is assumed, the properties of the fluids are taken at 100°C.

Fluid	Density (kg/m^3)	Kinematic Viscosity (m^2/s)	Surface Tension (N/m)
Water	958.40	0.29e-6	0.0059
Vapour	0.598	2.17e-5	

Table 1 Properties of Fluids

Field	Condition
U	Left – *fixedValue (0 0 0)* Right – *inletOutlet (0 0 0)* Top,Bottom, Front and Back - *empty*
p_rgh	Left - *zeroGradient* Right – *fixedValue (1e5)* Top,Bottom, Front and Back - *empty*
alpha.water	Left - *zeroGradient* Right – *zeroGradient* Top,Bottom, Front and Back - *empty*

Table 2 Boundary Conditons (Sucking Interface)

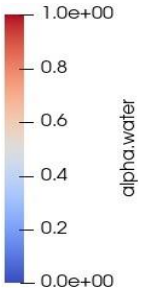
As for the mass transfer, an initial value of 0.005 kg/m^3s is assumed. Here, alpha.water is the phase indicator. The cells with value 1 are the ones with water, and the cells with value 0 are vapour cells.

4.1.3 Results:

The interMassFoam solver was executed. The following contours are those of the volume fraction, alpha.water. The liquid phase boils at the vapor–liquid interface, and the interface moves to the right due to the volume expansion of the vapor.

Figure 1 t=0 sec

Figure 2 t = 10 sec



From the contours, it is visible that there is a generation of vapour at the interface (alpha.vapour =0.5), which pushes the water to the right side. This verifies the transfer of mass from water to vapour.

4.2 Rising Bubble Simulation

Since the interfacial mass transfer has been implemented, the simulation of a rising vapour bubble can be modelled using the modified solver. The vapor bubble rises due to the buoyancy, and simultaneously shrinks as a consequence of condensation at its surface.

4.2.1 Initial and Boundary Conditions:

The fluid properties are the same as those in the sucking interface problems.

Field	Condition
U	Atmosphere– *pressureInletOutletVelocity (0 0 0)* Bottom – *noSlip* Walls - *slip* Front and Back - *empty*
p_rgh	Atmosphere – *totalPressure 0* Bottom – *zeroGradient* Walls - *zeroGradient* Front and Back - *empty*
alpha.vapour	Atmosphere, Bottom, Walls – *zeroGradient* Front and Back - *empty*

Table 3 Boundary conditions for Rising Bubble Case

Similar to the sucking interface problem, the isothermal condition is assumed. The properties are taken at 100°C. Here, alpha.vapour is the phase indicator.

4.2.2 Geometry and Mesh

The two-dimensional geometry is straightforward. At the first timestep, a vapour bubble of diameter 0.04 m is in the middle of a 0.5 m x 0.5 m domain, completely filled with water. The properties of vapour and water are the same as those in Table 1. Meshing is done using the blockMesh command. There are 40,000 hexahedral cells in the domain. The phases are defined using the setFields command.

