Description of interCondensatingEvaporatingFoam and implementation of SGS term into volume fraction equation

Yaquan Sun

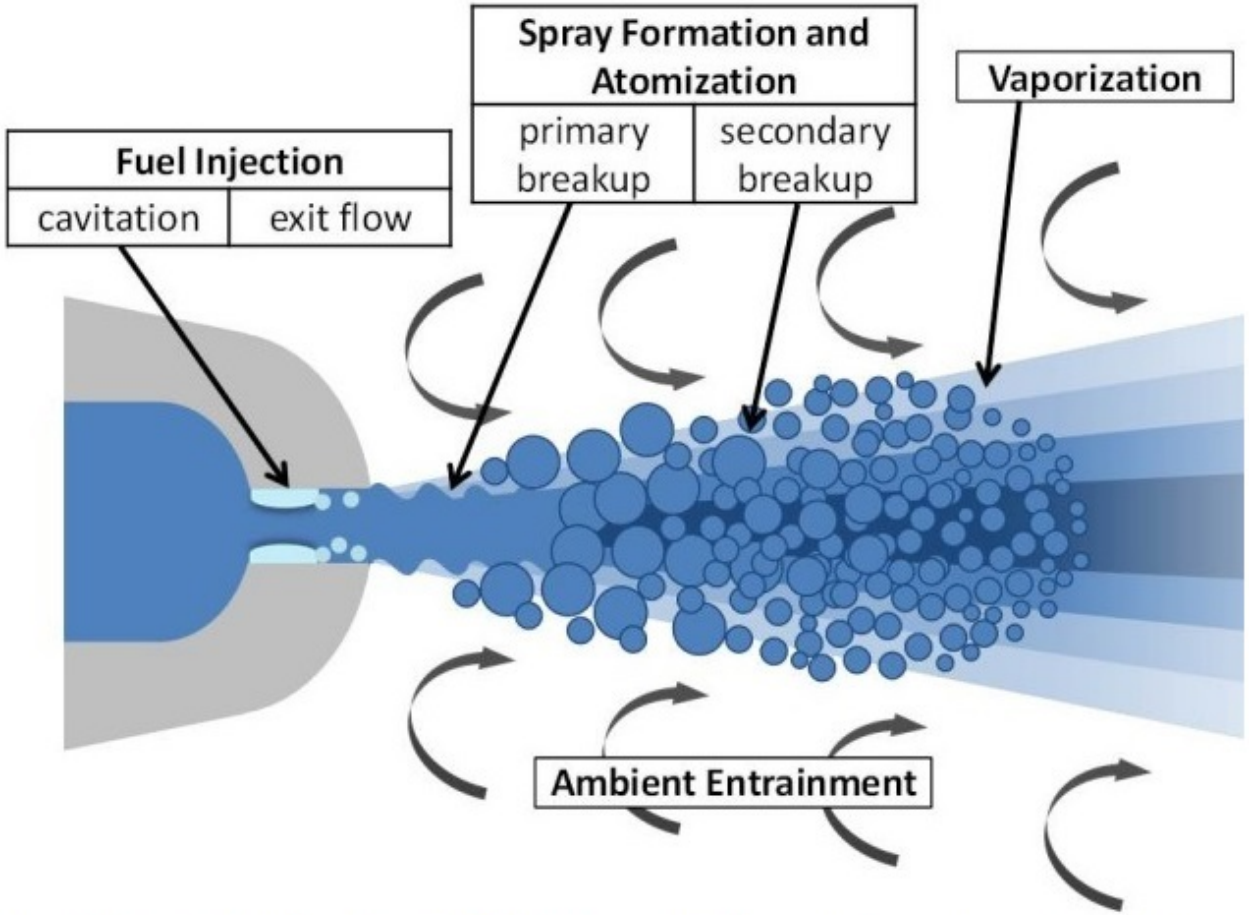
Institute of Reactive Flows and Diagnostics
Technical University of Darmstadt,
Darmstadt, Germany

January 19, 2023

Contents

- Background
- Theory
- interCondensatingEvaporatingFoam
- Implementation
- Test case and results

Background



Luo, K., Shao, C., Chai, M. and Fan, J., 2019. Level set method for atomization and evaporation simulations. Progress in Energy and Combustion Science, 73, pp.65-94.

A schematic of spray atomization process.

