Free-surface flow simulation: implementation and running cases with OpenFOAM®

Dr. Edin Berberović

Polytechnic faculty, University of Zenica

eberberovic@ptf.unze.ba

CEEPUS III Network
Building Knowledge and Experience Exchange in CFD

Overview

1. Introduction
2. Isothermal flow solver `interFoam`
3. Create new non-isothermal flow solver `interThermalFoam`

foam-extend release[1] The Extend Project of OpenFOAM®[2]

- A short introduction to the existing solver `interFoam`
- Set-up and run a case of an isothermal drop impact with the `interFoam` solver
- Extension of the existing code to account for the energy transport
  - edit/extend the existing the `transportModel` library
  - create a new `interThermalFoam` solver to enable the solution of the energy equation
  - set-up and run a case of a non-isothermal drop impact with the new solver `interThermalFoam`

[1] [foam-extend, 2016]
[2] [OpenFOAM, 2016]

Solver interFoam

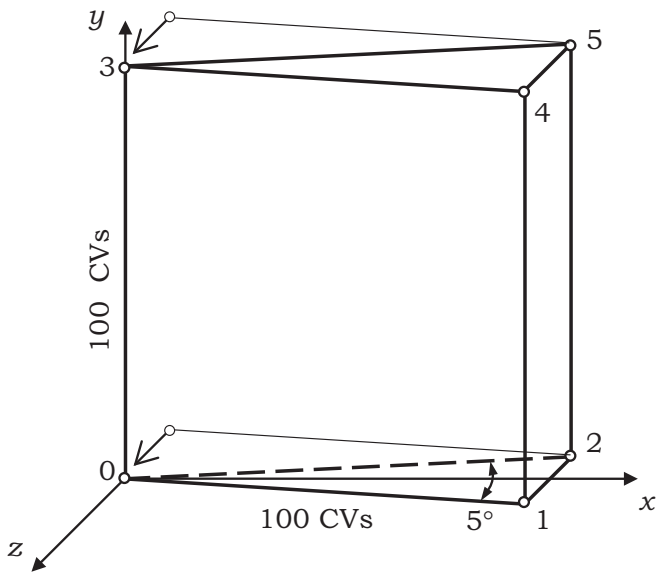
- Include all the necessary files from the library
- Create the Time and fvMesh objects
- Read/create the fields p_d, α, $\mathbf{U}$, ρ, $\rho\mathbf{U}_f \cdot \mathbf{S}_f$, ...
- Create twoPhaseMixture and interfaceProperties objects
- Read simulation/solution controls and set the time step
- Time loop
  - calculate the kinematic viscosity field of the mixture $\rightarrow$ new ν
  - solve the phase fraction equation $\rightarrow$ new α
  - calculate mass fluxes, interface curvature and density $\rightarrow$ new $\rho\mathbf{U}_f \cdot \mathbf{S}_f$, κ, and ρ
  - PISO loop
    - solve the momentum equation
    - construct and solve pressure equation $\rightarrow$ new p_d
    - update volume fluxes and velocities $\rightarrow$ new $\mathbf{U}_f \cdot \mathbf{S}_f$ and $\mathbf{U}$
  - calculate the static pressure from p_d $\rightarrow$ new p
  - calculate turbulence (if needed)

