HybGe-Flow3D User Manual, Version 1.0.0

Timothy B. Costa

February 25, 2016

Contents

Chapter 1

HybGe-Flow3D Overview

HybGe-Flow3D (HGF) is a software package that solves multiscale laminar fluid flow problems in complex, uncertain geometries. 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

HybGe-Flow3D Copyright (C) Timothy B. Costa.

This program is free software; you can redistribute and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the license, or any later version.

This software is distributed AS IS and WITHOUT ANY WARRANTY.

A copy of the GNU General Public License is included in the source and can also be found at http://www.gnu.org/licenses/.

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

Costa, T., "HybGe-Flow3D", Package Version 1.0.0, http://github.com/costat/HybGe-Flow3D.

1.2 Model & Discretization

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

$$-\mu\nabla^2 u + \frac{1}{\eta}\chi_{\Omega_g} u = -\nabla p, \quad x \in \Omega,$$

$$u = u_D, \quad x \in \partial\Omega_D$$

$$\nabla u \cdot n = 0, \quad x \in \partial\Omega_N.$$

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 function 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.

HybGe-Flow3D has several problem scenarious built in. In the simplest case, HGF will simply solve the above model on a given geometry, and produce a constant upscaled absolute permeability based on the flow direction chosen. Additionally, HGF can be instructed to solve the flow problem in all principal axis directions and produce the full upscaled absolute permeability tensor.

More interesting are the full multiscale simulation capabilities of HGF. In later sections thsese will be examined in detail. Briefly, HGF can ben instructed to cut a geometry into subdomains and solve local porescale flow problems. On these subdomains absolute permeabilities are computed, and used in a constructed pore-network model. This pore-network model is then solved to produce a permeability for the full domain.

Additionally, HGF can be instructed to sample a stochastic resistive term on each subdomain and produce empirical probability density functions relating the permeability to the presence of growth of an obstruction in the void space. These densities are then sampled for throat permeabilities in a pore-network solve. In future work we anticipate this being useful for transient simulations at pore-network or corescale with reactive transport.

Chapter 2

Installation

HybGe-Flow3D has two variants. The first uses Paralution for linear algebra, and the second uses MAGMA.

In addition to requiring either Paralution or MAGMA. HybGe-Flow3D requires CUDA and BOOST C++ Libraries. In particular, HGF requires that the system, filesystem, and serialization libraries be compiled. In this section we review the build process for Paralution, MAGMA, BOOST, CUDA, and HybGe-Flow3D on a linux machine. We make no promises that these steps work universally, but they have worked reliably for the author on a Debian based machine.

2.1 Building Paralution

Paralution 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-1.1.0.tar.gz
```

The following steps build paralution using cmake.

```
$ tar zxvf paralution-1.1.0.tar.gz
$ cd paralution-1.1.0
$ mkdir build
$ cd build
$ cmake ..
$ make
```

Next we write an environment module file for Paralution. We assume for exposition that this file is placed at /path/to/modulefiles/paralution-1.1.0, and describe the process of adding the folder /path/to/modulefiles/ to the module path while using a bash terminal.

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

```
module-whatis "paralution/1.1.0"

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
```

To see the modulefiles within /path/to/modulefiles we add the following line to our .bashrc:

```
module use-append /path/to/modulefiles
```

Open a new terimal or type

```
$ source  /.bashrc
```

Finally, type

```
$ module avail
```

and verify that paralution-1.1.0 is an available module.

2.2   Building MAGMA

2.3   Building BOOST

Many BOOST libraries do not require separate compilation, as they are header-only libraries. In our case, though, we require the following compiled libraries:

- serialization

- filesystem

- system

2.4   Building CUDA

2.5   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.

Chapter 3

Running HybGe-Flow3D

Chapter 4

Examples