Assignment: LSU Spring CSC2262 Final Project
Instructor: James Ghawaly
Due Date: April 30, 2024

Objective

In this project, you will be using numerical methods with Python to simulate a variety of different models of neurons in the human brain. You will explore these models to see how they respond to a variety of different stimuli, analyze the error of the simulation resulting from numerical approximations, and gain insight into programming scientific problems.

Groups and Grading

Groups: You can work in groups of 1 to 5 people. Groups larger than 5 will not be allowed under any circumstances. All members of the team are expected to contribute equally to the project and participate in writing the code and report. **You must form your group by April 11**. To do so, **one person** from your group should email me with a list of all group member's names by **April 11**. If you are not in a group by that time, then I will assume that you are not participating in the project. After the groups are formed, I will assign your group a "Group ID", which you will need for turning in the project.

Grading: The grade will be determined as follows: 70% correctness of simulation and code and 30% for the quality/clarity of the report. At the end of the project, there will be a peer assessment. If it comes to my attention that a student did not fairly contribute to the project based on peer assessments, that student may receive a lower grade.

Background Information

When tackling a scientific programming project, as a computer scientist, it is important to try to familiarize yourself with the underlying scientific theory before working on the code. In this project, we will be looking at biological models of neurons in the human brain. Whereas transistors are the underlying unit of computation in a traditional computer, the fundamental computational unit in the human brain is the **neuron**. The brain contains billions of these neurons, each with unique characteristics and each being able to learn and adapt to their environment based on internal and external stimuli. These neurons connect to other neurons to produce what is called a **neural network**. The connection between two neurons is called the **synapse**, with the sending neuron referred to as the **presynaptic** neuron and the receiving neuron referred to as the **postsynaptic** neuron. Information only travels in one direction through a synapse – a postsynaptic neuron cannot send information to a presynaptic neuron without creating a new synapse between the two (which would be called a **recurrent connection**). A single neuron can have thousands of presynaptic neurons that are connected to it, with each connection being a unique synapse. When our brains learn, the strength (**weight**) of these synapses changes. This is called **plasticity**. The weight of a synapse is a scalar and is described in more detail below.

Artificial Neurons:

The field of **artificial intelligence (AI)**, particularly **artificial neural networks (ANNs),** is built upon a very rudimentary model of neurons, which we will refer to as continuous output neuron (CON) models. Figure 1 below shows a basic diagram of how a CON model neuron operates. Each neuron has an input, it processes that input, and produces an output signal that gets sent to other neurons that it is connected to. Figure 1 shows a high-level diagram of a simple neural network containing just two CON neurons with a synapse between them. In this diagram, neuron 1 has a *total* input signal (summed over all inputs), x_1, it then processes that input through what is called an activation function, $\varphi(x)$, the output of this function, y_1, then becomes that output signal of neuron 1. Neuron 1 is connected to neuron 2 through a synapse with a weight value, w. The input signal to neuron 2 is then equal to the output of neuron 1 multiplied by the weight value, $x_2 = w * y_1$. In modern AI, the activation function of a neuron must by continuously differentiable in order to be used with current gradient-based training methods.

Note: If other neurons were also connected to neuron 2, for example, if n neurons were connected to neuron 2, then $x_2 = \sum_{i=1}^{n} w_{i,2} * y_i$. In other words, the input to neuron 2 would be the sum of the outputs from all its presynaptic neurons each multiplied by the respective weight values for each synapse. In this project, you will not need to deal with more than one input neuron.

