# CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Hakan Nilsson

Author: Reza Gooya

Project work:

# Porous Media Modeling

Developed for OpenFOAM-2.2.x

# Introduction

fluid movement in porous material

Darcy's law

$$-\frac{\partial p}{\partial X} = \frac{\mu v}{k}$$

Forchheimer and Brinkmann

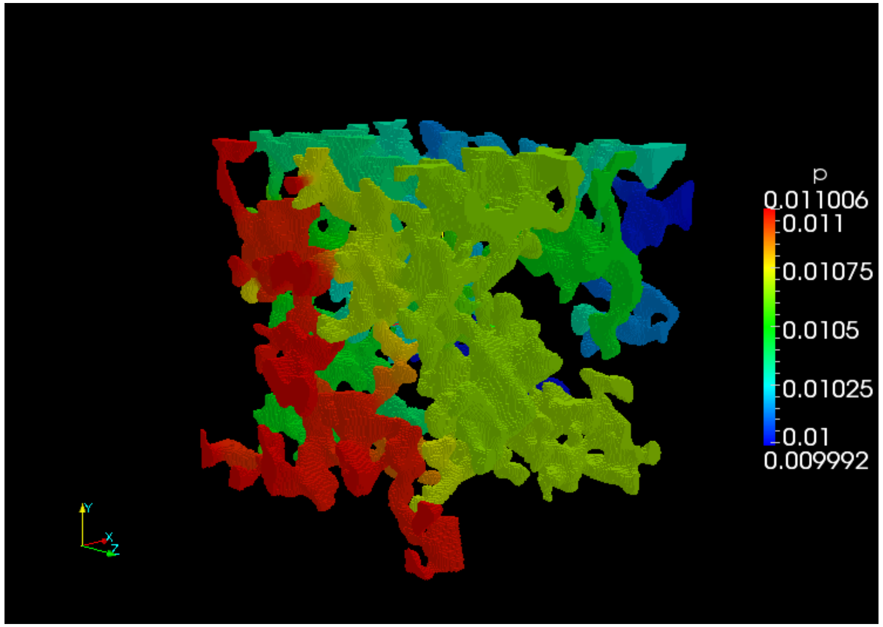
$$-\frac{\partial p}{\partial X} = \frac{\mu v}{k} + \mu \beta v^2$$

$$-\frac{\partial p}{\partial X} = \frac{\mu v}{k} - \mu \nabla^2 v$$

# Introduction

## Direct Modelling of Porous Media

- Monte Carlo
- Navier-Stokes
- Network Base
- Lattice Boltzmann



# OF-porous media-Tutorial

- Constant
- porosityproperties
- polymesh/blockmesh

```
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] (5e7 -1000 -1000);
      coordinateSystem
      {
         e1  (1 1 0);
         e2  (0 0 1);
      }
   }
}
```

```
Blocks
(
   hex (0 1 2 3 4 5 6 7) porosity (20 20 20)
simpleGrading (1 1 1)
)

boundaryCondition
(
    wall porosityWall
    faces
           (…
)
```

# OF-porous media-Solver

- Incompressible/porousSimpleFoam

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

```
pZones.addResistance(UEqn());
```

# OF-porous media-Porous model

- Darcy-Forchheimer

```
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);
```

$$\boxed{dP = (\mu * D * v + 0.5 * \rho * F * v^2)*L}$$

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

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

# OF-porous media-Porous model

## PowerLaw:

```
   const scalar C0 = C0_;
   const scalar C1m1b2 = (C1_ - 1.0)/2.0;
```

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

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

            Udiag[cellI] +=

V[cellI]*rho[cellI]*C0*pow(magSqr(U[cellI]),
C1m1b2);
        }
```

# Modifications

1- removing F parameter from Darcy-Forchheimer equation

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

  const tensor Cd = mu[cellI]*D;


           const scalar isoCd = tr(Cd);
```

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

# Modifications

2- Adding source term of brinkmann equation

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

```
brinkmannFoam.C

EXE = $(FOAM_USER_APPBIN)/
brinkmannFoam
```

# Modifications

Adding nu to createFields.H

```
 Info<< "Reading transportProperties\n" <<
endl;

    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
    );

    dimensionedScalar nu
    (
        transportProperties.lookup("nu")
    );
```

# Modifications

Adding nu to transportProperties

```
transportModel  Newtonian;

nu              nu [0 2 -1 0 0 0 0] 1e-06;
```

Adding library to controlDict

```
libs ("libmyfiniteVolume.so");
```

# Modifications

Change in porosityProperties

```
porosity1
{
   type            Brinkmann1;
   active          yes;
   cellZone        porosity;

   Brinkmann1Coeffs
   {
      d   d [0 -2 0 0 0 0 0] (2678000000
-1966000000 -2841000000);

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

# Implementation