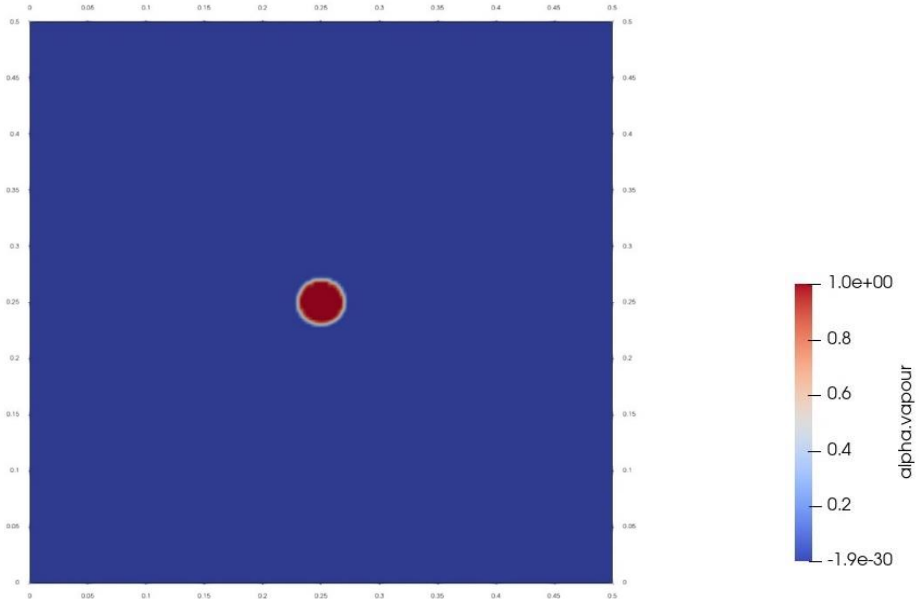


Figure 3 Domain for the rising bubble simulation

4.2.3 Case 1: $\dot{m} = 0.025$ kg/m^{3}s:

Two simulations were run with different values of $\dot{m}$. In case 1, the value of mass transfer rate is 0.025 kg/m^{3}s.Visibly, it can be observed that the bubble of the simulation with the higher $\dot{m}$ value has reduced in size. This clearly indicates the increased rate of mass transfer between the phases. The simulation is run for 0.6 sec.

4.2.4 Case 2: $\dot{m} = 0.05$ kg/m^{3}s:

In case 2, the value of $\dot{m}$ is 0.05 kg/m^{3}s, which is twice the value of the mass transfer rate in case 1. Therefore, a visible reduction in bubble size is expected at the same timestep as that of case 1. The contours of alpha.vapour for both cases are captured at 0.5 sec.

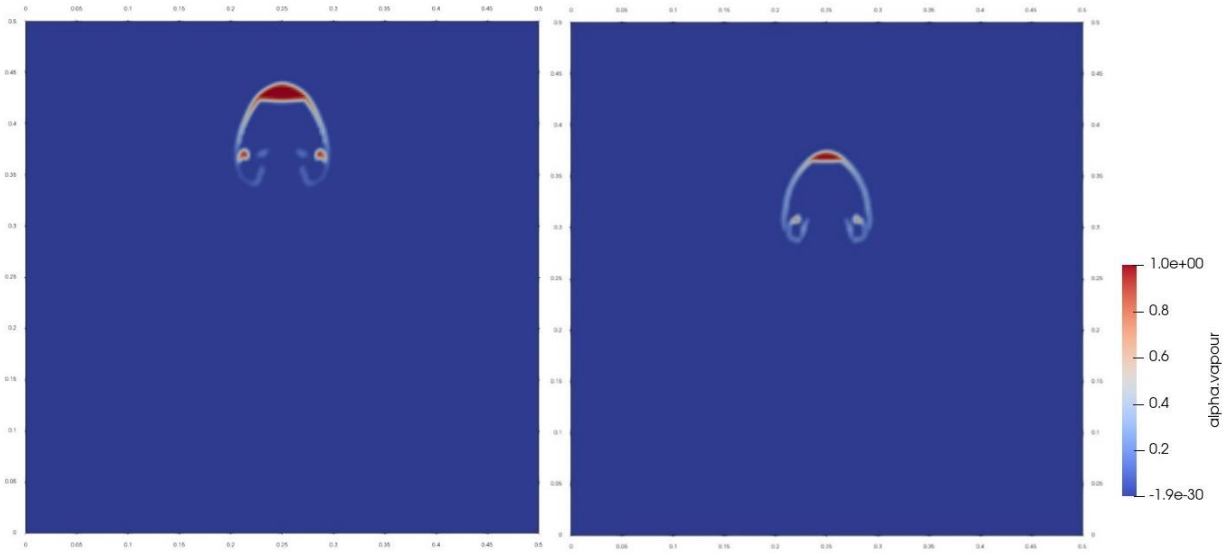


Figure 4 Case 1 ṁ= 0.025 kg/m³s

Figure 5 Case 2: ṁ = 0.05 kg/m³s

5. Results and Discussion

The new solver, interMassFoam, was modified, compiled and executed for two problems. Mass transfer between phases and phase change were validated using the sucking interface problem, in which vapour generation at the interface was noted. In the rising bubble simulation, two cases of condensing vapour bubbles with similar initial conditions, but with different mass transfer values, are simulated. The vapour bubble from the simulation with a higher mass transfer rate value is visibly smaller at the same timestep.