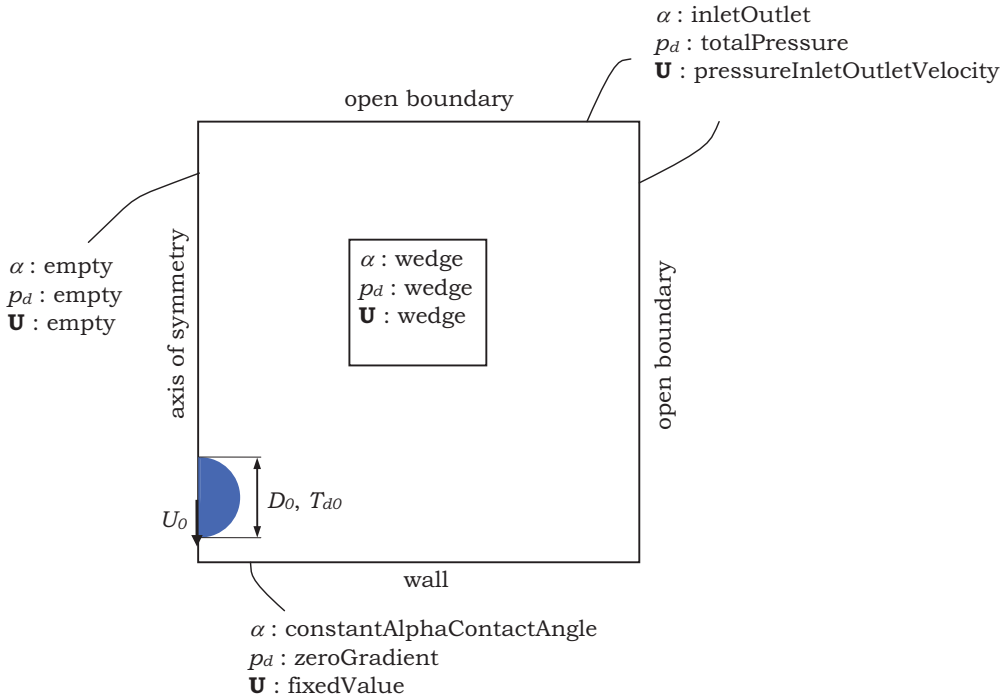
Preparing the case dropImpact

- Creating the mesh
  - `blockMeshDict` for 2D-axisymmetric case
- Initialization
  - 0 directory for initial field values and boundary conditions
  - `setFieldsDict` for initialization
- Transport properties
  - `transportProperties` for setting thermophysical properties for both fluids
- Discretization schemes
  - `fvSchemes` for space and time discretization schemes
- Iterative solvers
  - `fvsolution` for iterative solvers of algebraic equation systems
- Simulation parameters
  - `controlDict` for solution controls and parameters

dropImpact: axisymmetric case configuration

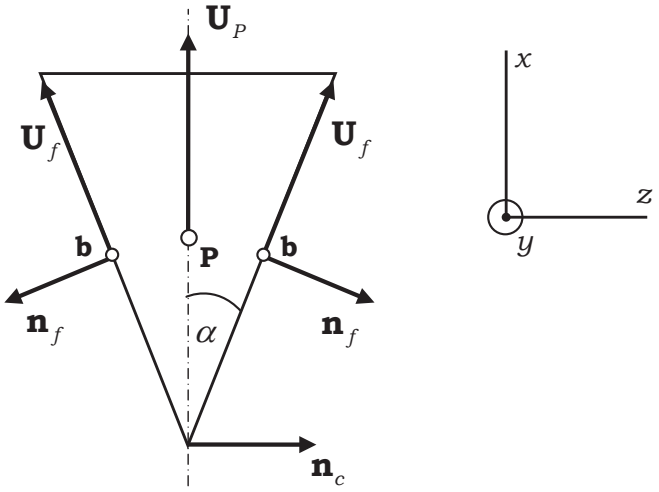
- Axisymmetric mesh (wedge-type slice)
- Case parameters: D_0, U_0, θ, ρ_l, ν_l, ρ_g, ν_g, σ





Boundary conditions

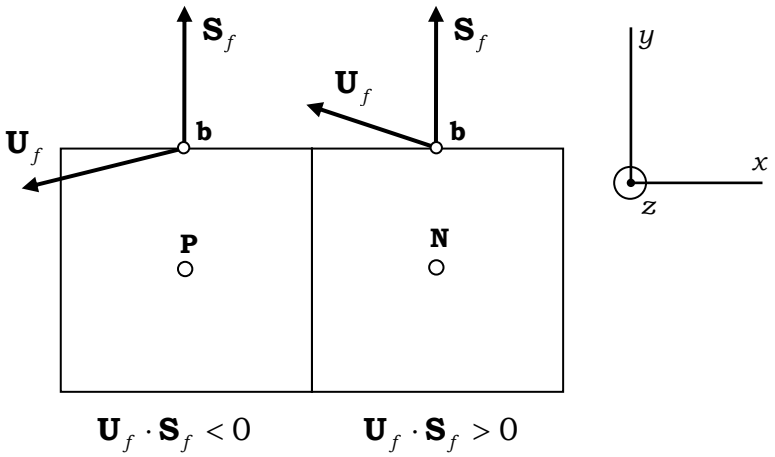
- `empty`, suppresses the matrix coefficients
- `wedge`, transfers `scalarFields` and rotates `vectorFiels`



$$\mathbf{U}_b = \begin{vmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{vmatrix} \cdot \begin{vmatrix} U_{Px} \\ U_{Py} \\ 0 \end{vmatrix} = U_{Px}\cos\alpha\mathbf{x}_0 + U_{Py}\mathbf{y}_0 + U_{Px}\sin\alpha\mathbf{z}_0$$

