Chalmers
Spring 2009

Porous Media in OpenFOAM

Haukur Elvar Hafsteinsson

Introduction

This tutorial gives a detailed description of how to do simulations with porous media in `OpenFOAM-1.5`. The porous media class will be rewritten to cylindrical coordinates and applied for modeling the honeycomb of the `ercoftacConicaldiffuser` case study. It is showed how porous media can be used to alter the flow direction.

The Governing Equations

Porous media is modeled by attenuating the time derivative and by adding a sink term to the Navier-Stokes equations.

$$\frac{\partial}{\partial t}(\gamma\rho u_i) + u_j\frac{\partial}{\partial x_j}(\rho u_i) = -\frac{\partial p}{\partial x_i} + \mu\frac{\partial \tau_{ij}}{\partial x_j} + S_i \tag{1}$$

The value of γ should be between 0 and 1, where the latter is a complete porosity. The source term, S_i, is composed of two parts, a viscous loss term and an inertial loss term, creating a pressure drop that is proportional to the velocity and velocity squared, respectively.

$$S_i = -\left(\mu D_{ij} + \frac{1}{2}\rho\,|u_{kk}|F_{ij}\right)u_i \tag{2}$$

This equation is known as the Darcy-Forchheimer equation. In the case of simple homogeneous porous media it becomes

$$S_i = -\left(\mu D + \frac{1}{2}\rho\,|u_{jj}|\,F\right)u_i \tag{3}$$

where D_{ij} and F_{ij} are represented as the scalars D and F.
The source term can also be modeled as a power law of the velocity magnitude, i.e.

$$S_i = -\rho\,C_0\,|u_i|^{(C_1-1)/2} \tag{4}$$

where C_0 and C_1 are user defined empirical coefficients.

The Class

The porous media source files in `OpenFOAM-1.5` are located in the following directory:
`$FOAM_SRC/finiteVolume/cfdTools/general/porousMedia/`
The `porousMedia` folder contains the following files:

- `porousZones.H`
- `porousZones.C`
- `porousZone.H`
- `porousZone.C`
- `porousZonesTemplates.C`
- `porousZoneTemplates.C`

The implementation of the porosity equations are found in the file `porousZoneTemplates.C`. The member function used to modify the time-dependent term with the value, γ, is named `modifyDdt`. The viscous and inertial source terms from eqn. 2, are defined in the `addViscousInertialResistance` member function and the power law from eqn. 4, is defined in the `addPowerLawResistance` member function. Those two member functions used for calculating the source term are overloaded for explicit and implicit use. The first one that appears in the template is used for explicit calculations and the second one for implicit.

The Solver

In OpenFOAM-1.5, the use of the porousZones class is exemplified by the rhoPorousSimpleFoam solver, which is based on the rhoSimpleFoam solver as the name indicates. It is a steady-state solver for turbulent flow of compressible fluids with implicit or explicit porosity treatment. The implicit porosity solver is supposed to be more robust and is needed if the resistances are large, heavily anisotropic or not aligned with the global coordinates. The rhoPorousSimpleFoam solver is located in the following directory:
$FOAM_SOLVERS/compressible/rhoPorousSimpleFoam
There we find a Make folder and the following files:

- rhoPorousSimpleFoam.C
- initConvergenceCheck.H
- convergenceCheck.H
- createFields.H
- UEqn.H
- pEqn.H
- hEqn.H

In UEqn.H the momentum equation is constructed and the source term, S_i, is added by calling the member function addResistance, taking in different number of input variables depending on which solver is supposed to be used, explicit or implicit. The member function addResistance invokes the member functions addViscousInertialResistance and addPowerLawResistance from the file porousZoneTemplate.C
In createFields.H an object, named pZones, of the class porousZones is created. In the dictionary <case>/system/fvSolution it is checked if an implicit or an explicit solver should be used. If nUCorrectors = 0 an explicit solver is used, otherwise the solver is implicit.
If a transient solver is to be created with porous media, the modifyDdt member function from the file porousZoneTemplate.C should to be applied to the time-derivative term needed in the UEqn.H

