entcode multiarray project – pipeline

Lindsay Waldrop

9/24/2020

Introduction

This simulation and data analysis pipeline for the entcode multiarray project. It includes all the steps necessary to generate results through each step of the pipeline from cloning the project on Github to figures for publication.

Overview of the Pipeline

This pipeline takes the following inputs:

- IBAMR source code for rigid body constraint method.
- Generalized polynomial chaos (gPC) simulation set with n number of simulations (either 165, 681, or 1233) establishing parameter values for angle of array to flow, gap width to diameter ratio, and Reynolds number.

This pipeline produces the following outputs:

- IBAMR flow data
- Calculated shear around hairs
- Calculated leakiness through each row of the array

The pipeline has the following components:

1. The generation of directories and compiling of the main code (main2d) from C and C++ IBAMR source code.
2. The generation of input files necessary for IBAMR (vertex files, input2d files) and data analysis in VisIt (csv files, lineout files) for the gPC simulation set.
3. The execution of IBAMR simulations for the gPC simulation set.
4. The execution of VisIt data analysis to extract fluid speed information from each IBAMR simulation.
5. The execution of Matlab code for advection-diffusion model.
6. The generation of figures for publication from analyzed data.

1. Cloning from Github, establishing directories, and compiling IBAMR code

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

```
$ git clone https://github.com/lindsaywaldrop/entcode.git
```

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

```
$ WD=$PWD
```

Note that `WD` will need to be reset with the same process for each new instance of bash.

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_sniff.sh "$WD"
```

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.

2. Generation of input files for IBAMR

For IBAMR to successfully run, each simulation needs two files: an input2d file, and a vertex file containing hair positioning data. You will need to generate each of these files based on which gPC simulation set you wish to use.

Input2d file generation

Input2d files are generated during the execution of `compile_sniff.sh`. You need to take no further action.

The input2d files will be located in `data/input2d-files`.

Vertex file generation

Vertex files describe the positions of the antennule and each hair in the array. These are divided up by the number of hairs in each array:

- 3 hairs (one row)
- 5 hairs (one row)
- 7 hairs (two rows)
- 12 hairs (three rows)
- 18 hairs (four rows)
- 25 hairs (five rows)

Each vertex file needs to be different depending on the parameters set by the gPC simulation set. Additionally, the center x and y coordinates of each hair are required for data analysis in csv files. Both sets of files are generated by `generate_grid2d_multirow.R` in `src/r-scripts`. To run this file, enter in the appropriate directory and in R and edit the following lines of this script:

- Line 29: the endrun parameter should equal the number of simulations in the gPC set (either 165, 681, or 1233)
- Line 30: the number of hairs in the array (either 7, 12, 18, or 25)

Once changed, return to the main directory and run the file using the R console:

```
> setwd(getwd())
```

```
> source('src/r-scripts/generate_grid2d_multirow.R')
```

If you are running the one row with 5 hairs, it is necessary to run a separate script: `generate_grid2dSY.R` in the same folder. In this file, change the lines:

- Line 23: the endrun parameter should equal the number of simulations in the gPC set.

Then run the script:

```
> setwd(getwd())
```

```
> source('src/r-scripts/generate_grid2dSY.R')
```

The vertex files will be located in `data/vertex-files` in a directory corresponding to the number of hairs in the array. The csv files will similarly be located in `data/csv-files`.

3. 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: `runhairs.job` and `setrunhairsjobs.sh`. `setrunhairsjobs.sh` generates a set of temporary job files and submits them via sbatch, based on `runhairs.job` as a template.

Each simulation takes between 30-40 minutes to run on Bridges as it is currently configured. However, should you wish to change parameters, lines 2 through 6 in `runhairs.job` set out options passed to SLURM during submission.

To run, you will need to change the following lines in `runhairs.job`:

- Line 15: the number of hairs in the array you wish to run
- Line 18: the path to the top directory of the entcode project
- Line 24: any modules that must be loaded to run IBAMR

After altering these, you can run any number in a series of simulations at one time using `setrunhairsjobs.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 setrunhairsjobs.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/nhair_runs` (where n is the number of hairs in the array) in keeping with the structure of the project.

4. Execution of VisIt data analysis

In order to make calculations on flow speed, we need to extract velocities from the data generated by IBAMR. The extraction will take place through VisIt (using a python script). In this script, VisIt will draw a line through the end hairs of the array and sample velocities along that line. For an example, in a 12-hair array, we would want to sample along the first row (hair 3 to hair 2), the second row (hair 7 to hair 6), and the third row (hair 8 to hair 12).

Checking for simulation set completeness

All simulations must be complete before running data analysis in VisIt. To check for completeness of the simulations, run the shell script `checkruns.sh` in command-line bash using the top-level directory as the first argument, the number of hairs in the array as the second, and the number of simulations as the third:

```
$ sh checkruns.sh "$WD" 25 165
```

This script will return "Simulation found" if it exists, is unzipped, and is complete. If these conditions are not met, it will return the number of the simulation missing.

Generating lineout files

Fluid speeds need to be sampled along a line that intersects all hairs in a row. In order for VisIt to extract fluid speeds, it needs to know where each hair exists in the doman for each array configuration in each simulation. Then these values need to be put in a standard form so that VisIt can read them automatically from the python script.

This is done by way of a standard lineout text file, which you can generate using `generate_lineouts.R` in `src/r-scripts`. Be sure to edit:

- Line 13 (nohairs) with the number of hairs in the array. (Note: separate files are produced with running either 5 or 25 hairs, but 7, 12, 18, and 25 hair arrays may use the 25-hair lineout files.)

- Line 18 (endrun) with the number of simulations in the set.

In the R console in the main directory, run:

```
> setwd(getwd())
```

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

This script will read in the csv files in `results/csv-files` and deposit the lineout files into `data/lineout-files`. This only needs to be run once, as all the hairs in all the arrays conform to the same standards.

Running VisIt python scripts

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

```
$ sh setlineout_multihair.sh "$WD" 25 1
```

For this example, the 25-hair array is processed for row 1. VisIt will write these values to a .curve file and store them in `results/visit/nhair_runs/simi/hairlinej` where n number of hairs in the array, i is the simulation number, and j is the row number.

This script needs to be executed independently for each row in the array.

For the second script, through the command line, enter:

```
$ sh setlineout_flux.sh "$WD" 25
```

Specify only the number of hairs in the array after the working directory. It will deposit curve files in `results/visit/nhair_runs/simi/hairline_flux`. Four files will be produced for each hair. This script needs to be run only once.

For the third script, through the command line, enter:

```
$ sh setlineout_shear.sh "$WD" 25 165
```

Specify the number of hairs in the array after the working directory and the number of simulations in your set after that. It will deposit curve files in `results/visit/nhair_runs/simi/shear`, with one file per hair. This script only needs to be run once.

Checking for VisIt results completeness

To check that VisIt provided a complete data set for R to do final calculations, run the following script in the src/visit directory:

```
$ sh checkvisitresults.sh "$WD" 25 165
```

where the first argument is the top working directory, the second is the number of hairs in the array, and the third is the number of simulations to check. This script will provide a list of files in the flux, hairline (leakiness), and `Umean` that are missing so you can check individual simulations and rerun as necessary.

Execution of R scripts for calculation of final flow values

The final values for leakiness, shear around hairs, flux through hair area, and mean velocity around hairs are calculated in R. Each script should be changed in two places before execution, the value `n'' which is the number of simulations in the set and`nohairs" which is the number of hairs in the array.

After changing these values, run each script by in the top directory:

```
> setwd(getwd())
```

And then for leakiness:

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

Shear:

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

Flux:

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

and mean velocity around hairs:

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

Final values will be saved in csv files in `results/r-script-csv/nhair_results/` where n is the number of hairs in the array.

5. The execution of Matlab code for advection-diffusion model

As a part of the full model, there is a set of scripts and functions that run a advection-diffusion model of odor capture in Matlab. These scripts are straightforward to run on an HPC resource, included here are instructions for running these on Bridges.

Checking for IBAMR simulation set completeness and correct file structure

If you have not already done so, check for the presence of all IBAMR viz_IB2d files by running the following script located in `src/visit`:

```
$ sh checkruns.sh "$WD" 25 165
```

Any missing simulations should be rerun and placed in the correct directories.

Matlab will rely on a standard file structure in order to create and save data. These should be in the format: `$WD/resultsodorcapture/{num}hair_array/{fluid}` where `{num}` should be replaced with the number of hairs in the array and `{fluid}` should be replaced with the fluid you are running the simulation in, either air or water. If these directories don't exist, create them before running the Matlab code.

Running Matlab simulations in parallel

Matlab simulations are run on the RM partition in Bridges due to memory constraints. They are run using Matlab's `parloop` parallelization. Each RM node can support 8 simulations at a time, as the code is set up to execute 8 simulations for each job on Bridges.

In order to set these up, you will need to change the run job file. Navigate to the `src/bridges` folder and enter the `runent-sniff.job` file and change the following lines: - Line 13: array should equal the number of hairs in your array - Line 16: fluid should equal the fluid which you wish to simulate. (Be sure to leave the slashes and back ticks alone, they are important the way they are!) - Line 25: change the path to your entcode directory

To submit a job, use the following:

```
$ sh setrunent.sh 1 165
```

Jobs will be submitted to Bridges in sets of 8 per job ID. If you enter a non-multiple of 8, the remainder will be submitted as its own job.

Running final R code to calculate values

First, check to see if all hair concentration files are present in the necessary directory before running the R code:

```
for i in `seq -f "%04g" 1 165`;do if [ ! -f hairs_c_${i}.mat ]; then echo "File ${i} not
found"; fi; done
```

The R script in `src/r-scripts` can be run either on your own machine (after downloading the resulting mat files in the form `hairs_c_*.mat`) or on Bridges. To do so, simply open R and source the script:

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

A csv file will all results will be deposited in `results/r-csv-files/`.