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

Bjørn Christian Dueholm

Department of Materials and Production,
Aalborg University,
Aalborg, Denmark

January 20, 2022

Agenda

- Background
- interPhaseChangeFoam
- Schnerr and Sauer
- Modifications
- Implementation
- Test-case

The technology

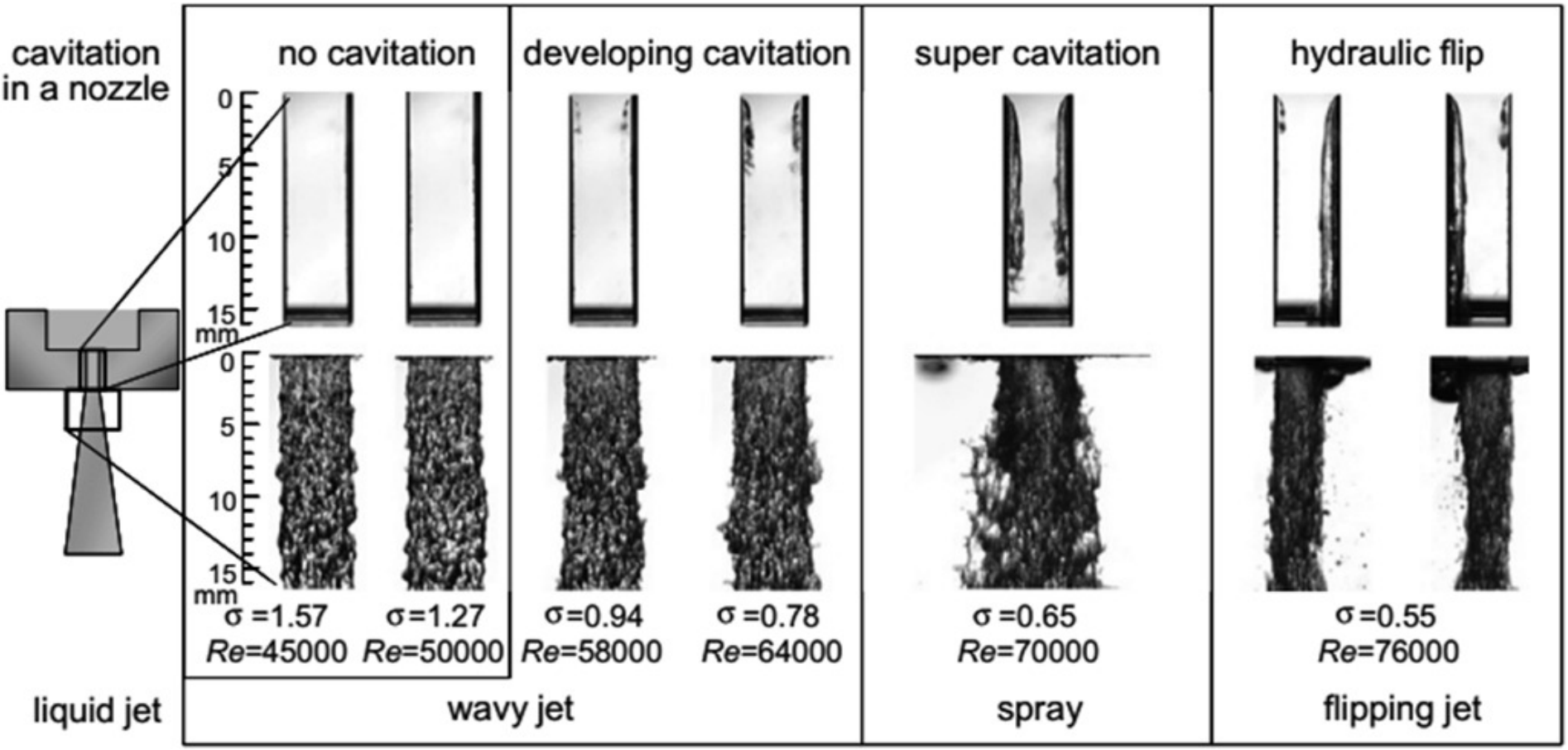
Marine engines

Wartsila Sulzer RTA96-C

http://www.emma-maersk.com/engine/Wartsila_Sulzer_RTA96-C.htm

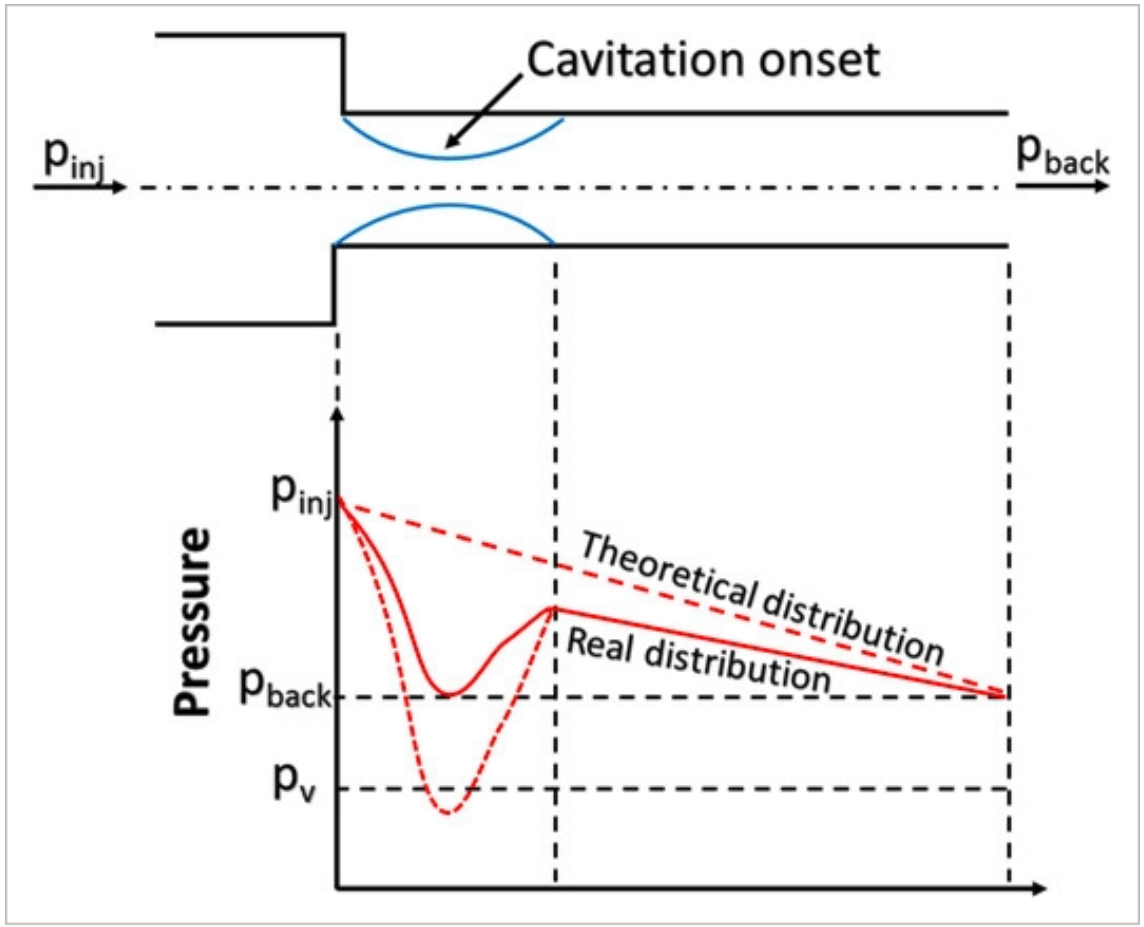
Typical injector

Physical phenomena



Sun, Y., Guan, Z. and Hooman, K. (2019), Cavitation in Diesel Fuel Injector Nozzles and its Influence on Atomization and Spray. Chem. Eng. Technol. 42: 6-29.

Physical phenomena



Sun, Y., Guan, Z. and Hooman, K. (2019), Cavitation in Diesel Fuel Injector Nozzles and its Influence on Atomization and Spray. Chem. Eng. Technol. 42: 6-29.

VoF and Governing Equations

$$\alpha_i = \frac{V_i}{V} \qquad \sum_{i=1}^{2} \alpha_i = 1$$

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

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

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

Alpha equation manipulation

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

To stabilise the solution the source term is modified.