The Case

There are two tutorials using rhoPorousSimpleFoam included in OpenFoam-1.5. They are located in the following directory:
$FOAM_TUTORIALS/rhoPorousSimpleFoam.
Both tutorials use the same geometry, see figure 1. It is a duct with a rectangular cross section with a sharp 45° bend at the center. The porous media is added where the duct is bending and it goes halfway up the angled duct. The cases are named angledDuctExplicit and angledDuctImplicit. The difference between them, as one would guess, is an explicit and an implicit porosity treatment. The directory structure for both cases is the same, see figure 2. The porosity is added to the geometry in the file constant/porousZones, which will be described later.

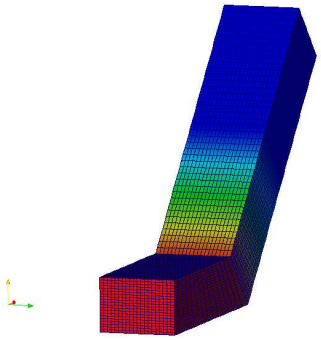

Figure 1: The `angledDuct` geometry.

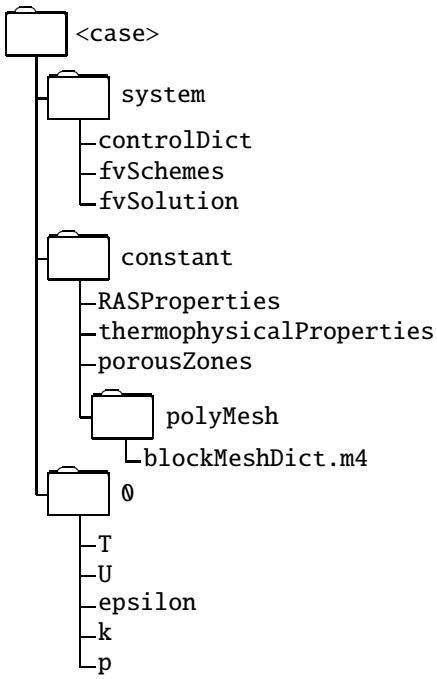

Figure 2: Directory structure for the `rhoPorousSimpleFoam` cases.

In order to choose whether an explicit or an implicit solver is to be used, the `system/fvSolution` file needs to be modified. For the implicit solver to be used the variable `nUcorrectors` inside `SIMPLE` needs to be a value larger than zero. If the explicit solver is to be used the variable `nUcorrectors` can be removed. Its default value is set to zero in `createFields.H` which is included in `rhoPorousSimpleFoam.C`. Therefore it is unnecessary to set it to zero.
Here is an example of how to "remove" `nUCorrectors` by standard C++ commenting.

```
SIMPLE
{
    // nUCorrectors                0;
    nNonOrthogonalCorrectors    0;
    pMin                        pMin   [1 -1 -2 0 0 0 0] 100;
}
```

For the explicit case the solver for the velocity U is needed to be defined. This is done inside the keyword `solver`:

```
solver
{
    U smoothSolver
    {
        smoother         GaussSeidel;
        nSweeps          2;
        tolerance        1e-6;
        relTol           0.1;
    };

    ...

}
```

while for the implicit case it is not needed to specify the solver for the velocity U. The under-relaxation factors are also different between the explicit and the implicit case, being higher for the implicit case. The under-relaxation factors can be higher for the implicit solver because it is more robust than the explicit one.

```
relaxationFactors //implicit
{
    p           0.3;
    rho         1;
    U           0.7;
    k           0.9;
    epsilon     0.9;
    h           0.9;
}

relaxationFactors //explicit
{
    p           0.3;
    rho         0.05;
    U           0.7;
    k           0.7;
    epsilon     0.7;
    h           0.5;
}
```

