Content

- Geometry and mesh generation.
- Initial and boundary conditions setup
- Implementation of power law velocity inlet boundary condition
- Model setup (transport properties, turbulence, setFields)
- Post-processing

First steps

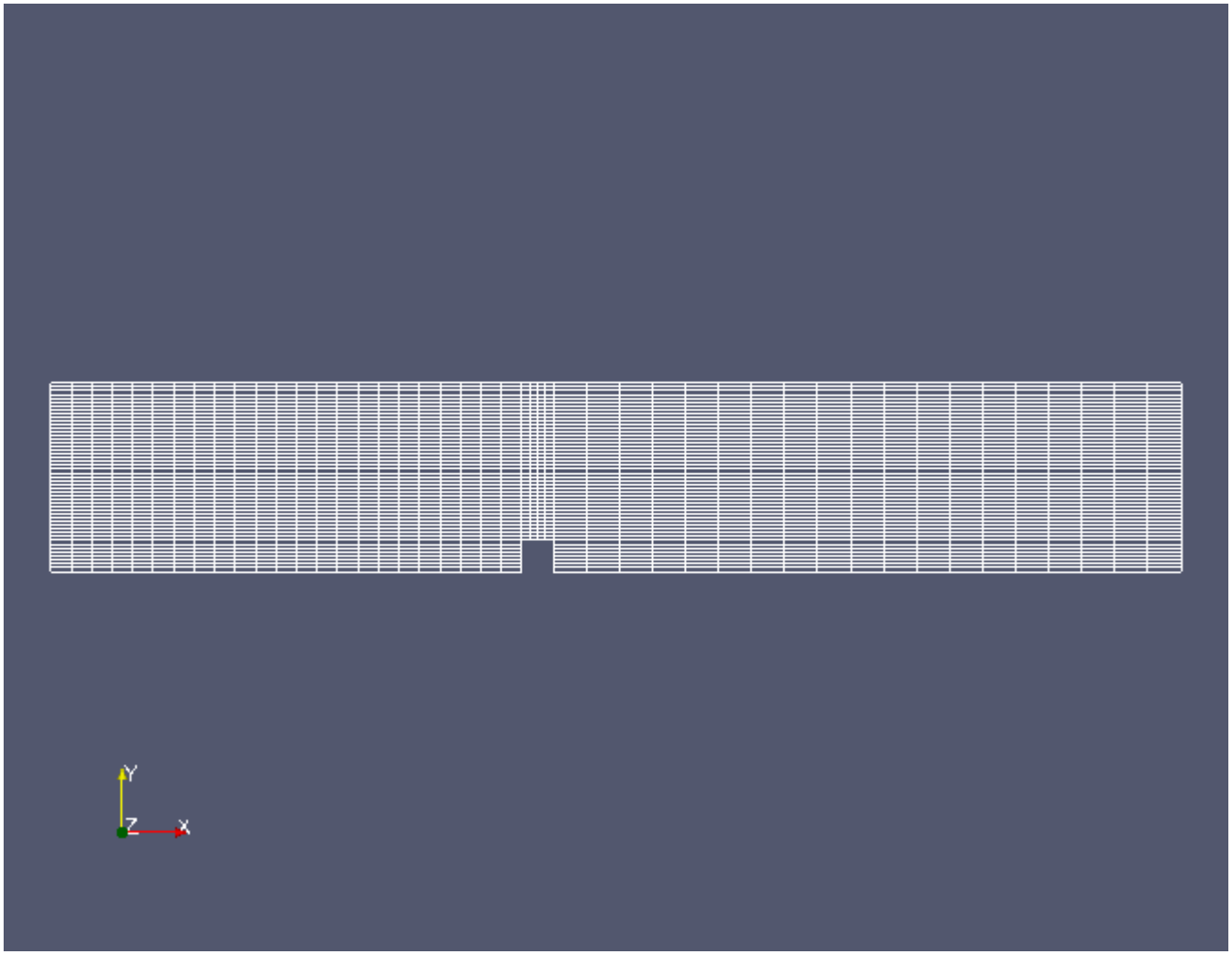
- As a templates, the damBreak related cases are used

```
run
cp -r $FOAM_TUTORIALS/multiphase/interDyMFoam/ras/damBreakWithObstacle .
mv damBreakWithObstacle snowDrift
cd snowDrift
```

