Lab #1 (basic bioelectricity)

How to work as a group

This lab involves two parts – coding/simulating, thinking about the questions, and writing a report. You only need to turn in one report per group.

The goal of having only one written report per group is to save you time in writing and save me time in reading. However, my expectation is that everyone in the group runs the simulations and discusses the questions together. The goal is *not* to have only one person learn the material 😊.

Overview:

In this lab, we'll

- get our first exposure to BITSEY (a friendly bioelectric simulator)
- run simulations of single cells to see what voltages they settle to

BITSEY is a smaller, simpler version of BETSE[1], meant specifically for classroom work. It's open-source Python code that you can find online, but it's easier to just grab it on the Halligan system from the link on the class web page. If you look at the BITSEY code, you will see five files:

- *main_vlab1_SS.py*: the main entry point. All of the functions that you are responsible for writing or modifying will go in main_vlab1_SS.py.
- *sim.py*: a library file containing the main bioelectric simulation routines.
- *edebug.py*: a library file with various debug-printing routines, to aid in figuring out why a simulation isn't giving you the results you want
- *eplot.py*: a library file with several nice routines that help make pretty plots of, e.g., cell voltage over time.
- *sim_toolbox.py*: a library file with some basic physics models (ion channels and pumps)

Instructions:

Please copy all five files to your own work directory, and then open main_vlab1_SS.py to take a look. You will notice that the very end of the file contains a call to the main function *command_line*(). This function first checks the command line to find out which simulation to set up and how long to simulate for. It then calls *sim.sim*() to actually run the simulation. Finally, it prints out the simulation results.

Let's run a simulation. Try **python3 main_vlab1_SS.py lab1 5**. This will call the function *setup_lab1*() to set up a simulation that instantiates four cells, and then simulates for 5 seconds of virtual time. For the moment, all four cells are identical. After a short simulation, it calls *post_lab1*() for any plotting that you like. It is originally set up to plot a graph of V_{mem} in each cell (and, since the four cells are identical, the four plots will all lie on top of each other). Feel free to experiment with changing the simulation time (via the command line) or the plots (by editing *main_vlab1_SS.post_lab1*()). The header of the file *eplot.py* lists some of the other plotting functions that may be useful.

Note the various debug data (such as the per-cell V_{mem}, ion concentrations and various other information) that also gets printed, both during the simulation and at the end. When debug information is printed, then typically each row (if there are multiple rows) is for one ion; each column is the data for one cell. See the documentation for the function *dump*() for more detail (at the top of the file *edump.py*).

Now that you know how to run a short simulation, it's time to do it for real. This time, we will run three simulations, each one for 100K seconds of virtual time. Here are the three simulations:

1. Exactly as you did at first, but now running for the full 100K seconds of virtual time. Since all four cells are identical, hopefully they behave identically – so their graphs will likely overlap.
2. Altered initial concentrations. Leave cell[0] the same (it will be your reference). Double the initial $[Na]_{int}$, $[K]_{int}$ or $[Cl]_{int}$ in cells [1], [2] and [3] respectively (i.e., each of those cells will have one ion concentration doubled). Remember to preserve charge neutrality by altering another of $[Na]_{int}$, $[K]_{int}$ or $[Cl]_{int}$ accordingly (but please do not touch [P]). Simulate for 100K seconds again.
3. Altered density of ion channels. As before, leave cell[0] the same (it will again be your reference). Double D_{Na}, D_K and D_{Cl} in cells [1], [2] and [3] respectively.

What to turn in:

Your version of main_vlab1_SS.py.

A lab report that contains several things. First, your graphs of V_{mem} for all three sims. For simulation #2, also turn in graphs of [Na], [K] and [Cl] across all cells. There's a call to *eplt.plot_ion* () already in the *post_lab1*() function; you should add K and Cl as well.

Finally, turn in the answers to the questions below in your report.

You need only turn in one report for each group.

Questions:

1. At the end of each simulation, the function *dump*() dumps out various information about the system's final state. Note what it says about the flow rates (in moles/m^3·sec) for Na, K and Cl ion channels and pumps (and note that *dump*() skips printout of any zero-valued flow rates). Based on this, is the circuit essentially at steady state at the end of the simulation?

