Hodgkin-Huxley? No, Parakh-Varanasi.

Pranav Parakh, Varun Varanasi

December 15, 2022

Abstract

Brain function has been one of physiology's largest mysteries. Over the years scientists have understood the role of electricity in neural signaling, neuron firing mechanisms, and even neural pathways, but the emergent phenomena of memory and consciousness still remain open questions. The goal of this project is to provide an analytic framework to understand neural pathways. Our work can be understood in two parts: exploration of neuron function and an exploration of neuron pathways. The first phase of our project focuses on analyzing the Hodgkin-Huxley mathematical model for neurons. We implement the Hodgkin-Huxley model and study the effect of various model parameters on neuron function. We pay specific attention to complex and non-linear behavior. The second phase of this project can be seen as an application of of Hodgkin-Huxley neurons to neural pathways in the brain. In this portion we construct neural networks of Hodgkin-Huxley neurons and study the signal processing abilities of these neural circuits. Our findings indicate that our Hodgkin-Huxely neural net model, the Parakh-Varanasi model, is capabale of preserving certain signal characteristics through the network. Furthermore, we provide a basic proof of concept that variation of certain parameters allow us to control Parakh-Varanasi outputs. Finally, we conclude our project outlining a future project proving the signal re-producing capabilities of these Hodgkin-Huxley network models. We hope that this analytic frame work work plays a part in uncovering the role of neural circuits and higher cognitive function.

Contents

1 Introduction

Among the open questions in the physiology, perhaps none has gripped humanity as strongly the mystery of how our brains work. Containing approximately 14- 16 billion neurons, the human brain is by far the most intricate and least understood part of the body. Over the years science has been able to uncover individual components of the brain, but the high level function of the brain is still largely a mystery. The goal behind this project is to explore the mathematical relationship between neurons, neural circuits, and memory.

1.1 Historical Background: Neurons

Before Luigi and Lucia Galeazzi Galvani's landmark 1791 discovery of animal electricity, the human body's function was largely misunderstood. The scholars of the time believed that the bodily functions were the product of fluid movement and pressure; however, after the discovery that frog legs responded to electric impulses, the scientific community raced to uncover the relationship between bodily functions and electricity. By the 1930's scientists had discovered neurons, action potentials, and a relationship between ion flow and neuron function, but the details of the relationship still remained a mystery. The intricacies of this relationship remained a unsolved for nearly 20 years until Alan Hodgkin and Andrew Huxley published their seminal 1951 paper explaining the relationship between ion channel kinetics and neuron function. This accomplishment later earned them the 1963 Noble Prize in Physiology/Medicine.

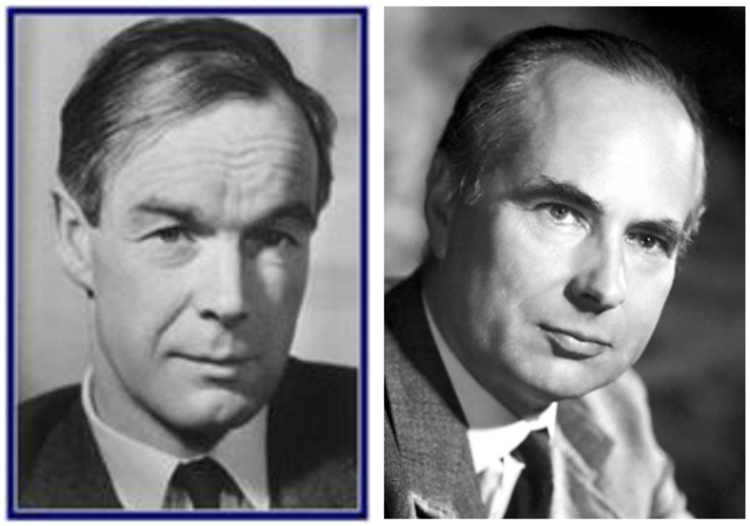


Figure 1: Alan Hodgkin (left) and Andrew Huxley (right)

Before they were Noble prize winning scientists, Hodgkin and Huxley's professional relationship began in 1939 when Alan Hodgkin invited then graduate student Andrew Huxley to join him in Plymouth, Massachusetts studying squid axons. After a short hiatus due to World War 2, the pair published their first major scientific breakthrough: a dual-electrode voltage clamp. This newly improved voltage clamp allowed the scientists to accurately measure ion flow across a membranes without affecting the membrane potential. With this device in hand, the pair turned their attention towards studying neuron action and soon after published their work detailing the mathematical relationship between ion channels and action potentials.

1.2 Biological Background: Neurons

Before delving into the details of action potentials and Hodgkin and Huxley's results, it is essential to understand the morphological structure of the cell itself. As seen below, the key components of a neuron cell membrane are the plasma membrane (pink), ion channels, and pumps.

Typically, the plasma membrane, a phospholipid bilayer is impermeable to ion flow; however due to the presence of ion channels and pumps, cells are able to regulate the concentrations of specific

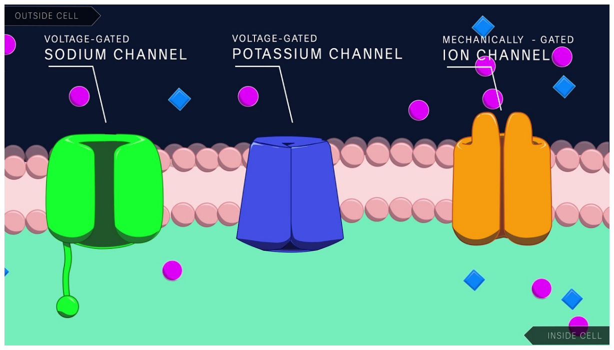

OUTSIDE CELL
VOLTAGE-GATED SODIUM CHANNEL
VOLTAGE-GATED POTASSIUM CHANNEL
MECHANICALLY - GATED ION CHANNEL
INSIDE CELL

Figure 2: Neuron Cell Membrane Diagram

ions within the cell. Constructed from glycoproteins, these ion channels typically open and close in response to membrane potentials. The interactions between the regulatory behaviors of these bumps create complex behaviours within the cell. Action potential, one of such behaviours, are the electrical impulses transmitted across our bodies. They are characterized by a rapid spike in the membrane potential. The life cycle of an action potential can be seen in the diagram below.

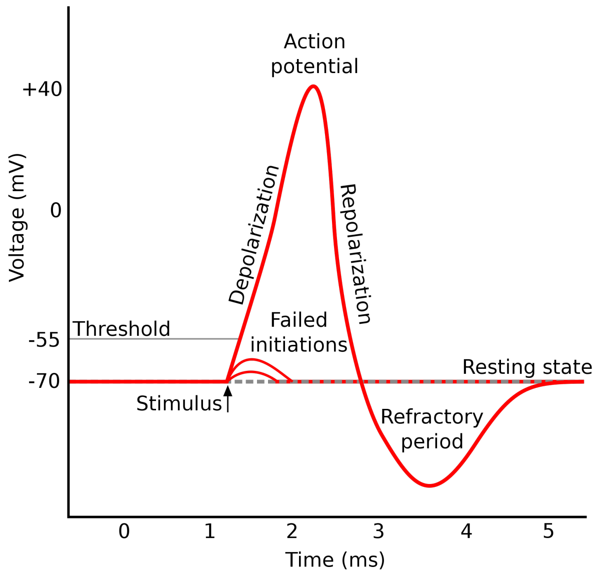


Figure 3: Action Potential Diagram

At a high level action potentials can be thought of as the neurons response to stimulus. As the incoming signals surpass the threshold, the neuron will open the voltage-gated Na+ gates and positive ions will influx the cell. As the cell depolarizes and reaches its peak voltage, the Na+ gate closes and the K+ gate opens allowing K+ to exit the cell. Eventually the cell potential returns to the original level and enters a refractory period during which the sodium gates are inactivated.

1.3 The Hodgkin-Huxley Model

The original Hodgkin Huxley model was described in the 1952 paper titled "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve." This paper outlined a mathematical model to explain the initiation and propagation of action potentials in the giant squid axon. The model is a series of nonlinear differential equations that can be generically applied to describing action potentials in neurons.

In this section, we will present the original Hodgkin-Huxley model and will examine what each of the parameters in the model represent. This model described the components of the cell as electrical circuit elements.

The original paper described three different kinds of current that were observed in the squid axon: flow of sodium ions, flow of potassium ions, and leak current. It was found that the leak current mainly consisted of chlorine ions. The model describes the resulting state of the axon after an input current has been applied. We'll first examine a schematic diagram of the cell shown in Figure 1 and will then outline the mathematical formalism.

The circuit diagram in Figure 1 is a useful tool to visualize the semi-permeable cell membrane as an electrical circuit. We see that the top of the figure represents outside the cell membrane, and the bottom of the cell is inside the cell membrane. Let's examine what each of the parameters of the model mean:

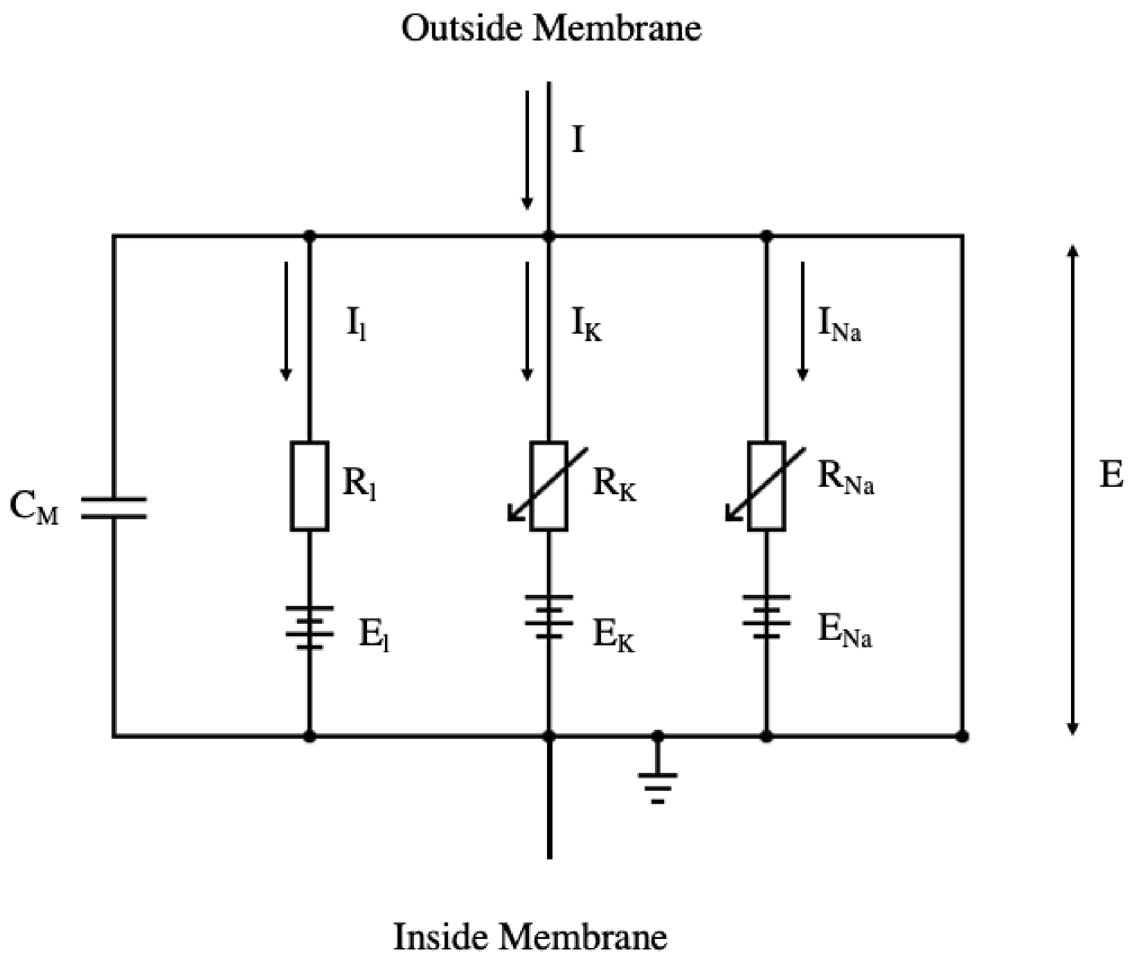


Figure 4: Circuit diagram representation of semi-permeable axon membrane

I: This is the current that is applied to the cell, and is equal to the total membrane current density. This input current on the cell must either pass through the capacitor or one of the three resistors. The Hodgkin-Huxley model describes the response of a giant squid axon to an input stimulus, so when simulating, we can examine a variety of different input currents affecting the squid axon.

I_l, I_K, I_{Na}: These currents represent the ionic current densities. They are the current passing through each of the three channels. The three channels are the leakage channel, the potassium channel, and the sodium channel.

$\mathbf{R_l}$: This resistor describes the resistance of the leakage channel, which mainly consists of chlorine ions

$\mathbf{R_K}$, $\mathbf{R_{Na}}$: These resistances correspond to the ionic current densities of potassium and sodium through their respective channels in the cell membrane. They are different from the resistance of the leaky channel, as these resistances are not fixed an may change. For example, as the chanels open and close, these resistances will be altered.

$\mathbf{C_M}$: This is the capacitance of the cell membrane. It is the membrane capacity per unit area, and is assumed to be constant in the model.

$\mathbf{E_l}$, $\mathbf{E_K}$, $\mathbf{E_{Na}}$: The potential that drives ion flow in an out of the membrane is denoted as a battery. This is the Nernst potential, driven by the difference in ion concentration inside and outside of the membrane

E: This is the overall potential difference across the membrane, and is equal to the voltage across the capacitor.

We can translate this circuit representation of the model can now be described in differential equations. Hodgkin and Huxley originally present the following single equation:

$$I = C_M \frac{dV_{\mathrm{M}}}{dt} + I_i \tag{1}$$

They then define each of the ion currents to be:

$$I_{\mathrm{Na}} = g_{\mathrm{Na}}(E - E_{\mathrm{Na}}) = g_{\mathrm{Na}}(V_M - V_{\mathrm{Na}}) \tag{2}$$

$$I_{\mathrm{K}} = g_{\mathrm{K}}(E - E_{\mathrm{K}}) = g_{\mathrm{K}}(V_M - V_{\mathrm{K}}) \tag{3}$$

$$I_{\mathrm{l}} = \bar{g_\ell}(E - E_\ell) = \bar{g_\ell}(V_M - V_\ell) \tag{4}$$

They then redefine the conductance's to account for additional activation parameters n, m, and h.

$$g_K = \bar{g_K} n^4 \tag{5}$$

$$g_{Na} = \bar{g_{\mathrm{Na}}} m^3 h \tag{6}$$

We can then use equations 2-6 to expand equation 1 into a series of 4 coupled nonlinear differential equations as follows:

$$I = C_M \frac{dV_m}{dt} + \bar{g_K} n^4 (V_m - V_k) + \bar{g_{Na}} m^3 h (V_m - V_{Na}) + \bar{g_\ell}(V_m - V_\ell) \tag{7}$$

$$\frac{dn}{dt} = \alpha_{\mathrm{n}}(V_m)(1-n) - \beta_{\mathrm{n}}(V_m)n \tag{8}$$

$$\frac{dm}{dt} = \alpha_{\mathrm{m}}(V_m)(1-m) - \beta_{\mathrm{m}}(V_m)m \tag{9}$$

$$\frac{dh}{dt} = \alpha_{\mathrm{h}}(V_m)(1-h) - \beta_{\mathrm{h}}(V_m)h \tag{10}$$

Hodgkin and Huxley further define the rate parameters, α_{i} and β_{i} for $i = m, n, h$ as follows:

$$\alpha_n(V_m) = \frac{0.01(10 - V)}{e^{\frac{10-V}{10}} - 1} \tag{11}$$

$$\beta_n(V_m) = 0.125 e^{-\frac{V}{80}} \tag{12}$$

$$\alpha_m(V_m) = \frac{0.01(25 - V)}{e^{\frac{25-V}{10}} - 1} \tag{13}$$

$$\beta_m(V_m) = 4 e^{-\frac{V}{18}} \tag{14}$$

$$\alpha_h(V_m) = 0.07e^{-\frac{V}{20}} \tag{15}$$

$$\beta_h(V_m) = \frac{1}{e^{\frac{30-V}{10}} + 1} \tag{16}$$

Let's examine what each of the parameters in the above equations represent. Some of the parameters described in the circuit are similar to the ones used in the differential equations, but for completeness, we will describe what each of the parameters in the differential equation system represents here:

I is the current across the entire membrane. It is dependent on the current passing through each of the channels as well as the resistance and potential for each of the ion flows.

$\mathbf{C_M}$ is the membrane capacitance, and is treated as a constant in the system.

t is time.

$\bar{\mathbf{g_K}}$ is the conductance for the potassium channel. This conductance value is used to define the resistance that is present in our circuit diagram of the cell membrane. The resistance is defined as the inverse of the conductance.

$\bar{\mathbf{g_{Na}}}$ is the conductance for the sodium channel. This value is used to define the resistance of the sodium channel present in our circuit diagram.

$\bar{\mathbf{g}}_\ell$ is the conductance for the leak channel. This value is used to define the resistance of the leak channel in our circuit diagram.

n satisfies equation 8 and is a dimensionless quantity between zero and one. It is a probability that describes the potassium sub-unit channel activation. Notice, in our equation we have n^4. This is because in the giant squid axon model, the potassium channel is made up of four sub-units which all must be activated in order for potassium ions to be able to flow into the membrane.

m is similar to n in that it is a dimensionless probability between zero and one which satisfies equation 9. m describes the activation of the sodium sub-unit channel in the membrane. In the model, m is raised to the power of three because exactly three of the sodium sub-units need to be activated for sodium ions to flow through the channel.

h is also a dimensionless probability between zero and one which satisfies equation 10. It describes the sodium channel sub-unit inactivation. This is included in the model because one sodium sub-unit must be inactive for sodium ions to flow through the membrane.

$\mathbf{V_m}$ is the displacement of the membrane potential from its equilibrium value. That is, V_m is the resulting membrane potential.

V_K is the potassium reverse potential. This is the electrochemical gradient of potassium ions inside and outside of the axon that drives movement through the channels.

V_{Na} is the sodium reverse potential. This is the electrochemical gradient of sodium ions inside and outside of the axon that drives movement through the channels.

$\mathbf{V}_\ell$ is the leak current reverse potential. This number represents an average of the electrochemical gradient of all of the other ions flowing through the leak channel.

$\alpha_\mathbf{i}$ **and** $\beta_\mathbf{i}$ and rate constants that determine the speed of movement of through the i^{th} ion channel for $i = m, n, h$

1.4 Historical Background: Neuron Circuits

Concurrently with the discovery of neuron function, the scientific community began to study the relationships between neurons and bodily functions. By 1959, Warren Sturgis McCulloch and Walter Pitts published the first foundational literature describing processing characteristics of neural networks. Specifically, in this paper they showed the theoretical underpinnings of arithmetic and logic in artificial neural networks.

1.5 Biological Background: Neuron Circuits

At a high level, brain function can be understood as a response of neural pathways to given stimuli. Neurons consist of 4 main parts: dendrite, synapse, soma, and axon. The dendrite receives signals for connected neurons, the soma processes the signal, the axon transmits the signal to other neurons, and the synapse serves as the point of connection between adjacent neurons. Our nervous system functions through a complex and interconnected pathway of neurons known as neural circuits.

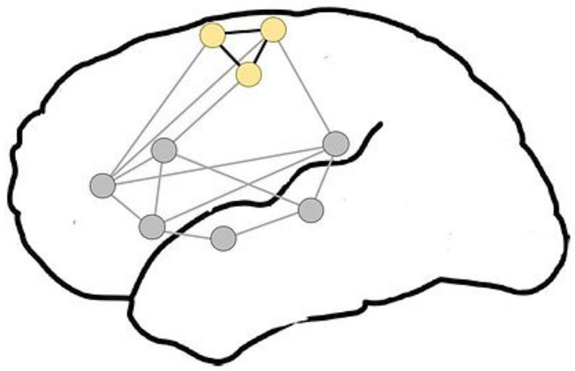


Figure 5: Neuron Circuit Diagram

These neural circuits conjoin to form large-scale brain networks which are complex networks activated for specific bodily functions. At the moment these systems are our best understanding of cognition.

1.6 Project Outline

With this background in mind, our project's focus is two-fold: explore the dynamics of the Hodgkin-Huxley model and model neural circuits of Hodgkin-Huxley neurons. We begin with an initial implementation and exploratory analysis of the Hodgkin-Huxley system with a specific focus on non-linear behavior. Afterwards, we construct a neural net of Hodgkin-Huxley Neurons and explore the efficacy of the networks as signal transmitters. Finally, we explore the possibility of reproducing input signals via weight re-parameterization of the network.

