MODELING & SIMULATION OF BIOLOGICAL NEURAL NETWORKS USING KOCH MODEL

By

Arsalan Gohar

(2010-NUST-MS-CS&E-17)

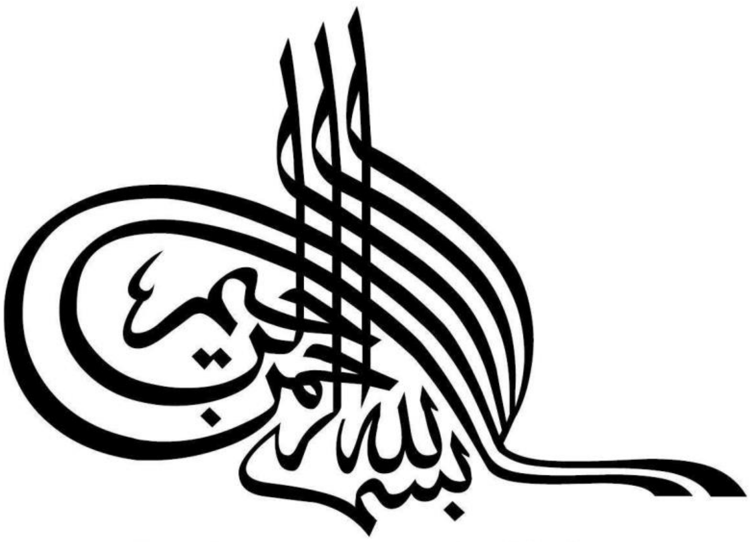
A thesis submitted in partial fulfillment of the requirements of the degree of Master of Science Computational Science and Engineering

Research Centre for Modeling and Simulation,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

2012



In the name of Allah,
the Most Beneficent,
the Most Merciful

DEDICATION

To My Family for their Prayers without which I would not have been able to carry out my studies

ACKNOWLEDGEMENTS

I would like to express my thanks and sincere gratitude to my advisor **Dr. Jamil Ahmed**, Assistant Professor, RCMS, NUST, for his continuous support, helpful advice and valuable guidance throughout this project work. His emphasis for excellence kept me well-directed and focused. He has made me able to follow the roadmap of thesis plan very smoothly through his well-defined execution and co-operation day in and day out. His cheerful and enthusiastic encouragement was a source of strength for me to complete this project.

I am immensely grateful to **Dr. Umer Khan Niazi,** Assistant Professor, (RCMS), NUST, for giving me valuable inputs. His vast experience of the field has assisted me in an unmatchable fashion right to the completion.

I am thankful to **Mr. Tariq Saeed,** Assistant Professor (RCMS) NUST for his unconditional help in programming of the code and guidance thought the thesis.

My sincere gratitude is to **Air Cdre Sikandar Hayat Mirza**, Director, (RCMS), NUST, whose emphasis for excellence kept me focused on to my project and helped me complete it on time. I thank him for providing the necessary resources and facilities to carry out this project in time.

My sincere appreciation goes to **Dr. Meraj Mustafa Hashami**, Assistant Professor, (RCMS) NUST for his support in mathematical problems.

I would also acknowledge the active support; I got from all other Departmental Seniors, MS students and staff for providing the necessary resources and facilities to carry out this project in time.

Finally, I am extremely gratified and indebted to my family members for their enormous support, persistent encouragement and earnest prayers throughout this project.

Table of Contents

List of Figures

List of Tables

Nomenclature

V_m	Membrane Voltage
E_{Na}	Potential Energy of Sodium (Na) ion
E_K	Potential Energy of Potassium (K) ion
ECa	Potential Energy of Calcium (Ca) ion
I_M	Current due to M ion (M is the name of the ion carrier)
I_{ext}	External Stimulus (current) given to the neuron
C_m	Capacitance of the Neuron Membrane
$\frac{Xm}{dt}$	$\frac{Xm}{dt}$ is the activation potential equation for the X current channel
$\frac{Xh}{dt}$	$\frac{Xh}{dt}$ is the inactivation potential equation for the X current channel
$X\infty$	$X\infty$ are different constants us in solving the differential equations for simplification proposes
g_X	g_X is the conductance of the X ion

ABSTRACT

Neurons are of essential importance in biology and its applications. Neurons are the simplest unit of data (information) processing in the nervous system of humans and other animals. Besides their importance for biology and medicine, networks of neurons (the human brain) are the most complex and advanced computational devices known, and the study of neurons individually and working in concert is seen as a step toward understanding consciousness and cognition.

A.L. Hodgkin and A.F. Huxley in 1950's developed a system of nonlinear ordinary differential equations to explain the behavior of a neuron found of a giant squid. These nonlinear equations have since been used to model the behavior of a host of neurons and other excitable cells like heart muscles. Hodgkin-Huxley category models take a set of parameters as input and produce data relating the electrical behavior of the neuron as a function of time.

The cornerstone of modern neurobiology is the analysis by Hodgkin and Huxley of the initiation and propagation of the action potential in the squid giant axon. Their description accounted for two ionic currents: the fast sodium current I_{Na} and a delayed potassium current, I_K. However, while the Hodgkin-Huxley formula has been singularly important to biophysics, their equations do not describe a number of important phenomena such as adaptation to long-lasting stimuli or the dependency of some conductance on various ionic concentrations

The Koch model is the extension of the famous Hodgkin-Huxley model which is based on the fast sodium and delayed potassium currents, while the Koch model incorporates numerous ionic membrane currents and also takes into account the calcium dynamics of a neuron.

Spike Timing Dependent Plasticity (STDP) is a temporally asymmetric form of Hebbian learning encouraged by constricted temporal correlations between the spikes of pre- and postsynaptic neurons. As with additional forms of synaptic plasticity, it is broadly believed that it inspires learning and information storage in the brain, as well as the progress and improvement of neuronal circuits during brain development. In STDP, frequent presynaptic spike arrival a few milliseconds before postsynaptic action potentials points in many synapse types to long-term potentiation (LTP) of the synapses, whereas recurring spike arrival after postsynaptic spikes points to long-term depression (LTD) of the same synapse.

We for the first time have combined KOCH neuron model and Spike Timing Dependent Plasticity (STDP). In the model we have also incorporated delays due to the length and diameter of a neuron. This study helps in understanding the working of neural networks and learning behaviors. The approach is not only adaptable, but it is also scalable to very large network (billions of neurons). Different neural diseases affect the conductance of nerves such as peripheral neuropathy and mononeuritis multiplex.

CHAPTER 1: INTRODUCTION

Neuroscience is of fundamental importance in biology, as it plays a major role in the spread of information in living organisms. Neurons are the essential unit of information processing in the nervous system of humans and animals alike. In addition to their importance for biology, networks of neurons (like the human brain) are the most advanced computational devices known, and its study is seen as a stride toward better understanding consciousness and cognition.

Neuron functions by preserving a voltage difference across its cell membrane. Quick variations in voltage difference can be introduced in response to external stimuli, including chemical or electrical signals. Some neurons also display unprompted activity, firing steady patterns of electrical impulses even in the nonexistence of external stimulus. Spikes spread down the length of the neuron at a speed of roughly 20 meters per second, permitting quick broadcast of information over macroscopic distances inside an organism.

In spite of having been studied by many scientists, the comprehensive processes by which even a single neuron functions are still the subject of strong research. In recent developments in computational resources and mathematical modeling methods have allowed neuroscientists to initiate studying neurons using computer models and simulation while continuing using ever more sophisticated biological, physical, and chemical tools.

The brain is remarkably adept at acquiring, coordinating, and disseminating information about the body and its environment. Such information must be processed within milliseconds, yet it also can be stored away as memories that endured for years. Neurons within the central and peripheral nervous systems perform these functions by generating sophisticated electrical and chemical signals.

1.1 Neuron

A neuron is a cell that is the basic building block of the nervous system. Neurons have many similarities to other cells in the human body, but there is one important difference between neurons and other cells. That difference is that neurons are dedicated to transmit information throughout the body.

Nerve produce electrical signals that convey information. Even though neurons are not fundamentally good conductors of electricity, they have evolved intricate mechanisms for producing these signals centered on flow of ions across their membranes. Ordinarily, neurons produce a negative potential, called the “resting potential”, that can be measured by recording the voltage between the inside and the outside of nerve cells.

These exceedingly dedicated nerve cells are in charge for communicating information in chemical and electrical forms. There are also numerous varied types of neurons in control for different tasks in the human body.

Neurons that are in the sensory part convey information from the sensory receptor cells all over the body to the brain. Information from the brain to the muscles of the body are transferred by Motor neurons transfer. Interneurons are responsible for communicating information between different neurons in the body.

Human brains are linked to thousands of other neurons and contain tens of billions of neurons. There are trillions of specialized neuron connections in the brain known as synapses. Neurons have many dimensions, which regulate their roles.

1.1 Neuron Structure

A classic neuron has identical parts to that any cell would have, and a few dedicated structures that differentiate it from the rest. Soma is the main portion of the cell. It comprises of a nucleus that contains chromosomes.

Dendrites are large extensions of a neuron. They resemble branches or spikes spreading out from the cell body. Principally it is the exteriors of the dendrites that get chemical messages from other neurons. It is tough to differentiate axon from the dendrites, in others it is straightforwardly distinguished by its length. The basic structure of a neuron is shown in Figure 1-1, which highlights different parts of a neuron

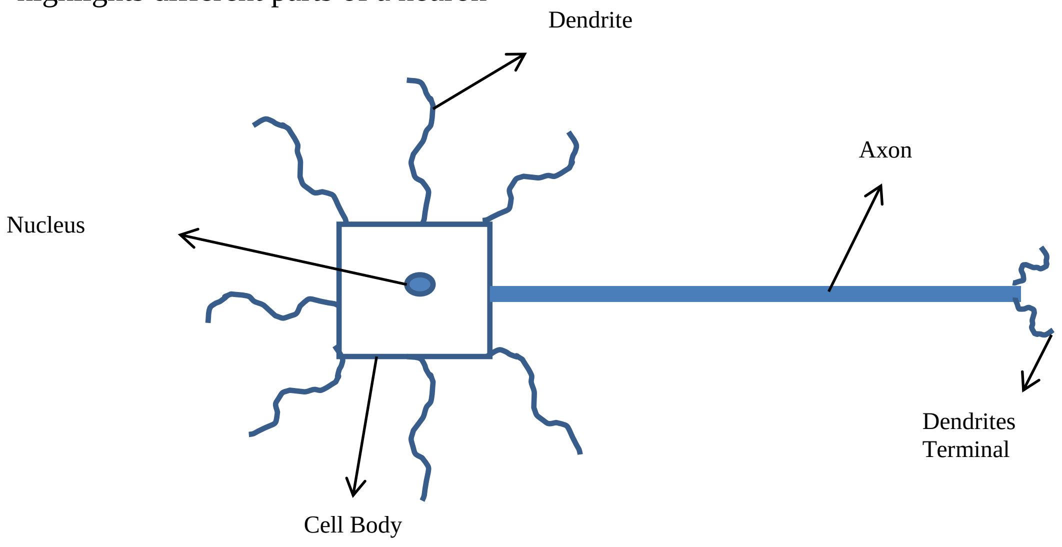


Figure 1-1: Basic Structure of a neuron, showing different parts of the structure of a neuron

Sandwiched between the axon ending and the dendrite of the next neuron is a very tiny gap called the synapse (or synaptic gap, or synaptic cleft), which we will discuss in a little bit. For every neuron, there are between 1000 and 10,000 synapses.

1.1.1 Synapse

The nervous system consists of a large number of neurons that are linked together to form functional conducting pathways. Where two neurons come into close proximity and functional inter-neuronal communication occurs, the site of such communication is referred as **Synapse**. Most neurons may make synaptic connections to a 1000 or more other neurons and may receive up to 10,000 connections from other neurons. (http://neuroscience.uth.tmc.edu/s1/introduction.html)

Synapses are the special surface contact sites where impulses are transmitted from a presynaptic cell (a neuron) to a postsynaptic cell (which may be a neuron or an effector cell, e.g. a muscle cell or gland cell). The synapse permits neurons to contact with each other or with the effector cells

Communications at synapse, under physiologic conditions, takes place in one direction only. Synapses occur in a number of forms. Synapses between the neurons are commonly classified into the following three types:

1) **Axosomatic synapse**, i.e., synapse between an axon and cell body (soma) of a neuron
2) **Axodentritic synapse,** i.e., synapse occurring between the axon of a neuron and the dendrite of another neuron

3) **Axoaxonic synapse,** i.e., synapse between an axon and another axon

Impulse transmission at synapses can occur electrically or chemically and, therefore synapses are classified as electrical synapses and chemical synapses.

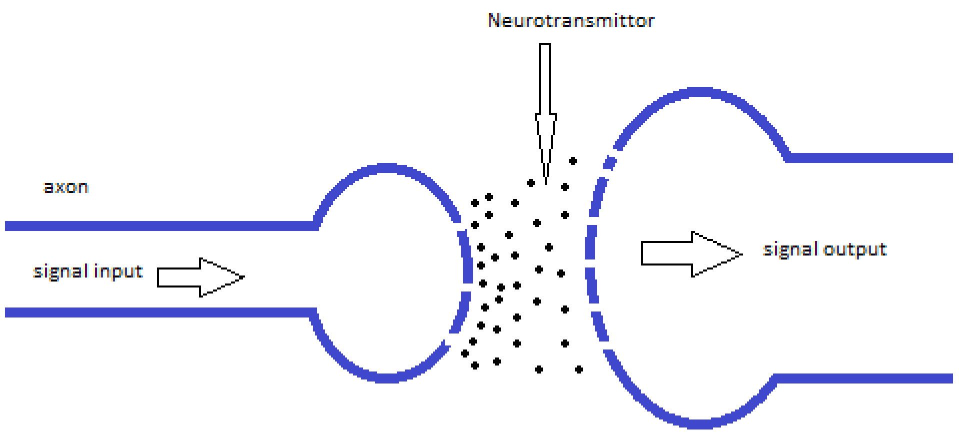


Figure 1-2: Synapse contains gap between two neurons, neurotransmitters transmitting the signal from one neuron to the other.

The **Electrical Synapses** contain gap junctions that permit free movements of ions from one neuron to another as shown in Figure 1-2. Movement of ions causes a flow of electrical current from one neuron to the other. Impulse transmission is much faster across the electrical synapses than across the chemical synapses. Electrical synapses have been found to be present in the retina, brainstem and cerebral cortex.

In **Chemical Synapses** conduction of impulses occurs by the release of chemical substances that are called neurotransmitters. Chemical synapses constitute the most common variety of synapses in the nervous system.

1.2 Action Potential

When the nerve cell is excited (stimulated) by electrical, mechanical, or chemical means, a rapid change in membrane permeability to Na^+ ions takes place, and Na^+ ions diffuse through the plasma membrane into the cell cytoplasm the tissue fluid. This results in the membrane becoming progressively depolarized, summarized in Figure 1-3. The sudden influx of Na^+ ions followed by the altered polarity produces the so-called **action potential**, which is approximately +40mV. This is very brief, lasting about 5 msec. The increased membrane permeability for Na^+ ions quickly ceases and the membrane permeability for K^+ ions increases. Therefore K^+ ions start to flow from the cell cytoplasm and return the localized area of the cell to the resting state. Figure 1-4 summarizes ion transfers at different regions of the action potential.

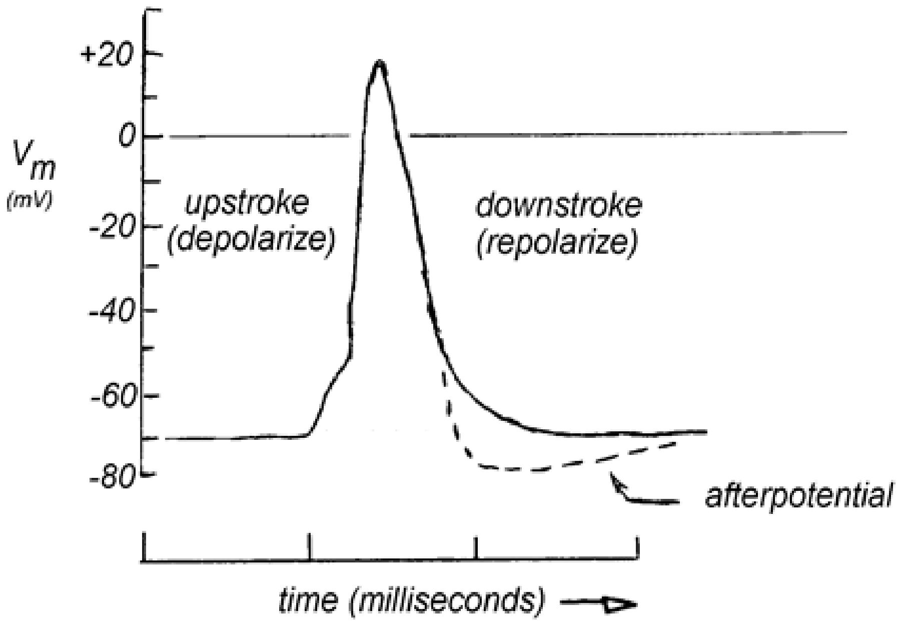


Figure 1-3: Action Potential with detail different regions

Once generated, the action potential spreads over the plasma membrane, away from the site of initiation, and is conducted along neuritis as the **nerve impulse.** This impulse is self-propagated and its size and frequency do not alter. Once the nerve impulse has spread over a given region of plasma membrane, another action potential cannot be elicited immediately. The duration of this non-excitable state is referred to as the **refractory period**, and it controls the maximum frequency that the action potentials can be conducted along the plasma membrane.

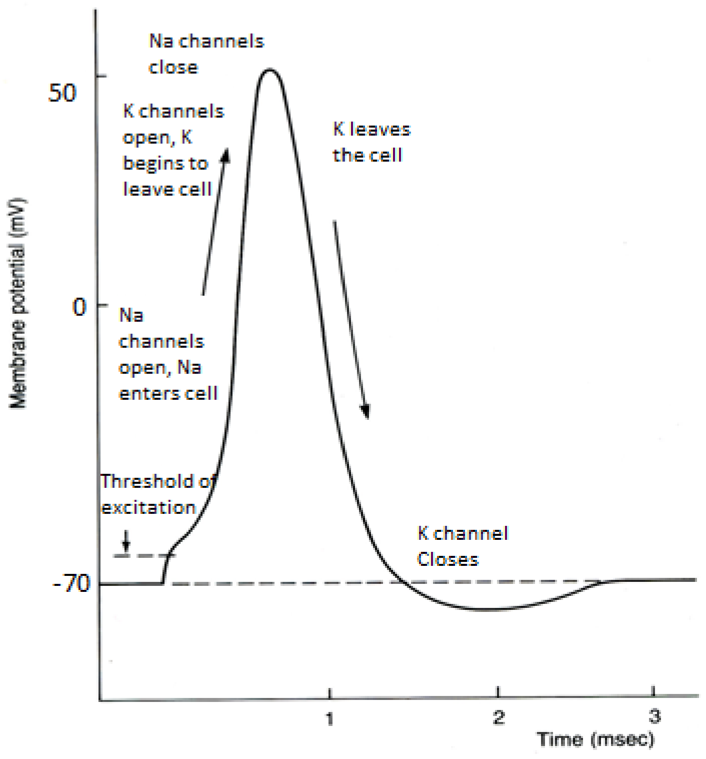


Figure 1-4: The working of ion channels during an action potential

Because electrical signals are the basis of information transfer in the nervous systems, it is essential to understand how these signals arise. Remarkably all of the neuronal electric signals are produced by similar mechanism that reply upon the movement of ions across the neuronal membrane.

