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?

   The main observation is that the flow rates are consistent with being in steady state. For example, here are the results from simulation #1:
   t=100000.0: dumping long-format...
   Vm = [-49.94 -49.94 -49.94 -49.94]mV
   Na ionCh: [818 818 818 818] mV/s*10^1
   Na pump: [-817 -817 -817 -817] mV/s*10^1
   K ionCh: [-542 -542 -542 -542] mV/s*10^1
   K pump: [545 545 545 545] mV/s*10^1
   You will notice that the Na ion-channel flow rates (-81.7 mV/s) almost exactly balance the Na pump flow rates (81.8 mV/s). The same is true for K (-54.2 vs 54.4 mV/s). Finally, the flow rates for Cl are effectively zero (which is why they are not even printed); this is expected, since there is no Cl pump.

   If you ran your simulation in fast mode, you may have gotten slightly more inaccuracy in the final numbers.

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?
   There is a big-picture question here: given the ECF ion concentrations, and given the ion-channel conductances, is the cell guaranteed to reach a unique final operating point (i.e., a final V_{mem} and final ion concentrations)? If so, then the final simulation results will indeed be insensitive to the cell interior's initial [Na], [K] and [Cl]. But how can we argue that the final values are indeed unique? Might there not be more than one final V_{mem} that is stable, or even none?

   Here's an argument for uniqueness. Consider the following system of N equations and N unknowns. Let there be four unknowns: $[Na]_{final}$, $[K]_{final}$, $[Cl]_{final}$, and V_{mem}. How many equations do we have relating these variables? There are three obvious equations: zero net charge flow for each of Na, K and Cl. For, e.g., Na, we have the drift current (which is a function of [Na] and of V_{mem}), the diffusion current (which is a function of [Na]), and the pump current (which is effectively constant). The sum of these three must equal 0, since the final concentrations must be at steady state. Finally, there is a fourth equation: Q=CV. The charge in the cell is the sum of [Na] + [K] – [Cl] (and any other ions that may be in the cell). The capacitance C is fixed, thus relating V_{mem} to $[Na]_{final}$, $[K]_{final}$ and $[Cl]_{final}$.

   With four equations and four unknowns, the odds are quite good that the system has a unique solution. You will note that the *initial* ion concentrations were not variables in our equations, and indeed did not enter the picture at all. They are thus irrelevant.

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.

The external [Cl], which remains constant, is 140 moles/m^3; the final internal concentration printed by dump() is 22 moles/m^3. We thus get V_{Nernst} = (26 mV) * ln (140/22) ≈ 48mV. Our simulation result, printed by dump(), was 49.9mV. These are reasonably close: remember that the final $[Cl]_{int}$ of 22 moles/m^3 was rounded to the nearest integer. If you ran your simulation in fast mode, you may have gotten slightly more inaccuracy as well.

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?

   Cl does not have an ion pump. At steady state, we must thus have drift and diffusion currents for Cl being equal and opposite. Now consider what happens when we, e.g., double D_{Cl}. If the system remains at the same final state (i.e., the same [Cl] and V_{mem}), then both drift and diffusion double. Both drift and diffusion are from ions flowing through the Cl ion channels; having more ion channels will affect both drift and diffusion equally. If drift and diffusion currents for Cl were originally equal and opposite, then doubling both of them will leave them still being equal and opposite. Conclusion: the previous V_{mem} still works as a solution to the new cell parameters. Thus, changing D_{Cl} does not affect the final results!

   Na and K are a bit harder. The simplest way to think of this is to remember our equivalent-circuit model of a cell. In this model, the final V_{mem} is a weighted sum of $V_{Nernst,Na}$, $V_{Nernst,K}$ and $V_{Nernst,Cl}$, with the ion-channel conductances as the weights. Since $V_{Nernst,Na}$ is positive and $V_{Nernst,K}$ is negative, then increasing D_{Na} will tend to make the final V_{mem} more positive, with the opposite for D_K. Note that the equivalent circuit as we analyzed it is valid at quasi steady state rather than steady state, but still gives reasonable intuition here.

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