VOF-LES

$$\frac{\partial \bar{\rho}}{\partial t} + \nabla \cdot (\bar{\rho}\widetilde{\mathbf{U}}) = 0 \tag{1}$$

$$\frac{\partial(\rho\widetilde{\mathbf{U}})}{\partial t} + \nabla \cdot (\bar{\rho}\widetilde{\mathbf{U}} \otimes \widetilde{\mathbf{U}} - \bar{\mu}\widetilde{\mathbf{D}} - \sigma\bar{\mathbf{n}}\bar{\kappa}\delta_{\mathbf{s}}) + \bar{p}\mathbf{I} = -\nabla \cdot (\tau_{\rho uu} + \tau_{\mu\mathbf{s}}) + \tau_{nn} + \bar{\rho}\mathbf{g} \tag{2}$$

$$\frac{\partial \overline{\alpha}}{\partial t} + \nabla \cdot (\overline{\alpha}\widetilde{\mathbf{U}}) = 0 \qquad \longleftarrow \textit{SGSterm} \quad \tau_{u\alpha} \tag{3}$$

$$\alpha = \begin{cases} 0 & \text{phase 2,} \\ 0 < \alpha < 1 & \text{interface,} \\ 1 & \text{phase 1 (tracked phase).} \end{cases} \tag{4}$$

α equation with phase change manipulation

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) = \frac{\dot{m}}{\rho_l} \tag{5}$$

$$\frac{\partial \overline{\alpha_v}}{\partial t} + \nabla \cdot (\overline{\alpha_v}\widetilde{\mathbf{U}}) = -\frac{\dot{m}}{\rho_v} \tag{6}$$

$$\nabla \cdot \widetilde{\mathbf{U}} = (\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m} \tag{7}$$

Adding and subtracting $\overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}}$ from the right hand side of Eqn. 5 and using Eqn. 7 we get

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) = (\frac{1}{\rho_l} - \overline{\alpha_l}(\frac{1}{\rho} - \frac{1}{\rho_v}))\dot{m} + \overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}} \tag{8}$$

Here we define $\dot{V}' = \frac{1}{\rho_l} - \overline{\alpha_l}(\frac{1}{\rho_l} - \frac{1}{\rho_v})$
Since $\dot{m} = \overline{\alpha_l}\dot{m}_v + (1 - \overline{\alpha_l})\dot{m}_c = \overline{\alpha_l}(\dot{m}_v - \dot{m}_c) + \dot{m}_c$ we get

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) = \dot{V}'(\overline{\alpha_l}(\dot{m}_v - \dot{m}_c) + \dot{m}_c) + \overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}} \tag{9}$$

α equation with phase change manipulation

Then $\dot{V}_v = \dot{V}'\dot{m}_v$ and $\dot{V}_c = \dot{V}'\dot{m}_c$ and substituting them into Eqn. 9 and reorganizing we get

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) - \overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}} = \overline{\alpha_l}\dot{V}_v - \overline{\alpha_l}\dot{V}_c + \dot{V}_c \tag{10}$$

Eqn. 10 is implemented in OpenFOAM (`alphaEqn.H` file) and the code of the equation will be explained in detail.

Lee's phase change model and energy equation

Mass transfer term is calculated by Lee's phase change model

$$\dot{m}_c = r_c \overline{\alpha_v} \rho_v \frac{T_{sat} - \widetilde{T}}{T_{sat}} \qquad \widetilde{T} < T_{sat}(condensation) \tag{11}$$

$$\dot{m}_v = r_v \overline{\alpha_l} \rho_l \frac{\widetilde{T} - T_{sat}}{T_{sat}} \qquad \widetilde{T} > T_{sat}(vaporization) \tag{12}$$

Energy equation

$$\frac{\partial \rho C_p \widetilde{T}}{\partial t} + \nabla \cdot (\rho \widetilde{\mathbf{U}} C_p \widetilde{T}) = \nabla \cdot (K \nabla \widetilde{T}) + \Delta h_v \dot{m} \tag{13}$$

Subgrid scale term in α equation

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l} \widetilde{\mathbf{U}}) = \frac{\dot{m}}{\rho_l} + \tau_{u\alpha} \tag{14}$$

$$\frac{\partial \overline{\alpha_v}}{\partial t} + \nabla \cdot (\overline{\alpha_v} \widetilde{\mathbf{U}}) = -\frac{\dot{m}}{\rho_v} + \tau_{u\alpha} \tag{15}$$

$$\tau_{u\alpha} = \widetilde{\mathbf{U}} \cdot \nabla \overline{\alpha} - \overline{\mathbf{U} \cdot \nabla \alpha} \tag{16}$$

A functional closure model is applied to close the SGS term in the volume fraction equation by the gradient approximation in the Eqn. 16

$$\tau_{u\alpha} = \frac{\nu_{sgs}}{\sigma_{t,u\alpha}} \nabla^2 \bar{\alpha}_l, \tag{17}$$

The subgrid scale viscosity in the wall-adapting linear eddy-viscosity model (WALE) model

$$\nu_{sgs} = (\mathrm{C}_W \Delta)^2 \frac{(\mathbf{S}^d_{ij}\mathbf{S}^d_{ij})^{3/2}}{(\bar{\mathbf{D}}_{ij}\bar{\mathbf{D}}_{ij})^{5/2} + (\mathbf{S}^d_{ij}\mathbf{S}^d_{ij})^{5/4}} \tag{18}$$

$\mathbf{S}_{ij}$ stands for the traceless symmetric part of the square of the velocity gradient tensor.

Volume fraction equation

The reorganized volume fraction Eqn. 10 is implemented in `alphaEqn.H` file.