Nerve cells produce electrical signals to send information over extensive distances and also transmit the information to other cells through synaptic connections. These signals eventually rest on variations in the resting electrical potential across the membrane. A resting potential arises as nerve cell membranes are permeable to one or more ion species subject to an electrochemical gradient.

1.3 Hodgkin Huxley

After 1950, when A.L. Hodgkin and A.F. Huxley established a system of equations unfolding the electrical activity of the squid giant axon [7], [8], neuroscientists have been equipped with a leading theoretical framework for studying neuronal function and behavior. The equations are not only largely valid to many classes of neurons, but the internal dynamics of the Hodgkin-Huxley model in many ways emulate the physiology of the neuron. The difficulty of working with living neurons, computational neuroscience has become an increasingly important tool for the study of neuronal physiology and behavior.

Through series of experiments on the axon of the squid, Hodgkin and Huxley reached a basic insight that the neuronal cell membrane has independent permeability mechanisms for different types of ions. And along with this this the results helped us understand the membrane's conductance for each type of ion which is a function of time and the trans-membrane voltage. Using nonlinear ordinary differential equations (given data specifying the internal state of the neuron and its initial conditions [7]), Hodgkin and Huxley with help of quantitative model reproduced the experimental data (as obtained from experiments above [7]), and to simulate the electrical activity of the neuron, it can be solved numerically. This model was the first complete description of the excitability of a single cell.

The ODE equations of Hodgkin and Huxley are the basis for almost all ionic current-based neuronal models for example sodium, potassium, calcium and after extending these equations, they have been used successfully in the study of neurons playing various physiological roles across a comprehensive spectrum of species.

The neuron, though active transport mechanisms that expend energy to transport ions across the cell membrane, There is maintenance of a voltage difference, which is also called the membrane potential, across the cell membrane separating the cytoplasm inside the cell from the extracellular fluid. An enzyme that is found in the cell membrane, the sodium-potassium pump, is one of the active transport mechanism in our body. With each pumping cycle, it causes the movement of three Na+ ions out of the cell and two K+ ions into the cell. Under the action of the sodium-potassium pump, it results in net outflow of positive charge setting up a steady state membrane voltage difference, which is called the resting potential. This causes the charge inside of the cell to be electronegative compared to the cell's surroundings. Ionic concentration imbalances across the cell membrane can also occurs due to active transport of ions across the cell membrane. For example, in the intracellular fluid, the concentration of Na+ ions lean towards lower side and the concentration of K+ ions lean towards higher than in the extracellular fluid. This is due to the continuous action of the sodium-potassium pump. The main driving force that decides whether an action potential is allowed or not when a stimulus arrives is the imbalance between the membrane resting potential and the ionic concentration imbalances. This is what allows the neuron to respond rapidly to stimuli causing the membrane to become permeable to the flow of ions resulting in an action potential.

There are specific types of ions in the cell membrane which consists many different ionic current pathways (ion channels) which allows the flow of these ions. In the response to voltage difference through the cell membrane these ionic channels open and close, neutralizing the voltage difference an then coming back to the resting state. The reason several reason for the ions flow through these channels. One being the electrostatic pressure which is developed because of the membrane potential, and other being the pressure that results from intracellular/extracellular concentration imbalances which is a caused by active transport. There is a state for each ion in this membrane where the two pressures are well-balanced where no net trans flow occurs. There are several different types of voltage-gated ion channels in a classic neuron and either impulsively or in response to external stimuli, the interaction between these channels allows an individual neuron to reveal a range of behaviors. Through the opening and closing of voltage-gated ion channels causes the conductance of the cell membrane to ions of the similar type and other types. This is because ions carry charge, and their flow across the cell membrane effects the membrane potential. This enables a sophisticated system of feedback loops that triggers the neuron's electrical activity.

As explained in the Hodgkin-Huxley model, this is a general explanation of the processes underlying an action potential. There is a substantial inconsistency in the ionic current pathways present in biological neurons. Currently developed ionic current-based models tend to be more complex than the primary Hodgkin-Huxley model, but the form of the original equations and the processes which they define generally follow the model developed by Hodgkin and Huxley relatively close.

1.4 Memory

The ability to store information on basis of experience they have and to retrieve much of it at one's own will is one of the most fascinating brain's complex functions. This gives our brain many of the cognitive functions for example learning, memory. The process by which new information is attained by the nervous system and is evident through changes in behavior is called learning. The encoding, storage and retrieval of learned information is referred as learning. Similarly intriguing is the normal aptitude to forget information. In daily life, there are various diseases and disorders due to memory loss which has helped us in understanding of pathological forgetfulness and amnesia which is one of the foremost challenges of modern neuroscience, a challenge that has only initiated to be encountered.

The critical first step to create a new memory is called encoding. It allows the observed item of concern to be altered into a construct that can be stored within the brain, and then when needed recalled later from either short-term or long-term memory depending other factors.

With perception through the senses, gives rise to a biological event which is encoding. The process of setting down a memory begins with attention, in which neurons fires more frequently to create a memorable event, making the experience more powerful and growing the probability that the event is set as a memory.

Encoding occurs on different levels however the exact mechanism is not fully understood. In first step, the formation of short-term memory from the ultra-short term sensory memory occurs, followed by the transformation to a long-term memory by a process of memory consolidation.

Basically human memory is associative, meaning that a new piece of information is remembered well better if it can be related with previously acquired knowledge that is already firmly anchored in memory. The more personally meaningful the association is, the more operative the encoding and consolidation.

After the primary acquisition the process of stabilizing memory process is called consolidation. It may possibly be thought as a part of the process of encoding or storage or it may be deliberated as a memory process in its own way. It is typically considered to be comprised of two specific processes; one being the synaptic consolidation (which happens within the first few hours after learning) and the other being the system consolidation.

The process of consolidation exploits a phenomenon called long-term potentiation, neurologically, which permits a synapse to rise in strength as rising numbers of signals are transmitted between the two neurons. The process by which synchronous firing of neurons makes those neurons more persuaded to fire together in the future is called Potentiation. On the other hand, long-term potentiation occurs when the same group of neurons fire together so often that they become permanently sensitized to each other [7]. As new experiences increases, the brain produces more and more connections and pathways, and it may "re-wire" itself by re-routing connections and re-arranging its association.

The storage is the more or less submissive process of recalling information in the brain, whether in the sensory memory, the short-term memory or the more permanent long-term memory [25]. Each of these different stages of human memory functions as a sort of filter that aids to guard us from the overflow of information that provoke us on a daily basis and also avoiding surplus of information and aiding to keep us healthy. The more the information is frequent, the chances it has to be retained in long-term memory.

Recall or recovery of memory refers to the successive re-accessing of events or information from the past, which have been formerly encoded and stored in the brain. In common dialect, it is known as remembering. During recall, the brain "replays" a pattern of neural activity that was initially produced in reply to a certain event, echoing the brain's perception of the actual event. These repetitions are not reasonably identical to the former otherwise we would not identify the difference between the genuine experience and the memory - but are mixed with consciousness of the present situation.

1.5 Neural Signaling

Neurons employ several different signals to encode and transfer information [19]. The resting membranes potential in neurons usually generates a negative potential that can be calculated by recording the voltage between the inside and outside of the nerve cells. Transiently, the action potential eliminates the negative resting potential and creates the trans member potential positive. The action potentials are transmitted along the length of axons and are the central signal that transmits information from one place to another in the nervous system. Still other types of electrical signals are created by the activation of synaptic contacts between neurons or by the actions of the activation of synaptic contacts between neurons or by the actions of external forms of energy on sensory neurons. All these electrical signals ascend from ion fluxes taken about by nerve cell membranes being selectively permeable to different ions, and from the non-uniform distribution of these ions through the membrane.

The electrical signals produced by neurons are caused by responses to stimuli, which then change the resting membrane potential. Receptor potentials are due to the activation of sensory neurons by external stimuli, such as light, sound, or heat.

Another type of electrical signal is associated with communication between neurons at synaptic contacts. Activation of these synapses generate synaptic potentials, which allow transmission of information from one neuron to another [2].

A fundamental problem for neurons is that their axons, which can be quite long are not good conductors. Although neurons and wires are both capable of passively conducting electricity, the electrical properties of neurons compare poorly to an ordinary wire. To compensate for this deficiency, neurons have evolved a booster system that allows them to conduct electrical signals over great distances despite their intrinsically poor electrical characteristics [22]. The electrical signals produced by this booster system are called action potentials.

1.6 Conduction Velocity

The process of nerve conduction has been extensively studied in the past, both experimentally and theoretically. Mathematical theories of nerve conduction based on the modern cable concept were developed a long time ago by Hursh [26], and eventually by Hodgkin and Huxley [7] [8].

Conduction velocity [10] is the speed at which the action potential moves from one neuron to the other. Conduction velocity plays a very important role in the learning functions. Because it is the conduction velocity which defines the delay in the transmission and reception of the impulse along with the couple of other variables such as the neurotransmitters, temperature and ionic concentrations [11]. We have varied the different dimensions of a neuron to see their effect on the conduction velocity of a neuron.

The electrical properties of the axon (as used in Hodgkin & Huxley and KOCH Model) are used to find the conduction velocity. In the resting state, the potential inside the axon at position x along the axon at time t, V(x,t) satisfies

$$Cm\frac{\partial V}{\partial t} + \frac{1}{rm}(V - Er) = \frac{1}{ri}\frac{\partial V^2}{\partial x^2}$$

Where Cm is the membrane capacitance, rm the membrane resistance, Er the emf of the membrane and ri is the longitudinal resistance of the axon in the resting state. The equation states that the membrane current, consisting of the capacitive and ohmic components, is directly related to the second derivative of the potential, $\frac{\partial v^2}{\partial x^2}$, by way of conduction of the axon interior, 1/ri.

1.7 Literature Review

Hodgkin Huxley model was the first neuron model that accurately described the initiation and propagation of an action potential. It was for their work that they were jointly awarded the NOBEL Prize in 1963. The Hodgkin-Huxley Model is a set of non-linear ordinary differential equations that approximate the electrical characteristics of excitable cells such as neurons.

Previously a lot of work has been done in this area but every one used Hodgkin Huxley model for simulating an action potential. Lyle N. Long has worked extensively in this area and has a couple of papers in this area; “A Review of Biologically Plausible Neuron Models for Spiking Neural Networks”, "Spike time dependent plasticity of neural circuits," and An Adaptive Spiking Neural Network with Hebbian Learning.

Koch model which is similar to Hodgkin and Huxley can be used for simulation and analysis of a single and network of neurons. The difference between the two is that Koch model has more number of parameters, giving it more flexibility and hence more realistic to the actual neuron. This is the reason we opted for Koch model [19]. Koch model is being used for detail analysis in most of the neural network [27] and [28]. We have implemented the model in Matlab (Appendix A) by model using predictor corrector method.

In brain, there are two types of axons; one being the myelinated while the other non-myelinated. Myelinated axons are used for long distances. Conduction speed of action potentials depends whether its myelinated axon or not. If myelinated, action potential travels at rapidly because the membrane capacitance is much smaller in those regions. Due to this phenomena, it’s particularly advantageous when signals must transmit information over long distances.

However, the range of traveling is limited to a few millimeters before the signal decays [29]. Nerve conduction abnormalities occur in many diseases [30], therefore its analysis and on-time detection is very necessary for cures of many diseases.

Spike time dependent plasticity is one of the essential parameter for analysis of memory. Many experiments have been carried out on this in 1983 and examined at millisecond level the effect of relative timing of pre and postsynaptic action potentials on plasticity. The experiments revealed that time frame relating pre- and post-synaptic activity and synaptic change. There are several reasons suggested for timing-dependent plasticity For example, STDP might provide a substrate for Hebbian learning during development [31] [32].

1.8 Contribution

This thesis elaborates a biological neural networks, based on different models and for the first time incorporated KOCH model for simulating a neural network. This would enable us in simulating neural networks that would have all the properties and complexities of a biological neural network.

KOCH model itself has so many different parameters that enable us to simulate different conditions that could affect the working of the neural network. Incorporating neural conduction model and spike time dependent model we have presented a unique model that has such properties. The neural model that we presented is very scalable and can be scaled to thousands of neurons and could also incorporate further neural models that could unable the neural network to have behavior as that of an actual biological neural network.

A paper was published titled “Nerve Conduction Using KOCH Model” in Symposium on Frontiers of Computational Sciences (ISFCS2012) Islamabad, Pakistan 2012.

CHAPTER 2: METHODOLOGY

The neural network is designed from multiple models combined, such as the KOCH neural model, conduction model and spiking time dependent plasticity. Each model is individually solved and incorporated in the network.

The differential equations of the KOCH model are solved using predictor-corrector scheme. Each of the differential equation is solve using the scheme. Range Kutte method could also be applied to solve the differential equations but the number of differential equations involved increases the complexity exponentially. So we used the predictor-corrector scheme.

The conduction model was also solved using the predictor-corrector method. The conduction model uses the values of voltage produced by KOCH model and updates them according to the conduction model.

The spiking time dependent plasticity uses the updated voltage value and computes the synaptic weights. So the synaptic weights incorporate the effects of the KOCH model and the effects of conduction model.

When the whole model is simulated it, the overall behavior of the action potential generated incorporated the effects of conduction velocity due to different sizes and also incorporate synaptic plasticity, which mimics the learning in neurons.

2.1 Hodgkin-Huxley Model

The Hodgkin-Huxley model is created on the idea that the electrical properties of a sector of nerve membrane can be molded by a circuit of the form shown in Fig. 1.8. The current flow across the membrane in the circuit, has two foremost modules, one linked with charging the membrane capacitance and one linked with the effort of specific types of ions across the membrane. The ionic current is further subdivided into three distinct components, a sodium current I_{Na}, a potassium current I_K , and a small leakage current I_L that is primarily carried by chloride ions.

The extracellular which separates medium from the cytoplasm of the cell, acts as a capacitor with capacitance C in HH. The current due to ion channels provides parallel ways through which charge can conductance (pass) through the cell membrane. HH model uses three currents based on ions in their description of the squid giant axon; potassium current I_K, sodium current I_{Na}, and a leakage current IL. The variable resistances of potassium and sodium currents that represent the voltage gated conductance related with the membrane ion channels. Overall current I is the sum of the capacitive current which represents the rate of accumulation of charge on opposite sides of the cell membrane and ionic currents. In electrical circuit theory for calculation of capacitive current, is Cdv/dt , where v is the membrane potential.

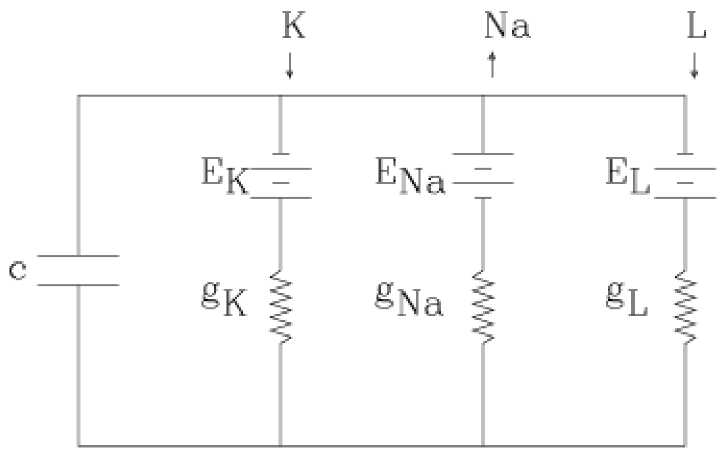


Figure 2-1: Hodgkin Huxley Electrical Circuit

The behavior of an electrical circuit of the type shown in Fig. 2-1 can be described by a differential equation of the form:

$$C_m \frac{dV_m}{dt} + I_{ion} = I_{ext}$$

where Cm is the membrane capacitance, Vm is the intracellular potential (membrane potential), I_{ion} is the net ionic current that flows across the membrane and I_{ext} is an externally stimulus applied.

The HH model consist of the following differential equations

$$\frac{dm}{dt} = \frac{m\infty - m}{\tau_m}$$

$$\frac{dh}{dt} = \frac{h\infty - h}{\tau_h}$$

$$\frac{dn}{dt} = \frac{n\infty - n}{\tau_n}$$

$$\frac{dV}{dt} = \frac{I_M - I_K - I_{Na} - I_L}{C_m}$$

$I_{Na} = \bar{g}_{Na} m^3 h(V - E_{Na})$

$I_K = \bar{g}_K n^4(V - E_{Na})$

$$\begin{aligned}
\tau_m &= \frac{1}{\alpha_m + \beta_m} \\
m_\infty &= \frac{\alpha_m}{\alpha_m + \beta_m} \\
\tau_h &= \frac{1}{\alpha_h + \beta_h} \\
h_\infty &= \frac{\alpha_h}{\alpha_h + \beta_h} \\
\tau_n &= \frac{1}{\alpha_n + \beta_n} \\
n_\infty &= \frac{\alpha_n}{\alpha_n + \beta_n}
\end{aligned}$$

With initial conditions

$m(0) = m\infty(V_0,0)$, $n(0) = n\infty(V_0,0)$

$h(0) = h\infty(V_0,0)$, $V(0) = V_0$

2.2 KOCH Neural Model

The Koch model [9] is the extension of the famous Hodgkin-Huxley model which is based on the fast sodium and delayed potassium currents, while the Koch model incorporates numerous ionic membrane currents and also takes into account the calcium dynamics of a neuron.

In recent years, numerous ionic membrane currents have been described, which differ in principal carrier, voltage and time dependence, and dependence on internal calcium. Our understanding of these currents and to a lesser extent, the role they play in impulse formation, has been accelerated by various technical innovations such as single-cell isolation and patch clamping.

KOCH model is a very comprehensive model that incorporates many different parameters that the Hodgkin Huxley model did not incorporate. Thus KOCH model represents a much closer to reality action potential and taking into consideration the different parameters. Figure 2-2 shows different ionic currents that KOCH model comprises of.

