PoreFoam package for direct simulation ofcapillaru dominated two-phase flow.

Ali Q Raeini, Mosayeb Shams, Branko Bijeljic and Martin J. Blunt
Department of Earth Science and Engineering, Imperial College London, UK, SW7 2AZ

Summary

To simplify and automate the use of the preprocessing, processing and post-processing tools, several bash script are developed for different types of simulations using porefoam codes. This document presents a description of the script used to do direct two-phase flow simulations: primary-drainage simulation followed by water-injection simulations.

A short description of the tools used by the script is also given. Although knowing such details is not necessary for using the script; they can be useful to make modifications to the script for changing the simulation set-up not foreseen in the script.

1 Installation

1.1 Prerequisites

Except standard Linux compilers and libraries (g++, cmake and an mpi library), all other prerequisites are provided in a folder named thirdparty. The thirdparty includes zlib, libtiff and a minified openfoam.

The minified openfoam library, after being compiled, can reside side-by-side with other openfoam installations withought any conflict. This means you can install and use other openfoam versions alongside the porefoam codes, but if you do so, you better delete the /works/thirdparty/-foamx3mini/applications contents so that you don't have mutiple copies of the same application.

1.2 Downloading and extracting files

Create a directory in your home (~) folder named "works" and download and extract the codes inside it. You can choose any other folder and the script will work, but this ducument assumes you extract the source codes in the " /works" floder for simplicity. The source codes, will reside in subfolders "`~/works/thirdparthy`" and "`~/works/apps`" (called `msSrc` in the script). When compiling the codes, the executables, libraries and the header files will be generated in "`~/works/bin`" "`~/works/lib`" "`~/works/include`" folders.

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/script ~/works/apps/porefoam* ~/works/apps/voxelImage
```

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

1.3 Compiling the codes

The codes requires a recent C++ compiler, which support -std=c++11 flag. The compiler is set through the variable `psCXX` in file `~/works/apps/Makefile.in`. The default (g++) most likely will work so you don't need to do any change.

To compile the codes, open a terminal and type the following commands:

```
(cd ~/works/ && make -j)
```

After everything compiled and working, you can run the following commands to clean the temporary files

```
(cd ~/works/ && make clean)
```

If instead of the `clean` argument you use `distclean`, the `~/works/lib, ~/works/bin, ~/works/include and ~/works/shared` folders will be deleted, so *be extra carefull* when using this option.

1.4 Setting the Environmental variables:

Edit your `~/.bashrc` file (type `'gedit ~/.bashrc'` to open it) and add the following line at its end:

```
source ~/works/src/script/bashrc
```

This makes the porefoam script available in any new terminal you open.

Input file structure

The format specification for the micro-CT images and their header files are given in the sample header file `Image.mhd` location in the `~/apps/docs/` directory.

Background: "OpenFOAM case" or sometimes simply "case" refers to the directory in which the input files required by OpenFOAM are, and is the directory where the results are saved. It should have two subdirectories named "constant" and "system" and as many directories which have numbers as their names such as "0" or "0.1" ...

Some of the input parameters are controlled through the script which are used to automate simulation set-up, specifically a script AllRunImageTwoPhase , placed in $msSrc/macros/preprocessing folder which is discussed further in the next section. There are comments added to this file for how to set the simulation parameters, the most important of which are given below (copied from the sample AllRunImageTwoPhase script):

```
# contact angle for drainage measured through phase 1 (oil)
 theta0=180

# Darcy velocity(s) for oil phase:
 UDs=" 0.001 "

# The value for this keyword should be 1 or 2. 1 means that the mesh
# will have a single resolution. 2 (the default) means that
# grid-blocks will be twice smaller in the throats
```

```
SHMSuffix=2

# fill a small portion of the image near the inlet with oil (onjecting phase)
oilFilldFrac=0.07

# In case you want to refine the mesh increase this, or vise-versa.
nRefineLevels=1.3

# number of CPU nodes used for flow simulation (minimum is 2)
nProc=8
```

See Appendix D for further information.

Running two-phase drainage simulation

Copy the sample input data file in a directory where you have enough disk space and in the same directory run the script AllRunImageTwoPhase:

```
# in case you haven't put these in your ~/.bashrc file:
source ~/src/script/bashrc

cd PATH_TO_.raw_.mhd_FILES

AllRunImageTwoPhase #prepare the input script/files, note: no ./
```

The above command prepares a base folder and a local `AllRunImageTwoPhase` script for you if they don't already exist. You can change the input parameters as you wish in the local script and the base folder: See Appendix D for more details. Then you can set up and run the simulations by running the local AllRunImageTwoPhase script three times (or more if you want to continue the simulations for longer period), i.e in a terminal type (replace `Image.mhd with the name of your image`):

```
./AllRunImageTwoPhase Image.mhd # Generate the mesh

./AllRunImageTwoPhase Image.mhd # Decompose and set initial and BCs

./AllRunImageTwoPhase Image.mhd # Run a drainage simulation
```

Important: You have to visualize the generated mesh and the decomposed mesh, by paraview before running the flow simulations

Running two-phase imbibition simulation

After finishing with the drainage simulations, the boundary and initial conditions should be changed to make the case ready for an imbibition simulation. The script PrepareForImbibition and PrepareForReverseImbibition, placed in macros/preprocessing folder, are written to make the necessary adjustments. PrepareForReverseImbibition does the same job as the PrepareForImbibition script, except that it prepares the imbibition simulation so that the water is injected from the opposite direction.

To use these script to prepare drainage simulation results files to a case ready for an imbibition simulation, you have to type in a terminal:

```
PrepareForImbibition arg1 arg2 arg3 arg4 arg5 arg6 arg7
```

where:

`arg1`: time at/before the end of drainage to be used as the start of water-injection

`arg2`: darcy velocity (m/s)

`arg3`: contact angle at solid walls

`arg4`: fraction of the image to be filled with water (initial condition), from the inlet.

`arg5`: directory of the drainage case

`arg6`: base name for the water-injection case

Example:

```
export PATH=$PATH:~/apps/macros/preprocessing #in case$

PrepareForImbibition  0.1 0.001 135 0.1 Berea8_0.007 BereaImb
 #  or:
PrepareForReverseImbibition 0.1 0.001 135 0.1 Berea8_0.007 BImbRev
# this command reverses the flow direction
```

After preparing the case for imbibition, open the case in a terminal and run the iInterFoam code manually (Note: you can do this to run a drainage simulation as well, instead of the third `./AllRunImageTwoPhase` run discussed in the previous section):

```
cd BereaImb*/ # go to the directory created for imbibition simulation

source ~/apps/porefoam/bashrc

mpirun -np 8 iInterFoam103 -parallel

# replace 8 with the number of domains which the flow domain is decomposed into ( =
    number of BereaImb*/*/processor* subfolders)
```

A Overview of main applications

Third-party software:

OpenFOAM utilities and libraries: used for used for pre/post-processing and writing specialized pre/post-processing and simulation codes

Paraview: visualization and some post-processing tasks.

OpenSCAD: used for automatizing creation of surface models for simple geometries

In-house developed codes (C++): when there were no alternative available

Linux bash utilities: used as a user-interface to do simple calculations, change input-parameters, and run the pre/post-processing and simulation codes.

OpenFOAM applications:

- **`blockMesh`**: creates simple meshes / background mesh for snappyHexMesh.
- **`renumberMesh`**: renumbers mesh for improving the performance
- **`decomposePar`**: decomposes the mesh into several pieces for parallel runs
- **`reconstructPar`**: reconstruct the decomposed mesh, not needed, everything is run in parallel
- **`setFields`**: used to set/modify the initial condition for the indicator function
- **`splitMeshRegions`**: used to delete the isolated parts of the mesh and keep the largest piece.
- **`createPatch`**: The version of snappyHexmesh used for our mesh generations messes up with the boundaries (called 'patch'es in OpenFOAM); createPatch is used to recreate the inlet/outlet boundaries.

New/modified applications:

- **`voxelToFoam`**: converts .raw/.tif/.am files to OpenFOAM format, used for single-phase flow simulation.
- **`calc_perm`**: calculates single-phase permeability and porosity of single-phase simulations.
- **`voxelToSurface`**: creates a 3D surface between void and solid, used in mesh generation with snappyHexMesh for two-phase flow simulations
- **`surfaceSmoothVP`**: smoothes voxelToSurface output for snappyHexMesh, preserves volume.
- **`calc_gradsImage`**: calculates works and energy losses, used to plot relative permeability.
- **`imageFileConvert`** : converts between raw file format and ascii format, does simple image processing like cropping and thresholding.
- **`iInterFoam`** and `libiInterfaceProperties`, direct two-phase flow simulator, including a new surface tension model, pressure-velocity-surface-tension coupling algorithm and several new boundary conditions,

Installing openfoam-extend-4.0 on Ubuntu OS

To install openfoam-extend-4.0 on Ubuntu OS, type in a terminal (make sure that the each command finishes successfully before running the next!):

