Modeling and simulation

Part 1 report

Sara Aljamal

Table of Contents

[**Part 1** 3](#_Toc170072897)

[**1.1** Describe and investigate the single-neuron modeling principles 3](#_Toc170072898)

[**1.2** Investigate the benefits of applying modeling and simulation for simulating the neuronal dynamics 3](#_Toc170072899)

[**1.3** Simulate each equation using Python and plot the results for each equation (Code) 4](#_Toc170072900)

[**1.4** Explain the theoretical principles of each equation and the effect of varying the differential equation's parameters (First order). 4](#_Toc170072901)

[**1.6** Explain the theoretical principles of the second-order system and the effect of varying the differential equation's parameters 8](#_Toc170072902)

[**1.7** Use empirical data to tune the parameters of a simulation model (Code) 9](#_Toc170072903)

[**1.8** Apply different optimization techniques to tune simulation parameters (Code) 9](#_Toc170072904)

[**Part 2** 9](#_Toc170072905)

[**2.1** Design a detailed workflow for solving a specific problem using modeling and simulation 9](#_Toc170072906)

[**2.2** Evaluate the performance of the second-order system in achieving desired behavior 9](#_Toc170072907)

[**2.3** Analyze the effectiveness of each optimization technique for tuning the parameters of the simulation using empirical data 10](#_Toc170072908)

[**Part 3** 11](#_Toc170072909)

[**3.1** Investigate previous methods and work done in the field of neuronal modeling 11](#_Toc170072910)

[**3.2** Critically analyze and compare the performance of different models used for neuronal modeling and simulation 12](#_Toc170072911)

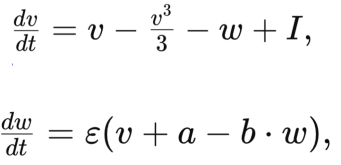
[**References** 14](#_Toc170072912)

# **Part 1**

## **1.1** Describe and investigate the single-neuron modeling principles

A neuron is the basic building block of the nervous system, which allows humans to feel, understand, and interact with things around them. Understanding how neurons work can be described as a fundamental step in neuroscience, as understanding how one neuron functions enable us to discover how a network of neurons function. Single neuron modeling aims to mimic how the neuron works and transmit information by modeling the input and output of a neuron.

There is a lot of modern and optimized models for modeling a single neuron, but the history goes back to 1950s, when Hodgkin and Huxley came up with a model that revealed key features of the ionic conductance underlying the nerve action potential, and they awarded the Nobel prize for this great achievement. Since then, there has been a noticeable effort by researchers to develop more models for modeling a single neuron, the models varied to be simpler versions from the complex model developed by Hodgkin and Huxley to even more complex ones. Nagumo model, which was developed by Richard Fitzhugh in 1961, which is a simplified version of Hodgkin Huxley model as it contains less parameters, the model aims to describe the fundamental features of excitability through the two equations shown in figure 1 where v represents the output voltage, w represents a recovery variable, and I which represents the input current. [1][2]

Figure 1: UML diagram for the MLP

## **1.2** Investigate the benefits of applying modeling and simulation for simulating the neuronal dynamics

Modeling and simulating how a single neuron works derives us to conclusions and results that aid neuroscientists in making more discoveries and unveiling mysteries of the brain and its activity, as well as enabling them to explore more areas in the field. These models can be considered as a step forward in the journey of understanding information processing and transmission in large neuronal networks. Furthermore, single-neuron modeling reveals accurate and reliable information about neurons or even sub-neuron-level electrophysiology. [3][4]