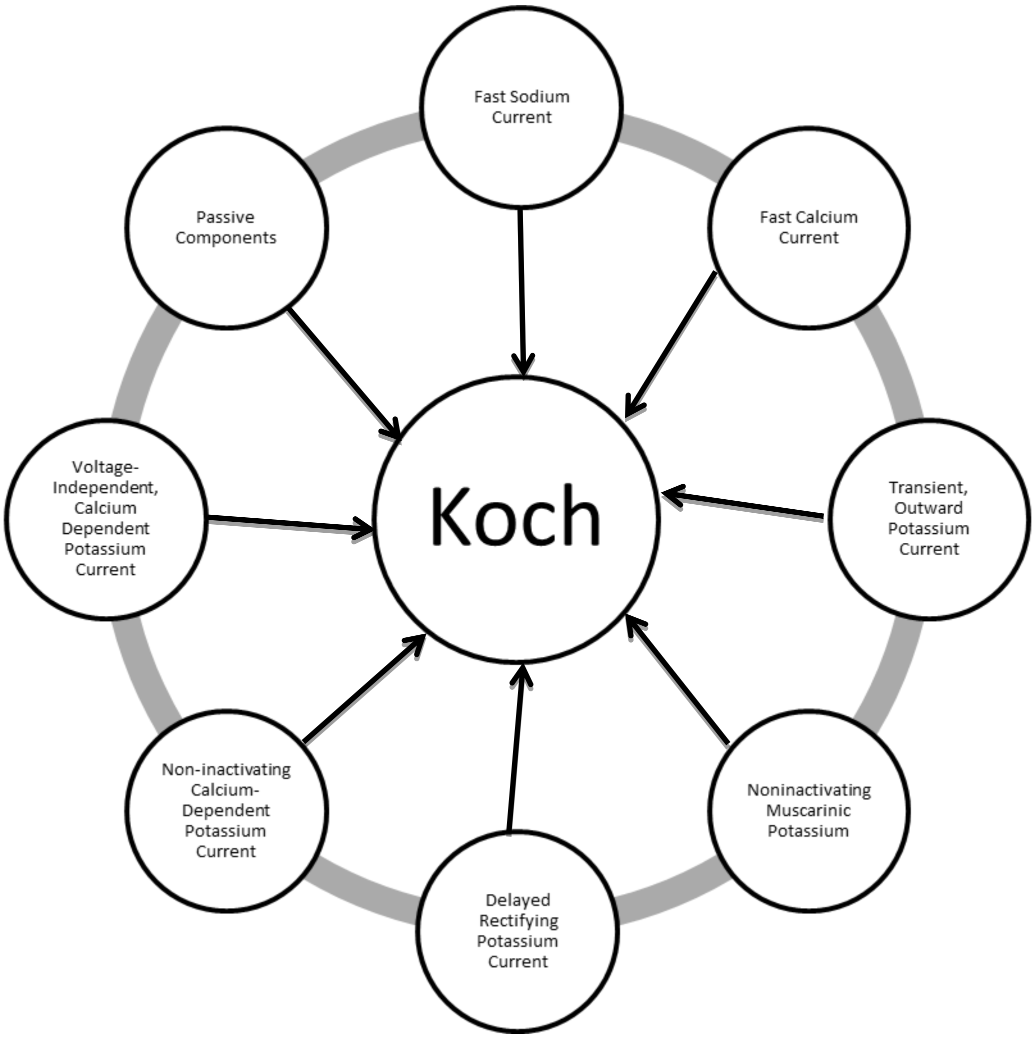

Fast Sodium Current
Passive Components
Fast Calcium Current
Koch
Voltage-Independent, Calcium Dependent Potassium Current
Transient, Outward Potassium Current
Non-inactivating Calcium-Dependent Potassium Current
Noninactivating Muscarinic Potassium
Delayed Rectifying Potassium Current

Figure 2-2: Parameters of Koch model

Principal Equation

$$C_N \frac{d}{dt}V + I_{Input} + I_{Na} + I_{Ca} + I_K + I_M + I_A + I_C + I_{APH} + I_{leak} = 0$$

All variables have the following units:

Voltage	mV
Current	nA
Time	ms
Concentration	millimols per liter (mM)
Conductance	µS
Resistance	MΩ
Capacitance	nF

Fast Sodium Current

$$I_{Na} = \bar{g}_{Na} m^2 h(V - E_{Na}) \qquad \frac{dm}{dt} = \frac{m_\infty - m}{\tau m} \qquad \tau_m = \frac{2}{\alpha m + \beta m} \qquad m_\infty = \frac{\alpha m}{\alpha m + \beta m}$$

$$\alpha_m = \frac{0.36(V+33)}{1 - e^{-(V+33)/3}} \qquad \beta_m = \frac{-0.4(V+42)}{1.0 - e^{(V+42)/20}}$$

$$\frac{dh}{dt} = \frac{h\infty - h}{\tau h} \qquad \tau_h = \frac{2}{\alpha h + \beta h} \qquad h_\infty = \frac{\alpha h}{\alpha h + \beta h} \qquad \alpha_h = \frac{-0.1(V+55)}{1 - e^{(V+55)/6}} \qquad \beta_h = \frac{4.5}{1.0 + e^{V/10}}$$

Fast Calcium Current

$$I_{Ca} = \bar{g}_{Ca} m h(V - E_{Ca}) \qquad \frac{dm}{dt} = \frac{m\infty - m}{\tau m} \qquad \tau_m = \frac{7.8}{e^{+(V+6)/16} + e^{-(V+6)/16}}$$

$$m_\infty = \frac{1}{1 + e^{(V-3)/8}} \qquad h = \frac{K}{K + \{Ca\}i}$$ where K = 0.01mM

Transient, Outward Potassium Current

$I_A = \bar{g}_A mh(V- E_K)$ $\qquad \frac{dm}{dt} = \frac{m\infty - m}{\tau m}$ $\qquad \tau_m = 1.38$

$m_\infty = \frac{1}{1+ e^{-(V+42)/13}}$ $\qquad \frac{dh}{dt} = \frac{h\infty - h}{\tau h}$ $\qquad h_\infty = \frac{\alpha h}{\alpha h + \beta h}$

$\tau_h = 50$ if $V < -80$mV; else 150

Noninactivating Muscarinic Potassium Current

$I_M = \bar{g}_M m(V- E_K)$ $\qquad \frac{dm}{dt} = \frac{m\infty - m}{\tau m}$ $\qquad \tau_m$

$= \frac{1000}{3.3(e^{+(V+35)/40} + e^{-(V+35)/20})}$ $\qquad m_\infty = \frac{1}{1+e^{-(V-35)/10}}$

Delayed, Rectifying Potassium Current

$I_K = \bar{g}_K m^2h(V- E_K)$ $\qquad \frac{dm}{dt} = \frac{m\infty - m}{\tau m}$ $\qquad \tau_m = \frac{1}{\propto m(V) + \beta m(V)}$

$m_\infty = \frac{\propto(V-20)}{\propto m(V-20) + \beta m(V-20)}$ $\qquad \alpha_m(V)= \frac{-0.0047(V+12)}{e^{-(V+12)/12} - 1}$

$\beta_m(V)= e^{-(V+147)/30}$ $\frac{dh}{dt} = \frac{h\infty - h}{\tau h}$ $\tau_h = 6,000$ if $V < -25$mV: else 50

$h_\infty = \frac{1}{1+ e^{+(V+25)/4}}$

Noninactivating Calcium-Dependent Potassium Current

$I_C = \bar{g}_C m(V- E_K)$ $\qquad \frac{dm}{dt} = \frac{m\infty - m}{\tau m}$ $\qquad \tau_m = \frac{1}{f(V,Ca) + b(V)}$

$m_\infty = \frac{f(V,Ca)}{f(V,Ca) + b(V)}$ $\qquad f(V,Ca) = 250[Ca^{+2}]_i\, e^{+V/24}$ $\qquad b(V) = 0.1e^{-V/24}$

Voltage-Independent Calcium-Dependent Potassium Current

$I_{AHP} = \bar{g}_{AHP}\, m^2(V- E_K)$ $\frac{dm}{dt} = \frac{m\infty - m}{\tau m}$ $\tau_m = \frac{1000}{f(Ca) + b}$

$m_\infty = \frac{f(Ca)}{f(Ca) + b}$ $f(Ca) = 1.25*10^8[Ca^{+2}]^2_n$ and $b=2.5$

Passive Components where m is the activation variable and

$I_{leak} = \bar{g}_{leak}\, m^2(V- E_{leak})$ h is the inactivation variable

2.3 Hebbian Learning

Learning is a learning rule which is the oldest and most famous of all learning rules.

" *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such as A's efficiency as one of the cells firing B, is increased*"

Hebb [9] proposed this change as a basis of associative learning. We may expand this as a two-part rule:

- "*If two neurons on either side of a synapse (connection) are activated simultaneously then the strength of that synapse is selectively increased.*"
- "*If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.*"

Such a synapse is called a Hebbian synapse. More precisely, we define a Hebbian synapse as a synapse that uses a time-dependent, highly local, and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the pre-synaptic and post-synaptic activities.

2.4 Spike-Timing Dependent Plasticity:

Spike Timing Dependent Plasticity (STDP) [10] is temporally asymmetric form of Hebbian learning induced by tight temporal correlations between the spikes of the pre and postsynaptic neurons. Synaptic plasticity is widely believed that it lies behind learning and information storage and also in refinement of neuronal circuits, during brain development.

Spike Timing Dependent Plasticity, repeated with the arrival of presynaptic spike a few milliseconds before postsynaptic action potentials leads in many synapse types to long-term potentiation (LPT) of the synapses that is when repeated spike arrives after postsynaptic spikes proceeds to long-term depression (LTD) of the same synapse. The change of the synapse plotted as a function of the relative timing of pre- and postsynaptic action potentials is called the Spike Timing Dependent Plasticity (STDP) function or learning window is shown in Figure 2-3. Figure 2-3 shows the effect of different timing of the post synaptic neuron t_0 and pre synaptic neuron t_1 on the synaptic weights,

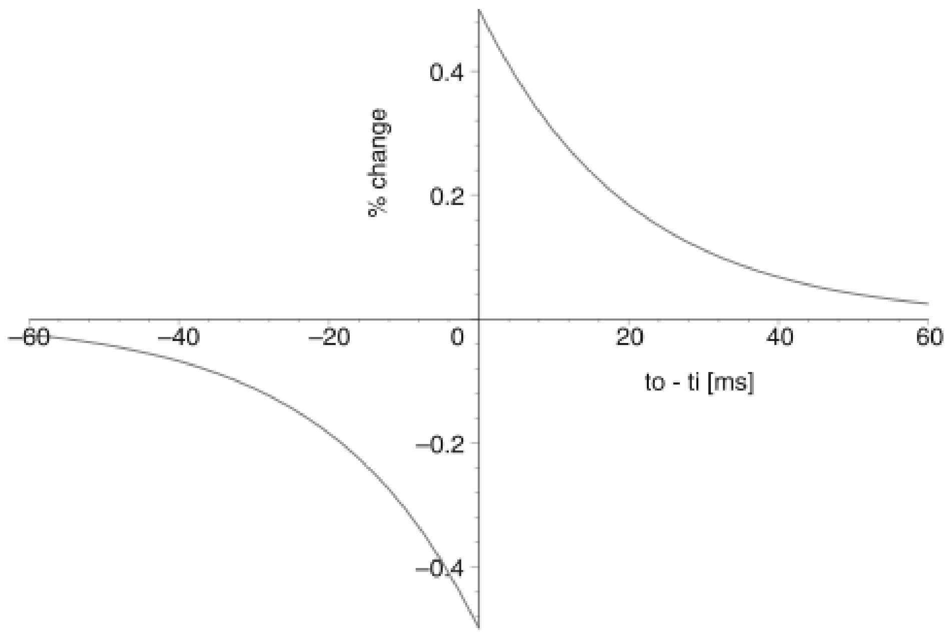
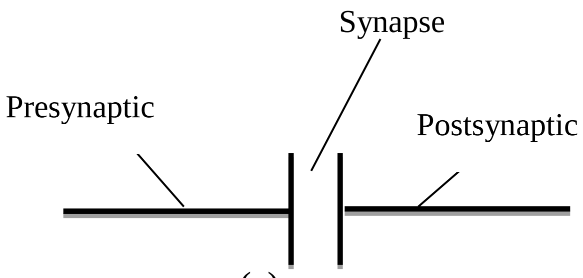


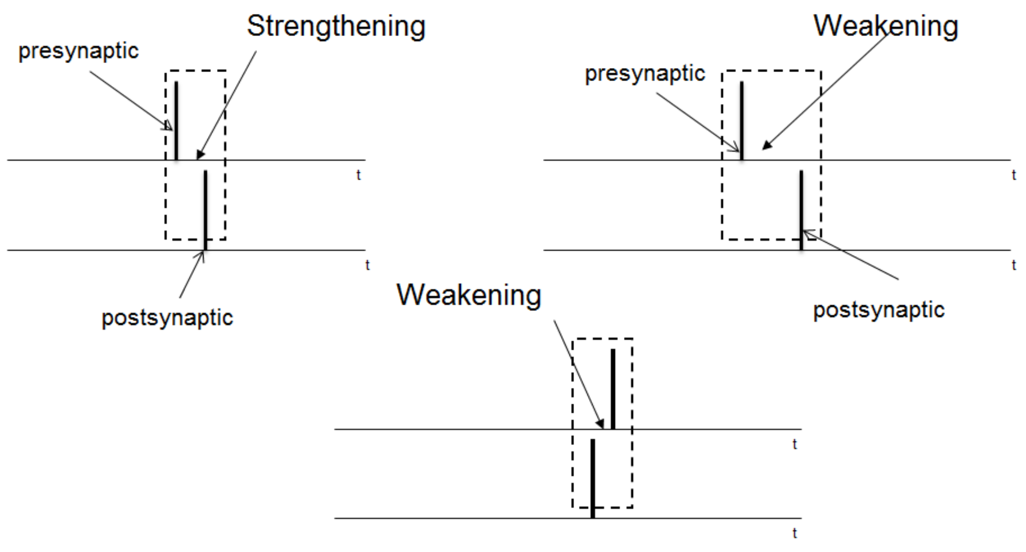
Figure 2-3: Learning Window for the spiking time dependent plasticity where t_0 is the time of firing of post synaptic neuron and t_1 is the time of firing of the presynaptic neuron

Following we explain the spiking time dependent plasticity with the help of the graphs. In the following graphs we represent the three possibilities of an impulse, first if the presynaptic impulse arrives before the post synaptic impulse but within the required time, for this case as the presynaptic impulse caused the firing of the impulse this causes an increase in the synaptic weights, the second show that the presynaptic impulse arrives very early and thus does not trigger the postsynaptic impulse thus weakening the synaptic weights. The third possibility is that the presynaptic impulse arrives after the post synaptic impulse and so the presynaptic impulse does not trigger the postsynaptic impulse hence the synaptic weights decrease.

We use the same neural network used previously, so that we have a common network that we could refer to. We used the same dimensions as length and diameter so that along with the constant neural network, we could have a good references. For using spiking time dependent plasticity model we used two different conditions: uniform frequency and different frequency. Figure 2-4 (a) shows basic Structure of neuron representing pre & post neuron and Figure 2-4 (b) summarizes the strengthening and weakening of plasticity with respective to time



(a)



(b)

Figure 2-4: Spiking Time Dependent Plasticity working principle (a) Basic Structure of neuron representing pre & post neuron (b) Strengthening and weakening of plasticity with respective to time

Algorithm for Spiking Time Dependent Plasticity

For each synapse, synapse plasticity is calculated he with respect to pre-neuron and post-neuron. If the time taken by pre-neuron is less than post neuron and within a specific time frame it causes strengthening of the synapse and else if pre-neuron is less again but not with in specific time frame, it causes weakening of the synaptic plasticity. There exists a special case in which the post-neuron signal reaches the before pre-neuron, then instead of increasing, it decrease the learning that is synaptic plasticity.

```
For each synapse
    if timeold(PostNeuron) < timenew(PreNeuron)< imenew(PostNeuron)
                weight = weight + dwPlus
        else
                weight = weight + dwMinus
        dwPlus = dwMinus  = $\alpha\, e^{dt}$

        dt = timenew(PreNeuron) - timenew(PostNeuron
```

CHAPTER 3: RESULTS

3.1 Action Potential

Action Potential being the most unique part of neurons, it is action potential that helps in the transfer of information. The way action potential is modeled is very important as it represents in a way the whole neural network, the more it is close to the actual action potential the better and much closer result to the actual environment.

We gave our system an input impulse of 4 nA at time 80 mSec shown in Figure 3-1 and the following is the result of the action potential produced. The action potential shows two action potential when we provided a single input, the reason is that the first impulse is initial action potential that the system required so that the system settles down. The resultant action potential very accurately behaviors like the actual potential as shown in Fig 3-2.

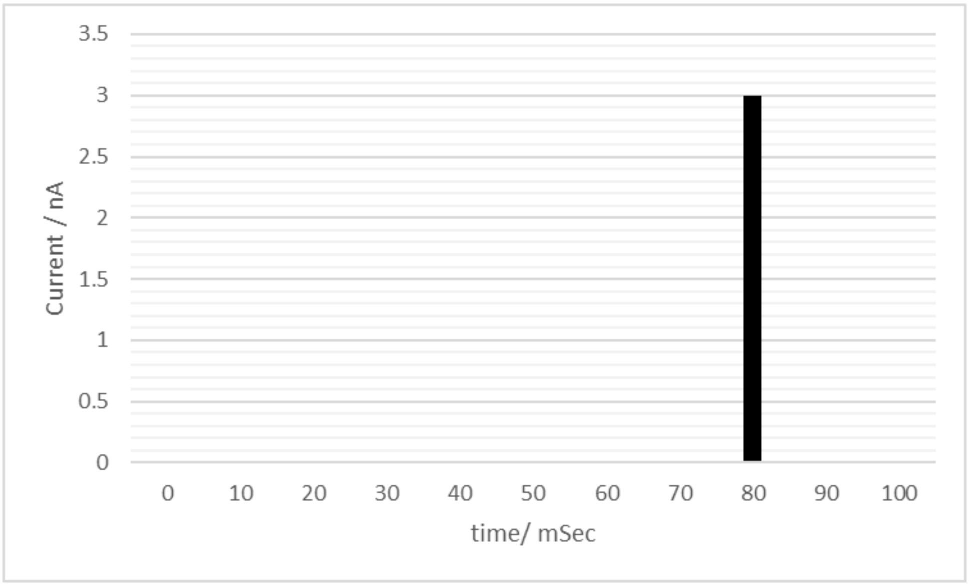


Figure 3-1: External stimulus of 4 nA

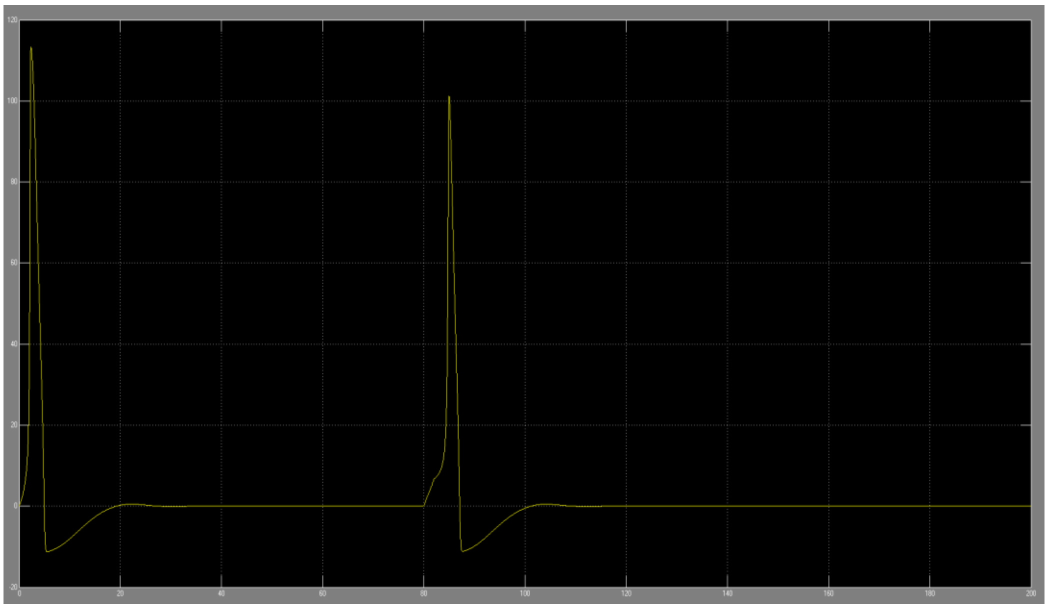


Figure 3-2: Action Potential with an input of 4 nA stimulus

Then we gave our system an input impulse of 8 nA at the 80 mSec shown in Fig 3-3 and the following is the result of the action potential produced shown in Fig 3-4. The reason that we provided such a high input current to see its behavior. Increasing the amplitude does not affect the action potential and similar is the result of our simulations. The action potential shows only very slightly increase in amplitude, even though we doubled the input stimulus and the resulting increase in amplitude is negligible.

