Cite as: Dueholm, B. C.: Implementation of a VoF solver with phase change for the simulation of internal cavitation and droplet breakup in injectors. In Proceedings of CFD with OpenSource Software, 2021, Edited by Nilsson. H., http://dx.doi.org/10.17196/OS_CFD#YEAR_2021

CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Håkan Nilsson

Implementation of a VoF solver with phase change for the simulation of internal cavitation and droplet breakup in injectors

Developed for OpenFOAM-v2106

Author:
Bjørn Christian Dueholm
Aalborg University

Peer reviewed by:
Mohsen Zarepour
Mohammad Hossein Arabnejad

Licensed under CC-BY-NC-SA, https://creativecommons.org/licenses/

Disclaimer: This is a student project work, done as part of a course where OpenFOAM and some other OpenSource software are introduced to the students. Any reader should be aware that it might not be free of errors. Still, it might be useful for someone who would like learn some details similar to the ones presented in the report and in the accompanying files. The material has gone through a review process. The role of the reviewer is to go through the tutorial and make sure that it works, that it is possible to follow, and to some extent correct the writing. The reviewer has no responsibility for the contents.

January 20, 2022

Learning outcomes

The main requirements of a tutorial in the course is that it should teach the four points: How to use it, The theory of it, How it is implemented, and How to modify it. Therefore the list of learning outcomes is organized with those headers.

The reader will learn:

How to use it:

- How to use the new solver `inter3PhaseChangeFoam`

The theory of it:

- The theory of `inter3PhaseChangeFoam` and volume of fluid in general.
- The theory of working with three phases with regards to the `SchnerrSauer` cavitation model.

How it is implemented:

- How the `SchnerrSauer` cavitation model is derived and implemented in `OpenFOAM`

How to modify it:

- How to merge `phaseChangeTwoPhaseMixture` and `immiscibleIncompressibleThreePhaseMixture` to make the framework for solving three phases with phase change between two of them.
- How to modify the `interPhaseChangeFoam` solver to have an additional phase.

Prerequisites

The reader is expected to have some knowledge about the following in order to get maximum benefit out of this report:

- How to run standard tutorials like damBreak and throttle tutorial.
- The formulation of `interPhaseChangeFoam`
- Basic object-oriented programming knowledge.
- How to customise a solver and do top-level application programming.

Contents

1 Introduction **5**

2 Theoretical background **6**
2.1 Volume of Fluid . . . 6
2.1.1 Governing equations . . . 6
2.2 `interPhaseChangeFoam` . . . 7
2.2.1 Volume fraction equation . . . 7
2.2.2 Pressure correction equation . . . 8
2.2.3 Cavitaion model . . . 8

3 Accommodating a non-condensing phase **12**
3.1 Volume of Fluid with an additional phase . . . 12
3.1.1 Volume fraction equation . . . 13
3.1.2 Pressure correction equation . . . 14

4 Implementation of `inter3PhaseChangeFoam` **15**
4.1 `phaseChangeThreePhaseMixtures` . . . 15
4.2 Modifying the solver . . . 18

5 Test Case **25**

A Mathematical derivations **31**
A.1 Density term for two phases . . . 31
A.2 Density term for three phase flows . . . 33
A.3 Condensation and evaporation terms for three phases . . . 34

B Modified codes to implement the third phase **37**
B.1 `alphaEqn.H` . . . 37

Nomenclature

Acronyms

CFD	Computational Fluid Dynamics
VoF	Volume of Fluid

English symbols

$\dot{m}$	Mass flow rate	kg/s
$\dot{V}$	Volumetric flow rate	m^3/s
A	Area	m^2
D_p	Term in pressure correction equation.	
g	Gravitational acceleration	$9.81 m/s^2$
$H[U]$	Term in pressure correction equation.	
n_0	Initial number of nuclei	$1/m^3$
N_B	Number of bubbles	–
p	Pressure	Pa
R	Radius of bubble	m
t	Time	s
U	Velocity vector	m/s
V	Volume	m^3

Greek symbols

α	Volume fraction	–
μ	Fluid dynamic viscosity	$kg \cdot s/m$
ρ	Fluid density	kg/m^3
τ	Deviatoric stress tensor	N/m^2

Superscripts

$*$	New time step

Subscripts

∞	Far field
B	Bubble
i	i^{th} entry
l	Liquid
nc	Non-condensing
p	Central Coefficient
sat	Saturation
v	Vapour

Chapter 1

Introduction

This tutorial contains a description of how to extend the `OpenFOAM` solver for cavitating flows called `interPhaseChangeFoam`. `interPhaseChangeFoam` is a solver for two phases, extended here to take a third non-condensing phase into account. This is done by implementing functionality from the solver `interMixingFoam` and heavily modifying the `alphaEqn.H` file of `interPhaseChangeFoam`. The SchnerrSauer cavitation model is derived and implemented for three-phase flows based on the works of W. Yuan and G. H. Schnerr [1]. This is done in order to investigate injector nozzles. In injector nozzles there are often cavitation inside the nozzle and air outside. To investigate the influence of cavitation on how a spray is formed, it is necessary to take all three phases into account.
This tutorial will go through the derivation and implementation of the `SchnerrSauer` model for two phases in Chapter 2 and afterwards the derivation and implementation for the `SchnerrSauer` model for three phases in Chapter 3. After all the mathematical derivations, the tutorial will show how to implement the changes necessary to take three phases into account and implement the three phase `SchnerrSauer` model in Chapter 4. Lastly, the new solver called `inter3PhaseChangeFoam` will be demonstrated on a small test case in Chapter 5.

Chapter 2

Theoretical background

This chapter contains a description of the theory behind `interPhaseChangeFoam`, the Volume of Fluid method and some challenges in working with three phases instead of two.

2.1 Volume of Fluid

The Volume of Fluid (VoF) method is an Euler-Euler approach where one set of momentum and continuity equations are solved using the effective fluid properties. It is assumed that each phase is incompressible and isothermal. Transport equations are solved for each phase together with the momentum and continuity equation. The variable α is used for each phase fraction. It is defined as

$$\alpha_i = \frac{V_i}{V}. \tag{2.1}$$

Here V is the volume of each cell, and i denotes the ith phase. α is used to track different phases as they are solved as one mixture. The phase fractions are summed up to unity as

$$\sum_{i=1}^{2} \alpha_i = 1. \tag{2.2}$$

The properties of the mixture is also blended between the phases as

$$\rho = \sum_{i=1} \alpha_i \rho_i, \qquad \mu = \sum_{i=1} \alpha_i \mu_i, \tag{2.3}$$

where ρ is the density and μ is the dynamic viscosity.

Now that the phase fraction variable has been defined, the next step is the governing equations.

2.1.1 Governing equations

The continuity and momentum equations are given as

$$\nabla \cdot U = -\frac{1}{\rho}\frac{d\rho}{dt}, \tag{2.4}$$

$$\frac{\partial(\rho U)}{\partial t} + \nabla \cdot (\rho U \otimes U) = -\nabla \hat{p} + \nabla \cdot \tau + f_\sigma - g \cdot x \nabla \rho, \tag{2.5}$$

respectively. Here $\hat{p}$ is modified pressure, which is constructed by removing the hydrostatic part of the static pressure p. τ is the deviatoric stress tensor, f_σ is the surface tension force, g is the acceleration of gravity, and $g \cdot x \nabla \rho$ is a term added as a consequence of using $\hat{p}$.

The transport equation of the volume fraction of each phase is written as

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l}, \tag{2.6}$$

$$\frac{\partial \alpha_v}{\partial t} + \nabla \cdot (\alpha_v U) = -\frac{\dot{m}}{\rho_v}. \tag{2.7}$$

Here $\dot{m}$ is mass transfer rate between the two phases. The mass transfer term is calculated in `OpenFOAM` by a cavitation model. In this project the `SchneerSauer` model is used.

2.2 interPhaseChangeFoam

This section illustrates how the implemented version of source terms in `interPhaseChangeFoam` have been derived. A large part of sections 2.2.1 and 2.2.2 is based the PhD thesis of A. Asnaghi [2].

2.2.1 Volume fraction equation

Because mass is transferred between the two phases, the divergence of velocity is no longer zero. Equation 2.6 and 2.7 equations can be rewritten as

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l}, \tag{2.8}$$

$$\frac{d \alpha_l}{dt} + \alpha_l \nabla \cdot (U) = \frac{\dot{m}}{\rho_l}, \tag{2.9}$$

where d denotes total derivative throughout the report. By Adding Equation 2.6 and 2.7 on the rewritten form, it can be shown that the divergence of velocity is different from zero as

$$\frac{d}{dt}(\alpha_l + \alpha_v) + (\alpha_l + \alpha_v)\nabla \cdot U = \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\dot{m}, \tag{2.10}$$

$$\nabla \cdot U = \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\dot{m} \tag{2.11}$$

If there were no phase change source terms, the divergence of velocity would have been 0.
A little trick is performed on the volume fraction equations to make solution more stable. The volume fraction equation for liquid can be written as

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l} - \alpha_l \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\dot{m} + \alpha_l \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\dot{m}. \tag{2.12}$$

Essentially 0 is added to the right hand side of Equation 2.6. Substituting Equation 2.11 into Equation 2.12 it becomes

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \left(\frac{1}{\rho_l} - \alpha_l \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\right)\dot{m} + \alpha_l \nabla \cdot U. \tag{2.13}$$

With the first term on the right hand side being defined as $\left(\frac{1}{\rho_l} - \alpha_l \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\right)\dot{m} = \dot{V}$ and the mass transfer rate being split up into two terms, one for condensation and one for vaporisation as

$$\dot{m} = \alpha_l \dot{m}_v + (1 - \alpha_l)\dot{m}_c = \alpha_l (\dot{m}_v - \dot{m}_c) + \dot{m}_c, \tag{2.14}$$

the transport equation becomes

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \left(\nabla U + \dot{V}_v - \dot{V}_c\right)\alpha_l + \dot{V}_c. \tag{2.15}$$

This is the form that is implemented in interPhaseChangeFoam as shown below. This makes it possible to ass a part of source term to the diagonal part of the matrix, to stabilise the solution. The vapour fraction is afterwards obtained from Equation 2.2.

alphaEqn.H

```
fvScalarMatrix alpha1Eqn
(
    fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
  + fv::gaussConvectionScheme<scalar>
    (
        mesh,
        phi,
        upwind<scalar>(mesh, phi)
    ).fvmDiv(phi, alpha1)
  - fvm::Sp(divU, alpha1)
 ==
    fvm::Sp(vDotvmcAlphal, alpha1)
  + vDotcAlphal
);
```

2.2.2 Pressure correction equation

This section will not go through the entire process of deriving the pressure correction equation from the continuity and momentum equations. If the reader is interested in this, please refer to A. Asnaghi [2]. As shown by A. Asnaghi [2], with some manipulation it is possible to get the pressure correction equation on the form

$$\nabla \cdot H[U^*]_p + \nabla \cdot (-D_p p) = \nabla \cdot U_p^*. \tag{2.16}$$

For the definition of D_p and $H[U]$ please refer to H. Jasak [3]. As shown earlier, the divergence of U is the mass transfer rate term, so Equation 2.16 can be written as

$$\nabla \cdot H[U^*]_p + \nabla \cdot (-D_p p) = \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \dot{m}. \tag{2.17}$$

The mass transfer term is then split by defining $\dot{m}_p = \frac{\dot{m}}{(p - p_{sat})}$, and $\dot{m}_p$ can again be split into two terms as $\dot{m}_p = \dot{m}_{pv} - \dot{m}_{pc}$. Considering $\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \dot{m}_p = \dot{V}_p$, the same manipulation that was performed on the volume fraction equation is done on the pressure correction equation to yield

$$\nabla \cdot H[U^*]_p + \nabla \cdot (-D_p p) = \left(\dot{V}_{pv} - \dot{V}_{pc}\right) p - \left(\dot{V}_{pv} - \dot{V}_{pc}\right) p_{sat}. \tag{2.18}$$

This equation is implemented in the `interPhaseChangeFoam` as below.

pEqn.H

```
fvScalarMatrix p_rghEqn
(
    fvc::div(phiHbyA) - fvm::laplacian(rAUf, p_rgh)
  - (vDotvP - vDotcP)*(mixture->pSat() - rho*gh)
  + fvm::Sp(vDotvP - vDotcP, p_rgh)
);
```

Here the laplacian operator is used because D_p already contains ∇, Refer to A. Asnaghi [2] for the definition.