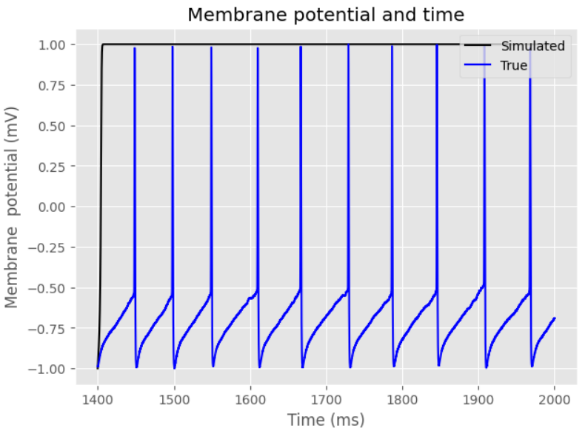
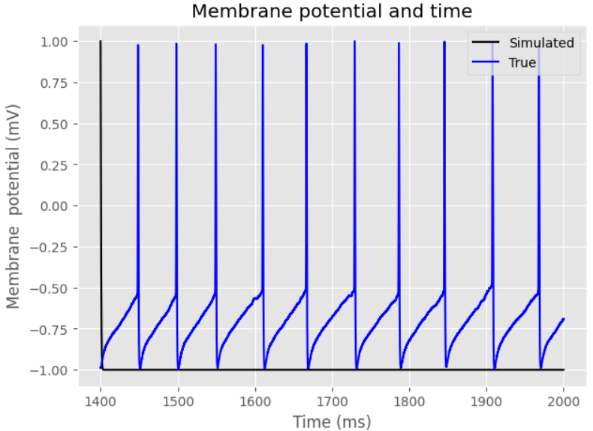
## **1.3** Simulate each equation using Python and plot the results for each equation (Code)

## **1.4** Explain the theoretical principles of each equation and the effect of varying the differential equation's parameters (First order).

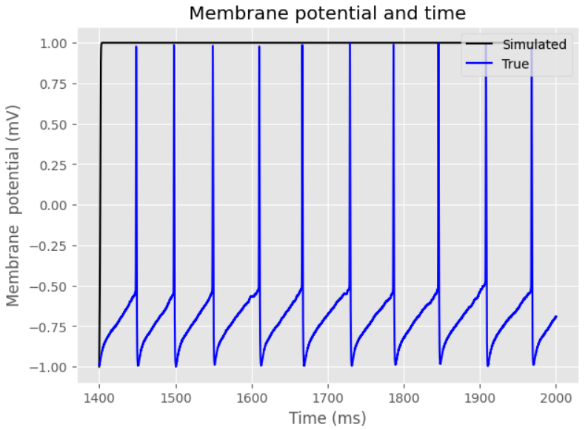
The fist equation shown in figure 1 simulates the change in the membrane potential (Change in the voltage) relying on the voltage (v), the recovery variable (w), and the input which is the current (I), through the fist equation, we can monitor how the neuron gets stimulated when changing the parameters such as:

* The input current (I): It represents the stimulus current, which is the input that excites the neuron, large (I) values indicate higher stimulus, which may result in an action potential for that neuron, while lower values usually indicate there is no excitation or stimulation.
* The recovery variable (w): It’s the variable that helps the neuron return to its resting mode after an action potential. When the neuron is in the resting mode, (w) will have lower values, and when it gets stimulated, (w) values will be higher to be able to bring the neuron back to the resting state.
* The voltage (v): Indicates the electrical state of the neuron, when it’s low, then the neuron in the rest mode or state, and higher values indicates action potential, when (v) gets higher (an action potential occurs), (w) will work to make the (v) value lower which will return the neuron to its resting mode, that’s why (w) is a parameter in the dv/dt equation as it values affects (v) values. (v) is the variable that we are monitoring it’s change, so the previous value will definitely affect the next value for it, so the new value for the voltage= the previous one + the change in the voltage which is we are trying to calculate with respect to the time in millisecond (dv/dt).

In my project, I simulated this equation to examine if I can get the same results as the true action potential in the actual data we have, to do so, I tried to manipulate the values of 9w) and (I) to get better results, while (v) values where calculated each iteration through adding the change to the previous (v) value from the iteration before. In order to better understand the effect of each of (w) and (I) parameters, I changed each parameter twice while holding the other parameter still and setting the initial value for v as -1. Figure 2 shows the simulated voltage values in the black line where the (w) was set to -0.5 and (I) to 0.3, while figure 3 shows the simulated (v) values for the same (I) values but a positive (w) value, figure 4 displays the line when (w)= -0.5 and (I) =1, and lastly, figure 5 represents the simulated membrane potential line where (w) was set to 0.5 and (I) to 1. When comparing the results for each combination through the 4 figures, we can notice that the change in (w) affected the displayed line greatly, lower (w) value which was (-0.5) results in higher membrane potential, while higher values result in the opposite of it. Moving to (I), while holding still the (w) values in changing only (I), it’s clear from the figures that (I) values are not affecting the membrane potential too much, this might refer to the reason that (w) is making the stronger effect so it outstands the effect of (I), or because the initial value of (v) affects the whole simulation, which reflects on the effect of (w) and (I) negatively and reduce their effect. When it comes to evaluating the overall performance of this simulation, it’s very clear that the simulation is not good as it has the same values after the first iteration always, which indicates that the simulation can’t capture any true information regarding the true membrane potential, which can also be seen from the difference between the blue line (real membrane potential values) and the black line (simulation values) in the figures below.

