HybGe-Flow3D User Manual, Version 0.0.1

Timothy B. Costa

October 6, 2015

Contents

Chapter 1

HybGe-Flow3D Overview

HybGe-Flow3D (HGF) is a software package that solves laminar fluid flow problems in complex geometries and produced upscaled conductivites. This software is distributed freely AS IS and with ABSOLUTELY NO WARRANTY, and with the hope that it will be used and extended by the scientific computing community. In this document we describe how to compile and run Hybge-Flow3D.

This software was developed under the partial support of the National Science Foundation, on the project NSF-DMS 1115827 "Hybrid modeling in porous media."

1.1 License & Citation

Publications making use of HybGe-Flow3D should cite this software package. An example citation is given as:

> Costa, T., "HybGe-Flow3D", Package Version 0.0.1, http://github.com/costat/HybGe-Flow3D.

1.2 Model & Discretization

The basic equations solved by HGF are the Stokes equations, modified with a resistive term corresponding to an immersed boundary representation of complex geometry.

$$\begin{aligned} -\mu\nabla^2 u + \frac{1}{\eta}\chi_{\Omega_g} &= -\nabla p, \quad x \in \Omega, \\ u &= u_D, \quad x \in \partial\Omega_D \\ \nabla u \cdot n &= 0, \quad x \in \partial\Omega_N. \end{aligned}$$

Here, $\Omega \subset \mathbb{R}^d$ with $d = 3$ or $d = 2$ is the simulation domain, and $\partial\Omega$ is its boundary. $\partial\Omega_D$ refers to the 'Dirichlet boundary' where the fluid velocity is prescribed. This is used for no-slip and inflow boundary conditions. $\partial\Omega_N$ refers to the 'Neumann boundary' where a homogeneous diffusive flux is prescribed. This is used for outflow conditions. Additionally we have $\Omega = \Omega_g \cup \Omega_f$, $\Omega_g \cap \Omega_f = \emptyset$, where Ω_g refers to the immersed boundary and Ω_f the fluid flow domain. In these equations u is the fluid velocity, p is the fluid pressure, and μ is the fluid viscosity. χ_{Ω_g} is the indicator functions for the immersed boundary domain Ω_g, and η is a penalization parameter, taken to be very small. HybGe-Flow3D solves the above fluid flow model by a staggered-grid finite volume discretization.

Chapter 2

Installation

HybGe-Flow3D uses the Paralution linear algebra package. In this section we review the process of building Paralution, installing an environment module file for Paralution, and then compiling HybGe-Flow3D on a Linux machine.

2.1 Building Paralution

Paralution 1.0.0 is straightforward to build on Linux machines using cmake. The instructions contained here can be found at
http://www.paralution.com/download/.

First, navigate to the directory in which you will build paralution. Then grab the tar file containing Paralution.

```
$ wget http://www.paralution.com/downloads/paralution-1.0.0.tar.gz
```

The following steps build paralution using cmake.

```
$ tar zxvf paralution-1.0.0.tar.gz
$ cd paralution-1.0.0
$ mkdir build
$ cd build
$ cmake ..
$ make
```

Next we write an environment module file for Paralution. This file is placed at /usr/share/modulefiles/paralution-1.0.0.

```
#%Module 1.0
#
#  Paralution module
#

module-whatis "paralution/1.0.0"

set paralution_home /path/to/paralution/build
prepend-path    PATH                $paralution_home/bin
```

```
prepend-path    LIBRARY_PATH          $paralution_home/lib
prepend-path    LD_LIBRARY_PATH       $paralution_home/lib
prepend-path    CMAKE_LIBRARY_PATH    $paralution_home/lib

prepend-path    INCLUDE_PATH          $paralution_home/inc
prepend-path    C_INCLUDE_PATH        $paralution_home/inc
prepend-path    CPLUS_INCLUDE_PATH    $paralution_home/inc
prepend-path    CMAKE_INCLUDE_PATH    $paralution_home/inc

setenv          PARALUTION_HOME       $paralution_home
setenv          PARALUTION_DIR        $paralution_home
```

Finally, type

```
$ module avail
```

and verify that paralution-1.0.0 is an available module.