```
mkdir ~/foam
cd ~/foam
git clone git://git.code.sf.net/p/openfoam-extend/foam-extend-4.0 foam-extend-4.0
. ~/foam/foam-extend-4.0/etc/bashrc
(cd ~/foam/foam-extend-4.0/ThirdParty/ && ./AllMake)
(cd ~/foam/foam-extend-4.0/ && ./Allwmake )
```

B Post-processing output data

The average of varius flow parameters is computed every 10 time steps and written in a file named `data_out_for_plot`. A header file is also written to help extract the relevant parameter in Excel or in Matlab, named `data_out_for_plot_header`, which look like (all antries in a single line):

```
t maxMagU aAvg aAvgL aAvgR avgUAlpha1_0 avgUAlpha1_1 avgUAlpha1_2 avgUAlpha2_0
    avgUAlpha2_1 avgUAlpha2_2 QIn QOut Dp Dpc pcAvg ADarcy S1-alpha S1-U S1-vol S1-f_1
    S1-dpddz S1-dpcdz S1-dpcdz_1 S1-dpddz_1 S1-viscz S1-viscz_1 S1-phiu S1-phiu_1
    S1-delPdelZ S1-delPcelZ S1-viscInterf_1 S1-viscE S1-viscE_1 S1-dpEc S1-dpEc_1
    S1-dpEd S1-dpEd_1 S1-phiE S1-phiE_1 S1-ZERO S1-Pc S1-xDropAvg S1-xDrop1 S1-xDrop2
    S1-x1 S1-x2 S2-alpha S2-U S2-vol S2-f_1 S2-dpddz S2-dpcdz S2-dpcdz_1 S2-dpddz_1
    S2-viscz S2-viscz_1 S2-phiu S2-phiu_1 S2-delPdelZ S2-delPcelZ S2-viscInterf_1
    S2-viscE S2-viscE_1 S2-dpEc S2-dpEc_1 S2-dpEd S2-dpEd_1 S2-phiE S2-phiE_1 S2-ZERO
    S2-Pc S2-xDropAvg S2-xDrop1 S2-xDrop2 S2-x1 S2-x2
```

Here is a more detailed description of these parameters:
Variables defined over the whole flow domain:

`t`: time (seconds)

`maxMagU`: maximum of magnitude of velocity field (U)

`aAvg`: average of indicator function

`aAvgL`: average of indicator function on Left-side (small x) boundary (usually inlet)

`aAvgR`: average of indicator function on Right-side (large x) boundary (usually inlet)

`avgUAlpha1_0`: average of phase 1 (oil) velocity in x direction (U1x)

`avgUAlpha1_1`: average of phase 1 (oil) velocity in y direction (U1y)

`avgUAlpha1_2`: average of phase 1 (oil) velocity in z direction (U1z)

`avgUAlpha2_0`: average of phase 0 (water) velocity in x direction (U0x)

`avgUAlpha2_1`: average of phase 0 (water) velocity in y direction (U0y)

`avgUAlpha2_2`: average of phase 0 (water) velocity in z direction (U0z)

`QIn`: flow rate at the left side boundary

`QOut`: flow rate at the right side boundary

Dp: average (arithmatic) dynamic pressure drop over the flow domain (`Pd_left-Pd_right`)

Dpc: average (arithmatic) capillary pressure drop over the flow domain (`Pc_left-Pc_right`)

pcAvg: average (volume-weighted) capillary pressure difference between the two phase

ADarcy: Darcy area (=Dy x Dz)

Variables defined over each control-volume (numbered, S1, S2, S3 ... based on their order in the system/postProcessDict):

S1-alpha: saturation (average of the indicator function, alpha)

S1-U: average pore velocity (of both phases)

S1-vol: volume of the control volume

S1-f_1: fractional flow rate of phase 1 (oil)

S1-dpddz: average force per unit volume due to dynamic pressure gradients

S1-dpcdz: average force per unit volume due to microscopic capillary pressure gradients (i.e. imbalance between microscopic capillary pressure and capillary forces)

S1-dpcdz_1: average force per unit volume due to microscopic capillary pressure gradients in phase 1

S1-dpddz_1: average force per unit volume due to microscopic dynamic pressure gradients in phase 1

S1-viscz: average force per unit volume due to viscous forces

S1-viscz_1: average force per unit volume due to viscous forces inside phase 1

S1-phiu: average force per unit volume due to advection of momentum

S1-phiu_1: average force per unit volume due to advection of momentum in phase 1

S1-delPdelZ: rate of energy (per unit time and volume, in SI units) entring/exiting the boundaries of the control volume due to dynamic pressure differences

S1-delPcelZ: rate of energy entering/exiting the boundaries of the control volume due to capillary pressure differences

S1-viscInterf_1: rate of energy crossing the boundary between the two fluid

S1-viscE: rate of energy loss due to viscous forces - used for computing pressure drop

S1-viscE_1: rate of energy loss due to viscous forces inside phase 1 (oil)

S1-dpEc: rate of energy introduced by microscopic capillary pressure gradient

S1-dpEc_1: rate of energy introduced by microscopic capillary pressure gradient in phase 1

S1-dpEd: rate of energy introduced by microscopic dynamic pressure gradient

S1-dpEd_1: rate of energy introduced by microscopic dynamic pressure gradient in phase 1

S1-phiE: rate of kinetic (inertial) energy introduced

S1-phiE_1: rate of kinetic (inertial) energy introduced in phase 1

S1-ZERO: dummy

S1-Pc: average Pc (the difference between pc of the two phases) in the control volume

S1-xDropAvg: an estimate of the length of the bounding box of the control volume

S1-xDrop1: an estimate of the length of the bounding box of the phase 1 in the control volume

S1-xDrop2: an estimate of the length of the bounding box of the phase 0 (water) in the control volume

S1-x1: smalled x covered by the control volume (x of left side)

S1-x2: largest x covered by the control volume (x of right side)

C Modifying simulation parameters

After running the "`AllRunImageTwoPhase`" command for the first time, a local copy of this script and a base folder are copied to the local directory which you can edit for a customized simulation set-up. In the following the parameters which you may need to change to get more accurate or more stable results are discussed briefly, along with some general guidelines.