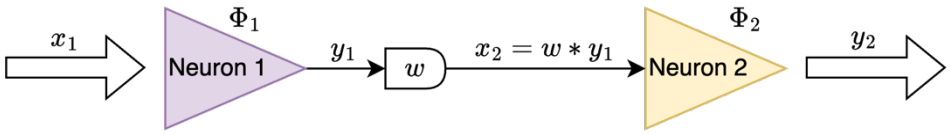


Figure 1: Simple diagram of a neural network containing only 2 neurons

Biological Neurons:

In contrast to the CON model, biological neurons are not characterized by continuously differentiable and temporally invariant activation functions, instead, they are nonlinear spatiotemporally dynamic systems with binary communication. In other words, the output of a biological neuron does not just depend on the input from other neurons, x, but also on time, t. Input signals to a neuron come in the form of a time-varying electric current pulse. This current pulse increases the voltage of the neuron (called the membrane potential) and when it increases above a certain threshold, the neuron dumps all of its charge as an electrical output current signal (called a **spike**) that travels to other neurons. In addition, the neuron's voltage **leaks** over time.

A Simplified Analogy: Image each neuron as a bucket with a small leak at the bottom, signifying the neuron's natural tendency to lose charge over time. Pipes of various sizes connect these buckets, symbolizing the synaptic connections between neurons. The diameter of each pipe

represents the synaptic weight, with wider pipes allowing more water (signal strength) to flow into a bucket (neuron) and narrower pipes allowing less.

As water flows from one bucket into another through these pipes, it mirrors the transmission of electrical signals from one neuron to another. The volume of water a bucket receives through these pipes is determined by the diameters of the connecting pipes, analogous to how a neuron's incoming signal strength is determined by the synaptic weights of its connections.

Once a bucket is filled to a certain level—reaching its threshold—it tips over, emulating a neuron firing an action potential. This action spills its contents, akin to the neuron transmitting a signal to other neurons. The spilled water then flows into connected buckets through their respective pipes, potentially causing them to tip over as well if they reach their threshold.

This analogy captures the essence of neuronal communication and signal integration within a network, highlighting how different synaptic weights influence the strength and propagation of signals through the system, leading to the sequential firing of neurons based on the integrated input they receive.

Leaky Integrate and Fire Neuron Model:

Dozens of mathematical models have been developed to simulate biological neurons, with varying levels of biological realism and computational complexity. Typically, the more biologically realistic the model, the more complex it is to simulate on a computer system. Researching these systems is of keen interest to computational neuroscientists as well as computer science researchers focused on neuromorphic computing. To effectively study networks that consist of thousands to billions of neurons, it is essential to develop numerical software that is both accurate and efficient for simulation purposes. Without numerical methods, simulating these systems is essentially impossible.

In this project, you will look at a neuron model called the leaky integrate and fire (LIF) neuron. At a high-level, the LIF neuron is like a resistor capacitor (RC) circuit, which you may have seen in introductory physics or circuits courses. The membrane voltage, V_m, of a LIF neuron model is described by a first order differential equation (ODE), given by Equation 1. In this equation, the variables are as follows:

- t is time. Units are seconds (s).
- V_r is a constant called the resting potential of the neuron. In other words, it is the voltage of the neuron when the neuron is at “rest”. Units are volts (V).
- I_{syn} is the total input current to the neuron from all presynaptic neurons. Units are amps (A).
- C_m is a constant called the membrane capacitance. It is the capacitance of the neuron, which determines how much charge it can hold. A neuron with a higher C_m will require more input current to raise its membrane potential. Using our analogy, C_m can be thought of as the size of the bucket. Units are farads (F).
- t_s is the time of the last spike. Units are seconds (s)

- t_r is a constant called the refractory period. It is the amount of time after the neuron spikes that the neuron is inactive and cannot be excited (increase voltage). Units are seconds (s).
- τ_m is the decay time constant of the neuron. It describes how fast the neuron leaks charge. The higher the decay time constant of the neuron, the slower the rate of leakage from the neuron. Units are seconds (s).