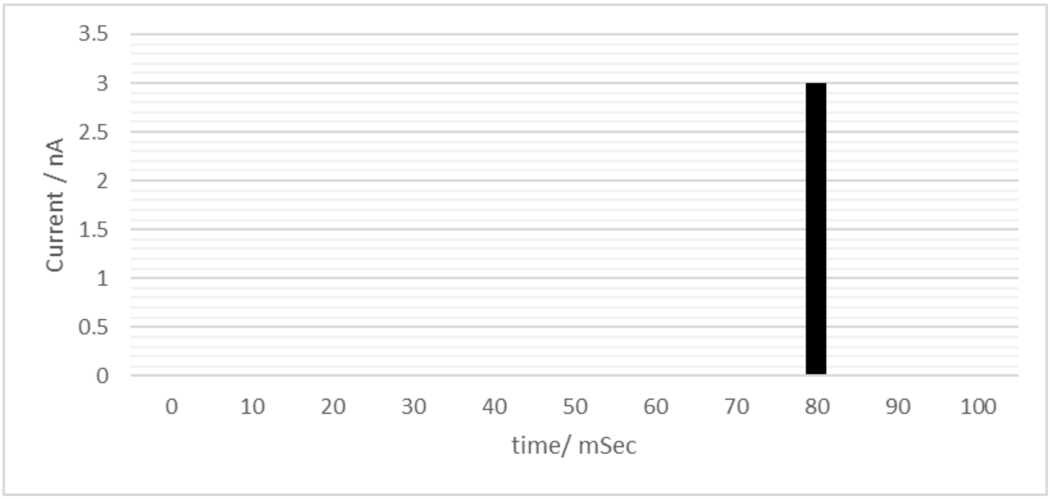


Figure 3-3: External Stimulus of 8 nA

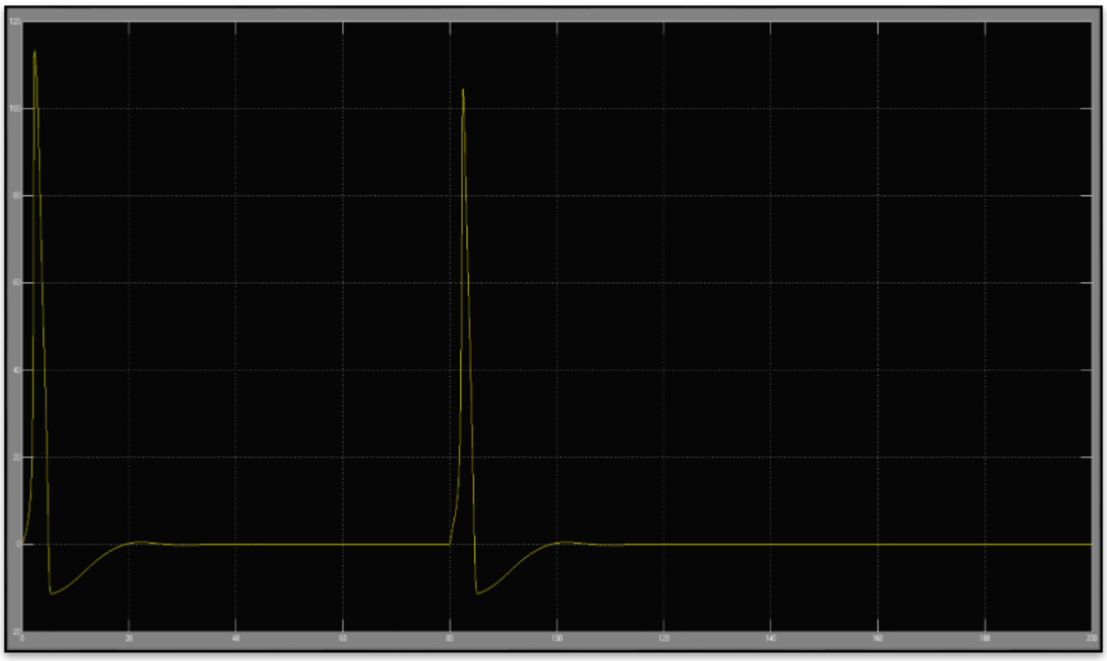


Figure 3-4: Action Potential with an input of 8 nA stimulus

Then we gave our system an input impulse of 2 nA at 80 mSec shown in Fig 3-5 and the following is the result of the action potential produced shown in Fig 3-6. The reason that we provided such a low input current as an input was to analyze its behavior. The current stimulus should be above the threshold, if it's not then the action potential should not fire. Our simulations represent the similar results, as the threshold is not crossed the action potential is not fired.

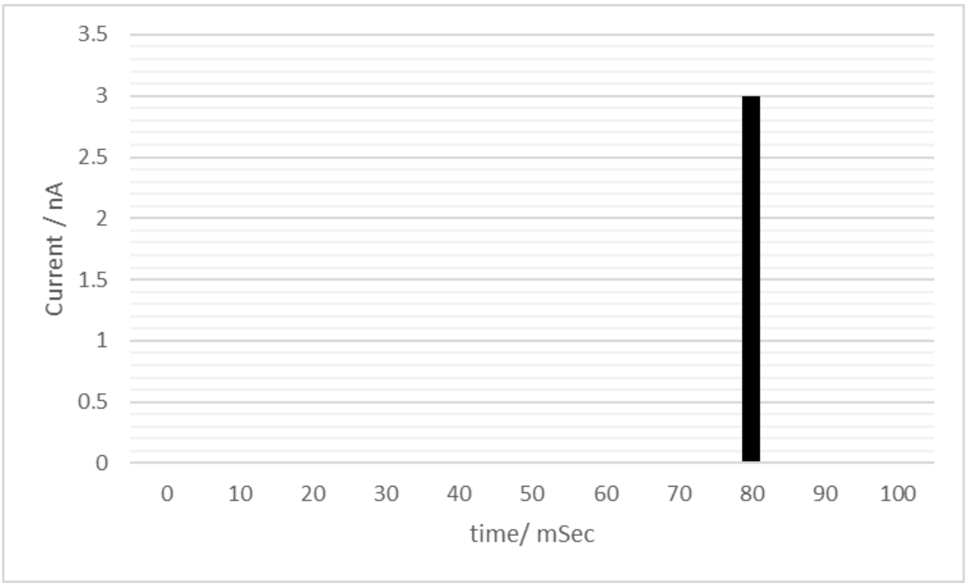


Figure 3-5: External Stimulus of 2 nA

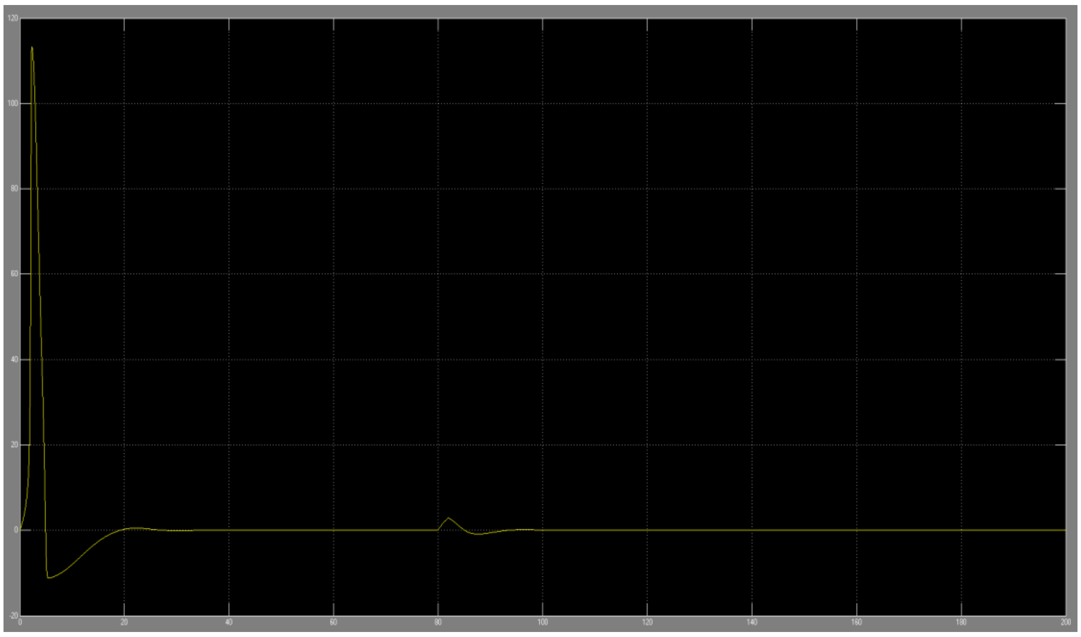


Figure 3-6: Action Potential with an input of 2 nA stimulus

Then we gave our system an input impulse of 3 nA at 80 mSec shown in Fig 3-7 and the following is the result of the action potential produced shown in Fig 3-8. The reason being we wanted to show how sensitive our model is to the input current. Our results prove that our model is sensitive enough that with even just 1 nA below the previous firing. After continuing this process we found out that the threshold for our system is 3.84nA

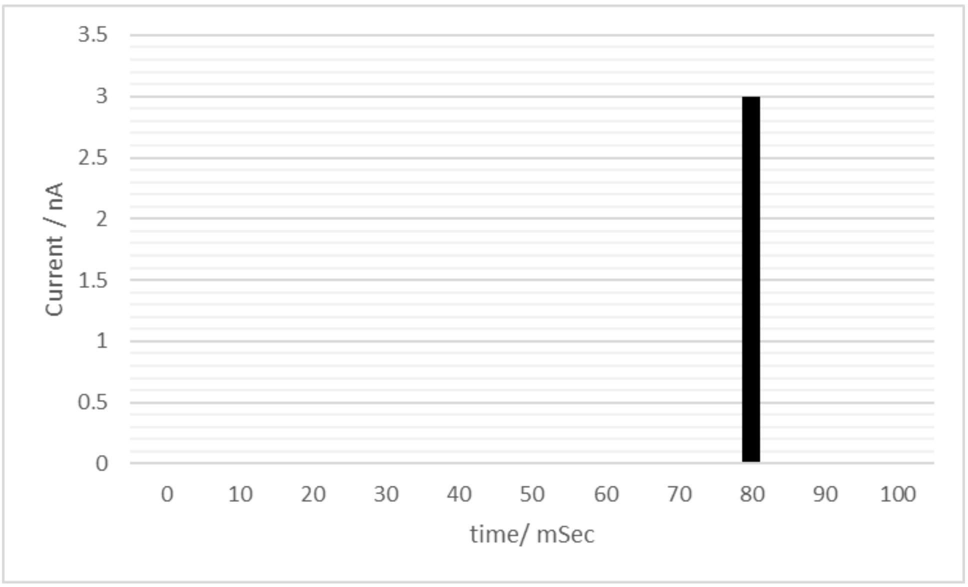


Figure 3-7: External stimulus of 3 nA

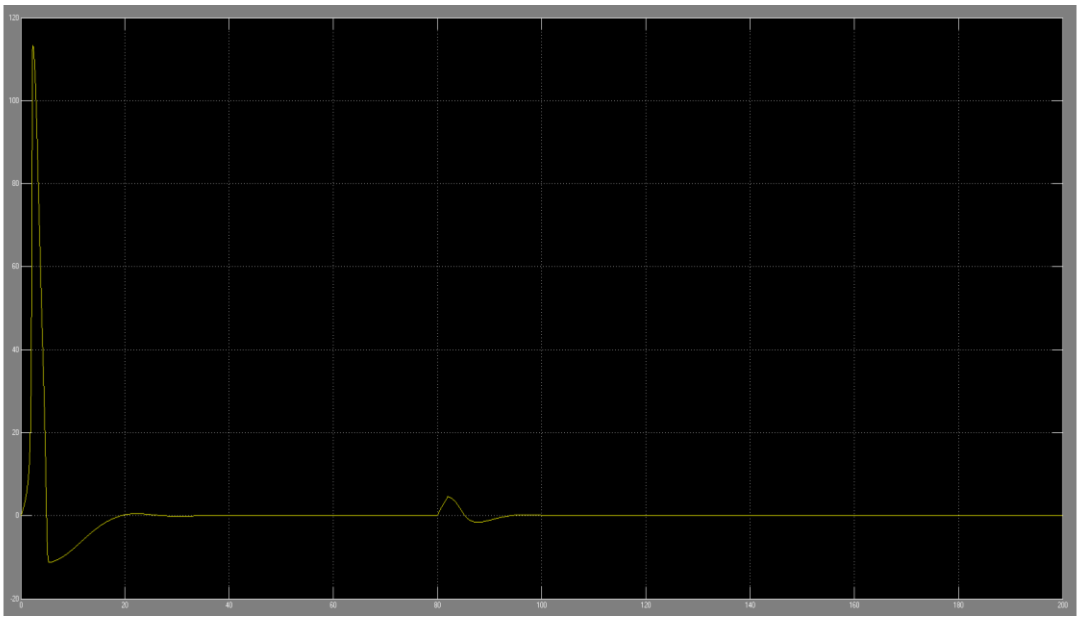


Figure 3-8: Action Potential with an input of 3 nA stimulus

3.2 Conduction Velocity

Conduction velocity [10] is the speed at which the action potential moves from one neuron to the other. Conduction velocity plays a very important role in the learning functions. Because it is the conduction velocity which defines the delay in the transmission and reception of the impulse along with the couple of other variables such as the neurotransmitters, temperature and ionic concentrations [11]. We have varied the different dimensions of a neuron to see their effect on the conduction velocity of a neuron.

We simulated two models, in one model we kept the radius of the neurons constant and varied the lengths of the neuron. In the second model we kept the length of the neurons constant and varied the radius of the neurons. Through this process we could see the effect of length and radius on the conduction velocity of the neuron. We used the same network of neurons so that we could better judge the effect of length and radius on the conduction velocity

Constant Radius with varying lengths of neurons

We used the following network to calculate the conduction velocity of the neurons. The radius and length of the neuron are provided in the Fig 3-9. The result is summarized in table 1 and shown in Fig 4:10.

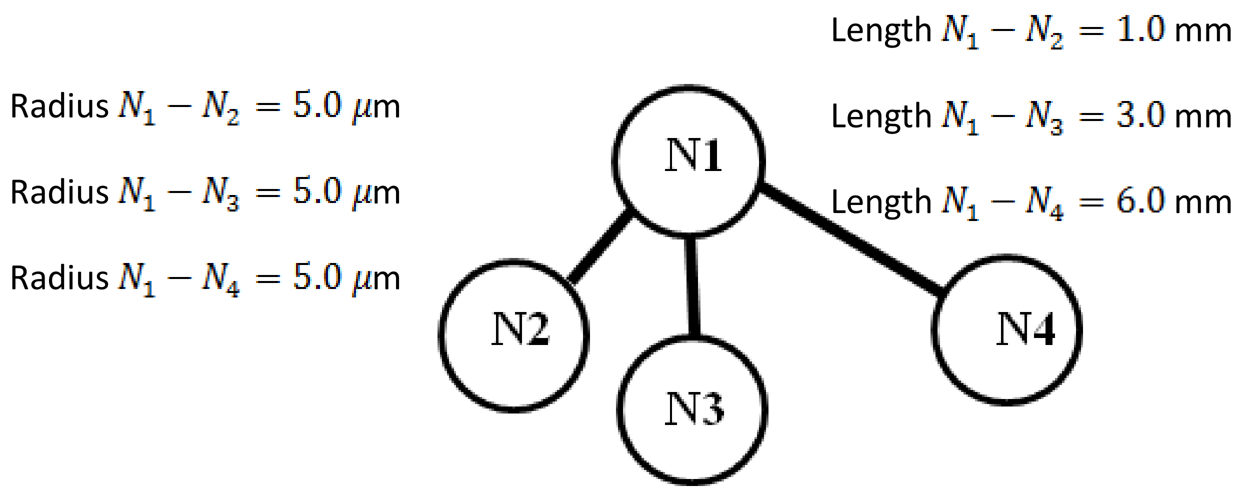

Radius $N_1 - N_2 = 5.0\ \mu$m
Radius $N_1 - N_3 = 5.0\ \mu$m
Radius $N_1 - N_4 = 5.0\ \mu$m
Length $N_1 - N_2 = 1.0$ mm
Length $N_1 - N_3 = 3.0$ mm
Length $N_1 - N_4 = 6.0$ mm
N1
N2
N3
N4

Figure 3-9: Network of Neurons for constant radius

	Neuron	Delay	Velocity
	N2	43.38 µsec	22.82 m/s
	N3	131.45 µsec	22.82 m/s
	N4	262.90 µsec	22.82 m/s

Table 1: Result for Neural Network with Constant Radius

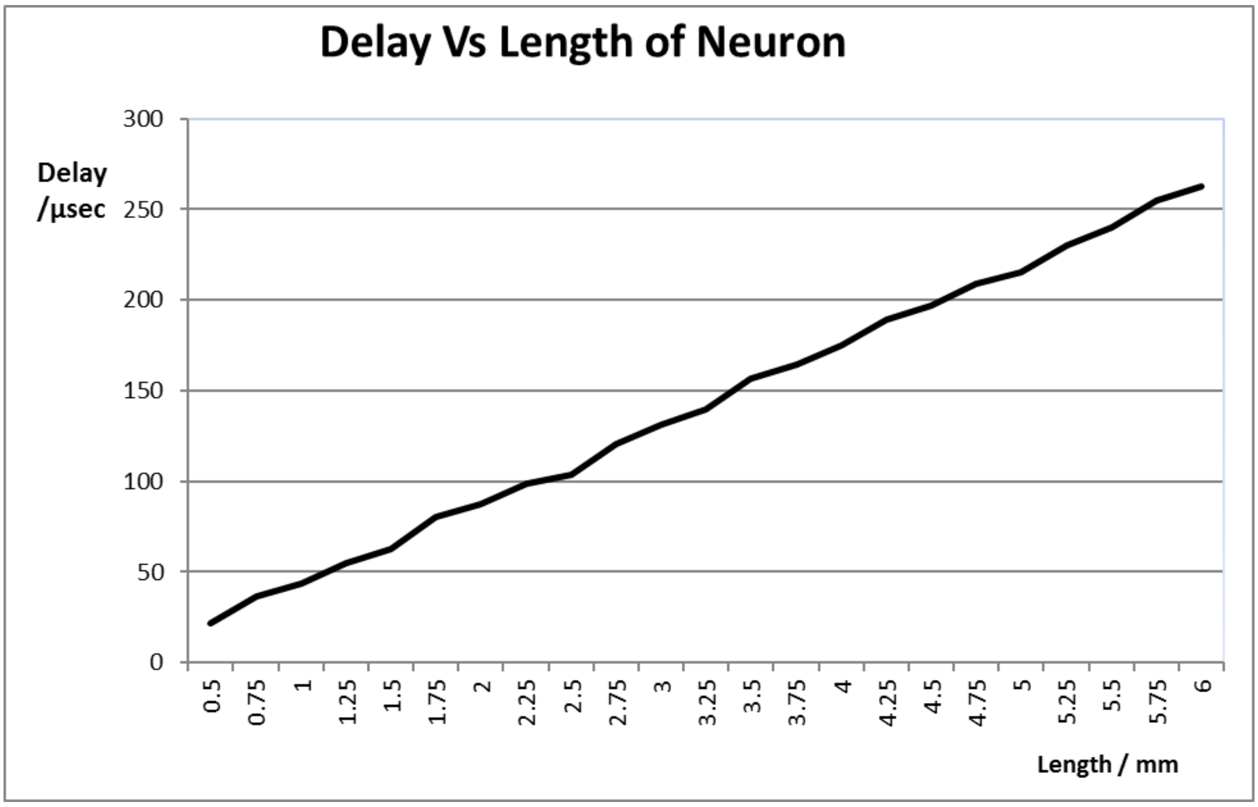
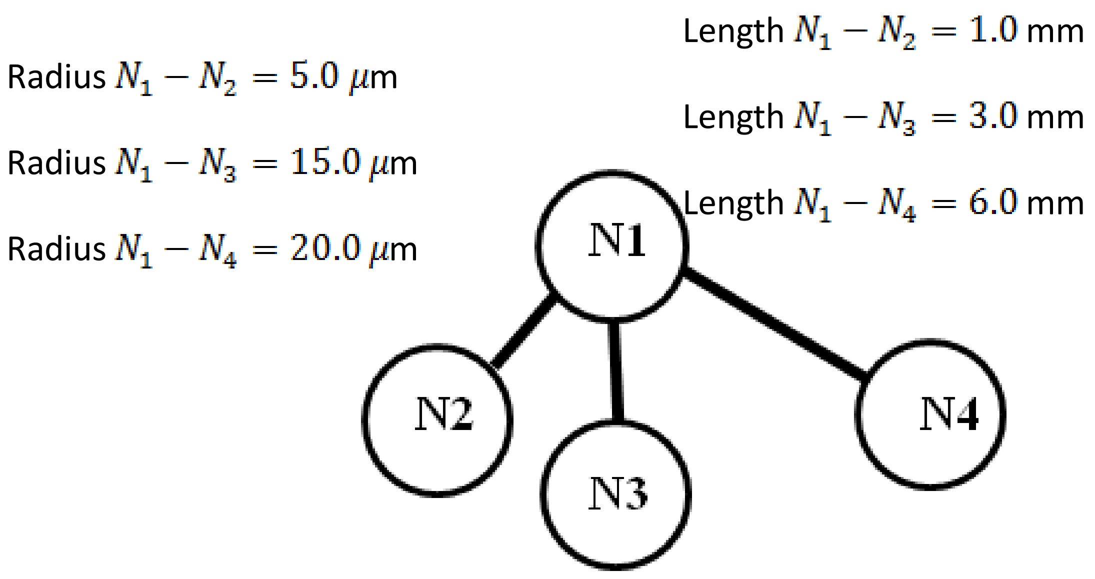


