CFD with OpenSource Software

A course at Chalmers University of Technology
Taught by Hakan Nilsson

Project work:

Description of porousSimpleFoam and adding the Brinkmann model to the porous models

Developed for OpenFOAM-2.2.x

Author: Reza Gooya

Peer Reviewed by:

Olivier Petit

Chuck (Boxiong) Chen

February 01, 2014

Contents

1-Introduction

Simulation of fluid movement in porous material is of great importance and has many applications in different fields such as potroleum engineering, fuel cell and catalysis [1]. The most common way to model fluid transport in porous media is by using of Darcy's law. According to this law, the gradient of pressure is proportional to fluid velocity according to equation (1).

$$-\frac{\partial \mathrm{p}}{\partial \mathrm{x}} = \frac{\mu v}{\mathrm{k}} \qquad (1)$$

This law is based on the experimental observation of water flow in packed sands and many researchers have done mathematical derivations of it. After Darcy's law, many experiments show a different trend and so researchers such as Forchheimer and Brinkmann added another term to Darcy's law to make a correct model of fluid in porous media [2].

The purpose of this project is to clarify the porousSimpleFoam solver in openFoam. This solver located in the incompressible directory, includes some models for porous media and this report would try to dig deeper into this solver.

2-Theory

Simulation of porous media is performed by applying Velocity and Pressure equations with an additional source term being added to the momentum part as below[3]:

$$\frac{\partial}{\partial t}(\gamma \rho u_i) + u_j \frac{\partial}{\partial x_j}(\rho u_i) = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial \tau_{ij}}{\partial x_j} + S_i \qquad (2)$$

Forchheimer observed some discrepancies between his experimental results and Darcy's law so added a part to Darcy's law which is related to velocity square as follow:

$$-\frac{\partial \mathrm{p}}{\partial \mathrm{x}} = \frac{\mu \mathrm{v}}{\mathrm{k}} + \mu \beta \mathrm{v}^2 \qquad (3)$$

By considering macroscopic shearing effect between the fluid and the pore walls, Brinkmann added the second-order derivatives of the velocity to the Darcy equation resulting in:

$$-\frac{\partial \mathrm{p}}{\partial \mathrm{x}} = \frac{\mu \mathrm{v}}{\mathrm{k}} - \mu \nabla^2 \mathrm{v} \qquad (4)$$

3-Porous Media in OpenFOAM

Simulation of porous media in OpenFOAM can be categorized into 3 main parts. One part is tutorial which gives an example of porous media and will be explained in 3.1 section. Next part is porousSimpleFoam which calculate the pressure and velocity equations and another part is porous media models which calculate the source term for momentum equations and will be added to this in solver part.

3-1-Tutorial (or case study)

Tutorial of porous media is located in: `tutorials/incompressible/porousSimpleFoam` and tree chart of `angleDuctImplicit` directory is as follow:.

```
├── 0 -> ../angledDuctImplicit/0
│   ├── epsilon
│   ├── k
│   ├── nut
│   ├── p
│   ├── T
│   └── U
├── constant -> ../angledDuctImplicit/constant
│   ├── polyMesh
│   │   ├── blockMeshDict
│   ├── porosityProperties
│   ├── RASProperties
│   └── transportProperties
└── system
    ├── controlDict
    ├── fvSchemes
    └── fvSolution
```

Main difference between this tutorials and others is in `constant/porosityProperties`. In the DarcyForchheimer model we have D and F constant which is defined in the `porosityProperties` directory as follow:

```
porosity1
{
    type            DarcyForchheimer;
    active          yes;
    cellZone        porosity;

    DarcyForchheimerCoeffs
    {
        d   d [0 -2 0 0 0 0 0] (5e7 -1000 -1000);
        f   f [0 -1 0 0 0 0 0] (0 0 0);

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

To better explain the constant we simplify the equations as follow:

$$-\frac{\partial p}{\partial X} = \frac{\mu v}{k} - \beta \rho v2 \qquad (5)$$

the first term on the right-hand-side refers to friction and the second one refers to inertia.

this equation is something like this

$dP = (\mu * D * v + 0.5 * \varrho * F * v2)*L \qquad (6)$

if the Darcy-Forchhiemer equation is simplified as follow:

$dP = B*v + A*v2 \qquad (7)$

so from Eq. 5 and Eq. 7, $B = \frac{\mu}{K} v$ and $A = \frac{\beta}{\rho}$.

Where k is intrinsic permeability coefficient and β is representative of non-darcy behavior in DarcyForchheimer coefficients. From Eq. 6 and Eq. 7 we have:

$D = \frac{B}{\mu}$

$F = \frac{2A}{\rho}$

And also e1 and e2 are representative of porosity direction, for example when you have a horizontal cube, e1 and e2 are:

e1 (1 0 0)

e2 (0 1 0)

in this way the third direction is defined perpendicular to this coordination. So e1 and e2 could be defined as two of these three: (1 0 0), (0 1 0) and (0 0 1).

3-2-Porous Solver

porousFoam solver is located in `applications/solvers/incompressible/simpleFoam/porousSimpleFoam` and the whole directory is as follow:

```
├── createZones.H
├── Make
│   ├── files
│   └── options
├── pEqn.H
├── porousSimpleFoam.C
├── porousSimpleFoam.dep
└── UEqn.H
```

As it shows it includes main code which is `porousSimpleFoam.C` and in main part of this code it includes:

```
#include "UEqn.H"
#include "pEqn.H"
```

`Ueqn.H` is as follow:

```
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
  ==
    fvOptions(U)
);

mrfZones.addCoriolis(UEqn());

UEqn().relax();
```

Which in above part, momentum part is constructed and then momentum equations in implicit form or explicit form are solved.

```
    tmp<volScalarField> trAU;
    tmp<volTensorField> trTU;

    if (pressureImplicitPorosity)
    {
        tmp<volTensorField> tTU = tensor(I)*UEqn().A();
        pZones.addResistance(UEqn(), tTU());
        trTU = inv(tTU());
        trTU().rename("rAU");

        fvOptions.constrain(UEqn());

        volVectorField gradp(fvc::grad(p));

        for (int UCorr=0; UCorr<nUCorr; UCorr++)
        {
            U = trTU() & (UEqn().H() - gradp);
        }
        U.correctBoundaryConditions();

        fvOptions.correct(U);
    }
    else
    {
        pZones.addResistance(UEqn());

        fvOptions.constrain(UEqn());

        solve(UEqn() == -fvc::grad(p));

        fvOptions.correct(U);

        trAU = 1.0/UEqn().A();
        trAU().rename("rAU");
    }
```

The porous model is added in the `pZones.addResistance`(UEqn()) which pZones is the porosity zone, constructed in the blockMesh.

3-3- porous media model

The porous media models is located in `src/finiteVolume/cfdtools/general/porosityModels` and DarcyForchheimer model directory is as follow:

├── DarcyForchheimer.C

├── DarcyForchheimer.dep

├── DarcyForchheimer.H

└── DarcyForchheimerTemplates.C

The main definition of porous model is located in `DarcyForchheimerTemplate.H` as follow:

```
forAll(cellZoneIds_, zoneI)
    {
        const labelList& cells =
mesh_.cellZones()[cellZoneIds_[zoneI]];

        forAll(cells, i)
        {
            const label cellI = cells[i];

            const tensor Cd = mu[cellI]*D +
(rho[cellI]*mag(U[cellI]))*F;

            const scalar isoCd = tr(Cd);

            Udiag[cellI] += V[cellI]*isoCd;
            Usource[cellI] -= V[cellI]*((Cd - I*isoCd) & U[cellI]);
        }
    }
```

As it showed before, the D and F are constants from DarcyForchheimer equations which are read in `DarcyForchheimer.C` as:

```
adjustNegativeResistance(d);

    D_.value().xx() = d.value().x();
    D_.value().yy() = d.value().y();
    D_.value().zz() = d.value().z();
    D_.value() = (E & D_ & E.T()).value();
```

and:

```
    adjustNegativeResistance(f);

    // leading 0.5 is from 1/2*rho
    F_.value().xx() = 0.5*f.value().x();
    F_.value().yy() = 0.5*f.value().y();
    F_.value().zz() = 0.5*f.value().z();
    F_.value() = (E & F_ & E.T()).value();
```

D and F values considered as negative values and are added in three directions. If noticing equation 6, it's clear that where is 0.5 source in F value.

4-modifications

To implement the model of interest, at first we should modify the porosity models in `DarcyForchheimerTemplate.H` file and remove the F parameter from our code to be like follow:

```
const tensor Cd = mu[cellI]*D;
```

And also in continue:

```
forAll(cells, i)
        {
            const label cellI = cells[i];
            AU[cellI] += mu[cellI]*D;
        }
```

by adding the second part of brinkmann equation to the `UEqn.H`, source term is added to equations as follow:

```
    tmp<fvVectorMatrix> UEqn
    (
        fvm::div(phi, U)
      - fvm::laplacian(nu, U)
      + turbulence->divDevReff(U)
      ==
        fvOptions(U)
    );
```

Since v is introduced, we have to add it in `createField.H` as well.

Now we should add nu amount in the `constant/transportproperties` and remove F from `constant/porosityproperties` in our case.

To achieve the above mentioned modifications, the following command can be used:

```
Foam
cp –r –parents src/finiteVolume/cfdTools/general/porosityModel/DarcyForchheimer
$WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/
src/finiteVolume/cfdTools/general/porosityModel/DarcyForchheimer
mv DarcyForchheimer Brinkmann1
cd Brinkmann1
mv DarcyForchheimer.C Brinkmann1.C; mv DarcyForchheimer.H Brinkmann1.H; mv
DarcyForchheimerTemplates.C Brinkmann1.C
```

Then we shoud remove mentioned part (related to F value) in `Brinkmann1Templates.C`

And we should modify Make files/options in finiteVolume directory. Make/files to:

```
porosity = $(general)/porosityModel
$(porosity)/Brinkmann1/Brinkmann1.C
LIB = $(FOAM_USER_LIBBIN)/libmyfiniteVolume
```

And Make/options:

```
EXE_INC = \
-I$(LIB_SRC)/triSurface/lnInclude \
-I$(LIB_SRC)/meshTools/lnInclude \
-I$(LIB_SRC)/finiteVolume/lnInclude
LIB_LIBS = \
-lOpenFOAM \
-ltriSurface \
-lmeshTools
wclean
wmake libso
```

And for solver part:

```
foam
cp -r -parents applications/solvers/incompressible/simpleFoam/porousSimpleFoam
$WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/
applications/solvers/incompressible/simpleFoam/porousSimpleFoam
mv porousSimpleFoam brinkmannFoam
cd brinkmannFoam
mv porousSimpleFoam.C brinkmannFoam.C
```

Then the laplacian part should be added to UEqn.H

Modify Make directory to:

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

And Make option file should be like this:

```
EXE_INC = \
-I.. \
-I$(LIB_SRC)/turbulenceModels \
-I$(LIB_SRC)/turbulenceModels/incompressible/RAS/RASModel \
-I$(LIB_SRC)/transportModels \
-I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
-I$(LIB_SRC)/finiteVolume/lnInclude \
-I$(LIB_SRC)/meshTools/lnInclude \
-I$(LIB_SRC)/fvOptions/lnInclude \
-I$(LIB_SRC)/sampling/lnInclude
EXE_LIBS = \
```

```
-lincompressibleTurbulenceModel \
-lincompressibleRASModels \
-lincompressibleTransportModels \
-lfiniteVolume \
-lmeshTools \
-lfvOptions \
-lsampling
wclean
wmake
```

And for runnig the case:

```
run
cp -r ~/Downloads/Project.tgz .
tar xzf Project.tgz
cp -r Project/case .
cd case
blockMesh
brinkmannFoam
```

5-Results

We calculated velocity and pressure profile in DarcyForchheimer and brinkmann model. The post-processing of simulation has been done by paraview program. ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. To extract pressure and velocity field and also velocity vectors, the time is in the last time-step, and view direction has been set to –Z (figure 1). For pressure and velocity field, representation is by surface and the active variable shows the pressure and velocity magnitude (figure2). To show the velocity vectors, Glyph with the figure 3 properties has been created.

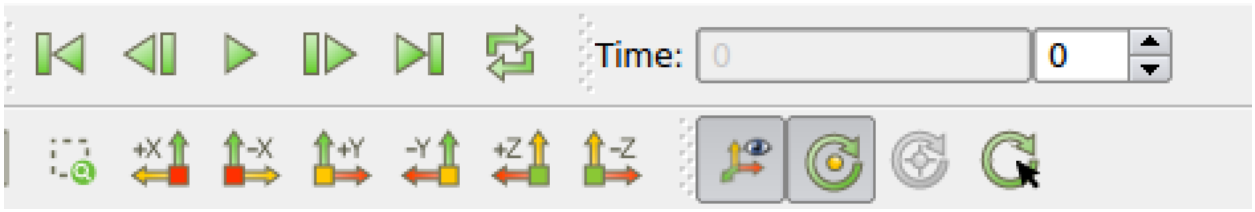

Figure 1. Time and Camera control part in paraview

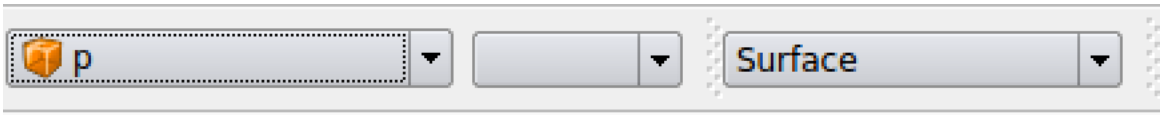