2.2.3 Cavitaion model

To close the mathematical system above, a model is needed for the mass transfer rates. In this project the Schnerr and Sauer [4] model is used. The model is based on the idea, that for cavitation to take place, it has to have a place to begin. The model defines the starting place, as an amount of small bubble nuclei, present in the liquid phase. The mass transfer rate terms are quantified in terms of growth and collapse of these nuclei. Looking at the code in `OpenFOAM`, the two mass transfer terms are defined as

$$\dot{m}_{ac} = C_c \alpha_l \frac{3\rho_l \rho_v}{\rho R_B} \sqrt{\frac{2}{3\rho_l}} \sqrt{\frac{1}{|p - p_{sat}|}} \cdot max(p - p_{sat}, 0), \tag{2.19}$$

$$\dot{m}_{av} = C_v \left(1 + \alpha_{nuc} - \alpha_l\right) \frac{3\rho_l \rho_v}{\rho R_B} \sqrt{\frac{2}{3\rho_l}} \sqrt{\frac{1}{|p - p_{sat}|}} \cdot min(p - p_{sat}, 0). \tag{2.20}$$

These equations are used to represent the mass transfer rates in Equation 2.15. The terms α_{nuc} and R_B, that represent the volume fraction of the initial nuclei and the bubble radius respectively, are defied as

$$\alpha_{nuc} = \frac{V_{nuc}}{1 + V_{nuc}} = \frac{\frac{\pi n_0 d_{nuc}^3}{6}}{1 + \frac{\pi n_0 d_{nuc}^3}{6}}, \tag{2.21}$$

$$R_B = \sqrt[3]{\frac{3}{4\pi n_0} \frac{1 + \alpha_{nuc} - \alpha_l}{\alpha_l}}. \tag{2.22}$$

In Equation 2.21 and Equation 2.22, n_0 is the initial number of nucleation sites per cubic meter of liquid and d_{nuc} is the diameter of the initial small nuclei. α_{nuc} is used as a term to start the cavitation, because the term $1 - \alpha_l$ would just be zero in the beginning when no vapour phase is present. α_{nuc} is then introduced as $1 + \alpha_{nuc} - \alpha_l$. To obtain the final implementable mass transfer rate terms, a term for the mass transfer rate is needed. Such a term is derived in Appendix A.1. The final term ends from the appendix is

$$\dot{m} = \rho_v \frac{d\alpha_v}{dt} \left(\frac{\rho_l}{\rho}\right) = \frac{d\alpha_v}{dt} \left(\frac{\rho_l \rho_v}{\rho}\right). \tag{2.23}$$

The term R_B is found by looking at the definition of α_v in the works of G. H. Schnerr and J. Sauer [4] as

$$\alpha_v = \frac{V_v}{V_{cell}} = \frac{N_{bubbles} \frac{4}{3}\pi R^3}{V_v + V_l} = \frac{n_0 V_l \frac{4}{3}\pi R^3}{n_0 V_l \frac{4}{3}\pi R^3 + V_l} = \frac{n_0 \frac{4}{3}\pi R^3}{n_0 \frac{4}{3}\pi R^3 + 1} = \frac{\alpha_v}{\alpha_l + \alpha_v}. \tag{2.24}$$

From Equation 2.23, it is apparent that a term for the derivative of the vapour volume fraction is needed. Looking at the definition of vapour volume fraction, it is apparent that the denominator is constant, so differentiating this equation regarding time yields

$$\frac{d\alpha_v}{dt} = \frac{n_0 \frac{d\left(\frac{4}{3}\pi R^3\right)}{dt}}{n_0 \frac{4}{3}\pi R^3 + 1}. \tag{2.25}$$

The derivative of volume for a sphere is

$$\frac{dV}{dt} = A \frac{dR}{dt}. \tag{2.26}$$

The derivative of α_v for a spherical bubble can be written as

$$\frac{d\alpha_v}{dt} = \frac{n_0 \frac{d\left(\frac{4}{3}\pi R^3\right)}{dt}}{n_0 \frac{4}{3}\pi R^3 + 1} = \frac{A}{V} \frac{dR}{dt} = \frac{3}{R} \frac{dR}{dt}. \tag{2.27}$$

Equation 2.23 can be rewritten as

$$\dot{m} = \frac{d\alpha_v}{dt} \left(\frac{\rho_l \rho_v}{\rho}\right) = \frac{3\rho_l \rho_v}{\rho R} \frac{dR}{dt}, \tag{2.28}$$

where R is determined from Equation 2.24 as

$$\frac{\alpha_v}{\alpha_l + \alpha_v} = \frac{n_0 \frac{4}{3}\pi R^3}{n_0 \frac{4}{3}\pi R^3 + 1}. \tag{2.29}$$

Rearranging the equation results in

$$R = \sqrt[3]{\frac{3}{4\pi n_0}\frac{\alpha_v}{\alpha_l}}. \tag{2.30}$$

Implementing $\alpha_n uc$ to make the occurrence of the cavitation possible in this equation yields

$$R = \sqrt[3]{\frac{3}{4\pi n_0}\frac{1 + \alpha_{nuc} - \alpha_l}{\alpha_l}}. \tag{2.31}$$

A term for the time rate of change of the radius of the bubble is needed as can also be seen in Equation 2.28. OpenFOAM uses the formulation presented by G. H. Schnerr and J. Sauer as

$$\frac{dR}{dt} = sign(p(R) - p_\infty)\sqrt{\frac{2}{3}\frac{p(R) - p_\infty}{\rho_l}}. \tag{2.32}$$

Here $p(R)$ is the pressure at the bubble boundary and p_∞ is the pressure far away from the bubble. Equation 2.32 is actually the Rayleigh-Plesset equation. If $p(R)$ is set to the saturation pressure and p_∞ is set to the pressure in the cell, Equation 2.32 can be rewritten as

$$\frac{dR}{dt} = -(p - p_{sat})\sqrt{\frac{2}{3}\frac{1}{\rho_l|p - p_{sat}|}} = \begin{cases} min(p - p_{sat}, 0)\sqrt{\frac{2}{3}\frac{1}{\rho_l|p-p_{sat}|}} & \text{if } p < p_{sat} \\ max(p - p_{sat}, 0)\sqrt{\frac{2}{3}\frac{1}{\rho_l|p-p_{sat}|}} & \text{if } p > p_{sat} \end{cases}. \tag{2.33}$$

The minus sign in this Equation is thrown away because of reasons described in appendix A.1. If Equation 2.33 is substituted in Equation 2.28 and multiplied with the corresponding cavitation constant and volume fraction, the complete mass transfer rate terms in Equation 2.19 and Equation 2.20 are obtained.
The implementation of α_{nuc}, the reciprocal of R_B, and most of the mass transfer term, can be seen in the code below.

SchnerrSauer.C

```
Foam::tmp<Foam::volScalarField>
Foam::phaseChangeTwoPhaseMixtures::SchnerrSauer::rRb
(
    const volScalarField& limitedAlpha1
) const
{
    return pow
    (
        ((4*constant::mathematical::pi*n_)/3)
       *limitedAlpha1/(1.0 + alphaNuc() - limitedAlpha1),
        1.0/3.0
    );
}


Foam::dimensionedScalar
Foam::phaseChangeTwoPhaseMixtures::SchnerrSauer::alphaNuc() const
{
    dimensionedScalar Vnuc = n_*constant::mathematical::pi*pow3(dNuc_)/6;
    return Vnuc/(1 + Vnuc);
}


Foam::tmp<Foam::volScalarField>
```

```
Foam::phaseChangeTwoPhaseMixtures::SchnerrSauer::pCoeff
(
    const volScalarField& p
) const
{
    volScalarField limitedAlpha1(min(max(alpha1_, scalar(0)), scalar(1)));
    volScalarField rho
    (
        limitedAlpha1*rho1() + (scalar(1) - limitedAlpha1)*rho2()
    );

    return
        (3*rho1()*rho2())*sqrt(2/(3*rho1()))
       *rRb(limitedAlpha1)/(rho*sqrt(mag(p - pSat()) + 0.01*pSat()));
}


Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::phaseChangeTwoPhaseMixtures::SchnerrSauer::mDotAlphal() const
{
    const volScalarField& p = alpha1_.db().lookupObject<volScalarField>("p");
    volScalarField pCoeff(this->pCoeff(p));

    volScalarField limitedAlpha1(min(max(alpha1_, scalar(0)), scalar(1)));

    return Pair<tmp<volScalarField>>
    (
        Cc_*limitedAlpha1*pCoeff*max(p - pSat(), p0_),

        Cv_*(1.0 + alphaNuc() - limitedAlpha1)*pCoeff*min(p - pSat(), p0_)
    );
}
```

After this `mDotAlphal()` is used in the `phaseChangeTwoPhaseMixture` as shown below.