Equation 1: $$\frac{dV_m}{dt} = \left(-\frac{V_m - V_r}{\tau_m} + \frac{I_{syn}}{C_m}\right)\delta(t - t_s - t_r)$$

In Equation 1, $\delta(t - t_s - t_r)$ is the Heaviside step function and is a simple piecewise function whose output is either 0 or 1. This is shown in Equation 2. Likewise, if V_m increases above the spike threshold, V_{thr}, such that $V_m > V_{thr}$ at time t, then the neuron will create a spike at time t. The neuron will then reset, that is, V_m is set to the value of V_r and t_s is set to the value of t at the time of the spike. For the purposes of visualization, we can set the voltage of the spike to some constant value called V_{spike}, however in reality, its value does not have any significance.

Equation 2: $$\delta(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

The final expression that we need to describe is that for I_{syn}. The spike traveling across a synapse does not instantaneously deposit charge into the postsynaptic neurons, instead, it takes the form of an alpha function. At a high level, when a neuron spikes, the current starts at 0, ramps up fairly quickly to a maximum value, and then more slowly decays back to 0. As such, it takes a bit of time for a spike to fully deposit all of its charge into a postsynaptic neuron. Equation 3 describes this model of a spike's synaptic current, called the **alpha synapse** model. In this equation, the variables are defined and interpreted as follows:

- V_{rev} is a constant called the reversal potential. Units are volts (V).
- w is a constant (for this project) and is the weight of the synapse, having a value between 0 and 1. Unitless.
- V_m is the membrane voltage of the postsynaptic neuron. Units are volts (V)
- $\bar{g}$ is a constant and is the maximum conductance (inverse of resistance) of the synapse. Units are Siemens (S)
- t_0 is the time that the spike started and t is the current time of the simulation. Units are seconds (s).
- τ_{syn} is the decay time constant of the alpha synapse. Units are seconds (s).

Equation 3: $$I_{syn} = w\bar{g}(V_{rev} - V_m)\frac{t - t_0}{\tau_{syn}}\exp\left(\frac{-(t - t_0)}{\tau_{syn}}\right)$$

Assignment Guidelines & Instructions

You will simulate a single LIF neuron model and alpha synapse model using Python 3 and numerical methods. As previously states, Equation 1 is a first-order ODE. It can be solved using Euler's method with a small but finite temporal step size, Δt. The list below outlines requirements and guidelines for your code:

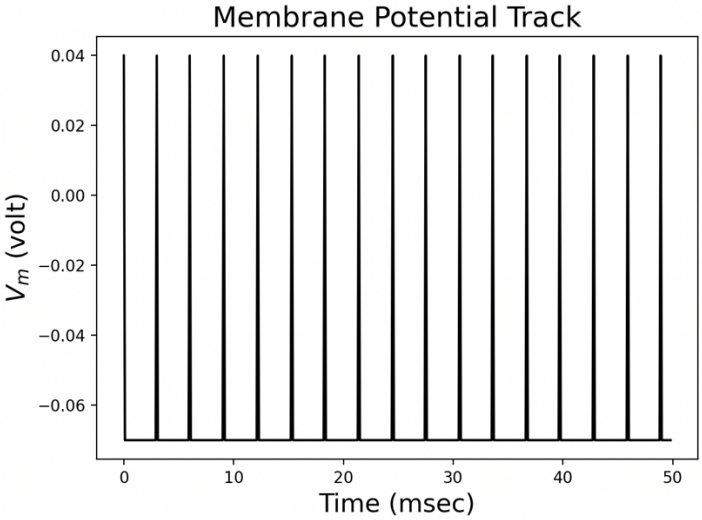
1. Your code should be written in Python 3.10.
2. The only 3rd party libraries you can use are numpy, matplotlib, and tqdm.
3. Your software must have at least one Python file name "neuro_sim.py" that is the main file to be executed. You can have as many other Python files as you would like, to organize your code.
4. Your code should have a Json configuration file named "config.json" in the same directory as neuro_sim.py, that allows you to modify the following constants: V_r, V_{rev}, V_{thr}, V_{spike}, C_m, τ_m, t_r, τ_{syn}, g, and w. I have provided a template Json file for you to use on Moodle. You will need to fill in the values based on the assignment instructions below.
5. Your code will use the built-in argparse library to handle command line arguments. The command line should accept the following inputs:
    a. mode (-m, required, str): Can be either "spike" or "current". The "spike" mode will simulate a neuron being excited by a series of input spikes. The "current" mode will simulate a neuron being excited by a constant input current.
    b. sim_time (-s, required, float): Amount of time to run the simulation for in units of milliseconds.
    c. spike_rate (--spike_rate, only required for "spike" mode, int): Allows the user to specify the input spike rate in units of Hz (30Hz would be 30 spikes per second). Only used for "spike" mode
    d. current (--current, only required for "current" mode, float): Allows the user to specify the input current for the "current" mode. Units should be in nanoamps (nA).
6. In both modes, the program should produce a plot of the membrane voltage, V_m, on the y-axis and time on the x-axis. Spikes should be plotted with the value of V_{spike}. Axes must have labels and units. Plot should have a title with your group's ID.