Figure 2. Active variable control and representation toolbar in paraview

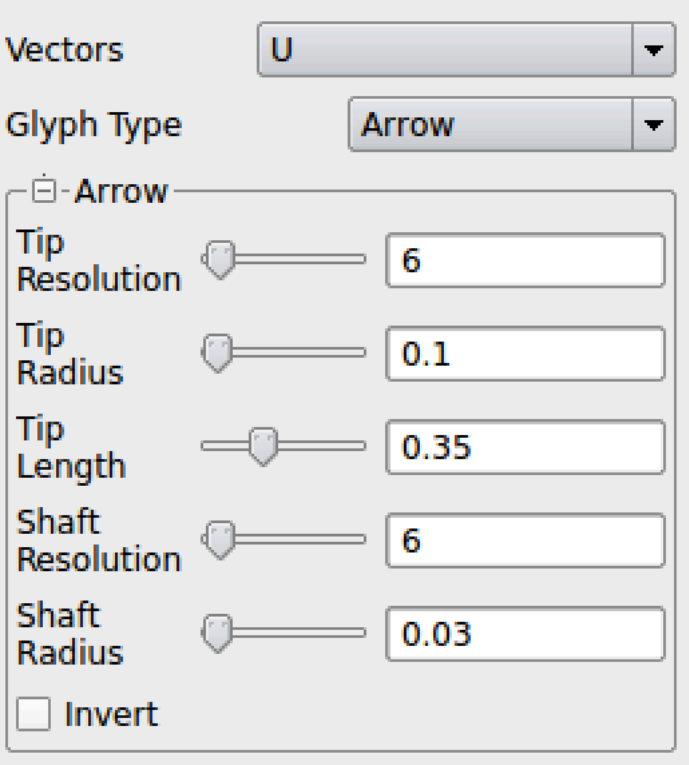

Figure 3. Object inspector for Glyph

The figures below are pressure, velocity and vector field of a cube with 65 μm size and for Brinkmann model. Table 1 shows the average velocity for considered models with 1 pascal pressure difference. As it shows the velocity results for Brinkmann model is less than that of Forchheimer since the velocity is low and second derivatives in brinkmann equation make more effect than velocity square part in DarcyForchheimer model. Generally `porosityProperties` file is important in porous modeling with OpenFOAM since in this file, all input properties for porous zone should be put.

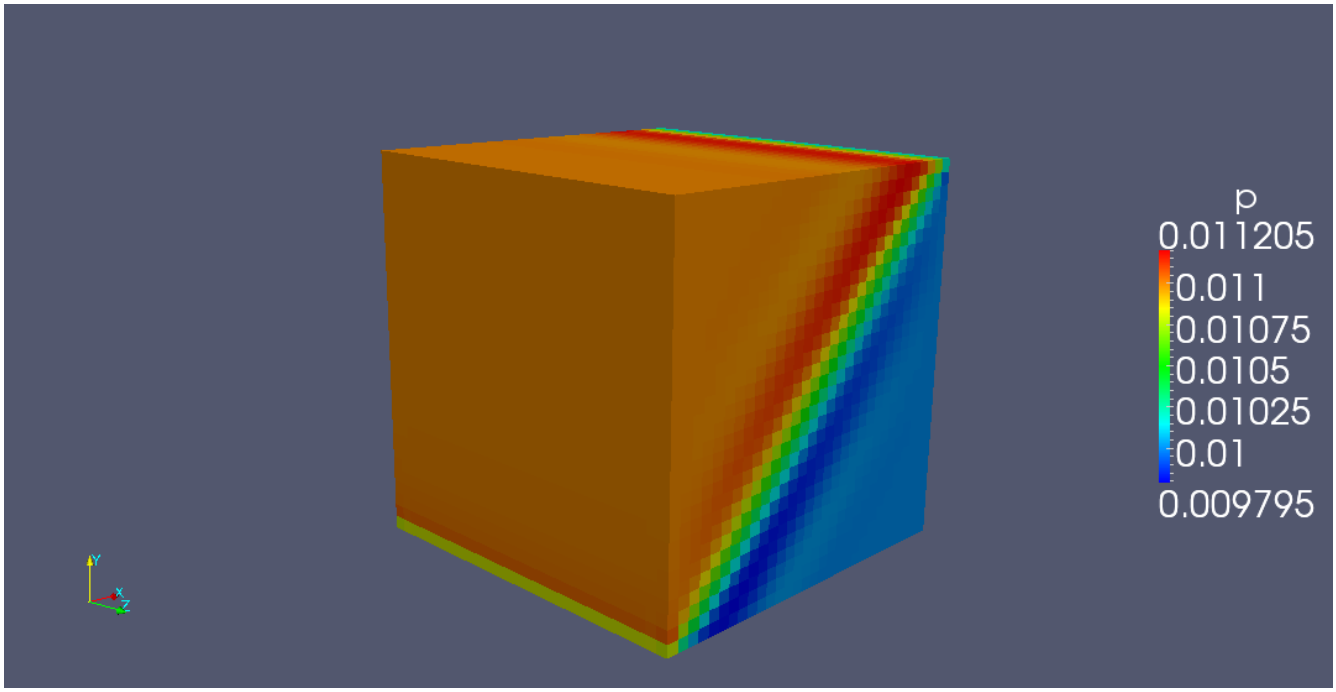

Figure 4. pressure field from Brinkmann model