2 Hodgkin-Huxley Model implementation

In problem set 5, we implemented a very reduced version of the Hodgkin-Huxley model. Here, we provide code to implement the full model as it was described. Then, we'll vary several of the parameters in the model to try and understand what they do.

2.1 Implementation of the Full Hodgkin-Huxley Model

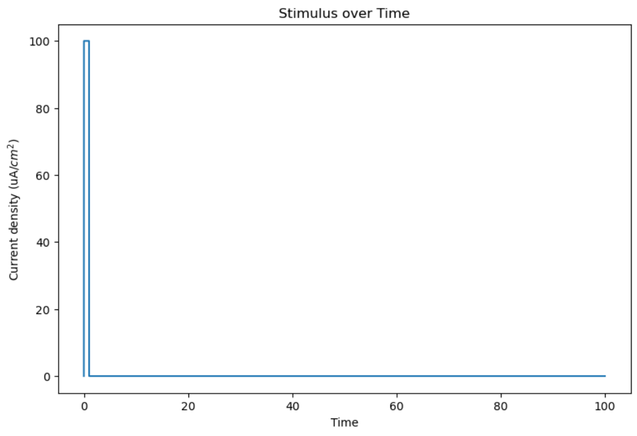
The python script for the implementation of the Hodgkin-Huxley model defined in the previous section can be found in Appendix B. Using this model, we can define arbitrary input pulses that will act on the axon we are simulating. To visualize this, we can think back to the circuit diagram in Figure 3. The pulse we apply is modeled by the I entering the cell. Our main output is the potential response of the membrane (V_m), which in the circuit corresponds to the potential across the circuit (ΔE).

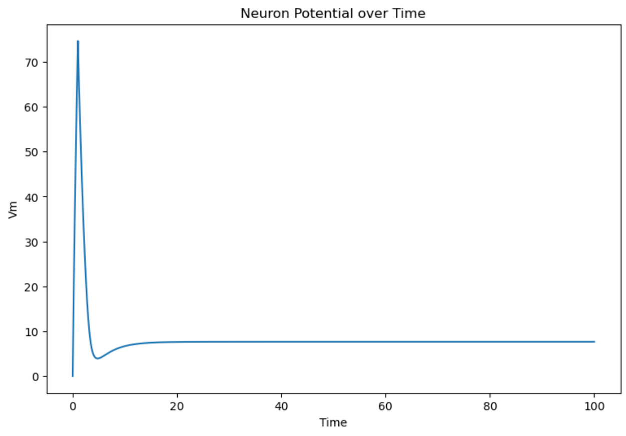
Using the model, we we use the following parameters as a baseline.

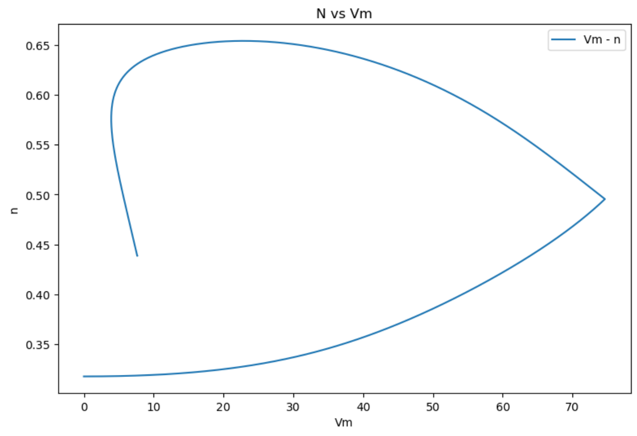
- Start time (t_0, in s): 0
- End time (t_1, in s): 100
- Membrane Capacitance (C, in $\frac{\mu F}{cm^2}$): 1.0

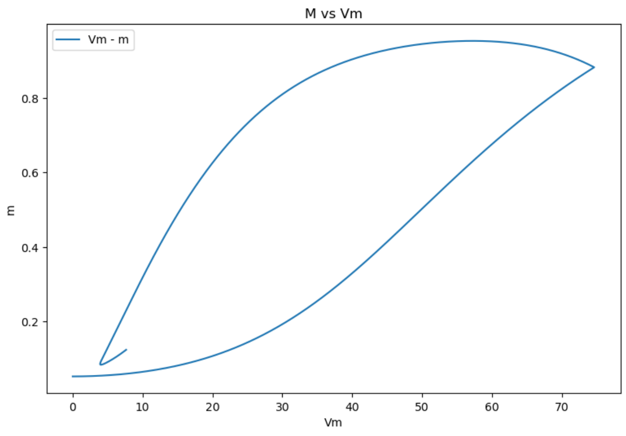
- Potassium potential (VK, in mV): -10.0
- Sodium potential (VNa, in mV): 110.0
- Leak potential (Vl, in mV): 10.0
- Potassium channel conductance (gK, in $\frac{mS}{cm^2}$): 4.0
- Sodium channel conductance (gNa, in $\frac{mS}{cm^2}$): 4.0
- Potassium channel conductance (gL, in $\frac{mS}{cm^2}$): 1.0

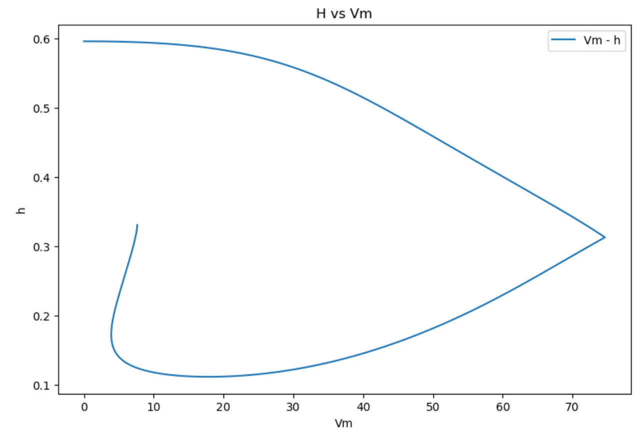
Now we apply a simple square pulse at $t = 0$, and simulate the response of the axon over time.

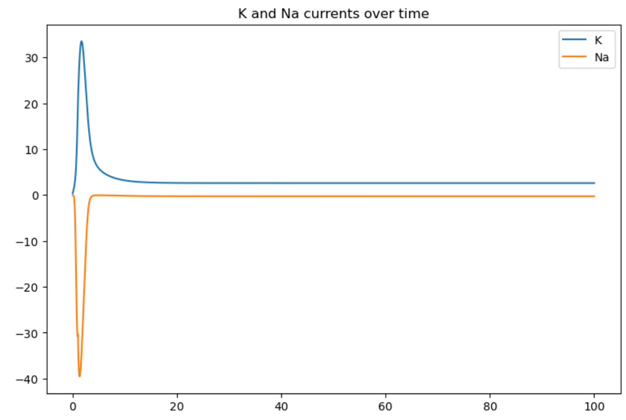












We can also plot the three activation parameters over time in response to this pulse.

Figure 6: Plot of input stimulus and axon potential over time, along with subplots of n, m, and h, the activation parameters, against Vm. The final plot on the bottom left is a plot of the Potassium and Sodium currents over time

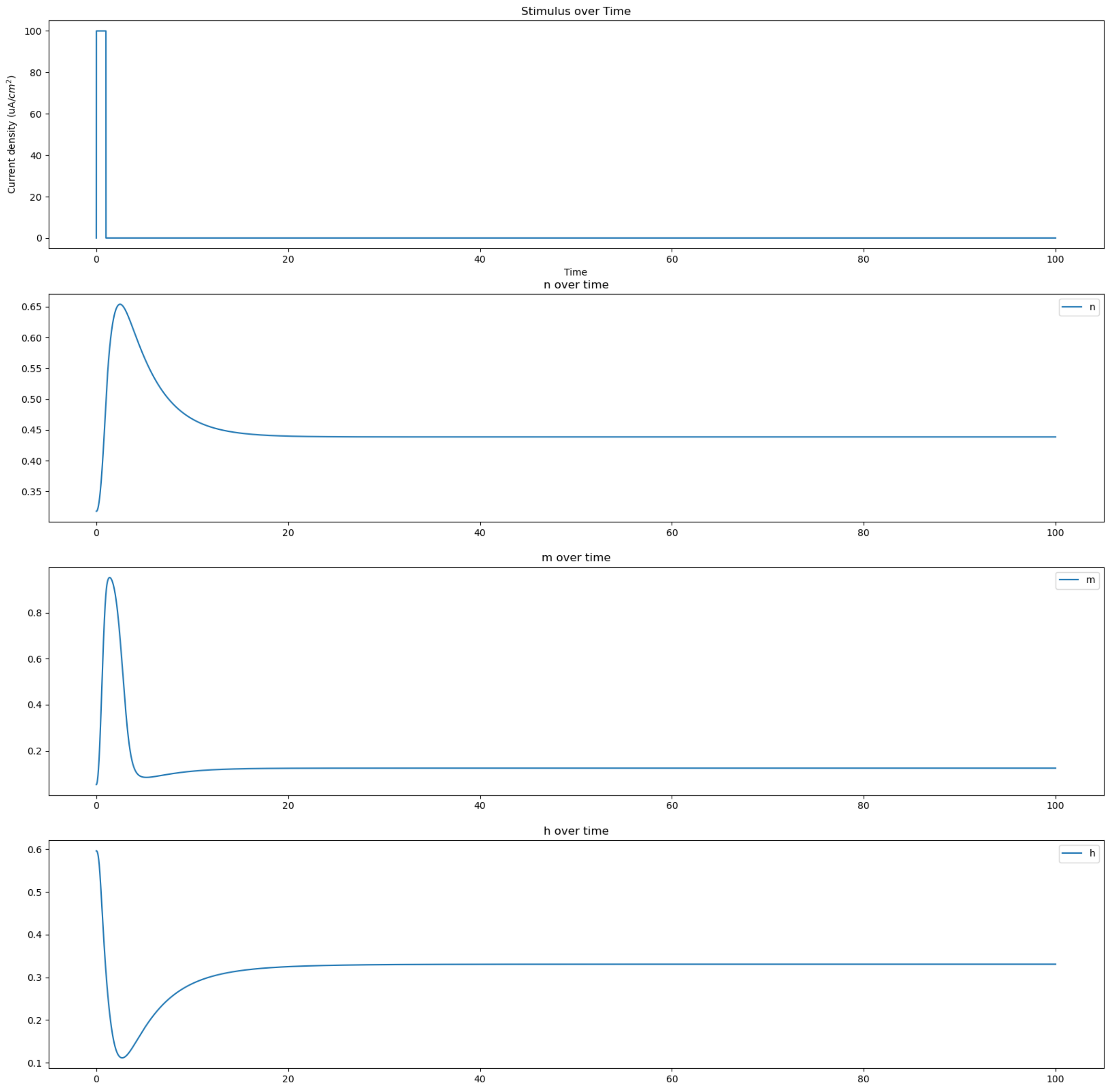


Figure 7: Plot of n (potassium sub-unit channel activation), m(sodium sub-unit channel activation), and h(sodium channel sub-unit inactivation) against time. n, m, and h are probabilities that take values between zero and one

We see that our model seems to be working, and has some interesting results that can help us better understand what is going on. First, in Figure 5, we see that the membrane potential responds directly to a current as a spike in potential, which makes sense. We notice that our simulated output for the potential fits Figure 2's action potential very well. We start with depolarization and repolarization, and see our system enter a refractory period at $t = 5$, and then it decays to its steady state. We also see that the potassium and sodium currents spike when the stimulus is applied, and then gradually decay over time.

It is interesting to see that the plots of m, n, and h against the membrane potential are almost cyclic, indicating that there is some sort of dependent pattern between the membrane's potential and how the activation/inactivation of the channels. We also notice that these plots are multi-valued, suggesting that the activation does not depend only on the potential, but also on its previous values. We will see that some interesting limit cycles emerge from this idea when we apply more complicated stimulus pulses.

2.2 Variation of Parameters in the Hodgkin-Huxley Model

To get a better understanding of what effect each of the parameters in the Hodgkin-Huxley model has on the system, here, we vary each parameter while holding the others constant and examine the effect on the system

2.2.1 Input Stimulus

Here, we vary the input stimulus from a single square pulse to a double square pule, a Gaussian pulse, and then a repeated pulse to examine the effects on the system.