Examples

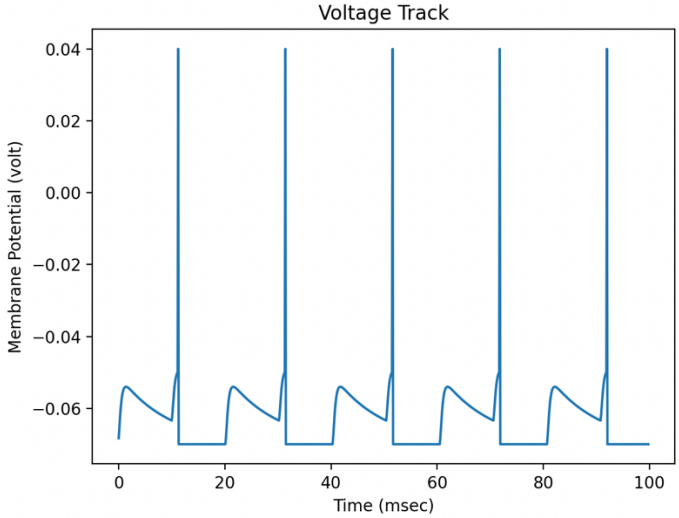
The following are some example command line calls to the program with expected outputs that you can use to test your program, including the expected plot that should be produced.. The following parameters (config) were used for these tests:

$V_r = -70$ 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: $\Delta t = 0.1$ ms	

Example 1: "python neuro_sim.py current 50 --current 1" should run a 50 millisecond long simulation where a single LIF neuron is exposed to a constant input current, I_{syn}, of 1 nA. The following plot would be produced, which shows a total of 17 spikes.



Example 2: For this example, change $w = 0.01$:
"python neuro_sim.py spike 100 --spike_rate 100" should run a 100 millisecond long simulation where a single LIF neuron is exposed to input spikes at a rate of 100 spikes per second. Each input spike should be modeled as an alpha synapse. Assuming the first spike starts at time 0, the following plot would be produced, which shows a total of 5 spikes.



Assignment Report

A template report has been supplied to you on Moodle. All sections must be filled out. The report should total between 3 to 5 pages. This template report also includes several commands that you must run with your code and display the plots. I have also included them here below with addition details:

For all experiments, you should produce the plot of the membrane voltage over time for the full duration of the experiment. This plot should be included in your report. You should also answer questions based on the plot, which are provided in the report template.

For experiments 1, 2, and 3, the following parameters 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 Command: *python neuro_sim.py current 250 --current 0.003*

Experiment 2 Command: *python neuro_sim.py current 6 --current 0.003*

Experiment 3 Command: *python neuro_sim.py spike 100 --spike_rate 50*

For experiment 4, you should change $\Delta t = 3$ ms

Experiment 4 Command: *python neuro_sim.py current 250 --current 0.003*

Experiment 5: Repeat experiment 3, but replace the exponential term in Equation 3, $\exp\left(\frac{-(t-t_0)}{\tau_{syn}}\right)$, with a 10th order Taylor Series approximation of e^x, centered at 0.