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) - \overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}} = \overline{\alpha_l}\dot{V}_v - \overline{\alpha_l}\dot{V}_c + \dot{V}_c$$

Alpha equation

```
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
        );
```

Volume fraction equation

$$\frac{\partial \overline{\alpha_l}}{\partial t} + \nabla \cdot (\overline{\alpha_l}\widetilde{\mathbf{U}}) - \overline{\alpha_l}\nabla \cdot \widetilde{\mathbf{U}} = \overline{\alpha_l}\dot{V}_v - \overline{\alpha_l}\dot{V}_c + \dot{V}_c$$

$$\dot{V}_v = \dot{V}'\dot{m}_v \qquad \dot{V}_c = \dot{V}'\dot{m}_c$$

The definition of `vDotmcAlpha`, `vDotvAlpha` and `vDotcAlpha` can be found in the `alphaEqn.H` file from line 10 to 12.

Definition of vDotvmcAlphal

```
9  const volScalarField& vDotcAlphal = vDotAlphal[0]();
10 const volScalarField& vDotvAlphal = vDotAlphal[1]();
11 const volScalarField vDotvmcAlphal(vDotvAlphal - vDotcAlphal);
```

`vDotvAlphal`: $\dot{V}_v$
`vDotcAlphal`: $\dot{V}_c$

Volume fraction equation

In line 10 and 11, vDotAlpha function is defined in temperaturePhaseChange TwoPhaseMixture.C.

$$\dot{V}_v = \dot{V}'\dot{m}_v \qquad \dot{V}_c = \dot{V}'\dot{m}_c \qquad \dot{V}' = \frac{1}{\rho_l} - \overline{\alpha_l}(\frac{1}{\rho_l} - \frac{1}{\rho_v})$$

Definition of vDotAlphal

```
69 Foam::Pair<Foam::tmp<Foam::volScalarField>>
70 Foam::temperaturePhaseChangeTwoPhaseMixture::vDotAlphal() const
71 {
72     volScalarField alphalCoeff
73     (
74         1.0/mixture_.rho1() - mixture_.alpha1()
75         *(1.0/mixture_.rho1() - 1.0/mixture_.rho2())
76     );
77     Pair<tmp<volScalarField>> mDotAlphal = this->mDotAlphal();
78     return Pair<tmp<volScalarField>>
79     (
80     alphalCoeff*mDotAlphal[0],
81     alphalCoeff*mDotAlphal[1]
82     );
83 }
```

Volume fraction equation

From line 72 to line 76, the alphalCoeff which is $\dot{V}'$ in Eqn. 9 is defined as $\frac{1}{\rho_l} - \overline{\alpha_l}(\frac{1}{\rho_l} - \frac{1}{\rho_v})$.

The values returned in vDotAlpha function are `alphalCoeff*mDotAlphal[0]` and `alphalCoeff*mDotAlphal[1]`, which are $\dot{V}_v = \dot{V}'\dot{m}_v$ and $\dot{V}_c = \dot{V}'\dot{m}_c$ in Eqn. 10, respectively.

The mass flow rate term `mDotAlphal[0]` for $\dot{m}_v$ and `mDotAlphal[1]` for $\dot{m}_c$ will be explained in the phase change model.

Pressure correction equation

As stated in Eqn. 7, the divergence of velocity equals to the mass transfer term instead of 0.

$$\nabla \cdot \widetilde{\mathbf{U}} = (\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}$$

So the contribution of mass transfer term is added to the pressure correction equation.

The definition of the pressure correction equation in the `pEqn.H` in the source code file.

Pressure correction equation

```
while (pimple.correctNonOrthogonal())
    {
        fvScalarMatrix p_rghEqn
        (
            fvc::div(phiHbyA)
          - fvm::laplacian(rAUf, p_rgh)
          ==
            vDotv + vDotc
        );
```

Pressure correction equation

In line 37, we see that mass transfer term $(\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}$ is split to vDotv $(\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}_v$ and vDotc $(\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}_c$.

The definition of `vDotv` and `vDotc` can be found in `pEqn.H`

vDotv and vDotc

```
const volScalarField& vDotc = vDot[0]();
const volScalarField& vDotv = vDot[1]();
```

Definition of vDot function

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::temperaturePhaseChangeTwoPhaseMixture::vDot() const
{
    dimensionedScalar pCoeff(1.0/mixture_.rho1() - 1.0/mixture_.rho2());
    Pair<tmp<volScalarField>> mDot = this->mDot();

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

Pressure correction equation

In line 72, the `pCoeff` is defined here as $\frac{1}{\rho_l} - \frac{1}{\rho_v}$. This is the coefficient of the mass transfer term in Eqn. 7. Therefore, the return values `pCoeff*mDot[0]` and `pCoeff*mDot[1]` in line 75 are $(\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}_v$ and $(\frac{1}{\rho_l} - \frac{1}{\rho_v})\dot{m}_c$, respectively. The `mDot` in the return value in the mass flow rate term will be explained in the phase change model.

