Class: LSU CSC 2262, Spring
Instructor: James Ghawaly

Group ID: 8

Group Team Members: Gayoon Nam, Kristen Averett, Cassidy McDonald, Lora Elliott, Lynn Nguyen

Perception of Brain Dynamics: Computational Assessment and Implementation

Introduction

For this project, we modeled several types of neurons in the human brain using numerical methods provided by Python. We analyzed each model and how they reacted to multiple stimuli, then examined the error due to numerical approximations. Through this process, we had a deeper understanding of how to navigate programming scientific problems.

By programming this project, we can integrate data science with computer science which expands the scope of what we can do with our skillset. By using Python and numerical methods, we bridged the gap between theoretical neuroscience and computational implementation. In addition, we can gain insight on brain function and behavior. By analyzing the various behaviors of neurons paired with varying conditions, we were able to grasp the logistics of neural dynamics and how they influence the cognitive system.

Furthermore, by completing this project, we are provided with a realistic introspection to projects computer scientists are tasked with daily. From understanding the complexity of the basis of the task to navigating difficult algorithms, we found our way through the obstacles that mirror those of professionals in our field. We were also able to get a good representation of how error can skew the accuracy of computational results and how to effectively reduce error, specifically for Euler's method.

In summary, our project and report represent the comprehension of the convergence of data science and programming. It should reflect our prospects as aspiring programmers and scientists as well as our apprehensions surrounding the work.

Methods

Our Python program implements various third-party libraries such as NumPy, Matplotlib, TQDM. We also utilized the built in Argparse library as well as JSON file provided to us by the instructor. Using such libraries, we simulated a single LIF neuron model and alpha synapse.

Firstly, the JSON file that we received contained a variety of constants that we used to incorporate into the methods we have been tasked to construct. Another constraint that we were required to abide by is the list of command line Argparse arguments that we were instructed to use. To run our final program, the parameters used from the Argparse library request a series of inputs that, when entered, create a customized simulation based upon the inputs given. One of the two required inputs was a string that specified whether the graph was simulating a neuron excited by a series of input spikes (“spike” mode) or a neuron excited by a constant input current (“current” mode). The second required input is a float representing the number of milliseconds required to run the simulation. The secondary inputs are the spike rate and the current; based upon whichever mode is chosen, the user either inputs an int representing the spike rate measured in Hz, or a float representing the input current measured in nanoamps (nA).

As for implementing Matplotlib, we first stored our times and voltages into separate lists. After putting that information into our lists, the results were plugged into Matplotlib and mapped out on a graph. Using Matplotlib, we plotted both the membrane potential track and the voltage track. Both graphs that can be produced measure the membrane potential of a neuron over the course of a set number of milliseconds.

For current mode, our implementation used Euler’s method on the provided equation 1 with the constant Isyn value. Once the current met the threshold and enough time had elapsed, we trigger a spike that went to the v_spike value. After the spike, the current value is reset and the spike time is recorded. This continues until the end of the simulation time. Spike mode was quite similar, except for Isyn needing to be calculated separately and not staying constant. Isyn was calculated using the provided equation 3, which used the spike rate given by the user, along with the current time and time of the last spike. Once Isyn is calculated, we use the same process as current mode, by plugging in our values into equation 1, triggering spikes as necessary, and resetting the values after spikes.

As we were programming the simulation, we encountered issues regarding logical errors. The issues were showing up in our spike and current modes, as they were not displaying correctly. Due to this, the output graph that was produced was inaccurate. There was also an issue with our function labeled “LIF_model_eulers_method” that was set to calculate and return the membrane voltage for each time the function was called and looped, further complicating the problems that we were already encountering in our debugging process.

In our efforts to fix the problems within our code, we used combinations of different problem-solving techniques. All of us met up and discussed what we could do to make our code run properly and by combining our unique perspectives and diverse expertise, we created a multifaceted plan. We did the first few calculations by hand to compare our output and see if our graphs were reflective of what the first couple of values should be. One of the main ways we used was to change the isolated variables in our code and run it to see precisely what was causing the issues. Since the program was producing an output but was not producing a correct output, we knew the error was logical. We iteratively tested multiple scenarios and edge cases during this process.

