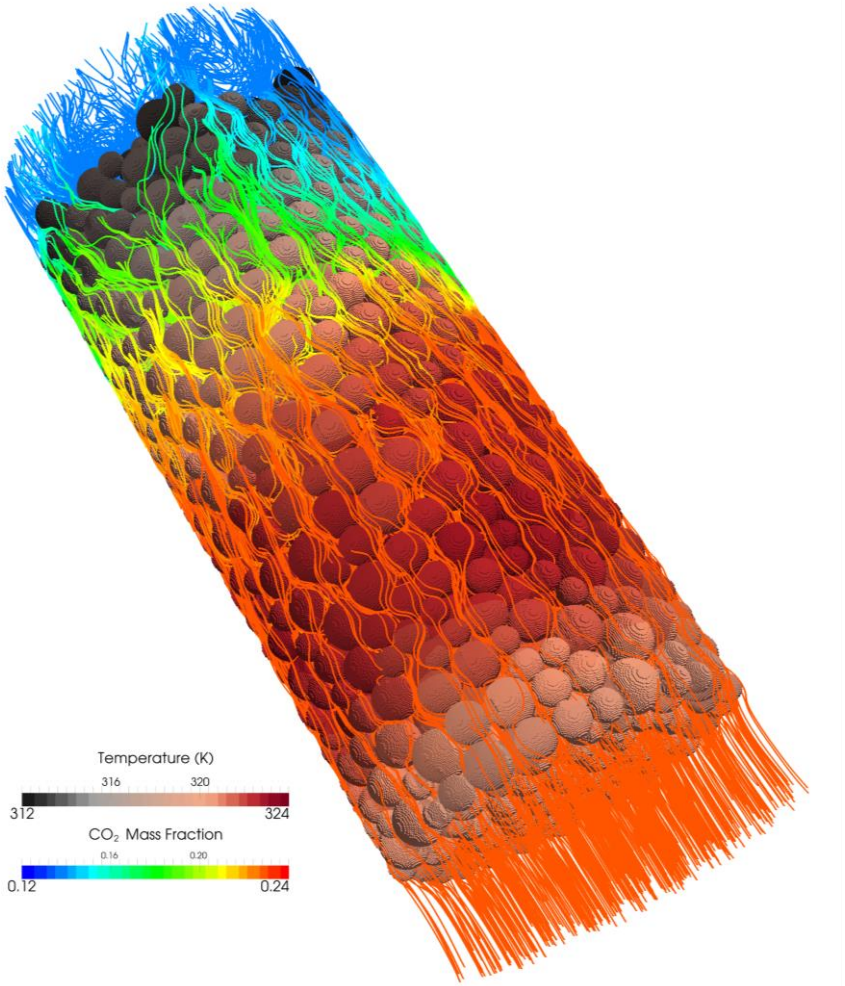
# OpenFOAM® Basic Training

# Tutorial Eight




3rd edition, Feb. 2015

www.thvt.at

Editors and Contributors:

- Bahram Haddadi (TU Wien)
- Christian Jordan (TU Wien)
- Jozsef Nagy (JKU Linz)
- Clemens Gößnitzer (TU Wien)
- Vikram Natarajan (TU Wien)
- Sylvia Zibuschka (TU Wien)
- Michael Harasek (TU Wien)

Cover picture from:

- Bahram Haddadi, The image presented on the cover page has been prepared using the Vienna Scientific Cluster (VSC).

# interFoam – damBreak (multiphase)

## Simulation

Use the interFoam solver to simulate breaking of a dam for 2s.

## Objectives

- Understanding how to set viscosity, surface tension and density for two phases

## Post processing

See the results in ParaView.

## Step by step simulation

### *Copy tutorial*

Copy tutorial from the following folder to your working directory:

```
~/OpenFOAM/OpenFOAM-
2.3.0/tutorials/multiphase/interFoam/laminar

/damBreak
```

### *0 directory*

In the 0 directory following files exist:

```
alpha.water.org  p_rgh  U
```

In the alpha.water.org and p_rgh files the initial values and also boundary conditions for phase water and also pressure are set. Copy alpha.water.org to alpha.water (remember: the *.org files are back up files, and solvers do not use them). E.g. alpha.water:

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

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    leftWall
    {
        type            zeroGradient;
    }

    rightWall
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }

    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

*Note: The* `inletOutlet` *and the* `outletInlet` *boundary conditions are used when the flow direction is not known. In fact, these are derived types and are a combination of two different boundary types.*

- `inletOutlet`: *When the flux direction is toward the outside of the domain, it works like a zeroGradient boundary condition and when the flux is toward inside the domain it is like a fixedValue boundary condition.*

- `outletInlet`: *This is the other way around, if the flux direction is toward outside the domain, it works like a fixedValue boundary condition and when the flux is toward inside the domain, it is like a zeroGradient boundary condition.*

*E.g. if the velocity field outlet is set as* `inletOutlet` *and the* `inletValue` *is set to* `(0 0 0)`*, it avoids backflow at the outlet! The "*`inletValue`*" or "*`outletValue`*" are values for* `fixedValue` *type of these boundary conditions and "*`value`*" is a dummy entery for OpenFOAM® for finding the variable type. Using* `(0 0 0)`*, OpenFOAM® understands that the variable is a vector.*

***constant directory***

In the transportProperties file the properties of two phases can be set under each phase sub-dictionary, e.g. `water` or `air`:

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

phases (water air);

water
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}

air
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
```

```
    }
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0.07;

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

In both phases the coefficients for different models of viscosity are given, e.g. `nu`, `CrossPowerLawCoeffs` and `BirdCarreauCoeffs`.