Boundary conditions

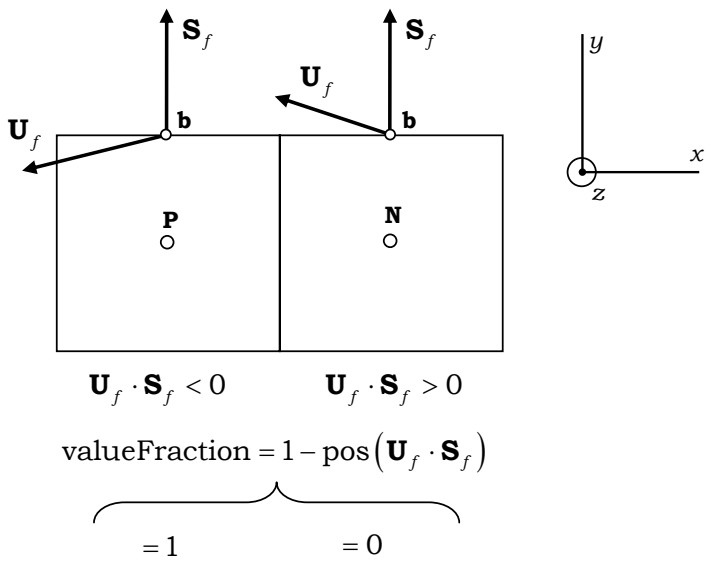
- `totalPressure`, adjusts static pressure from total pressure $p_0 = p_d + 0.5\rho\mathbf{U}^2$ according to the sign of the flux



$$p_{d,b} = \begin{cases} p_0 - \frac{1}{2}\rho\mathbf{U}_b^2, \text{ for } \mathbf{U}_f \cdot \mathbf{S}_f < 0 \\ p_0, \text{ for } \mathbf{U}_f \cdot \mathbf{S}_f > 0 \end{cases}$$

Boundary conditions

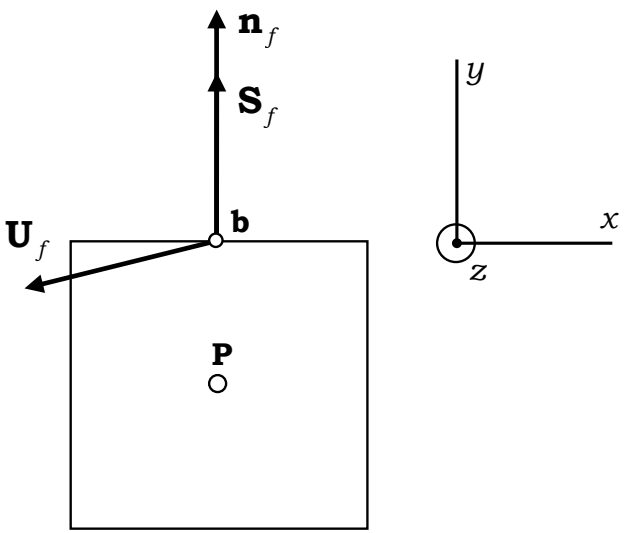
- `inletOutlet`, switches between `fixedValue` and `zeroGradient` depending on the sign of the flux



$$\phi_b = \begin{cases} \phi_N, & \text{for } \mathbf{U}_f \cdot \mathbf{S}_f > 0 \\ \phi_{b,ref}, & \text{for} \quad \text{for } \mathbf{U}_f \cdot \mathbf{S}_f < 0 \end{cases}$$