In `system/fvSchemes` the following line is added in the `laplacianSchemes` for the explicit case, otherwise the two files are identical.

```
laplacianSchemes
{
    ...

    // pCorrect
    laplacian((rho|A(U)),p) Gauss linear corrected;

    ...
}
```

In the file `constant/porousZones` the porous media is defined.

```
1
(
    porosity
    {
        coordinateSystem
        {
            e1  (0.70710678 0.70710678 0);
            e2  (0 0 1);
        }

        Darcy
        {
            d   d [0 -2 0 0 0 0 0] (5e7 -1000 -1000);
            f   f [0 -1 0 0 0 0 0] (0 0 0);
        }
    }
)
```

Porous media is added to a certain cell zone belonging to the mesh. Here the name of the cell zone is `porosity`, but it can refer to any cell zone defined in the `constant/polyMesh/blockMeshDict.m4` file, i.e. `inlet`, `porosity` or `outlet`.
Inside the cell zone a local coordinate system can be defined. The global coordinate system of the geometry is set by default. The coordinate system is defined with the `coordinateSystem` class. Which has several constructors. Here it is specified with two vectors. The vector `e1` is created as a linear combination of the global `x`- and `y`-axes so it is aligned with the angled duct and the vector `e2` is set orthogonal to `e1`. The vector `e3` is then created in a right handed order by the `coordinateSystem` class orthogonally to both `e1` and `e2`. More examples of how to create a local coordinate system is found in the description at the header of `coordinateSystem.H`, located in `$FOAM_SRC/meshTools/coordinateSystems`.
In `Darcy` the viscous and inertial resistance are defined, i.e. the source term from eqn. 2. The two dimensioned vectors, `d` and `f` are added to the diagonal of the tensors D_{ij} and F_{ij} respectively. The coordinates of these vectors corresponds to the local coordinate system. Default values of `d` and f is zero. At least one component of the vectors needs to be greater than zero. The code will automatically multiply negative values with the largest component of the vector and switch the sign to a positive value. This is done with the method `adjustNegativeResistance` which is implemented in `porousZone.C`.
The following code is an example of the entry in the `porousZones` dictionary, when using the `powerLaw` porosity model from eqn. 4. The default values for `C0` and `C1` are zero. The code also shows how to set `porosity`, the value γ from eqn. 1, its default value is 1. But the steady-state solver `rhoPorousSimpleFoam` will not use it since it belongs to the time-dependent term of the momentum equations.

```
   <cellZoneName>
    {
        powerLaw
        {
            C0  0.5;
            C1  2;
        }
        porosity 0.5;
    }
```

Many porous zones can be added to the case. The number of porous zones is limited to the number of cell zones in the mesh.

```
1
(
    <cellZoneName>
    {
        Defintions.
    }
)
2
(
    <anotherCellZoneName>
    {
        Defintions.
    }
)
```

The names of the cell zones is defined inside `blocks` in the file `constant/polymesh/blockMeshDict.m4`.
Here we have three cell zone names, i.e. inlet, porosity and outlet.

```
blocks
(
    // inlet block
    hex2D(in1, join1, join2, in2)
    inlet ( ninlet ncells ncells ) simpleGrading (1 1 1)

    // porosity block
    hex2D(join1, poro1, poro2, join2)
    porosity ( nporo ncells ncells ) simpleGrading (1 1 1)

    // outlet block
    hex2D(poro1, out1, out2, poro2)
    outlet ( noutlet ncells ncells )  simpleGrading (1 1 1)
);
```

In the `0` directory the boundary conditions for different field variables are defined as usual in `OpenFOAM`. At a porosity wall the velocity `U` has a `slip` boundary condition while at a regular wall a `no-slip` boundary condition is set, i.e. `U = 0 m/s`.