Figure 3-10: Graph of the delay due to the length of the neuron

Constant Radius with varying lengths of neurons

We use the following network to model the effect of radius on the conduction velocity. In this network shown in Fig 3-11 the lengths of the neurons have been kept constant as in the previous model. The reason of doing this was that we could see the effect of radius on the conduction velocity and would compare it with the previous result. The results are summarized in Table 2 and shown in Fig 3-12



Radius $N_1 - N_2 = 5.0\ \mu$m
Radius $N_1 - N_3 = 15.0\ \mu$m
Radius $N_1 - N_4 = 20.0\ \mu$m
Length $N_1 - N_2 = 1.0$ mm
Length $N_1 - N_3 = 3.0$ mm
Length $N_1 - N_4 = 6.0$ mm
N1
N2
N3
N4

Figure 3-11: Network of neurons to see the effect of varying radius

	Neuron	**Delay**	**Velocity**
1	N2	43.81 µsec	22.82 m/s
2	N3	75.89 µsec	39.52 m/s
3	N4	131.45 µsec	45.64 m/s

Table 2-2: Result of the Network

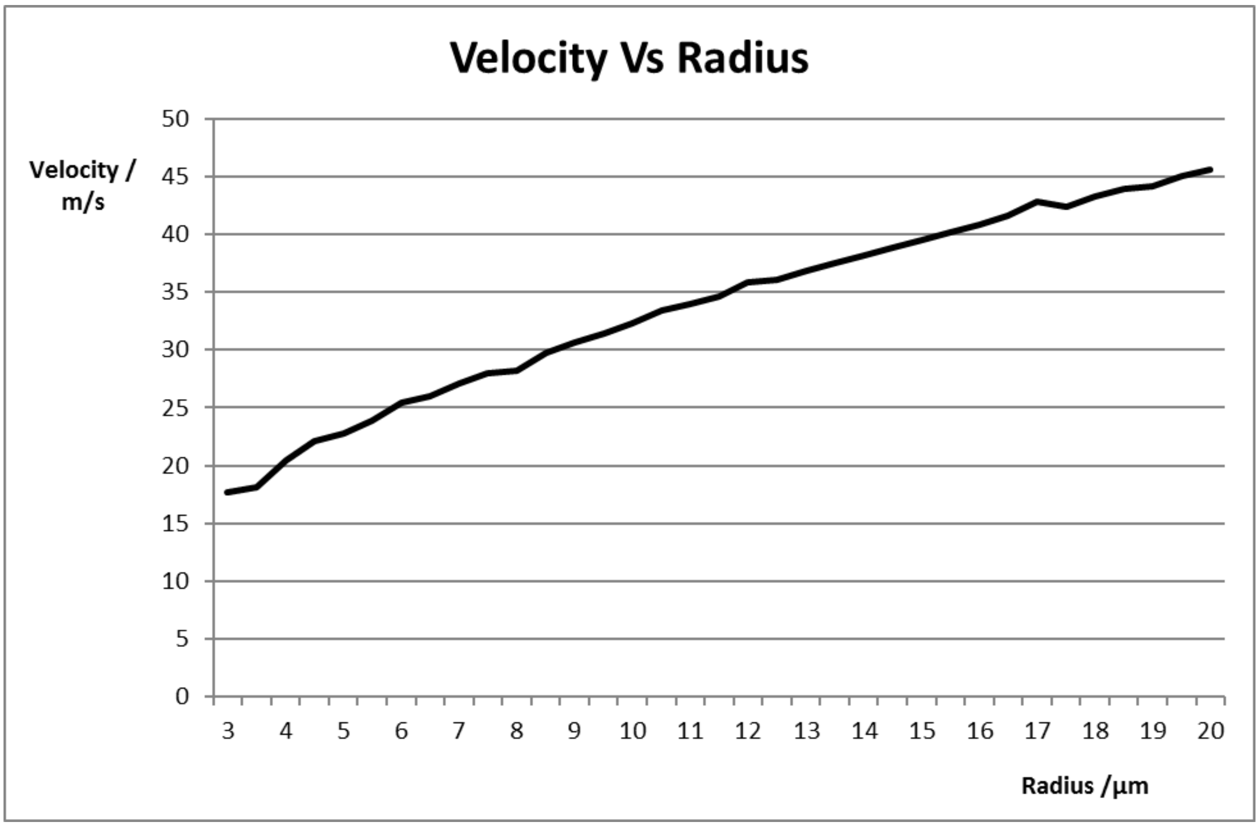


Figure 3-12: Graph of delay vs. the radius of the neuron

In Fig 3-13we summarize the results. The graph clearly shows the effect of radius and length on the delay in reception of the action potential. The results clearly show as we increase the length of the neuron the delay increases and as the radius of the neuron increases the delay decreases. The conduction velocity is the gradient of the curves and we can see that as the radius of the neurons increase the conduction velocity increases. The length of the neuron does not affect the conduction velocity.

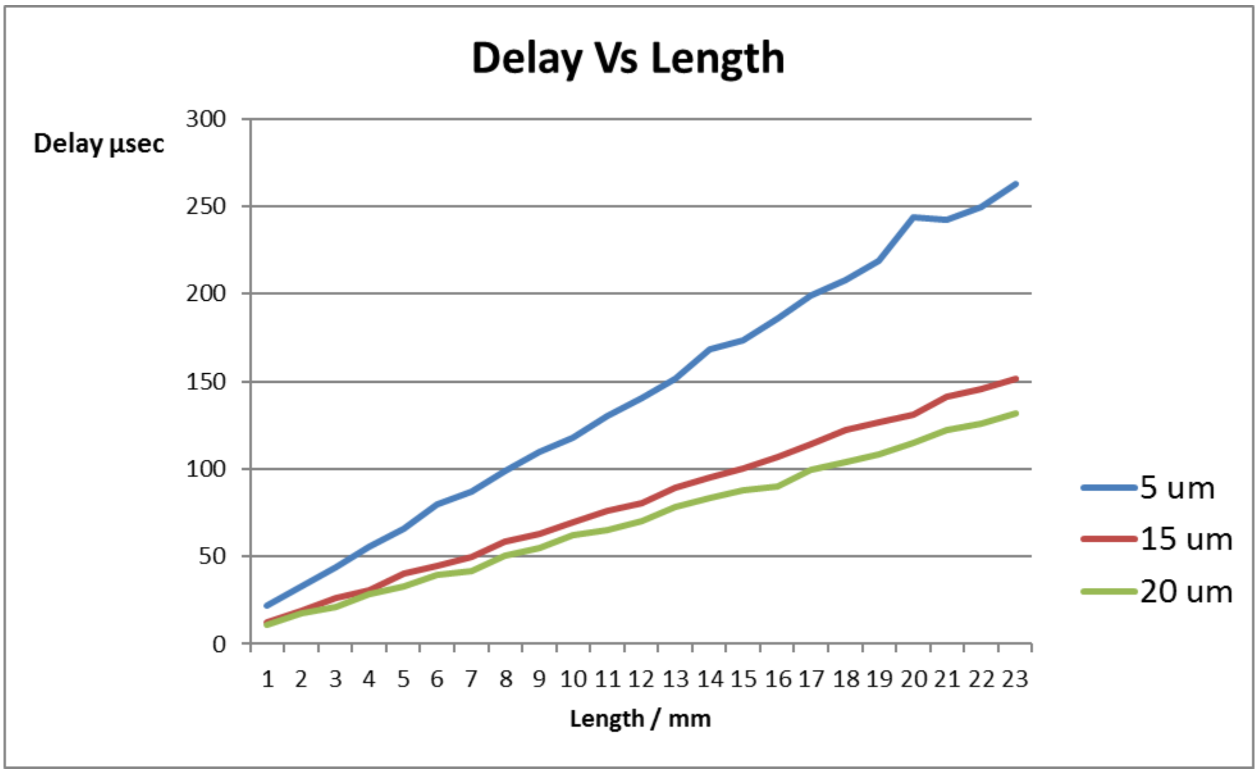


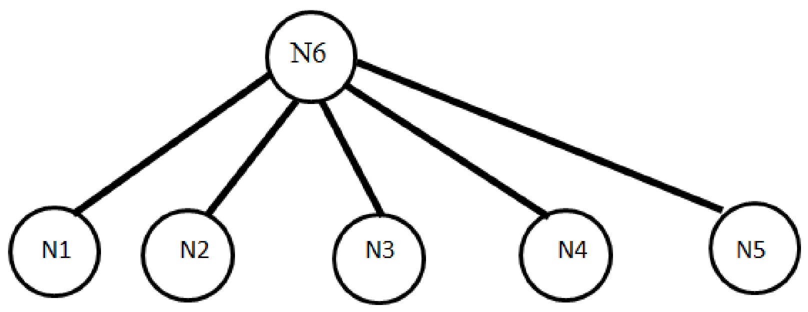
Figure 3-13: Summary of the results

3.3 Spiking Time Dependent Plasticity

3.3.1 Uniform Spiking Frequency

For uniform spiking frequency we use uniform spiking frequency for each of the neurons. The following figure shows the network shown in Figure 3-14 used and the dimensions of the neurons. The Figure 3-15 shows the synaptic weights before the firings and Figure 3-16 shows the synaptic weights after firing.

As the firing rate is constant the only factor effecting the weights are the dimensions. As the lengths of the neurons differ in which case the speed of the action potential differs and hence the time the presynaptic action potential arrives differs because of the lengths, as the radius is constant.



Radius $N_1 - N_6 = 5.0\ \mu m$	Length $N_1 - N_6 = 1.0\ mm$
Radius $N_2 - N_6 = 5.0\ \mu m$	Length $N_2 - N_6 = 3.0\ mm$
Radius $N_3 - N_6 = 5.0\ \mu m$	Length $N_3 - N_6 = 4.0\ mm$
Radius $N_4 - N_6 = 5.0\ \mu m$	Length $N_4 - N_6 = 5.0\ mm$
Radius $N_5 - N_6 = 5.0\ \mu m$	Length $N_5 - N_6 = 6.0\ mm$

Figure 3-14: Network of Neurons for constant radius

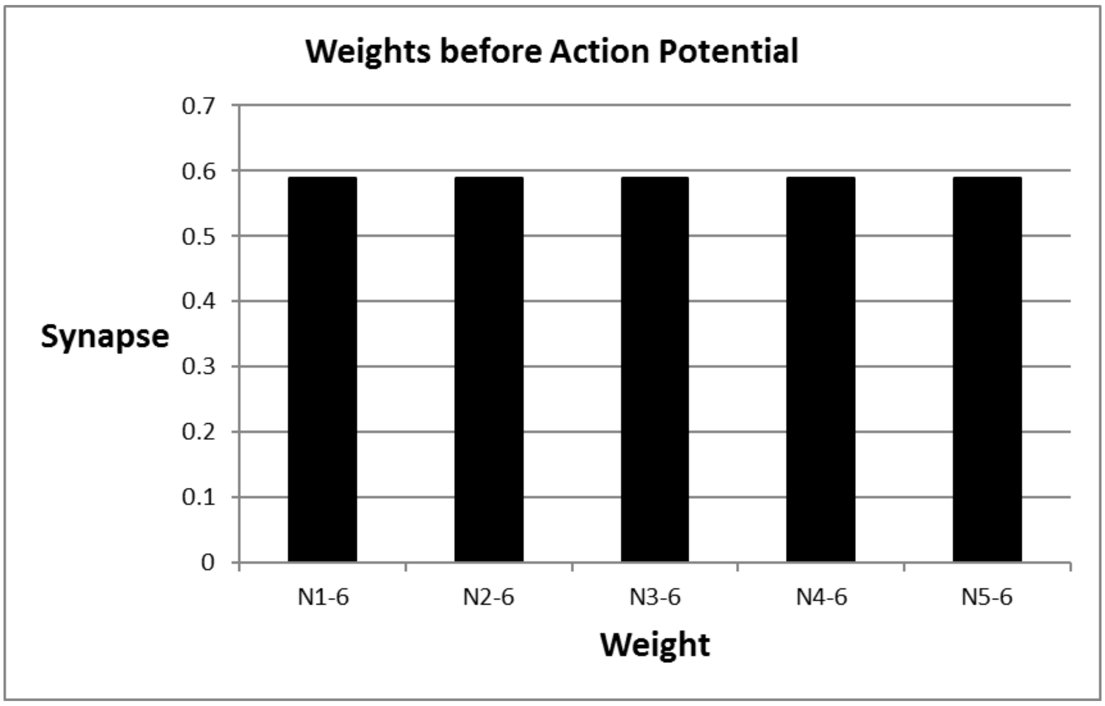


Figure 3-15: Synaptic weights before action potential

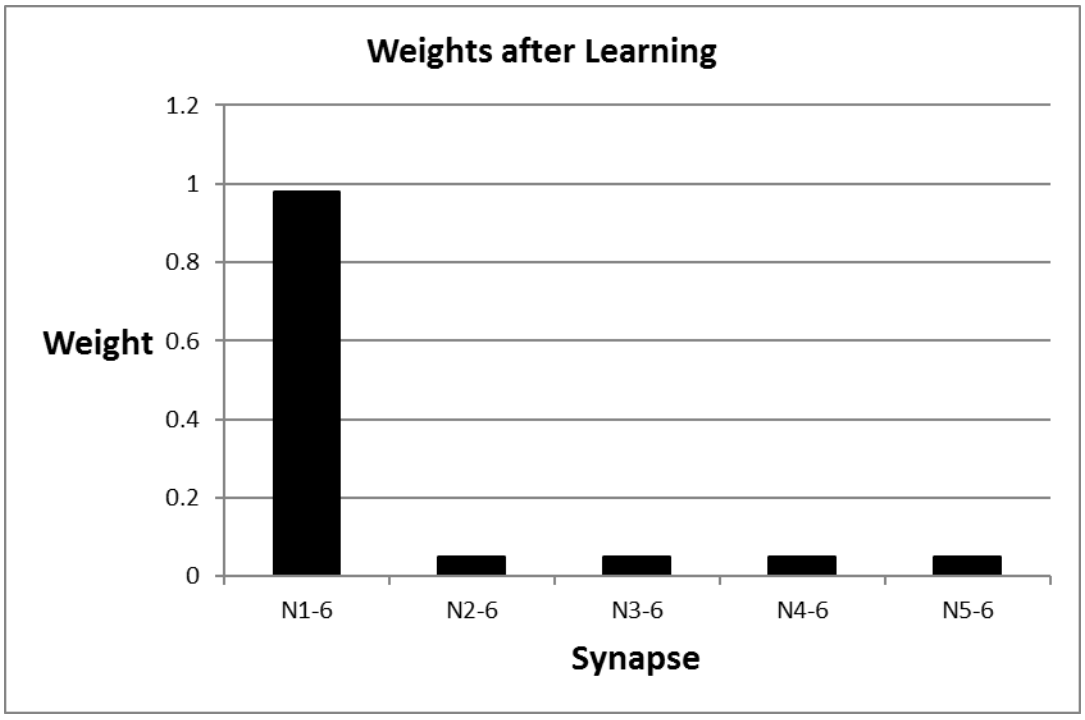
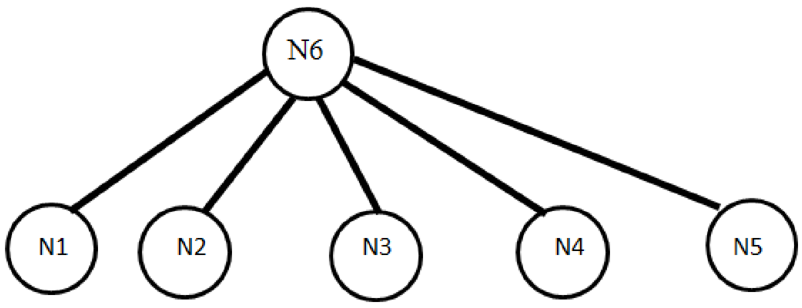


Figure 3-16: Synaptic weights after action potential

3.3.2 Different Spiking Frequency

For different spiking frequency we use uniform spiking frequency for each of the neurons. The following figure shows the network shown in Figure 3-17 used and the dimensions of the neurons and Figure 3-18 shows the spiking frequency used.. The Figure 3-19 shows the synaptic weights before the firings and figure 3-20 shows the synaptic weights after firing

As the firing rate are different and the same dimensions are used as previously the only factor effecting the weights is frequency. As we can see from the results below the weights after the action potential follow the same pattern as that of the frequency distribution. Hence showing that the synaptic weights are dependent on the frequency of the action potentials which is also known as the firing rate of the action potentials of each neurons.