Energy equation

The energy equation Eqn. 13 is implemented in TEqn.H in the source code folder

$$\frac{\partial \rho C_p \widetilde{T}}{\partial t} + \nabla \cdot (\rho \widetilde{\mathbf{U}} C_p \widetilde{T}) = \nabla \cdot (K \nabla \widetilde{T}) + \Delta h_v \dot{m}$$

T equation

```
18     fvScalarMatrix TEqn
19     (
20         fvm::ddt(rhoCp, T)
21       + fvm::div(rhoCpPhi, T)
22       - fvm::Sp(fvc::ddt(rhoCp) + fvc::div(rhoCpPhi), T)
23       - fvm::laplacian(kappaEff, T)
24       + mixture->TSource()
25     );
```

Similarly, the mass transfer term is also introduced to energy equation in line 24. We will go through this source term in the phase change model.

Phase change model

Phase change term in volume fraction equation
$\dot{m}_v$: `mDotAlphal[0]`
$\dot{m}_c$: `mDotAlphal[1]`

Phase change term in pressure correction equation
$\dot{m}_v$: `mDot[0]`
$\dot{m}_c$: `mDot[1]`

Phase change term in energy equation
$\dot{m}$: `TSource()`

Their declarations and definitions are in the constant folder in temperaturePhaseChangeTwoPhaseMixtures.

Phase change model

The declaration of the member functions of the evaporation model

```
    // Member Functions

        //- Return the mass condensation and vaporisation rates as a
        //  coefficient to multiply (1 - alphal) for the condensation rate
        //  and a coefficient to multiply  alphal for the vaporisation rate
        virtual Pair<tmp<volScalarField>> mDotAlphal() const;

        //- Return the mass condensation and vaporisation rates as coefficients
        virtual Pair<tmp<volScalarField>> mDot() const;

        //- Source for T equation
        virtual tmp<fvScalarMatrix> TSource() const;
```

Definition of mDotAlphal function

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::temperaturePhaseChangeTwoPhaseMixtures::constant::mDotAlphal() const
{
    const volScalarField& T = mesh_.lookupObject<volScalarField>("T");

    const twoPhaseMixtureEThermo& thermo =
        refCast<const twoPhaseMixtureEThermo>
        (
            mesh_.lookupObject<basicThermo>(basicThermo::dictName)
        );

    const dimensionedScalar& TSat = thermo.TSat();

    const dimensionedScalar T0(dimTemperature, Zero);

    return Pair<tmp<volScalarField>>
    (
        coeffC_*mixture_.rho2()*max(TSat - T, T0),
        -coeffE_*mixture_.rho1()*max(T - TSat, T0)
    );
}
```

Phase change model

Definition of mDot function

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::temperaturePhaseChangeTwoPhaseMixtures::constant::mDot() const
{
    volScalarField limitedAlpha1
    (
        min(max(mixture_.alpha1(), scalar(0)), scalar(1))
    );

    volScalarField limitedAlpha2
    (
        min(max(mixture_.alpha2(), scalar(0)), scalar(1))
    );

    ...

}
```

Phase change model

Definition of mDot function

```
Foam::Pair<Foam::tmp<Foam::volScalarField>>
Foam::temperaturePhaseChangeTwoPhaseMixtures::constant::mDot() const
{
    ...

    volScalarField mDotE
    (
        "mDotE", coeffE_*mixture_.rho1()*limitedAlpha1*max(T - TSat, T0)
    );
    volScalarField mDotC
    (
        "mDotC", coeffC_*mixture_.rho2()*limitedAlpha2*max(TSat - T, T0)
    );

    return Pair<tmp<volScalarField>>
    (
        tmp<volScalarField>(new volScalarField(mDotC)),
        tmp<volScalarField>(new volScalarField(-mDotE))
    );
}
```

Phase change model

Definition of Tsource()

```
Foam::tmp<Foam::fvScalarMatrix>
Foam::temperaturePhaseChangeTwoPhaseMixtures::constant::TSource() const
{
    ...
    dimensionedScalar L = mixture_.Hf2() - mixture_.Hf1();
    ...
    const volScalarField Vcoeff
    (
        coeffE_*mixture_.rho1()*limitedAlpha1*L*pos(T - TSat)
    );
    const volScalarField Ccoeff
    (
        coeffC_*mixture_.rho2()*limitedAlpha2*L*pos(TSat - T)
    );

    TSource =
        fvm::Sp(Vcoeff, T) - Vcoeff*TSat
      + fvm::Sp(Ccoeff, T) - Ccoeff*TSat;
        ...
}
```

SGS term

The implementation of the SGS term starts by copying the `interCondensatingEvaporatingFoam` solver, changing its name to `myInterCondensatingEvaporatingFoam`, and renaming its files and functions accordingly.

```
cd $WM_PROJECT_USER_DIR
mkdir -p applications/solvers/multiphase
cd applications/solvers/multiphase
cp -r $FOAM_SOLVERS/multiphase/interCondensatingEvaporatingFoam .

mv interCondensatingEvaporatingFoam //
myInterCondensatingEvaporatingFoam

cd myInterCondensatingEvaporatingFoam

mv interCondensatingEvaporatingFoam.C //
myInterCondensatingEvaporatingFoam.C
```