Depending on which model is selected, the coefficients from the corresponding sub-dictionary are read. The selected model is `Newtonian`, only the `nu` coefficient is used and the others remain unused (`CrossPowerLawCoeffs` and `BirdCarreauCoeffs`).

`sigma` is the surface tension between two phases, in this example it is the surface tension between air and water.

Checking the g file, the gravitational field and also its direction are defined, it is `9.81` in the negative y direction.

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

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 -9.81 0 );

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

***Running simulation***

```
>blockMesh

>setFields

>interFoam
```

***Exporting simulation***

The simulation results are as follows (these are not the results for the original mesh, but a 2x refined finer mesh):

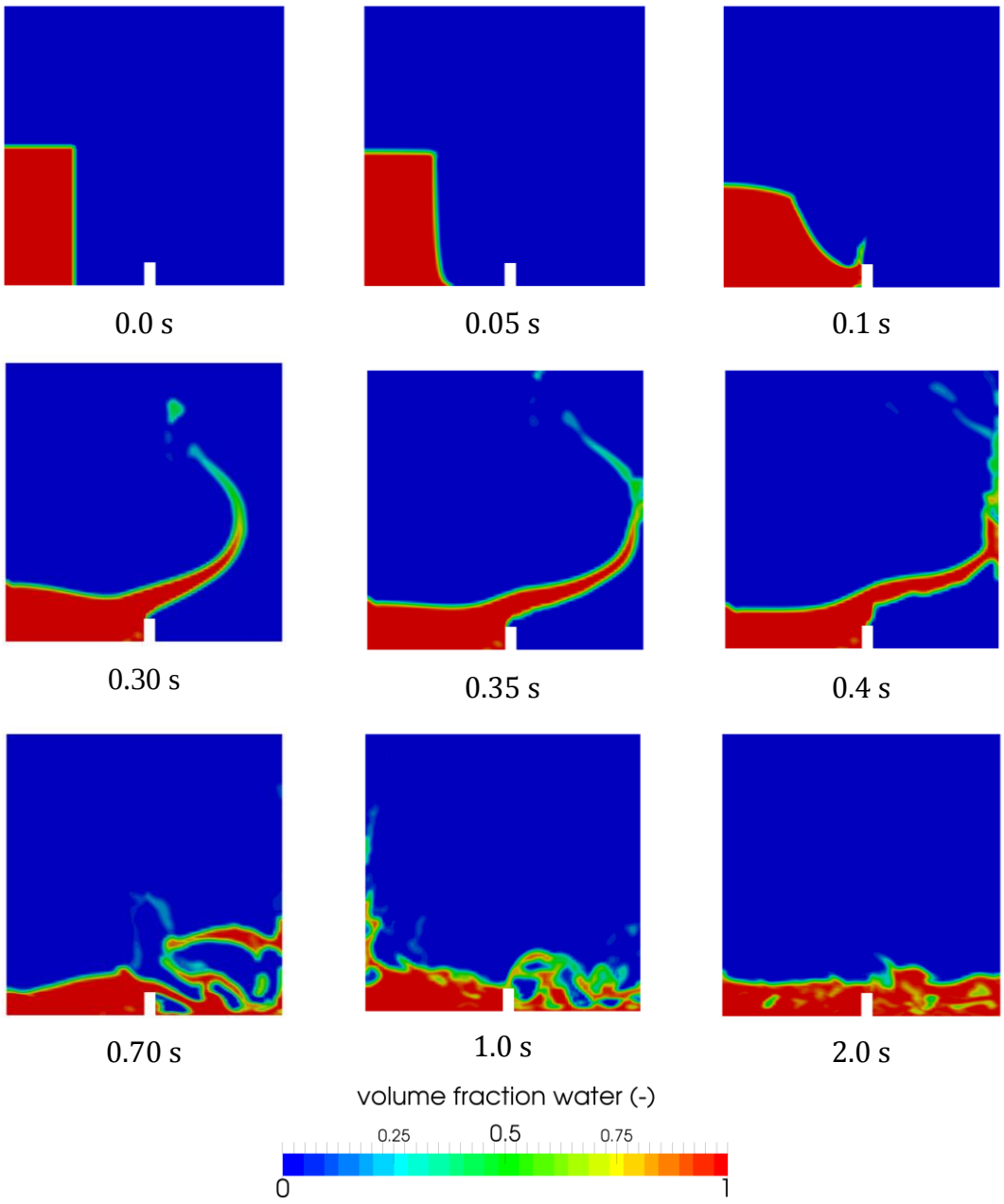



**Figure 8.1** Contours of the water volume fraction at different time steps