Porous media as a flow control

Porous media can be used to change flow direction by adding low porosity, i.e. high values for the source term, in a plane intersecting the flow direction with a certain angle. This can be thought as if a honeycomb, or some guide vanes, are set up in the flow field with its holes/vanes at some angle from the main flow direction. This procedure will here be illustrated step-by-step with the porous media class from `OpenFOAM-1.5`. The geometry from the `ercoftacConicaldiffuser-Case1` will be used. See a description at:

`http://openfoamwiki.net/index.php/Sig_Turbomachinery_/\`
`_ERCOFTAC_conical_diffuser#Case1:_Extended_base_case_set-up`

The flow direction will be changed at the inlet of the diffuser at cross section `CD`, by adding porous media with low porosity in a plane with 45° angle to the flow direction. It is recommended that you copy the case into your `run` directory:

```
run
svn checkout http://openfoam-extend.svn.sourceforge.net/\
svnroot/openfoam-extend/trunk/Breeder_1.5/OSIG/ \
TurboMachinery/ercoftacConicalDiffuser ercoftacConicalDiffuser
```

We will use the `rhoPorousSimpleFoam` solver and the `angledDuctImplicit` case as a base case. Copy it to your `run` directory and rename it as `conicalDiffuser`:

```
cp -r $FOAM_TUTORIALS/rhoPorousSimpleFoam/angledDuctImplicit/ .
mv angledDuctImplicit conicalDiffuser
```

Remove the old mesh information from `constant/polymesh` folder and copy the new mesh information from `ercoftacConicaldiffuser-Case1`:

```
cd conicalDiffuser/constant/polyMesh
rm *
cp -r $FOAM_RUN/ercoftacConicalDiffuser/\
cases/Case1/constant/polyMesh/blockMeshDict.m4 .
```

Generate the `blockMeshDict` by typing

```
m4 -P blockMeshDict.m4 > blockMeshDict
```

Take a look inside `blockMeshDict`. Open it with any text editor, i.e. `gedit, emacs, vim, e.t.c`.

```
vi blockMeshDict
```

There you will find inside `blocks` the definitions of the cell zones, `A-F`, for example the cross section `CD`:

```
blocks
(
...
    //Blocks between plane C and plane D:
    // block0 - positive x O-grid block
    hex (21 17 16 20 29 25 24 28 ) CD
    (15 20 6)
    simpleGrading (5 1 1)
    // block1 - positive y O-grid block
    hex (22 18 17 21 30 26 25 29 ) CD
    (15 20 6)
    simpleGrading (5 1 1)
    // block2 - negative x O-grid block
    hex (23 19 18 22 31 27 26 30 ) CD
    (15 20 6)
    simpleGrading (5 1 1)
    // block3 - negative y O-grid block
    hex (20 16 19 23 28 24 27 31 ) CD
    (15 20 6)
    simpleGrading (5 1 1)
    // block4 - central O-grid block
    hex (16 17 18 19 24 25 26 27 ) CD
    (20 20 6)
    simpleGrading (1 1 1)
...
)
```

Hint: Exit `vi` by entering `:q!` and hit the return key. The `porousZones` file needs to be modified to add the porous media at cross-section `CD`. Open `porousZones` and replace `"porosity"` with `"CD"`.

```
vi ../porousZones
```

The local coordinate system needs to be modified. The flow direction is along the axial direction of the diffuser, i.e. the global z-axis. Set the local x-axis as a linear combination of the global y- and z-axes to make the flow take a $\theta = 45°$ turn, i.e.

$$e1 = \cos(\theta)\, e_y + \sin(\theta)\, e_z$$

```
coordinateSystem
{
   e1  (0 0.70710678 0.70710678);
   e2  (1 0 0);
}
```

Also define the viscous loss `d` in the new local coordinate system. We will not define the inertial loss `f` this time:

```
Darcy
{
   d   d [0 -2 0 0 0 0 0] (5e7 1000 1000);
//   f   f [0 -1 0 0 0 0 0] (0 0 0);
}
```