6. Conclusions:

The mass transfer rate was defined explicitly. This solver can be extended further, by defining a transport equation for temperature, from which the value of mass transfer can be calculated and implemented in OpenFOAM using custom libraries. This solver can be applied to a variety of problems, including and not limited to study of condensing and boiling flows, species tracking and biological applications. Specifically, this solver could be extended appropriately to be used in reactor safety studies.

Bibliography

1. G. Giustini, R. I. Issa, M. J. Bluck (2020) “A method for simulating interfacial mass transfer on arbitrary meshes” arXiv:2012.13759
2. N. Samkhaniani, M.R. Ansari, “Numerical simulation of superheated vapor bubble rising in stagnant liquid “, DOI:10.1007/s00231-017-2031-6
3. S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, L. Tobiska (2008) “Quantitative benchmark computations of two-dimensional bubble dynamics”, DOI:10.1002/fld.1934
4. H Jasak, 1996 “Error analysis and estimation for the finite volume method with applications to fluid flows” https://spiral.imperial.ac.uk/bitstream/10044/1/8335/1/Hrvoje_Jasak-1996-PhD-Thesis.pdf
5. OpenFOAM User Guide, https://www.openfoam.com/documentation/user-guide

Appendix:

1. How to install *interMassFoam*:

1. In your WSL/Linux terminal, go to the location where the *interMassFoam* files are downloaded
2. Run the command: *wmake*

The solver will be installed in the *$FOAM_USER_APPBIN* directory

2. How to run *interMassFoam*:

1. Since the solver cannot calculate the mass transfer rate, it has to be defined by the user in the massTransfer file in the cases /constant directory.
2. The mass transfer rate is in kg/m^{3}s. In ensure stability of the solver, realistic values need to be provided.
3. Run blockMesh command.
4. Run setFields command.
5. Run interMassFoam.
6. Post processing can be visualized on Paraview.

3. *interMassFoam* files - alphaSuSp.H:

```
0 zeroField Su;
1 zeroField divU;
2 volScalarField Sp ("Sp", ((1.0/rho1 * mdot)/(alpha1 + SMALL)));
3 forAll(alpha1,celli)
4     {
5    if(alpha1[celli]==0 && alpha1[celli]==1)
6         {
7          Sp[celli] = 0;
8         }
9     }
```

3. *interMassFoam* files - createFields.H:

```
43 /*---------------------------------------------*/
44 /*           Mass Transfer Directory           */
45 /*---------------------------------------------*/
46
47 Info<< "Reading massTransfer\n" << endl;
48
49 IOdictionary massTransfer
50 (
51     IOobject
52     (
53         "massTransfer",
54         runTime.constant(),
55         mesh,
56         IOobject::MUST_READ_IF_MODIFIED,
57         IOobject::NO_WRITE
58     )
59 );
60
61
62 Info<< "Reading mdot\n" << endl;
63
64 dimensionedScalar mdot
65 (
66     massTransfer.lookup("mdot")
67  );
68
69 /*---------------------------------------------*/
70 /*           Mass Transfer Directory           */
71 /*---------------------------------------------*/
```

5. *interMassFoam* files - pEqn.H:

```
41 while (pimple.correctNonOrthogonal())
42  {
43    fvScalarMatrix p_rghEqn
44    ( //source term added here
45  fvm::laplacian(rAUf, p_rgh) == fvc::div(phiHbyA) - mdot*(1.0/rho1 -1.0/rho2)
46      //end );
47
48        p_rghEqn.setReference(pRefCell, getRefCellValue(p_rgh, pRefCell));
49
50        p_rghEqn.solve();
51
52        if (pimple.finalNonOrthogonalIter())
53        {
54            phi = phiHbyA - p_rghEqn.flux();
55            p_rgh.relax();
56
57            U = HbyA + rAU()*fvc::reconstruct((phig - p_rghEqn.flux())/rAUf);
58            U.correctBoundaryConditions();
59            fvOptions.correct(U);
60        }
61    }
```