Boundary conditions

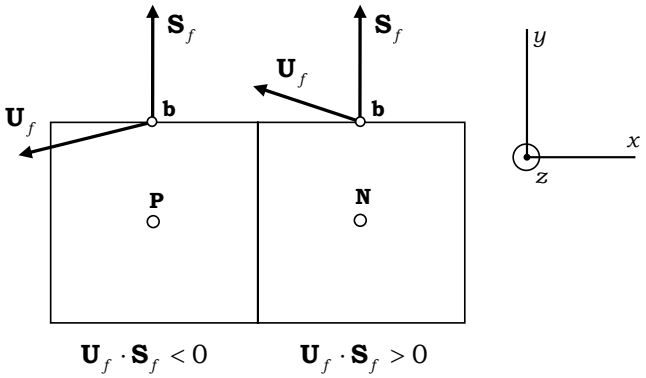
- `pressureInletVelocity`, velocity is calculated from the flux at the boundary



$$\mathbf{U}_b = \mathbf{n}_f \cdot \frac{\mathbf{U}_f \cdot \mathbf{S}_f}{|\mathbf{S}_f|} = (0, U_{fy}\mathbf{n}_f, 0)$$

Boundary conditions

- `pressureInletOutletVelocity`, combines `inletOutlet` and `pressureInletVelocity`



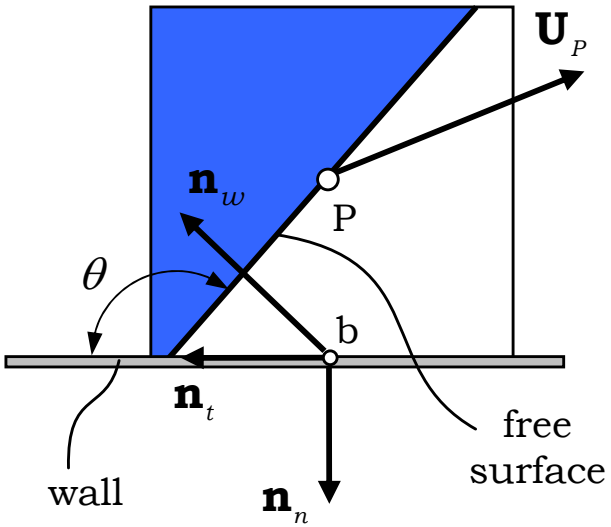
$$\text{valueFraction} = \underbrace{\text{neg}\left(\mathbf{U}_f \cdot \mathbf{S}_f\right)\left(\mathbf{I} - \mathbf{n}_f \mathbf{n}_f\right)}_{= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}}$$

$$\mathbf{U}_b = \begin{cases} \mathbf{U}_N, & \text{for } \mathbf{U}_f \cdot \mathbf{S}_f > 0 \\ \mathbf{U}_{Py}, & \text{for} \quad \text{for } \mathbf{U}_f \cdot \mathbf{S}_f < 0 \end{cases}$$

Boundary conditions

- Boundary condition for the contact angle
  - set the value for the contact angle θ at the wall boundaries
  - used to calculate the curvature in the cells closest to the walls
  - needed for proper calculation of the surface tension force in that cells $\sigma\kappa\nabla\alpha = \sigma(-\nabla\cdot\mathbf{n})\nabla\alpha$

$$\mathbf{n}_w = \left(\frac{\nabla\alpha}{|\nabla\alpha|}\right)_w = \mathbf{n}_n\cos\theta + \mathbf{n}_t\sin\theta$$