SGS term

```
cp $FOAM_SOLVERS/multiphase/interPhaseChangeFoam/alphaEqn.H .
cp $FOAM_SOLVERS/multiphase/interPhaseChangeFoam/UEqn.H .
cp $FOAM_SOLVERS/multiphase/interPhaseChangeFoam/alphaControls.H .
cp $FOAM_SOLVERS/multiphase/interPhaseChangeFoam/alphaEqnSubCycle.H
cp $FOAM_SOLVERS/multiphase/interFoam/correctPhi.H .
cp $FOAM_SOLVERS/multiphase/interFoam/initCorrectPhi.H .
cp $FOAM_SOLVERS/multiphase/interFoam/rhofs.H .
cp $FOAM_SOLVERS/multiphase/VoF/setDeltaT.H .
cp $FOAM_SOLVERS/multiphase/VoF/createAlphaFluxes.H .
```

Rename within the files in Make directory to change the name and path of the excutable:

```
sed -i s/inter/myInter/g Make/files
sed -i s/APPBIN/USER_APPBIN/g Make/files
```

Then change the `interCondensatingEvaporatingFoam` to `myInterCondensatingEvaporatingFoam` in `myInterCondensatingEvaporatingFoam.C`:

```
sed -i s/interCon/myInterCon/g myInterCondensatingEvaporatingFoam.C
```

SGS term

Rename the library to our library in the source code folder:

```
mv temperaturePhaseChangeTwoPhaseMixtures //
myTemperaturePhaseChangeTwoPhaseMixtures
```

Then the `Make/options` files in the source code folder and the `myTemperature PhaseChangeTwoPhaseMixtures` folder should be updated, respectively.

SGS term

Make/options in the source code

```
EXE_INC = \
    -ImyTemperaturePhaseChangeTwoPhaseMixtures/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/fvOptions/lnInclude\
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -I$(LIB_SRC)/dynamicFvMesh/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/twoPhaseMixture/lnInclude \
    -I$(LIB_SRC)/transportModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels/interfaceProperties/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/incompressible/lnInclude
```

SGS term

Make/options in the source code

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

SGS term

Then rename within the files in the `Make` directory of `myTemperaturePhaseChangeTwoPhaseMixtures` to change the name and path of the library:

```
sed -i s/libphase/libmyPhase/g  //
myTemperaturePhaseChangeTwoPhaseMixtures/Make/files

sed -i s/LIBBIN/USER_LIBBIN/g   //
myTemperaturePhaseChangeTwoPhaseMixtures/Make/files
```

Now we have a local version of the interCondensatingEvaporatingFoam solver. Try to compile it to see that everything works as intended

```
wmake myTemperaturePhaseChangeTwoPhaseMixtures
wmake
```

SGS term

Then we modify the `alphaEqn.H` to add the SGS source term to the volume fraction equation. First of all, we need to create a new file for SGS term.

$$\tau_{u\alpha} = \frac{\nu_{sgs}}{\sigma_{t,u\alpha}} \nabla^2 \bar{\alpha}_l \qquad \nu_{sgs} = (C_W \Delta)^2 \frac{(\mathbf{S}^d_{ij}\mathbf{S}^d_{ij})^{3/2}}{(\bar{\mathbf{D}}_{ij}\bar{\mathbf{D}}_{ij})^{5/2} + (\mathbf{S}^d_{ij}\mathbf{S}^d_{ij})^{5/4}}$$

Create `alphaTUA_WALE.H` file and add the following piece of code for the SGS term to the file:

Code for the SGS term

```
//Get the mesh size
volScalarField V
    (
            IOobject
        (
            mesh.V().name(),
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE,
            false
        ),
```

SGS term

Code for the SGS term

```
            mesh,
           dimensionedScalar(mesh.V().dimensions(), Zero),
           calculatedFvPatchField<scalar>::typeName
    );
       V.ref() = mesh.V();
volScalarField delta = pow(V, 1./3);
//Create the variables for SGS viscosity
volTensorField gradU(fvc::grad(U));
volSymmTensorField Sd(dev(symm(gradU & gradU)));
volScalarField magSqrSd(magSqr(Sd));
volScalarField nuSGS =
          sqr(0.5*delta/1.0)*
             sqrt(
                 pow(magSqrSd, 3.0)
                /(
                    sqr
                    (
                        pow(magSqr(symm(gradU)), 5.0/2.0)
                      + pow(magSqrSd, 5.0/4.0)
                    )
```

SGS term

Code for the SGS term

```
                + dimensionedScalar
                    (
                        "SMALL",
                        dimensionSet(0, 0, -10, 0, 0),
                        SMALL
                    )
                )
            );

//SGS term
volScalarField TUA
(
      nuSGS*fvc::laplacian(alpha1)
);
```

Now we have created the file for SGS term, the next step is to add this term to `alphaEqn.H` file.