Radius $N_1 - N_6 = 5.0\ \mu m$	Length $N_1 - N_6 = 1.0\ m$m
Radius $N_2 - N_6 = 5.0\ \mu m$	Length $N_2 - N_6 = 3.0\ mm$
Radius $N_3 - N_6 = 5.0\ \mu m$	Length $N_3 - N_6 = 4.0\ mm$
Radius $N_4 - N_6 = 5.0\ \mu m$	Length $N_4 - N_6 = 5.0\ mm$
Radius $N_5 - N_6 = 5.0\ \mu m$	Length $N_5 - N_6 = 6.0\ mm$

Figure 3-17: Neural Network for constant radius

<u>Spiking Frequency of Neurons</u>

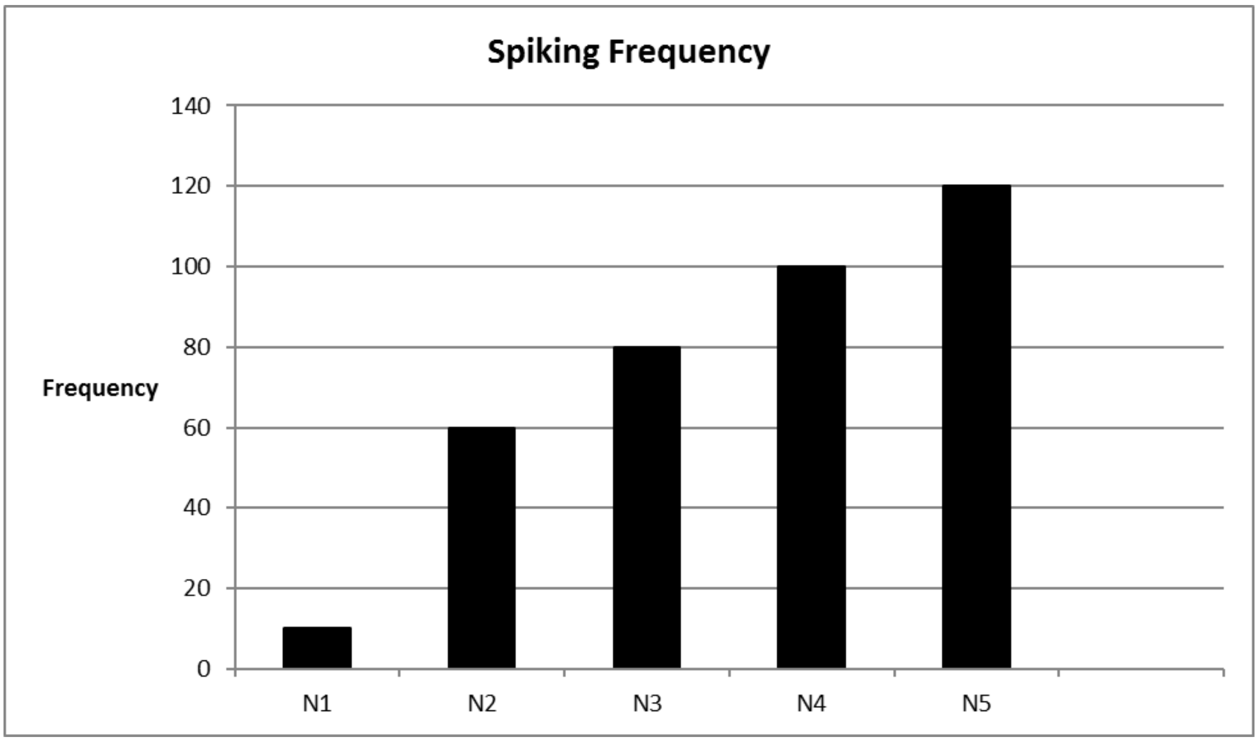


Figure 3-18: Synaptic weights after action potential

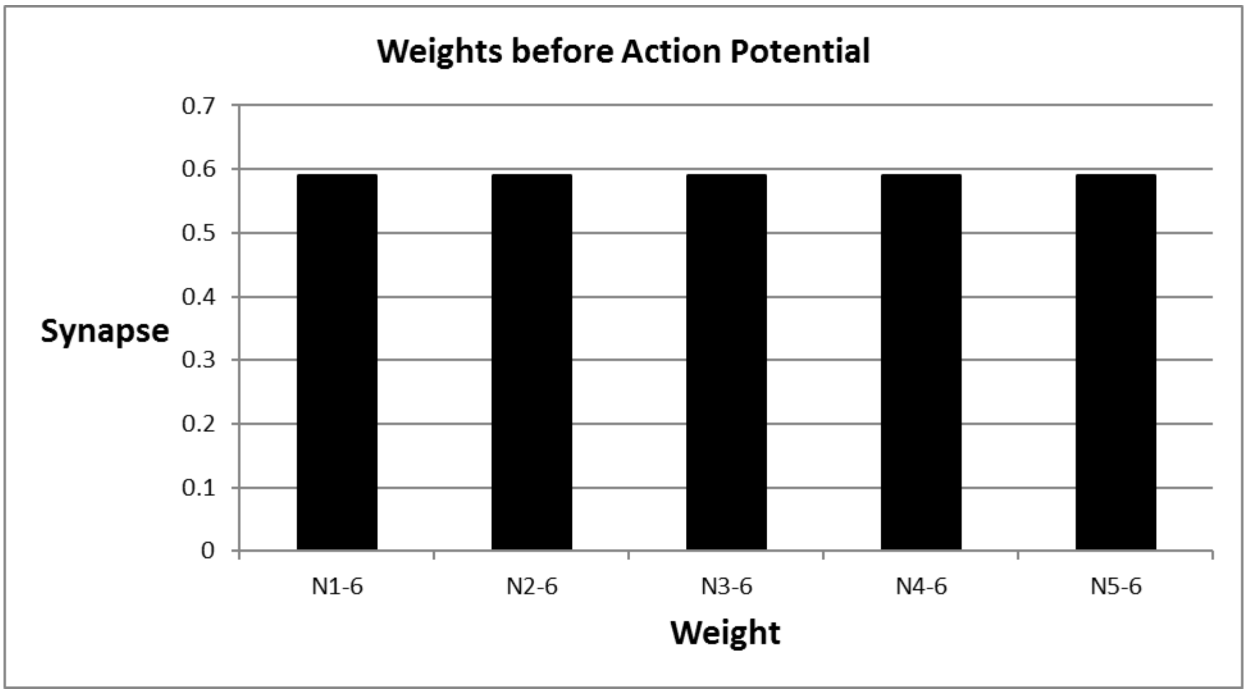


Figure 3-19: Synaptic weights after action potential

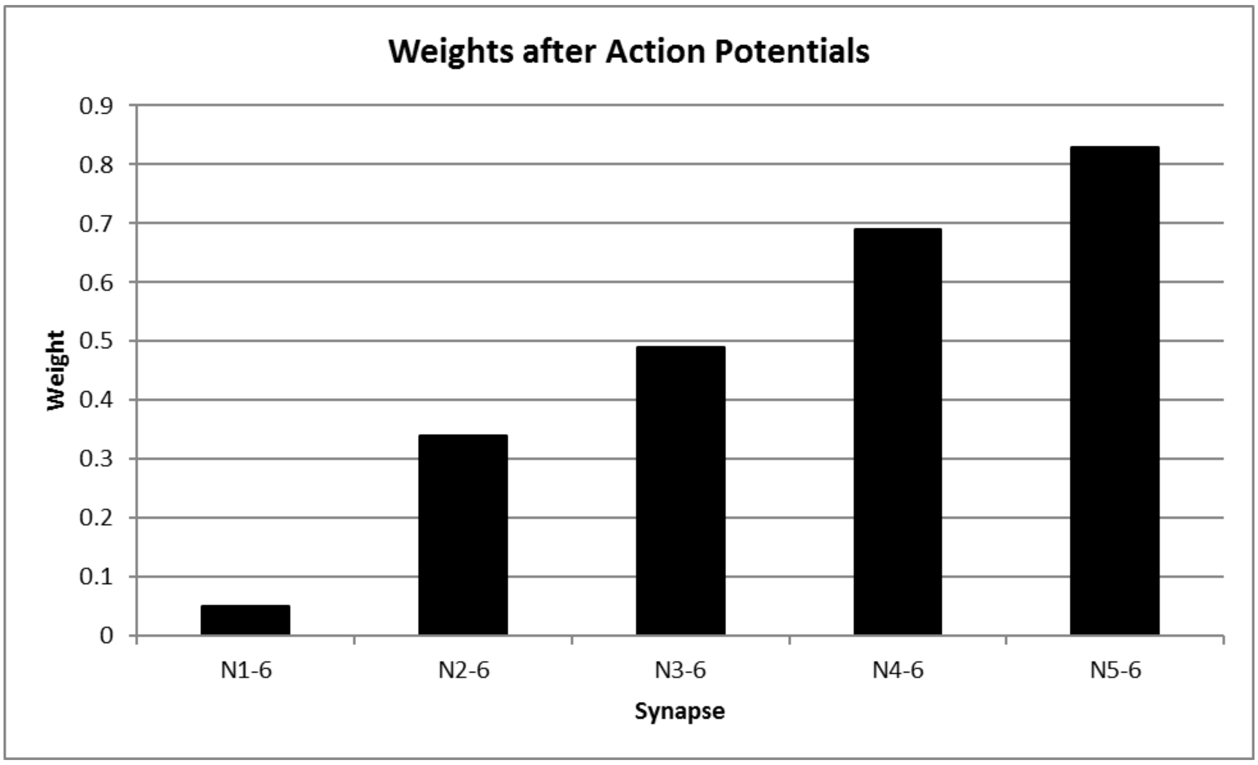


Figure 3-20: Synaptic weights after action potential

CHAPTER 4: CONCLUSION

This was of the first time that any model was made using the KOCH model for modeling a neural network that involved learning function and conduction velocity functions in the model. Previously work [13] has been done in simulating the neural network with learning functions but the neurons in those networks used Hodgkin Huxley model for neural simulation. KOCH model is a comprehensive model of a neuron compared to Hodgkin Huxley Model and hence the results are also more comprehensive. The results of our model are consistent with the previous works [13] and shows that KOCH model does provide and alternative more comprehensive neural model that could be used for modeling neurons.

Our Results showed that the speed of propagation of the action potential is related to the diameter of the neuron, which is consist ant with the previous findings [11]. The delays involved in the propagation of action potential are directly related to the features of a neuron. The delay decreases as the radius of the neuron increases and the delay increases when the length of the neuron increases.

We used the model with learning function using Spike Time Dependent Plasticity which showed the effect of different dimensions of a neuron on learning. The effects of delay on learning can be seen from the results. Even a time difference of 1-2 mSec have a profound effect on learning. Along with the effect of delays we also showed the effect of the frequency of the action potential affecting learning, as the frequency of action potential increased so did the learning weights.

For future the model could incorporate other functions such as neural degeneration [14], effects of temperature on propagation [15]. Incorporating such function would help in better understanding of many neural diseases that involve such functions. Functions such as Synaptogenesis: forming of new connections between neuron and Neurogenesis: creation of neurons as the requirement arises. The additions of these functions would further bring the model closer to the actual working of the brain.

In future the code should be parallelized so that as we increase the number of neurons to millions and to even billions of neurons as the estimated number of neurons in average human brain is approximately 100 billion neurons. Parallelizing the code is extremely essential to mimic the working of the brain.

REFERENCES

[1] L.F. Abbott, Lapicques. “Introduction of the integrate-and-fire model neuron” (1907). Brain Res. Bull., 50(5-6):303–304, 1999.

[2] Y. Amitai, J.R. Gibson, M. Beierlein, P.L. Patrick, A.M. Ho, B.W. Connors, and D. Golomb. “The spatial dimensions of electrically coupled networks of interneurons in the neocortex” J. Neurosci., 22(10):4142–4152, 2004.

[3] R. Azouz and C. M. Gray. “Cellular mechanisms contributing to response variability of cortical neurons in vivo” J. Neurosci., 19:2209–2223, 1999.

[4] M. Beierlein, J.R. Gibson, and B.W. Connors “ A network of electronically coupled interneurons drives synchronized activity in neocortex” Nature Neuroscience, 3:904–910, 2000.

[5] P.C. Bressloff and S. Coombes “Synchrony in an array of integrate-and-fire neurons with dendritic structure” Physical Review Letters, 78:4665–4668, 1997.

[6] E. Brown, J. Moehlis, and P. Holmes “On the phase reduction and response dynamics of neural oscillator populations” Neural Comp., 16:673–715, 2004.

[7] A. L. Hodgkin and A. F. Huxley “A quantitative description of membrane current and its applications to conduction and excitation in nerve" J.Physiol. (London.), 116 (1952), pp. 500–544.

[8] A.L. Hodgkin “The local electric changes associated with repetitive action in a non-medulated axon” J. Physiol., 107:165–181, 1948.

[9] Chirst of Koch & Idan Segev, “Multiple Channels and Calcium Dynamics” in book “Methods in neuronal modeling” MIT Press Cambridge 1989. Pages 97-133

[10] Guo-qiang Bi and Mu-ming Poo “Synaptic modification by correlated activity: Hebb’s Postulate Revisited” Annu. Rev. Neurosci. 2001. 24:139–66

[11] Yang Dan and Mu-ming Poo “Spiking Timing-Dependent Plasticity of Neural Circuits” Neuron, Vol. 44, 23–30, September 30, 2004

[12] Ichiji Tasaki, Gen Matsumoto “On the Cable Theory of Nerve Conduction” Bulletin of Mathematical Biology Volume 64, Issue 6, November 2002, Pages 1069–1082

[13] Aydn, Gülümser MDı; Keleş, Işk MDı; “Sensitivity of Median Sensory Nerve Conduction Tests in Digital Branches for the Diagnosis of Carpal Tunnel Syndrome” American Journal of Physical Medicine & Rehabilitation: January 2004 - Volume 83 - Issue 1 - pp 17-21

[14] Lyle N. Long “An Adaptive Spiking Neural Network with Hebbian Learning “ IEEE Symposium Series on Computational Intelligence, Paris, France, 2011

[15] Z J Koles, M Rasminsky, “A computer simulation of conduction in demyelinated nerve fibres” The Journal of Physiology (1972) Volume: 227, Issue: 2, Pages: 351-364

[16] K Todnem, G Knudsen, T Riise, H Nyland, J A Aarli, “The non-linear relationship between nerve conduction velocity and skin temperature” J Neurol Neurosurg Psychiatry 1989;52:497-501 doi:10.1136/jnnp.52.4.497

[17] B. Doiron, C. Laing, and A. Longtin. Ghostbursting "A novel neuronal burst mechanism" J. Comp. Neurosci., 12:5–25, 2002

[18] J. Duijnhouwer, M.W.H. Remme, A. van Ooyen, and J. van Pelt "Influence of dendritic topology on firing patterns in model neurons" Neurocomputing, 38–40:183–189, 2001.

[19] C. Koch and I. Segev "The role of single neurons in information processing" Nature, 3:1171–1177, 2000.

[20] M.A. Long, C.E. Landisman, and B.W. Connors "Small clusters of electrically coupled neurons generate synchronous rhythms in the thalamic reticular nucleus" J. Neurosci., 24(2):341–349, 2004.

[21] L. Rela and L. Szczupak. Gap junctions "Their importance for the dynamics of neural circuits" Molecul. Neurobio., 30(3):341–357, 2004.

[22] J. Rinzel and G.B. Ermentrout. "Analysis of Neural Excitability and Oscillations In: Koch C, Segev I (eds) Methods in Neuronal Modelling: from Synapses to Networks" MIT Press, Cambridge, Mass. pp 135-171, 1989.

[23] X.J. Wang. "Fast firing and short-term synaptic plasticity: A model of neocortical chattering neurons" Neuroscience, 89(2):347–362, 1999.

[24] W. Rall. "Theory of physiological properties of dendrites" Ann. N.Y. Acad. Sci., 96:1071–1092, 1962.

[25] M.J. Chacron, K. Pakdaman, and A. Longtin "Interspike interval correlations, memory, adaptation, and refractoriness in a leaky integrate-and-fire model with threshold fatigue" Neural Comp., 15:253–278, 2003.

[26] Hursh, J. B. (1939). "Conduction velocity and diameter of nerve fibers" Am. J. Physiol. 127,131–153.

APENDIX A

Algorithm for Spiking Time Dependent Plasticity

The algorithm used for simulating the spiking time dependent plasticity is shown below. This algorithm is used for simulating STDP in each neuron. This is very simple and accommodates all the properties of STDP, without putting any burden on the simulations.

```
For each synapse
  if timeold(PostNeuron) < timenew(PreNeuron)< imenew(PostNeuron)
          weight = weight + dwPlus
    else
          weight = weight + dwMinus
    dwPlus = dwMinus  = $\alpha e^{dt}$
    dt = timenew(PreNeuron) - timenew(PostNeuron
```

Matlab Code

The following is the MATlab code used to program the model. The code is divided into different functions which enable it to be edited for debugging and adding different parameters for the network. The names of the functions are shown in bold and are in the center. The detail of each function would be found in the functions, additional information about the formulas and the equations used are given the standard MATlab comments, which makes it easy to understand the code.

The following function tau_mNa, tau_mK3, tau_mK2, tau_mK1, tau_mCa, tau_mC, tau_mC, tau_mAHP, tau_hNa, tau_hK2 are used to calculate function mNA, hNA. mCa, hCa, mC, hC, mAPH, hAPH, mK1, mK2, hK2, Mk3.

tau_mNa

```
function y = tau_mNa(v)

y = 2 ./ ( alpha_mNa(v) + beta_mNa(v));

end
```

tau_mK3

```
function y = tau_mK3(v)

sum = v+35;

y = (1000.0) ./ ( 3.3 .* ( (exp(sum./40) + exp(-sum./20)) ));

end
```

tau_mK2

```
function y = tau_mK2(v)

y = 1 ./ (alpha_mK2(v) + beta_mK2(v));

end
```

tau_mK1

```
function y = tau_mK1(v)

y = 1.38;

end
```

tau_mCa

```
function y = tau_mCa(V)

sum = (V+6)/16;

y   = 7.8/( exp(sum) + exp(-sum) );

end
```

tau_mC

```
function y = tau_mC(V,Ca)

sum = V/24;

b   = 0.1*exp(-sum);

f   = 250*Ca*exp(sum);

y   = 1 / (f + b);

end\
```

tau_mAHP

```
function y = tau_mAHP(Ca)

 f = 1.25e8 *(Ca)*(Ca);

b = 2.5;

y = 1000/(f+b);
```

```
end
```

tau_hNa

```
function y = tau_hNa(v)

y = 2 ./ ( alpha_hNa(v) + beta_hNa(v));

end
```

tau_hK2

```
function y = tau_hK2(v)

if v < -25

y = 6000;

else

y = 50;

end

end
```

tau_hK1

```
function y = tau_hK1(v)

if v<-80

y = 50;

else

y = 150;

end

end
```