The patch names in the files `T, U, epsilon, k` and `p` inside the `0/` folder need to match the patch names defined in `constant/polyMesh/blockMeshDict`. Therefore open `T, U, epsilon, k` and `p` and replace:

```
front => wallProlongation
back => wallDiffuser
wall => rotSwirlWall
porosityWall => statSwirlWall
```

This can be done by using the Linux `sed` command:

```
cd ../../0
sed -i s/porosityWall/statSwirlWall/ *
sed -i s/wall/rotSwirlWall/ *
sed -i s/back/wallDiffuser/ *
sed -i s/front/wallProlongation/ *
```

Now everything is set.
Run the case with:

```
run
cd conicalDiffuser
blockMesh
rhoPorousSimpleFoam > log &
```

This will take few minutes. The simulation has finished when there is a time directory named `100`. Postprocess the case with `ParaFoam`:

```
paraFoam &
```

Figure 3 shows how the porous media alters the flow direction as expected.

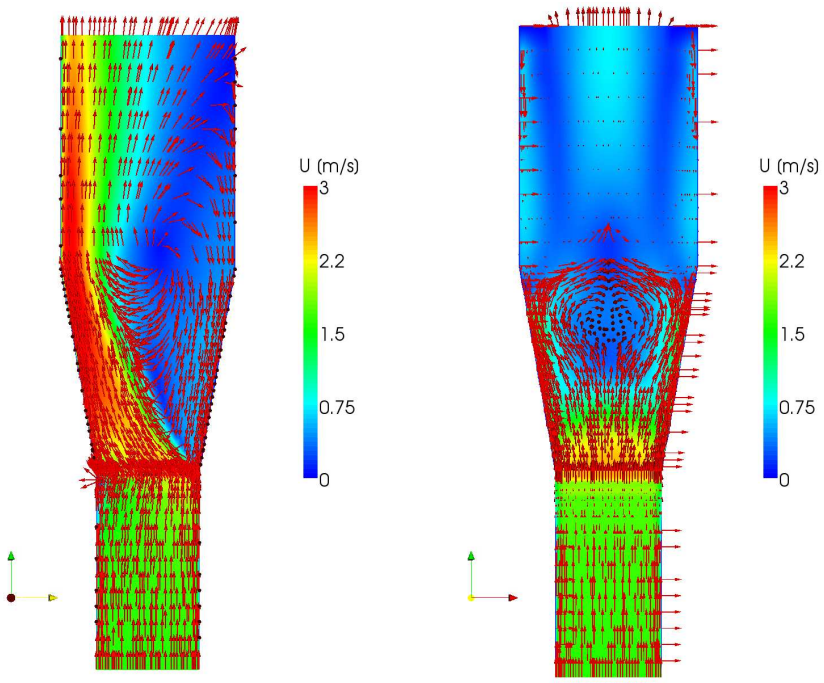

Figure 3: The flow turns because of high porosity in a plane with a 45° angle to the yz-plane.

Porous Media in Cylindrical Coordinates

The previous example showed how we can use the porous media class to add porosity to a geometry in order to change the flow direction. The porous media class is though limited to flow changes in Cartesian coordinates. If changes in tangential or radial direction are to be made using porosity then the porous media class needs to be rewritten. The following text describes one way to modify the porous media class in order to solve this problem. Figure 4 shows some definitions needed for the description of how to transform between the coordinate systems. If the flow is to start spinning, guide vanes are added that have a constant angle α along the radial axis e_r for all angles θ. If the flow is to start turning into the radial direction, guiding vanes are added that have constant turn β along the radial axis e_r for all angles θ. In order to do this a local coordinate system (e_1, e_2, e_3) needs to be set up along every radial axis. The coordinate system will vary with the polar coordinate θ. It is created by making polar-axes:

$$\begin{aligned} e_z &= e_z \\ e_r &= (x\,e_x + y\,e_y) / (x^2 + y^2)^{1/2} \\ e_\theta &= e_{axial} \times e_r \end{aligned}$$

and then building on top of them the axes:

$$\begin{aligned} e_1 &= \cos(\beta)\,e_r + \sin(\beta)\,e_{axial} \\ e_2 &= \cos(\alpha)\,e_\theta + \sin(\alpha)\,e_{axial} \\ e_3 &= e_1 \times e_2 \end{aligned}$$

