PoreFoam package for direct simulation of single-phase flow on 3D images of porous media.

Prepared by Ali Q Raeini
Department of Earth Science and Engineering, Imperial College London, UK, SW7 2AZ

Summary

This document presents the details of the poreFoam codes and scripts and the instructions for using them to run single-phase flow simulations on micro-CT images of porous media. The poreFoam package consists of pre-processing, processing and and post-processing applications, which includes standard C++ codes as well as applications written using OpenFOAM. To simplify and automate the preprocessing, processing and post-processing stages, Conventions on the file names has been set and bash scripts are developed for the simulation set up. This document presents a description of the keywords, file-name conventions and the scripts used to do direct single-phase flow simulations.

Contents

1 Installation

1.1 Prerequisites

You should have official OpenFOAM (version 1612+ in this document: https://www.openfoam.com/releases/openfoam-v1612+) installed in your Linux machine. Paraview should be installed for visualization, preferably through system provided packages.

1.2 Downloading and extracting files

Create a directory in your home (~) folder named "works" and extract/download the codes inside it in a folder named apps. You can choose any other folder and the scripts will work, but this ducument assumes you install the codes in "`~/works/apps`" folder.

To check that the directories are created correctly, type in a terminal :

```
# Important: replace ~/works/apps with the
# directory in which you have extracted the codes
ls  ~/works/apps/scripts ~/works/apps/poreFoam* ~/works/apps/voxelImage
```

and it should not show the error message "`No such file or directory`".

1.3 Compiling the codes

Edit the "`~/works/apps/poreFoam/bashrc`" file making sure it sources the OpenFOAM-1612+/etc/bashrc file based on your OpenFOAM installation directories.

The mesh generation code (rawToFoamPar) in requires a recent C++ compiler, you can set it either in the `voxelImage/Makefile` or by setting the variable `CXX` in file `~/works/apps/bashrc`. The default (g++) may work and hence you don't need to do any change.

Then open a terminal and type the following commands to compile the codes:

```
(cd ~/works/apps/poreFoam && ./AllMake)
```

1.4 Setting the Environmental variables:

Add the following line to your `~/.bashrc` file (optional but recomended),

```
source ~/works/apps/bashrc
```

This makes the poreFoam scripts available in any new terminal you open.

2 Usage

2.1 Input file format

The required input files are:

1. Input header file.
2. Micro-CT images should be given in ascii (suffix should be `.dat`) or binary files (better to have `.raw` suffix, the data should be in 8bit unsigned char). Compressed raw images are supported, for which the file suffix should be `.raw.gz`. You can use ImageJ, or the imageFileConvert from the voxelImage library to convert between image types.

The input header file for the flow simulation codes is a .mhd header file, which is a standard format compatible with Paraview and Fiji (ImageJ with plugins), with additional optional image manipulation commands specific to poreFoam package (and my other codes – image-processing, two-phase flow, network modelling codes etc.).

The main keywords of the .mhd header are as follows. (First to third keywords should not be changed):

1. `ObjectType = Image`
2. `NDims = 3`
3. `ElementType = MET_UCHAR`
4. keyword: `DimSize` – used to assign the dimensions of the image: Nx, Ny and Nz
5. keyword: `ElementSpacing` – used for assigning voxel size: δx, δy and δz should be equal
6. keyword: `ElementDataFile` – specifies the name of binary 8bit data file (.raw), ascii (.dat) files are supported too, but only by the poreFoam codes. For compressed images, the file suffix should be .raw.gz.

```
ObjectType = Image
NDims =    3
ElementType = MET_UCHAR

DimSize =    400   400   400
ElementSpacing = 5.345 5.345 5.345
Offset =     0     0     0

ElementDataFile = Berea.raw

####optional keywords, uncomment to activate:
#BinaryData = True
#threshold 0 0 # void space range in the image
#direction z   # turn image in y or z direction
#cropD    0 0 0 200 200 200 # crop the image
#resample 2 # coarsen (>1) or refine (<1), not tested
```

Figure 1: Sample input file "Berea.mhd"

Notes:

1. The order of the first 7 keywords should not be changed for compatibility with third-party software (ImageJ and Paraview). The rest of the keywords are optional.
2. "`#`" is used for comments
3. All keyword and its data should be given in a single line

For more advanced features you can edit the scripts in the `~/works/apps/scripts/SP` folder, other input parameters which are not handled through the scripts can be edited manually by editing the files in `~/works/apps/scripts/singlePhase/base` folder that follows, more or less, the OpenFOAM input file format conventions.