Running the case and viewing results

- Case parameters

$$D_0 = 2\ \mathrm{mm}, U_0 = 1\ \mathrm{m/s}, \theta = 60^\circ$$

$$\rho_l = 1000\ \mathrm{kg/m^3}, \nu_l = 10^{-6}\ \mathrm{m^2/s}$$

$$\rho_g = 1.18\ \mathrm{kg/m^3}, \nu_g = 1.5 \cdot 10^{-5}\ \mathrm{m^2/s}$$

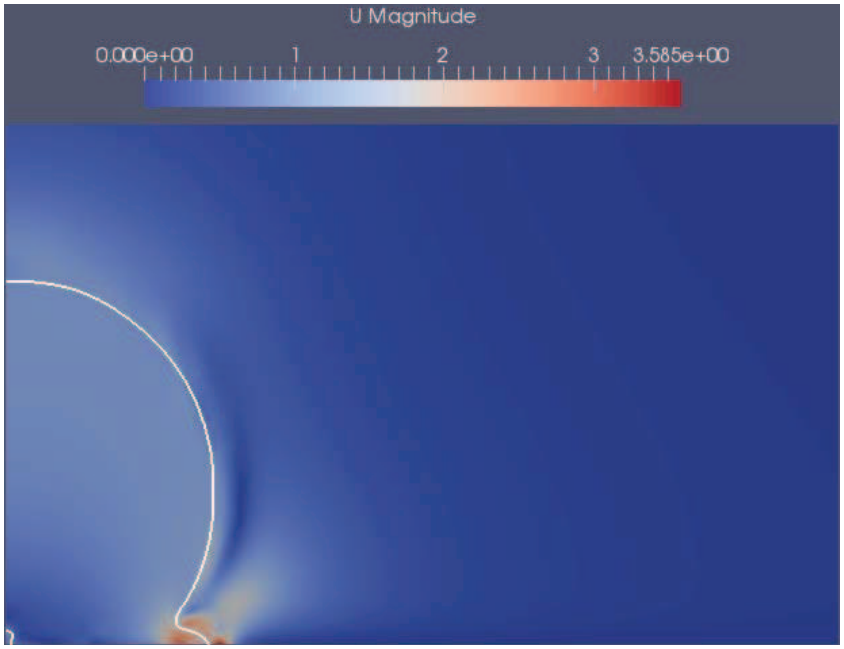
$$\sigma = 0.072\ \mathrm{N/m}$$

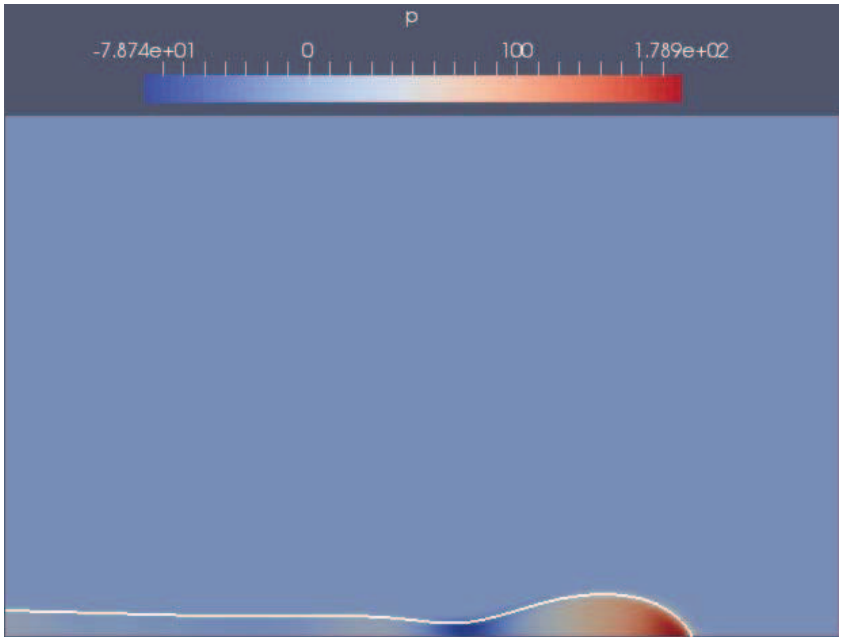
- Run the case in console from dropImpact directory
  - execute from command line

```
run
cd dropImpact
blockMesh
setFields
interFoam
```

Running and viewing results

- View the results in ParaView
  - execute the script `paraFoam`
  - show the free-surface: create the iso-contour of $\alpha = 0.5$
  - visualize the fields α, $\mathbf{U}$, p_d and p





Extending the transportModel library

- Copy the solver and the original library from the source directory and rename

```
cd $WM_PROJECT_USER_DIR/applications/solvers
cp -r $FOAM_APP/solvers/multiphase/interFoam .
mv interFoam interThermalFoam
cd interThermalFoam
mdkdir -p transportThermalModels
cd transportThermalModels
cp -r $FOAM_SRC/transportModels/incompressible .
```

- Edit the file Make/files

```
LIB = $(FOAM_USER_LIBBIN)/libincompressibleTransportThermalModels
```

twoPhaseMixture.H file

- Add data members for specific heats and conductivities

```
dimensionedScalar Cp1_;
dimensionedScalar Cp2_;
dimensionedScalar k1_;
dimensionedScalar k2_;
```

- Add member functions to provide access (similar for all four members)

```
const dimensionedScalar& Cp1() const
{
    return Cp1_;
}
```