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

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

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

Alpha equation manipulation

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

phaseChangeTwoPhaseMixture.C

```
55 Foam::Pair<Foam::tmp<Foam::volScalarField>>
56 Foam::phaseChangeTwoPhaseMixture::vDotAlphal() const
57 {
58     volScalarField alphalCoeff(1.0/rho1() - alpha1_*(1.0/rho1() - 1.0/rho2()));
59     Pair<tmp<volScalarField>> mDotAlphal = this->mDotAlphal();
60
61     return Pair<tmp<volScalarField>>
62     (
63         alphalCoeff*mDotAlphal[0],
64         alphalCoeff*mDotAlphal[1]
65     );
66 }
```

Alpha equation manipulation

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

alphaEqn.H

```
17 fvScalarMatrix alpha1Eqn
18 (
19     fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
20   + fv::gaussConvectionScheme<scalar>
21     (
22         mesh,
23         phi,
24         upwind<scalar>(mesh, phi)
25     ).fvmDiv(phi, alpha1)
26   - fvm::Sp(divU, alpha1)
27  ==
28     fvm::Sp(vDotvmcAlphal, alpha1)
29   + vDotcAlphal
30 );
```

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

Cavitation model

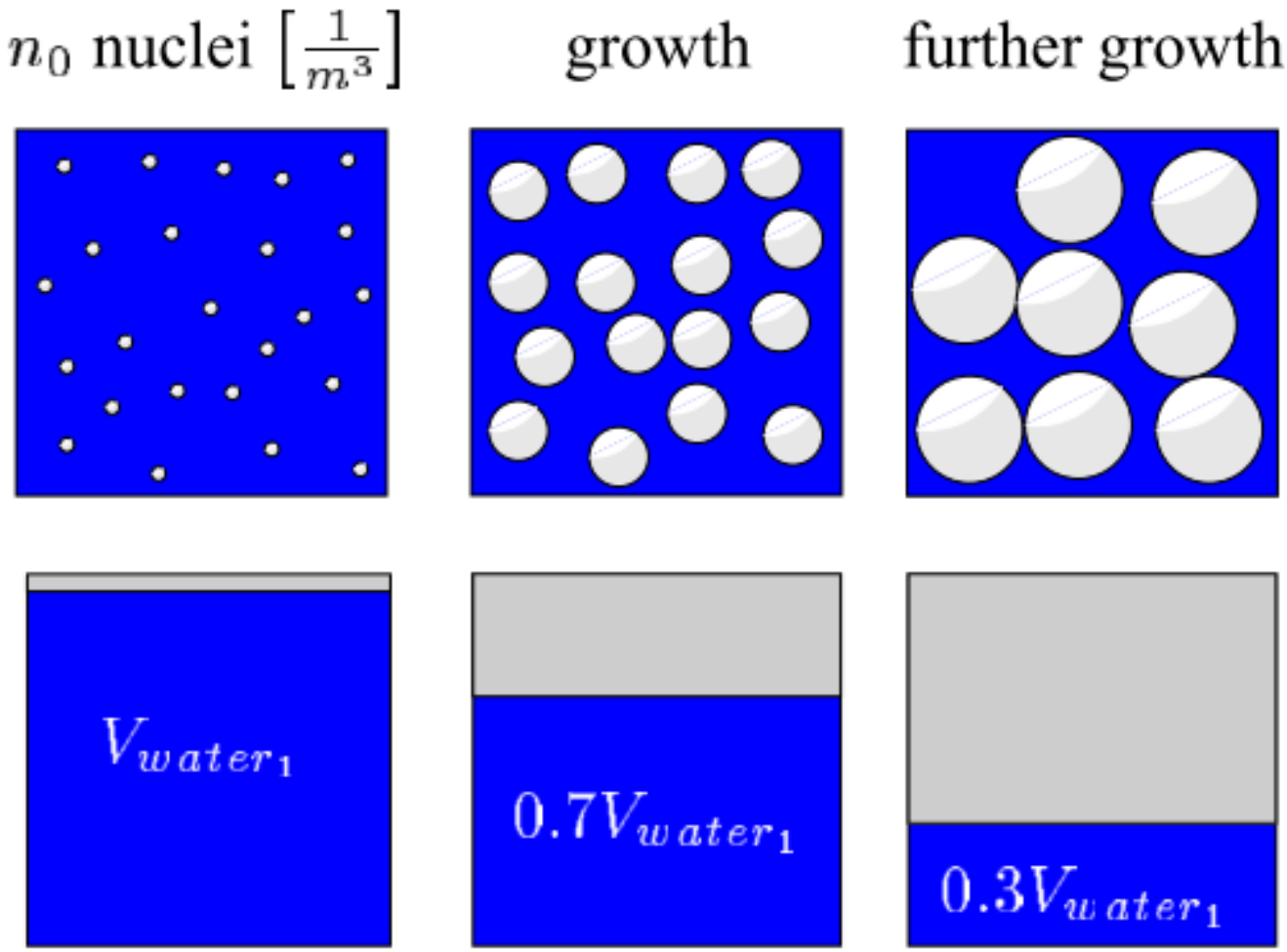
The cavitation model is responsible for delivering the mass transfer source terms. The model by Schnerr and Sauer has the mass transfer terms defined as