2.2 Building HybGe-Flow3D

Once paralution is built and a module file is in place, building HybGe-Flow3D is simple. First, clone (or download from www.github.com/costat/HybGe-Flow3D/) the repository.

```
$ cd /path/to/build/hgf
$ git clone git@github.com:costat/HybGe-Flow3D
```

Then navigate into the repository.

```
$ cd ./HybGe-Flow3D
```

Next, load the Paralution module.

```
$ module load paralution-1.0.0
```

Finally, execute the compile script.

```
$ ./compile.bash
```

Chapter 3

Running HybGe-Flow3D

The user needs to interact with two files to simulate a problem. First, a geometry file needs to be provided. Second, the boundary conditions and requested output are handled in the top section of the file stokesolve.py:

```
import numpy as np
import time
import sys
import hgf
import re

###############################################################################
### PROBLEM SETUP, USER DEFINES GRID, VISCOSITY, AND NUMBER OF OMP THREADS ###
###############################################################################

# GRID INFORMATION. USER PROVIDES PATH TO .DAT FILE CONTAINING
# VOXEL ARRAY OF 0S 1S AND 2S.
# ALSO,USER PROVIDES TOTAL GRID LENGTHS IN EACH DIRECTION.
gridfiles = './grids/test2d.dat'
L = 1.
W = 1.
H = 1.

# PRINCIPAL FLOW DIRECTION, 0 - X, 1 - Y, 2 - Z, SINGLE FLOW DIRECTION SOLVES,
# PRODUCES CONSTANT K FOR USE IN PORE-NETWORK THROATS
# 3 - ALL DIRECTIONS, PRODUCES UPSCALED K TENSOR
direction = 0

# SET VISCOSITY
visc = 0.001

# NUMBER OF OMP THREADS FOR USE IN PARALUTION LINEAR ALGEBRA
nThreads = 4

#########################################################
### SWIG TRANSLATIONS, USER SHOULD NOT EDIT BELOW HERE ###
#########################################################

.
.
.
```

We see here that the user needs to provide a grid data file, select total grid dimensions, choose a boundary condition problem type, set the viscocity, and choose the number of openMP threads available to Paralution. In the next two sections we review the details of these choices, beginning with describing the grid data file.

3.1 Problem Geometry

Solving a problem on a new geometry requires the creation of a file <geometryname>.dat. The file contains a voxel array detailing whether a cell in the geometry is inactive (not part of the computational domain) or active either fluid domain or immersed boundary domain. All geometry files describe a rectangle (in 2D) or a box (in 3D), but innactive cells are not part of the computational domain, and are ignored by the gridder. The following is a very simple 2D example.

```
nx = 21, ny = 13, nz = 0
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
```

In this file the first line lists the total number of cells along the x axis, nx, y axis, ny, and z axis, nz. For a 2D problem nz is set to 0. Next the array of 0s 1s and 2s describes the status of each cell in in the geometry. A cell value of 0 corresponds to the fluid flow domain, a cell value of 1 corresponds to inactive cells, and a cell value of 2 corresponds to the immersed boundary. So, the example above correponds to a simple pore geometry with an object blocking the flow in the center that is enforced by the immersed boundary term.

3.2 Problem Selection

After a geometry file has been created the user must edit the top section of the file stokesolve.py.

```
##############################################################################
### PROBLEM SETUP, USER DEFINES GRID, VISCOSITY, AND NUMBER OF OMP THREADS ###
##############################################################################

# GRID INFORMATION. USER PROVIDES PATH TO .DAT FILE CONTAINING
# VOXEL ARRAY OF 0S 1S AND 2S.
# ALSO,USER PROVIDES TOTAL GRID LENGTHS IN EACH DIRECTION.
gridfiles = './grids/test2d.dat'
L = 1.
W = 1.
H = 1.

# PRINCIPAL FLOW DIRECTION, 0 - X, 1 - Y, 2 - Z, SINGLE FLOW DIRECTION SOLVES,
# PRODUCES CONSTANT K FOR USE IN PORE-NETWORK THROATS
# 3 - ALL DIRECTIONS, PRODUCES UPSCALED K TENSOR
direction = 0

# SET VISCOSITY
visc = 0.001

# NUMBER OF OMP THREADS FOR USE IN PARALUTION LINEAR ALGEBRA
nThreads = 4
```

First, the geometry file location is given in line 16,

```
gridfiles = './path/to/<geometryname>.dat'
```

Second the total dimensions of the geometry need to be given. Note that these values correspond to the total length of the box or rectangle, including inactive cells. L is the length in the x direction, W is the width in

the y direction, and H is the height in the z direction. For a 2D problem H may be set to any value, as the nz = 0 term in the geometry file informs the solver that the problem is in 2D.