2.2 Running simulations

Sample input file is provided in docs/Berea.mhd, which is a ascii header file for the Berea image (from 2009) available on Imperial college pore-scale modelling website: http://www.imperial.ac.uk/earth-science/research/research-groups/perm/research/pore-scale-modelling/micro-ct-images-and-networks.

To run a single-phase flow simulation on a micro-CT image, first the header file should be prepared as discussed in the previous section, then they should be copied in a directory where there is enough disk space; the 3D simulation results take disk-spaces on the order of Gigabytes, depending on the size of the image . Then in this directory run one of the following commands:

```
# 1. Set the Environmental variables if you have not done so far:
#  First, you have to add the scripts for single-phase
#  flow calculation to the system path. To do so, type:
export PATH=$PATH:$HOME/works/apps/scripts/singlePhase
export nProcX=2 # optional
export nProcY=2 # optional
export nProcZ=3 # optional
# 2. Running the single-phase flow solver,
```

```
# to get help for how to run the scripts, type:
AllRunImagePar -h

# to run on image XXX.mhd in serial mode, type:
AllRunImage XXX.mhd
# or to run on all .raw files in the current directory in serial mode:
AllRunImage
# or to run on XXX.raw file in parallel mode on a single machine:
AllRunImagePar XXX.mhd
# or to run on all .mhd files in parallel on a single machine
AllRunImagePar
# or parallel mode on a distributed cluster, edit
    \lstinline|$PoreScaleDir/scripts/singlePhase/machines.txt| first and type:
AllRunImageParDistributed XXX.mhd
```

The same approach can be used to run the simulation on ascii files, provided that you have .dat as the file suffix (e.g. XXX.dat); of course you should replace "XXX.raw" to the name of ascii file, XXX.dat, or set the keyword BinaryData to False.

2.3 Simulation results and visualization

After running the simulation scrips the results will be saved in a subdirectory inside a directory with the same name as the micro-CT image (without the suffix), which includes the log files of individual applications, openfoam simulation results and the pressure and velocity fields converted into raw files, in either ascii (.dat) or binary (.raw) format: To change the default behavior, you have to open the script files AllRunImage/AllRunImagePar (located in folder `$PoreScaleDir/scripts/singlePhase/`) and change

"`: ${outPutFormat:=binary}`" to "`: ${outPutFormat:=ascii}`" or "`: ${outPutFormat:=oldBinary}`" or "`: ${outPutFormat:=oldBinary}`".

In binary format, the velocity and pressure fields are written in small-endian 32bit floating-point numbers. When oldAscii/oldBinary format is chosen the velocity files will have the same size as the original image, but otherwise the (default) Ufx, Ufy and Ufz files will have one additional layers in x, y and z directions respectively (to match the number of faces between all pairs of neighbouring voxels). They can be opened in image-processing software like Avizo and ImageJ. A helper file is written with these files named "`OpenMeInParaview.xmf`" which can be used to load the .raw velocity and pressure files in Paraview or other software supporting .xmf format.

The openfoam simulation parameters and results are saved inside a folder called openfoam "case", which includes subdirectories named "constant" and "system" and in many other "time" directories which have floating-point numbers as their names such as "0" or "0.1", ... To visualize the openfoam files, you can use Paraview and open the system/controlDict file. The permeability and effective (connected) porosity are reported in a file with prefix `summary_` in the case folder. Velocity distribution and formation factors are also computed and written into the same file.

3 References

The single-phase flow solver is derived from the two-phase direct simulation code discussed in the paper:

- Raeini, A.Q., Blunt, M. J. and Bijeljic B. (2012). Modelling Two-Phase Flow in Porous Media at the Pore Scale Using the Volume-of-Fluid Method, Journal of Computational Physics, 231, 5653-5668 http://dx.doi.org/10.1016/j.jcp.2012.04.011

Essentially the main difference with the openfoam-provided codes is the treatment of the no-slip boundary condition for improving the accuracy of permeability prediction. The details of the pre/post-processing tools are not documented, but you can have a look at the source code or send an email (see below) if you want to know more.