phaseChangeTwoPhaseMixture.C

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::phaseChangeTwoPhaseMixture::vDotAlphal() const
{
    volScalarField alphalCoeff(1.0/rho1() - alpha1_*(1.0/rho1() - 1.0/rho2()));
    Pair<tmp<volScalarField>> mDotAlphal = this->mDotAlphal();

    return Pair<tmp<volScalarField>>
    (
        alphalCoeff*mDotAlphal[0],
        alphalCoeff*mDotAlphal[1]
    );
}
```

`vDotAlphal()` is used directly in the volume fraction equation. Now that the cavitation model and the governing equations have been derived, the following Chapter will expand on how the cavitation model changes, when an additional non-condensing phase is taken into account.

Chapter 3

Accommodating a non-condensing phase

In this chapter the changes necessary for a third phase to be taking into account will be described. The changes to the `SchnerrSauer` cavitaion model is based on W. Yuan and G. H. Schnerr [1], but is described here in greater detail. If the subject is of interest, it is highly advised to read the aforementioned paper.

3.1 Volume of Fluid with an additional phase

Relating to section 2.1, the definition of alpha is still as

$$\alpha_i = \frac{V_i}{V}. \tag{3.1}$$

The volume fraction is now in three parts, but the sum still has to be equal as

$$\sum_{i=1}^{3} \alpha_i = 1. \tag{3.2}$$

Having three phases, l, v, nc are used as the subscripts of α to represent liquid, vapor, and non-condensating gas respectively. The physical properties of the mixture are evaluated as

$$\rho = \sum_{i=1} \alpha_i \rho_i, \qquad \mu = \sum_{i=1} .\alpha_i \mu_i \tag{3.3}$$

The system of volume fraction equations now contains an additional equation for the non-condensing phase as 3.4-3.6.

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l}, \tag{3.4}$$

$$\frac{\partial \alpha_v}{\partial t} + \nabla \cdot (\alpha_v U) = -\frac{\dot{m}}{\rho_v}, \tag{3.5}$$

$$\frac{\partial \alpha_{nc}}{\partial t} + \nabla \cdot (\alpha_{nc} U) = 0. \tag{3.6}$$

Summation of the equations again yields the a similar term as for two phase flows

$$\begin{aligned} \frac{d}{dt}(\alpha_l + \alpha_v + \alpha_{nc}) + (\alpha_l + \alpha_v + \alpha_{nc}) \nabla \cdot U &= \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \dot{m}, \\ \nabla \cdot U &= \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \dot{m}. \end{aligned} \tag{3.7}$$

Again it becomes necessary to determine a model to calculate mass transfer between the liquid and vapour phases. Here the modified Schnerr and Sauer model is used. The modification was done by W. Yuan and G. H. Schnerr [1]. In the modified Schnerr and Sauer model, it is again assumed, that there is a number of nucleation sites with initial diameter d_{nuc} in the liquid phase. That results in

$$\alpha_v = \alpha_l n_0 \frac{4}{3}\pi R^3. \tag{3.8}$$

Equation 2.26 that was used to simplify Equation 2.24 can not be used here, since there would be an additional contribution from α_{nc} in the denominator that would not cancel out when being divided by V_l. For three phases Equation 3.8 is instead used to obtain the terms needed for the mass transfer rates.

Very similar to what was done for the mass transfer rates for two phase, it can be shown that the divergence of velocity and the mass transfer rates are

$$\frac{d\alpha_v}{dt}\frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} = \nabla \cdot U, \tag{3.9}$$

$$\frac{d\alpha_v}{dt}\left(\frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right) = \dot{m}. \tag{3.10}$$

Derivation of these equations can be found in Appendix A section A.2. Using the definition of total derivative and Equation 3.8, the rate of change of the vapour volume fraction is calculated. Equation 3.8 is used in two different forms as a basis to derive the time rate of change of α_v in both directions as

$$\alpha_v = \alpha_l n_0 \frac{4}{3}\pi R^3, \qquad \alpha_l = \frac{3\alpha_v}{4\pi n_0 R^3}. \tag{3.11}$$

Taking the total derivative of Equation 3.11 and doing a lot manipulation, the rate of evaporation and condensation are

$$\frac{d\alpha_v}{dt} = \frac{3\alpha_v \frac{dR}{dt}}{\left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right)\frac{4}{3}\pi n_0 R^4 + R}, \tag{3.12}$$

$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt}}{1 + n_0 \frac{4}{3}\pi R^3 \left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right)}, \tag{3.13}$$

Respectively. The mathematical manipulation done is described in Appendix A section A.3. Here Equation 3.12 and 3.13 are respectively for condensation and evaporation. These terms are similar to what is obtained by W. Yuan and G. H. Schnerr [1]. At this point two relations for the radius and for the time rate of change of radius is needed to close the mathematical system. Similar to the system for two phases, the work by G. H. Schnerr and J. Sauer [4] are used as

$$R = \sqrt[3]{\frac{3}{4\pi n_0}\frac{1 + \alpha_{nuc} - \alpha_l - \alpha_{nc}}{\alpha_l}}, \tag{3.14}$$

$$\frac{dR}{dt} = -(p - p_{sat})\sqrt{\frac{2}{3}\frac{1}{\rho_l |p - p_{sat}|}} = \begin{cases} min(p - p_{sat}, 0)\sqrt{\frac{2}{3}\frac{1}{\rho_l|p-p_{sat}|}} & \text{if } p < p_{sat} \\ max(p - p_{sat}, 0)\sqrt{\frac{2}{3}\frac{1}{\rho_l|p-p_{sat}|}} & \text{if } p > p_{sat} \end{cases}. \tag{3.15}$$

The minus in the in Equation 3.15 is thrown away, as described in Appendix A.

3.1.1 Volume fraction equation

A bit of further manipulation is needed to improve the stability of the implemented system. The first observation is, that $\frac{d\alpha_v}{dt}$ must be the sum of the two terms as

$$\frac{d\alpha_v}{dt} = \alpha_v \left.\frac{d\alpha_v}{dt}\right|_{cond} + \alpha_l \left.\frac{d\alpha_v}{dt}\right|_{evap} = \left(\left.\frac{d\alpha_v}{dt}\right|_{evap} - \left.\frac{d\alpha_v}{dt}\right|_{cond}\right)\alpha_l + \left.\frac{d\alpha_v}{dt}\right|_{cond}(1 - \alpha_{nc}). \tag{3.16}$$

Here $\frac{d\alpha_v}{dt}\Big|_{cond}$ and $\frac{d\alpha_v}{dt}\Big|_{evap}$ are calculated using Equation 3.12 and 3.13, but excluding α_l and α_v. Doing the manipulation shown in Equation 3.16 allows to descritise everything multiplied with α_l implicitly and the rest explicitly. This aids the stability of the system. The idea is the same as what is done in `OpenFOAM` in `interPhaseChangeFoam`, as was described in Section 2.2. At this point the same trick used in Section 2.2 that added and subtracted $\alpha_l \nabla \cdot U$ on the right hand side of the transport equation is used again, so the transport equation for α_l becomes

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (U\alpha_l) = \frac{\rho_v}{\rho + \alpha_{nc}(\rho_l - \rho_n c)} \frac{d\alpha_v}{dt} - \alpha_l \frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_n c)} \frac{d\alpha_v}{dt} + \alpha_l \nabla \cdot U.$$

Rearranging this equation yields

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (U\alpha_l) = \frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_n c)} \frac{d\alpha_v}{dt} \left(\frac{1}{\rho_l} - \alpha_l \left(\frac{1}{\rho_l} - \frac{1}{\rho_v} \right) \right) + \alpha_l \nabla \cdot U \tag{3.17}$$

Please note that in Equation 3.17 the $\frac{d\alpha_v}{dt}$, which comes from Equation 3.9 does contain a minus, hence the switch in the innermost bracket. $\frac{d\alpha_v}{dt}$ is here discretised as shown in Equation 3.16. Equation 3.17 is the final form, that will be implemented for α_l.

3.1.2 Pressure correction equation

For the pressure correction equation, the implementation of `interPhaseChangeFoam` will again be used as inspiration. The mass transfer rate was defined as $\dot{m}_p = \frac{\dot{m}}{(p - p_{sat})}$ and $\dot{m}_p$ was defined as the evaporation minus the condensation, so $\dot{m}_p = \dot{m}_{pv} - \dot{m}_{pc}$ and finally $\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \dot{m}_p$ was used to get volume transfer rates, as it is more efficient to solve for volume flow in the finite volume approach. This means that the final equation for the pressure correction becomes.

$$\nabla \cdot H[U^*]_p + \nabla \cdot (-D_p p) = \left(\dot{V}_{pv} - \dot{V}_{pc}\right) p - \left(\dot{V}_{pv} - \dot{V}_{pc}\right) p_{sat}, \tag{3.18}$$

Where $\dot{m}_{pv}$ and $\dot{m}_{pc}$ are defined as

$$\dot{m}_{pc} = \frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} \frac{3\alpha_v}{\left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right) \frac{4}{3}\pi n_0 R^4 + R} pos0(p - p_{sat}, 0) \sqrt{\frac{2}{3} \frac{1}{\rho_l |p - p_{sat}|}}, \tag{3.19}$$

$$\dot{m}_{pc} = -\frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} \frac{\alpha_l 4 n_0 \pi R^2}{1 + n_0 \frac{4}{3}\pi R^3 \left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right)} neg(p - p_{sat}, 0) \sqrt{\frac{2}{3} \frac{1}{\rho_l |p - p_{sat}|}}. \tag{3.20}$$

The `pos0` and `neg` are Boolean functions that return 1 if the contents are respectively positive and negative. Otherwise they return 0. In Equation 3.20, a minus sign is added, since the `neg` function returns 1 if the input is negative, so the minus is added in front.
The following sections will go into how to modify the current `OpenFOAM` code, to accommodate the third phase in `interPhaseChangeFoam`.

Chapter 4

Implementation of `inter3PhaseChangeFoam`

This chapter describes how the solver for three phases is implemented together with the modified cavitation model.

4.1 `phaseChangeThreePhaseMixtures`

First a library is needed to handle calculation of the mixture properties, contact angle and so on. Luckily there is a library in a solver called `interMixingFoam` that has these capabilities. `interMixingFoam` is a solver for three incompressible fluids, two of which are immiscible using VOF based interface capturing. The goal is now to mix the classes called `phaseChangeTwoPhaseMixtures` and `immiscibleIncompressibleThreePhaseMixture` to get a class that can handle three phases and phase change.

Before listing all the changes necessary, it is advised to have the supplied implementation readily nearby. A lot of the copy pasting can be done from the files supplied, since it then keeps the structure indentation wise. First `interMixingFoam` and `interPhaseChangeFoam` are copied to the project directory. Assuming the terminal is opened in the project directory and `v2106` is sourced, this is done with the following lines.

copying the libraries

```
cp -r $FOAM_APP/solvers/multiphase/interPhaseChangeFoam .
cp -r $FOAM_APP/solvers/multiphase/interFoam/interMixingFoam .
```

Following is a list of actions to be done to compile the library.

- Delete `overInterPhaseChangeDyMFoam` and `interPhaseChangeDyMFoam` from the `interPhaseChangeFoam` directory.

  Deleting unnecessary solvers