- Substitute the geometry by the one from damBreak case

```
rm -rf constant/polyMesh
cp -r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak/ \
constant/polyMesh constant
```

Geometry and mesh



Geometry and mesh

- First change all the vertices in `blockMeshDict` to have desired dimensions

```
 convertToMeters 1;
vertices
(
    (0 0 0)
    (15 0 0)
    (16 0 0)
    (36 0 0)
    (0 1 0)
    (15 1 0)
    (16 1 0)
    (36 1 0)
    (0 6 0)
    (15 6 0)
    (16 6 0)
    (36 6 0)
)
```

Geometry and mesh

- Rename patches

```
patches
(
    patch inlet
    (
        (0 12 16 4)
        (4 16 20 8)
    )
    patch outlet
    (
        (7 19 15 3)
        (11 23 19 7)
    )
```

Geometry and mesh

- . . . and . . .

```
patches
(
    wall obstacle
    (
        (1 5 17 13)
        (5 6 18 17)
        (2 14 18 6)
        (0 1 13 12)
        (2 3 15 14)
    )
    wall sky
    (
        (8 20 21 9)
        (9 21 22 10)
        (10 22 23 11)
    )
```

Geometry and mesh

- . . . and finally . . .

```
empty frontAndBack
(
    (0 4 5 1)
    (2 6 7 3)
    (4 8 9 5)
    (5 9 10 6)
    (6 10 11 7)
    (12 13 17 16)
    (14 15 19 18)
    (16 17 21 20)
    (17 18 22 21)
    (18 19 23 22)
)
```

- **Now run** `blockMesh` **and** `checkMesh`

Boundary and initial conditions

- First copy the missing turbulence properties inside the `0` directory.

```
cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak/0/k 0/
cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak/0/epsilon 0/
```

Boundary and initial conditions

- Enter the `0` sub-directory and edit `U`, `p`, `k` and `epsilon`
- Mosty the patches have to be renamed according to their definition in `blockMeshDict`.
- Change the velocity inlet boundary so it looks as

```
    boundaryField
{
     inlet
     {
         type              powerLawVelocity;
         n                 (1 0 0);
         y                 (0 1 0);
         maxValue          10;
         value             uniform (0 0 0);
```

Boundary and initial conditions

- To have continuous volume fraction entering the domain, set the inlet in file `alpha1`

```
inlet
    {
        type            inletOutlet;
        inletValue      uniform 0.0001;
        value           uniform 0.0001;
    }
```

Power Law velocity profile

- To use velocity inlet defined in `U` as powerLaw, the new boundary condition has to be implemeted into a solver.

```
cp -r $FOAM_APP/solvers/multiphase/interFoam .
cp -r /chalmers/sw/unsup/OpenFOAM/OpenFOAM-1.5-dev/src/finiteVolume/ \
fields/fvPatchFields/derived/parabolicVelocity/* interFoam
mv interFoam snowInterFoam
cd snowInterFoam
wclean
```

Power Law velocity profile

- The file `files` in `Make` sub-directory has to be changed to contain

```
interFoam.C
powerLawVelocityFvPatchVectorField.C

EXE = $(FOAM_USER_APPBIN)/snowInterFoam
```

- The header of original solver file interFoam.C has to contain

```
#include "powerLawVelocityFvPatchVectorField.H"
```

Power Law velocity profile

- Rename everything called 'parabolic' to 'powerLaw'

```
sed -i s/parabolic/powerLaw/g parabolicVelocityFvPatchVectorField.H
sed -i s/parabolic/powerLaw/g parabolicVelocityFvPatchVectorField.C
mv parabolicVelocityFvPatchVectorField.C \
powerLawVelocityFvPatchVectorField.C
mv parabolicVelocityFvPatchVectorField.H \
powerLawVelocityFvPatchVectorField.H
```

Power Law velocity profile

- Take a look at Memeber Function in `powerLawVelocityFvPatchVectorField.C` and redefine it as follows

```
 void powerLawVelocityFvPatchVectorField::updateCoeffs()
{
 if (updated())
 {
     return;
 }
 // Get range and orientation
 boundBox bb(patch().patch().localPoints(), true);
 vector ctr = (bb.min()); //this lines defines minimum y value
 const vectorField& c = patch().Cf();
 // Calculate local 1-D coordinate for the powerLaw profile
 scalarField coord =((c - ctr) & y_)/((bb.max() - bb.min()) & y_);
 vectorField::operator=(n_*maxValue_*pow (coord/8,0.143));
}
```