Figure 2: Membrane potential when w=-0.5 and I=0.3 Figure 3: Membrane potential when w=0.5 and I=0.3

A graph of a diagram

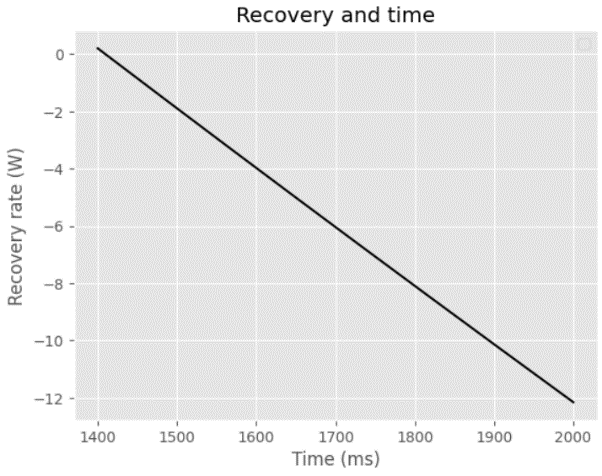
Description automatically generated with medium confidenceFigure 4: Membrane potential when w=-0.5 and I=1 Figure 5: Membrane potential when w=0.5 and I=1

The second equation displayed in figure 1 simulates the change in the recovery variable (Change in (w)) relying on the voltage (v), the recovery variable (w), and 3 other parameters (epsilon, a, and b), through the equation, we can monitor how (w) values change according to the following variables:

* The voltage (v): As mentioned earlier, it’s an indication for the electrical state of the neuron, here higher values indicate an action potential, while lower values represent the rest mode of a neuron. As the (w) affects (v) values, so does (v) affects (w), because (w) will have higher values when the neuron is not in the resting mode which means when (v) changes.
* The recovery variable (w): The variable that is responsible for returning the neuron to the rest mode after an action potential occurs. In the second equation, (w) is the variable we are interested in and monitoring its change, as the previous value for it affects the next one with dw/dt change that we are computing through the equation.
* Epsilon (ε): Is a parameter that controls how quickly (w) should response to the change in (v) values. Lower values result in lower response for (w).
* Variable (a): A variable that affects the change in (w) values, where higher values for (a) increases the change in (w) values, making the change higher in order to balance between (v) and (w) values.
* Variable (b): (b) controls how (w) influences (v), because it is multiplied by (w) in the equation, (b) either increases or decreases the result of w\*b.

In my project, and just as I did with the first equation, I simulated this equation to see how the recovery rate changes over time, the recovery rate change relies on, (w), (v), (a), (b), and (e), the initial value for (w) was 0.2, while the value for (v) was always -1, I manipulated (a), (b), and (e) values to see their effect on the recovery rate and its line. Figure 6 displays the recovery rate line when (a)= 0.7, (b)=0.001, and (e)= 0.07, it shows that the line is decreasing in a linear shape as the time increases, figure 7 shows the line with the same (b) and (e) values but a smaller value for (a) which is 0.04, the only difference between the figure 6 and figure 7 is the range of values for (w), so we can say that the change in (a) doesn’t affect the function’s shape but affects the range of values. In figure 8, (a)=0.04, (b) is changed to be 0.03, and (e) is the same, we can clearly notice that the function’s shape changed when changing (b) value, it has a bit of curve instead of straight linear, we see that also the range of (w) values changed, Figure 9 shows the biggest change that happened to the function’s shape than the other figures, where (a) =0.04, (b)= 0.03, and (e)=0.5, the function now looks more quadratic than linear, and we can see that between 1400-1600 the (w) values change and after that it kind of stays the same. To conclude, changing the values of (a), (b), and (e) all affects the change in the recovery rate in either the range of values, the function’s shape, or both. Although changing the parameters values results in different lines for the recovery rate but the results are not good enough as they can’t shows the up and down values for the recovery rate which must exist as in the real data the membrane potential arises and drops multiple times, and what drops it as higher values for (w) which are not captured in the simulation no matter how the parameters changed.