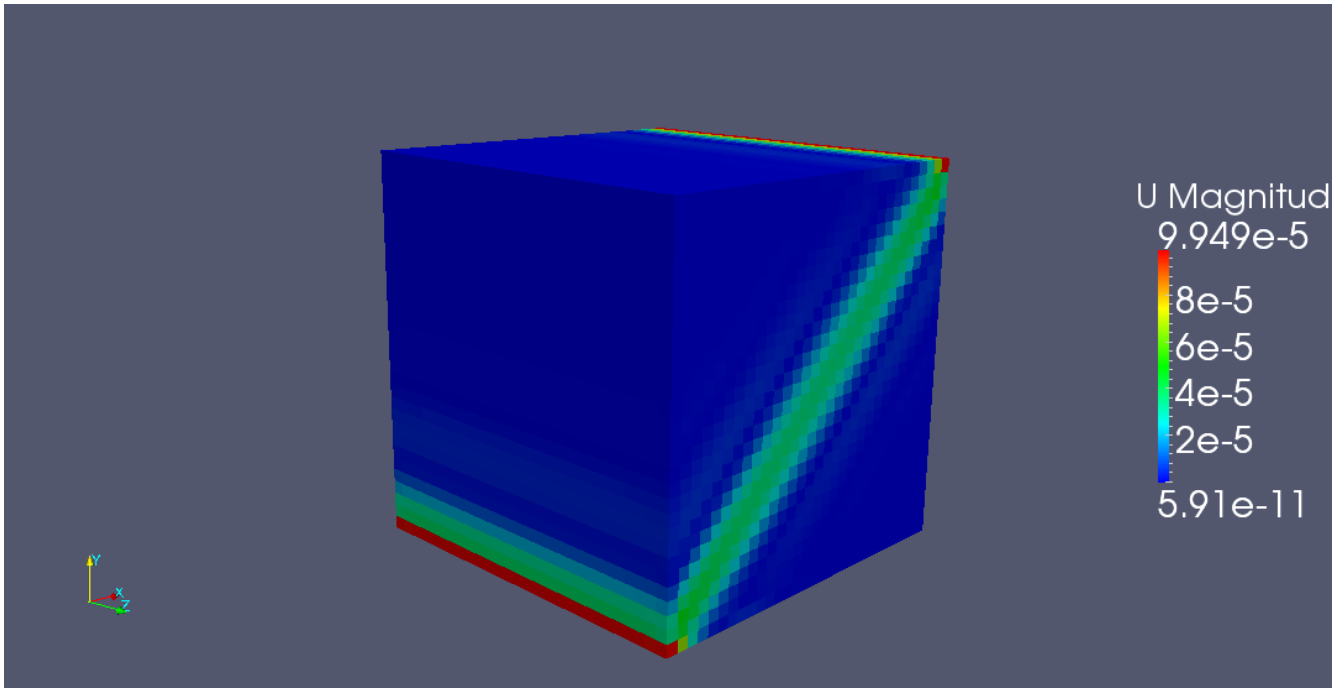

Figure 5. Velocity field from Brinkmann model

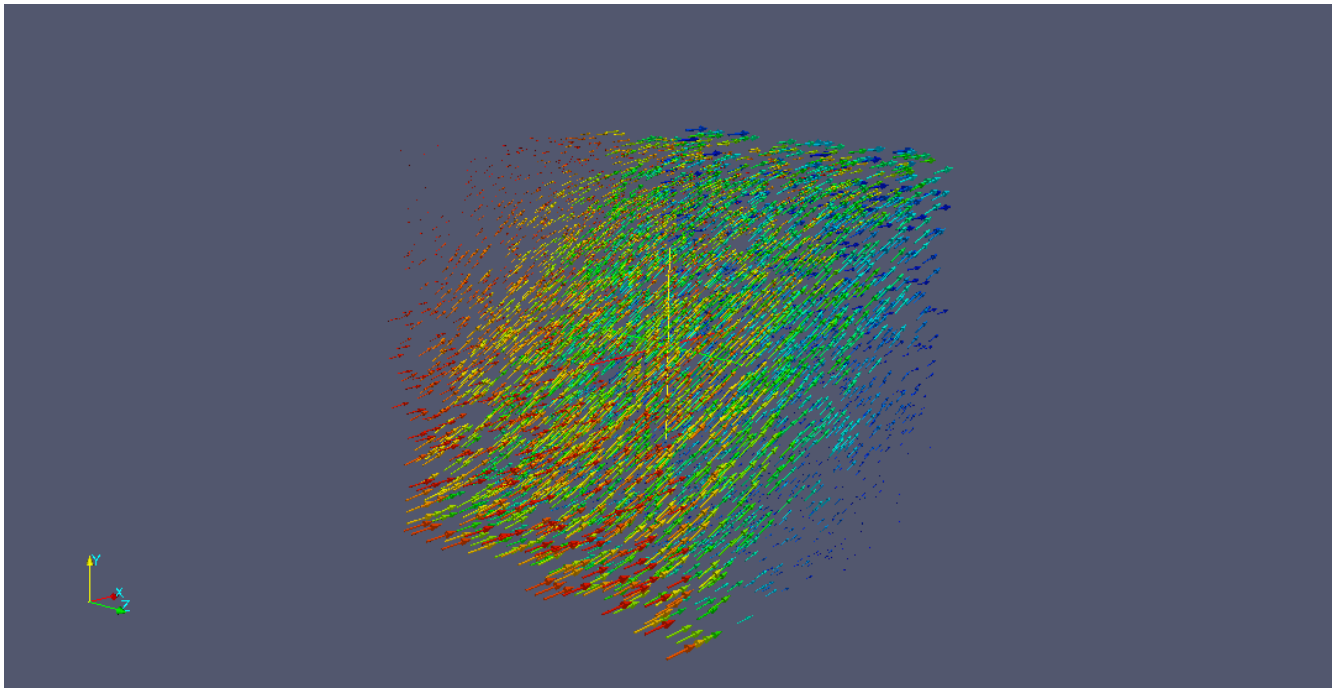

Figure 6. Vector field from Brinkmann model

Table 1. Comparison of Average velocity in DarcyForchheimer and Brinkmann model and direct simulation.

Model	Ux	Uy	Uz
DarcyForchheimer	9.8*10-6	9.8*10-6	-9.66*10-20
Brinkmann	5.02*10-6	5.02*10-6	-1.2*10-19

6. References

1. Chukwudozie, C.P., et al., *Prediction of Non-Darcy Coefficients for Inertial Flows Through the Castlegate Sandstone Using Image-Based Modeling.* Transport in Porous Media, 2012. **95**(3): p. 563-580.

2. Zeng, Z.W. and R. Grigg, *A criterion for non-Darcy flow in porous media.* Transport in Porous Media, 2006. **63**(1): p. 57-69.

3. Andersson Bengt and A. Rannie, *Computational Fluid dynamic for Engineers.* Cambridge Press, 2012.