- Compile the new solver `snowInterFoam`

```
wmake
```

Transport properties

- To get more buoyant fluid than in original damBreak case, the `transportProperties` have to be changed

```
 phase1
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 3.8e-10;
    rho             rho [ 1 -3 0 0 0 0 0 ] 250;
```

Activate turbulence model

- Change the file `turbulenceProperties`

```
simulationType  RASModel;
```

- Now, there is a need to specify `RASProperties`

```
cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak/constant/ \
RASProperties constant
```

setFieldsDict

- As in damBreak case, specify the volume of fluid at the inlet in `setFieldsDict`

```
defaultFieldValues
(
    volScalarFieldValue alpha1 0
    volVectorFieldValue U ( 0 0 0 )
);
regions
(
    boxToCell
    {
        box ( 0 0 0 ) ( 1 6 1 );
        fieldValues
        (
            volScalarFieldValue alpha1 0.0001
        );
    }
 );
```

fvSchemes and fvSolution

- **Remove the files `fvSchemes` and `fvSolution` in `system` and substitute them by the ones from damBreak tutorial case**

```
rm -rf system/fv*
cp -r $FOAM_TUTORIALS/multiphase/interFoam/ras/damBreak/system/fv* system/
```

controlDict and run

- Change the application and endTime lines in `controlDict`

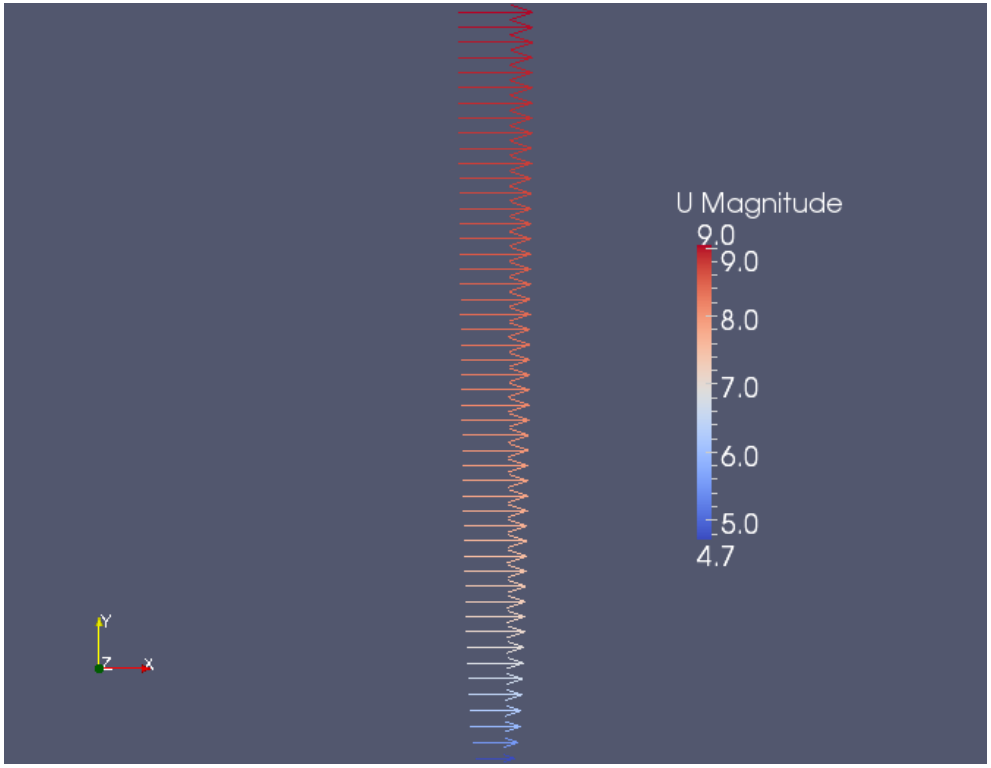
```
application     snowInterFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         60;

deltaT          0.001;
```

- Now, run setFields and the modified solver

```
setFields
snowInterFoam
```

Post-processing

- Check the inlet velocity profile using Cell Centers and Glyph filters



Post-processing

- Display `alpha1` values