Lab #1 (basic bioelectricity)

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.py*: the main entry point. All of the functions that you are responsible for writing or modifying will go in main.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)

Please copy all five files to your own work directory, and then open main.py to take a look. You will notice that the very end of the file contains a call to the main function *setup_and_sim*(). This function (defined just above) 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.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 then plots out graphs of V_{mem} and of [Na], [K] and [Cl] in each cell. Feel free to experiment with changing the simulation time (via the command line) or the plots (which you control by editing some code in *main.setup_and_sim*().

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.

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. Each simulation may take an hour or more, so you can't really do it during class. Here are the three simulations:

1. Exactly as you did at first, but merely 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.

Turn in your graphs of V_{mem} for all three sims.

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 mV/sec) for Na, K and Cl ion channels and pumps. Any observations?
2. Simulation #2 should show that your final results are insensitive to the cell interior's initial [Na], [K] and [Cl]. Can you explain why this is?
3. Given the final values for $[Cl]_{int}$ and $[Cl]_{ext}$ from simulation #2, compute V_{Nernst} for Cl. Does it agree with your final V_{mem}? Explain why.
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. Can you explain this?

Some ideas to stimulate your thinking:

- 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 change D_{Na} and D_K as being to change the resistors in your model.
- Assume that a cell has reached steady state. Suddenly you double the conductivity of its Cl channels. What happens to the Cl drift current? Cl diffusion current? If they were balanced before, are they still balanced?
- 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. Does this system of equations always have exactly one solution? If so, what does that say about the initial $[Na]_{int}$, $[K]_{int}$ and $[Cl]_{int}$?

Having to wait an hour or more for a simulation is not much fun. Some of the choices for final projects include methods of (hopefully) speeding up the simulation, or even using different numerical techniques to predict the result almost instantly, without simulating at all.

If you feel like digging a bit deeper, there's a 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 at the end of the simulation (in fact, the call is already in *main.setup_and_sim*() but commented out). 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