Double Square Pulse

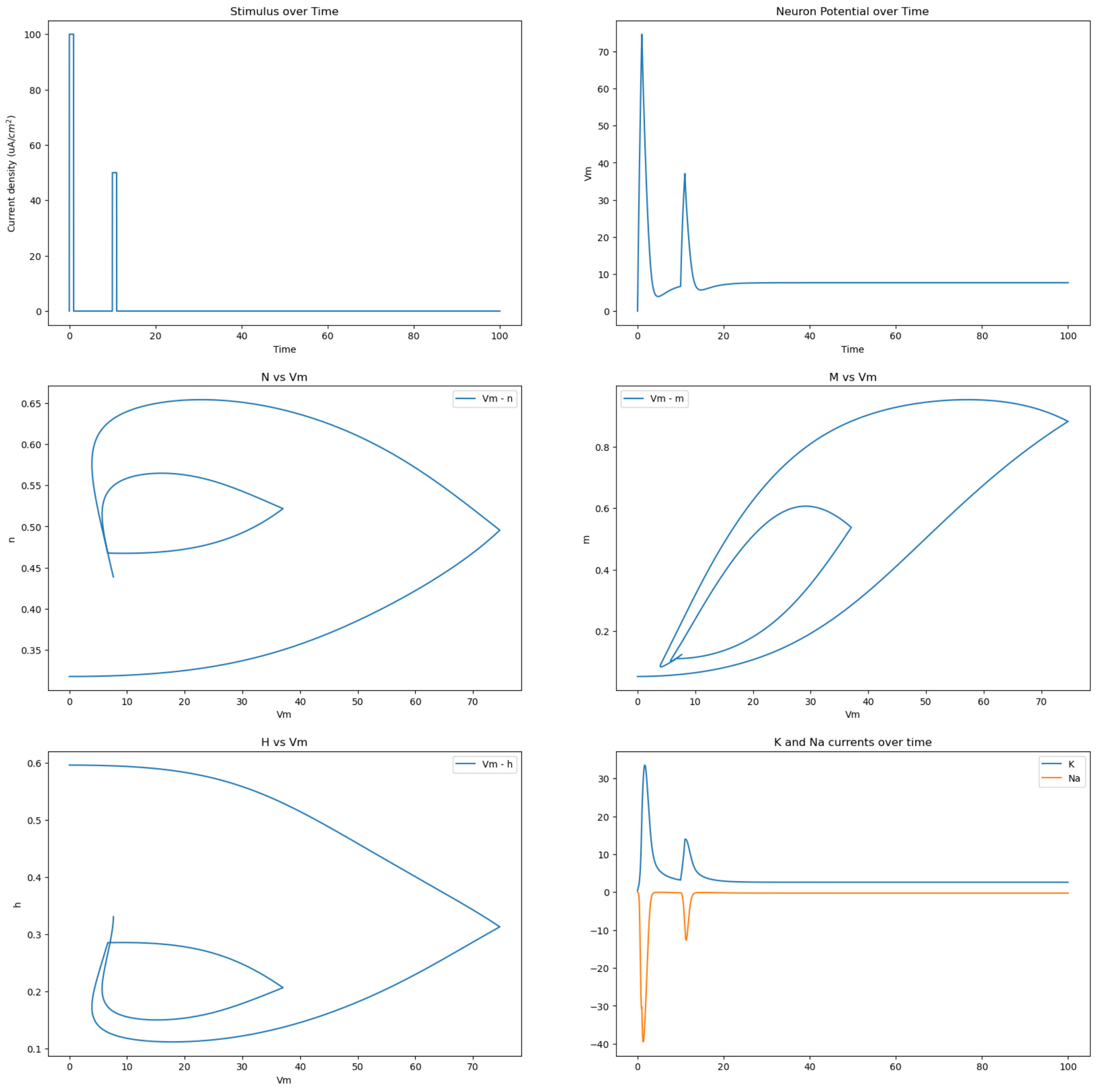


Figure 8: Same plot as Figure 5 but with a different input pulse

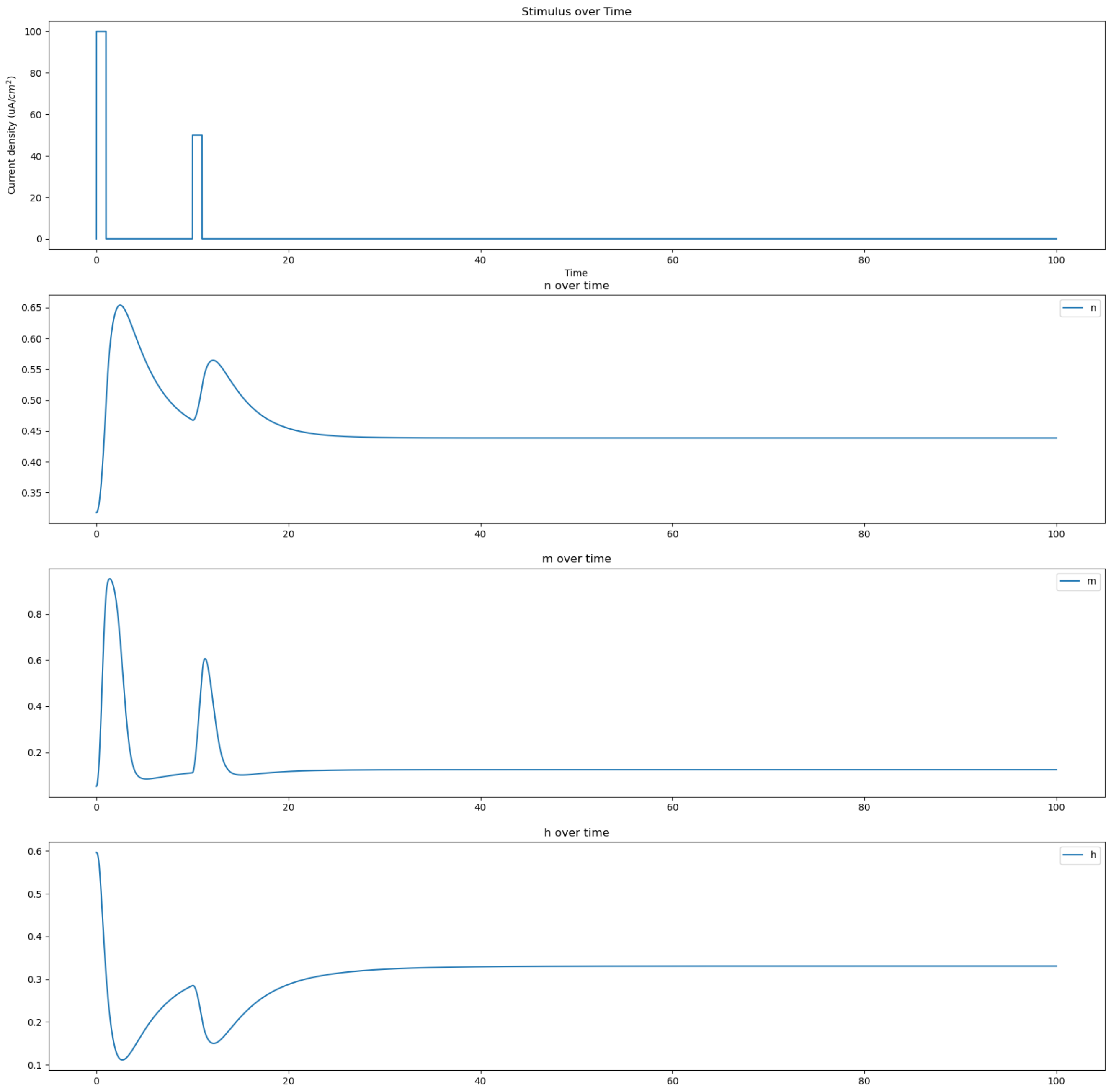


Figure 9: Same plot as Figure 6 but with a different input pulse

Gaussian Pulse

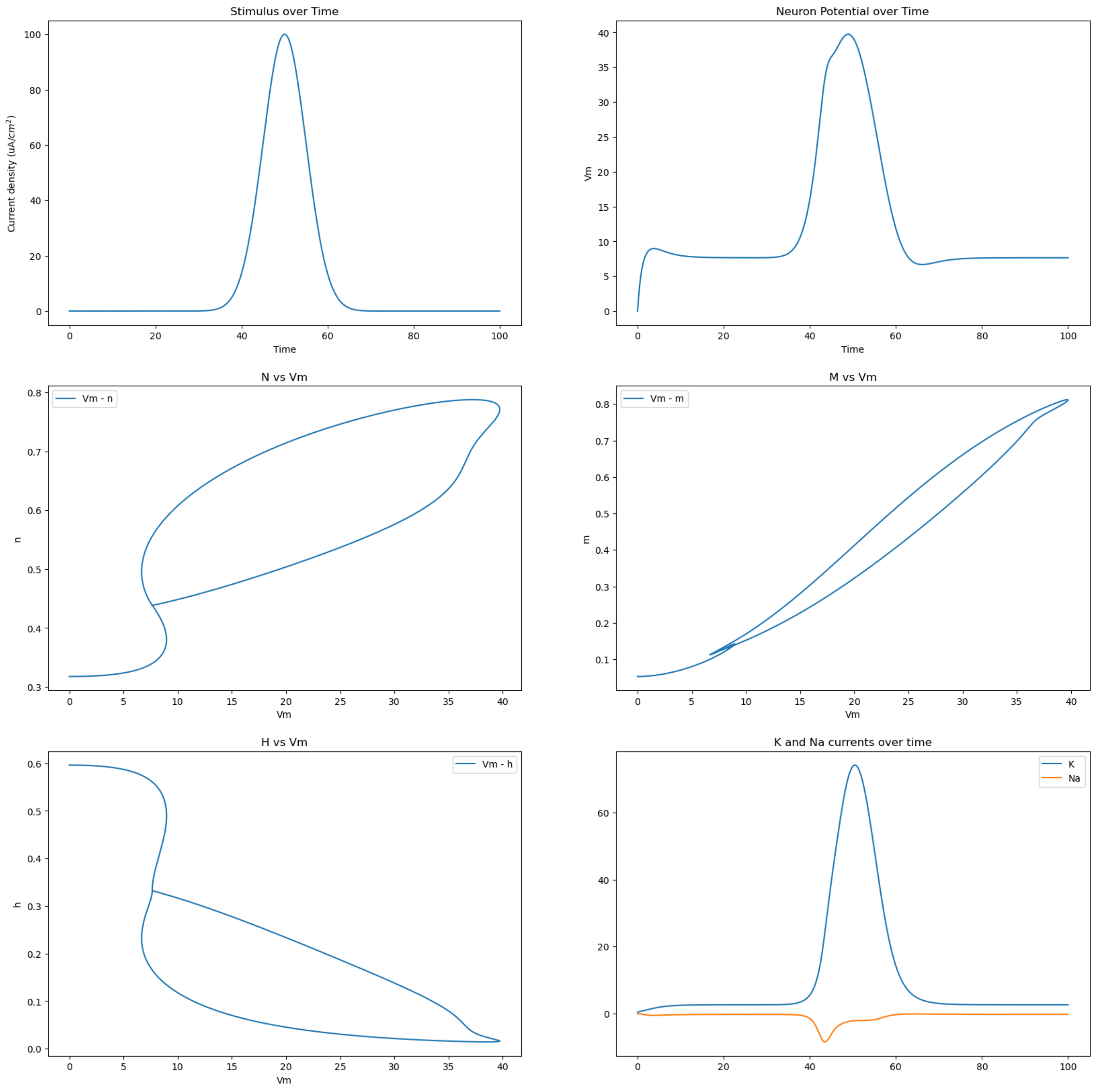


Figure 10: Same plot as Figure 5 but with a different input pulse

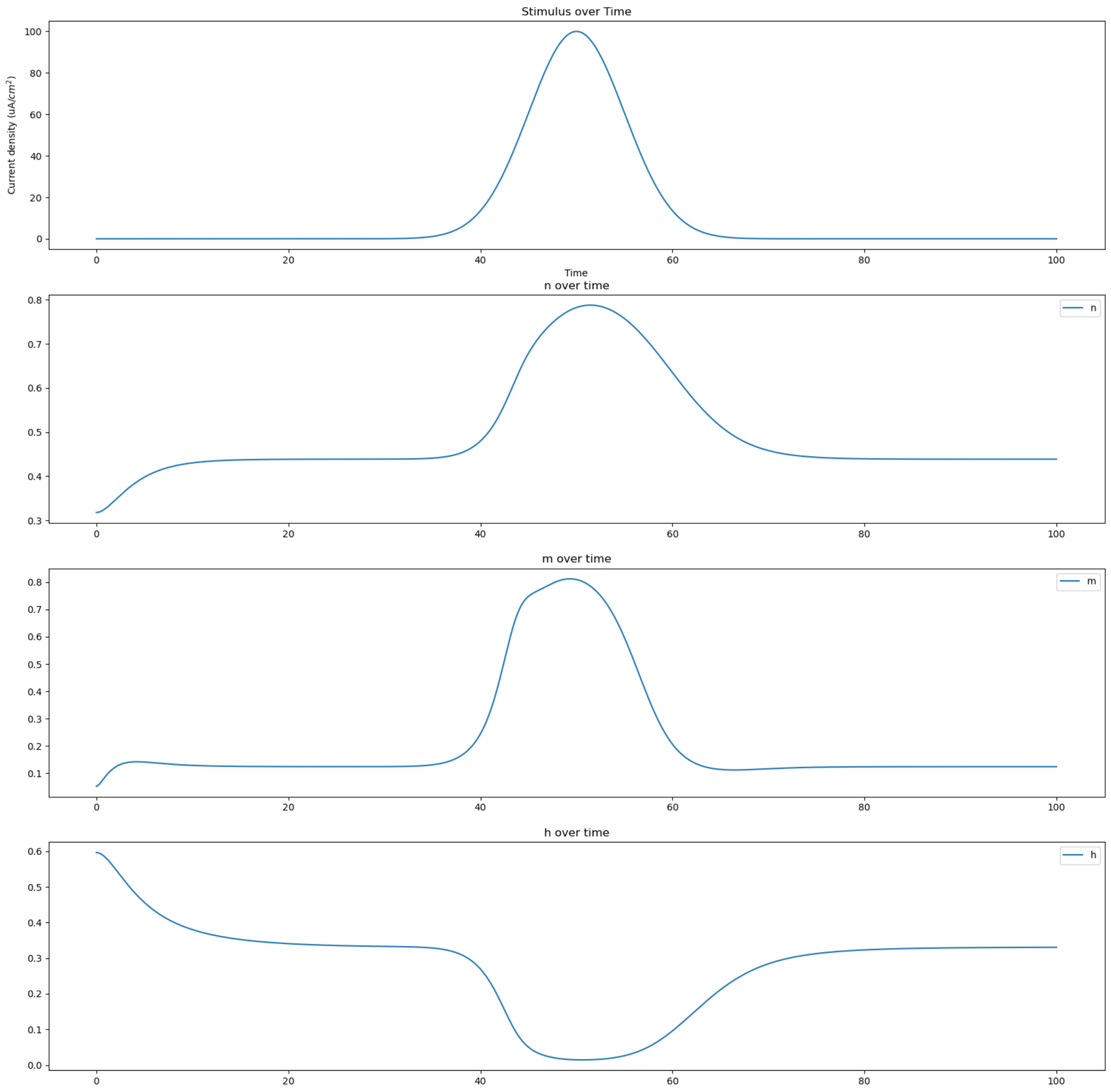


Figure 11: Same plot as Figure 6 but with a different input pulse

Repeated Square Pulse

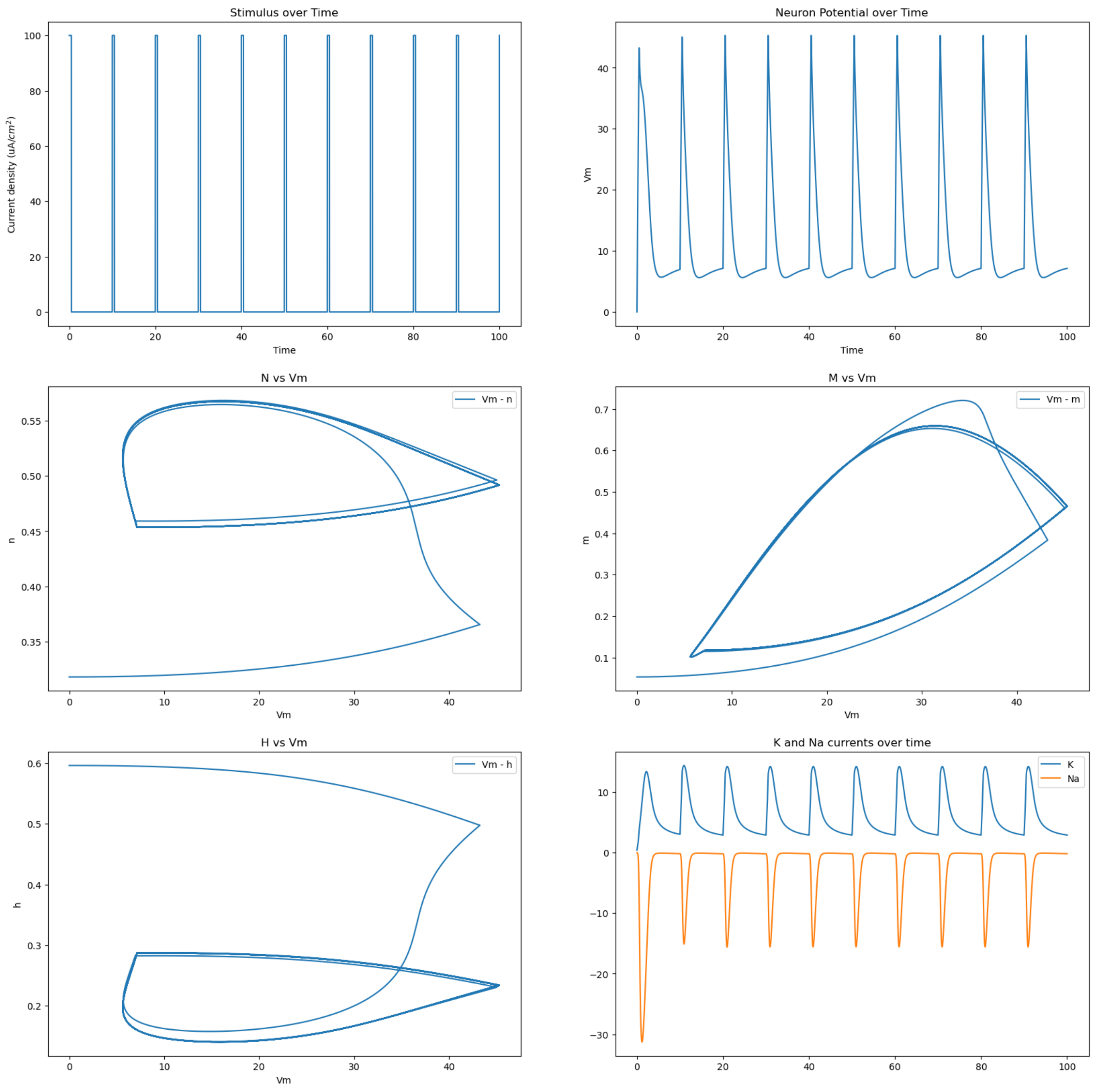


Figure 12: Same plot as Figure 5 but with a different input pulse

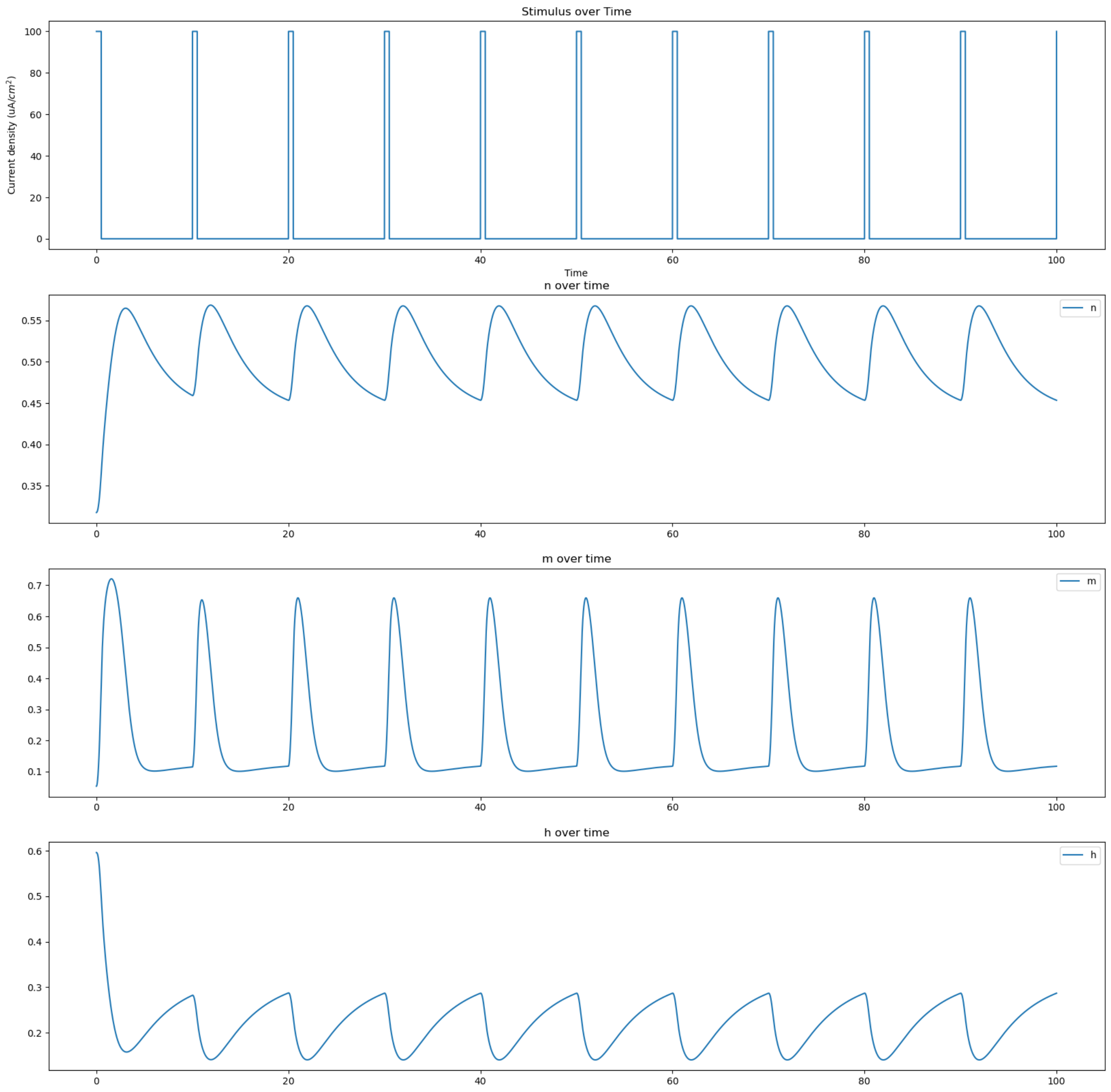


Figure 13: Same plot as Figure 6 but with a different input pulse

It is evident that the input pulse has huge impacts on the system. The potential of the system seems to respond directly to the shape and magnitude of the input pulse.

In the two square pulses, the system output is exactly what we expect - two spikes in axon potential, corresponding spikes in activation and inactivation parameters, and an eventual decay to steady state.

The Gaussian input pulse is also shows similar expected behaviour to the square pulses for all plots except for the Potassium and Sodium current plots over time (Bottom right of figure 9). We notice that in the plots with square pulses, the potassium and sodium currents are relatively equal in magnitude and opposite in sign. However, for the Gaussian pulse, the magnitude of the sodium peak is much smaller than potassium peak in current, and is shifted farther to the left. It seems that the Sodium current is being damped at a much more fast rate than the potassium current, which implies that the sodium resistance is higher than the potassium resistance.

Finally, we notice that when running the repeated pulse, m, n, and h, enter limit cycles over time and do not vary much from these cycles. This can be seen in the potassium and sodium currents over time, as well as in the axon response to the stimulus, which both stabilize and become periodic.

2.2.2 Membrane Capacitance

We recognize that the most important plot to understand and display is the membrane potential over time. So, when we vary the following parameters, we will plot only the neuron potential for various values of the parameters.

In this section, we vary the membrane capacitance (C_m) to see what effects it has on the system.

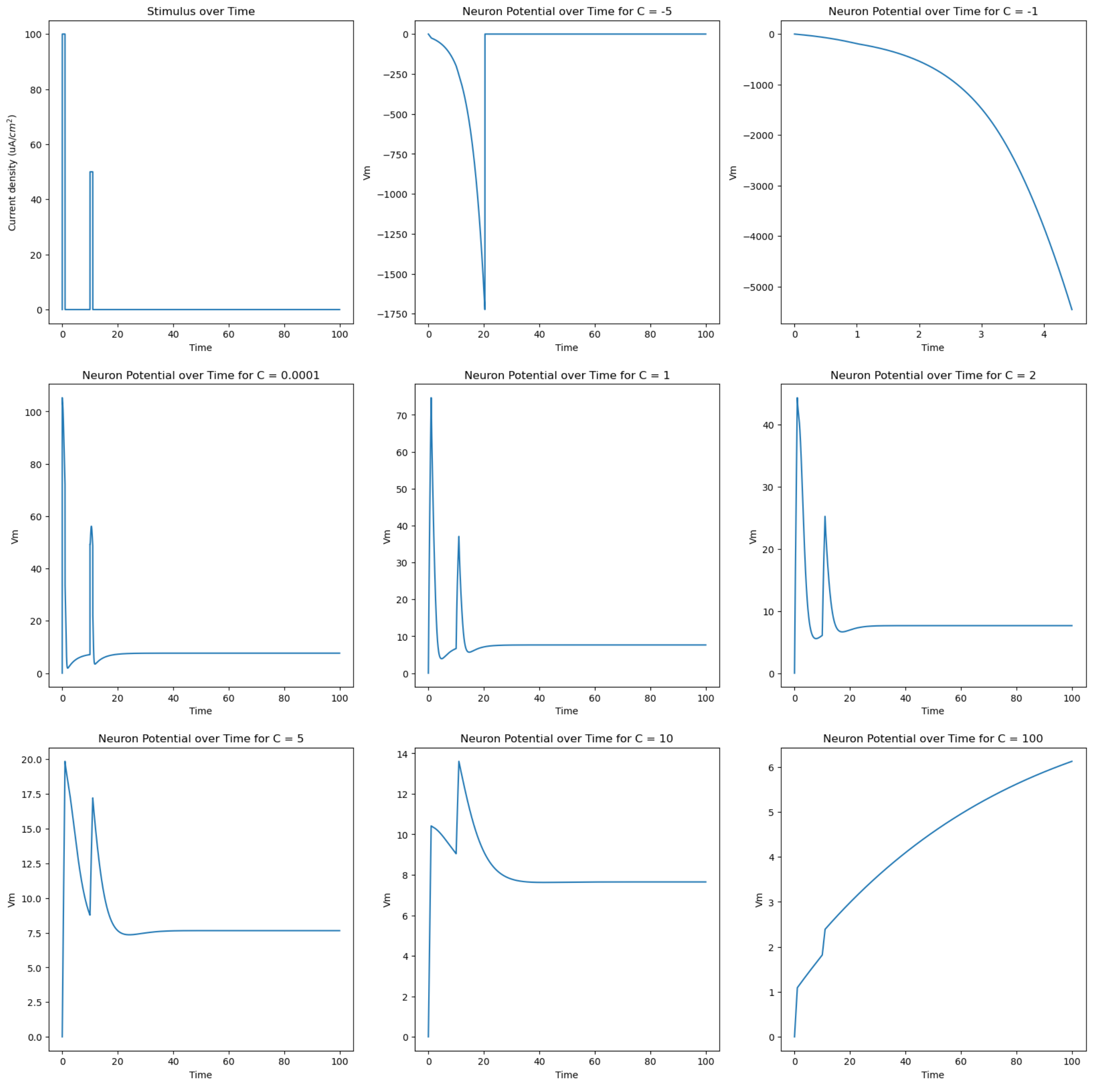


Figure 14: Plot of membrane potential response to a double square pulse while varying membrane capacitance from C = -5 to C = 100

We see that the membrane capacitance has an impact on the magnitude of the potential response function. We notice that as we increase the value of the capacitance, the magnitude of the potential decreases. However, the capacitance does not affect the steady state of the membrane potential for times long after the pulse was applied when the capacitance is within normal ranges. In the differential equation model of the system, we see that C_m appears as a factor in equation 7. If we rearrange the equation to solve for $\frac{dV_m}{dt}$, we see that the entire equation is multiplied by a factor of C_m. This is consistent with the results we simulated, in that C_m acts like a multiplicative factor on the magnitude of the response, but doesn't change its shape within small ranges.

We can conclude that the capacitance of a cell does not have a large impact on the equation, and should be chosen to most accurately fit experimental data.

2.2.3 Potassium Conductance

We vary the potassium conductance and plot the results.

