Rising Bubble with Mass Transfer using interFoam

Mano Prithvi Raj R

Velammal Engineering College, Chennai, Tamil Nadu (Graduated 2020)

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 vapor bubble undergoes condensation. Later this study can be extended to the temperature or pressure-based mass transfer variable term. For now, at the initial stage, a constant term is explicitly added in the continuity equation and validated with two different case studies.

1. Introduction

In multiphase modelling, the mass transfer is usually ignored when two fluids are immiscible. But at the micro-level, this kind of small physical phenomenon cannot be ignored. Hence, it becomes very crucial to add or include mass transfer terms in the system at the micro-level of physics. This mass transfer can be due to density difference, temperature difference, or pressure difference. Here, using the basic Volume of Fluid method of multiphase modelling, a constant mass transfer term is added in the continuity equation. An OpenFOAM solver interFoam is taken as a base solver and modified with C++ coding and wmake compiler.

The rising bubble simulation is a benchmark case study used to study interfacial flows and to validate the interface capturing ability of multiphase solvers. The interFoam solver uses the Volume of Fluid (VOF) method to capture the interface between the two phases. It solves problems that are incompressible and isothermal, and for fluids that are immiscible. The VOF method does not take interface mass transfer into consideration. However, there are a number of case studies, especially in the nuclear and energy industries, where condensing and

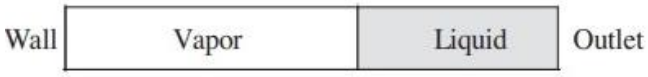
evaporating flows need to be validated. In these flows, the fluids are miscible and there is a transfer of mass between the phases.

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 equation 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. An appropriate mass transfer source term is added to the necessary equations and the values need to be defined explicitly, that is, they are not calculated by the solver directly. The value of mass transfer would be constant throughout the simulation and will be defined by the user prior to executing the solver. The new solver's ability to simulate mass transfer between two phases by solving the one-dimensional Sucking Interface problem, after which the solver will be used to run the Rising Bubble case.

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

2. Problem Statement

In order to validate phase change and interphase mass transfer in the new solver, we run the Sucking Interface problem. 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.



As for the rising bubble case, the simulation starts with a vapour bubble submerged in a column filled with water. During the simulation, the bubble should rise upwards with time. Isothermal conditions and miscible components are assumed. Since a constant mass transfer rate has been defined explicitly, it is expected that the bubble should mix with the fluid surrounding it during the movement. The solver works on the assumptions that flow is incompressible, isothermal and the fluids are Newtonian

3. Governing Equations:

3.1 interFoam Governing Equations:

3.1.1. Continuity Equation:

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

This is the global continuity equation. Mass is conserved throughout the simulation throughout the domain. However, if we take each phase individually, mass must be conserved with respect to each other. This means that each phase has its own continuity equation, which will be elaborated in Sec. 3.1.3, when the volume fraction equation is discussed.

3.1.2. Momentum Equation:

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

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 advection equation is the cornerstone of the VOF method. The native *interFoam* solver does not have source terms in the volume fraction equation. This is the equation in which mass transfer source term will be added. 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:

$$\rho = \alpha \rho_L + (1 - \alpha)\rho_V \quad (3)$$

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

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

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. As mentioned in Sec. 3.1.1, 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:

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

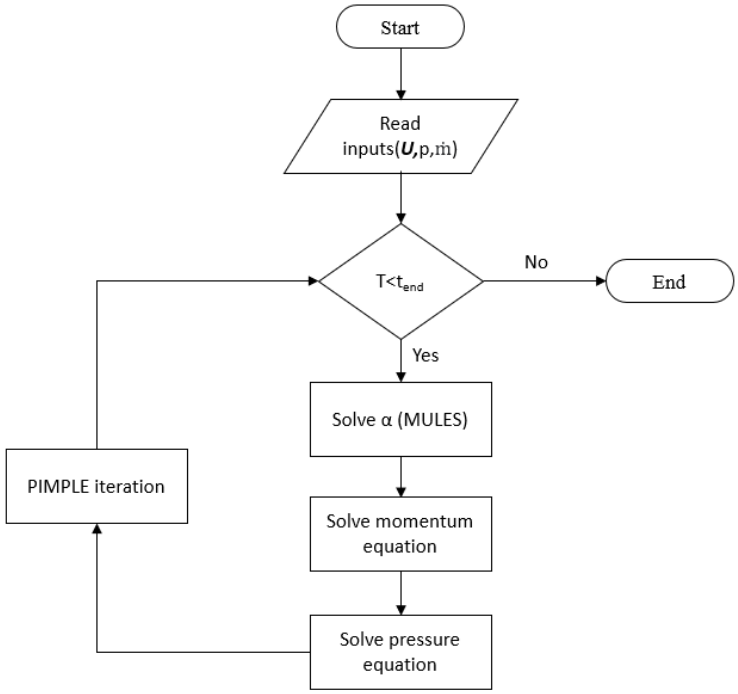
The first equation is used as the governing equation for the volume fraction equation.