Note that if α is to be used, then β should be zero and vice versa. The global coordinate system consists of the vectors (e_x, e_y, e_z), where e_z is in this case aligned along the axis of the conical diffuser. The angles α and β change the flow in tangential and radial direction respectively, see figure 4. A local-to-global transformation of the tensors D_{ij} and F_{ij} is needed in every point, i.e. every cell of the porous media.

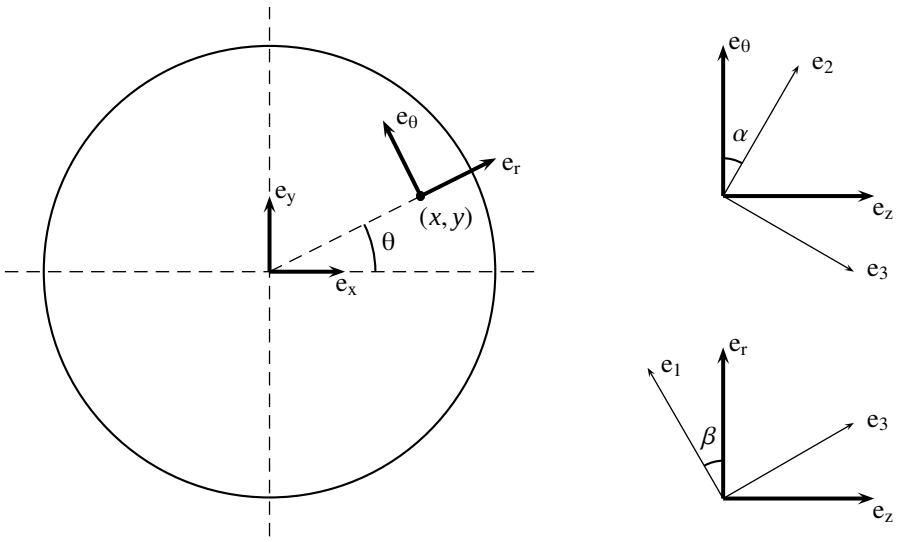

Figure 4: Schematic illustrating the relation between the local and global coordinate system.

These modifications have been made, and are now briefly described:
The classes in the `porousMedia` folder are modified. The name of the folder and the names of all the files as well have been replaced with the letters `cylP` instead of the first `p`, i.e. the folder `porousMedia` is now `cylPorousMedia`, e.t.c. The flow direction controlling variables, α and β are created as the member variables `rotationAngle_` and `radialAngle_` in the class `cylPorousZones.H`. The member function `addViscousInertialResistance` in `cylPorousZoneTemplates.C` is modified by removing the local-to-global transformation tensor `E` from `cylPorousZone.C` and adding it instead inside the `for-loop` where the source term is created.
The solver `rhoPorousSimpleFoam` is modified as well to use the new classes. The new solver name is `cylRhoPorousSimpleFoam` and it works in the same manner as before.

To use the new solver copy the files from the course home page:

```
wget http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/\
HaukurElvarHafsteinsson/haukur_case_class_solver.tgz
```

Extract it:

```
tar xzf haukur_case_class_solver.tgz
```

Enter the folder:

```
cd haukur_case_class_solver
```

Move the directories to keep similar directory structure as the original class:

```
mkdir -p $WM_PROJECT_USER_DIR/src/finiteVolume/cfdTools/general/
cp -r cylPorousMedia $WM_PROJECT_USER_DIR/src/finiteVolume/\
cfdTools/general/
```

Create the library:

```
cd $WM_PROJECT_USER_DIR/src/finiteVolume/cfdTools/general/cylPorousMedia
wmake libso
```

Move the directories to keep similar directory structure as the original solver:

```
mkdir -p $WM_PROJECT_USER_DIR/applications/solvers/compressible/
cp -r rhoCylPorousSimpleFoam \
$WM_PROJECT_USER_DIR/applications/solvers/compressible/
```

Compile the solver:

```
cd $WM_PROJECT_USER_DIR/applications/solvers/\
compressible/rhoCylPorousSimpleFoam
wmake
```

And `rehash` if necessary.