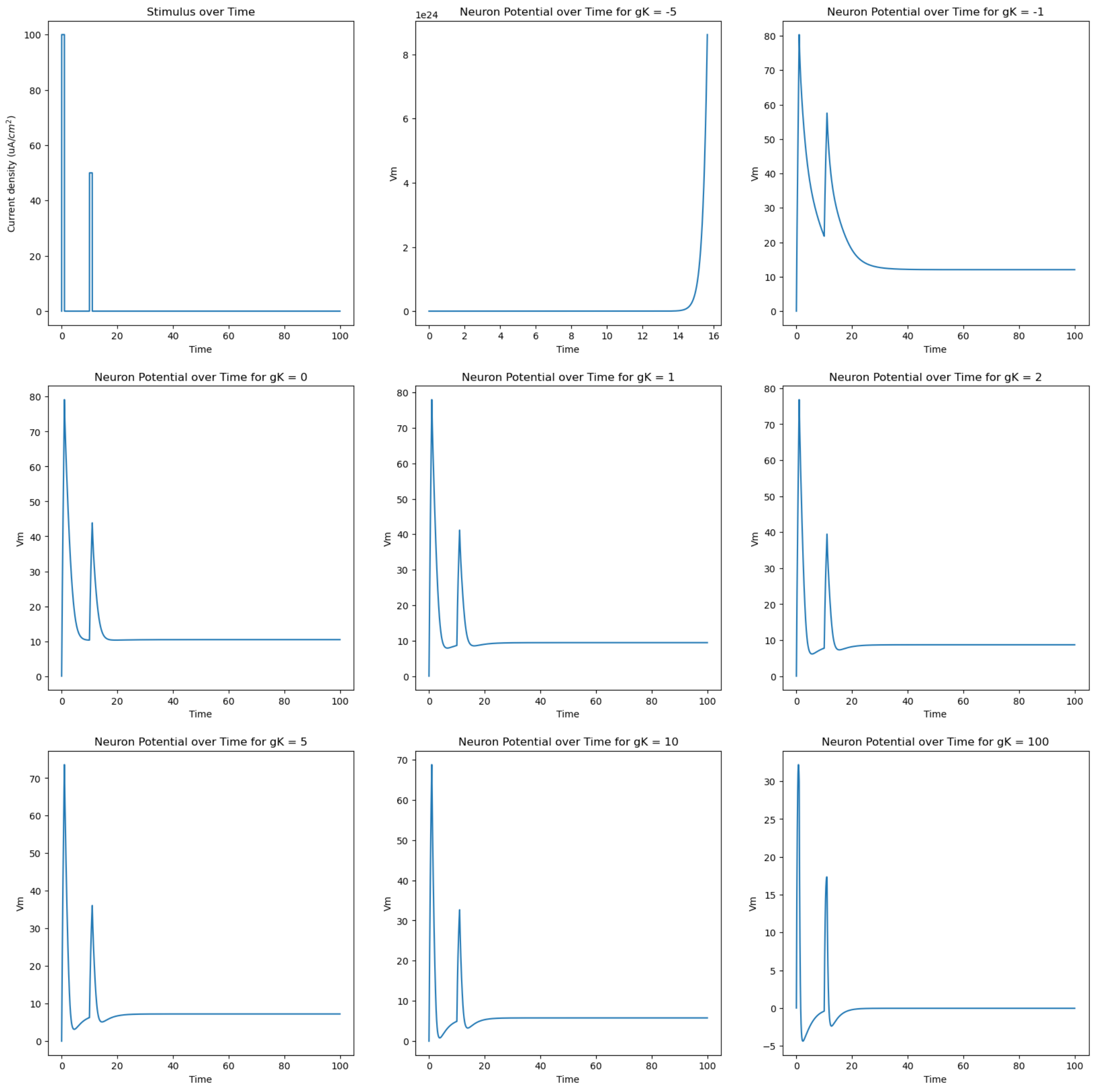


Figure 15: Plot of membrane potential response to a double square pulse while varying potassium conductance from g_k = -5 to g_k = 100

We notice that the potassium conductance has more nontrivial effects on the system than the membrane capacitance. Immediately, we see that increasing g_k decreases the potential response slightly, but also has an interesting affect on the refractory period. Indeed, we see that when $g_k \leq 0$, there is no refractory period in the system's response, and that increasing g_k increases the magnitude and length of the refractory period.

2.2.4 Sodium Conductance

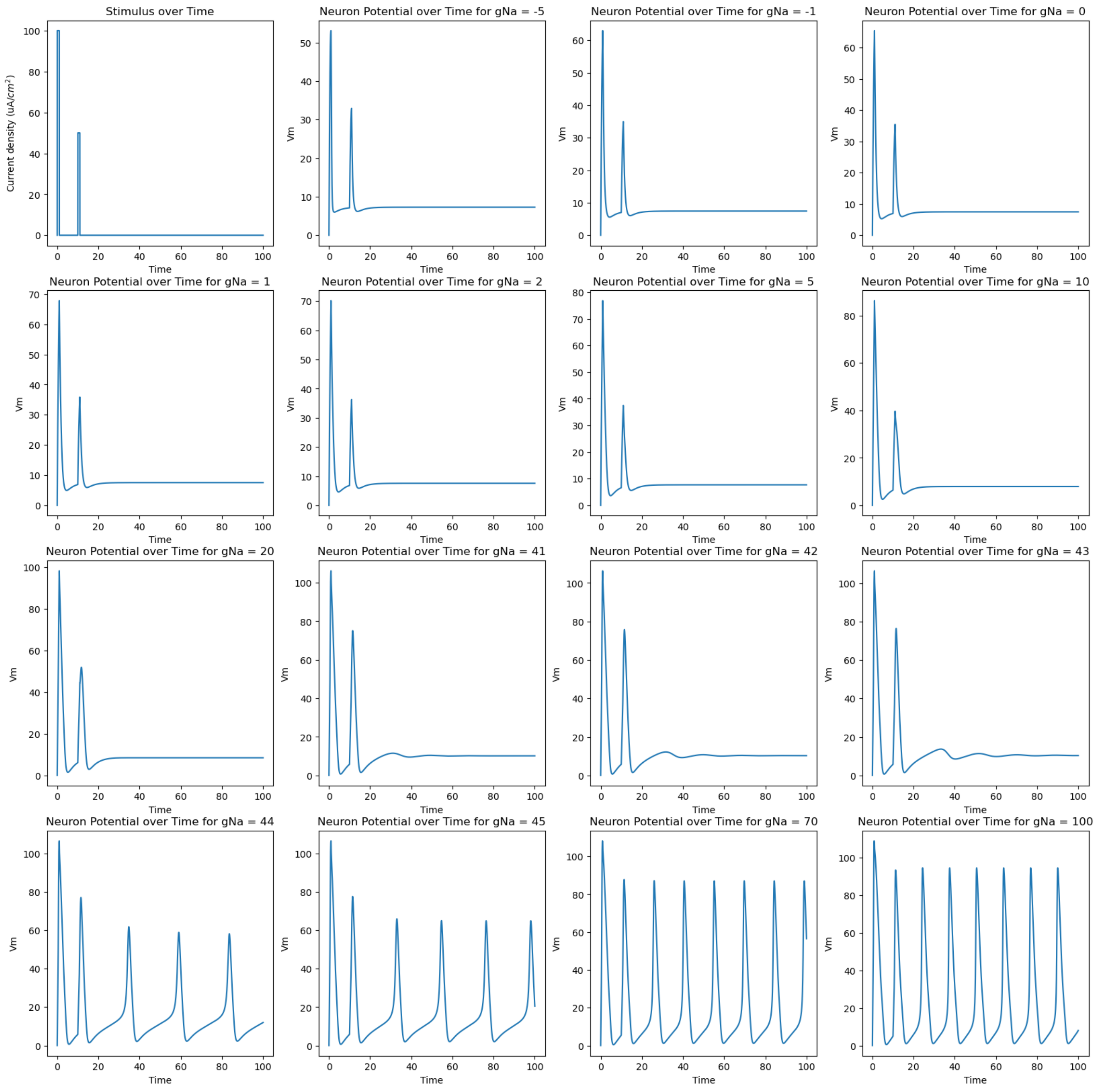


Figure 16: Plot of membrane potential response to a double square pulse while varying sodium conductance from g_{Na} = -5 to g_{Na} = 100

Examining g_{Na}'s effect on the system is quite interesting. Until $g_{Na} = 41$, we see that increasing g_{Na} increases the magnitude of the membrane potential spike and increases the potential dip during the refractory period. However, once we have $g_{Na} = 41$, we see that the refractory period does not reset to the stable state but actually crosses it and then approaches it slowly. Increasing g_{Na} more, we see that this behaviour increases, and it causes periodic spikes to emerge. by $g_{Na} = 45$. it is clear that we have some periodic behaviour emerging, which is even more evident when we show $g_{Na} = 100$.

This is quite odd, as the system is behaving as if it has a repeated pulse (like in Figure 11) while only being driven twice. This suggests that there is some bifurcation happening at this point, where the system goes from falling into a stable fixed point to entering a limit cycle. We know that the current through the neuron, and thus the potential, is determined by the activation and inactivation of the channels in and out of the neuron. So, to further examine this bifurcation, we plot the activation parameters m, n, and h over time and then against the membrane potential.

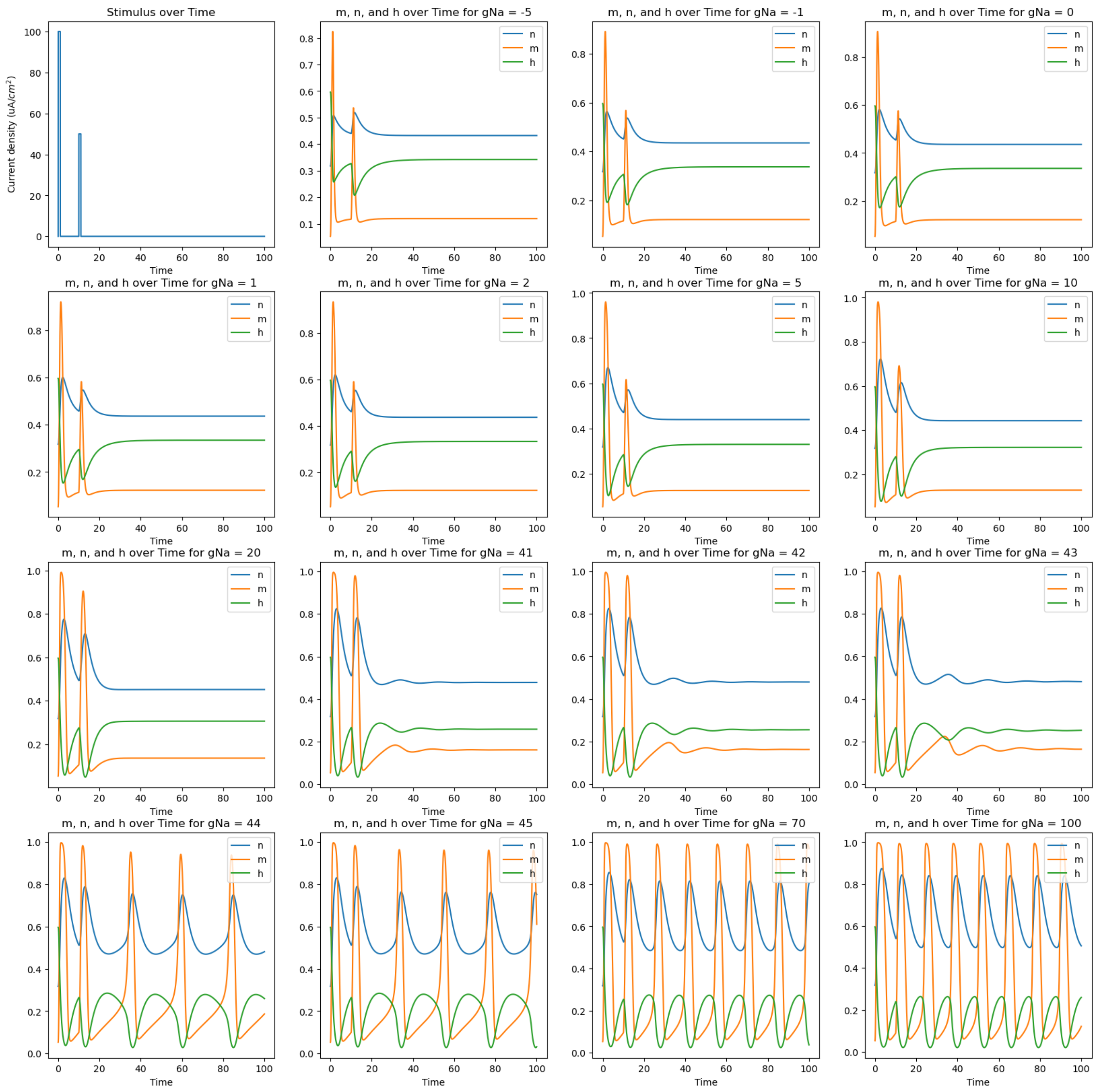


Figure 17: Plot of m, n, and h against time in response to a double square pulse while varying sodium conductance from g_{Na} = -5 to g_{Na} = 100

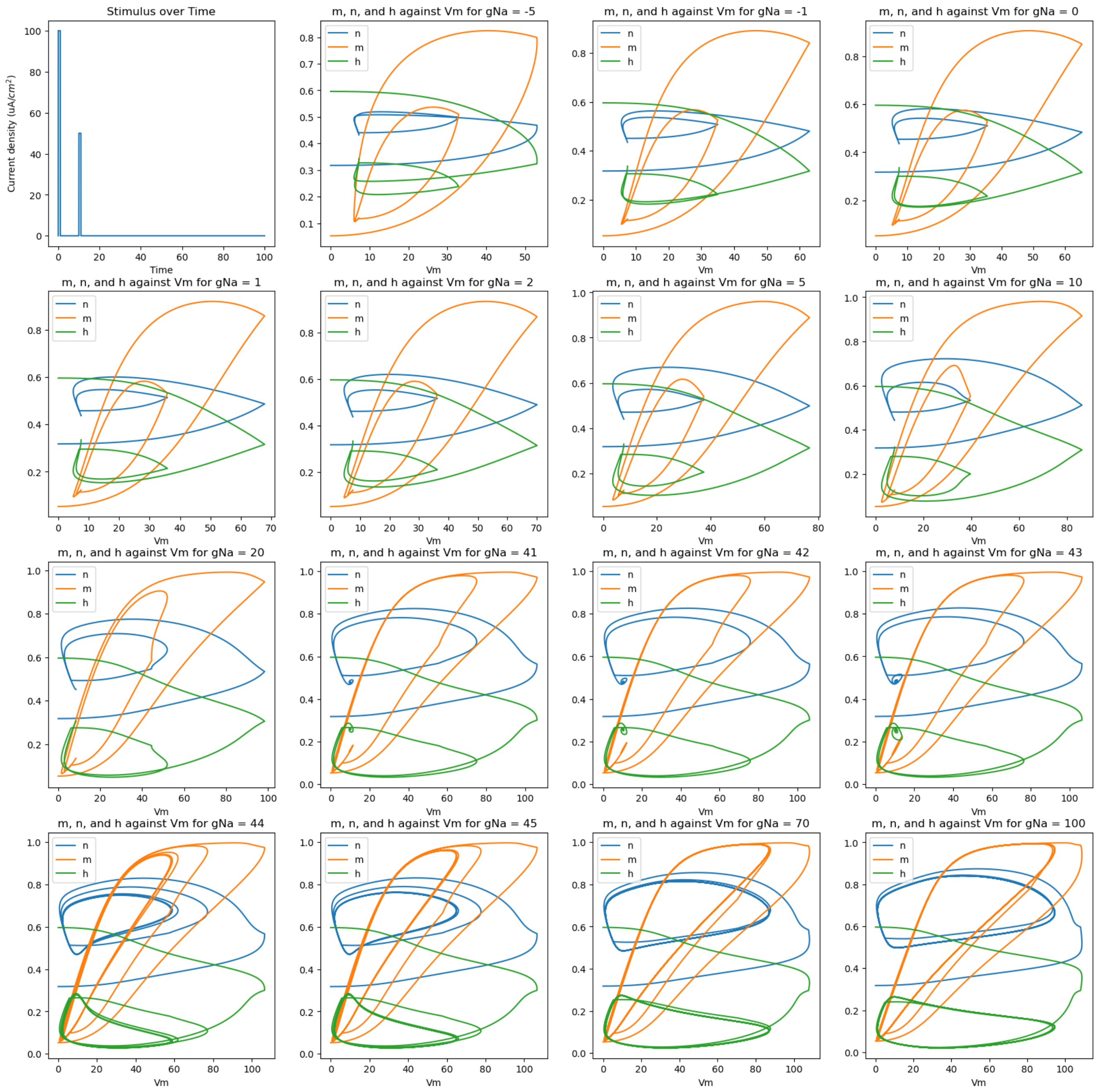


Figure 18: Plot of m, n, and h against membrane potential in response to a double square pulse while varying sodium conductance from g_{Na} = -5 to g_{Na} = 100

In figure 16, we see that there is clear evidence of the activation and inactivation parameters becoming periodic from $g_{Na} = 41$ to $g_{Na} = 45$. In figure 17, we can actually see the limit cycles that these parameters enter. Notice that for low g_{Na}, there is no limit cycle behaviour. However, it begins to emerge as we increase g_{Na} higher.

2.2.5 Potassium Reverse Potential

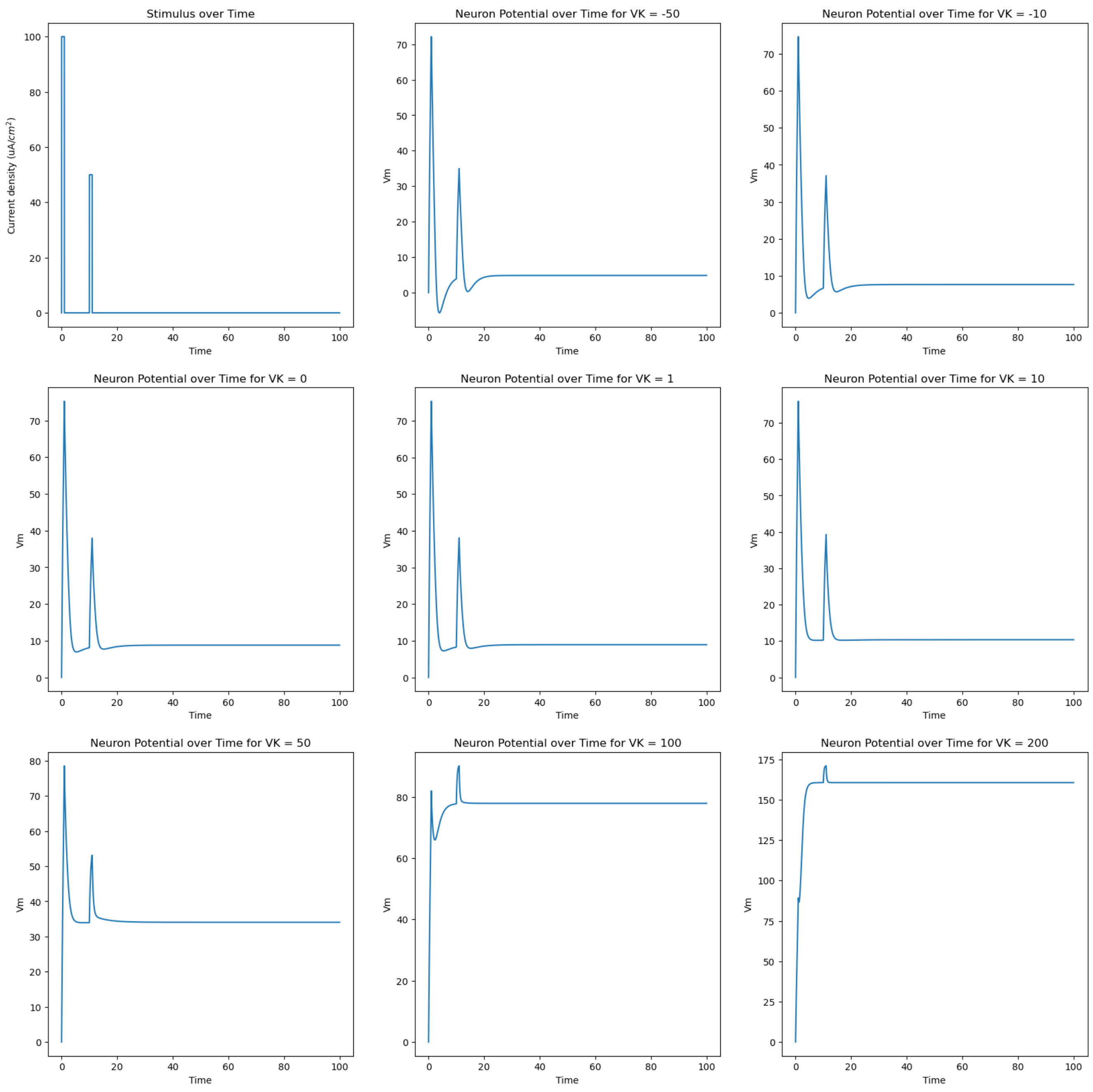


Figure 19: Plot of membrane potential in response to a double square pulse while varying potassium reverse potential from g_{Na} = -50 to g_{Na} = 200

It's evident that the potassium reverse potential mainly affects the refractory period, in that a lower potassium reverse potential means that the refractory period is greater. Indeed, when V_k gets large enough, our refractory period becomes so large that it dominates the system's response.