*A graph with a line

Description automatically generatedFigure 6: Recovery rate when a=0.7, b=0.001, and e=0.07 Figure 7: Recovery rate when a=0.04, b=0.001, and e=0.07*

A graph of recovery and time

Description automatically generatedA graph of recovery and time

Description automatically generated Figure 8: Recovery rate when a=0.04, b=0.03, and e=0.07 Figure 9: Recovery rate when a=0.04, b=0.03, and e=0.5

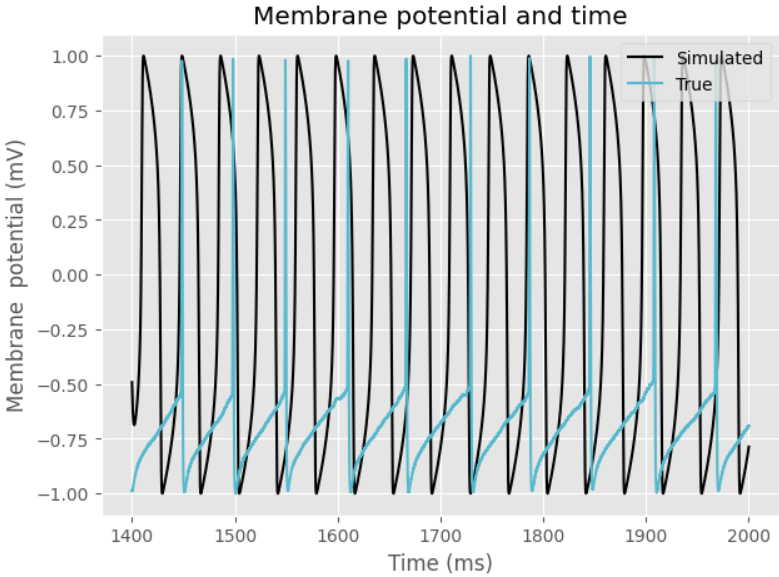
From the simulation for both equations, we can see that the first order simulation, in other words, simulating each equation independently does not show great results as the model was not able to simulate the membrane potential and the recovery rate in a great way, this leads us to think of a more complex and detailed model that may capture information regarding the real data which is the second order system which utilizes both equations together.

**1.5** Simulate the second-order system that consists of both equations together, where I(t) is the input, and V(t) is the output (Code)

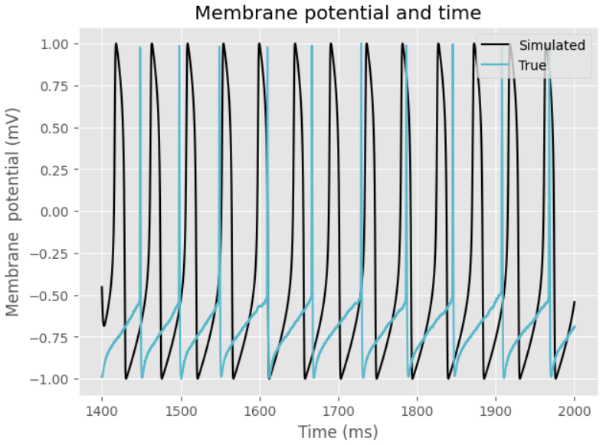
## **1.6** Explain the theoretical principles of the second-order system and the effect of varying the differential equation's parameters

The second-order system consists of the 2 equations displayed in figure 1 together, where any update in (v) value in the first equations is reflected also in (v) value in the second equation, and the same applies for (w), so instead of dealing with a constant (v) in the second equation, or a constant (w) in the first equation, we deal with the updated values for the 2 parameters, where the new values are the previous values from the previous iteration added to the change which is either dv/dt or dw/dt. In the second order model I built, (v) and (w) values are calculated each time, (I) is a constant, while (a), (b), and (e) are parameters that I changed to study their effect trying to get better simulation that is close to the real data. Figure 10 displays the simulated membrane potential when (a)= 0.04, (b)= 0.001, and (e)= 0.07, while figure 11 displays the same line but when changing (a) to 0.7, it shows less frequency or action potentials that are closer to the real data displayed in the blue line. Figure 12 looks very similar to figure 11 although the (b) value was changed to 0.03, the figure looks the same, so it seems like changing (b) is not impacting the membrane potential that much, in figure 13 (a) was the same which is 0.7, (b)= 0.03, while (e) changed to become 0.05, and it is noticeable that the figure is the closest one to the real data, and it seems that changing (e) affects the membrane potential giving us more accurate and close results. From the tested values for each parameter, we can say that (a) affects the frequency, (b) has no noticeable effect, while tuning (e) correctly results in more accurate and close results to the real ones. Collecting all together, simulating Nagumo model as a second order system achieved better results that simulating each equation alone as a first order which can be derived from the below figures when comparing them to the figures of the first order which are figures 2, 3, 4, 5. For further improvements, 3 optimization algorithms will be applied to get better values for (a), (b), and (e) which might result in closer and more accurate results.