Now we will use the new solver on the case created before, i.e. the conicalDiffuser. The porosity will be added to cross-section CD as before. We will make the flow start rotating clockwise and then counterclockwise, by adding a porosity plane with $\alpha = 45°$ and $\alpha = 135°$ angle from the tangential direction. Then we will make the flow turn outward from the center axis and towards the center axis by adding a porosity plane with $\beta = 45°$ and $\beta = 135°$ angle from the radial direction.

```
run
mv conicalDiffuser cylConicalDiffuser
cd cylConicalDiffuser/constant
mv porousZones cylPorousZones
```

The `cylPorousZone` needs to be modified. Change the object from `porousZones` to `cylPorousZones`, define the member variables `rotationAngle` and `radialAngle` and take out the definition of the `coordinateSystem`. It should look like this:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      cylPorousZones;
}
1
(
    CD
    {
        radialAngle 0;
        rotationAngle 45;
        Darcy
        {
            d   d [0 -2 0 0 0 0 0] (1000 5e7 1000);
            //f   f [0 -1 0 0 0 0 0] (0 0 0)
        }
    }
)
```

Now run the case by typing

```
rhoCylPorousSimpleFoam > log &
```

and view the results with `paraFoam` as before. Then re-run the case three more times with the following settings:

```
radialAngle 0;
rotationAngle 135;

radialAngle 45;
rotationAngle 0;