2.2.6 Sodium Reverse Potential

Varying V_{Na} has the opposite effect, as is predicted because the current is opposite for the sodium and potassium channels. When we increase V_{Na}, we actually see that the refractory period increases, as is clear from Figure 19.

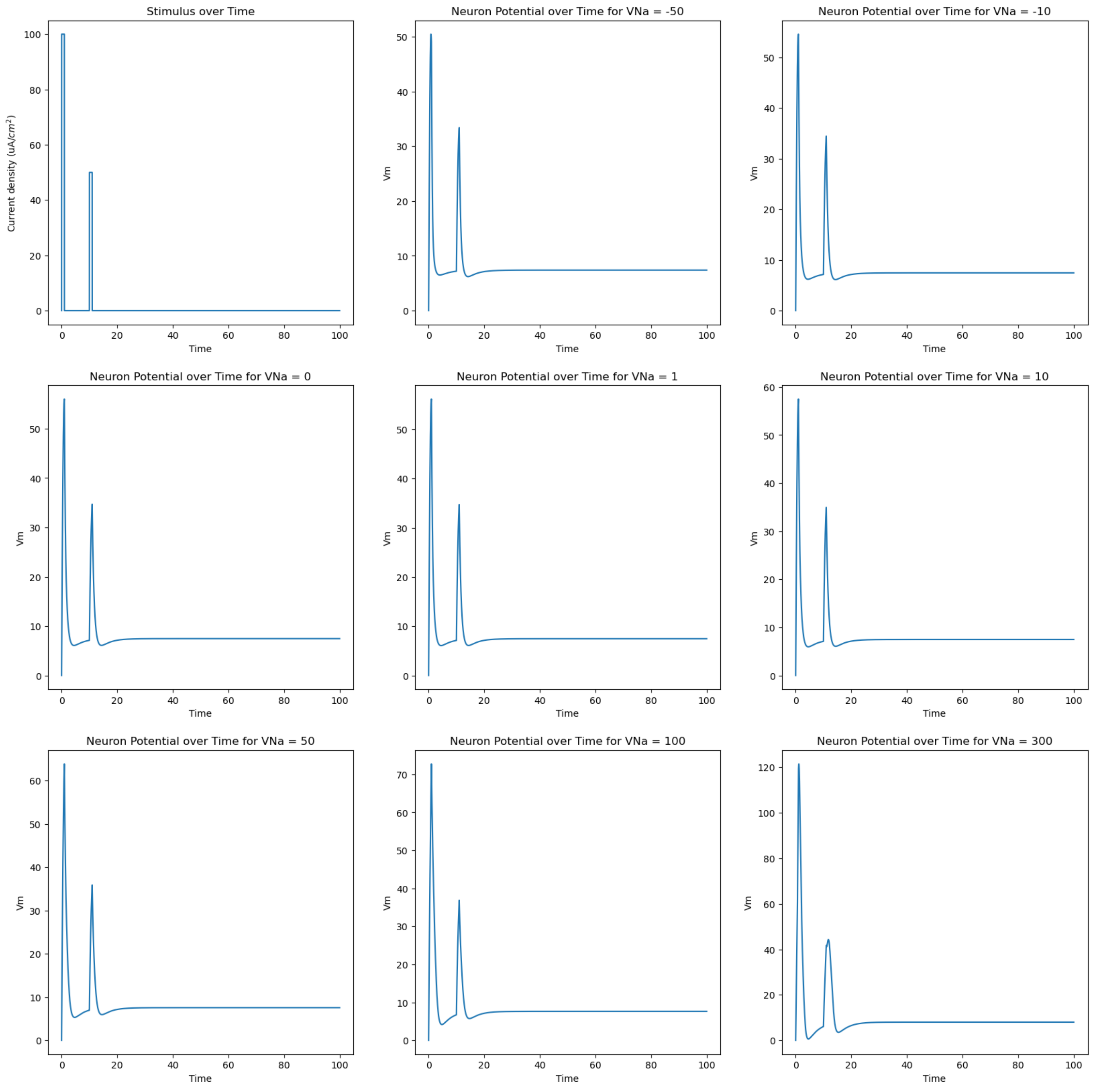


Figure 20: Plot of membrane potential in response to a double square pulse while varying sodium reverse potential from g_{Na} = -50 to g_{Na} = 300

3 Parakh-Varanasi: Hodgkin-Huxley Neural Network

With our understanding of the dynamics of Hodgkin-Huxley neurons, our next step is to construct a neural network of Hodgkin-Huxley neurons with the attention of elucidating neural circuit function. We design our model as follows:

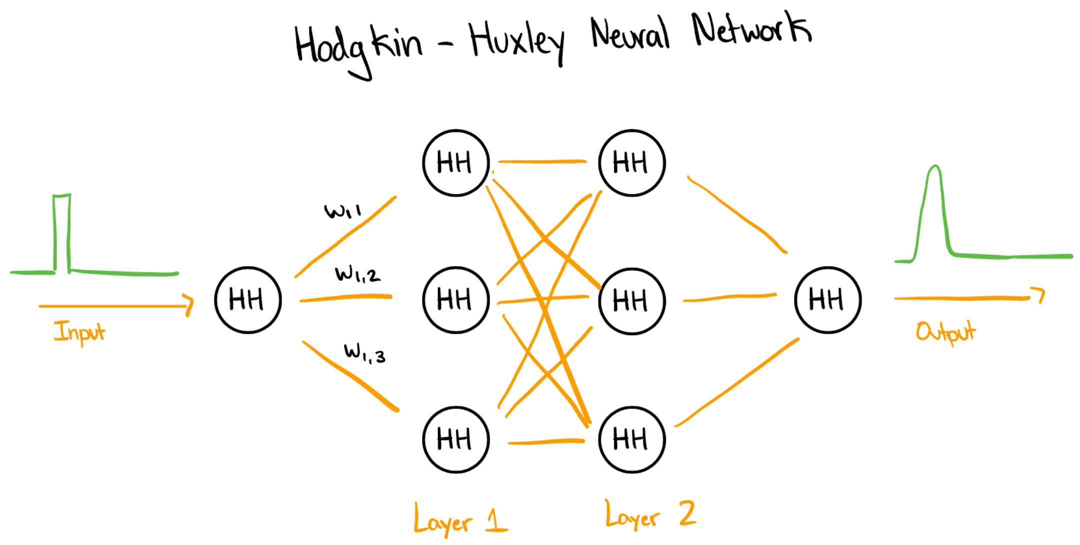


Figure 21: Parakh-Varanasi Diagram

At a high level, our models takes an input stimulus which is processed by an initial Hodgkin-Huxley neuron. The resulting neuron potential is then translated to a current via a simply Ohm's law relation: V = IR and transmitted along to the first layer of neurons. The diagram above depicts a network with 3 nodes in the first layer and 3 nodes in the second layer; however, the model is generalizable to n layer 1 neurons and m layer 2 neurons. The transmitted signal is then scaled by a weight parameter unique to each neural connection. The new scaled connection is then processed by each neuron in layer 1 and transmitted to layer 2. Once again, these signals are re-scaled by an additional weight parameter unique to each connection. For neurons recieving multiple signals, the signal is additive so that the processed input stimulus is simply a sum of the total input signals. Finally, the layer 2 neurons transmit their scaled signals to the final HH neuron which produces a final output signal.

3.1 Implementation of the Parakh-Varanasi Model

3.1.1 Parakh-Varanasi Model Structure

A full write up the code can be found in Appendix F: Parakh-Varansi Implementation, but this section will provide a brief walk through of the parameters and function of the model.

Inputs

- input_signal (10,000 x 1): Vector of electrical stimulus over time
- params (n x 1): Vector corresponding to network weights
- input_name (string): Plot label
- l1_size (integer): # of neurons in 1st layer
- l2_size (integer): # of neurons in 2nd layer

Following the Parakh-Varansi model the script uses the inputted parameters to produce an output signal and a series of plots describing the function of the network. In particular, the script produces a plot of input vs. output signal for each neuron in the network. Additionally, the script produces phase portraits for each opening/closing probability n,m, and h as a function of cell potential.

For a simple Gaussian pulse inputted into a network of 3 layer 1 nodes and 3 layer 2 nodes with uniform weights, the following outputs are produced:

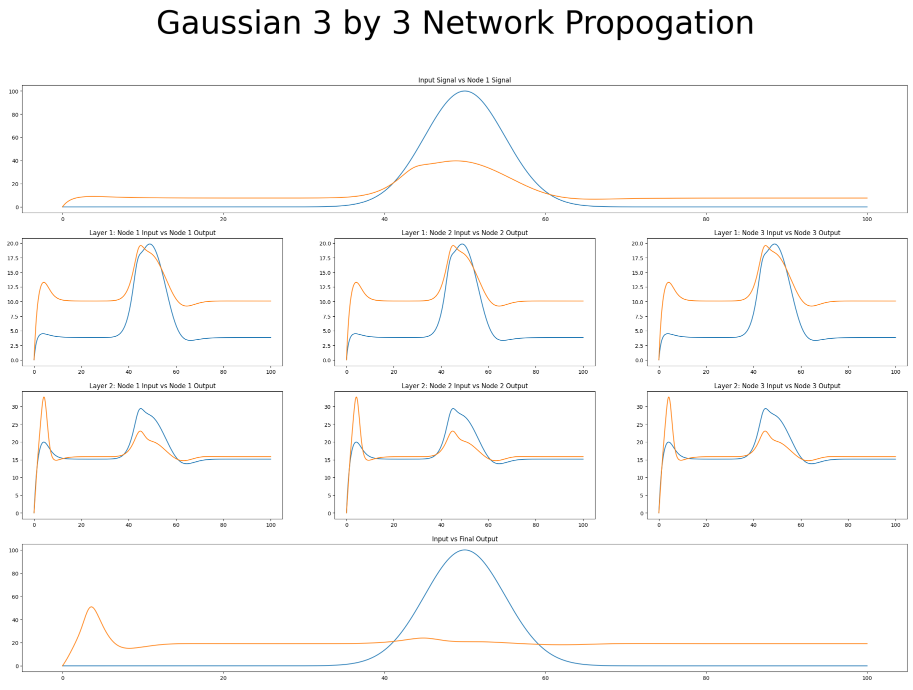


Figure 22: Uniform weight w/ Gaussian Stimulus

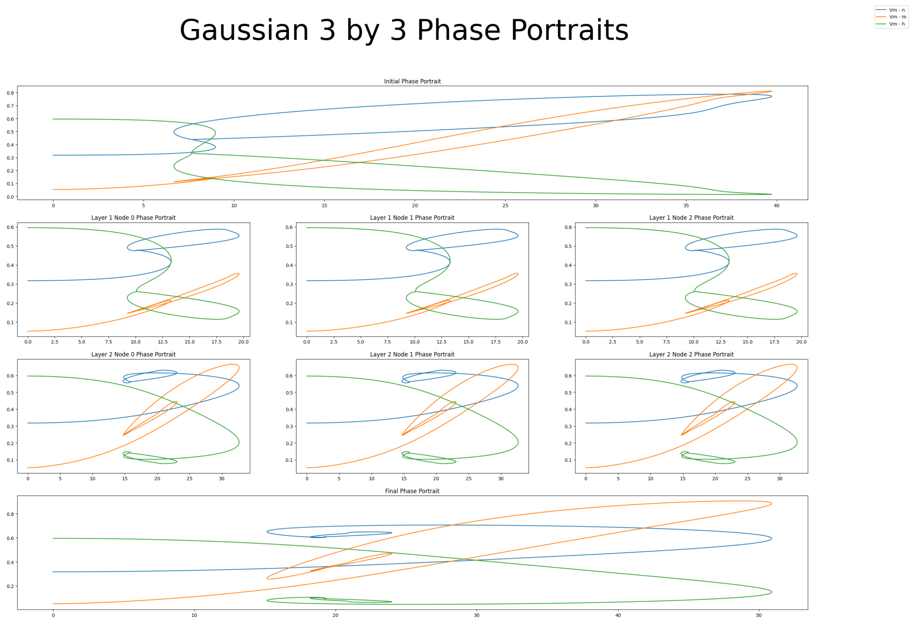


Figure 23: Phase Portrait of Uniform weight w/ Gaussian Stimulus

3.2 Input Pulses

With our constructed model, the first dimension we explore is how our model processes various input stimuli. We explore the same stimuli explored in the initial phase of this project.

3.2.1 Gaussian Pulse

The first stimulus we explore is the simple gaussian pulse.

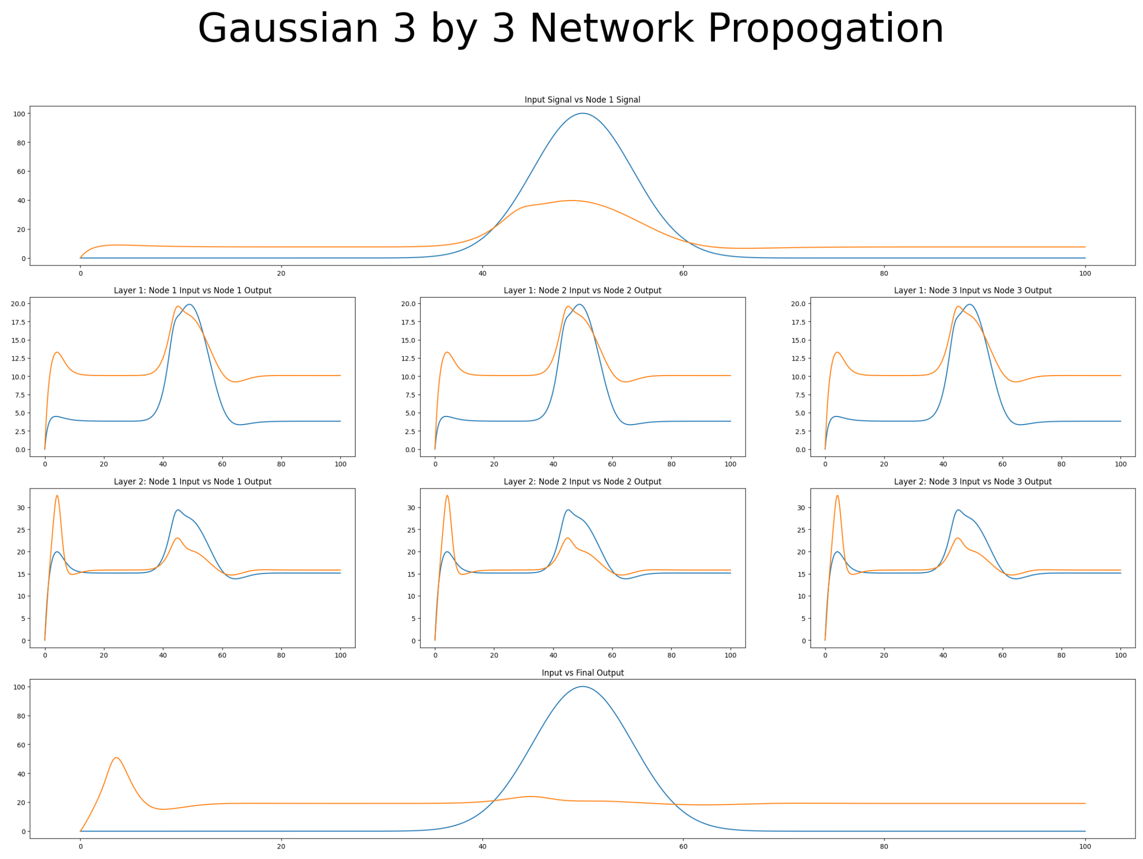


Figure 24: Uniform weight w/ Gaussian Stimulus

As you can see in the final and bottom-most plot, the Parakh-Varanasi network shifts the output peak from 50 seconds to approximately 5 seconds. In each of the preceding panels, you can notice the gradual development of initial impulse, with a smaller initial peak appearing in the first layer and eventually reaching its final form in the output. The occurrence of this signal can be explained by the small non-zero impulse exerted by the early stages of the Gaussian stimulus. The incoming current excites the cell to produce a potential change which is then translated forward as a larger current. With each subsequent layer of the network, this signal is amplified. As a consequence of this initial peak, the neuron enters its refractory period during which it is unable to respond to incoming signals. For this reason, we see that the initial Gaussian impulse is reduced in each layer of the network. With this, we can conclude that the network did not shift the peak, but rather produced a separate initial peak and smoothed out the original stimulus.

3.2.2 Square Pulse

The next stimuli we explore is the simple square pulse.

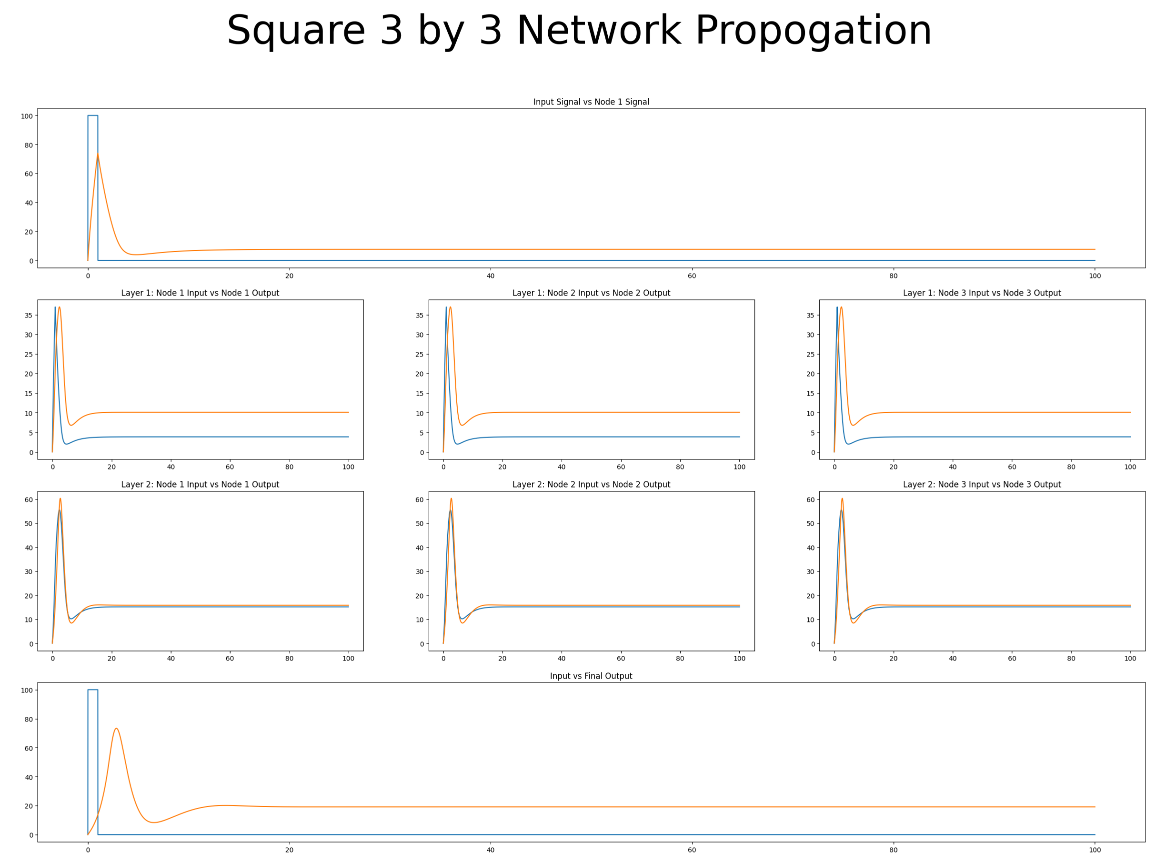


Figure 25: Uniform weight w/ Square Stimulus

We see that Parakh-Varanasi model translates the initial square stimulus into a shifted gaussian peak. In each of the individual neurons we notice that the produced peak closely emulates the provided signal; however, through the additive smoothing effects of the network, the final signal produced has a smaller magnitude and is more widely distributed than the original impulse.

3.2.3 Square Double Pulse

The next signal we explore is the square double pulse.