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

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

How did we get to this point?

SchnerrSauer



Schnerr Professor Dr.-Ing.habil, Günter. (2001). Physical and Numerical Modeling of Unsteady Cavitation Dynamics.

SchnerrSauer

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

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

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

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

Bubble Growth

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

The last part of the mass transfer source term comes from the Rayleigh-Plesset equation

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

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

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

Alpha Equations

$$\sum_{i=1}^{3} \alpha_i = 1$$

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

Sum of equations is still the same, which means there is a lot, that does not change by adding an additional non-condensing phase.

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

Alpha Equation

$$\dot{m} = \alpha_l \dot{m}_v + (1 - \alpha_l - \alpha_{nc})\, \dot{m}_c = \alpha_l \left(\dot{m}_v - \dot{m}_c\right) + (1 - \alpha_{nc})\, \dot{m}_c$$

Which means, that the implementation of the α_l transport equation must change as well

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

alphaEqn.H

```
fvScalarMatrix alpha1Eqn
(
    fvm::ddt(alpha1)
  + fvm::div(phi, alpha1, "div(phi,alpha)")
  - fvm::Sp(divU, alpha1)
  ==
    fvm::Sp(vDotvmcAlphal, alpha1)
  + vDotcAlphal*(1-alpha3)
);
```

Cavitation model

Following the derivation of the SchnerrSauer source terms, a new term can be obtained for the mass transfer.

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

Unfortunately, the same procedure can not be used to find a term for the time rate of change of α_v. Instead the following equation is used, which comes directly from the nucleation site assumption.

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

Cavitation model

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

Using the above, it is possible to show, that the time rate of change of α_v becomes the following.

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

And the total mass transfer terms become.

$$\dot{m}_{pc} = \frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} \frac{3}{\left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right)\frac{4}{3}\pi n_0 R^4 + R}\frac{dR}{dt}$$

$$\dot{m}_{pc} = -\frac{\rho_v \rho_l}{\rho + \alpha_{nc}(\rho_l - \rho_{nc})} \frac{4 n_0 \pi R^2}{1 + n_0\frac{4}{3}\pi R^3\left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right)}\frac{dR}{dt}$$

Phase change library

The first and biggest issue for doing the implementation is that the entire phase change library is inheriting from `incompressibleTwoPhaseMixture`

`phaseChangeTwoPhaseMixture.H`

```
class phaseChangeTwoPhaseMixture
:
    public incompressibleTwoPhaseMixture
{
```

A consequence of this is that all the inherited and used functions only take into account, that 2 phases can exist in the cell at the same time. Luckily `interMixingFoam` has the functionality needed.

interMixingFoam

```
interMixingFoam
├── immiscibleIncompressibleThreePhaseMixture
├── incompressibleThreePhaseMixture
└── threePhaseInterfaceProperties
```

immiscibleIncompressibleThreePhaseMixture inherits functionality from the two other libraries, so it is merged with phaseChangeTwoPhaseMixture to produce phaseChangeThreePhaseMixture