- Add declarations of member functions for the mixture conductivity

```
tmp<volScalarField> k() const;
tmp<surfaceScalarField> kf() const;
```

twoPhaseMixture.C file

- Add entries to the initialization list in the constructor

```
Cp1_(nuModel1_->viscosityProperties().lookup("Cp")),
Cp2_(nuModel1_->viscosityProperties().lookup("Cp")),
k1_(nuModel1_->viscosityProperties().lookup("k")),
k2_(nuModel1_->viscosityProperties().lookup("k")),
```

- Add entries to the function read()

```
nuModel1_->viscosityProperties().lookup("Cp") >> Cp1_;
nuModel2_->viscosityProperties().lookup("Cp") >> Cp2_;
nuModel1_->viscosityProperties().lookup("k") >> k1_;
nuModel2_->viscosityProperties().lookup("k") >> k2_;
```

twoPhaseMixture.C file

- Add definitions of member functions for the conductivity
  - at cell centres and similar at cell faces

```
tmp<volScalarField> twoPhaseMixture::k() const
{
    volScalarField limitedAlpha1 = min(max(alpha1_, scalar(0)), scalar(1));
    return tmp<volScalarField>
    (
        new volScalarField
        (
            "k_twoPhaseMixture",
            limitedAlpha1*k1_ + (scalar(1) - limitedAlpha1)*k2_
        )
    );
}
```

- Compile the new library

```
wclean
wmake libso
```

Creating the new interThermalFoam solver

- Rename the main file

```
cd $WM_PROJECT_USER_DIR/applications/solvers
cd interThermalFoam
mv interFoam.C interThermalFoam.C
```

- Edit the file Make/files

```
interThermalFoam.C
EXE = $(FOAM_USER_APPBIN)/interThermalFoam
```

- Edit the file Make/options

```
EXE_INC = \
    -ItransportThermalModels \
    -ItransportThermalModels/incompressible/lnInclude \
    ...
EXE_LIBS = \
    -L$(FOAM_USER_LIBBIN) \
    -lincompressibleTransportThermalModels \
    ...
```

createFields.H

- Add the object for the temperature field

```
Info<< "Reading field T\n" << endl;
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

- Add references to specific heats

```
const dimensionedScalar& Cp1 = twoPhaseProperties.Cp1();
const dimensionedScalar& Cp2 = twoPhaseProperties.Cp2();
```

createFields.H

- Add the objects for mass and energy fluxes

```
volScalarField rhoCp
(
    IOobject
    (
        "rhoCp",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT
    ),
    alpha1*rho1*Cp1 + (scalar(1) - alpha1)*rho2*Cp2
);
rhoCp.oldTime();
surfaceScalarField rhoCpPhi
(
    IOobject
    (
        "rhoCp*phi",
        ...
    ),
    fvc::interpolate(rhoCp)*phi
);
```

alphaEqn.H and alphaEqnSubCycle.H

- Calculate the energy content and fluxes
  - in alphaEqn.H

```
rhoCpPhi = phiAlpha*(rho1*Cp1 - rho2*Cp2) + phi*rho2*Cp2;
```

  - in alphaEqnSubcycle.H

```
surfaceScalarField rhoCpPhiSum = 0.0*rhoCpPhi;

rhoCpPhiSum += (runTime.deltaT()/totalDeltaT)*rhoCpPhi;

rhoCpPhi = rhoCpPhiSum;

rhoCp == alpha1*rho1*Cp1 + (scalar(1) - alpha1)*rho2*Cp2;
```

TEqn.H and interThermalFoam.C

- Add the temperature equation in a new file TEqn.H

```
volScalarField k("k", twoPhaseProperties.k());
fvScalarMatrix TEqn
(
    fvm::ddt(rhoCp,T)
  + fvm::div(rhoCpPhi,T)
  - fvm::laplacian(k,T)
);
TEqn.solve();
```

- Include the new TEqn.H file in interThermalFoam.C file after the PISO loop