  ```
  rm -r interPhaseChangeFoam/overInterPhaseChangeDyMFoam/
  rm -r interPhaseChangeFoam/interPhaseChangeDyMFoam/
  ```

- Copy the `incompressibleThreePhaseMixture` and `threephaseInterfaceProperties` folders to the `interPhaseChangeFoam` directory from the `interMixingFoam` directory. The `interMixingFoam` directory can now be deleted.

  Moving Libraries

  ```
  mv interMixingFoam/incompressibleThreePhaseMixture/ interPhaseChangeFoam/
  ```

```
mv interMixingFoam/threePhaseInterfaceProperties/ interPhaseChangeFoam/
rm -r interMixingFoam/
```

- Go into the `interPhaseChangeFoam` directory and rename the `phaseChangeTwoPhaseMixtures` directory to `phaseChangeThreePhaseMixtures`.

Renaming Library

```
cd interPhaseChangeFoam
mv phaseChangeTwoPhaseMixtures/ phaseChangeThreePhaseMixtures/
```

- Go into the `phaseChangeThreePhaseMixtures` directory and delete the `Kunz` and `Merkle` folder, then rename the `phaseChangeTwoPhaseMixture` folder to `phaseChangeThreePhaseMixture`.

Deleting and renaming

```
cd phaseChangeThreePhaseMixtures/
rm -r Kunz/
rm -r Merkle/
mv phaseChangeTwoPhaseMixture/ phaseChangeThreePhaseMixture/
```

- Go into the `phaseChangeThreePhaseMixture` and rename the three files to follow the pattern from above, so changing Two to Three in their names.

Deleting and renaming

```
cd phaseChangeThreePhaseMixture/
mv phaseChangeTwoPhaseMixture.C phaseChangeThreePhaseMixture.C
mv phaseChangeTwoPhaseMixtureNew.C phaseChangeThreePhaseMixtureNew.C
mv phaseChangeTwoPhaseMixture.H phaseChangeThreePhaseMixture.H
```

- Replace every occurrence of the word "Two" to the word "Three" in the three files in the phasChangeThreePhaseMixture folder. This is a lot of places, so use some form of search and replace if not working in linux.

Remaining in the three files

```
sed -i 's/Two/Three/g' *
```

- Open phaseChangeThreePhaseMixture.H and edit it, so the contents are changed to the following.
  Add the following line to the includes.

`phaseChangeThreePhaseMixture.H`

```
#include "threePhaseInterfaceProperties.H"
```

Change the inheritance so it becomes.

`phaseChangeThreePhaseMixture.H`

```
class phaseChangeThreePhaseMixture
:
    public incompressibleThreePhaseMixture,
    public threePhaseInterfaceProperties
```

Change the `correct()` function to the following.

phaseChangeThreePhaseMixture.H

```
        //- Correct the phaseChange model, get calcNu()
        virtual void correctNu()
        {
            incompressibleThreePhaseMixture::correct();
        }

        //- Correct the interface properties, gets calculateK()
        virtual void correctInterface()
        {
            threePhaseInterfaceProperties::correct();
        }
```

threePhaseInterfaceProperties is added as inheritance, so the library now has access to the interface functionality. One problem arises by doing this, which is, that both libraries have a correct() function, so the new library's correct() function is split in two functions that call the respective inherited correct() function. They are now called correctNu() and correctInterface().

- Open the phaseChangeThreePhaseMixture.C and change the constructor to the following.

phaseChangeThreePhaseMixture.C

```
Foam::phaseChangeThreePhaseMixture::phaseChangeThreePhaseMixture
(
    const word& type,
    const volVectorField& U,
    const surfaceScalarField& phi
)
:
    incompressibleThreePhaseMixture(U, phi),
    threePhaseInterfaceProperties
    (
        static_cast<incompressibleThreePhaseMixture&>(*this)
    ),
    phaseChangeThreePhaseMixtureCoeffs_(optionalSubDict(type + "Coeffs")),
    pSat_("pSat", dimPressure, *this)
{}
```

This adds the constructor from immiscibleIncompressibleThreePhaseMixture.C to the library, so the functionality imported from interMixingFoam is initialised correctly. immiscibleIncompressibleThreePhaseMixture.C is where incompressibleThreePhaseMixture and threePhaseInterfaceProperties is joined in interMixingFoam.

- Go into incompressibleThreePhaseMixture folder in interPhaseChangeFoam, open incompressibleThreePhaseMixture.C and change line 161 to the following.

incompressibleThreePhaseMixture.C

```
    alpha2_ == 1.0 - alpha1_ - alpha3_;
```

This is done, because the solver will only solve equations for phases 1 and 3, which is liquid and air.

- Open incompressibleThreePhaseMixture.H and change the beginning of the class declaration to the following.

incompressibleThreePhaseMixture.H

```
class incompressibleThreePhaseMixture
:
    public IOdictionary,
    public transportModel
{
```

```
protected:
    // protected data

        word phase1Name_;
        word phase2Name_;
        word phase3Name_;
```

This makes the data protected and not private, so `phaseChangeThreePhaseMixtures` can inherit from `incompressibleThreePhaseMixture`

- Go into the `phaseChangeThreePhaseMixture` folder and then into the `SchneerSauer` folder. In both files in the `SchneerSauer` folder. Replace every occurrence of the word "Two" to the word "Three". Use a find and replace functionality if not on linux. Assuming the terminal is still located in the `phaseChangeThreePhaseMixture` folder it can be done as follows.

Remaining in the two files

```
    cd ../SchnerrSauer/
    sed -i 's/Two/Three/g' *
```

- Go to the `Make` folder located in the `phaseChangeThreePhaseMixtures` folder. Open `files` and change its contents as.

phaseChangeThreePhaseMixtures/Make/files

```
phaseChangeThreePhaseMixture/phaseChangeThreePhaseMixture.C
phaseChangeThreePhaseMixture/phaseChangeThreePhaseMixtureNew.C
SchnerrSauer/SchnerrSauer.C

LIB = $(FOAM_USER_LIBBIN)/libphaseChangeThreePhaseMixtures
```

- Open the `options` file and change its contents as.

phaseChangeThreePhaseMixtures/Make/options

```
EXE_INC = \
    -I$(LIB_SRC)/transportModels/twoPhaseMixture/lnInclude \
    -I../incompressibleThreePhaseMixture \
    -I../threePhaseInterfaceProperties \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = \
    -ltwoPhaseMixture \
    -ltwoPhaseProperties \
    -lincompressibleTransportModels \
    -lfiniteVolume
```

- Compile the library by executing `wmake` with the terminal located in the `phaseChangeThreePhaseMixtures` directory.

Now the library should be compiled and it is time to modify the solver to accommodate the changes derived for three phases.

4.2 Modifying the solver

Now that the library has been compiled, it is time to implement the library in the solver. To do this, the following has to be done.

- Rename the `interPhaseChangeFoam` folder to `inter3PhaseChangeFoam`.

- Rename `interPhaseChangeFoam.C` to `inter3PhaseChangeFoam.C`.
- Delete the `Allwmake` and the `Allwclean` files from the `inter3PhaseChangeFoam` directory
- Remove line 7 from `alphaControls.H` regarding `MULESCorr` variable. This is done because this implementation does not use `MULES` for interface compression. It can be added later, but the first scope of the solver is to implement the three phases with phase change.
- In `createfields.H`, replace everything between the line where `createPhi.H` is included and where `volScalarField rho` is created with the following.

`createFields.H`

```
Info<< "Creating phaseChangeThreePhaseMixture\n" << endl;
autoPtr<phaseChangeThreePhaseMixture> mixture =
    phaseChangeThreePhaseMixture::New(U, phi);


volScalarField& alpha1(mixture->alpha1());
volScalarField& alpha2(mixture->alpha2());
volScalarField& alpha3(mixture->alpha3());

const dimensionedScalar& rho1 = mixture->rho1();
const dimensionedScalar& rho2 = mixture->rho2();
const dimensionedScalar& rho3 = mixture->rho3();
```

  Here the new library is initialised. Now the functionality is accessed with `mixture->`. Before `mixture->` was used for the mixture and `interface.` was used for the interface. This means, that everywhere `interface.` appears needs to be changed.

- In `createfields.H` change how `volScalarField rho` is calculated as.

`createFields.H`

```
volScalarField rho
(
    IOobject
    (
        "rho",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT
    ),
    alpha1*rho1 + alpha2*rho2 + alpha3*rho3
);
rho.oldTime();
```

- In `createfields.H` delete the following lines.

`createFields.H`

```
// Construct interface from alpha1 distribution
interfaceProperties interface(alpha1, U, mixture());
```

- In `createfields.H` delete all the lines below where `createFvOptions.H` is included.
- In `alphaEqnSubCycle.H` change `interface.` with `mixture->` in lines 3 and 9, and then delete the following line.

`alphaEqnSubCycle.H`

```
    rho == alpha1*rho1 + alpha2*rho2;
```

- Almost every line in `alphaEqn.H` has been changed, so to make this tutorial easier to follow, change the contents of `alphaEqn.H` to the following, again for copy-paste refer Appendix B.1.

alphaEqn.H

```
1  {
2      word alphaScheme("div(phi,alpha)");
3      word alpharScheme("div(phirb,alpha)");
4
5      surfaceScalarField phir("phir", phic*mixture->nHatf());
6
7      Pair<tmp<volScalarField>> vDotAlphal =
8          mixture->vDotAlphal();
9      const volScalarField& vDotcAlphal = vDotAlphal[0]();
10     const volScalarField& vDotvAlphal = vDotAlphal[1]();
11     const volScalarField vDotvmcAlphal(vDotvAlphal - vDotcAlphal);
12
13     for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
14     {
15         fvScalarMatrix alpha1Eqn
16         (
17             fvm::ddt(alpha1)
18           + fvm::div(phi, alpha1, "div(phi,alpha)")
19           - fvm::Sp(divU, alpha1)
20           ==
21             fvm::Sp(vDotvmcAlphal, alpha1)
22           + vDotcAlphal*(1-alpha3)
23         );
24         alpha1Eqn.solve();
25         alpha1 = min(max(alpha1, scalar(0)), scalar(1));
26
27         // Solve for alpha3
28         fvScalarMatrix alpha3Eqn
29         (
30             fvm::ddt(alpha3)
31           + fvm::div(phi, alpha3, "div(phi,alpha)")
32
33         );
34         alpha3Eqn.solve();
35         alpha3 = min(max(alpha3, scalar(0)), scalar(1));
36
37         alpha2 = 1.0 - alpha1 - alpha3;
38         alpha2 = min(max(alpha2, scalar(0)), scalar(1));
39     }
40
41     volScalarField rhoNew(alpha1*rho1 + alpha2*rho2 + alpha3*rho3);
42     rho == rhoNew;
43
44     rhoPhi == fvc::interpolate(rho)*phi;
45
46     Info<< "Liquid phase volume fraction = "
47         << alpha1.weightedAverage(mesh.V()).value()
48         << "  Min(" << alpha1.name() << ") = " << min(alpha1).value()
49         << "  Max(" << alpha1.name() << ") = " << max(alpha1).value()
50         << endl;
51
52     Info<< "Vapor phase volume fraction = "
53         << alpha2.weightedAverage(mesh.V()).value()
54         << "  Min(" << alpha2.name() << ") = " << min(alpha2).value()
55         << "  Max(" << alpha2.name() << ") = " << max(alpha2).value()
56         << endl;
57 }
```

Here everything regarding `MULES` is removed and the derived equation for α_l, Equation 3.17, and $\alpha_n c$, Equation 3.6, are implemented through lines 17-25 and 30-35, respectively. The terms from the cavitation model is called on lines 7-11.

- Change line 16 in `pEqn.H` as follows.

pEqn.H