Applications of the single-phase direct simulation code can be found in following papers:

- Bijeljic B, Raeini, A. Mostaghimi P., and Blunt, M.J. "Predictions of non-Fickian solute transport in different classes of porous media using direct simulation on pore-scale images", Physical Review E. 87, 013011 (2013), http://dx.doi.org/10.1103/PhysRevE.87.013011
- Bijeljic, B., P. Mostaghimi, and M. J. Blunt "Insights into non-Fickian solute transport in carbonates", Water Resources Research, 49, 2714-2728, (2013) http://dx.doi.org/10.1002/wrcr.20238
- Pereira Nunes, J.P., Bijeljic, B. and Blunt, M.J. "Time-of-Flight Distributions and Breakthrough Curves in Heterogeneous Porous Media Using a Pore-Scale Streamline Tracing Algorithm", Transport in Porous Media (2015) 109: 317. https://doi.org/10.1007/s11242-015-0520-y
- B.P. Muljadi, , M.J. Blunt, A.Q. Raeini, B. Bijeljic, "The impact of porous media heterogeneity on non-Darcy flow behaviour from pore-scale simulation", Advances in Water Resources, (2016) http://dx.doi.org/10.1016/j.advwatres.2015.05.019
- ...

For more information and recent publications, please refer to our website:
http://www.imperial.ac.uk/earth-science/research/research-groups/perm/research/pore-scale-modelling/publications/

If you have any questions, please contact us by email:
Ali Qaseminejad Raeini: a.qaseminejad-raeini09@imperial.ac.uk
Please also report any problems, feedback or suggestions using the email adress above.

A Overview the applications used in the scripts

OpenFOAM utilities:

renumberMesh: renumbers mesh to improve performance (optional)

redistributePar: using scotch to redistribute mesh between processors (optional)

Solvers and utilities not available in OpenFOAM:

rawToFoam(Par): converts raw file to OpenFOAM format, used for single-phase flow simulation. It also does simple image processing tasks like cropping and thresholding. It also lable the disconnected parts of the image and only keeps the largest connected pathway (to avoid linear solver crash). A parallel version of this code is developed but not yet integrated into the poreFoam package (to avoid complexity in compilations), please drop me an email if the gain in computational speed is important for you.

iInterFoam101SP: single-phase flow solver, trimmed version of the two-phase flow solver, the main differences with the OpenFOAM single-phase flow solvers is the implementation of the boundary conditions and the pressure-velocity coupling.

iPotentialFoam: used for initialization of pressure and velocity for improving the convergence of simulations (effective when the Reynold number is low and the image is a typical micro-CT image of a porous rock).

calc_distributions: post-processing, calculates effective porosity and permeability, formation factor and velocity distributions.

FOAM2Voxel: post-processing, converts Openfoam simulation results into .dat/.raw files.

Additional useful utilities:

Ufraw2Uc: converts face-centred Uf*.raw files into cell-centred velocities file, useful for importing velocities in Avizo ...

voxelImageConvert: converts between image formats, applies cropping, resampling etc, supports basic data types other than unsigned char (default); see .mhd header file specifications: https://itk.org/Wiki/ITK/MetaIO/Documentation for more details.

B Using zlib

The voxelImage library can read from .gz compressed raw binary files directly, and write .gz compressed files directly if zlib is installed, in a folder named thirdparty. i.e. "`apps/thirdparty/zlib-1.2.11`". The zlib package can be downloaded from http://zlib.net/. Once zlib is installed, you can activate it by adding (uncommenting) the following entries to the file `voxelImage/Makefile`:

```
ZLIBDIR=../thirdparty/zlib-1.2.11/
ZLIBFLAGS=-DZLIB -I$(ZLIBDIR) -I$(ZLIBDIR)/contrib/iostream3/
ZLINKFLAGS=zfstream.o -L$(ZLIBDIR) -lz
zstreamo=zfstream.o
```

Note: zlib support is experimental at this stage, further integration and testing is needed.