A graph of a wave

Description automatically generated with medium confidence*Figure 10: Recovery rate when a=0.04, b=0.001, and e=0.07 Figure 11: Recovery rate when a=0.7, b=0.001, and e=0.07*

A graph of a number of lines

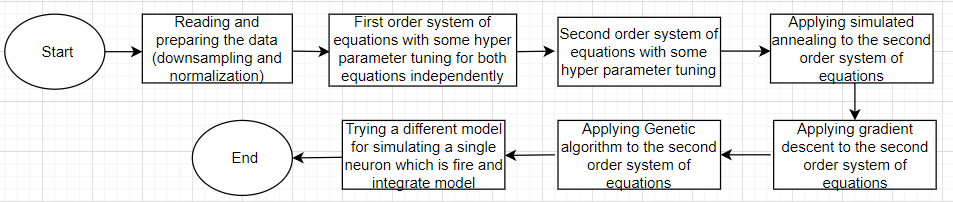
Description automatically generated with medium confidence Figure 12: Recovery rate when a=0.7, b=0.03, and e=0.07 Figure 13: Recovery rate when a=0.7, b=0.03, and e=0.05

## **1.7** Use empirical data to tune the parameters of a simulation model (Code)

## **1.8** Apply different optimization techniques to tune simulation parameters (Code)

# **Part 2**

## **2.1** Design a detailed workflow for solving a specific problem using modeling and simulation

*Figure 14: Flowchart for solving the problem using modeling and simulation*

## **2.2** Evaluate the performance of the second-order system in achieving desired behavior

The second order system in my project was built using a baseline and manual hyperparameter tuning, after that, 3 optimization algorithms were applied which are gradient descent, simulated annealing, and genetic algorithm. The best performance among all tested models was the baseline model with the following values for the parameters a=0.7, b=0.03, and e=0.05, which is displayed in figure 13, it was the best model that was able to simulate the spikes and their frequency, it’s a little bit surprising that a manual hyper parameter tuned model outperformed 3 optimization algorithms, while the best model out of the 3 optimization algorithms was gradient descent. Although not all models that performed good, but I got great results from others that were able to predict the spikes even if not 100% accurate, so I can describe the second order system as a great system for simulating the membrane potential and a better one from the first order, as it combines the 2 equations of Nagumo models providing improved behavior. Lastly, the second order system models that were built performed great and could perform even better with more iterations or hyperparameter tuning.

## **2.3** Analyze the effectiveness of each optimization technique for tuning the parameters of the simulation using empirical data

In my project, I used 3 optimization algorithms, which are gradient descent, simulated annealing, and genetic algorithm in order to find the best values for the parameters (a), (b), and (e). Starting with the gradient descent, it was able to simulate and capture spikes but not in a very good way as it showed more spikes than the real data, figure 15 shows the simulated action potential in the black vs the real data in the blue line. Moving to the simulated annealing optimization algorithm, as shown in figure 16, the model performed poorly, it didn’t capture any spike, instead, it started a spike and kept the membrane potential high indicting a continuous spike which is wrong as nothing like this exists in the real data, the reason of that is because the values that simulated annealing found for the parameters wasn’t really great, maybe giving it more time to simulate would achieve better results. Similar to Simulated annealing performed the genetic algorithm, the chosen values for the parameter by the algorithm didn’t enable the model to predict the spikes as shown in figure 17. Collecting all together, gradient descent was the best optimization algorithm among the 3 algorithms used, it’s performance might be even better if more iterations were added, talking about simulated annealing and genetic algorithm, increasing the number of generations in genetic algorithm or reducing the minimum temperature in the simulated annealing might give improved results and better performance, or it might be because gradient descent is more suitable for the problem I am working on.

