Institute of Mathematical Sciences and Technology

Norwegian University of Life Sciences

CFD with OpenSource software, assignment 3

Tutorial Power law velocity inlet, InterFoam

Author:
Jan Potac

Peer reviewed by:

December 13, 2009

1 Introduction

The goal of this tutorial is to setup and run 2D simulation of incompressible two-phase turbulent flow with velocity inlet profile. It is focused on geometry creation, boundary condition definition and turbulence model activation. A transient solver for incompressible and multiphase flows using volume of fluid method, interFoam, is chosen. The case files are constructed using different damBreak tutorial files.
The model is characterized by a fluid mixture entering domain, passing an obstacle and leaving. The flow is governed by power law velocity inlet profile.

2 Geometry

The simple geometry contains a wind tunnel with a cubic obstacle placed on the bottom. The tunnel is 6 m high and 36 m long. The cube has dimensions of 1x1 m and is located 15 m downstream. See Fig. 1. The mixture of air and ,for instance, snow enters the domain on the left side and leaving on the right. All the domain surfaces used in boundary condition definition and setup are called as INLET, OUTLET, OBSTACLE, SKY and FRONTANDBACK.
First, the tutorial case related to interDyMFoam was copied for a use of a template.

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

Since the geometry in this damBreakWithObstacle case is different then in damBreak described in user guide, the directory polyMesh is deleted and files from damBreak case are uploaded instead.

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

Now the file blockMeshDict can be rearranged. The vertices and patches have to be changed as follows:

```
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)
    (0 0 1)
    (15 0 1)
    (16 0 1)
    (36 0 1)
    (0 1 1)
    (15 1 1)
    (16 1 1)
    (36 1 1)
    (0 6 1)
    (15 6 1)
    (16 6 1)
    (36 6 1)
);
blocks
(
    hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1)
    hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1)
    hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1)
);

edges
(
);

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

);

mergePatchPairs
(
);
```

When ready the command `blockMesh` might be run. The generated mesh and geometry can be seen at Figure 1.

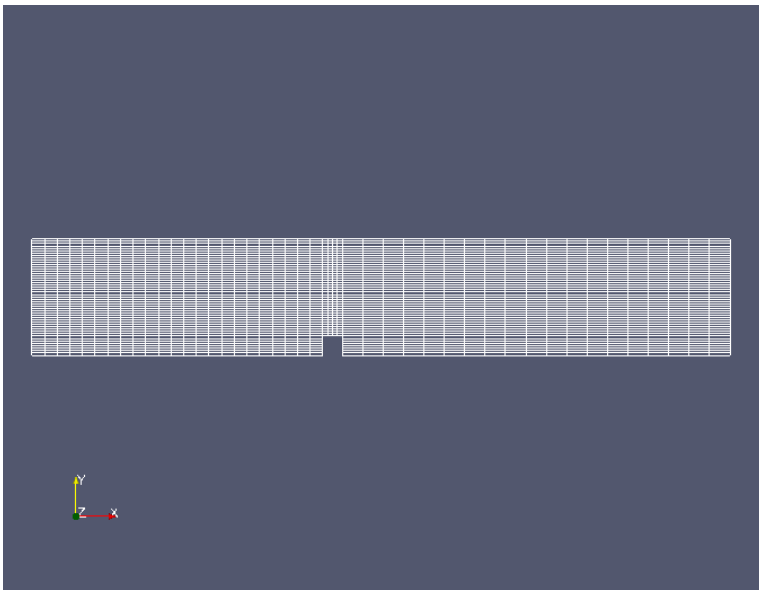

Figure 1: Geometry and mesh

3 Initial and boundary conditions

The next step covers setting up of initial and boundary conditions located in subdirectory `0`. The files `U`, `p`, `alpha1`, `k` and `epsilon` have to be redefined to fullfil boundary conditions definition names and also desired values. First copy the missing turbulence properties into `0` directory.

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

The file describing velocity properties at the boundaries looks as

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            powerLawVelocity;
        n               (1 0 0);
        y               (0 1 0);
        maxValue        10;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    obstacle
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    sky
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

As an inlet velocity covering the wind profile the powerLawVelocity boundary condition will be implemented. The more detailed about how to implement this is described in the text below.
The similar steps as for file `U` have to be done also for files `p`, `k` and `epsilon`.

```
 dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    obstacle
    {
        type            buoyantPressure;
        value           uniform 0;
    }

    sky
    {
        type            totalPressure;
        p0              uniform 0;
        U               U;
        phi             phi;
        rho             rho;
        psi             none;
        gamma           1;
        value           uniform 0;
    }

    frontAndBack
    {
        type            empty;
    }
}

 dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.1;

boundaryField
{
    inlet
```

```
    {
        type            fixedValue;
        value           uniform 1;
    }
    outlet
    {
        type            zeroGradient;
    }
    obstacle
    {
        type            kqRWallFunction;
        value           uniform 0.1;
    }
    sky
    {
        type            kqRWallFunction;
        value           uniform 0.1;
    }
    frontAndBack
    {
        type            empty;
    }
}

 dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 0.1;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.1;
    }
    outlet
    {
        type            zeroGradient;
    }
    obstacle
    {
        type            epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.1;
    }
    sky
```

```
    {
        type            epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.1;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

The last file in `0` directory is `alpha1`. Since the simulation is two phase, this file specifies volume fraction at the boundaries. The inlet condition should provide continuous volume fraction entering the domain.