phaseChangeThreePhaseMixture.H

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

Steps to get there

implementing the library

```
cp -r $FOAM_APP/solvers/multiphase/interPhaseChangeFoam .
cp -r $FOAM_APP/solvers/multiphase/interFoam/interMixingFoam .
rm -r interPhaseChangeFoam/overInterPhaseChangeDyMFoam/
rm -r interPhaseChangeFoam/interPhaseChangeDyMFoam/
mv interMixingFoam/incompressibleThreePhaseMixture/ interPhaseChangeFoam/
mv interMixingFoam/threePhaseInterfaceProperties/ interPhaseChangeFoam/
rm -r interMixingFoam/
cd interPhaseChangeFoam
mv phaseChangeTwoPhaseMixtures/ phaseChangeThreePhaseMixtures/
cd phaseChangeThreePhaseMixtures/
rm -r Kunz/
rm -r Merkle/
mv phaseChangeTwoPhaseMixture/ phaseChangeThreePhaseMixture/
cd phaseChangeThreePhaseMixture/
mv phaseChangeTwoPhaseMixture.C phaseChangeThreePhaseMixture.C
mv phaseChangeTwoPhaseMixtureNew.C phaseChangeThreePhaseMixtureNew.C
mv phaseChangeTwoPhaseMixture.H phaseChangeThreePhaseMixture.H
sed -i 's/Two/Three/g' *
cd ../SchnerrSauer/
sed -i 's/Two/Three/g' *
```

Steps to get there

In `phaseChangeThreePhaseMixture.H` include the following

`phaseChangeThreePhaseMixture.H`

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

Change inheritance as.

`phaseChangeThreePhaseMixture.H`

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

Steps to get there

Change the correct functions as.

phaseChangeThreePhaseMixture.H

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

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

Steps to get there

Merge the constructors from `phaseChangeThreePhaseMixture` and `immiscibleIncompressibleThreePhaseMixture`.

`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)
{}
```

Steps to get there

Open `incompressibleThreePhaseMixture.C` and change line 161 to the following.

`incompressibleThreePhaseMixture.C`

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

Then open `incompressibleThreePhaseMixture.H` and change the private data to protected to make the inheritance possible.

`incompressibleThreePhaseMixture.H`

```
class incompressibleThreePhaseMixture
:
    public IOdictionary,
    public transportModel
{
protected:
    // protected data
```

Steps to get there

Last thing is to change `files` and `options` to the following.

phaseChangeThreePhaseMixtures/Make/files

```
phaseChangeThreePhaseMixture/phaseChangeThreePhaseMixture.C
phaseChangeThreePhaseMixture/phaseChangeThreePhaseMixtureNew.C
SchnerrSauer/SchnerrSauer.C
LIB = $(FOAM_USER_LIBBIN)/libphaseChangeThreePhaseMixtures
```

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
```

Accommodating the new solver

The library should now be able to compile with `wmake`
Initialisation in `CreateFields.H`

`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();
```

This results in `interface.` having to be replaced with `mixture->` everywhere.

alphaEqn.H

alphaEqn.H