```
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
```

# Implementation

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
```

*…. Continue next Page*

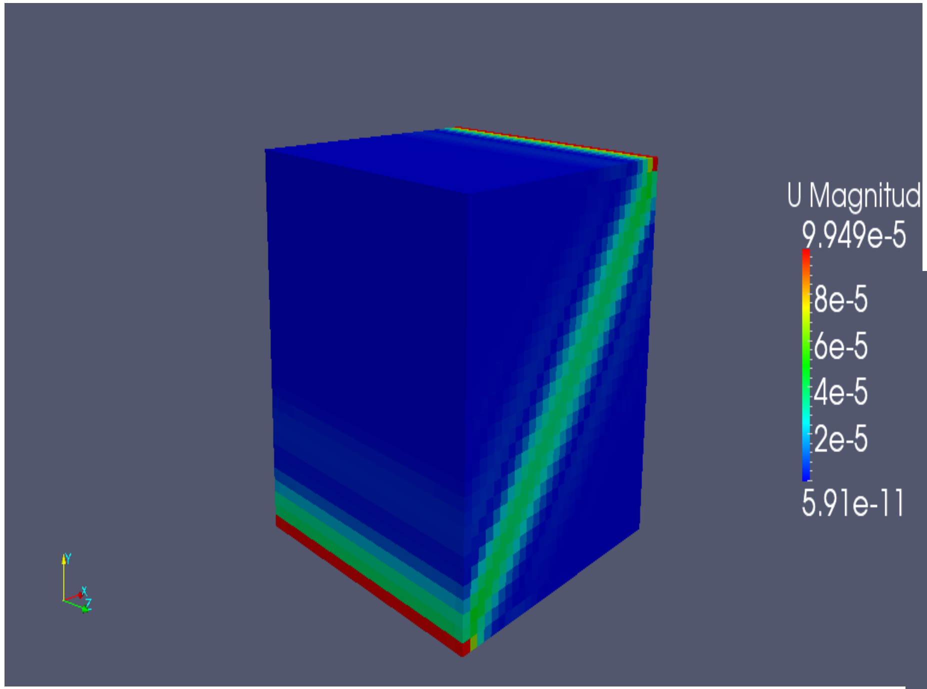
# Implementation

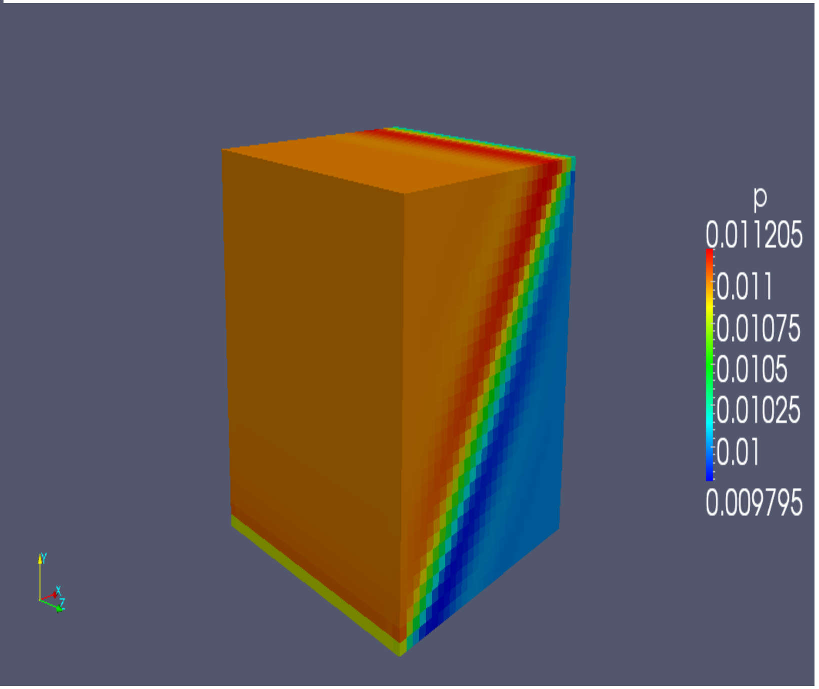
```
EXE_LIBS = \
    -lincompressibleTurbulenceModel \
    -lincompressibleRASModels \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools \
    -lfvOptions \
    -lsampling

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
```

# Results


U Magnitud
9.949e-5
8e-5
6e-5
4e-5
2e-5
5.91e-11
Y
X
Z


p
0.011205
0.011
0.01075
0.0105
0.01025
0.01
0.009795
Y
X
Z

# Results

| 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 |
| Calculated Values from Direct Simulation | 3.78*10-7 | 2.37*10-8 | 3.32*10-8 |