STDP

This is the implementation of the SPTD algorithm.

```
function w_cur= STDP(s,w_cur)

lam_pot = 0.1;

lam_dep = 0.1;

mui     = 1;

MAX     = 1.0;

MIN     = 0.2;

% w_curr(1)   = 1;

i = 2;

if (s<=0)       % Potentiation (pre-post)

tem     = lam_pot*(MAX-w_cur)^mui;

delta_w = tem*(exp(s/1));

y = exp(s/1);

w_cur = w_cur+delta_w;

else

tem     = lam_dep*w_cur^mui;

delta_w = tem*(-exp(-s/1));

y = -exp(-s/1);

w_cur = w_cur+delta_w;

if(w_cur<MIN)

w_cur = MIN;
```

```
end

end

i = i+1;

end
```

set_Pump

This code sets up the portion of the discretization matrix A due to the Calcium Pump. The method of calculations is shown as comments in the code. Please look at the comments to understand the method of solving the problem

```
function [A4,DV4] = set_Pump(X,A4,DV4,N,V,Parameter)

format long

%{

...

==========================================================


Parameter (1) = rsize;
Parameter (2) = rcore;
Parameter (3) = delta_t;
Parameter (4) = forward_binding_rate;
Parameter (5) = backward_binding rate;
Parameter (6) = Diffusion Constant
Parameter (7) = B_iT;
Binding Concentration below
level n

Parameter (8) = B_nT;

Binding Concentration at
level n
Parameter (9) = C_i;

Ca Concentration below
level n

Parameter (10) = C_n;

Ca Concentration at

level n

===================================================

        ...

        ...
        %}

rsize                   = Parameter (1);

rcore                   = Parameter (2);

delta_t                                 = Parameter (3);

forward_binding_rate                    = Parameter (4);

backward_binding_rate                   = Parameter (5);

Diffusion_Constant                      = Parameter (6);

B_iT                = Parameter (7);

B_nT                = Parameter (8);

C_i                 = Parameter (9);

C_n                 = Parameter (10);

C                   = zeros(1,N+1);

C                   = X(1:N+1);

B                    = zeros(1,N+1);
```

```
B                    = X(N+2:(2*N)+2);

BT                    = zeros(N+1,B_iT);

BT(N)            = B_nT;

%=========================================================

%       Add calcium pump term
%       C_N, t+delta_t ( 1+ delta_t /(2*tau_pump )) =
%               C_N, t ( 1 ? delta_t /(2*tau_pump )
%               + delta_t /tau_pump*Caˆeq

%=========================================================


calcium_equil   = 5.0e-5;
tau_pump        = 17.7*exp(V/35.0);
tau2            = delta_t/tau_pump;
A4(N+1,N+1)     = 1.0 + 0.5*tau2;
DV4(N+1)        = (1.0 - 0.5*tau2)*C(N+1) + (calcium_equil*tau2);

end
```

set_Ica

This code sets up the portion of the discretization matrix A due to the Calcium current ICA.

```
function [A3,DV3] = set_ICA(X,A3,DV3,N,m_CA1,V,Parameter)

format long

rsize                  = Parameter (1);

rcore                  = Parameter (2);

delta_t                = Parameter (3);

forward_binding_rate    = Parameter (4);

backward_binding_rate   = Parameter (5);

Diffusion_Constant      = Parameter (6);
```

```
B_iT                  = Parameter (7);

B_nT                   = Parameter (8);

C_i                   = Parameter (9);

C_n                   = Parameter (10);

C                     = zeros(1,N+1);

C      = X(1:N+1);

B      = zeros(1,N+1);

B      = X(N+2:(2*N)+2);

BT      = zeros(N+1,B_iT);

BT(N)   = B_nT;
```

Now add dynamics due to I CA1
([Ca+2]_N,t+delta_t ? [Ca+2]_N,t)/ delta_t = ?I_CA1 /(2F V_n)
or

C(N) at t+delta_t =
C(N) ?I_CA1*delta_t /(2 F V_N)

This means we set
A3 (N,N) = 1.0
DV3(N) = C(N) ?I_CA1*delta_t /(2 F V_N)

where these parameters are defined as follows:
Calcium Concentration in last shell update

note .1 micro m = 10ˆ?7 m = 10ˆ?5 cm
d Ca/ dt = mM/(liter ms)
I_CA1 = nAmps/ sec
    = 10ˆ?9 coulombs / sec
    = 10ˆ?9 coulombs / sec * 1 sec /10ˆ3 ms
    = 10ˆ?12 couombs / ms

cell is 20 micro m in radius = 20.0 x 10ˆ?4 cm
rcore is 19 micro meter in radius = 19.0 x 10ˆ?4 cm
each shell is .1 micro meter thick = 1.0 x 10ˆ?5 cm

shell N is rcore + N x rsize in radius

= 19 x 10ˆ?4 cm + 10_*(1.0 x 10ˆ?5) cm

= (19+1) x 10ˆ?4 cm

so shell is 10ˆ?5 cm thick

diff = (20.0 x 10ˆ?4 cm)ˆ3 ?(19.9 x10ˆ?4cm)ˆ3

diff = (8,000 ?78805.990) x 10ˆ?12 cc

diff = 1.194010 x 10ˆ2 x10ˆ?12 cc

diff = 1.194010 x 10ˆ?10 cc

diff = 1.194010 x 10ˆ?13 liters

volume of shell N is 4/3*pi*diff

= 5.0015 x 10ˆ?13 liters

So

V_n = volume of shell N:

= 5.0015 x 10ˆ?13 liters

Faraday's Constant is FC = 9.649 x 10ˆ13 coulomb/mM

So 1/(2.0_FC x V n) = mM/(coulomb liters)

And since 2.0 x FC x V_N = 96.5189 x 10ˆ13 x 10ˆ?13 (coulomb liter)/mM

= 96.5189 (coulomb l i t e r)/mM

= 9.65189 x 1 0 ˆ 1 (coulomb l i t e r)/mM

1 / (2.0 x FC x V_n) = 0.1036 x 10ˆ?1 mM/(coulomb liter)

= 1.036 x 10ˆ?2 mM/(coulomb liter)

I CA1 / (2.0 x FC x V_n)

= 10ˆ?12 x 1.036 x 10ˆ?2 coulombs mM

????????*???????????????

ms (coulomb liter)

= 1.036 x 10ˆ{?14} mM/(l i t e r ms)

h_CA1_0 = 0.01;

```
h_CA1           = h_CA1_0 /(h_CA1_0+C(N+1) );
g_CA1_bar       = 0.116;
CA_O            = 4.0;
Ryd             = 8.31;          %   Rydberg's Constant
T               = 276.0 + 22.0;  %   Kelvin Temperature
F               = 9.649e+4;      %   Faraday's constant
r31             = rcore + ((N*rsize)^3.0);
r32             = rcore + (((N-1)*rsize)^3.0);
V_N             = (4.0/3.0)*( r32-r31 );

RTF             = Ryd*(T/F)*1e+3;        %    express in mV

E_CA            = 0.5*RTF*log10(CA_O/(C(N+1)) );
I_CA1           = g_CA1_bar*m_CA1*h_CA1*(V-E_CA);
Ca_Term         = (1.0e-9)/(2.0*V_N*F);
A3(N+1,N+1) = 1.0;
DV3(N+1)        = X(N+1) - ( I_CA1*Ca_Term*delta_t );

end
```

set_diffusion

This code sets up the portion of the discretization matrix A due to the Calcium diffusion.

```
function [A2, DV2] = set_diffusion (X,A2,DV2,N,Parameter)

format long

    Parameter (1) = rsize;
    Parameter (2) = rcore;
    Parameter (3) = delta_t;
    Parameter (4) = forward_binding_rate;
    Parameter (5) = backward_bindingrate;
    Parameter (6) = Diffusion_Constant;
    Parameter (7  = B_iT;

    Binding Concent ration below
    leveln

    Parameter (8) = B_nT;
    Binding Concentration at
```

```
        level n

        Parameter (9) = C_i;
        Ca Concentration below
        level n
        Parameter (10) = C_n;
        Ca Concentration at
        level n


rsize                   = Parameter (1);
rcore                   = Parameter (2);
delta_t                 = Parameter (3);
forward_binding_rate    = Parameter (4);
backward_binding_rate   = Parameter (5);
Diffusion_Constant      = Parameter (6);
B_iT                    = Parameter (7);
B_nT                    = Parameter (8);
C_i                     = Parameter (9);
C_n                     = Parameter (10);

C                       = zeros(1,N+1);
C                       = X(1:N+1);


B     = zeros(1,N+1);

B     = X(N+2:(2*N)+2);

BT     = ones(N+1);

BT     = BT.*B_iT;

BT(N+1) = B_nT;



DF1 is set once ; does not depend on delta_t

units:

          R is cm
          DF1 is cmˆ2/ms 1/(cmˆ2) cm
              cm/ms
          DF1b is cm
```

```
                DF2 is cm/ms

                These terms in A times C or B give
                (DF1?DF2)(C or B) is
                        cm/ms mM/( liter ms)
                        (cm mM)/( liter msˆ2)

DF1 = zeros(N+1,N+1);
DF2 = zeros(N+1,N+1);
DF3 = zeros(N+1,N+1);
DF4 = zeros(N+1,N+1);
R   = zeros(1,N+1);

mu = ( Diffusion_Constant ) / ( 2.0*rsize*rsize);

for i=1:N+1
    R(i)        = rcore + (i-1)*rsize ;
end

DF1(1,1)    = -3.0*rsize*rsize / rcore;
DF1(1,2)    = 3.0*R(2)*rsize*rsize /( rcore*rcore );

for i=2:N
    DF1(i,i-1) = R(i-1);
    DF1(i,i)   = -2.0*R(i);
    DF1(i,i+1) = R(i+1);
end

DF1(N+1,N)      = R(N);
DF1(N+1,N+1)    = -R(N);

DF4         = (delta_t*mu).*DF1;

RDiag = zeros(N+1,N+1);
Temp  = zeros(1,N+1);

for i=1:N+1
     RDiag(i,i) = R(i);
end

DF2 = RDiag-DF4;
            DF3 = RDiag+DF4;
```

```
        for i=1:N+1
           for j=1:N+1
              A2(i,j) = DF2(i,j);
           end
        end

Temp = DF3*C' ;

        for i=1:N+1
           DV2(i) = Temp(i);
        end

        end
```

set_buffers

This code sets up the portion of the discretization matrix A due to the Calcium buffers.

```
function [A1,DV1] = set_buffers(X,A1,DV1,N,Parameter)

format long


==========================================================
Parameter (1) = rsize ;

Parameter (2) = rcore ;
Parameter (3) = delta_t ;
Parameter (4) = forward_binding_rate ;
Parameter (5) = backward_binding_rate ;
Parameter (6) = Diffusion Constant
Parameter (7) = B_iT ;

Binding Concentration below
level n

Parameter(8) = B_nT;

Binding Concentrationat
level n
```

```
Parameter(9) = C_i;

Ca Concentration below

level n

Parameter(10) = C_n;

Ca Concentration at

level n

=============================================================

rsize                  = Parameter(1);

rcore                  = Parameter(2);

delta_t                = Parameter(3);
forward_binding_rate    = Parameter(4);
backward_binding_rate   = Parameter(5);
Diffusion_Constant      = Parameter(6);
B_iT                   = Parameter(7);
B_nT                   = Parameter(8);
C_i                    = Parameter(9);
C_n                    = Parameter(10);


=============================================================

 X is size 2*no_shells + 2
 Ca+2 Concentration : size no_shells + 1
 C = X.slice (0,no_shells)
 C(1) = concentration at shell 0
 C(2) = concentration at shell 1
        .
        .
 C(no_shells+1) = concentration at shell no_shells
 Buffer Concentration : size no_shells + 1
 B = X.slice(no_shells + 1, 2*no_shells +1)
 B(1) = concentration at shell 0
 B(2) = concentration at shell 1
        .
        .
```

```
B(no_shells) = concentration at shell no_shells


u n i t s :

Now the rate of change of Ca and B wrt t is in
mM/( liter ms)


B is in mM/( liter ms)
U1 is

1 + ( forward_binding_rate*delta_t )*B
1+ (ms liter )/(mM ms ) ms mM/( liter ms)


U1 C is (mM)

( l i t e r ms)

U2 is backward_binding_rate*delta_t +( forward_binding_rate*delta_t)C

(1/ms ) ms + 1

U2 B is ( 1 + 1 ) (mM/( liter ms)
              mM/( l i t e r ms)

Same for U3 and U4

=====================================================


C      = zeros(1,N+1);
C      = X(1:N+1);

B      = zeros(1,N+1);
B      = X(N+2:(2*N)+2);

BT      = ones(N+1);
BT      = BT.*B_iT;
BT(N+1) = B_nT;

U1      = ones(1,N+1);
U1      = U1 + ((0.5*forward_binding_rate*delta_t).*B);
```

```
temp    = backward_binding_rate*delta_t*0.5;
U2      = ones(1,N+1);
U2      = U2.*temp;
temp    = 0.5*( forward_binding_rate*delta_t );
U2      = U2 + (temp.*C);

U3      = zeros(1,N+1);
temp    = 0.5*( forward_binding_rate*delta_t );
U3      = U3 + (temp.*B);

temp    = 1.0 + backward_binding_rate*delta_t*0.5;
U4      = ones(1,N+1);
U4      = U4 .*temp;
temp    = 0.5*(forward_binding_rate*delta_t);
U4      = U4 + (temp.*C);

============================================================
%       So A should be
%               U1 U2
%               U3 U4

============================================================
% set U1

for  i =1:N+1
   A1(i,i) = U1(i);
end

% set U2

for i=1:N+1
   A1(i,i+N+1) = U2(i);
end

% set U3
for i=1:N+1
   A1(i+N+1,i) = U3(i);
end

% set U4
for i=1:N+1
```

```
   A1(i+N+1,i+N+1) = U4(i);
end
=================================================================
%       Data Vector is DV1 of size 2*no_shells+1
%
%                   C ?( backward_binding_rate*delta_t*0.5)*B
%                   + backward_binding_rate*delta_t*BT
%       DV1= ------------------------------------------------------------
%               B ?( backward_binding_rate*delta_t*0.5)*B
%               + backward_binding_rate*delta_t*BT
%
===============================================================

DV1 = zeros(1,2*(N+1));

for i=1:N+1
   DV1(i) = C(i) - ( backward_binding_rate*delta_t*0.5)*B(i) +
backward_binding_rate*delta_t*BT(i);
end

for i=1:N+1
   DV1(i+N+1) = B(i) - ( backward_binding_rate*delta_t*0.5)*B(i)+
backward_binding_rate*delta_t*BT(i);
end

end
```

rest

```
function q  = rest (E_M,p,q )

g_NA_bar    = p(1);

g_K1_bar    = p(2);

g_K2_bar    = p(3) ;

g_K3_bar    = p(4) ;

g_CA1_bar   = p(5) ;

g_CA2_bar   = p(6);
```

```
g_CA3_bar   = p(7);

g_leak_bar  = p(8);

E_NA        = p(9);

E_L         = p(10);

C           = p(11);

K_I         = p(12);
K_O         = p(13);
E_K         = p(14);
CA_I        = p(15);
CA_O        = p(16);
E_CA        = p(17);

sum = E_M + 33.0;
if ( sum>0)
   alpha_mNA = 0.36*sum/(1.0 - exp(-sum/3.0) );

else
   alpha_mNA = 0.36*exp(sum/3.0)*sum/(exp(sum/3.0)- 1.0) ;
end

sum = E_M + 42.0;

if (sum>0)
   beta_mNA = (-0.4*exp(-sum/20.0)*sum)./(exp(-sum/20.0)-1.0) ;
else
   beta_mNA = (-0.4*sum) / (1.0 - exp(sum/20.0)) ;
end

m_NA_infinity = alpha_mNA /(alpha_mNA + beta_mNA ) ;

% inactivation parameter for I_NA

sum = E_M+55.0;
if ( sum<0)
   alpha_hNA = (-0.1*sum)/(1.0 - exp(sum/6.0));

else
   alpha_hNA = (-0.1*exp( -sum/ 6.0)*sum)/(exp( -sum/ 6.0) - 1.0);
end
```

```
if (E_M>0)
   beta_hNA = 4.5/(1.0+exp(-E_M/10.0)) ;
else
   beta_hNA = (4.5*exp(E_M/10.0)) / (exp(E_M/10.0) + 1.0);
end

h_NA_infinity = alpha_hNA/(alpha_hNA+beta_hNA);

I_NA = g_NA_bar*(E_M-E_NA)*m_NA_infinity*m_NA_infinity*h_NA_infinity
;

================================================================
Transient , Outward Potassium Current
================================================================
activation parameter for I_K

sum = E_M+42.0;

if ( sum<0)
   m_K1_infinity = exp(sum/13.0)/(exp(sum/13.0)+1.0);
else
   m_K1_infinity = 1.0/(1.0+exp(-sum/13.0) ) ;
end

% inactivation parameter for I_K

sum = E_M+110.0;

if ( sum<0)
   h_K1_infinity = 1.0/(1.0+exp(sum/18.0)) ;
else
   h_K1_infinity = exp(-sum/18.0)/(exp(-sum/18.0)+1.0);
end

I_K1 = g_K1_bar*(E_M-E_K)*m_K1_infinity*h_K1_infinity ;

================================================================
             Delayed, Rectifying Potassium Current
================================================================
activation parameter for I_K

sum = E_M+12.0;
```

```
if (sum<0)
   alpha_mK2 = (-0.0047*exp(sum/12.0)*sum)/(1.0 - exp(sum/12.0) ) ;
else
   alpha_mK2 = -0.0047*sum/(exp(-sum/12.0)-1.0) ;
end

sum = E_M+147.0;
if (sum>0)
   beta_mK2 = exp(-sum/30.0) ;
else
   beta_mK2 = exp(-sum/30.0 ) ;
end

t_m_K2 = 1.0/( alpha_mK2+beta_mK2 );

sum = (E_M-20.0)+12.0;

if (sum<0)
   alpha_mK2 = (-0.0047*exp(sum/12.0)*sum)/(1.0-exp(sum/12.0)) ;
else
   alpha_mK2 = (-0.0047*sum)/(exp(-sum/12.0) - 1.0);
end

sum = (E_M-20.0)+147.0;

if ( sum>0)
   beta_mK2 = exp(-sum/30.0);
else
   beta_mK2 = exp(-sum/30.0) ;
end

m_K2_infinity = alpha_mK2 /( alpha_mK2+beta_mK2 ) ;

% inactivation parameter for I_K

sum = E_M+25.0;

if (sum<0)
   h_K2_infinity = 1.0/(1.0+exp(sum/4.0) ) ;
else
   h_K2_infinity = exp(-sum/4.0 ) / (exp(-sum/4.0)+1.0) ;
end
```

```
if (E_M<-25.0)
   t_h_K2 = 6000.0 ;
else
   t_h_K2 = 50.0 ;
end

I_K2 = g_K2_bar*(E_M-E_K)*m_K2_infinity*m_K2_infinity*h_K2_infinity ;

==========================================================
        Non?inactivating Muscarinic Potassium Current
==========================================================

 activation parameter for I_K

sum     = (E_M+35.0) / 40.0 ;
t_m_K3  = 1000.0/(exp(sum) + exp(-2.0*sum));

sum     = (E_M+35.0) / 10.0;

if (sum>0)
   m_K3_infinity = exp(sum)/(exp(sum) + 1.0);
else
   m_K3_infinity = 1.0 / (1.0 + exp(-sum) ) ;
end

I_K3 = g_K3_bar*(E_M-E_K)*m_K3_infinity ;

==========================================================
        Fast Calcium Current
==========================================================

sum = (E_M+6.0) / 16.0 ;

if (sum>0)
   t_m_CA1 = (7.8*exp(-sum))/(1+exp(-2.0*sum) ) ;
else
   t_m_CA1 = (7.8*exp(sum))/(1+exp(2.0*sum) ) ;
end

if (E_M < -32.0)
   m_CA1_infinity = 0.0;
else
   sum = (E_M-3.0) / 8.0;
```

```
    if (sum>0)
        m_CA1_infinity = 1.0/(1.0+exp(-sum) ) ;
    else
        m_CA1_infinity = exp(sum) / (exp(sum)+1);
    end

end

h_CA1_0 = 0.01;

h_CA1 = h_CA1_0 /( h_CA1_0+CA_I ) ;

I_CA1 = g_CA1_bar*m_CA1_infinity*h_CA1*(E_M-E_CA);

=====================================================================
            Non?Inactivating Calcium Dependent Potassium Current
=====================================================================
sum = E_M/24.0 ;
s1 = 250.0*CA_I*exp(sum);
s2 = 0.1*exp(-sum);
t_m_CA2 = 1/( s1+s2 ) ;
m_CA2_infinity = s1 /( s1+s2 ) ;

sum = E_M/ 24.0 ;
t_m_CA2 = 1/(250.0*CA_I*exp(sum)+(0.1*exp(-sum)) ) ;

if (sum>0)
    m_CA2_infinity = (250.0*CA_I)/( 250.0*CA_I+0.1*exp(-2.0*sum) );
else
    m_CA2_infinity = (250.0*CA_I*exp(
2.0*sum))/(250.0*CA_I*exp(2.0*sum)+0.1);
end

I_CA2 = g_CA2_bar*m_CA2_infinity*(E_M-E_K);

==============v======================================================
                Voltage Independent Calcium Dependent Potassium Current
=====================================================================
s1 = ( 1.25e+8)*CA_I*CA_I;
t_m_CA3 = 1000.0/(s1 + 2.5);
m_CA3_infinity = s1 /(s1 +2.5) ;
```

```
I_CA3 = g_CA3_bar*m_CA3_infinity*m_CA3_infinity*(E_M - E_K) ;

sum = I_NA + I_K1 + I_K2 +I_K3 + I_CA1 + I_CA2 + I_CA3;

==============================================================
                    Find Initial Conditions
==============================================================

g_NA_inf    = g_NA_bar*m_NA_infinity*m_NA_infinity*h_NA_infinity;
g_K1_inf    = g_K1_bar*m_K1_infinity*h_K1_infinity;
g_K2_inf    = g_K2_bar*m_K2_infinity-h_K2_infinity ;
g_K3_inf    = g_K3_bar*m_K3_infinity;
g_CA1_inf   = g_CA1_bar*m_CA1_infinity*h_CA1;
g_CA2_inf   = g_CA2_bar*m_CA2_infinity;
g_CA3_inf   = g_CA3_bar*m_CA3_infinity*m_CA3_infinity;
g_total     = g_NA_inf + g_K1_inf + g_K2_inf + g_K3_inf + g_CA1_inf +
g_CA2_inf + g_CA3_inf + g_leak_bar;



fprintf('Initial activation and inactivations are:\n')
fprintf('m_Na(0)  = %f\n', m_NA_infinity)
fprintf('h_Na(0)  = %f\n',h_NA_infinity);
fprintf('m_K1(0)  = %f\n',m_K1_infinity);
fprintf('h_K1(0)  = %f\n',h_K1_infinity);
fprintf('m_K2(0)  = %f\n',m_K2_infinity);
fprintf('h_K2(0)  = %f\n',h_K2_infinity);
fprintf('m_K3(0)  = %f\n',m_K3_infinity);
fprintf('m_Ca1(0) = %f\n',m_CA1_infinity);
fprintf('m_Ca2(0) = %f\n',m_CA2_infinity);
fprintf('m_Ca3(0) = %f\n',m_CA3_infinity);

q(1)    = m_NA_infinity ;
q(2)    = h_NA_infinity ;
q(3)    = m_K1_infinity ;
q(4)    = h_K1_infinity ;
q(5)    = m_K2_infinity ;
q(6)    = h_K2_infinity ;
q(7)    = m_K3_infinity ;
q(8)    = m_CA1_infinity ;
q(9)    = m_CA2_infinity ;
q(10)   = m_CA3_infinity ;
q(11)   = K_O;
```

```
q(12)   = CA_I ;



end
```