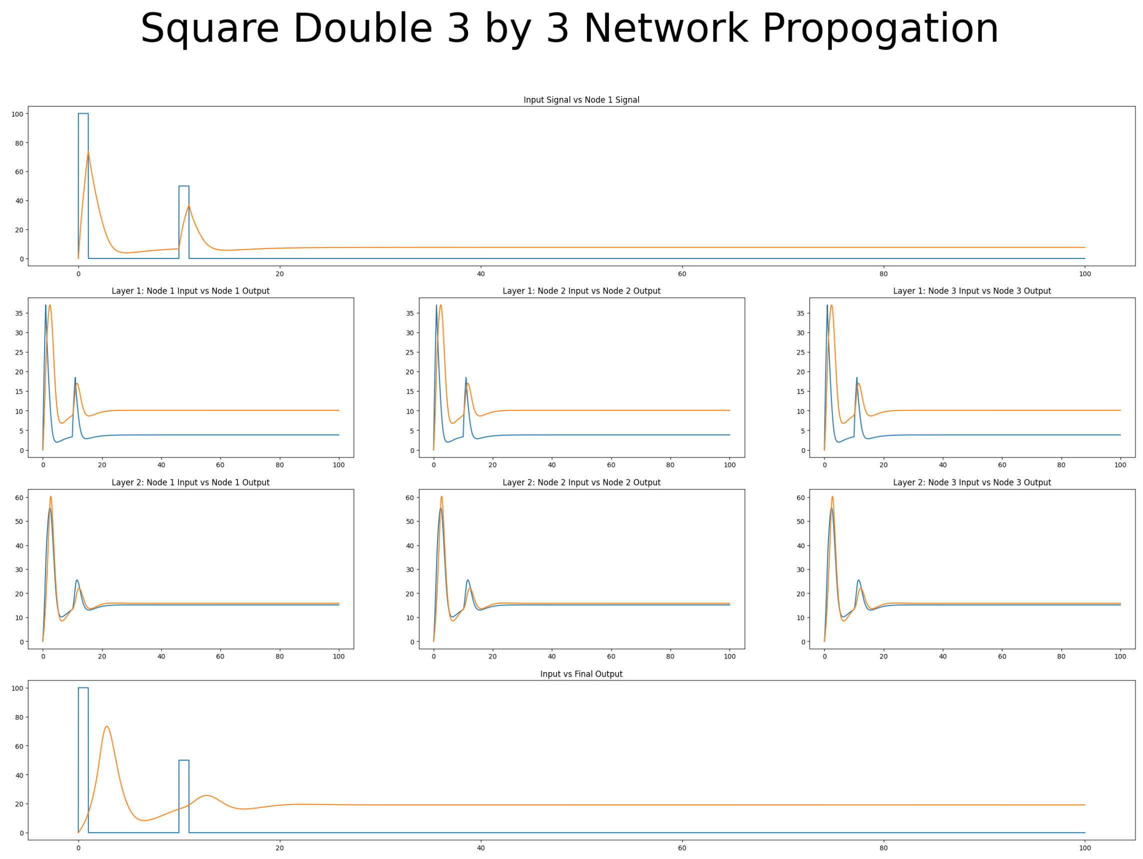


Figure 26: Uniform weight w/ Square Double Stimulus

Much like the single square pulse, we see that the individual neurons are able to aptly reproduce their provided input signals; however, as a whole the network reduces the magnitude and shifts the peak of the stimulus forward. One important feature to note is that the initial peak produces a much stronger output impulse, but the second and subsequent output peak is relatively smaller than the provided second stimulus peak. This action can once again be explained by the refractory period of the neuron during which it is less responsive to stimuli. Since the neuron was initially excited and has not had time to return to original conditions, subsequent excitements produce muted responses. We continue to notice this phenomena in our next phase, the repeated square pulse.

3.2.4 Repeated Square Pulses

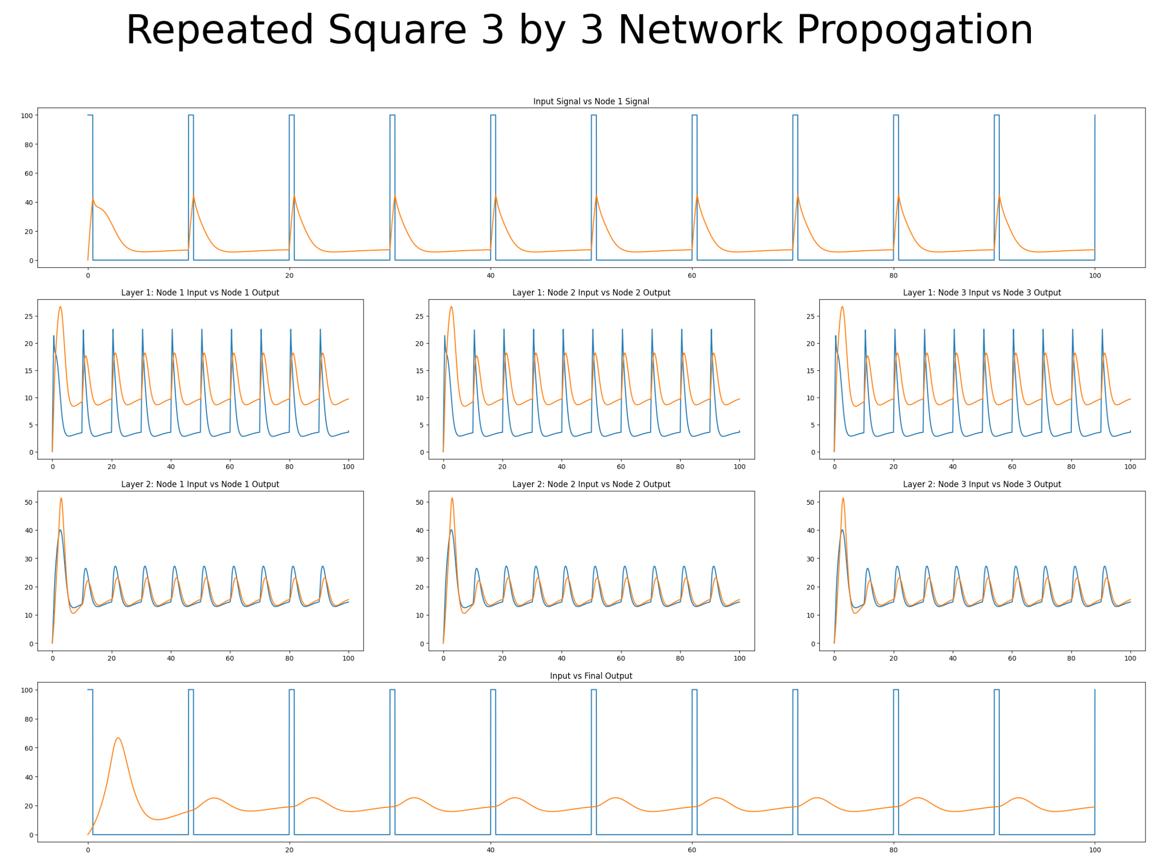


Figure 27: Uniform weight w/ Repeated Square Stimulus

Similar to the square double pulse, we see that the action of the Parakh-Varanasi model is to translate the inputted square stimuli into wider gaussian peaks that reach their apex a couple seconds after the peaks of the initial inputs. We also continue to notice the refractory phenomena in which responses after the initial stimuli are muted. One feature to note is that the non-preliminary responses are identical in magnitude and spacing indicating that the refractory period effect is equal in magnitude for each excitement.

3.3 Network Architecture

In this section, we explore the impact of network architecture on the signal processing behavior of our model. We label each network architectures n by m with n corresponding to the number of nodes in the first layer and m corresponding to the number of nodes in the second layer of the network. We begin with a recap of the 3 x 3 architecture extensively studied in the previous sections. Once again, we continue with a repeated square pulse for consistency.

3.3.1 3 x 3 Architecture

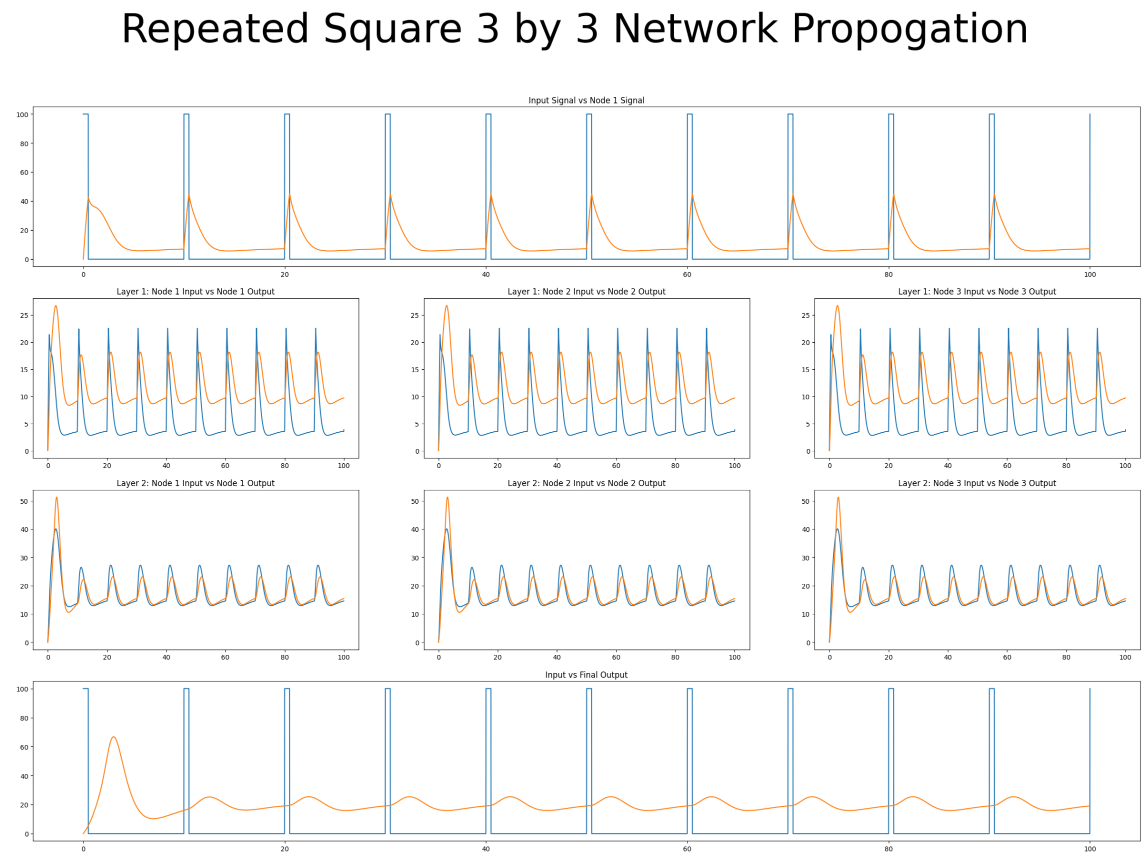


Figure 28: 3x3 Uniform weight w/ Repeated Square Stimulus

3.3.2 3 x 5 Architecture

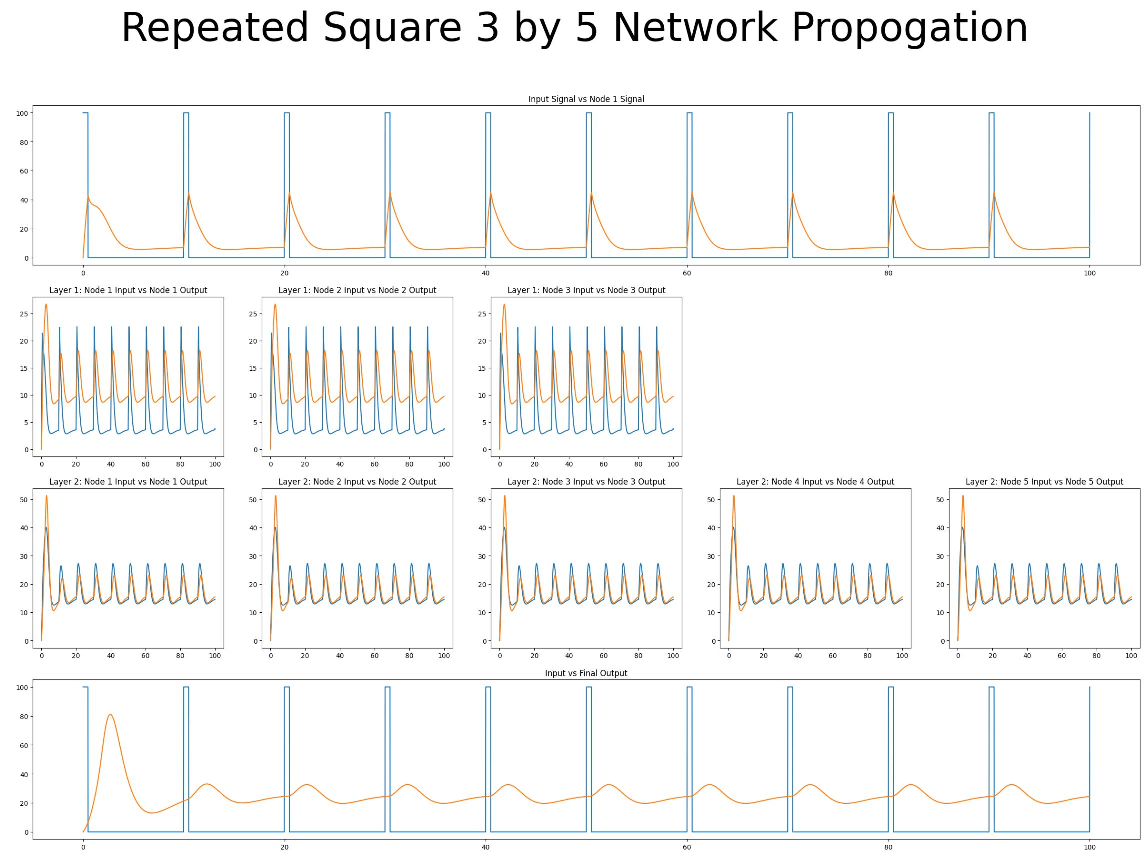


Figure 29: 3x5 Uniform weight w/ Repeated Square Stimulus

In comparison to the standardized 3 x 3 architecture, we see that the new network is able to produce slightly more complex output behaviour. In addition to altering the magnitude, we see that the output peaks are slightly sharper in the 3 x 3 counterpart.

3.3.3 5 x 5 Architecture

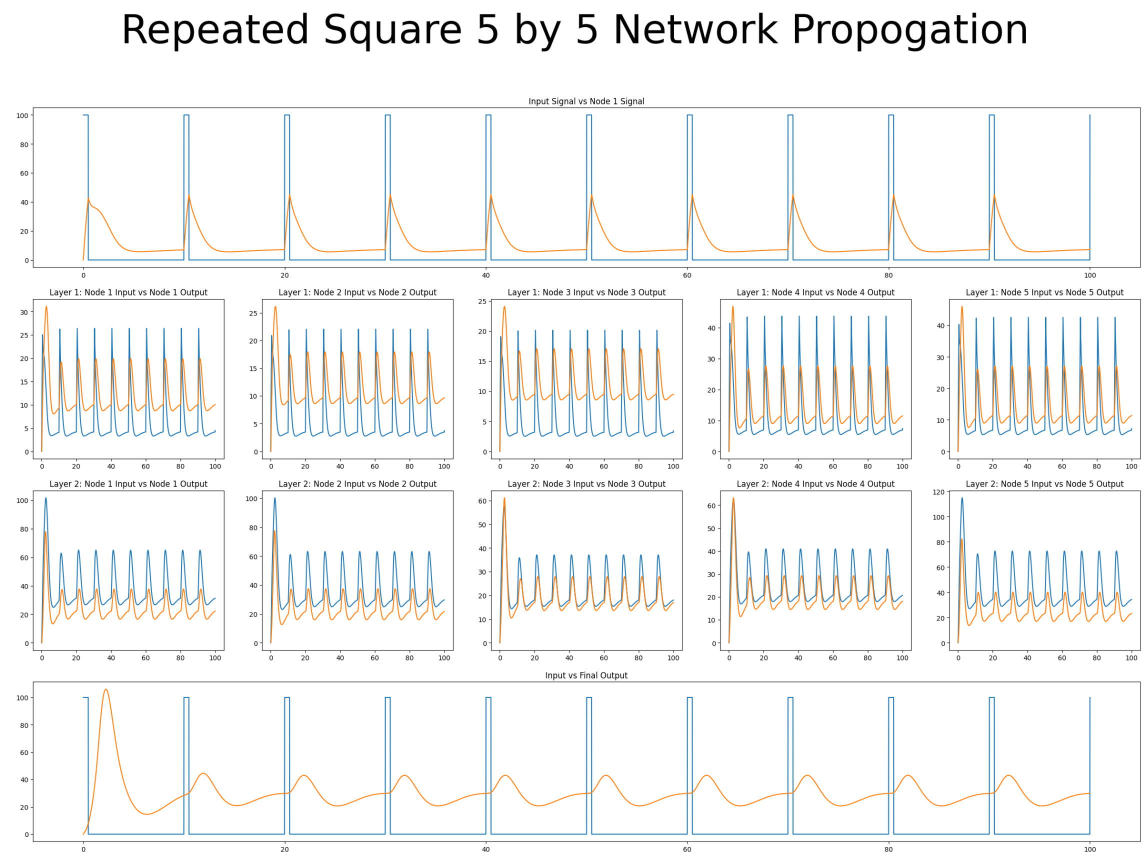


Figure 30: 5x5 Uniform weight w/ Repeated Square Stimulus

Similar to the 3 x 5 architecture, we see that the addition of additional nodes into the model produces more complex output behavior and stronger response peaks.

3.4 Network Weights

With our understanding of how our model responds to different network stimuli and architectures, we begin looking into the effect of network weights. For simplicity, the following section will focus on the repeated square pulse to standardize the comparison between network weights.

3.4.1 Uniform Weights

We begin with a recap of the model with uniform weights for each connection.

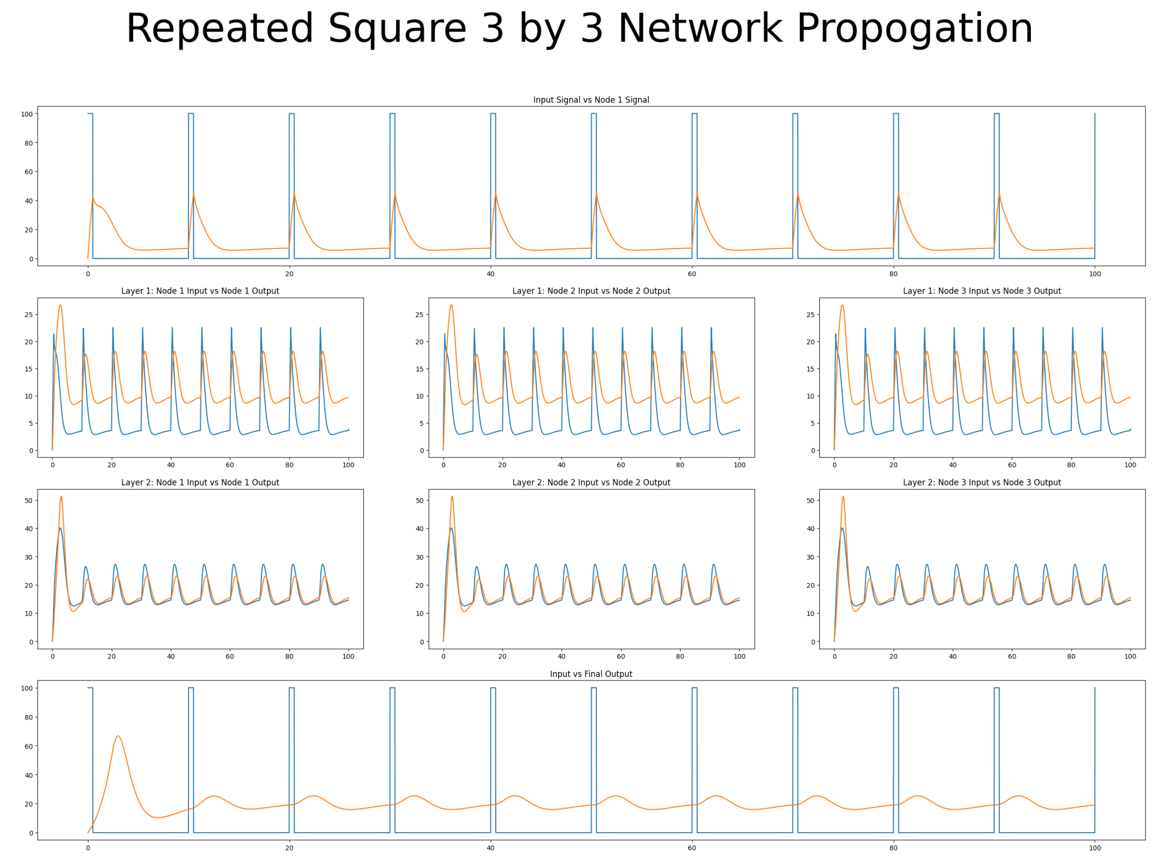


Figure 31: Uniform weight w/ Repeated Square Stimulus

3.4.2 Random Weights

In this section we explore the Parakh-Varanasi model with random weights. Specifically, each weight parameter arbitrarily takes a value between 0 and 1.