```
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));
```

Final Steps

Together with a long list of small changes, that is the main changes required to facilitate the new library.
The last thing to do is to implement the derived mass transfer terms in the `SchnerrSauer` model.

$$\dot{m}_{pc} = \frac{\rho_v \rho_l}{\rho + \alpha_{nc}\left(\rho_l - \rho_{nc}\right)} \frac{3}{\left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right) \frac{4}{3}\pi n_0 R^4 + R} \frac{dR}{dt}$$

$$\dot{m}_{pc} = -\frac{\rho_v \rho_l}{\rho + \alpha_{nc}\left(\rho_l - \rho_{nc}\right)} \frac{4 n_0 \pi R^2}{1 + n_0 \frac{4}{3}\pi R^3 \left(\frac{\rho+\alpha_{nc}(\rho_v-\rho_{nc})}{\rho+\alpha_{nc}(\rho_l-\rho_{nc})}\right)} \frac{dR}{dt}$$

Steps to get there

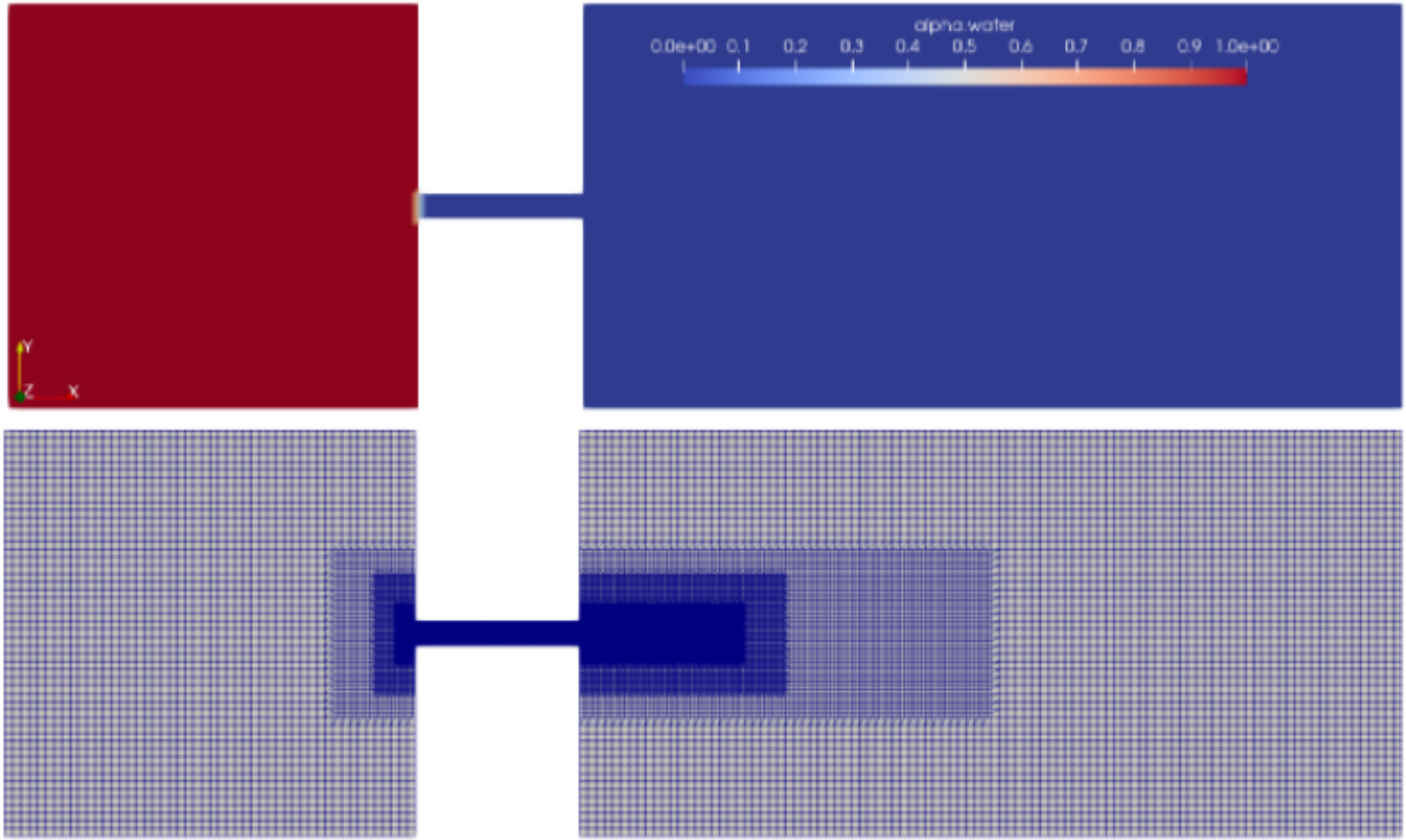
Implementation in `mDotAlphal()` in `SchnerrSauer.C`

SchnerrSauer.C

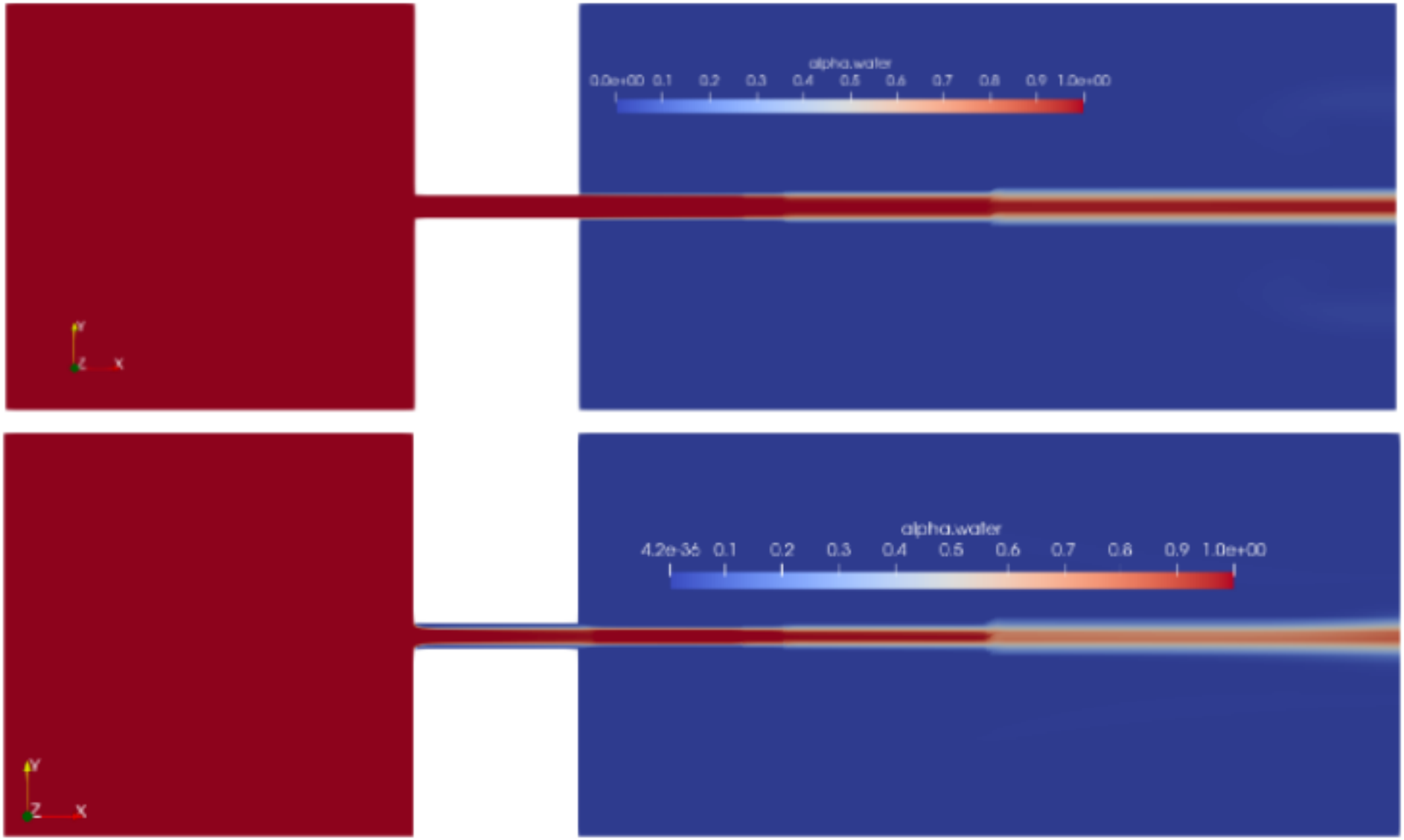
```
    volScalarField densityfrac
    (
        (rho+limitedAlpha3*(rho2()-rho3()))/(rho+limitedAlpha3*(rho1()-rho3()))
    );
    return Pair<tmp<volScalarField>>
    (
        Cc_
        *((3.0)
        /(rRb(limitedAlpha1, limitedAlpha3) + pow(rRb(limitedAlpha1,
     limitedAlpha3),4)
        *(4.0/3.0)*n_*constant::mathematical::pi*densityfrac))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