After reading the information about pre-calculating the spikes and going to office hours, we were able to determine two issues that were causing our code to not produce the intended results. First, we needed to calculate exactly where the spikes happen before they occur. The second issue was that our step size in the JSON file was not properly getting converted. Because of this, we had a step size that was too large and caused large errors in our Euler’s method results. After remedying those issues, we were able to make a program that successfully models spikes in a neuron and plots it appropriately.

With a working program for running the spike mode experiments, we were able to implement methods to use a 10th order Taylor Series approximation of the function e^x, replacing the exponential term in our implementation of the alpha synapse model equation to successfully run the last experiment. This was done by adding a function that approximates the value of e^x and creating another function that replaces the exponential term np.exp in calculating the Isyn value with our new function e(x). To toggle the use of the Taylor Series approximation, the argument --bonus 1 must be added to the end of the primary arguments.

For how the program is run, when the user runs the program, they must be in the correct directory in the terminal to have access to the JSON and Python file. Once you are in the correct directory, input *python file_name.py Argparse parameters* for the program to create a customized graph based upon the input provided. If you were to run the current program, you will receive a graph that produces a series of nearly straight spikes representing the constant flow of current that excites the neuron. If you were to run the spike program, you would instead see how the neuron gets excited and then too excited which leads to the "spike". Unlike current mode's constant Isyn value, spike mode requires us to calculate the Isyn value based on the number of spikes per minute expected.