Next the the boundary condition setup is determined by setting the integer 'direction' in line 24. This integer is set to a value of 0, 1, 2, or 3, which corresponds to the following flow problems:

- direction = 0 → inflow on x min wall, outflow on x max wall, no slip everywhere else.
- direction = 1 → inflow on y min wall, outflow on y max wall, no slip everywhere else.
- direction = 2 → inflow on z min wall, outflow on z max wall, no slip everywhere else.
- direction = 3 → solves directions 0, 1, and 2 (0 and 1 in 2D) so that a full upscaled hydraulic conductivity tensor can be computed.

Next the user sets the fluid viscosity in line 27. Finally, the user tells the software how many openMP threads to use in the Paralution package.

3.3 Simulation & Output

Now that the top section of stokesolve.py has been set up, the code is executed simply by

```
$ python stokesolve.py
```

Here is an example terminal output from solving the x-flow problem on the test geometry 'test3d.dat' on a single sandy bridge core.

```
$ python stokesolve.py

  Total grid generation time: 2.82711696625

  Solving the stationary Stokes problem...

  This version of PARALUTION is released under GPL.
  By downloading this package you fully agree with the GPL license.
  Number of CPU cores: 1
  Host thread affinity policy - thread mapping on every core
  Number of CPU cores: 1
  PARALUTION ver B1.0.0
  PARALUTION platform is initialized
  Accelerator backend: None
  GMRES(30) solver starts, with preconditioner:
  ILU(1) preconditioner
  ILU nnz = 58968
  IterationControl criteria: abs tol=1e-15; rel tol=1e-06; div tol=1e+08; max iter=1000000
  IterationControl initial residual = 0.298187
  IterationControl RELATIVE criteria has been reached: res norm=1.2121e-07; rel val=4.06488e-07; iter=21
  GMRES(30) ends
  Done. Total time: 0.323477983475
```

After a simulation is run, there will be two types of output files in the HGF directory.

If a single flow direction is solved, these files will be flowrun.dat, and and one of Kconstant¡X,Y,Z¿.dat, depending on the flow direction.

The file flowrun.dat contains everything necessary for visualizing the solution: each component of the velocity, the pressure, a list of cells in the immersed boundary (for blanking) and mesh information. The format of the file is written to be easily loaded into Tecplot for visualization, but Paraview (free) is also possible.

The file KConstant⟨X,Y,Z⟩.dat contains the constant conductivity computed from the solution.

If all flow directions are solved (direciton = 3 in stokesolve.py) then 7 files are produced. These are flowrun⟨X,Y,Z⟩.dat, KConstant⟨X,Y,Z⟩.dat and KTensor.dat. These files contains the solutions from each solution, the constant conductivities from each solution, and the upscaled conductivity tensor, respectively.

Chapter 4

Examples

The following examples are found in the included directory /path/to/hgf/grids/.

4.1 Examples in 2D

We begin with a simple problem: Poiseuille flow in a 2D pipe. This example is found at /path/to/hgf/grids/pflow2d.dat. Figure 4.1 shows the pressure and x-component of the velocity.

Next we simulate on a simple 2d pore geometry, found at /path/to/hgf/grids/ellipse.dat. Figure 4.2 shows the pressure and x-component of the velocity.

4.2 Examples in 3D

Next we show Poiseuille flow in a 3D pipe. This example is found at /path/to/hgf/grids/pflow3d.dat. Figure 4.3 shows the pressure and x-component of the velocity.