```
#include "TEqn.H"
```

- Compile the solver

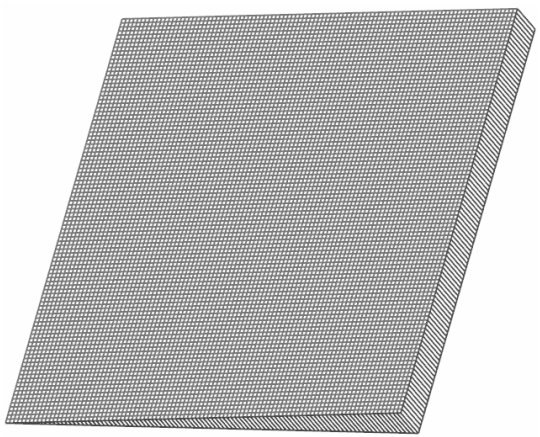
```
wclean
wmake
```

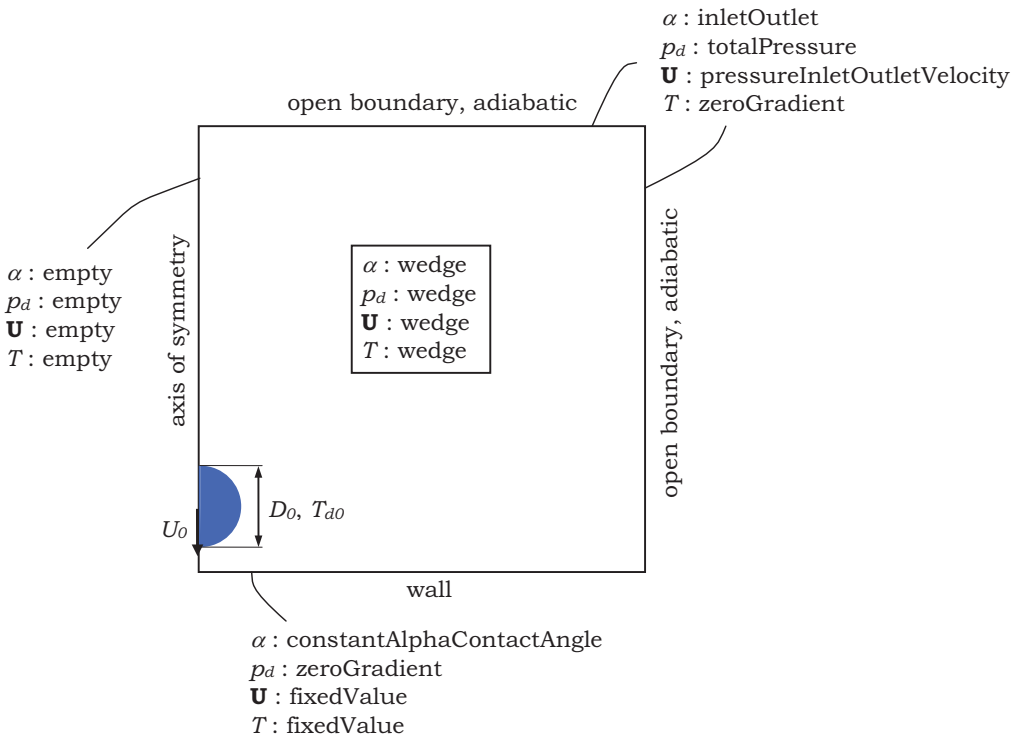
Preparing the case nonisothermalDropImpact

- Creating the mesh
  - `blockMeshDict` for 2D-axisymmetric case
- Initialization
  - create a file for the temperature field in the `0` directory
  - `setFieldsDict` for initialization
- Transport properties
  - Add thermal properties to the `transportProperties`
- Discretization schemes
  - Add the discretization schemes for new terms to the `fvSchemes`
- Iterative solvers
  - Add an entry for the solver for the temperature equation to the `fvsolution`
- Simulation parameters
  - change the solver name in the `controlDict`

nonisothermalDropImpact: axisymmetric case

- Case parameters:
  $D_0,\ U_0,\ T_{d0},\ \theta,\ \rho_l,\ \nu_l,\ c_{p,l},\ k_l,\ \rho_g,\ \nu_g,\ c_{p,g},\ k_g,\ \sigma$





Running the case and viewing results

- Case parameters

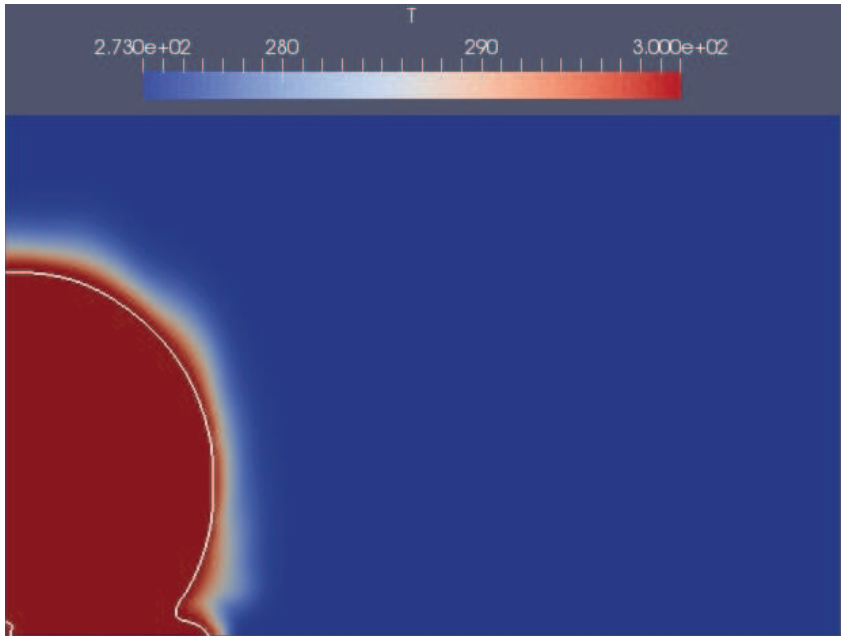