SGS term

Firstly, we include the file in line 17 and then add the source term we created in line 30.

Code for alpha equation

```
15      if (MULESCorr)
16      {
17          #include "alphaTUA_WALE.H"
18          fvScalarMatrix alpha1Eqn
19          (
20              fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
21            + fv::gaussConvectionScheme<scalar>
22              (
23                  mesh,
24                  phi,
25                  upwind<scalar>(mesh, phi)
26              ).fvmDiv(phi, alpha1)
27            - fvm::Sp(divU, alpha1)
28           ==
29              fvm::Sp(vDotvmcAlphal, alpha1)
30            + vDotcAlphal + TUA
31          );
```

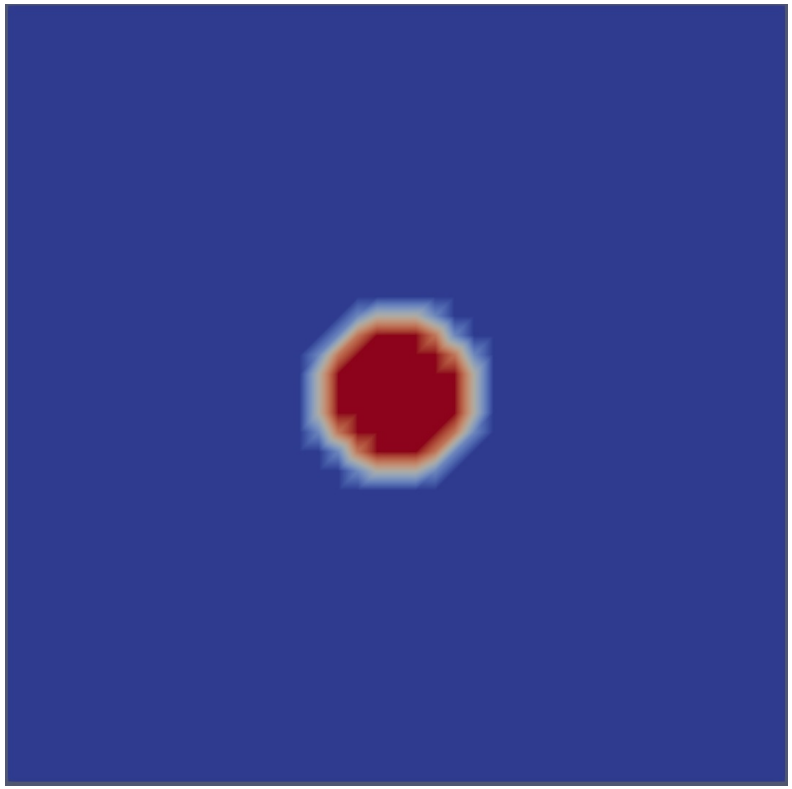
SGS term

Then the SGS source term has been implemented into the volume fraction equation. Try to compile it to see that everything works as intended

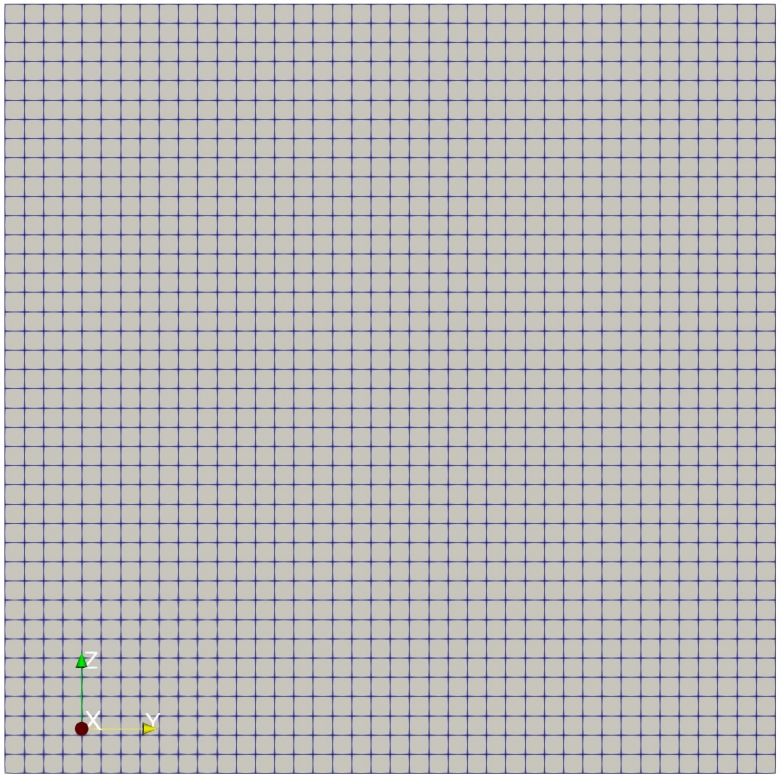
```
wclean
wmake
```

Test case

The original solver `interCondensatingEvaporatingFoam` and the modified solver `myInterCondensatingEvaporatingFoam` will be tested by applying them to a liquid droplet in vapor in a box. The droplet is initialized with a specific diameter and position.