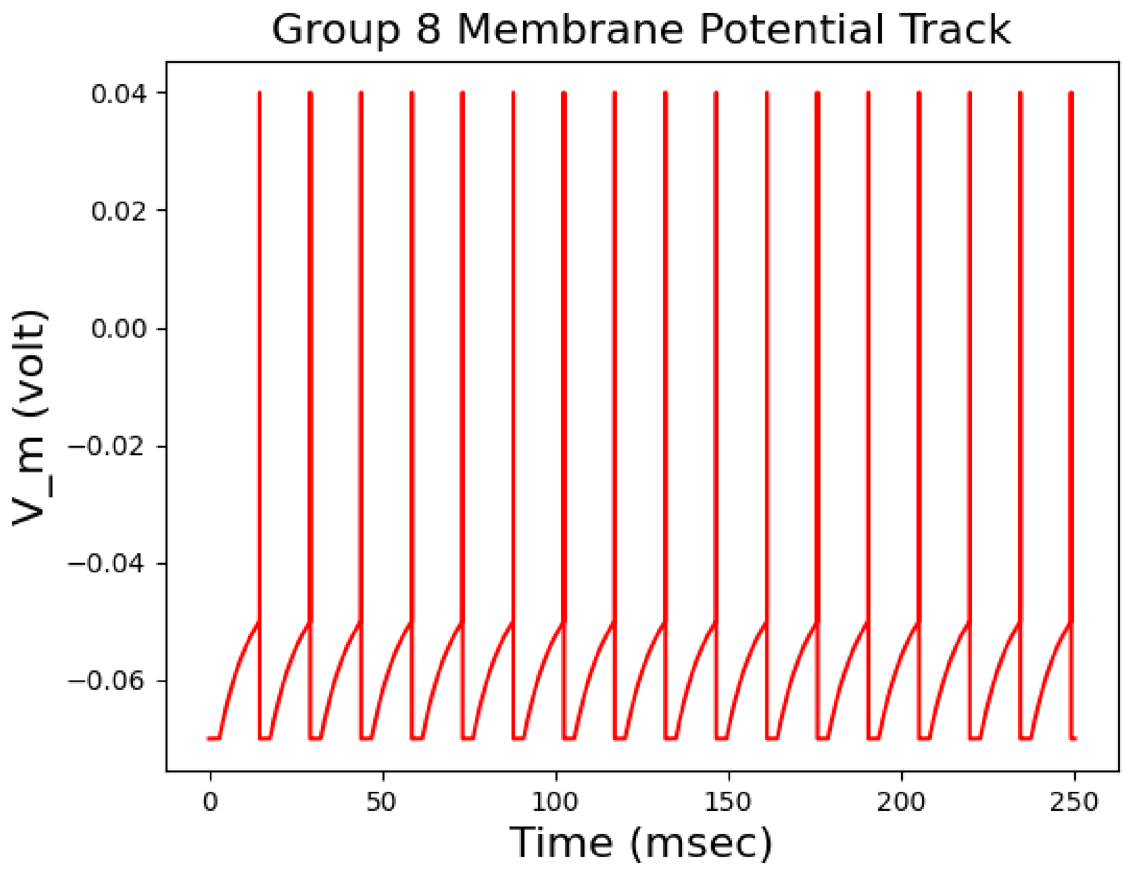
Results

For experiments 1, 2, and 3, the following parameter values should be used in your configuration.		
$V_r = -70$ mV	$V_{thr} = -50$ mV	$V_{spike} = 40$ mV
$V_{rev} = 0$ mV	$\tau_m = 9.37$ ms	$\tau_{syn} = 0.3$ ms
$C_m = 1$ pF	$\bar{g} = 100$ nS	$t_r = 3$ ms
$w = 1$	For Euler's Method: $\Delta t = 0.001$ ms	

Experiment 1 Run the following command:

python neuro_sim.py current 250 --current 0.003

Q1: Show the plot of the membrane voltage, V_m, for the duration of the experiment (250 s).



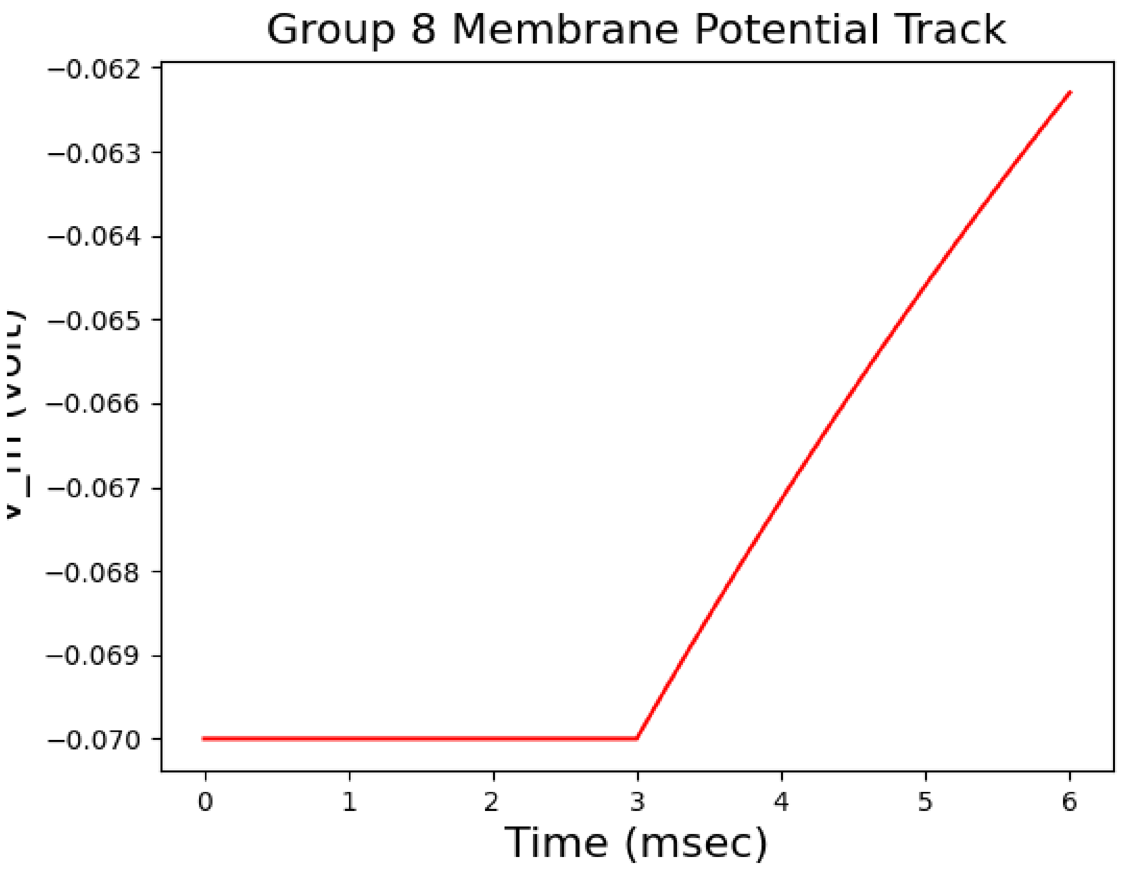
Q2: How many spikes were produced over the course of the experiment?