```
                mixture->surfaceTensionForce()
```

- Change line 20 in UEqn.H as follows

UEqn.H

```
                    mixture->surfaceTensionForce()
```

- Change the include files before the main function in inter3PhaseChangeFoam.C to the following.

inter3PhaseChangeFoam.C

```
#include "fvCFD.H"
#include "CMULES.H"
#include "subCycle.H"
#include "phaseChangeThreePhaseMixture.H"
#include "turbulentTransportModel.H"
#include "pimpleControl.H"
#include "fvOptions.H"
```

- Change the places in inter3PhaseChangeFoam.C where the correct() functions are called, so they become as follows.

inter3PhaseChangeFoam.C

```
            mixture->correctNu();

            #include "alphaEqnSubCycle.H"

            mixture->correctInterface();

            #include "UEqn.H"
```

- Go the Make folder in inter3PhaseChangeFoam and change to contents of files as follows.

inter3PhaseChangeFoam/Make/files

```
incompressibleThreePhaseMixture/incompressibleThreePhaseMixture.C
threePhaseInterfaceProperties/threePhaseInterfaceProperties.C
inter3PhaseChangeFoam.C

EXE = $(FOAM_USER_APPBIN)/inter3PhaseChangeFoam
```

- Change the contents of the options file to the following as follows

inter3PhaseChangeFoam/Make/options

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -I$(LIB_SRC)/transportModels/twoPhaseMixture/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels/interfaceProperties/lnInclude \
    -I$(LIB_SRC)/transportModels/twoPhaseProperties/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/incompressible/lnInclude \
    -IincompressibleThreePhaseMixture \
    -IthreePhaseInterfaceProperties \
    -IphaseChangeThreePhaseMixtures/lnInclude
```

```
EXE_LIBS = \
    -L$(FOAM_USER_LIBBIN) \
    -lfiniteVolume \
    -lfvOptions \
    -lmeshTools \
    -lsampling \
    -lphaseChangeThreePhaseMixtures \
    -ltwoPhaseMixture \
    -linterfaceProperties \
    -ltwoPhaseProperties \
    -lincompressibleTransportModels \
    -lturbulenceModels \
    -lincompressibleTurbulenceModels
```

That was all the changes necessary to accommodate the new library. Now the mass transfer terms derived are implemented in the two `SchnerrSauer` files. This is done in the following.

- Go the `SchnerrSauer` folder in `phaseChangeThreephaseMixtures` and change the most of the functions in `SchnerrSauer.C` as described in the following.
  Change the bubble radius function `rRb` as follows.

`SchnerrSauer.C`

```
Foam::tmp<Foam::volScalarField>
Foam::phaseChangeThreePhaseMixtures::SchnerrSauer::rRb
(
    const volScalarField& limitedAlpha1,
    const volScalarField& limitedAlpha3
) const
{
    return pow
    (
        (3.0/(4.0*constant::mathematical::pi*n_))
        *max((1.0 + alphaNuc() - limitedAlpha1 - limitedAlpha3)
        /(max(limitedAlpha1,ROOTVSMALL))
        ,ROOTVSMALL),
        1.0/3.0
    );
}
```

This allows the function to include α_{nc} as well. Some limits have been introduced to avoid dividing by 0, or taking the square root of a negative number.
Update the `pCoeff` function as follows.

`SchnerrSauer.C`

```
Foam::tmp<Foam::volScalarField>
Foam::phaseChangeThreePhaseMixtures::SchnerrSauer::pCoeff
(
    const volScalarField& p
) const
{
    volScalarField limitedAlpha1(min(max(alpha1_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha2(min(max(alpha2_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha3(min(max(alpha3_, scalar(0)), scalar(1)));
    volScalarField rho
    (
        limitedAlpha1*rho1() + limitedAlpha2*rho2() + limitedAlpha3*rho3()
    );

    return
           (rho1()*rho2())/(rho + limitedAlpha3*(rho1()-rho3()))
           *sqrt(2/(3*rho1()))*sqrt(1/(mag(p - pSat()) + 0.01*pSat()));

}
```

This implements the density term derived in Appendix A.2 together with some elements from Equation 3.15.
Change the `mDotAlphal()` and `mDotP()` function to what is displayed in the following code.

SchnerrSauer.C

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::phaseChangeThreePhaseMixtures::SchnerrSauer::mDotAlphal() const
{
    const volScalarField& p = alpha1_.db().lookupObject<volScalarField>("p");
    volScalarField pCoeff(this->pCoeff(p));

    volScalarField limitedAlpha1(min(max(alpha1_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha2(min(max(alpha2_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha3(min(max(alpha3_, scalar(0)), scalar(1)));

    volScalarField rho
    (
        limitedAlpha1*rho1() + limitedAlpha2*rho2() + limitedAlpha3*rho3()
    );
    volScalarField densityfrac
    (
        (rho+limitedAlpha3*(rho2()-rho3()))/(rho+limitedAlpha3*(rho1()-rho3()))
    );
    return Pair<tmp<volScalarField>>
    (
        Cc_
        *((3.0)
        /(rRb(limitedAlpha1, limitedAlpha3) + pow(rRb(limitedAlpha1, limitedAlpha3),4)
        *(4.0/3.0)*n_*constant::mathematical::pi*densityfrac))
        *pCoeff*max(p - pSat(), p0_),

        Cv_
        *((4.0*n_*constant::mathematical::pi*pow(rRb(limitedAlpha1, limitedAlpha3),2))
        /(1.0 + pow(rRb(limitedAlpha1, limitedAlpha3),3)
        *(4.0/3.0)*n_*constant::mathematical::pi*densityfrac))
        *pCoeff*min(p - pSat(), p0_)
    );
}


Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::phaseChangeThreePhaseMixtures::SchnerrSauer::mDotP() const
{
    const volScalarField& p = alpha1_.db().lookupObject<volScalarField>("p");
    volScalarField pCoeff(this->pCoeff(p));

    volScalarField limitedAlpha1(min(max(alpha1_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha2(min(max(alpha2_, scalar(0)), scalar(1)));
    volScalarField limitedAlpha3(min(max(alpha3_, scalar(0)), scalar(1)));

    volScalarField rho
    (
        limitedAlpha1*rho1() + limitedAlpha2*rho2() + limitedAlpha3*rho3()
    );
    volScalarField densityfrac
    (
        (rho+limitedAlpha3*(rho2()-rho3()))/(rho+limitedAlpha3*(rho1()-rho3()))
    );

    volScalarField apCoeff(limitedAlpha1*pCoeff);

    return Pair<tmp<volScalarField>>
    (
        (Cc_)*max((1.0 + alphaNuc() - limitedAlpha1 - limitedAlpha3),0.0)
        *((3.0)
        /(rRb(limitedAlpha1, limitedAlpha3) + pow(rRb(limitedAlpha1, limitedAlpha3),4)
        *(4.0/3.0)*n_*constant::mathematical::pi*densityfrac))
```

```
        *pos0(p - pSat())*pCoeff,

        (-Cv_)*limitedAlpha1
        *((4.0*n_*constant::mathematical::pi*pow(rRb(limitedAlpha1, limitedAlpha3),2))
        /(1.0 + pow(rRb(limitedAlpha1, limitedAlpha3),3)
        *(4.0/3.0)*n_*constant::mathematical::pi*densityfrac))
        *neg(p - pSat())*pCoeff
    );
}
```

This implements both the mass transfer terms derived in equations 3.12 and 3.13, for the pressure correction and the velocity equations. `mDotP()` returns the mass transfer terms for the pressure correction equation and `mDotAlphal()` returns the mass transfer terms for the volume fraction equation.

- Change the declaration of the bubble radius function in `SchnerrSauer.H` as follows.

`SchnerrSauer.H`

```
        //- Reciprocal bubble radius
        tmp<volScalarField>rRb(const volScalarField& limitedAlpha1, const volScalarField&
    limitedAlpha3) const;
```

This was the final change to the solver, and it is now ready to compile.

- Compile the solver by first executing `wmake` in the `phaseChangeThreePhaseMixtures` directory and afterwards in the `inter3PhaseChangeFoam` directory.

Now that the solver has been implemented, it is time to test it on a case. This is done in the following Chapter.

Chapter 5

Test Case

To demonstrate the new solver, the case from the `cavitatingFoam` tutorial is used. It is the case from `RAS` cases called `throttle`. Only the significant changes for using the solver with the case will the shown in this chapter. If further details are required by the reader, please refer to the case files. An overview of the case geometry is shown in Figure 5.1. As shown in Figure 5.1 the left box is

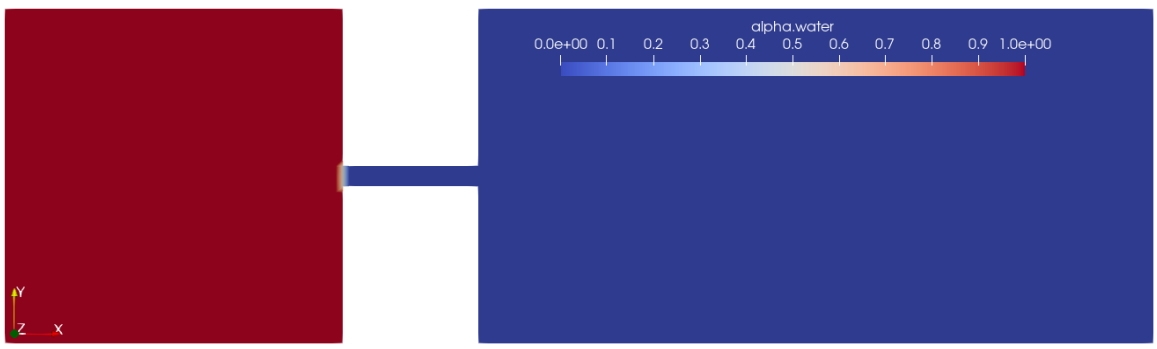

Figure 5.1: Volume fraction for alpha liquid in the entire simulation domain.

initialised with water and the rest of the domain is air. At this point in time there is no vapour anywhere. The mesh is shown in Figure 5.2 As it can be seen in Figure 5.2 the mesh is just a

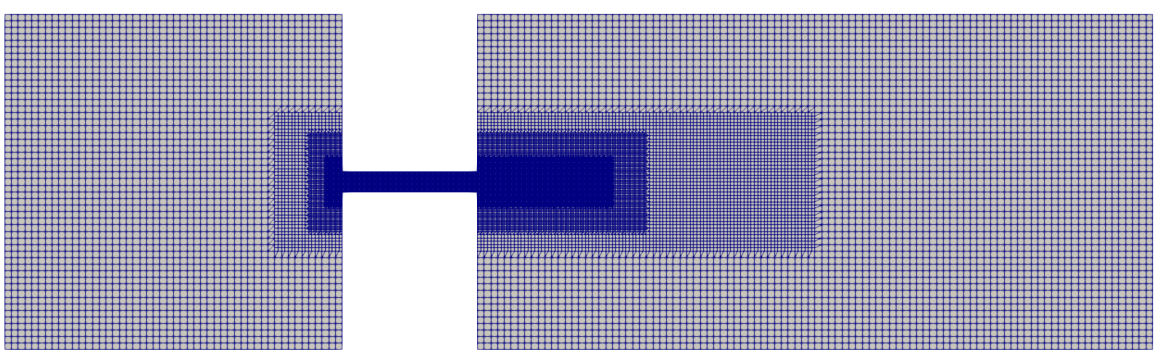

Figure 5.2: Mesh used for the simulation.

coarse background mesh with three different refined zones. The leftmost boundary is the inlet where pressurised water is entering at 100 bar and the rightmost boundary is the outlet where the different phases can exit and air can enter in case of reverse flow with a pressure of 1 bar. The reset of the boundaries are walls. The new solver is set up using `transportProperties` file. The contents of that file is shown below.

trasportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water vapor air);

phaseChangeThreePhaseMixture SchnerrSauer;

pSat        2300;

water
{
    transportModel  Newtonian;
    nu         6.47e-7;
    rho        1000;
}

vapor
{
    transportModel  Newtonian;
    nu         2.243e-5;
    rho        0.535;
}

air
{
    transportModel  Newtonian;
    nu         1.5684e-5;
    rho        1.177;
}


SchnerrSauerCoeffs
{
    n               1.0e+14;
    dNuc            3.0e-7;
    Cc              1;
    Cv              1;
}

// Surface tension coefficients
sigma12         0.03;
sigma13         0.03;

// ************************************************************************* //
```

