Workflow for the Heart Tube Project

Lindsay Waldrop

10/5/2020

Introduction

This is the simulation and data analysis pipeline for heart tube project with multi-circulatory systems. It includes all the steps necessary to generate results from cloning the project on Github to plotting figures for publications in R.

Overview of the pipeline

This pipeline takes the following inputs:

- IBAMR source code for the opposing sine-wave peristalsis
- Generalized polynomial chaos (gPC) simluation set with n number of simulations (either 165 or 681) establishing parameter values for the compression frequency, compression ratio, and Womersley number.

This pipleine produces the following outputs:

- IBAMR flow data
- Calculated velocities within the tubes
- Calculated volume flow rates
- Calculated work and cost of transport of fluid flow in the tube

The pipeline has the following components:

1. The cloning of project from Github.
2. The generation of directories, compiling of the main code (main2d) from C and C++ IBAMR source code, and input files and parameters files necessary for IBAMR and data analysis in VisIt (lineout files) for the gPC simulation set.
3. The generation of vertex, target, beam, and spring files in MATLAB.
4. The execution of IBAMR simulations for the gPC simulation set.
5. The execution of VisIt data analysis to extract fluid speed information from each IBAMR simulation.
6. The execution of R scripts for calculation of final values.
7. The generation of figures for publication from analyzed data.

1. The cloning of the projection from Github

To start the project, clone the project from Github using the following in command-line bash:

```
$ git clone https://github.com/lindsaywaldrop/peri-gPC-git.git
```

After entering the cloned directory, save a path to your top directory:

```
$ WD=$PWD
```

2. Generation of directories, compiling main code, and generation of input files

You will then need to set up the IBAMR code to run with your build of IBAMR. Enter the `src/ibamr` director and edit the following lines of the Makefile:

- Line 4: path to your IBAMR source code
- Line 5: path to the IBAMR build you wish to use

After that, enter the `src/bridges` directory. Run the script in command-line bash:

```
$ sh compile_heart.sh "$WD" track i name
```

where `track` should be the circulatory system you wish to run (either racetrack, branch, obstacles, or branchandobstacles); `i` should be the number of simulations in the gPC you wish to use; and `name` is your username for Bridges. (Note: if you are running this on another HPC resource, `src/bridges/runperi.job` file will need to be changed according to the needs of your resource.)

This shell script will set up the directories necessary for the organization of the project and then compile the `main2d` executable program that runs IBAMR simulations. It will also run the `setinput2d.sh` and `setparameters.sh` files in the `src/bridges` which will set up all the input2d and parameters files necessary to run the the IBAMR simulations.

3. Generation of IBAMR files in MATLAB

The spring, beam, vertex, and target files must be generated in MATLAB and then moved into the `bin` directory in order to run IBAMR simulations. These steps can be accomplished by running the `generate_ibamr-files.sh` script in the `src/matlab` directory. Once in this directory, enter:

```
$ sh generate_ibamr-files.sh "$WD" track
```

where `track` is the circulatory system you wish to use. The files will be generated, copied to the `bin` directory and then moved to `data/ibamr-files/track` directory.

4. Execution of IBAMR simulations

Note: these instructions are specifically for the Bridges cluster at PSC, but they can be modified to include any standard HPC cluster running SLURM as a job-management system.

There are two files in `src/bridges` that control submitting batch jobs to the Bridges cluster: `runperi.job` and `setrunperijobs.sh`. `setrunperijobs.sh` generates a set of temporary job files and submits them via sbatch, based on `runperi.job` as a template.

Each simulation takes between 5 and 7 hours to run on Bridges as it is currently configured. However, should you wish to change parameters, lines 2 through 6 in `runperi.job` set out options passed to SLURM during submission. All other aspects of the file have already been updated when `compile_heart.sh` was run.

After altering these, you can run any number in a series of simulations at one time using `setrunperijobs.sh`. In command-line bash, simply provide the start number as the first number after the shell script and the end number as the second. In this example, simulations 1 through 20 are started using `sbatch`:

```
$ sh setrunperijobs.sh 1 20
```

When these runs have finished successfully, the viz_IB2d files are automatically zipped and will be deposited in `results/runs`. Log files will be located in `results/log-files`. It is recommended that you move completed simulation directory `results/ibamr/runs` to `results/ibamr/track_runs` (where track is the name of the circulatory system you used) in keeping with the structure of the project.

5. Execution of VisIt data analysis scripts

Check for simulation set completeness

Each simulation needs to be complete before the VisIt script will run successfully. In order to check runs, in the `src/visit` folder, run:

```
$ sh checkruns.sh "$WD" track n
```

where `WD` is the top working directory, `track` is the name of your racetrack, and `n` is the number of simulatinos to check. This should produce a list of simulations not found or that are incomplete (it returns simulation not found for both situations).

Execute VisIt Scripts

To execute the python script, use the shell script `bashpy.sh` in the folder `src/visit` through command-line bash:

```
$ sh bashpy.sh "$WD" track s e
```

where `WD` is the top working directory, `track` is the name of your racetrack, `s` is the start number of you simulation set and `e` is the end number. For instance, if you'd like to run all simulations for the branch and obstacles track, from `src/visit` you would enter: `sh bashpy.sh "$WD" branchandobstacles 1 165`.

The script will deposit curve files in `results/visit/{track}_runs/sim{i}/`. A number of files will be produced for each simulation. This script needs to be run only once.

Checking for VisIt results completeness

To check the visit result curve files, which are needed for the R scripts in the next steps, run the following in `src/visit`:

```
$ sh checkvisitresults.sh "$WD" track n
```

where `WD` is the top working directory, `track` is the name of your racetrack, and `n` is the number of simulations in the set. The script will output a list of the specific files that are missing in each simulation, including stand-alone files and time-series files.

6. Execute final R scripts to calculate results

Two scripts need to be run to produce the final csv files with calculated results. These are the `alldata.R`, which uses individual curve files, and `vel_profile.R` files, which uses time-series data.

In both R scripts, you will need to change the value for `track` to your circulatory system and the value of `n` to the number of simulations in your set.

For the all data file, in the top working directory, run R and enter:

```
> setwd(getwd()) > source('src/r-scripts/alldata.R')
```

For the velocity profile file, enter next:

```
> source('src/r-scripts/vel_profile.R')
```

This will produce a folder with time series data for each simulation and some combined results of Q and other measures of fluid flow in the circulatory system.