The code I intend to present is a lab-brewed simulation of spiral ganglion neuron stimulation by a cochlear implant electrode. My work tends to involve adding additional structural elements to the model and exploring a physiologically relevant parameter space. I then look at how such modifications alter the membrane voltage and spiking of the neurons in response to different stimuli.

The core simulation is written in C and has been gradually built up over the course of decades. My mentor actually wrote the first version of it in Fortran when he was a graduate student in the 80s. As you might imagine, it has some idiosyncrasies from this protracted development process. The inputs to this C simulation are 3 ASCII files containing: model parameter values (a .modl file), optional parameter values (including flags for switches in the program) (a .opts file), and parameters describing the stimulus (a .stim file). The output of the simulation is a binary file containing spike times (an .Spks file), with the option to also output raw voltage data (a .vltg file).

The ANF model consists of a segmented cable with some segments corresponding to nodal segments and the remainder to internodal segments (we do not currently model a cell body). The membrane potential within each of these segments is represented by a partial differential equation relating the currents through the segment in the style of Hodgkin and Huxley. The nodal segments contain multiple species of ion channels who’s gating is modeled by a continuous Markov Process. At each timestep in the simulation (typically 1 µs), the ion channel states of each node are updated, the coefficients of each of the differential equations are calculated using the membrane potential values during the previous step, and the entire system of differential equations is solved using the Crank-Nicolson method. In short, this simulation is pretty computationally expensive and when simulating multiple fibers, over multiple trials, with anything but the simplest stimuli some sort of high performance computing is necessary.

When I started working on the code it had been fitted with a MatLab wrapper that enabled the creation of the Modl, Opts, and Stim files described above to span different parameter spaces (e.g. stimulus intensity, fiber caliber). This wrapper then generates a text file containing a list of commands that call the simulation with the appropriate files from console. Finally, the wrapper called a Python function that interfaced with Hyak and ran this list of commands. Once all the jobs for the experiment are completed, a separate MatLab program reads in the spiking and voltage information saved in all the binary files by the C code. From there, theses data are analyzed locally in MatLab.

I have spent quite a bit of time trying to improve both this wrapper and the C simulation itself. I’ve tried making each piece of the input and output more modular with easy access to parameters that are likely to be altered but with others somewhat hidden. Notably, I’ve separated out the creation of fiber populations so that it’s possible to run many experiments on the same populations. I’ve also migrated the management of commands to a parallel SQL database. This allows my jobs on Hyak to pull commands from this database and have database keeps track of which have been picked-up and completed.

Despite these modifications I still feel there must be a better way to structure the whole thing to make it easier to use. One possibility is to move to calling the C code directly from MatLab, or Python, and passing the parameters in working memory instead of in a file. Then the parallelism could be managed directly with the wrapper. My concern is that I’m inexperienced at dealing with systems this complex and feel like I could use some support in moving this code to a more modern format. This is where I’m hoping to tap the knowledge of the Working Group and perhaps get some support from the eScience institute.