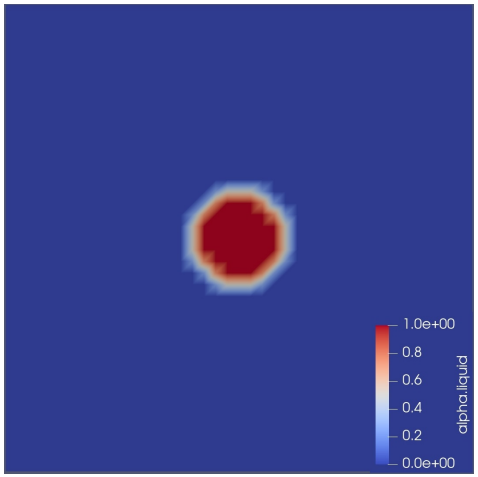
Computational domain



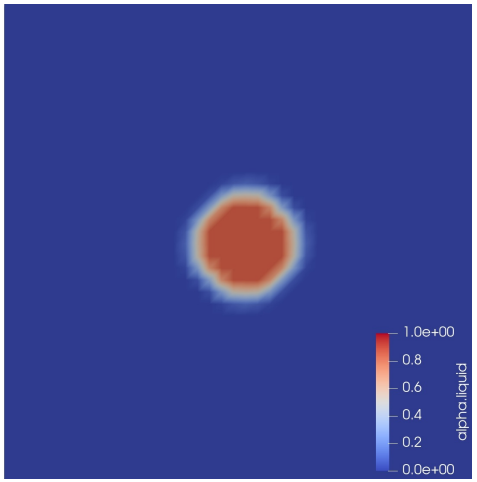
Mesh

Results

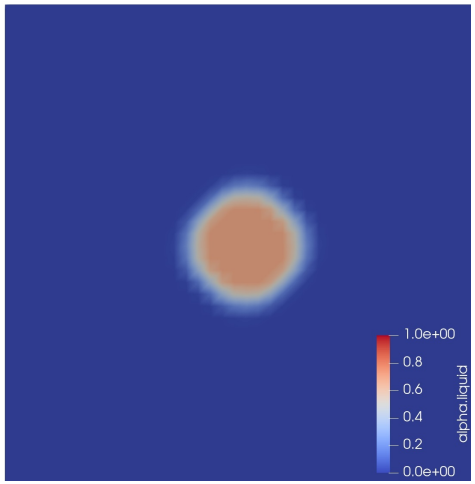
`interCondensatingEvaporatingFoam`



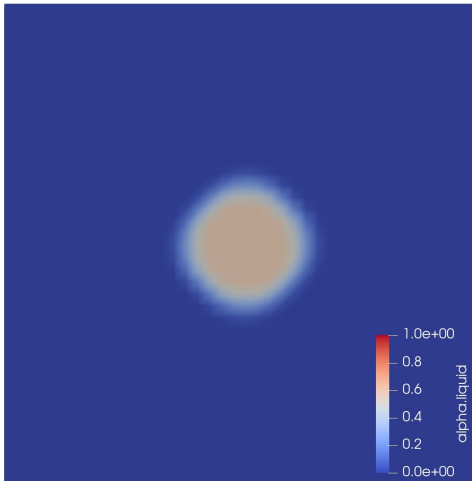
0 s



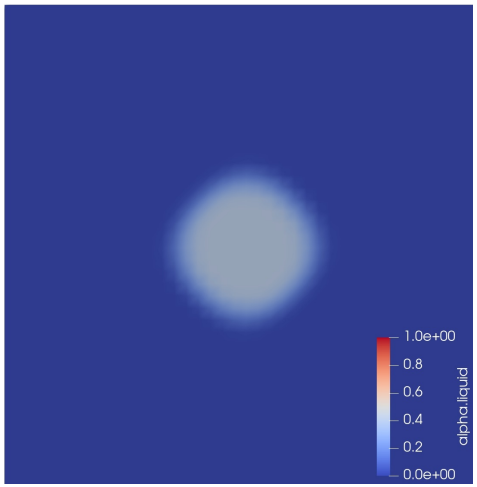
1e-7 s



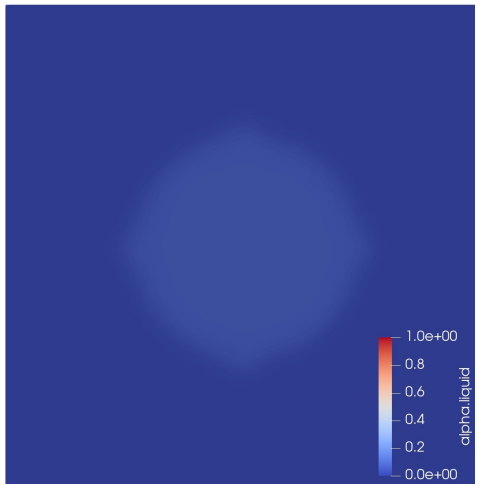
3e-7 s



5e-7 s



