entcode master project – pipeline

Lindsay Waldrop

2023-02-13

Introduction

The entcode master project is a computational fluid dynamic simulation of odor capture by a chemosensory hair array consisting of three hairs and a supporting, non-chemosensory antenna. The simulations are in two dimensions, representing a cross-section of the array, and represent arrays with either 3 or 25 hairs. The project consists of setup for 2,000 individual simulations, each representing a unique combination of the following parameters:

- Angle of array to oncoming flow: 0 to 180 degrees
- Gap-to-hair-diameter ratio of the gaps between chemosensory hairs to their widths: 1 to 50
- Antenna-to-hair-diameter ratio of the diameter of a chemosensory hair to the diameter of the supporting antenna: 1 to 50
- Distance of the center hair from the antenna: 0.001 to 0.02 m
- Reynolds number: 0.1 to 10
- Odor diffusion coefficient: 1e-9 to 1e-4 m^2 s^{-1}
- Width of the initial odor stripe: 0.5 to 5 % of the domain width
- Initial odor concentration: 1e3 to 1e7 (no units)

This simulation and data analysis pipeline for the entcode master 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.
- MATLAB
- Deep neural network simulation set with 2,000 number of simulations establishing parameter values listed in the Intro section.

This pipeline produces the following outputs:

- IBAMR flow data
- Calculated shear around hairs
- Calculated leakiness through each row of the array
- Odor capture for the hair 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 antenna and each hair in the array. 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`. Be sure to change the number of hairs on line 21 (options: 3 or 25)

To run this file, go to main directory and run the file using the the bash console:

```
$ Rscript 'src/r-scripts/generate_grid2d_multirow.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 (options: 3 or 25)
- 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`.

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.

Checking for simulation set completeness

All simulations must be complete, unzipped, and in the appropriate directory 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 simulations as the second, and whether you want to check for folders (1) or zip files (other).

```
$ sh checkruns.sh "$WD" 2000 1
```

This script will return "Simulation found" if it exists. If these conditions are not met, it will return the number of the simulation missing. All simulations need to be unzipped and complete for the VisIt analysis to work properly.

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 domain 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`. In the R console in the main directory, run:

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

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

Or from shell, add the R module and run from the top working directory run:

```
$ Rscript 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" 1 2000
```

After the top working directory, enter the range of simulations you would like analyzed. VisIt will write these values to a .curve file and store them in `results/visit/3hair_runs/simi/hairlinej` where i is the simulation number and j is the row number.

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

```
$ sh setlineout_flux.sh "$WD" 1 2000
```

It will deposit curve files in `results/visit/3hair_runs/simi/hairline_flux`. Four files will be produced for each hair.

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" 2000
```

where the first argument is the top working directory and the second is the number of simulations to check. This script will provide a list of files in the flux, shear, 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.

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

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

And then:

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

or in the command line:

```
$ Rscript src/r-scripts/R_analysis_shear.R
```

Final values will be saved in csv files in `results/r-script-csv/3hair_results/` if the run set is complete.

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 zip files by running the following script located in `src/visit`:

```
$ sh checkruns.sh "$WD" 3 2000 1
```

Put the number of hairs in the array after the working directory. Any missing simulations should be rerun and placed in the correct directories. You can also use option `0` in place of `1` to check for zip files.

Matlab will rely on a standard file structure in order to create and save data. This should be in the format: `$WD/results/odorcapture/3hair_array/`. If this directory doesn't exist, create it before running the Matlab code.

Running Matlab simulations in parallel

Matlab simulations are run individually on the cpu-long.q partition in the Keck cluster due to memory and wall-time constraints. The regular nodes have 144 CPUs with 192043 MB (192 GB) or 257574 MB (275.5 GB) per node. They are run individually, although the code does support using Matlab's `parloop` parallelization for running multiple simulations on a single job. The code is set up to execute 1 simulation for each jobs on the Keck Cluster.

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 14: array should equal the number of hairs in your array

To submit a job, use the following:

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

Jobs will be submitted to the cluster with their own job IDs.

Checking for Matlab simulation files results completeness

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

```
$ sh checkmatlab.sh "$WD" 3 2000
```

where the first argument is the top working directory and the second is the number of hairs in the array, and third is the number of simulations to check. This script will provide a list of Matlab data files with the prefixes `c_`, `hairs_c_`, `initdata_`, and `velocity_` are missing so you can check individual simulations and rerun as necessary.

Running final R code to calculate values

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/`.