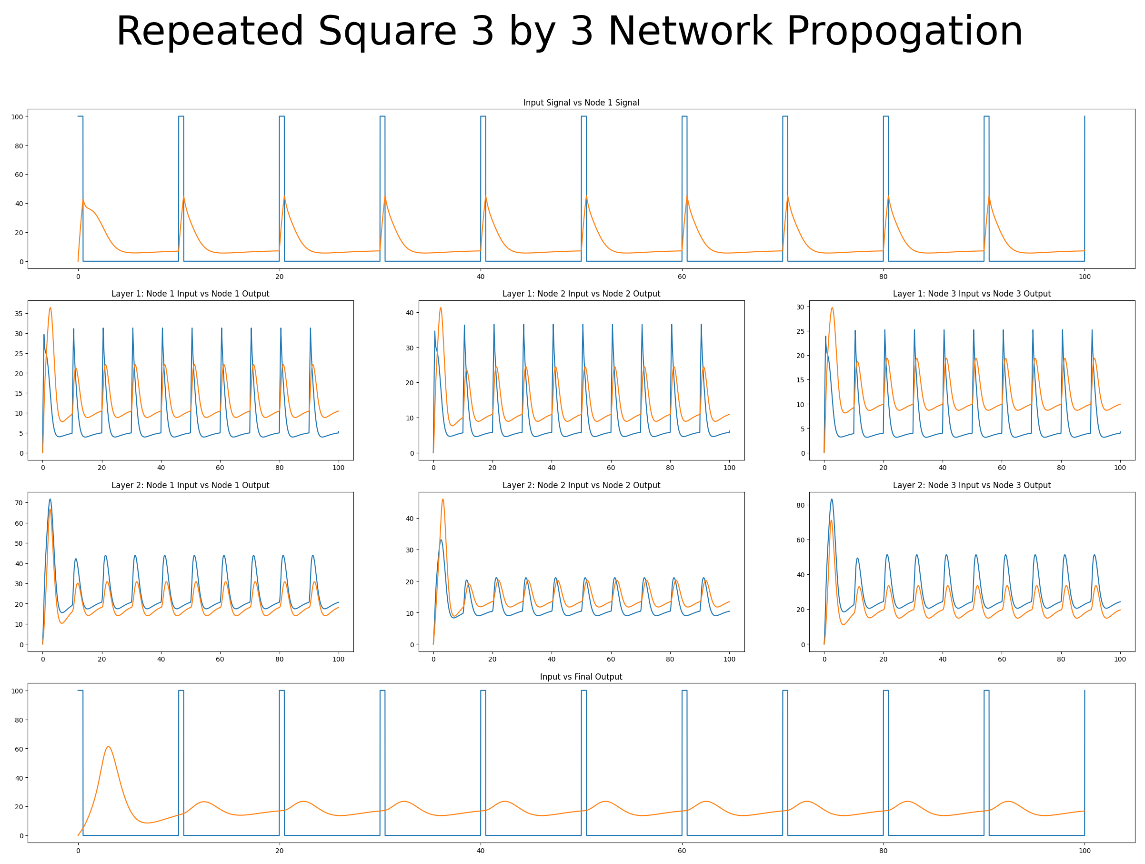


Figure 32: Random weight w/ Repeated Square Stimulus

Although it is slightly obscured by the scaling of the graphs, we notice that the weights are able to control for the magnitude of peaks produced in the network. We notice that the cyclic and output behaviors pointed out in earlier sections are preserved through network weight transformations. We posit that through sufficient manipulation of network weights; we may be able to excite complex behavior from the network.

3.5 Complex Behaviour

In this section of our exploratory analysis we look into the complex behavior of our Parakh-Varanasi system. In particular, we analyze the phase diagrams of n, m, and h, the probability of sub-unit activation and deactivation as a function of cell potential. Once again for consistency, we are looking at a 3 x 3 network architecture with uniform weights.

3.5.1 Gaussian Pulse

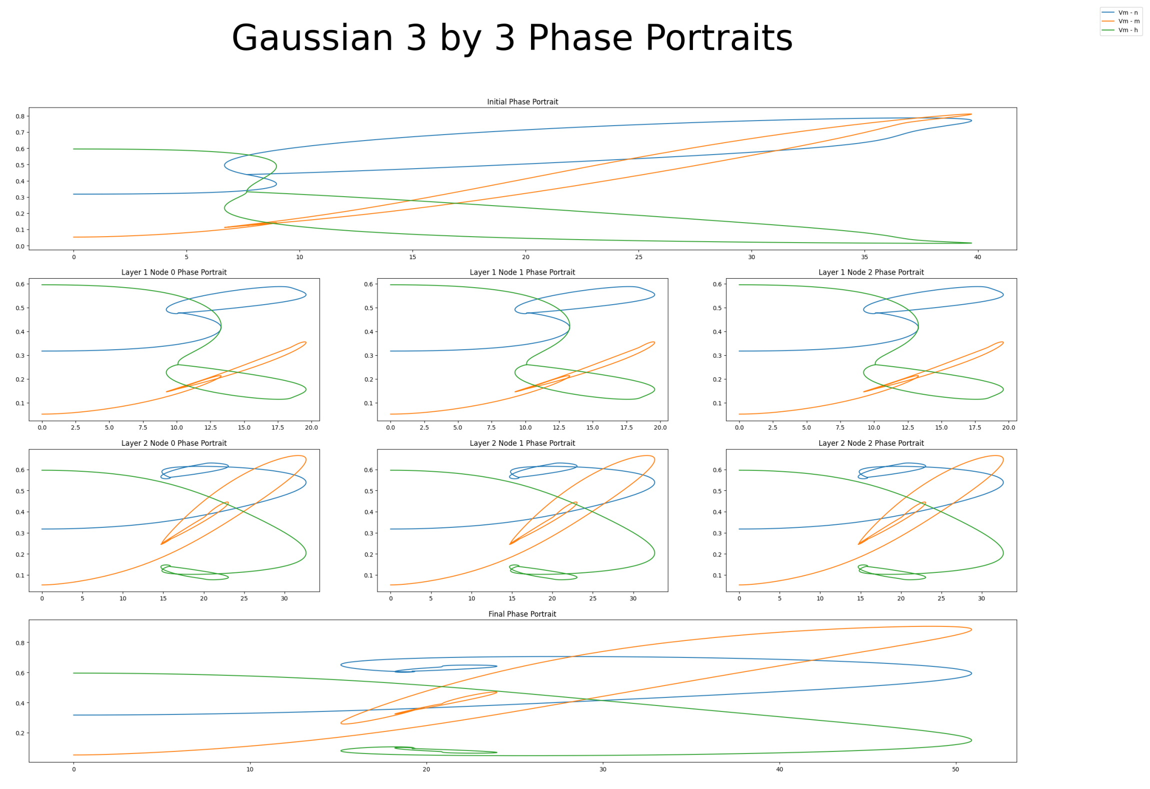


Figure 33: Phase Portraits of Gaussian Stimulus

As you can see in the phase portraits, there appears to be a limit cycle around which the three activation/deactivation parameters orbit. We notice that as we propagate our signal through the network, we retain the same macro scale non-linear behavior. For example, the limit cycles in the first neuron are preserved in our final neuron. However, despite the macro-scale structure being preserved, the limit cycle appears to become tighter. This phenomena is especially apparent between the 1st and 2nd layer in which we see the diameter of the orbit decrease.

3.5.2 Square Pulse

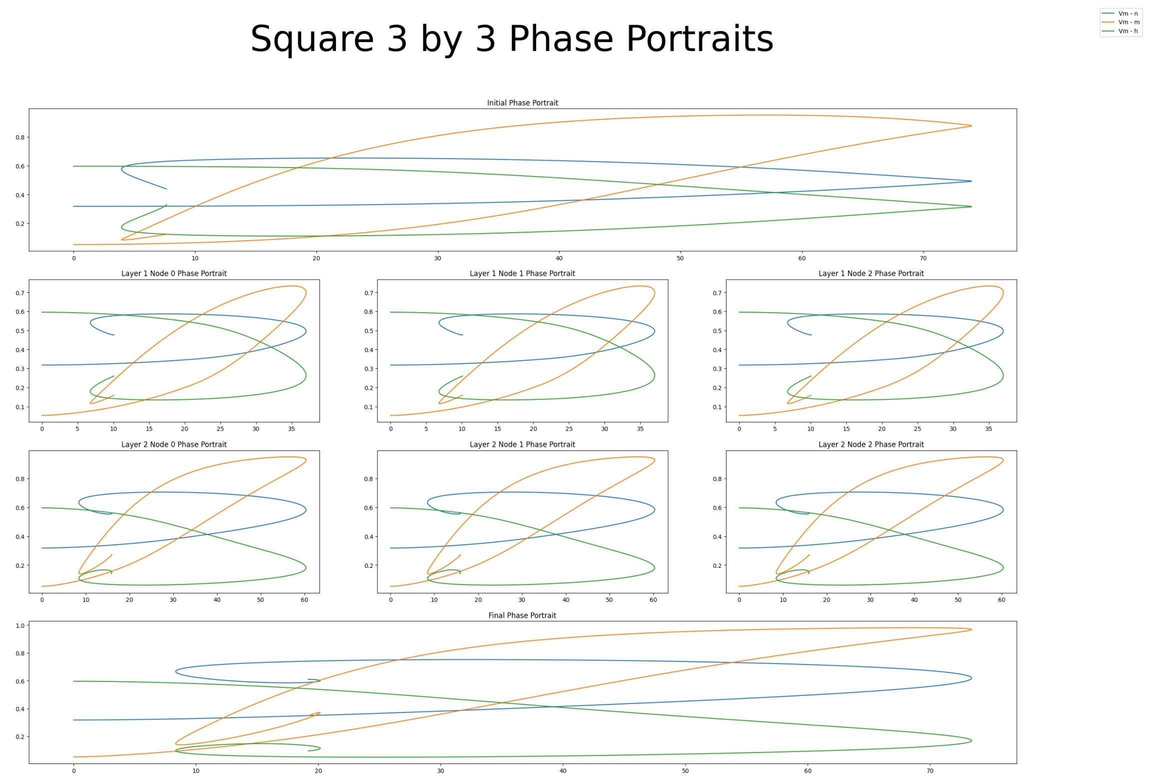


Figure 34: Phase Portraits of Square Stimulus

With the square pulse stimulus, we observe an attracting fixed point int0 which the activation/diffusion parameters settle in. We once again observe that the network is able to preserve complex behavior with the fixed points existing in each stage of the network. Like the Gaussian pulse we see that the network translates the fixed points as stimulus traverses through the system.

3.5.3 Square Double Pulse

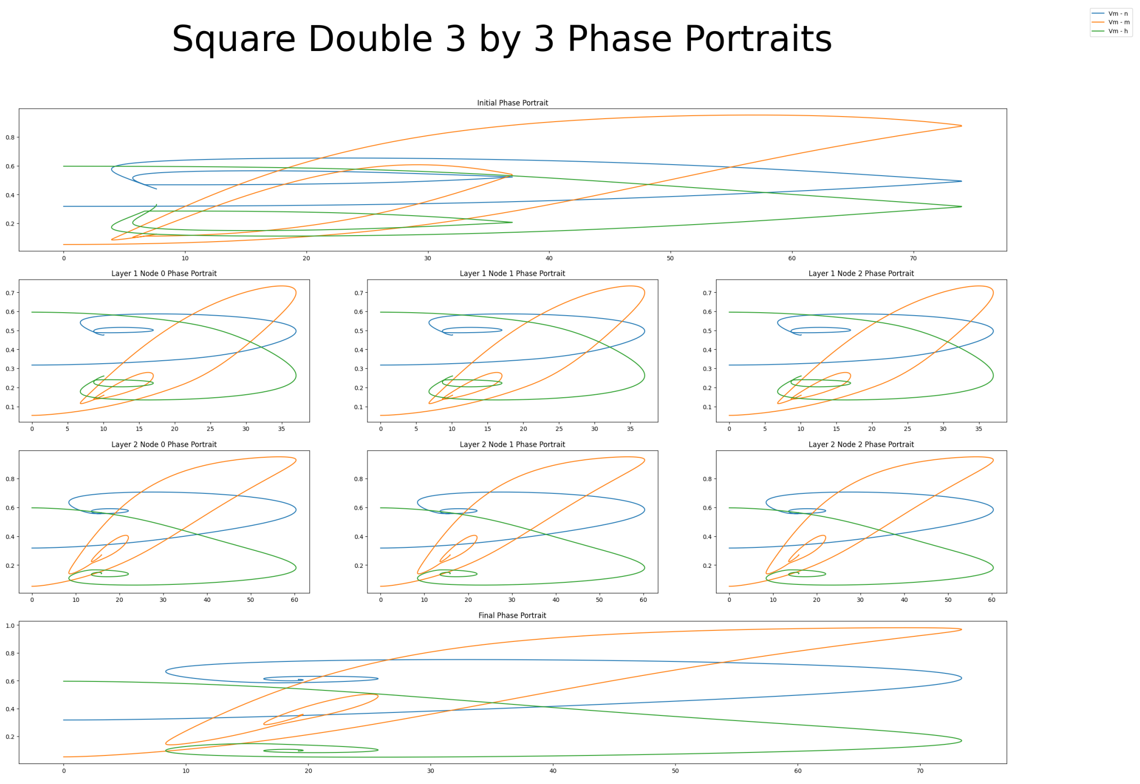


Figure 35: Phase Portraits of Square Double Stimulus

With the square double pulse input we notice psuedo-limit cycle behavior in each stage of the network. Unlike true limit cycle behavior, the response observed in this situation is partially a response to the secondary response. See see that the fixed point in the single pulse model evolves into an orbit here.

3.5.4 Repeated Square Pulses

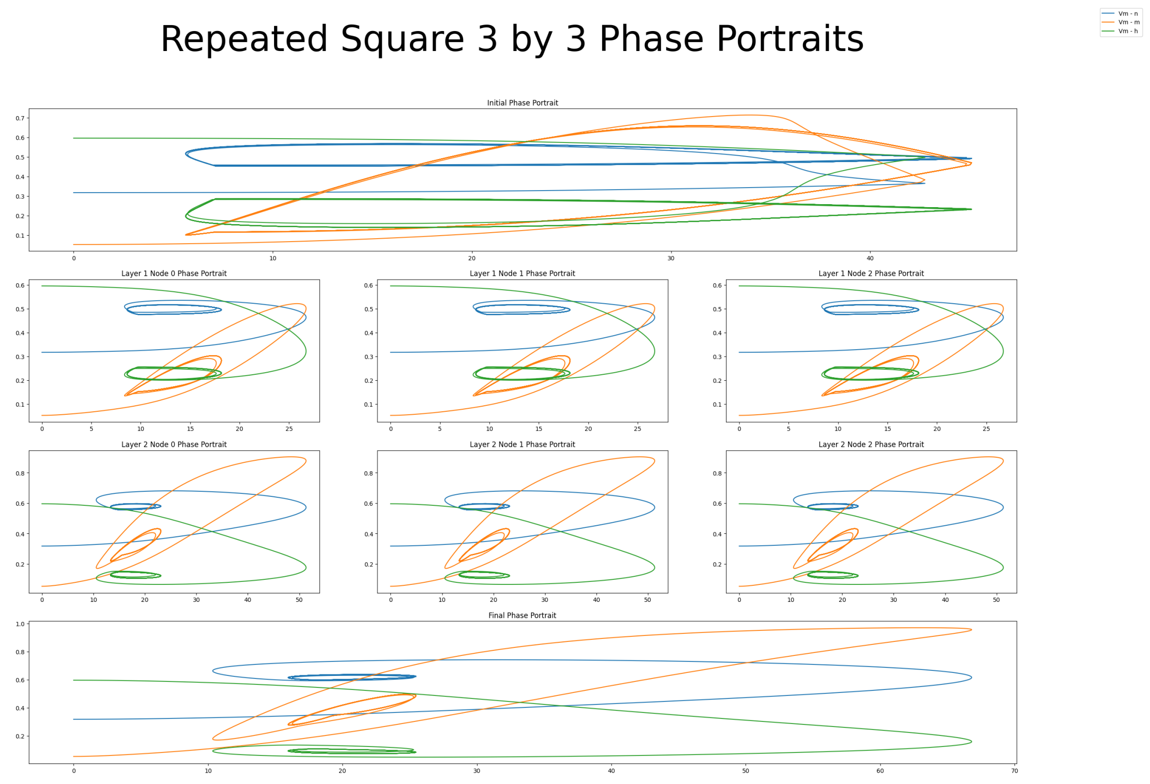


Figure 36: Phase Portraits of Repeated Square Stimulus

We continue to observe the same phenomena with the repeated square pulse as we see strongly defined cyclic behavior in each phase of the network. The tightening of the limit cycle is much stronger in this model with the final output limit cycles having a much smaller radius than the initial input phase diagram. As observed in the other phase portraits we see that the model preserves complex behavior.

4 Future Work: Neural Circuit Simulation

With our understanding of the dynamics Parakh-Varanasi model our next step is to optimize the parameters of our model to produce specified outputs. In the framework of memory and neural circuits the idea is to learn the neural network conditions that are able to reproduce a specified output given a series of inputs.

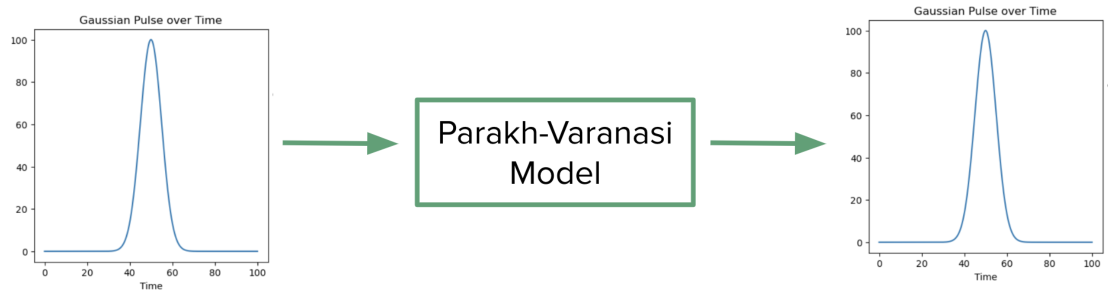
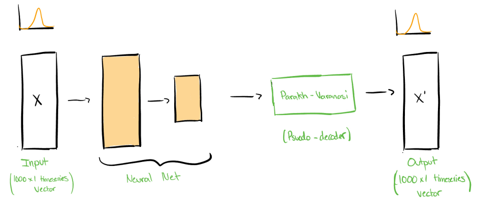


Figure 37: Example of Signal Reproduction

A full work through of this goal is outside the scope of this project, so we outlined a series of steps and avenues that we would study if we were to continue this project. We posit that the Parakh-Varanasi model would be able to reproduce a given signal via a psuedo-fourier decomposition of the desired signal into the space of outputs producible by our model. //

4.1 Model Design

At a high level the model would be trained and developed as follows:



mm

Figure 38: Training Model Architecture

Given an initial input signal, a neural net would re-parameterize the signal into the inputs of our Parakh-Varanasi model. At the simplest stage, this would correspond to the weights of the connections in our network. These paramteres would then be passed into the Parakh-Varanasi model along with the initial input signal to produce an output signal. This output signal would then be compared to our desired output form and a simple mean-sqaured error loss function would be applied. With this loss function the network would be able to learn the parameters of the Parakh-Varanasi model that would produce desired outputs.

4.2 Model Parameters

Once a basic minimum viable produce is completed, we outline the following areas as possible additions to the model.

- Neuron Parameters: Adding functionality to adjsut the parameters of each neuron allows us to prodcue new and complex behaviours with our model. For example, allowing certain cells to have lowered membrane capacitance or higher channel conductance will modify the dynamics of the signal processing. By doing so, we will provide more flexibility to our model.

- Internal Layers: Adding more layers to the model allows us to better replicate brain function and reproduce more complex output characteristics.
- Layer Interconnections: Adding inter-layer connections and feedback loops allows the model to better model true neural circuits and produce more intricate behaviours.