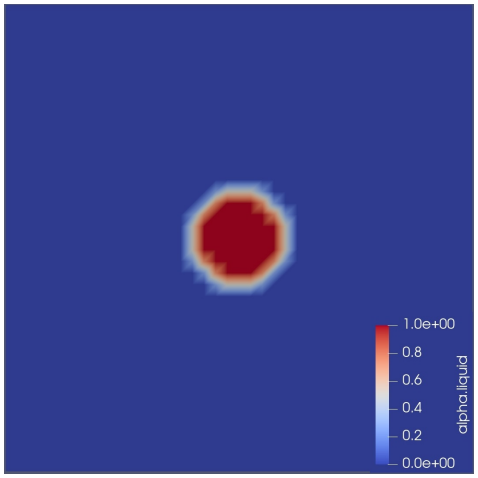
1e-6 s



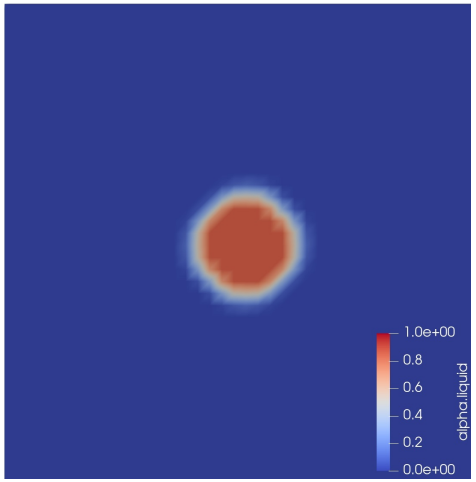
1e-5 s

Results

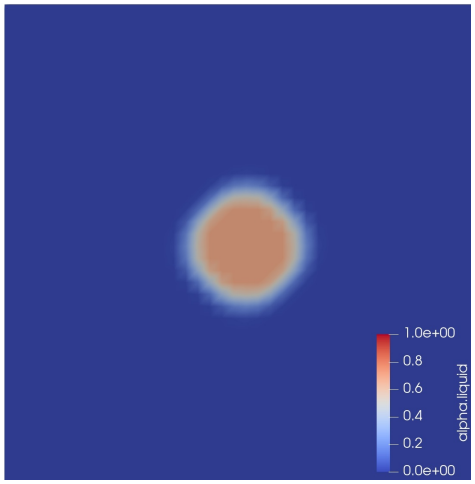
`myInterCondensatingEvaporatingFoam`



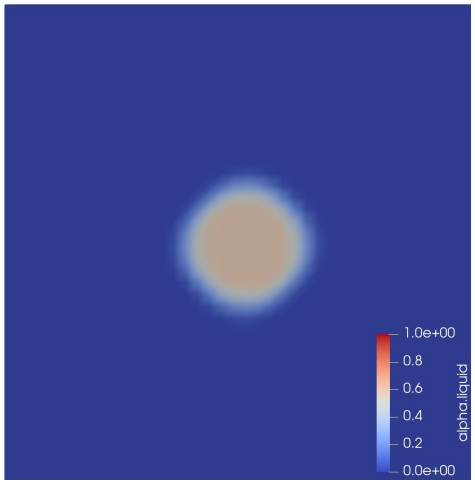
0 s



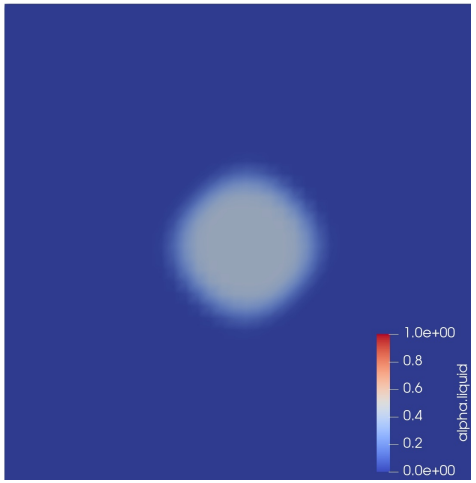
1e-7 s



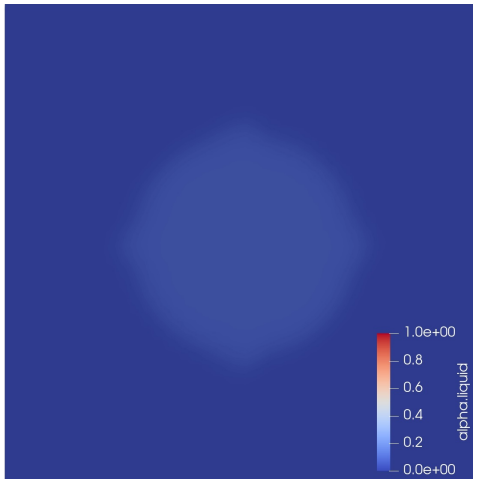
3e-7 s



5e-7 s



1e-6 s



6e-6 s