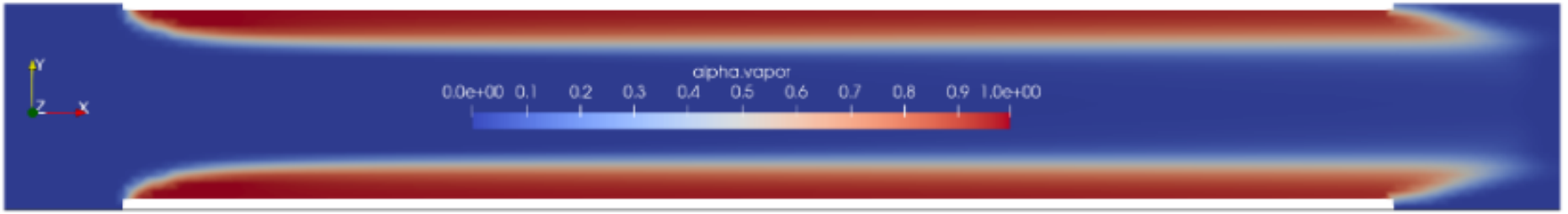
cavitatingFoam tutorial

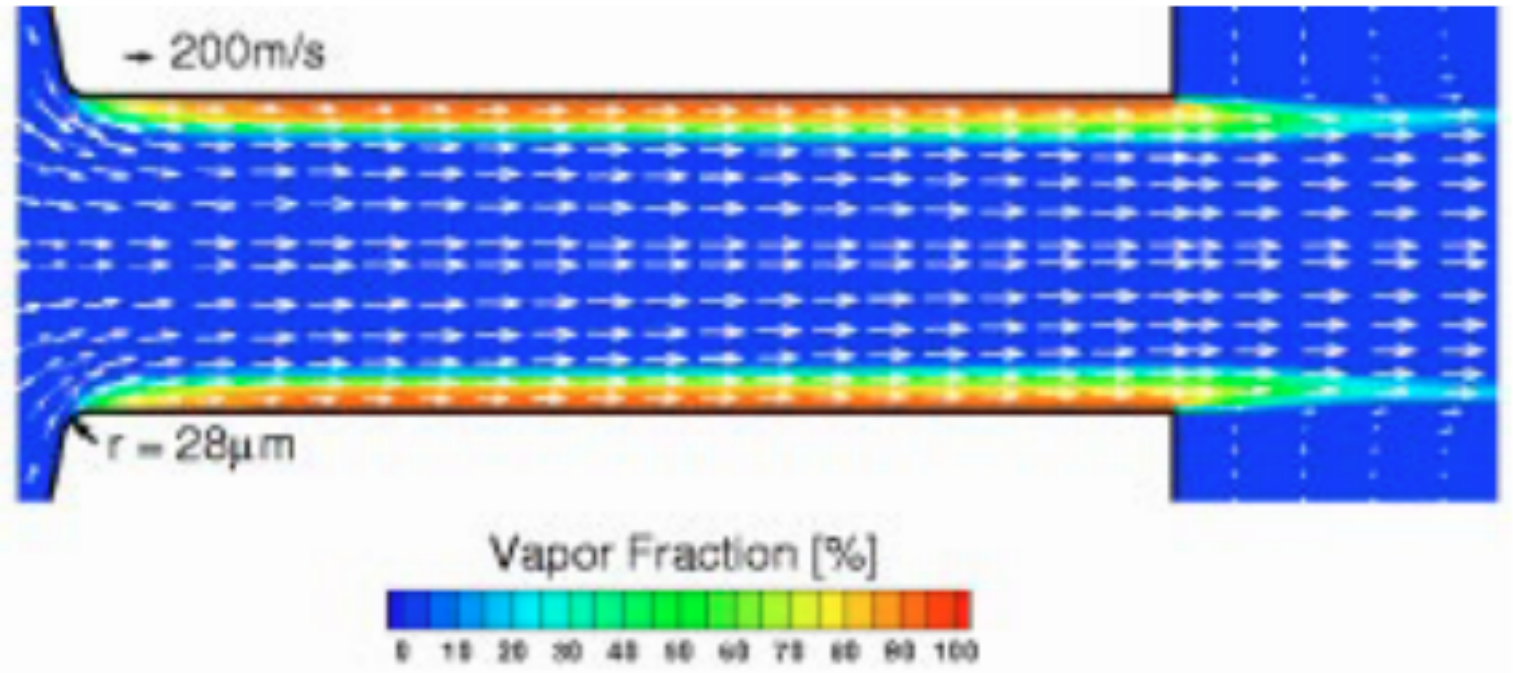


Cavitation terms on



Cavitation pockets





Yuan, W. and Schnerr, G. H. (2003), Numerical simulation of two-phase flow in injection nozles: Interaction of cavitation and external jet formation.

Journal of Fluids Engineering. Volume 125 Issue 6

That's it

Thank you for your attention!