radialAngle 135;
rotationAngle 0;
```

Figures 5-7 visualize the results from these different settings.

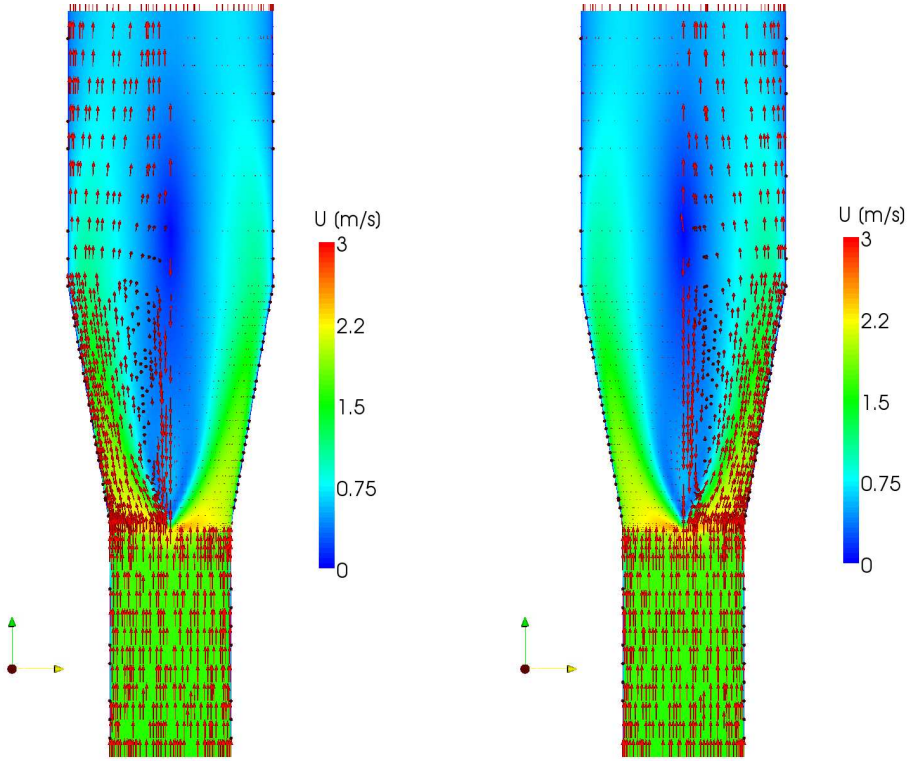

Figure 5: Porosity planes with angles $\alpha = 45°$ (left) and $\alpha = 135°$ (right) from the tangential direction.

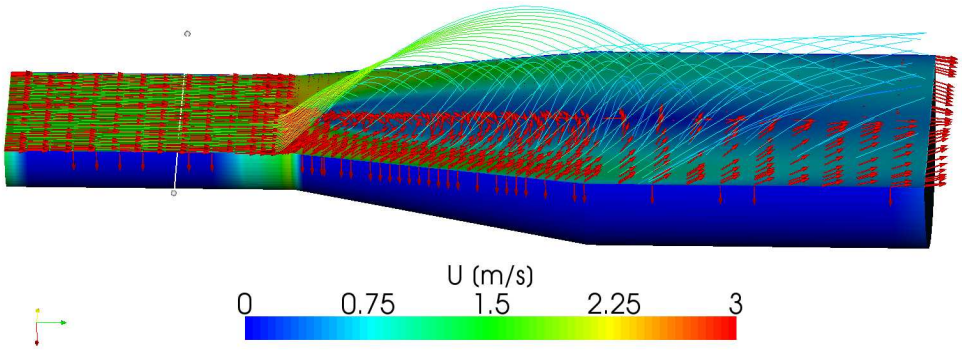

Figure 6: Streamlines showing the rotation of the flow caused by the porous media, $\alpha = 135°$.

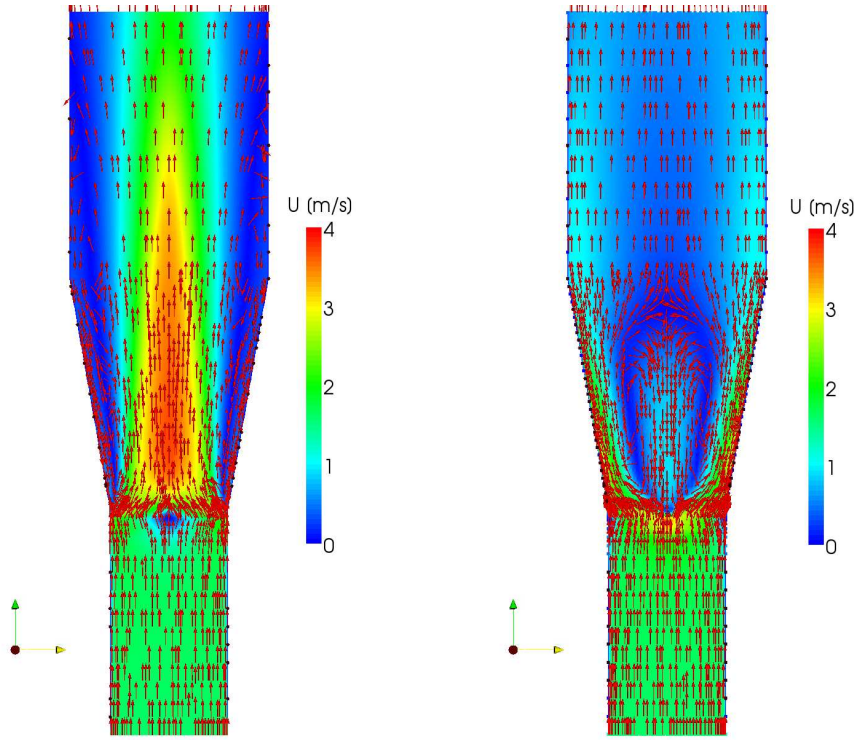

Figure 7: Porosity planes with angles $\beta = 45°$ (left) and $\beta = 135°$ (right) from the radial direction.