First **note** that for the first and second runs of the "`./AllRunImageTwoPhase Berea.raw`" which does the mesh generation and decomposition, respectively, the values assigned in the "./AllRunImageTwoPhase" take precedence overwrite the values in the base or Berea folder will, respectively. The first "`./AllRunImageTwoPhase Berea.raw`" run will copy the data from the base folder, the second run copies the data from the Berea folder and the third run justs launches the flow simulator without making any changes to the input files.

Parameters adjustable from AllRunImageTwoPhase script:

cPc=0.2: Capillary pressure compression coefficient, you don't need to change this. But in case you do, your value should be preferably between 0.1-0.49.

cAlpha=1.0: Indicator function (alpha) compression coefficient, you don't need to change this. But in case you do, your value should be preferably between 0.5-2.0.

cPcCorrection=.1: A correction coefficient to eliminate the components of capillary pressure gradient which are parallel to interface and hence non-physical. The value should be between 0.05-0.2. Higher value will lead to less spurious currents but higher stick-slip behavior. The stick-slip behavior is because of the variations in the computed curvature as the interface moves between grid-blocks due to numerical errors and this keyword will increase

such variations and hence increases the stick-slip behavior which becomes dominant as the capillary number becomes smaller.

SmoothingKernel=12: Smoothing coefficient for computation of interface normal-vectors which are used in the computation of interface curvature. Any value between 10(no smoothing) to 19 (9 smoothing iterations) can be given. For coarse meshes a lower value is recommended because otherwise the indicator-function and the computed curvature/pc may become decoupled and the simulations will diverge. A higher degree of smoothing will result in better capillary pressure estimation and is recommended when the mesh resolution is high.

SmoothingRelaxFNearInterf=0.7: relaxation coefficient for smoothing interface normal-vectors. Any value between 0.5-1.0 can be given. For coarse meshes a value close to 1 is recommended because otherwise the indicator-function and the computed curvature/pc may become decoupled and the simulations will diverge. A higher degree of smoothing will be achieved as this value is reduced and consequently will result in better capillary pressure estimation and is recommended when the mesh resolution is high.

wallSmoothingKernel=0: smoothing coefficient for wall normal vectors. Use this to achieve better accuracy if the mesh is generated from a complex voxelized image. If the solid-walls are smooth, then you can set the value to zero which may lead to better accuracy.

Ufilter1=0.015:: the value assigned to this keyword filter (deletes) capillary fluxes (forces and velocities generated due to capillary pressure and curvature force imbalance) when the capillary force is in close equilibrium with the capillary pressure gradient. A value of 0.01, roughly speaking, deletes capillary force imbalances when the imbalance is less than 1% of the capillary force. This will lead to smoother interface motion and also gets rid of small spurious currents. Any value between 0.005-0.02 will lead to physical results. Lower values may let the spurious currents, higher values is considered over-filtering and may lead to under-prediction of trapping.

maxDeltaT=1e-5: largest dt allowed, this should be proportional to grid-size.

system/controlDict file:

maxCo 0.1; maximum courrant number, should be between 0.05 and 0.2, be catioius if you choose higher values. Lower values will lead to higher the accuracy of time discretization higher simulation time.

maxAlphaCo 30; maximum interface Courant number (see Raeini et al 2012 for more details), should be between 0.05 and 0.2, be cautious if you choose higher values. Lower values will lead to higher accuracy of time discretization but also higher simulation time.

system/fvSolution file:

cAlpha 1; same as cAlpha in AllRunImageTwoPhase script.

cPc .2; same as cPc in AllRunImageTwoPhase script.

cBC 250; boundary condition correction coefficient, makes the boundary-condition second-order accurate.

cPcCorrection .1; same as `cPcCorrection` in `AllRunImageTwoPhase` script.

cPcCorrRelax2 1.0; (relaxes) reduces the amount of filtering applied by the cPcCorrection keyword, should be equal or smaller than 1, but anything smaller than 1 may lead to presence of spurious currents.

smoothingKernel 12; same as `cPcCorrection` in `AllRunImageTwoPhase` script.

smoothingRelaxFactor 1.0; same as `cPcCorrection` in AllRunImageTwoPhase script.

wallSmoothingKernel 5; same as `cPcCorrection` in AllRunImageTwoPhase script.

uFilter1 .01; same as `cPcCorrection` in AllRunImageTwoPhase script.

lambda 0; slip length

lambdaS 0; slip length near interface

cSSlip 0.05; threshold value for indicator function (alpha) for detecting interface location for applying the interface slip length (lambdaS). lambdaS is applied to all cells with alpha<1.0-cSSlip.

NSlip 1.0; the distance away from the interface (unit is number of cells) that lambdaS is applied

UBoundByUSmoothFactor 2.0; a filtering factor to eliminate locally high velocities which can potentially cause interface destabilization when the interface is not represented accurately (in coarse meshes, or in very thin film). The value can be anything higher than one, but assigning a value less than 1.5 may lead significant error in the computed velocity. Essentially any cells velocity which is more than its neighboring cell velocity by more than this factor is penalised to this factor multiplied by the average of neighboring cell velocities.

constant/transportProperties file:

```
nu          nu [ 0 2 -1 0 0 0 0 ] 1e-06; // viscosity divided by density (SI units)

rho         rho [ 1 -3 0 0 0 0 0 ] 1000; //density (SI units)}
```

The above keywords should be assigned to each phase separately, phase0 is water and phase1 is oil.

```
sigma       sigma [ 1 0 -2 0 0 0 0 ] 0.03; //surface tension (SI units)
```

References

For more technical detail on the direct single and two-phase flow solvers, please refer to the following papers.

Two-phase direct simulation:

1) 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.

Raeini, A.Q. , Bijeljic B and Blunt, M. J. (2014). Numerical Modelling of Sub-pore Scale Events in Two-Phase Flow in Porous Media, Transport in Porous Media, 101, 191-213,http://dx.doi.org/10.1007/s11242-013-0239-6.

Raeini, A.Q., M.J. Blunt, B. Bijeljic, "Direct simulations of two-phase flow on micro-CT images of porous media and upscaling of pore-scale forces", Advances in Water Resources 74 116–126, (2014) http://dx.doi.org/10.1016/j.advwatres.2014.08.012

A.Q. Raeini, B. Bijeljic and M.J. Blunt, "Modelling capillary trapping using finite-volume simulation of two-phase flow directly on micro-CT images", Advances in Water Resources, 83, 102-110 (2015) http://dx.doi.org/10.1016/j.advwatres.2015.05.008

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

If you have any questions, please contact my by email:

Ali Qaseminejad Raeini: a.qaseminejad-raeini09@imperial.ac.uk