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

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 Multidimensional Universal Limiter with Explicit Solution (MULES) algorithm, which in principle restricts the undershooting and overshooting of the volume fraction value.

4. Simulation Procedure

4.1 Working of interMassFoam:



The solution process of interMassFoam solver is shown in the flow chart above. The solver first reads the initial values. Time step is calculated based on the Courant–Friedrichs–Lewy (CFL) condition, to maintain solver stability. The source term is read by the solver and is added to the volume fraction equation, which is then solved using the MULES algorithm along with an inbuilt compression algorithm, to ensure boundedness and control smearing of the interface. Despite the fact that there are two phases present, we use the averaged physical properties and solve one momentum equation, rather than 2. The PIMPLE algorithm, a combination of PISO (Pressure Implicit with Splitting of Operator) and SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) was applied to solve the pressure-velocity coupling relationship. The pressure equation is solved and the velocity values are modified through the PIMPLE iteration, and this loop goes on until the end time is reached, or until the solution converges.

The discretization schemes are kept the same, even for the modified volume fraction equation, because these schemes work best with the MULES algorithm and therefore maintain the overall stability of the solution.

```
divSchemes
{
    div(rhoPhi,U)  Gauss upwind;
    div(phi,alpha)  Gauss interfaceCompression vanLeer 1;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
```

4.2 Sucking Interface Problem:

The Sucking Interface problem is one-dimensional. As elaborated in the previous sections, this problem is used to validate phase change and mass transfer at the interface. The right side of the domain is occupied by vapour and the left side is an outlet for the water to flow out the domain. It is assumed that the liquid is superheated, therefore, there should be a generation of vapour at the interface, which will push the liquid out. Gravity effects are neglected.

4.2.1 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 uniform hexahedral cells. The phases are defined using the setFields command.

4.2.2 Initial and Boundary Conditions:

Since the isothermal condition is assumed, the properties of the fluids are taken at 100°C. The left side is a wall and the right side is open for outlet. Flow is laminar.

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

Since the problem is one-dimensional, the boundary conditions for the front, back, top and bottom faces are declared as *empty* so that OpenFOAM solves it so.

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)

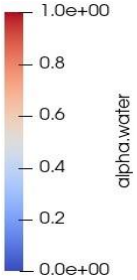
As for the mass transfer, an initial value of 0.005 kg/m^{3}s is assumed. Here, alpha.water is the phase indicator. The cells with value 1 are the ones with water, and the cells with value 0 are vapour cells.

4.2.3 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.

4.3 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.

4.3.1 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.

4.3.2 Geometry and Mesh

The rising bubble case is created as a two-dimensional problem in order to ensure that the simulation is not computationally expensive. The geometry is straightforward. 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.

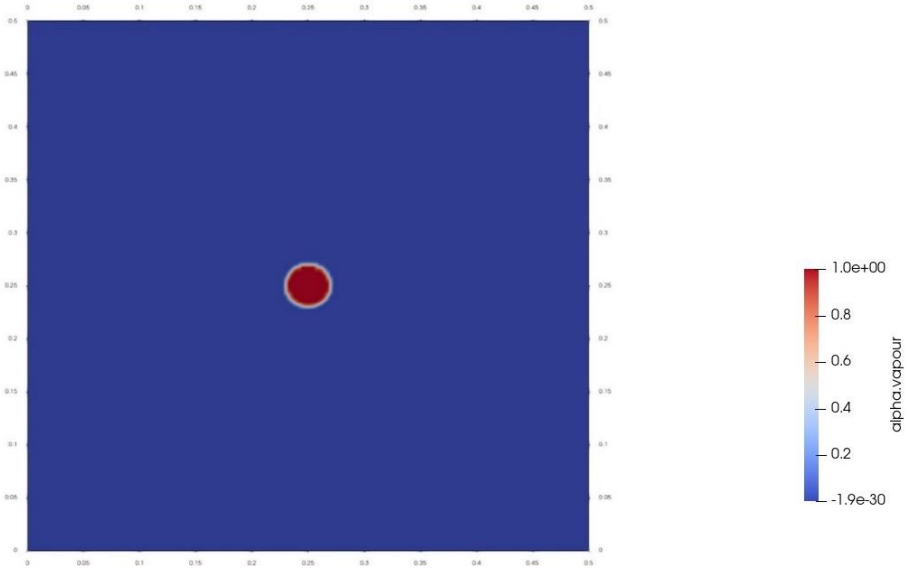


Figure 3 Domain for the rising bubble simulation

4.3.3 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.

4.3.4 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.