The setup is fairy similar to the one in `interPhaseChangeFoam` tutorial, except there is now three phases and two surface tension coefficients. It is very important that the order the phases are stated in is `(liquid vapour non-condensing)`, otherwise the solver will not work. In the beginning `Cc` and `Cv` are set to zero, so no cavitation occurs, then the case is solved until a straight jet appears as shown in Figure 5.3. In Figure 5.3 the jet is becoming more smeared in the edges due to lower mesh resolution the closer it gets to the outlet. Now `Cc` and `Cv` are set to 1 and the simulation is

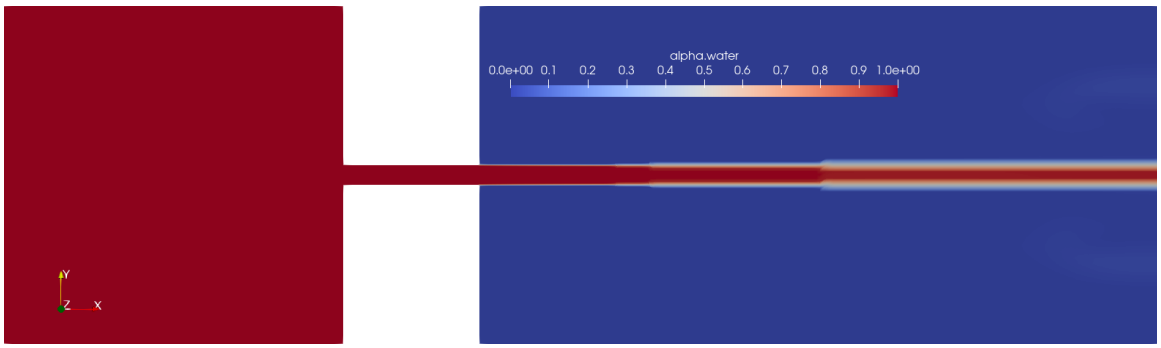

Figure 5.3: Initial jet coming out of the box at t = 0.0018 s

continued. Due to the cavitation not being physically correct in the beginning the simulation is run until the cavitation pockets are stabilised. The steady result is shown in Figure 5.4. This results

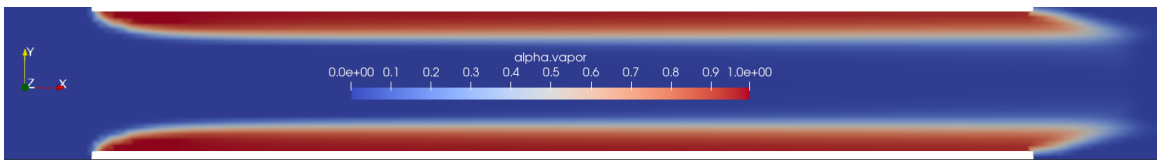

Figure 5.4: The stable cavitation pockets.

in a thinning of the jet coming out of the box, which is a textbook example of a hydraulic flip as shown by H. Chaves [5]. The model seems to perform somewhat like the simulation performed by W. Yuan and G. H. Schnerr [1], but the basis for comparison is not strong, since the same geometry or boundary conditions are not the same. All three phases are shown in Figures 5.5-5.7.

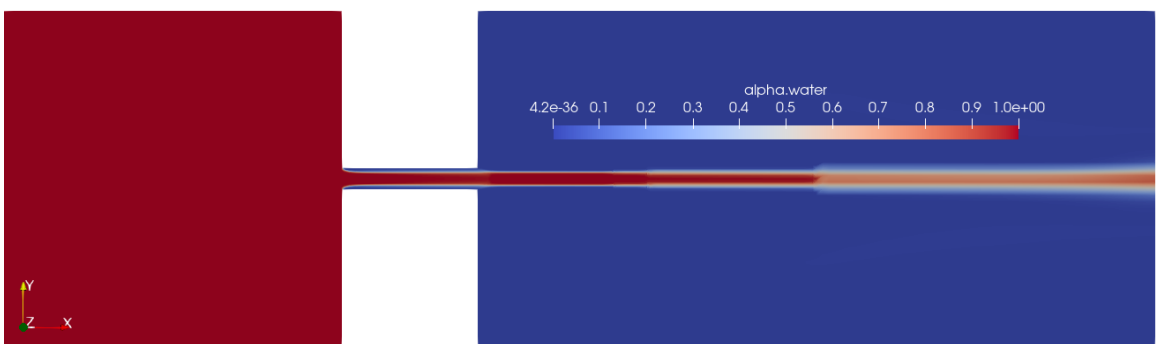

Figure 5.5: Contours of water volume fraction at final time-step.

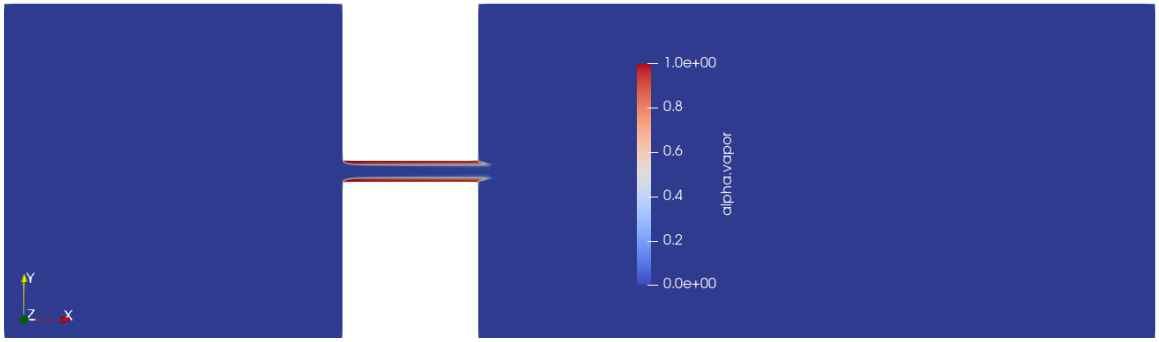

Figure 5.6: Contours of vapour volume fraction at final time-step.

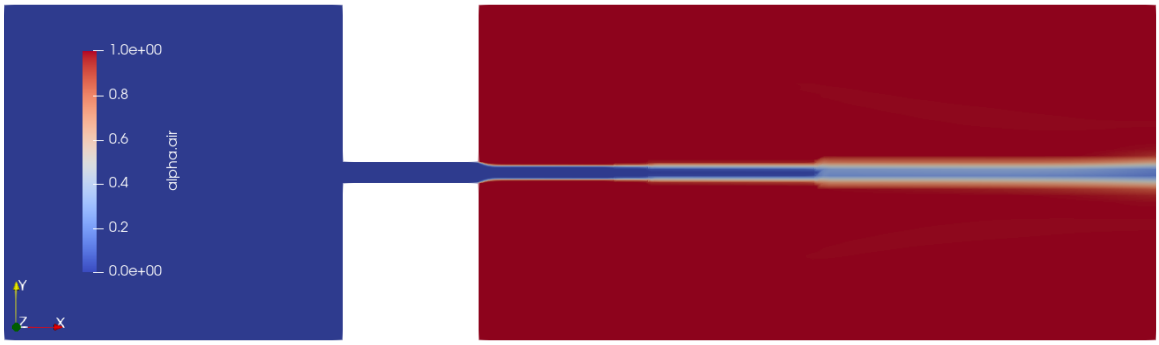

Figure 5.7: Contours of air volume fraction at final time-step.

Bibliography

[1] W. Yuan and G. H. Schnerr, "Numerical simulation of two-phase flow in injection nozzles: Interaction of cavitation and external jet formation," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 125, no. 6, pp. 963–969, 2003.

[2] A. Asnaghi, *Developing Computational Methods for Detailed Assessment of Cavitation on Marine Propellers*. PhD thesis, 06 2015.

[3] H. Jasak, "Error analysis and estimation for the finite volume method with applications to fluid flows," *Direct*, vol. M, 01 1996.

[4] G. H. Schnerr and J. Sauer, "Physical and Numerical Modeling of Unsteady Cavitation Dynamics," *4th International Conference on Multiphase Flow*, no. June, pp. 1–12, 2001.

[5] H. Chaves, M. Knapp, A. Kubitzek, F. Obermeier, and T. Schneider, "Experimental study of cavitation in the nozzle hole of diesel injectors using transparent nozzles," *SAE Technical Papers*, no. 412, 1995.