----> 17 spikes

Experiment 2 Run the following command:

python neuro_sim.py current 6 --current 0.003

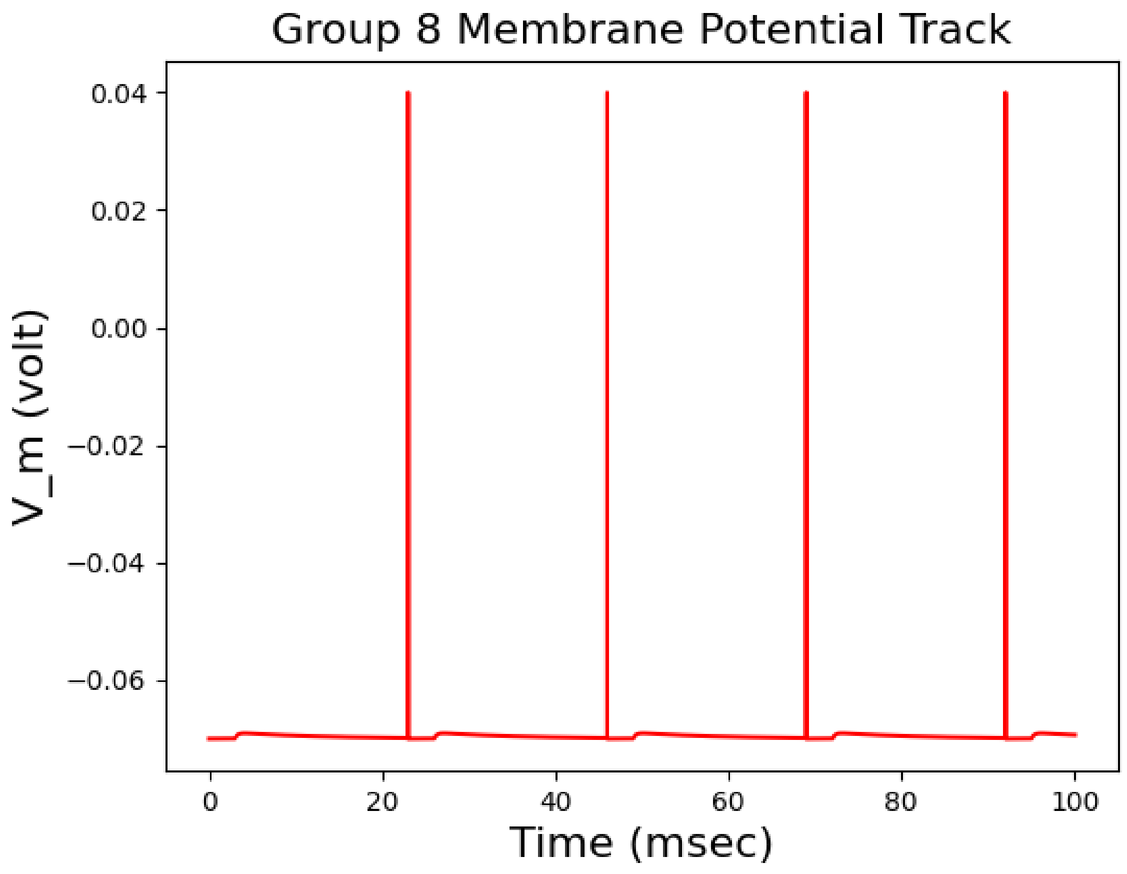
Q1: Show the plot of the membrane voltage, V_m, for the duration of the experiment (6 s).



Experiment 3 Run the following command:

python neuro_sim.py spike 100 --spike_rate 50

Q1: Show the plot of the membrane voltage, V_m, for the duration of the experiment (100 s).