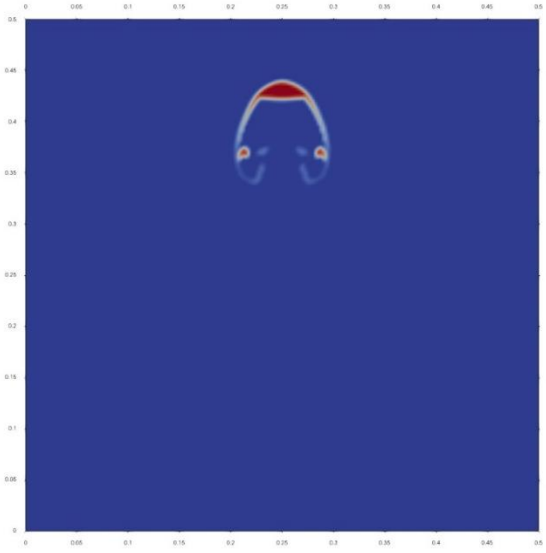


Figure 4 Case 1 $\dot{m}$= 0.025 kg/m^{3}s

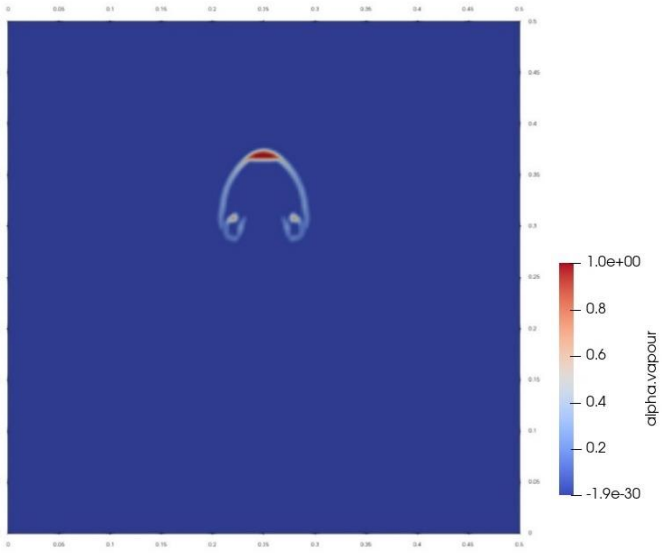


Figure 5 Case 2: $\dot{m}$ = 0.05 kg/m^{3}s

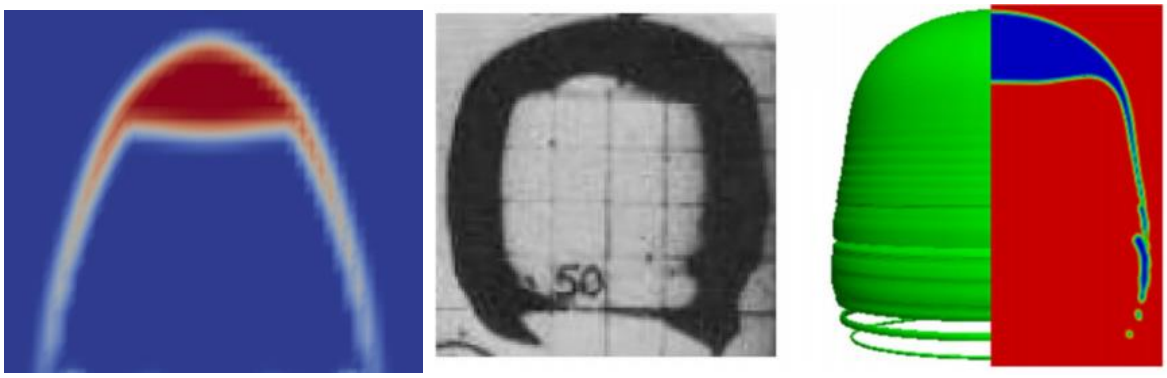


Figure 7 Bubble Shape from the present study, an experimental Study by Bhaga (7) and Bower (8) 3D Simulation

5. Results and Discussion

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 than the simulation with a smaller value of mass transfer rate at the same timestep. Moreover, the shape of the bubble captured and the skirting of the bubble from the first case conforms with an experimental study carried out by Bhaga (7) and a simulation by Bower (8).

6. Conclusions:

The new solver, interMassFoam, was modified, compiled and executed for two problems. A constant, user-defined mass transfer rate was added to the solver and mass transfer between phases and phase change were validated using the sucking interface problem, in which vapour generation at the interface was observed. The solver was then used to simulate two cases of rising vapour bubble, with different mass transfer rates. The results of the simulation were as expected. This shows that the solver has been extended to solve problems involving miscible fluids.

However, interMassFoam relies on the mass transfer rate to be 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, interMassFoam 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. Sato, Y., & Ničeno, B. (2013). "A sharp-interface phase change model for a mass conservative interface tracking method". doi:10.1016/j.jcp.2013.04.035
3. Kunkelmann, C., & Stephan, P. (2009). "CFD Simulation of Boiling Flows Using the Volume-of-Fluid Method within OpenFOAM." doi:10.1080/10407780903423908
4. N. Samkhaniani, M.R. Ansari,(2017) "Numerical simulation of superheated vapor bubble rising in stagnant liquid ", DOI:10.1007/s00231-017-2031-6
5. 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
6. 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
7. Bhaga D, Weber M (1981), "Bubbles in viscous liquids: shapes, wakes and velocities"
8. Amaya-Bower L, Lee T (2010) "Single bubble rising dynamics for moderate Reynolds number using lattice Boltzmann method."
9. 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^3s. 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    }
```