Study questions

1. In which order must the phases be listed in `transportProperties`, when using the new solver?
2. Why is $\dot{m}$ decomposed into $\dot{m}_v$ and $\dot{m}_c$?
3. How do you check which libraries a solver uses?
4. What is the role of α_{nuc}?
5. Which schemes should be used for the temporal and divergence terms for α, for boundedness to be ensured?
6. What are the assumptions of the `SchnerrSauer` model?

Appendix A

Mathematical derivations

In this appendix some parts of the mass transfer terms are derived for a system with two and three phases.

A.1 Density term for two phases

The system of equations is

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l},$$
$$\frac{\partial \alpha_v}{\partial t} + \nabla \cdot (\alpha_v U) = -\frac{\dot{m}}{\rho_v}.$$

They can be rewritten to the form

$$\frac{d\alpha_l}{dt} + \alpha_l \nabla \cdot (U) = \frac{\dot{m}}{\rho_l}, \tag{A.1}$$
$$\frac{d\alpha_v}{dt} + \alpha_v \nabla \cdot (U) = -\frac{\dot{m}}{\rho_v}. \tag{A.2}$$

The first relation to realise is that the change of the sum of alpha is 0 with the change in time and space, that yields

$$\frac{d\alpha_l}{dt} + \frac{d\alpha_v}{dt} = 0. \tag{A.3}$$

If $\frac{d\alpha_l}{dt}$ is isolated in the above transport equation and substituted into Equation A.3 it becomes

$$-\alpha_l \nabla \cdot U + \frac{\dot{m}}{\rho_l} + \frac{d\alpha_v}{dt} = 0.$$

Rearranging for $\dot{m}$ yields

$$\dot{m} = \rho_l \alpha_l \nabla \cdot U - \rho_l \frac{d\alpha_v}{dt}. \tag{A.4}$$

The non-conservative continuity equation is given by

$$\nabla \cdot U = -\frac{1}{\rho}\frac{dp}{dt},$$

and considering that each phase is incompressible results in

$$\nabla \cdot U = -\frac{1}{\rho}\left(\frac{d(\rho_l \alpha_l)}{dt} + \frac{d(\rho_v \alpha_v)}{dt}\right),$$
$$= -\frac{1}{\rho}\left(\rho_l \frac{d\alpha_l}{dt} + \rho_v \frac{d\alpha_v}{dt}\right).$$

Substituting $\rho_l \frac{d\alpha_l}{dt}$ with A.1 the equation becomes

$$\nabla \cdot U = -\frac{1}{\rho}\left(-\rho_l \alpha_l \nabla \cdot U + \dot{m} + \rho_v \frac{d\alpha_v}{dt}\right).$$

Rearranging for $\dot{m}$ yields

$$\rho \nabla \cdot U = \rho_l \alpha_l \nabla \cdot U - \dot{m} - \rho_v \frac{d\alpha_v}{dt}, \tag{A.5}$$

which with a bit further rearrangement becomes

$$\dot{m} = (\rho_l \alpha_l - \rho)\nabla \cdot U - \rho_v \frac{d\alpha_v}{dt}. \tag{A.6}$$

Equating A.4 and A.6 yields

$$\rho_l \alpha_l \nabla \cdot U - \rho_l \frac{d\alpha_v}{dt} = (\rho_l \alpha_l - \rho)\nabla \cdot U - \rho_v \frac{d\alpha_v}{dt},$$

that can be rearranged as

$$(\rho_l \alpha_l - \rho_l \alpha_l + \rho)\nabla \cdot U = (\rho_l - \rho_v)\frac{d\alpha_v}{dt}.$$

Solving for $\nabla \cdot U$ yields

$$\nabla \cdot U = \frac{(\rho_l - \rho_v)}{\rho}\frac{d\alpha_v}{dt}. \tag{A.7}$$

Now if the transport Equation A.2 is solved for $\dot{m}$ and Equation A.7 is substituted into it, it yields

$$\begin{aligned}
\dot{m} &= -\rho_v \frac{d\alpha_v}{dt} - \alpha_v \rho_v \frac{(\rho_l - \rho_v)}{\rho}\frac{d\alpha_v}{dt} \\
&= \rho_v \frac{d\alpha_v}{dt}\left(-1 - \frac{\alpha_v(\rho_l - \rho_v)}{\rho}\right) \\
&= \rho_v \frac{d\alpha_v}{dt}\left(\frac{-\rho - \alpha_v \rho_l + \alpha_v \rho_v)}{\rho}\right) \\
&= \rho_v \frac{d\alpha_v}{dt}\left(\frac{-\rho_l \alpha_l - \rho_v \alpha_v - \alpha_v \rho_l + \alpha_v \rho_v)}{\rho}\right) \\
&= \rho_v \frac{d\alpha_v}{dt}\left(\frac{-\rho_l(\alpha_l + \alpha_v)}{\rho}\right) \\
&= \frac{d\alpha_v}{dt}\left(\frac{-\rho_v \rho_l}{\rho}\right).
\end{aligned}$$

Here the sign is dependant on which of the mass transfer terms was chosen as negative. Having the negative sign on the Equation A.1, the final term will become

$$\dot{m} = \frac{d\alpha_v}{dt}\left(\frac{\rho_v \rho_l}{\rho}\right). \tag{A.8}$$

So the positive version is used, since the minus sign will cancel out with another one mentioned in the derivation in Chapter 2.

A.2 Density term for three phase flows

The system of equations is

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U) = \frac{\dot{m}}{\rho_l},$$
$$\frac{\partial \alpha_v}{\partial t} + \nabla \cdot (\alpha_v U) = -\frac{\dot{m}}{\rho_v},$$
$$\frac{\partial \alpha_{nc}}{\partial t} + \nabla \cdot (\alpha_{nc} U) = 0.$$

They can be rewritten to the form

$$\frac{d\alpha_l}{dt} + \alpha_l \nabla \cdot (U) = \frac{\dot{m}}{\rho_l}, \tag{A.9}$$
$$\frac{d\alpha_v}{dt} + \alpha_v \nabla \cdot (U) = -\frac{\dot{m}}{\rho_v}, \tag{A.10}$$
$$\frac{d\alpha_{nc}}{dt} + \alpha_{nc} \nabla \cdot (U) = 0. \tag{A.11}$$

The procedure from this point is similar to the density term for two phases in Appendix A.1, but for the sake of completeness, the derivation will also be done here. It is again used that the change of the sum of volume fractions is 0 as A.12.

$$\frac{d\alpha_l}{dt} + \frac{d\alpha_v}{dt} + \frac{d\alpha_{nc}}{dt} = 0. \tag{A.12}$$

Substituting $\frac{d\alpha_l}{dt}$ and $\frac{d\alpha_{nc}}{dt}$ from the above transport equations yields

$$-\alpha_l \nabla \cdot U + \frac{\dot{m}}{\rho_l} + \frac{d\alpha_v}{dt} - \alpha_{nc} \nabla \cdot U = 0.$$

Solving for the mass flow rate yields

$$\rho_l \left(\alpha_l + \alpha_{nc}\right) \nabla \cdot U - \rho_l \frac{d\alpha_v}{dt} = \dot{m}. \tag{A.13}$$

Now one of terms for the mass transfer rate is obtained. The other is derived by looking at the non-conservative continuity equation,

$$\nabla \cdot U = -\frac{1}{\rho}\frac{dp}{dt}.$$

Considering that each phase is incompressible, this equation can be written as

$$\begin{aligned}\nabla \cdot U &= -\frac{1}{\rho}\left(\frac{d(\rho_l \alpha_l)}{dt} + \frac{d(\rho_v \alpha_v)}{dt} + \frac{d(\rho_{nc}\alpha_{nc})}{dt}\right),\\ &= -\frac{1}{\rho}\left(\rho_l \frac{d\alpha_l}{dt} + \rho_v \frac{d\alpha_v}{dt} + \rho_{nc}\frac{d\alpha_{nc}}{dt}\right).\end{aligned}$$

Substituting $\rho_l \frac{d\alpha_l}{dt}$ and $\rho_{nc}\frac{d\alpha_{nc}}{dt}$ with Equations A.9 and A.11 yields

$$\nabla \cdot U = -\frac{1}{\rho}\left(-\rho_l \alpha_l \nabla \cdot U + \dot{m} + \rho_v \frac{d\alpha_v}{dt} - \rho_{nc}\alpha_{nc}\nabla \cdot U\right).$$

Rearranging to isolate for the mass transfer rate yeilds

$$\dot{m} = -\rho_v \frac{d\alpha_v}{dt} + \left(\rho_l \alpha_l + \rho_{nc}\alpha_{nc} - \rho\right) \nabla \cdot U. \tag{A.14}$$

Equating Equation A.13 and Equation A.14 yields

$$\rho_l\left(\alpha_l+\alpha_{nc}\right)\nabla\cdot U-\frac{d\alpha_v}{dt}\rho_l=-\frac{d\alpha_v}{dt}\rho_v+\left(\rho_l\alpha_l+\rho_{nc}\alpha_{nc}-\rho\right)\nabla\cdot U. \tag{A.15}$$

rearranging gives

$$\begin{aligned}\rho_l\left(\alpha_l+\alpha_{nc}\right)\nabla\cdot U+\frac{d\alpha_v}{dt}\left(-\rho_l+\rho_v\right)&=\left(\rho_l\alpha_l+\rho_{nc}\alpha_{nc}-\rho\right)\nabla\cdot U,\\ \frac{d\alpha_v}{dt}\left(-\rho_l+\rho_v\right)&=\left(\left(\rho_l\alpha_l+\rho_{nc}\alpha_{nc}-\rho\right)-\rho_l\left(\alpha_l+\alpha_{nc}\right)\right)\nabla\cdot U,\\ \frac{d\alpha_v}{dt}\left(-\rho_l+\rho_v\right)&=\left(\rho_{nc}\alpha_{nc}-\rho-\rho_l\alpha_{nc}\right)\nabla\cdot U.\end{aligned}$$

Solving for the divergence of velocity gives

$$\begin{aligned}\frac{d\alpha_v}{dt}\frac{\left(-\rho_l+\rho_v\right)}{-\rho+\alpha_{nc}\left(\rho_{nc}-\rho_l\right)}&=\nabla\cdot U,\\ \frac{d\alpha_v}{dt}\frac{\left(\rho_l-\rho_v\right)}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}&=\nabla\cdot U.\end{aligned} \tag{A.16}$$

Now if the transport equation for α_v is solved for $\dot{m}$ and Equation A.16 is substituted into it, it yields

$$\begin{aligned}\dot{m}&=-\rho_v\frac{d\alpha_v}{dt}-\alpha_v\rho_v\frac{\left(\rho_l-\rho_v\right)}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\frac{d\alpha_v}{dt}\\ &=\rho_v\frac{d\alpha_v}{dt}\left(-1-\frac{\alpha_v(\rho_l-\rho_v)}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right)\\ &=\rho_v\frac{d\alpha_v}{dt}\left(\frac{-\left(\rho+\alpha_{nc}\rho_l-\alpha_{nc}\rho_{nc}\right)-\alpha_v\rho_l+\alpha_v\rho_v)}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right)\\ &=\rho_v\frac{d\alpha_v}{dt}\left(\frac{-\alpha_l\rho_l-\alpha_v\rho_v-\alpha_{nc}\rho_{nc}-\alpha_{nc}\rho_l+\alpha_{nc}\rho_{nc}-\alpha_v\rho_l+\alpha_v\rho_v)}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right)\\ &=\rho_v\frac{d\alpha_v}{dt}\left(\frac{-\rho_l(\alpha_l+\alpha_v+\alpha_{nc})}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right)\\ &=\frac{d\alpha_v}{dt}\left(\frac{-\rho_v\rho_l}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right)\end{aligned} \tag{A.17}$$

Here the sign is, as for two phases, dependant on which of the mass transfer terms was chosen as negative. Having the negative sign on the Equation A.9, the final term will become

$$\frac{d\alpha_v}{dt}\left(\frac{\rho_v\rho_l}{\rho+\alpha_{nc}\left(\rho_l-\rho_{nc}\right)}\right), \tag{A.18}$$

which will be used for implementation.