*Figure 15: Gradient descent performance Figure 16: Simulated annealing performance*

*Figure 17: Genetic algorithm performance*

**A graph of a waveform

Description automatically generatedA graph of a number of cells

Description automatically generated with medium confidence**A graph of a number of lines

Description automatically generated with medium confidence

# **Part 3**

## **3.1** Investigate previous methods and work done in the field of neuronal modeling

In the field of neuronal modeling, there has been a lot of ongoing work and research. A paper proposed a review of map-based neuron models, through examining their validity especially those from integrate and fire family and comparing them to traditional continuous-time models, after a thorough examination, they concluded that map-based neuron models have great behavior and are suitable for modeling related to nervous systems and have a potential for understanding neural networks dynamics [5]. Another paper proposed a modified version of Sutton- Barto neuronal model, they modified the model to account and include a wide range of classical conditioning phenomena and they suggested some experimental tests for the model, the resulting model succussed in predicting a lot of classical conditioning phenomena and also resolved some inconsistencies that existed in the Sutton-Barto model [6]. Another model was proposed as an enhancement to Hindmarsh- Rose neuron model which took into consideration the effects of magnetic flow on neuronal electrical activity, this was done by introducing a fourth variable to the model for the magnetic flow, the proposed model showed that including the magnetic flow significantly impacts the neuron dynamics [7]. Cross- correlation functions were also used to study the neuron behavior by adding a constant threshold resetting the neuron to the resting level after an output, they used a combination of periodic and Poisson inputs that better characterize the neuron, they concluded that further research is needed [8]. Another study proposed a probabilistic spiking neuron model. The model considered the influence of genes and protein on spiking activities, the paper outlined the potential applications for such a model like classification and associative memory [9]. A paper provided a clear understanding for Hindmarsh-Rose model and proposed a simpler and circuit friendly version that may replicate the behavior of that model, the results showed similar behaviors for the model indicating the effectiveness of the model [10]. A computer-based model was proposed to simulate neural networks, the model includes the ion channels such as Na+, and K+ channels which are crucial for modeling action potentials, the model showed ability to simulate the specified task, and also highlighted a potential for more applications [11]. Furthermore, a new stochastic model was proposed to simulate the neuron behavior more accurately, they incorporated stochastic differential equations with Brownian motion to model the ion channels operations, the proposed model achieved great results and outperformed previous deterministic models [12]. Lastly, another paper examined a mathematical model of neuron membrane potential in the presence of refractoriness, it provided formulas that enhance the effect of refractoriness on neuronal activity [13]. In single neuron modeling, estimating the parameters has to be a good and accurate process, a paper investigates 2 parameter estimation methods aiming to decide which is more efficient and accurate, the first one was time domain method while the second was frequency domain method, both provided accurate and great results, but time domain method is slower and might be less accurate sometimes [14].

## **3.2** Critically analyze and compare the performance of different models used for neuronal modeling and simulation

In the concept of neuronal modeling, there exists a lot of models to simulate the membrane potential, the main model that I used in my project was Nagumo model, I tried the model using first order, second order, and optimization techniques in second order, a baseline model in the second order that was displayed in figure 13, as well as the gradient descent model displayed in figure 15, provided the best results among all models that I tried. In order to better compare the behavior and performance of Nagumo model, I implemented another model called the leaky integrate and fire model which works on the concept of a threshold, where a voltage value higher than the threshold, indicates a spike and a lower value indicates resting mode, when implementing the model, I also started with a baseline model, then I applied the same 3 optimization techniques used in Nagumo model. Starting with the baseline model, it performed poorly as it didn’t capture any spike. The same thing happened with the gradient descent, it also wasn’t able to capture spikes, the performance for both of them can be seen from figures 18, and 19. Simulated annealing was the best performing model in the LIF model, as it was able to simulate spikes, but it had a problem with their frequency as it simulated much more spikes the real ones in the real data as shown in figure 20. Lastly, the genetic algorithm model also didn’t perform well as displayed in figure 21, so we can say that simulated annealing was the best technique to implement the LIF model.