5 Appendices

5.1 Appendix A: Dependencies

There are several dependencies that are required to run the code in Appendix B. The following packages must be installed onto your local machine:

NumPy 1.22.3

SciPy 1.9.1

Python 3.9.12

Matplotlib 3.5.2

Once installed, you can then import the required modules and classes from these packages by running the following imports.

```
# Imports
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint
```

5.2 Appendix B: Hodgkin-Huxley Implementation

Here we provide the code implementation for the Hodgkin Huxley Model written in Python 3.9.12. This does not include any plotting methods used, or the parameter changes. These are provided in other appendices.

```
# Start and end time (in milliseconds)
t0 = 0
t1 = 100
t = np.linspace(t0, t1, 100000)

# Membrane capacitance (uF/cm^2)
C = 1.0

# Potassium potential (mV)
VK = -10

# Sodium potential (mV)
VNa = 110

# Leak potential (mV)
Vl = 10.0

# Potassium channel conductance (mS/cm^2)
gK = 4.0

# Sodium channel conductance (mS/cm^2)
gNa = 4.0

# Leak channel conductance (mS/cm^2)
gL = 1.0

# Rate functions as defined in model
def alpha_n(Vm):
    return (0.01 * (10.0 - Vm)) / (np.exp((10.0 - Vm) / 10.0) - 1.0)
```

```

def beta_n(Vm):
    return 0.125 * np.exp(-Vm / 80.0)

def alpha_m(Vm):
    return (0.1 * (25.0 - Vm)) / (np.exp((25.0 - Vm) / 10) - 1.0)

def beta_m(Vm):
    return 4.0 * np.exp(-Vm / 18.0)

def alpha_h(Vm):
    return 0.07 * np.exp(-Vm / 20.0)

def beta_h(Vm):
    return 1.0 / (np.exp((30 - Vm) / 10) + 1.0)

# Now define the n, m, and h's long term behvious as described in the
    model
def n_inf(Vm=0.0):
    return alpha_n(Vm) / (alpha_n(Vm) + beta_n(Vm))

def m_inf(Vm=0.0):
    return alpha_m(Vm) / (alpha_m(Vm) + beta_m(Vm))

def h_inf(Vm=0.0):
    return alpha_h(Vm) / (alpha_h(Vm) + beta_h(Vm))

# In the differential equations defined, external stimulus is not
    depicted
# we encode it as a current density in the input signal.
# Single square pulse implimentation
def I_square(t):
    if 0.0 < t < 1.0:
        return 100.0
    return 0.0
    # return 100.0 * np.exp(-np.power(t - 50.0, 2.0) / (2.0 * np.power
    (5.0, 2.0)))

def derivatives(y, t0, C = C, VK = VK, VNa = VNa, Vl = Vl, gK = gK, gNa =
    gNa, gL = gL, pulse = I_square):
    dy = np.zeros((4,))

    Vm = y[0]
    n = y[1]
    m = y[2]
    h = y[3]

    # dn/dt
    dy[1] = (alpha_n(Vm) * (1.0 - n)) - (beta_n(Vm) * n)

    # dm/dt
    dy[2] = (alpha_m(Vm) * (1.0 - m)) - (beta_m(Vm) * m)

    # dh/dt
    dy[3] = (alpha_h(Vm) * (1.0 - h)) - (beta_h(Vm) * h)

    # dVm/dt
    dy[0] = (pulse(t0) / C) - (((gK / C) * np.power(n, 4.0))
                                * (Vm - VK)) - (((gNa / C) * np.power(m, 3.0) * h
    )
                                * (Vm - VNa)) - ((gL / C) * (Vm - Vl))

    return dy

# Solve ODE system
state = np.array([0.0, n_inf(), m_inf(), h_inf()])
results = odeint(derivatives, state, t, args = (C, VK, VNa, Vl, gK, gNa,
    gL, I_square))
```

5.3 Appendix C: Pulse Definitions

Here we provide code that was used to define the different types of input pulses to the squid axon. These pulses must be passed into the derivatives function as the pulse parameter when using SciPy odeint.

5.3.1 Single Square Pulse

This is the code used to define a single square pulse as the external stimulus to the system.

```
# Single square pulse implementation
def I_square(t):
    if 0.0 < t < 1.0:
        return 100.0
    return 0.0
```

5.3.2 Double Square Pulse

This is the code used to define a double square pulse as the external stimulus to the system. This type of pulse is applied to the system when varying other parameters such as the membrane capacitance, various channel conductances, etc.

```
# Double square pulse implementation
def I_double_square(t):
    if 0.0 < t < 1.0:
        return 100.0
    elif 10.0 < t < 11.0:
        return 50.0
    return 0.0
```

5.3.3 Gaussian Pulse

This is the code used to define a Gaussian pulse centered at 50ms with a standard deviation of 5ms.

```
# Gaussian pulse implementation
def I_gaussian(t):
    return 100.0 * np.exp(-np.power(t -
    50.0, 2.0) / (2.0 * np.power(5.0, 2.0)))
```

5.3.4 Repeated Square Pulse

This is the code used to define a square pulse that repeats every 10ms for 100ms in duration.

```python
# Repeated square pulse implementation
def I_square_repeated(t):
    # apply a pulse every 10 seconds
    return 100.0 * (t % 10.0 < 0.5)
```

We provide a plot of each of the four current stimuli here:

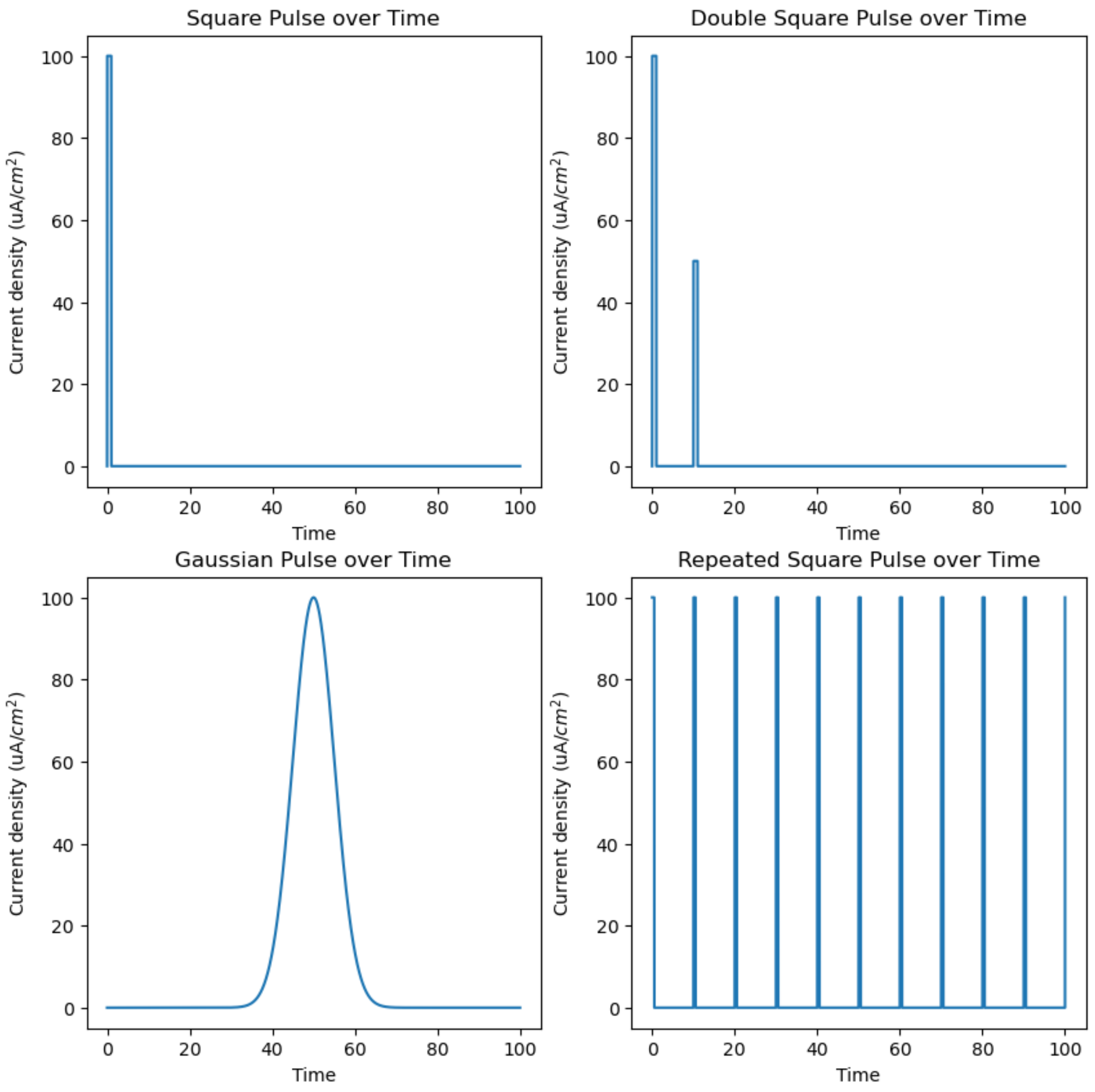


Figure 39: Plot of a square pulse, double square pulse, Gaussian pulse, and repeated square pulse. These pulses are used as the external stimulus to our system

5.4 Appendix D: Sweeping over Parameters

In our analysis, we do a sweep over various parameters defined in the Hodgkin-Huxley model and plot the membrane potential as we change them. We do this for all the parameters in the model. Here we provide an example python script for how to sweep over and plot the membrane capacitance, C_M.

```python
# Sweep over a range of membrane capacitances
t0 = 0
t1 = 100
t = np.linspace(t0, t1, 100000)

fig, ax = plt.subplots(3,3,figsize=(20, 20))
ax = ax.reshape(-1)

# Plotting the stimulus over time
ax[0].plot(t, Idv)
ax[0].set_ylabel(r'Current density (uA/$cm^2$)')
ax[0].set_xlabel('Time')
ax[0].set_title('Stimulus over Time')

# Define the different membrane capacitances we want to test. This can be
    changed by the user
# Membrane capacitance (uF/cm^2)
Cs = [-5, -1,0.0001, 1,2, 5,10, 100]

# Now iterate through those capacitances and apply each one in the solver
    .
for i, Curr in enumerate(Cs):

    # Solve ODE system
    state = np.array([0.0, n_inf(), m_inf(), h_inf()])
    results = odeint(derivatives, state, t, args = (Curr, VK, VNa, Vl, gK
    , gNa, gL, I_double_square))

    # Input stimulus
    Idv = [I_double_square(t) for t in t]

    # Plot neuron potential over time
    ax = ax.reshape(-1)
    ax[i+1].plot(t, results[:, 0])
    ax[i+1].set_xlabel('Time')
    ax[i+1].set_ylabel('Vm')
    ax[i+1].set_title('Neuron Potential over Time for C = {}'.format(C))
```

5.5 Appendix E: Plotting Example

In our analysis, we provide several plots of the system over time. Here we give some example python scripts on how to generate these plots. Note that these python scripts use the above Hodgkin-Huxley model code, and require that the model is compiled and run before these examples can be used.

Here is a plotting example with the repeated square input pulse. First we plot the input stimulus, the membrane potential, parameters versus the membrane potential, and channel currents over time.

```python
def I_square_repeated(t):
    # apply a pulse every 10 seconds
    return 100.0 * (t % 10.0 < 0.5)

t0 = 0
t1 = 100
t = np.linspace(t0, t1, 100000)

# Solve ODE system
state = np.array([0.0, n_inf(), m_inf(), h_inf()])
results = odeint(derivatives, state, t, args = (C, VK, VNa, Vl, gK, gNa,
    gL, I_square_repeated))

# Input stimulus
Idv = [I_square_repeated(t) for t in t]

# Plot neuron potential over time
fig, ax = plt.subplots(3,2,figsize=(20, 20))
ax = ax.reshape(-1)
ax[1].plot(t, results[:, 0])
ax[1].set_xlabel('Time')
ax[1].set_ylabel('Vm')
ax[1].set_title('Neuron Potential over Time')
# Plotting the stimulus over time
ax[0].plot(t, Idv)
ax[0].set_ylabel(r'Current density (uA/$cm^2$)')
ax[0].set_xlabel('Time')
ax[0].set_title('Stimulus over Time')

# Plot some trajectories
ax[2].plot(results[:, 0], results[:, 1], label='Vm - n')
ax[3].plot(results[:, 0], results[:, 2], label='Vm - m')
ax[4].plot(results[:, 0], results[:, 3], label='Vm - h')
ax[2].set_title('N vs Vm')
ax[2].set_xlabel('Vm')
ax[2].set_ylabel('n')
ax[2].legend()
ax[3].set_title('M vs Vm')
ax[3].set_xlabel('Vm')
ax[3].set_ylabel('m')
ax[3].legend()
ax[4].set_title('H vs Vm')
ax[4].set_xlabel('Vm')
ax[4].set_ylabel('h')
ax[4].legend()

# plot Ik and Ina over time
ax[5].plot(t, gK * np.power(results[:, 1], 4.0) * (results[:, 0] - VK),
    label='K')
ax[5].plot(t, gNa * np.power(results[:, 2], 3.0) * results[:, 3] * (
    results[:, 0] - VNa), label='Na')
# ax[5].plot(t, gL * (results[:, 0] - Vl), label='L') ## LEAK CURRENT
    PLOT
ax[5].set_title('K and Na currents over time')
ax[5].legend()
```

Next we plot the three parameters, m, n, and h over time using the following python script.

```
# Now plot n, m, and h over time
fig, ax = plt.subplots(4,1,figsize=(20, 20))

# Plotting the stimulus over time
ax[0].plot(t, Idv)
ax[0].set_ylabel(r'Current density (uA/$cm^2$)')
ax[0].set_xlabel('Time')
ax[0].set_title('Stimulus over Time')

ax[1].plot(t, results[:, 1], label='n')
ax[2].plot(t, results[:, 2], label='m')
ax[3].plot(t, results[:, 3], label='h')
ax[1].set_title('n over time')
ax[1].legend()
ax[2].set_title('m over time')
ax[2].legend()
ax[3].set_title('h over time')
ax[3].legend()

plt.show()
```

5.6 Appendix F: Parakh-Varanasi Implementation

```
def HH_net (input_signal, params,input_name, l1_size =3 , l2_size =3):


    #Read in Weights
     weights1 = np.array(params[:l1_size]).reshape(1, l1_size)
     weights2 = np.array(params[l1_size:l1_size + l1_size*l2_size]).
    reshape(l1_size, l2_size)
     weights3 = np.array(params[l1_size + l1_size*l2_size:]).reshape(
    l2_size ,1)

    #Creat subplot figures
     fig=plt.figure(figsize=(30,20))
     fig1=plt.figure(figsize=(30,20))

    #Figure Titles
     fig.suptitle(" {} {} by {} Network Propogation".format(input_name,
    l1_size, l2_size), size = 60)
     fig1.suptitle(" {} {} by {} Phase Portraits".format(input_name,
    l1_size, l2_size), size = 60)


     cols = max([l1_size, l2_size])
     gs=GridSpec(4,cols )
     ax1=fig.add_subplot(gs[0,:])
     axf=fig.add_subplot(gs[3,:])

    gs1=GridSpec(4,cols )
    pp_ax1=fig1.add_subplot(gs1[0,:])
    pp_axf=fig1.add_subplot(gs1[3,:])



    #Create Neural Net Layers
    layer1= []
    for i in range(l1_size):
        state = np.array([0.0, n_inf(), m_inf(), h_inf()])
        H = HH_neuron(state)
        layer1.append(H)

    layer2= []
    for i in range(l2_size):
        state = np.array([0.0, n_inf(), m_inf(), h_inf()])
        H = HH_neuron(state)
        layer2.append(H)

    # Pass Signal through first neuron
    t= np.linspace(0,100, t_count)
    input_state = np.array([0.0, n_inf(), m_inf(), h_inf()])
    results = odeint(derivatives_data(input_signal), state, t)

    V_initial = results[:, 0]

    #Plot Initial node plots
    ax1= fig.add_subplot(gs[0,:])
    ax1.plot(t, input_signal, label = "Input Signal")
    ax1.plot(t, V_initial, label = 'Network Signal')
    ax1.set_title("Input Signal vs Node 1 Signal")


    ax= fig1.add_subplot(gs1[0,:])
    ax.plot(V_initial, results[:, 1])
    ax.plot(V_initial, results[:, 2])
    ax.plot(V_initial, results[:, 3])
    ax.set_title("Initial Phase Portrait")


    #Pass input signal to 1st layer
    V1s = []
    V1_in = np.array(V_initial, ndmin= 2).T @ weights1


    for j in range(l1_size):

        val, val_n, val_m,val_h= Vin_to_Vout(V1_in[:,j], layer1[j])

        ax2 = fig.add_subplot(gs[1,j])
        ax2.set_title("Layer 1: Node {} Input vs Node {} Output".format(j
   +1,j+1))
        ax2.plot(t, V1_in[:,j])
        ax2.plot(t, val)

        ax2= fig1.add_subplot(gs1[1,j])
        ax2.plot(val, val_n)
        ax2.plot(val, val_m)
        ax2.plot(val, val_h)
        ax2.set_title("Layer 1 Node {} Phase Portrait".format(j))


        V1s.append(val)

    v1 = np.array(V1s).T
    V2_in = v1 @ weights2

    # Pass inputs to 2nd layer
    V2s = []
    for j in range(l2_size):

        val, val_n, val_m,val_h= Vin_to_Vout(V2_in[:,j], layer2[j])



        ax2 = fig.add_subplot(gs[2,j])
        ax2.set_title("Layer 2: Node {} Input vs Node {} Output".format(j
+1,j+1))
        ax2.plot(t, V2_in[:,j])
        ax2.plot(t, val)


        ax2= fig1.add_subplot(gs1[2,j])
        ax2.plot(val, val_n)
        ax2.plot(val, val_m)
        ax2.plot(val, val_h)
        ax2.set_title("Layer 2 Node {} Phase Portrait".format(j))


        V2s.append(val)

    v2_out = np.array(V2s).T

    #Pass inputs through final layer
    I_final_in = v2_out @ weights3

    input_state = np.array([0.0, n_inf(), m_inf(), h_inf()])
    results = odeint(derivatives_data(I_final_in), state, t)

    I_out= results[:, 0]

    #Produce Final Plots
    axf.plot(t, input_signal, label = "Input Signal")

    axf.plot(t, I_out.flatten(), label = 'Network Signal')
    axf.set_title("Input vs Final Output")

    fig.show()

    print(params)

    #produce final phase portrain plots
    pp_axf.plot(I_out, results[:, 1], label='Vm - n')
    pp_axf.plot(I_out, results[:, 2], label='Vm - m')
    pp_axf.plot(I_out, results[:, 3], label='Vm - h')
    pp_axf.set_title("Final Phase Portrait")
    fig1.legend()
    fig1.show()

    return(I_out)
```

5.7 Appendix G: Parakh-Varanasi Neural Net

```
def HH_net_loss(y_true, y_pred):
    print(y_true.shape)
    print(y_pred.shape)

    #batch_size = weights.shape[0]
    #mse = 0

    I_finals = HH_tf(y_true, tf.transpose(y_pred), l1_size= l1_size,
   l2_size=l2_size)
    print(I_finals.shape)
    #print(input_sig.shape)

    Y_final = tf.convert_to_tensor(y_true)
    print(Y_final.shape)

    I_final = tf.convert_to_tensor(I_finals)
    print(I_final.shape)

    mse = tf.keras.losses.MeanSquaredError(Y_final, I_final)

    return(mse)
```

6 References

Cordy, Benjamin. \The History of Electrophysiology." Grey Matters, Grey Matters, 25 Sept. 2021, https://greymattersjournal.org/the-history-of-electrophysiology/.

Fang, Xiaoyan, et al. \Memristive Hodgkin-Huxley Spiking Neuron Model for Reproducing Neuron Behaviors." Frontiers in Neuroscience, U.S. National Library of Medicine, 23 Sept. 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8496503/.

HODGKIN, A L, and A F HUXLEY. \A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve." The Journal of Physiology, U.S. National Library of Medicine, Aug. 1952, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/.

Hodgkin-Huxley Spiking Neuron Model in Python." Giuseppe Bonaccorso, 30 Sept. 2017, https://www.bonaccorso.eu/2017/08/19/hodgkin-huxley-spiking-neuron-model-python/.

Hodgkin{Huxley Model." Wikipedia, Wikimedia Foundation, 23 Nov. 2022, https://en.wikipedia.org/wiki/Hodgkin%E2%80%93Huxley_model.

The Ising Model." The Ising Model, https://stanford.edu/~jeffjar/statmech/intro4.html.

Ising Model." Wikipedia, Wikimedia Foundation, 5 Dec. 2022, https://en.wikipedia.org/wiki/Ising_model#Historical_significance.

Ising Model." Wikipedia, Wikimedia Foundation, 5 Dec. 2022, https://en.wikipedia.org/wiki/Ising_model.

The Physiological Society - Wiley Online Library. https://physoc.onlinelibrary.wiley.com/.

V Fig. 1 Typical Membrane Circuit Containing Active Na and K Channels ... https://pages.jh.edu/motn/coursenotes/nonlinear.pdf.