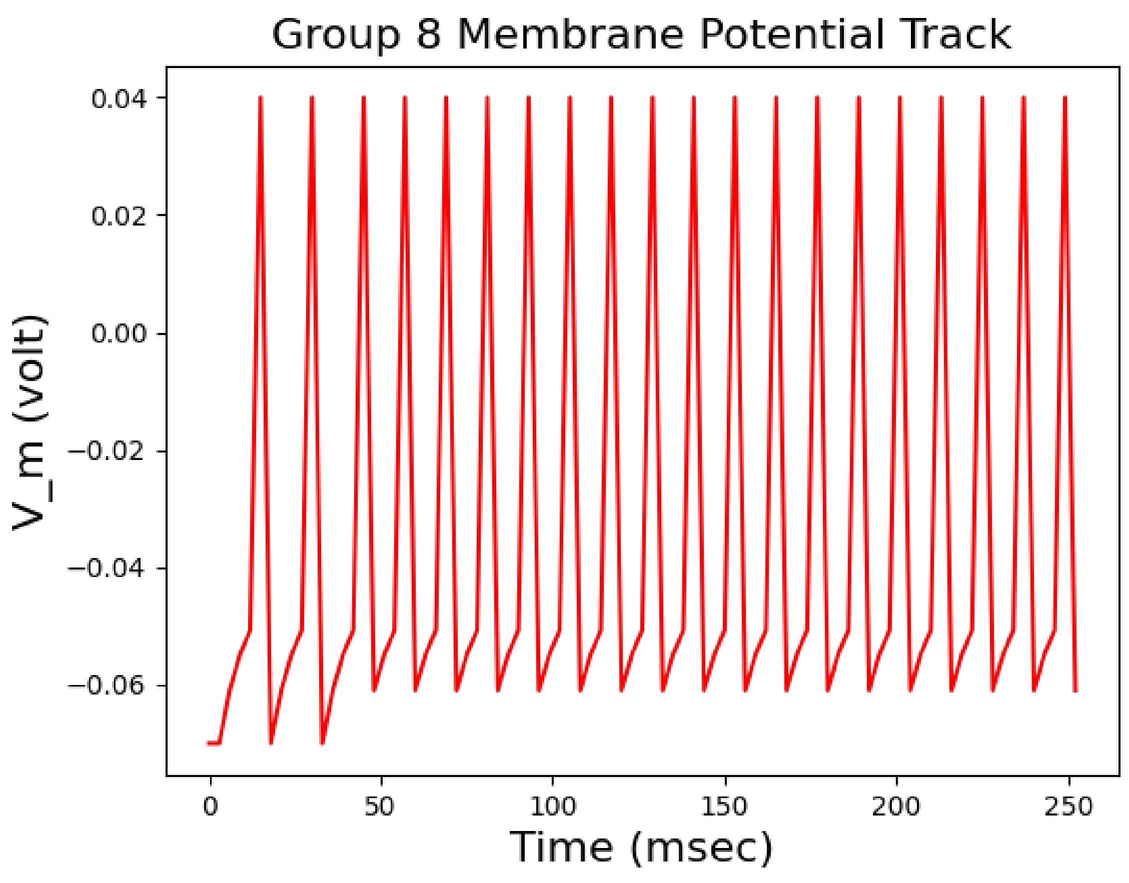
Q2: How many spikes were produced over the course of the experiment?

------> 4 spikes

Experiment 4 Run the following command, <u>but change the Euler's method time step to $\Delta t = 3$ ms:</u>

```
python neuro_sim.py current 250 --current 0.003
```

Q1: Show the plot of the membrane voltage, V_m, for the duration of the experiment (100 s).



Q2: How many spikes were produced over the course of the experiment?