Comparing the 2 models used which are Nagumo model and the leaky integrate and fire model, in both models there was n implemented model that was able to capture spikes, but I can say from the results that Nagumo model performs better than integrate and fire model as it captured the spikes more accurately. To sum up, the project involved 2 models to simulate the single neuron, while LIF model was able to capture spikes, Nagumo model outstands in the concept of neuron modeling provide better performance and more accurate results.

A graph of a number of lines

Description automatically generated with medium confidence*A graph of a number of cells

Description automatically generated with medium confidenceFigure 18: Baseline LIF performance Figure 19: Gradient descent LIF performance*

*A graph of a diagram

Description automatically generated with medium confidenceA graph of a number of lines

Description automatically generated with medium confidence*A graph of a number of lines

Description automatically generated with medium confidence*A graph of a number of cells

Description automatically generated with medium confidenceFigure 20: Simulated annealing LIF performance Figure 21: Genetic algorithm LIF performance*

# **References**

[1] “Preface,” The GENESIS Simulator, http://www.genesis-sim.org/GENESIS/iBoG/iBoGpdf/index.html (accessed May 18, 2024).

[2] E. M. Izhikevich and R. FitzHugh, “Fitzhugh-Nagumo Model,” Scholarpedia, http://www.scholarpedia.org/article/FitzHugh-Nagumo\_model (accessed May 18, 2024).

[3] Author links open overlay panelIdan Segev a b et al., “Single neurone models: Oversimple, complex and reduced,” Trends in Neurosciences, https://www.sciencedirect.com/science/article/abs/pii/016622369290003Q?via%3Dihub (accessed May 19, 2024).

[4] P. Gupta, N. Balasubramaniam, H.-Y. Chang, F.-G. Tseng, and T. S. Santra, “A single-neuron: Current trends and future prospects,” MDPI, https://www.mdpi.com/2073-4409/9/6/1528 (accessed May 19, 2024).

[5] Author links open overlay panelB. Ibarz a *et al.* (2010a) *Map-based models in neuronal dynamics*, *Physics Reports*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0370157310003224 (Accessed: 22 June 2024).

[6] Klopf, A.H. (2013) *A neuronal model of classical conditioning - psychobiology*, *SpringerLink*. Available at: https://link.springer.com/article/10.3758/BF03333113/metrics (Accessed: 22 June 2024).

[7] Lv, M. *et al.* (2016) *Model of electrical activity in a neuron under magnetic flow effect - nonlinear dynamics*, *SpringerLink*. Available at: https://link.springer.com/article/10.1007/s11071-016-2773-6 (Accessed: 22 June 2024).

[8] (No date) *Cross-correlation functions for a neuronal model*. Available at: https://www.cell.com/biophysj/pdf/S0006-3495(74)85936-9.pdf (Accessed: 22 June 2024).

[9] Author links open overlay panelNikola Kasabov *et al.* (2009) *To spike or not to Spike: A probabilistic spiking neuron model*, *Neural Networks*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0893608009002317 (Accessed: 22 June 2024).

[10] Storace, M., Linaro, D. and de Lange, E. (2008) *The Hindmarsh–Rose Neuron Model: Bifurcation analysis and piecewise-linear approximations*, *AIP Publishing*. Available at: https://pubs.aip.org/aip/cha/article-abstract/18/3/033128/342125/The-Hindmarsh-Rose-neuron-model-Bifurcation?redirectedFrom=fulltext (Accessed: 22 June 2024).

[11] Ekeberg, &Ouml;. *et al.* (no date) *A computer based model for realistic simulations of neural networks - biological cybernetics*, *SpringerLink*. Available at: https://link.springer.com/article/10.1007/bf00202382 (Accessed: 22 June 2024).

[12] Author links open overlay panelAntti Saarinen *et al.* (2006) *Modeling single neuron behavior using stochastic differential equations*, *Neurocomputing*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0925231205003772 (Accessed: 22 June 2024).

[13] Author links open overlay panelA. Buonocore a *et al.* (2002) *A neuronal modeling paradigm in the presence of refractoriness*, *Biosystems*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0303264702000618 (Accessed: 22 June 2024).

[14] Tabak, J., Murphey, C.R. and Moore, L.E. (2021) *Parameter estimation methods for single neuron models - Journal of Computational Neuroscience*, *SpringerLink*. Available at: https://link.springer.com/article/10.1023/A:1026531603628 (Accessed: 22 June 2024).