$$D_0 = 2\ \mathrm{mm}, U_0 = 1\ \mathrm{m/s}, T_0 = 300\ \mathrm{K}, \theta = 60^\circ$$
$$\rho_l = 1000\ \mathrm{kg/m^3}, \nu_l = 10^{-6}\ \mathrm{m^2/s}$$
$$c_{p,l} = 4190\ \mathrm{J/(kg\ K)}, k_l = 0.58\ \mathrm{W/(m\ K)}$$
$$\rho_g = 1.18\ \mathrm{kg/m^3}, \nu_g = 1.5 \cdot 10^{-5}\ \mathrm{m^2/s}$$
$$c_{p,g} = 1000\ \mathrm{J/(kg\ K)}, k_g = 0.025\ \mathrm{W/(m\ K)}$$
$$\sigma = 0.072\ \mathrm{N/m}$$

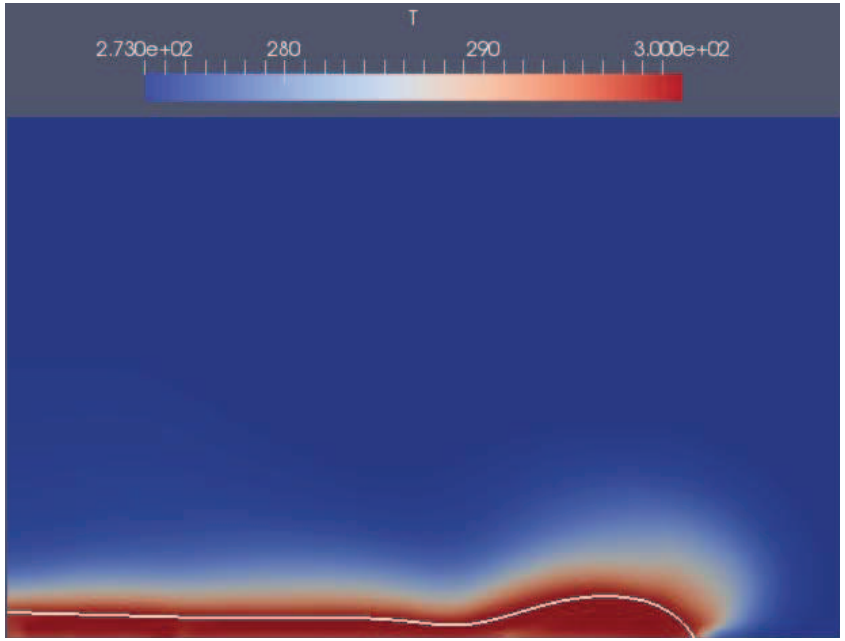
- Run the case from nonisothermalDropImpact directory

```
run
cd nonisothermalDropImpact
blockMesh
setFields
interThermalFoam
```

Running and viewing results

- View the results in ParaView
  - execute the script `paraFoam`
  - show the free-surface: create the iso-contour of $\alpha = 0.5$
  - visualize the field T





References

Extend-Project (2016)
The OpenFOAM®-Extend Project, *http://www.extend-project.de*.

OpenCFD Ltd, ESI Group (2016)
OpenFOAM®, the open source CFD toolbox, *http://www.openfoam.com*.