```
 dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            inletOutlet;
        inletValue      uniform 0.0001;
        value           uniform 0.0001;
    }
    outlet
    {
        type            zeroGradient;
    }
    obstacle
    {
        type            zeroGradient;
    }
    sky
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

3.1 Power Law Velocity profile

Wind velocity inside Earth's boundary layer changes with increasing height. There exist a few wind profiles valid for certain conditions. This power law wind profile can be calculated using expression

$$u(y) = u_{ref} \left(\frac{y}{y_{ref}} \right)^{\alpha} \tag{1}$$

where u_{ref} and y_{ref} are reference values obtained from measurements. Exponent α describes stability of the atmosphere, and is approximately 0.143.
To implement powerLawVelocity boundary condition the parabolicVelocityFvPatchVectorField is used as a template.

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

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";
```

Before the new condition can be compiled within the new solver called snowInterFoam, everything called 'parabolic' should be replaced by 'powerLaw'. To do so the following commands can be applied

```
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
```

Let's take a look back at the power law function which has to be specified in powerLowVelocityFvPatchVectorField.C. The whole profile calculation is located in Member Function.

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

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));//power law equation

}
```

Now, the solver can by compiled by running `wmake`.

4 Transport properties, activation of turbulence model, fields setup, and run of the solver

In original damBreak case, the water is considered as a fluid. In this case, the density and viscosity is decreased to reach more buouyant fluid properties. This is done by changing the file `transportProperties`.

```
 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;
```

Before the final run of the solver will be done, there is a need to activate turbulence model and set volume fields.
First, the turbulence model must be activated in file called `turbulenceProperties` in `constant` sub-directory

```
 simulationType  RASModel;
```

Then the file `RASModel` has to be set up as well. In this case the file is copied from damBreak case.

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

Now, when turbulence equations are activated, there is a need to set volume fields using file `setFieldsDict` in `system` sub-directory. The box region with volume fraction of 0.0001 is defined in the domain inlet.

```
 efaultFieldValues
(
    volScalarFieldValue alpha1 0
    volVectorFieldValue U ( 0 0 0 )
);

regions
```

```
(
    boxToCell
    {
        box ( 0 0 0 ) ( 1 6 1 );
        fieldValues
        (
            volScalarFieldValue alpha1 0.0001
        );
    }
 );
```

Since the content of files `fvSolution` and `fvSchemes` does not fit with turbulence model equation setup, one can substitute these files by taking from from damBreak tutorial.

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

The last steps are to run command setFileds and snowInterFoam. Before that the `controlDict` file should be checked.

```
application     snowInterFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         60;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   2;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
```

```
adjustTimeStep  yes;

maxCo           0.1;

maxDeltaT       1;
```

Finally the case is ready and can be run using first `setFields` and then `snowInterFoam`.

5 Post-processing

First, let's check the proper velocity inlet profile. Running `paraFoam` and applying the `Cell Centers` filter and `Glyph` for inlet patch the velocity profile can be seen as in Figure 2.

The mass distribution inside the domain using plotting alpha1 can be seen at Figure 3

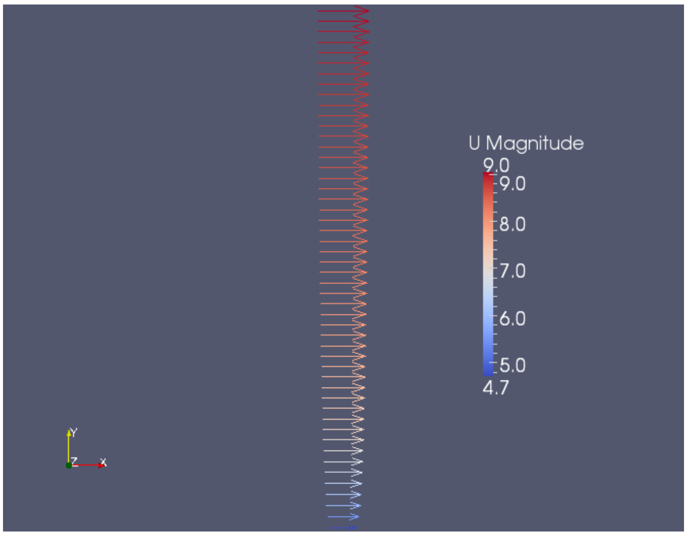

Figure 2: Power law velocity inlet profile

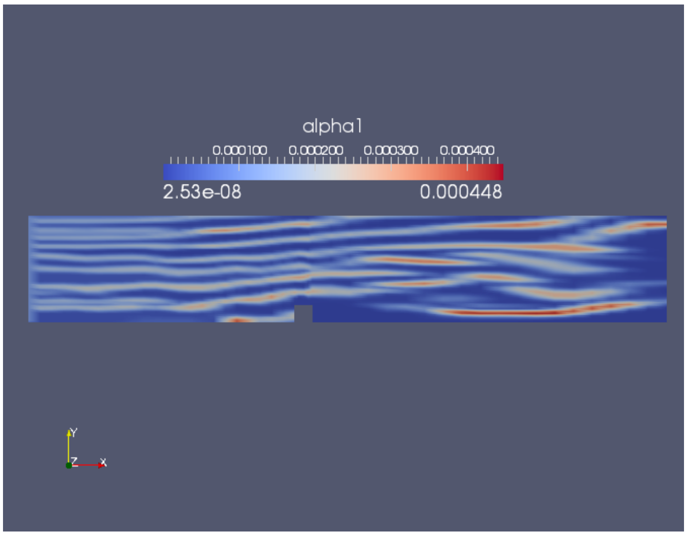

Figure 3: Power law velocity inlet profile