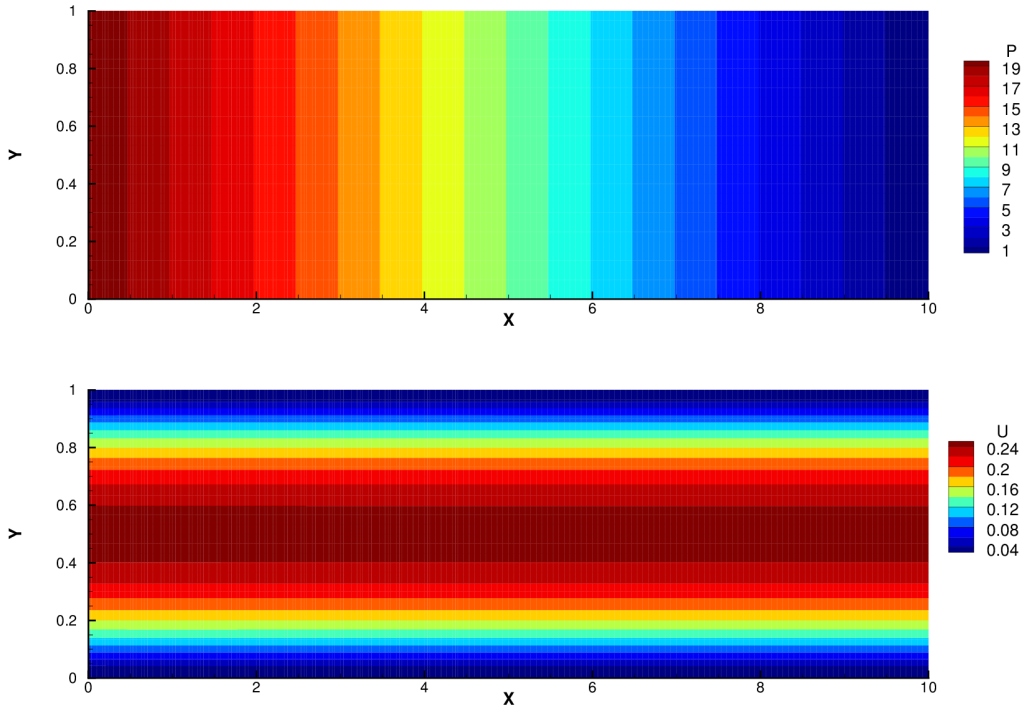

Figure 4.1: Poiseuille flow in a 2D pipe.

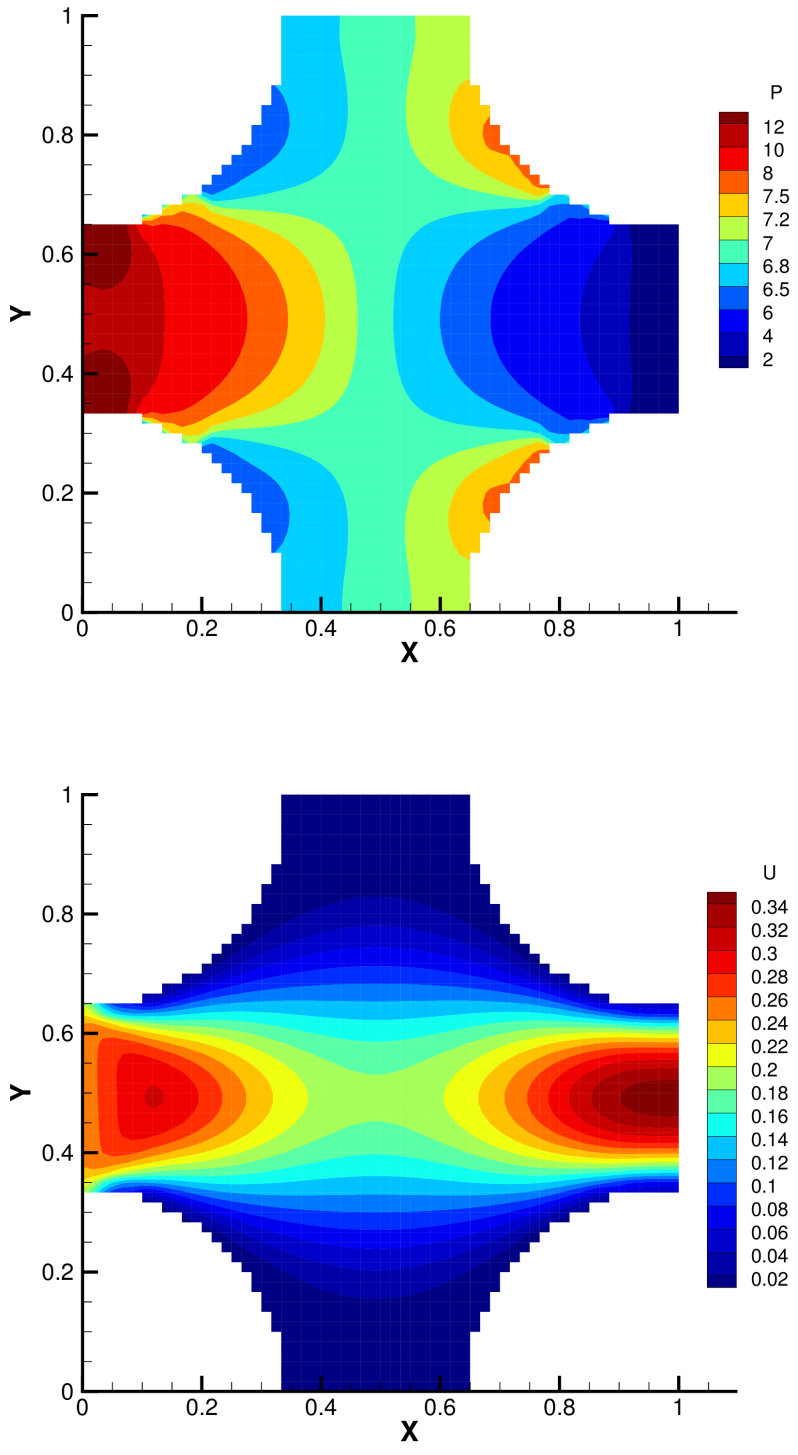

Figure 4.2: 2D ellipse geometry.