A.3 Condensation and evaporation terms for three phases

Since there is not a smart way of rewriting the expression for the change in the vapour volume fraction for three phases, deriving terms for condensation and evaporation becomes more complicated. The derivation still starts with the definition from the Schnerr and Sauer model restated in Equation A.19, which is a rearranged version of Equation 2.24 as

$$\alpha_v=\alpha_l n_0\pi\frac{4}{3}R^3. \tag{A.19}$$

Taking the total derivative of Equation A.19 yields

$$\frac{d\alpha_v}{dt} = \alpha_l 4\pi n_0 R^2 \frac{dR}{dt} + n_0 \frac{4}{3}\pi R^3 \frac{d\alpha_l}{dt}. \tag{A.20}$$

From the volume fraction equation system is

$$\frac{d\alpha_l}{dt} + \alpha_l \nabla \cdot (U) = \frac{\dot{m}}{\rho_l}, \tag{A.21}$$

$$\frac{d\alpha_v}{dt} + \alpha_v \nabla \cdot (U) = -\frac{\dot{m}}{\rho_v}, \tag{A.22}$$

$$\frac{d\alpha_{nc}}{dt} + \alpha_{nc} \nabla \cdot (U) = 0. \tag{A.23}$$

Rearranging the last equation yields

$$\frac{d\alpha_{nc}}{dt} = -\alpha_{nc} \nabla \cdot U. \tag{A.24}$$

Looking at the time rate of change of the sum of volume fractions it follows that

$$\frac{d\alpha_l}{dt} + \frac{d\alpha_v}{dt} + \frac{d\alpha_{nc}}{dt} = 0,$$
$$\frac{d\alpha_l}{dt} = -\frac{d\alpha_v}{dt} - \frac{d\alpha_{nc}}{dt}.$$

Substituting Equation A.24 into this equation yields

$$\frac{d\alpha_l}{dt} = -\frac{d\alpha_v}{dt} + \alpha_{nc} \nabla \cdot U. \tag{A.25}$$

Substituting Equation A.25 and Equation A.16 into Equation A.20 yields the following, with intermediate rearrangement steps shown as

$$\frac{d\alpha_v}{dt} = \alpha_l 4 n_0 \pi R^2 \frac{dR}{dt} + \left(-n_0 \frac{4}{3}\pi R^3 + n_0 \frac{4}{3}\pi R^3 \alpha_{nc} \frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right) \frac{d\alpha_v}{dt},$$
$$\left(1 + n_0 \frac{4}{3}\pi R^3 + n_0 \frac{4}{3}\pi R^3 \alpha_{nc} \frac{(\rho_v - \rho_l)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right) \frac{d\alpha_v}{dt} = \alpha_l 4 n_0 \pi R^2 \frac{dR}{dt},$$
$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt}}{1 + n_0 \frac{4}{3}\pi R^3 + n_0 \frac{4}{3}\pi R^3 \alpha_{nc} \frac{(\rho_v - \rho_l)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}},$$
$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt} (\rho + \alpha_{nc}(\rho_l - \rho_{nc}))}{\rho + \alpha_{nc}(\rho_l - \rho_{nc}) + n_0 \frac{4}{3}\pi R^3 (\rho + \alpha_{nc}(\rho_l - \rho_{nc})) + n_0 \frac{4}{3}\pi R^3 \alpha_{nc}(\rho_v - \rho_l)},$$
$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt} (\rho + \alpha_{nc}(\rho_l - \rho_{nc}))}{\rho + \alpha_{nc}(\rho_l - \rho_{nc}) + n_0 \frac{4}{3}\pi R^3 (\rho + \alpha_{nc}\rho_l - \alpha_{nc}\rho_{nc} + \alpha_{nc}\rho_v - \alpha_{nc}\rho_l)},$$
$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt} (\rho + \alpha_{nc}(\rho_l - \rho_{nc}))}{\rho + \alpha_{nc}(\rho_l - \rho_{nc}) + n_0 \frac{4}{3}\pi R^3 (\rho + \alpha_{nc}(\rho_v - \rho_{nc}))},$$
$$\frac{d\alpha_v}{dt} = \frac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt}}{1 + n_0 \frac{4}{3}\pi R^3 \left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right)}. \tag{A.26}$$

Similar to the derivation above, implementing

$$\alpha_l = \frac{3\alpha_v}{4\pi n_0 R^3}, \tag{A.27}$$

as a rewritten form of Equation A.19 and again using the definition of total derivative yields

$$\frac{d\alpha_l}{dt} = -\frac{3\alpha_v \frac{dR}{dt}}{\frac{4}{3}\pi n_0 R^4} + \frac{3}{4\pi n_0 R^3}\frac{d\alpha_v}{dt}. \tag{A.28}$$

Substituting Equation A.25 and Equation A.16 into Equation A.28 a term for condensation rate can be obtained, as shown below with all the intermediate rearrangement steps shown as

$$\begin{aligned}
-\frac{d\alpha_v}{dt} + \alpha_{nc}\frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\frac{d\alpha_v}{dt} &= -\frac{3\alpha_v \frac{dR}{dt}}{\frac{4}{3}\pi n_0 R^4} + \frac{3}{4\pi n_0 R^3}\frac{d\alpha_v}{dt}, \\
\left(-1 + \alpha_{nc}\frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} - \frac{3}{4n_0\pi R^3}\right)\frac{d\alpha_v}{dt} &= -\frac{3\alpha_v \frac{dR}{dt}}{\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{-3\alpha_v \frac{dR}{dt}}{\left(-1 + \alpha_{nc}\frac{(\rho_l - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} - \frac{3}{4n_0\pi R^3}\right)\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{-3\alpha_v \frac{dR}{dt}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)}{\left(-\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right) + \alpha_{nc}(\rho_l - \rho_v) - \frac{3}{4n_0\pi R^3}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)\right)\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{-3\alpha_v \frac{dR}{dt}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)}{\left(-\rho - \alpha_{nc}\rho_l + \alpha_{nc}\rho_{nc} + \alpha_{nc}\rho_l - \alpha_{nc}\rho_v - \frac{3}{4n_0\pi R^3}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)\right)\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{-3\alpha_v \frac{dR}{dt}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)}{\left(-\rho + \alpha_{nc}(\rho_{nc} - \rho_v) - \frac{3}{4n_0\pi R^3}\left(\rho + \alpha_{nc}(\rho_l - \rho_{nc})\right)\right)\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{-3\alpha_v \frac{dR}{dt}}{\left(\frac{-\rho + \alpha_{nc}(\rho_{nc} - \rho_v)}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} - \frac{3}{4n_0\pi R^3}\right)\frac{4}{3}\pi n_0 R^4}, \\
\frac{d\alpha_v}{dt} &= \frac{3\alpha_v \frac{dR}{dt}}{\left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right)\frac{4}{3}\pi n_0 R^4 + R}.
\end{aligned} \tag{A.29}$$

The terms for condensation and evaporation can be summarised as

$$\frac{d\alpha_v}{dt} = \begin{cases} \dfrac{3\alpha_v \frac{dR}{dt}}{\left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right)\frac{4}{3}\pi n_0 R^4 + R} & \text{if } p > p_{sat} \\ \dfrac{\alpha_l 4 n_0 \pi R^2 \frac{dR}{dt}}{1 + n_0 \frac{4}{3}\pi R^3\left(\frac{\rho + \alpha_{nc}(\rho_v - \rho_{nc})}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})}\right)} & \text{if } p < p_{sat} \end{cases}. \tag{A.30}$$

Appendix B

Modified codes to implement the third phase

B.1 alphaEqn.H

alphaEqn.H

```
{
    word alphaScheme("div(phi,alpha)");
    word alpharScheme("div(phirb,alpha)");

    surfaceScalarField phir("phir", phic*mixture->nHatf());

    Pair<tmp<volScalarField>> vDotAlphal =
        mixture->vDotAlphal();
    const volScalarField& vDotcAlphal = vDotAlphal[0]();
    const volScalarField& vDotvAlphal = vDotAlphal[1]();
    const volScalarField vDotvmcAlphal(vDotvAlphal - vDotcAlphal);

    tmp<surfaceScalarField> talphaPhi;

    for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
    {
        fvScalarMatrix alpha1Eqn
        (
            fvm::ddt(alpha1)
          + fvm::div(phi, alpha1, "div(phi,alpha)")
          - fvm::Sp(divU, alpha1)
          ==
            fvm::Sp(vDotvmcAlphal, alpha1)
          + vDotcAlphal*(1-alpha3)
        );
        alpha1Eqn.solve();
        alpha1 = min(max(alpha1, scalar(0)), scalar(1));

        // Solve for alpha3
        fvScalarMatrix alpha3Eqn
        (
            fvm::ddt(alpha3)
          + fvm::div(phi, alpha3, "div(phi,alpha)")

        );
        alpha3Eqn.solve();
        alpha3 = min(max(alpha3, scalar(0)), scalar(1));

        alpha2 = 1.0 - alpha1 - alpha3;
        alpha2 = min(max(alpha2, scalar(0)), scalar(1));
    }
```

```
    volScalarField rhoNew(alpha1*rho1 + alpha2*rho2 + alpha3*rho3);
    rho == rhoNew;

    rhoPhi == fvc::interpolate(rho)*phi;

    Info<< "Liquid phase volume fraction = "
        << alpha1.weightedAverage(mesh.V()).value()
        << "  Min(" << alpha1.name() << ") = " << min(alpha1).value()
        << "  Max(" << alpha1.name() << ") = " << max(alpha1).value()
        << endl;

    Info<< "Vapor phase volume fraction = "
        << alpha2.weightedAverage(mesh.V()).value()
        << "  Min(" << alpha2.name() << ") = " << min(alpha2).value()
        << "  Max(" << alpha2.name() << ") = " << max(alpha2).value()
        << endl;
}
```