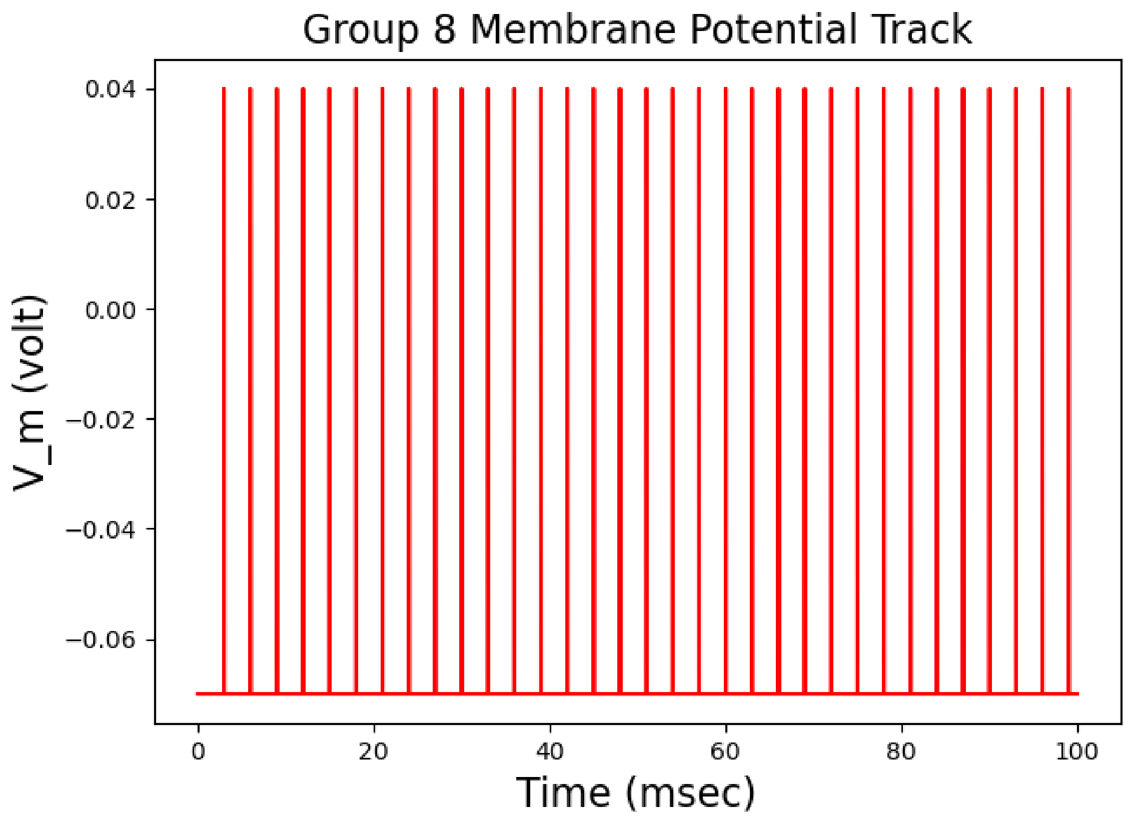
-------> **20 spikes**

Q3: How does this compare to the result from Experiment 1?

There are more spikes than in Experiment 1 and the spikes are strange looking. In the graph, the first two graphs look somewhat normal, but the remaining are floating. This is probably due to the much larger deltaT. A larger step size increases the error and could produce this result.

Experiment 5: Repeat experiment 3, but replace the exponential term in Equation 3, , with a 10th order Taylor Series approximation of , centered at 0.

Q1: Show the plot of the membrane voltage, , for the duration of the experiment (100 s).



Q2: How many spikes were produced over the course of the experiment?

-------> 33 spikes

Q3: How does this compare to your result with Experiment 3?

Experiment 3 only had 4 spikes in the span of 100msec, in comparison to 33 in 100 msec.

Conclusion

The primary goal of our simulation was to learn the intricacies of combining computer science with real world scientific computations. By learning the basics of simulating an artificial neural network (ANN), we are better equipped to integrate our programming skills in areas that will be important in our future careers. Along with learning how to program a neurological simulation, we also had to learn how to use our skill set to debug any potential problems we faced throughout the process. An important skill in any job is the ability to work well in a team. As this was a collaborative project, we were able to navigate any issues by communicating and divvying up work. Our outcome was a program that successfully simulates energy spikes in a neuron under a constant current and under differing currents where we know the spikes per minute. Our program plots the results in an easy-to-understand visual representation. This project also clearly demonstrates the practical usage of Euler’s method and how larger step sizes can result in error that really affects the results.