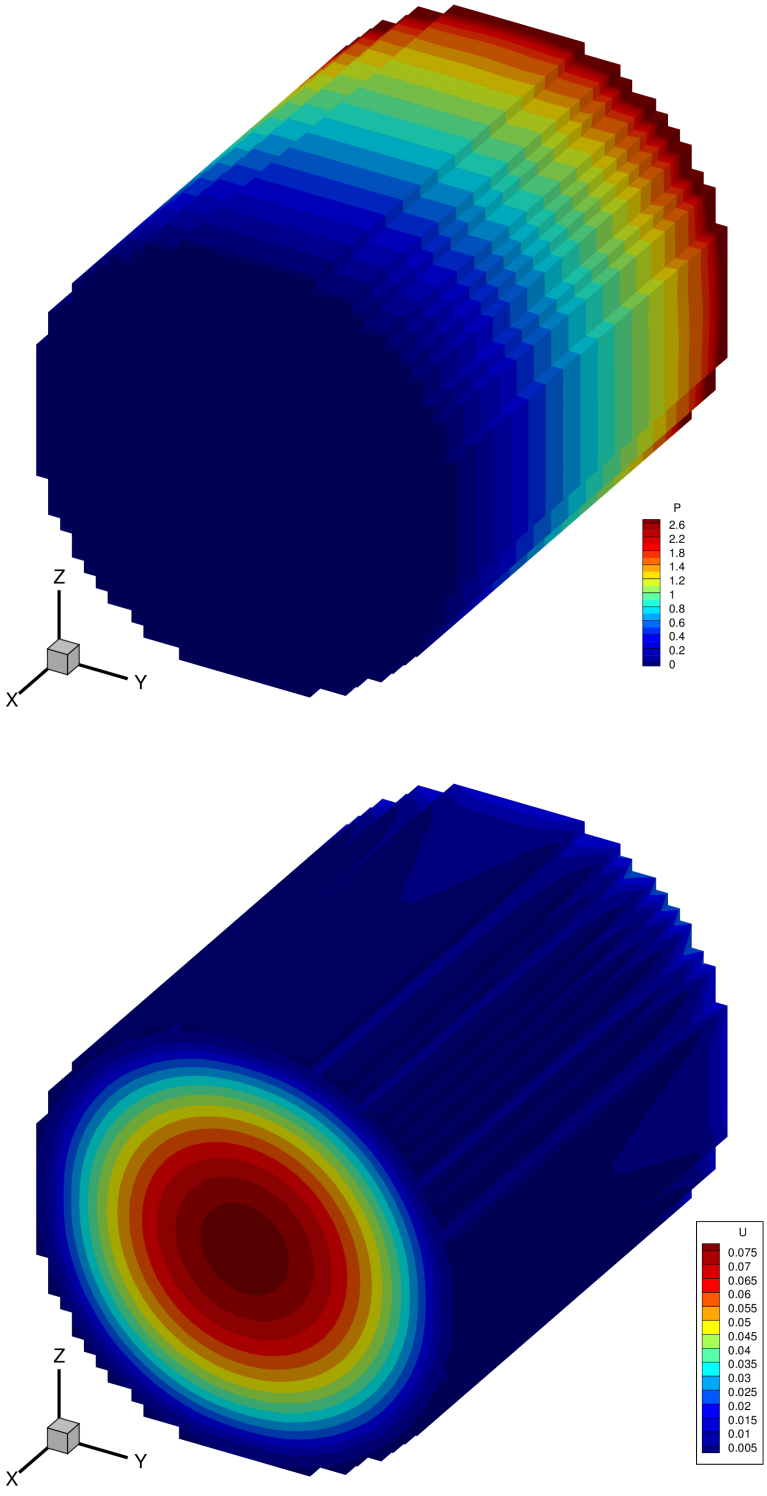

Figure 4.3: Poiseuille flow in a 3D pipe.