2. Simulation #2 should show that your final steady-state results are insensitive to the cell interior's initial [Na], [K] and [Cl] – let's explore why. You may want to look at the plots of $[Na]_{int}$, $[K]_{int}$ and $[Cl]_{int}$ for cells #1-3 and compare them to cell #0.
   a. Conservation of mass means that it is impossible for Na, K or Cl to appear out of nowhere or to mysteriously vanish. Is it physically realistic for, e.g., cell #1 to start with more $[Na]_{int}$ than cell #0 and wind up with the same $[Na]_{int}$? What assumption have we made that makes this reasonable?
   b. Our upcoming unit on QSS bioelectricity talks about negative feedback. In (e.g.,) cell #1, you increased $[Na]_{int}$ but did not change $V_{mem.}$ How would you expect this to (at least early in the sim) affect diffusion? Drift? Can you argue that the combined effect of the new drift and diffusion tend to have a restorative effect (pulling the cell closer to cell #0's results) or the opposite?

   c. Would the cell-0 final SS solution also be a valid SS solution for cells #1-3? I.e., if cells #1-3 *somehow* eventually reach the same final cell-internal ion concentrations as does cell #0, can you argue that cells #1-3 would then be in steady state?
   d. (harder). After (a)-(c), you have hopefully argued that it is quite reasonable for a cells #1-3 to return to the same point as cell #0. But can you argue that there is no other reasonable result? Use idea #1 below to make that argument.

   If you've made it through all four of these steps, then you've shown that (a) our experimental method is not just plain unbelievable, (b) cells #1-3 have a tendency to move back to what cell #0 does, (c) when cells #1-3 do reach what cell #0 does, it will be SS for them too, and (d) all of this is not only believable, but mathematically correct.

3. Given the final values for $[Cl]_{int}$ and $[Cl]_{ext}$ from simulation #2, compute V^{Nernst} for Cl. Does it roughly agree with your final V_{mem}? Explain why. (Note that the simulation results only show ICF concentrations and do not show ECF concentrations. To find ECF concentrations, look in the file *sim.py* for the function *init_big_arrays*(). You can see it initializing *c_out* for each ion, which is the default ECF concentration for that ion).

4. Simulation #3 should show that D_{Cl} does not affect your final results at all; that increasing D_{Na} makes the final V_{mem} more positive, and that increasing D_K makes the final V_{mem} more negative. Let's try to explain this. Remember that in BITSEY, *D* acts an a stand-in for the ion-channel density.
   a. For each of the three cases (cell #1 and Na channels; cell #2 and K channels; cell #3 and Cl channels), is the net (drift + diffusion) flow through the ion channel inwards, outwards or zero? Based on that, can you argue that your results make sense?
   b. Since Cl does not have a pump, it's especially easy to understand. Consider only cell #0, and assume it has reached steady state. Suddenly you double the conductivity of its Cl channels, effectively morphing cell #0 into cell #3. What happens to the Cl drift current? Cl diffusion current? If they were balanced before, are they still balanced? Can you use this to make another argument for the results you saw in cell #3?
   c. In our equivalent-circuit model, the density of ion channels corresponds to the three parameters G_{Na}, G_K and G_{Cl}. Can you argue that, based solely on circuit analysis of this model, that increasing *D* should give the results that you saw? (In our next unit, we will see that this is more properly applicable to QSS than to SS, but for now we won't worry about the difference). You don't have to go through the detailed circuit analysis to completely show this result, but at least argue intuitively using concepts like Ohm's Law and KVL.

Some ideas to stimulate your thinking if you like:

1. (This may help for question 2d). Let's say you built a set of equations and unknowns to compute the final cell voltage. You might have variables (i.e., unknowns) for the final $[Na]_{int}$, $[K]_{int}$ and $[Cl]_{int}$ and for V_{mem}. You might have one equation that says Q=CV (i.e., once you know [Na], [K] and [Cl] you automatically know V_{mem}). You might have another equation that says the total Na current is 0, and two more equations for K and Cl. You can assume that a system of *N* equations in *N* unknowns

always has one solution (even though this is not strictly true). What does that say about the initial $[Na]_{int}$, $[K]_{int}$ and $[Cl]_{int}$? This

2. V^{Nernst} for Na and K will certainly change as you change D_{Na} and D_K. However, the changes will likely not be drastic. Given that, you can think of the main effect of changing D_{Na} and D_K as being to change the resistors in your model. (This justifies our use of the equivalent model for question 4b).

Summary of what we learned from this lab

Hopefully a few lessons come from this lab:

- The cell develops voltage in a way that's largely determined by its ion-channel conductances. This gives our bodies a nice way to change cell voltages.
- Even though ion concentrations can vary due to many disturbances, the cell robustly reaches whatever voltage the ion channels tell it.
- Simulation is easy and fun 😊

Resources

If you feel like digging a bit deeper, there's a debugging function *edb.analyze_equiv_network* () that looks at the current ion concentrations and builds an equivalent model just like we did in class. You can call it from *post_lab1*(). Does it give you the same V_{nernst} for Na, K and Cl that you would calculate by hand? It should (at least, within a reasonable tolerance)! Does it predict the ion flow rates correctly (i.e., the same as reported by *dump*())? Remember that our linear model is just an approximation.

[1] *Bioelectric gene and reaction networks: computational modeling of genetic, biochemical and bioelectrical dynamics in pattern regulation*, Alexis Pietak 2017