

Lab #4 – Cardiac gap junctions

How to work as a group

This lab involves 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 😊.

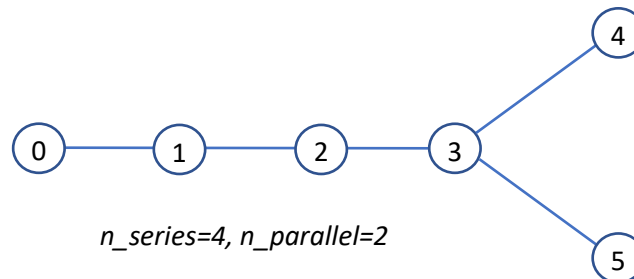
Overview:

In this lab, we will look at how the heart uses gap junctions (GJs) to conduct an action potential from one cardiomyocyte to its neighbors. While most (but nowhere near all) neurons talk to each other via chemical synapses, heart cells communicate electrically, via GJs.

In fact, GJs help the heart perform a delicate balancing act: APs can only travel from one cardiomyocyte to the next if the GJs connecting the two have just the right conductivity: not too high, not too low, but just right. In this lab, we will explore what “just right” means – what happens if there are too few or too many GJs connecting neighboring cells.

Detailed instructions

Start with the file `main_cardiac_GJs.py`, which you can get from the course website. It uses BITSEY to create the following network of cells connected by GJs:



You control your simulation by setting the value of four parameters (near the top of the file):

- n_series and $n_parallel$: the number of series and parallel cells in the network, respectively. The final series cell drives all of the parallel cells.
- G_GJ_ser and G_GJ_par : the conductance of the GJs (G_{GJ}) in the serial and parallel networks, respectively. As usual, these are relative to some standard conductance.

The simulation seeds an AP in cell #0, and then we watch it propagate (or not!) to the remaining cells.

You should run the following simulations until $t=300$ seconds (the action-potential spike starts at $t=200$) with the command line `python main_vlab4_cardiac_GJ.py cardiac_GJ 300`:

n_{series}	$n_{parallel}$	G_{GJ_ser}	G_{GJ_par}
4	0	10	Don't care
4	0	30	Don't care
4	0	120	Don't care
4	0	150	Don't care
4	2	50	10
4	2	50	20
4	2	50	200
4	5	50	10
4	5	50	15
4	5	50	45
4	5	50	100

You should turn in one V_{mem} -vs-time graphs for each of these 11 simulations (zooming around the plot so as to remove the boring part from $t=0$ to $t=200$ would be useful).

Questions

Please answer the following questions.

1. First consider the simulations of just four cells in series. Hopefully you saw that with too low G_{GJ_ser} , the AP could not propagate from one cell to the next. Within a good range of G_{GJ_ser} , it did. And with too-high G_{GJ_ser} , no APs formed at all, even in cell #0. Can you explain this? You should give a mostly intuitive explanation, bolstered by a simple equation or two (think of our usual $\Delta Q = C\Delta V$, $\Delta Q = I\Delta t$, etc., and think about charge transfer from cell to cell through the GJs). Assume that cardiomyocytes, being much shorter than many neurons, do not have substantial internal resistance along their length (in fact, BITSEY assumes that all cells have zero internal resistance). You can model each GJ as a simple resistor connecting two cells (with conductance proportional to the appropriate G_{GJ}).
2. Now consider the next three cases, with 4 series cells and the final series cell (cell #3) driving two parallel cells. These simulations all have G_{GJ_ser} of the series chain set to a nice just-right value to propagate APs, and then simulated a range of G_{GJ_par} between cell #3 and all the parallel cells. Hopefully you saw that with the conductance from cell #3 to the remaining cells too low, the AP did not propagate after cell #3 at all; with a just-right G_{GJ_par} , it propagated fine. And with G_{GJ_par} too large, while the AP did propagate, the parallel cells all have their AP roughly superimposed on cell 3. Can you explain this behavior? And why didn't the large G_{GJ_par} completely break the AP, like you saw in the first sims? You can use the same concepts as in the first problem, and now perhaps think a bit about RC time constants.
3. Finally, consider the sims with 5 parallel cells. You should have seen similar results to the 2-parallel-cells simulations – except that now setting G_{GJ_par} too high will crush the

AP in cell #3. Any explanation why? I.e., why do the extra parallel cells change the circuit's behavior?

4. A heart attack, by definition, is when the blood flow to a portion of your heart muscle is blocked. Often, a consequence of this is that parts of your myocardium die (and myocardium essentially never regrows). When this happens, the dead cells can no longer drive their downstream cells, and those downstream cells now (hopefully) get driven by other nearby (and still living) cells. Given what we've learned in this lab so far, can you make an argument for why heart attacks often lead to cardiac arrest?

Summary of what we learned from this lab

Hopefully a few lessons come from this lab:

- Our simple model of a neuron and of GJ connectivity can explain how APs propagate from one cardiomyocyte to another in the human heart.
- Simple analysis of charge transfer can explain why and when the system fails; analysis of RC time constants can explain the cell-to-cell delays.
- Our model also predicts that the heart only works if GJ conductance is set somewhere in a “reasonable” range. This is widely accepted as correct – but surprisingly, the mechanisms by which it happens are still unknown.

Bonus question (extra credit): consider again the simple series circuit of four cells. For the two cases that conduct good APs, read off the net delay from the peak of the AP in cell #0 to the peak of the AP in cell #3. Hopefully you saw a longer delay for the $G_{GJ_ser}=30$ case than for $G_{GJ_ser}=120$. Explain why this is; talk about self-triggering circuits connected by RC delay lines.

What to turn in:

You should turn in one file: your report with the graphs and the answers to the questions.