nerst

```
function y = nerst(t,ion_in, ion_out,ii)

R     = 8.3144621;

z     = 1/ii;

T     = t+273;

F     = 96485.3365;

ratio = (ion_out./ion_in);

c     = (R.*T./F);

y     = z*58*log10(ratio);

end
```

m_inf_Na

```
function y = m_inf_Na(v)

 y = alpha_mNa(v) ./ ( alpha_mNa(v) + beta_mNa(v));

end
```

m_inf_K3

```
function y = m_inf_K3(v)
```

```
sum = v+35;

y   = 1 ./ ( 1.0 + exp(-sum/10) );


end
```

m_inf_K2

```
function y = m_inf_K2(v)

y   = alpha_mK2(v) ./ (alpha_mK2(v) + beta_mK2(v));

end
```

m_inf_K1

```
function y = m_inf_K1(v)

sum = v+42.0;
y   = ( 1 ) ./ (1.0 + exp(-sum./13.0));

end
```

m_inf_Ca

```
function y = m_inf_Ca(V)

sum = (V-3);
y   = 1/(1 + exp(-sum/8));

end
```

m_inf_C

```
function y = m_inf_C(V,Ca)
sum = V/24;
```

```
   b   = 0.1*exp(-sum);
   f   = 250*Ca*exp(sum);
   y   = f / (f + b);

         end
```

m_inf_AHP

```
function y = m_inf_AHP(Ca)

   f = 1.25e8*(Ca)*(Ca);
   b = 2.5;
   y = f /(f+b);

end
```

KOCH Model

This is the main KOCH model that calculates the action potential, this function uses all the function that are defined before.

```
function V =  koch(dt,b)

format('long')

%dt        = 1e-3;              % Number of Steps
a         = 0;                  % Staring Point Of The Time
% b         = 30;               % Ending Point Of The Time
n         = ((b-a)./ (dt)) +1;  % Number Of Loops
tol       = 7.0e-1;             % Convergence Criteria
no_shells = 10;                 % Number Of Shells



tem     = 22.0;
V0      = -70;
V       = zeros(n,1);
I_Na    = zeros(n,1);
I_A     = zeros(n,1);
I_K     = zeros(n,1);
I_M     = zeros(n,1);
I_Ca    = zeros(n,1);
```

```
I_C     = zeros(n,1);
I_AHP   = zeros(n,1);
I_syn   = zeros(n,1);
%K_in   = zeros(1,n);
Ca_in   = zeros(n,1);
I_L     = zeros(n,1);
mNa     = zeros(n,1);
hNa     = zeros(n,1);
mK1     = zeros(n,1);
hK1     = zeros(n,1);
mK2     = zeros(n,1);
hK2     = zeros(n,1);
mK3     = zeros(n,1);
mCa     = zeros(n,1);
hCa     = zeros(n,1);
mC      = zeros(n,1);
mAHP    = zeros(n+0,1);
gNa     = zeros(n,1);
E_K     = zeros(n,1);
E_Ca    = zeros(n,1);
gK1     = zeros(n,1);
time    = zeros(n,1);
K_out   = zeros(n,1);
EX      = zeros(n,1);
I_K     = zeros(n,1);
p       = zeros(17,1);
q       = zeros(12,1);
B       = zeros(no_shells+1,1);
Ca      = zeros(no_shells+1,1);

g_Na_bar   = 2.0;
g_A_bar    = 1.2;
g_K_bar    = 1.17;
g_M_bar    = 0.084;
g_Ca_bar   = 0.116;
g_C_bar    = 1.20;
g_AHP_bar  = .054;
g_leak_bar = 0.02;

G_L         = g_leak_bar;
C           = 0.15; % Capacitance

K_out(1)    = 5;
```

```
K_in        = 140.0;
Ca_out      = 5.0;
Ca_in(1)    = 0.00005;

E_Na    = nerst(tem,10,145,1);
E_K(1)  = nerst(tem,K_in,K_out(1),1);
E_L     = -10;
E_Ca(1) = nerst(tem,Ca_in(1),Ca_out,2);
E_m     = V0;
E_syn   = -10;

p(1)    =      g_Na_bar;
p(2)    =      g_A_bar;
p(3)    =      g_K_bar;
p(4)    =      g_M_bar;
p(5)    =      g_Ca_bar;
p(6)    =      g_C_bar;
p(7)    =      g_AHP_bar;
p(8)    =      g_leak_bar;
p(9)    =      E_Na;
p(10)   =      E_L ;
p(11)   =      tem;
p(12)   =      K_in;
p(13)   =      K_out(1);
p(14)   =      E_K(1);
p(15)   =      Ca_in(1);
p(16)   =      Ca_out;
p(17)   =      E_Ca(1);

q = rest( E_m,p,q);

V0 = E_m;
V(1)        = V0;
mNa(1)      = q(1);
hNa(1)      = q(2);
mK1(1)      = q(3);
hK1(1)      = q(4);
mK2(1)      = q(5);
hK2(1)      = q(6);
mK3(1)      = q(7);
mCa(1)      = q(8);
mC (1)      = q(9);
mAHP(1)     = q(10);
```

```
K_out(1)    = q(11);
Ca_in(1)    = q(12);

time    = a:dt:b;


I_Na(1)  = (V(1) - E_Na).*(g_Na_bar.*(mNa(1)).*(mNa(1)).*(hNa(1)) );
I_A(1)  = (V(1) - E_K(1)).*(g_A_bar.*(mK1(1)).*(hK1(1)) );
I_K(1)  = (V(1) - E_K(1)).*(g_K_bar.*(mK2(1)).*(mK2(1)).*(hK2(1)) );
I_M(1)  = (V(1) - E_K(1)).*(g_M_bar.*(mK3(1)));
I_Ca(1)  = (V(1) - E_Ca(1)).*(g_Ca_bar.*(mCa(1)).*(hCa(1)) );
I_C (1)  = (V(1) - E_K(1)).*(g_C_bar.*(mC (1)));
I_AHP(1) = (V(1) - E_K(1)).*(g_AHP_bar).*(mAHP(1)).*(mAHP(1));
I_L(1)   = (V(1) - E_L).*g_leak_bar;


Ca(1:(no_shells+1)) = 50e-6;
B(1:no_shells,1)    = 3e-3;
B((no_shells+1),1)  = 30e-3;

start = 0 % Start of the
i = 1;


while time(i) < b

   i;


=================================================================
                 Calculation Of Potentials
=================================================================

   E_L     = -10;
   E_Ca(i) =nerst(tem,Ca_in(i),Ca_out,2);


=================================================================

           Calculation of the Differential Equations

=================================================================
   % dV(t.dt)/dt
   dmNa  = ((m_inf_Na(V(i))-mNa(i))./tau_mNa(V(i)));  % Activation Fast
Sodium Current
   dhNa  = ((h_inf_Na(V(i))-hNa(i))./tau_hNa(V(i)));  % Inactivation Fast Sodium
Current
   dmK1  = ((m_inf_K1(V(i))-mK1(i))./tau_mK1(V(i)));  % Activation Transient,
Outwarad Potassium Current
   dhK1  = ((h_inf_K1(V(i))-hK1(i))./tau_hK1(V(i)));  % Inactivation Transient,
Outward Potassium Current
   dmK2  = ((m_inf_K2(V(i))-mK2(i))./tau_mK2(V(i)));  % Delayed, Rectifying
Potassium Current
   dhK2  = ((h_inf_K2(V(i))-hK2(i))./tau_hK2(V(i)));  % Delayed, Rectifying
Potassium Current
   dmK3  = ((m_inf_K3(V(i))-mK3(i))./tau_mK3(V(i)));  % Non-Inactivating
Muscarinic Potassium Current
   dmCa  = ((m_inf_Ca(V(i))-mCa(i))./tau_mCa(V(i)));  % Fast Calcium Current
   dmC   = ((m_inf_C(V(i),Ca_in(i))-mC (i))./tau_mC (V(i),Ca_in(i)) );  % Non-
inactivating Calcium-Dependent Potassium Current
   dmAHP = ((m_inf_AHP(Ca_in(i))-mAHP(i))./tau_mAHP(Ca_in(i)) ); %
Voltage-Independent, Calcium-Dependent Potassium Current

=============================================================
              Activation Fast Sodium Current
```

```
   pre       = mNa(i)+dmNa.*dt;
   dmNa_p_dt = ( (m_inf_Na(V(i))-pre)./tau_mNa(V(i)) );
   corr      = mNa(i)+(dt/2).*(dmNa_p_dt + dmNa);
   diff      = abs( corr-pre );

   while diff > tol

      dmNa      = ( (m_inf_Na(V(i))-corr)./tau_mNa(V(i)) );
      pre       =  corr+dmNa.*dt;
      dmNa_p_dt = ( (m_inf_Na(V(i))-pre)./tau_mNa(V(i)) );
      corr      = mNa(i)+(dt/2).*(dmNa_p_dt+dmNa);
      diff      = abs(corr-pre );



   end

mNa(i+1)  = corr;
```

```
=============================================================

                Inactivation Fast Sodium Current

=============================================================



  pre      = hNa(i)+dhNa.*dt;
  dhNa_p_dt = ( (h_inf_Na(V(i))-pre)./tau_hNa(V(i)) );
  corr     = hNa(i)+(dt/2).*(dhNa_p_dt+dhNa);
  diff     = abs(corr-pre);

  while diff > tol

     dhNa      = ( (h_inf_Na(V(i))-corr)./tau_hNa(V(i)) );
     pre       =  corr+dhNa.*dt;
     dhNa_p_dt = ( (h_inf_Na(V(i))-pre)./tau_hNa(V(i)) );
     corr      = hNa(i)+(dt/2).*(dhNa_p_dt+dhNa);
     diff      = abs( corr-pre );

  end

  hNa(i+1)  = corr;


=============================================================

          Activation Transient, Outwarad Potassium Current

=======================================================



  pre      = mK1(i)+dmK1.*dt;
  dmK1_p_dt = ( (m_inf_K1(V(i))-pre)./tau_mK1(V(i)) );
  corr     = mK1(i)+(dt/2).*(dmK1_p_dt+dmK1);
  diff     = abs( corr-pre );

  while diff > tol

     dmK1       = ( (m_inf_K1(V(i))-corr)./tau_mK1(V(i)) );
     pre        = corr+dmK1.*dt;
```

```
        dmK1_p_dt   = ( (m_inf_K1(V(i))-pre)./tau_mK1(V(i)) );
        corr        = mK1(i)+(dt/2).*(dmK1_p_dt+dmK1);
        diff        = abs( corr-pre );

    end

    mK1(i+1) =  corr;
```

==

Inactivation Transient, Outward Potassium Current

==

```
    pre       = hK1(i)+dhK1.*dt;
    dhK1_p_dt = ( (h_inf_K1(V(i))-pre)./tau_hK1(V(i)) );
    corr      = hK1(i)+((dt*2).*(dhK1_p_dt+dhK1));
    diff      = abs( corr - pre );

    while diff > tol

        dmK1        = ( (h_inf_K1(V(i))-corr)./tau_hK1(V(i)) );
        pre         = corr + dhK1.*dt;
        dhK1_p_dt   = ( (h_inf_K1(V(i)) - pre)./tau_hK1(V(i)) );
        corr        = hK1(i) + ((dt*2).*(dhK1_p_dt + dhK1));
        diff        = abs( corr - pre );
    end

    hK1(i+1)  = corr;
```

==

Delayed, Rectifying Potassium Current

===

```
    pre         = mK2(i) + dmK2.*dt;
    dmK2_p_dt   = ( (m_inf_K2(V(i)) - pre) ./ tau_mK2(V(i)) );
    corr        = mK2(i) + (dt/2).*(dmK2_p_dt + dmK2);
```

```
   diff        = abs( corr - pre );

   while diff > tol

      dmk2        = ( (m_inf_K2(V(i)) - corr) ./ tau_mK2(V(i)) );
      pre         = corr + dmK2.dt;
      dmK2_p_dt   = ( (m_inf_K2(V(i)) - pre) ./ tau_mK2(V(i)) );
      corr        = mK2(i) + (dt/2).*(dmK2_p_dt + dmK2);
      diff        = abs( corr - pre );
   end

   mK2(i+1) = corr;


==============================================================

            Delayed, Rectifying Potassium Current

==============================================================

   pre         = hK2(i) + dhK2.*dt;
   dhK2_p_dt   = ( (h_inf_K2(V(i)) - pre) ./ tau_hK2(V(i)) );
   corr        = hK2(i) + (dt/2).*(dhK2_p_dt + dhK2);
   diff        = abs( corr - pre );

   while diff > tol

      dhk2        = ( (h_inf_K2(V(i)) - corr) ./ tau_hK2(V(i)) );
      pre         = corr + dhK2.dt;
      dhK2_p_dt   = ( (h_inf_K2(V(i)) - pre) ./ tau_hK2(V(i)) );
      corr        = hK2(i) + (dt/2).*(dhK2_p_dt + dhK2);
      diff        = abs( corr - pre );
   end

   hK2(i+1) = corr;



==============================================================

                Non-Inactivating Muscarinic Potassium Current

==============================================================
   pre         = mK3(i) + dmK3 .* dt;
   dmK3_p_dt   = ( (m_inf_K3(V(i)) - pre) ./ tau_mK3(V(i)) );
   corr        = mK3(i) + (dt/2).*(dmK3_p_dt + dmK3);
   diff        = abs( corr - pre);

   while diff > tol

      dmK3        = ((m_inf_K3(V(i))-corr)./tau_mK3(V(i)));
      pre         = corr+dmK3.*dt;
      dhK3_p_dt   = ((m_inf_K3(V(i))-pre)./tau_mK3(V(i)));
      corr        = mK3(i)+(dt/2).*(dmK3_p_dt+dhK3);
      diff        = abs(corr-pre);
   end

   mK3(i+1) = corr;


%==================================================

%                  Fast Calcium Current

%==================================================

   pre         = mCa(i)+dmCa.*dt;
   dmCa_p_dt   = ( (m_inf_Ca(V(i))-pre)./tau_mCa(V(i)));
   corr        = mCa(i)+(dt/2).*(dmCa_p_dt+dmCa);
   diff        = abs( corr-pre);

   while diff > tol

      dmCa        = ((m_inf_Ca(V(i))-corr)./tau_mCa(V(i)));
      pre         = corr+dmCa.*dt;
      dmCa_p_dt   = ((m_inf_Ca(V(i))-pre)./tau_mCa(V(i)));
      corr        = mCa(i)+(dt/2).*(dmCa_p_dt+dmCa);
      diff        = abs( corr-pre );

   end

   mCa(i+1) = corr;


==================================================


               Fast Calcium Current
```

```
==================================================================

        hCa(i+1) = 0.01/(0.01+Ca_in(i));

==================================================================

     Non-inactivating Calcium-Dependent Potassium Current

==================================================================

  pre         = mC(i)+dmC.*dt;
  dmC_p_dt    = ( (m_inf_C(V(i),Ca_in(i))-pre)./tau_mC(V(i),Ca_in(i)) );
  corr        = mC(i)+(dt/2).*(dmC_p_dt+dmC);
  diff        = abs(corr-pre);

  while diff > tol

    dmC         = ( (m_inf_C(V(i),Ca_in(i))-corr)./tau_mC(V(i),Ca_in(i)) );
    pre         = corr+dmC.*dt;
    dmC_p_dt    = ( (m_inf_C(V(i),Ca_in(i))-pre)./tau_mC(V(i),Ca_in(i)) );
    corr        = mC(i)+(dt/2).*(dmC_p_dt+dmC);
    diff        = abs( corr-pre );

  end

  mC(i+1) = corr;

     Voltage-Independent, Calcium-Dependent Potassium Current
====================================================================

  pre         = mAHP(i)+dmAHP.*dt;
  dmAHP_p_dt  = ( (m_inf_AHP(Ca_in(i))-pre)./tau_mAHP(Ca_in(i)) );
  corr        = mAHP(i)+(dt/2).*(dmAHP_p_dt+dmAHP);
  diff        = abs( corr-pre);

  mAHP(i+1) = pre;

==================================================================
              Potassium Accumulation
==================================================================
  V_peri    = 11.749137;
  t_K_diff  = 7.0;
  K1 = I_A(i);
  K2 = I_K(i);
  K3 = I_M(i);
  IC = I_C(i);
  IK_Total  = (I_A(i)+I_K(i)+I_M(i)+I_C(i)+I_AHP(i));
  F         = 9.649e4;
  K_rest    = 2.5;
  lead      = (IK_Total/(V_peri*F));
  other     = ((K_out(i)-K_rest)./t_K_diff);
  dK_out_dt = lead - other;

  pre       = K_out(i)+dK_out_dt.*dt;
  K_out(i+1) = pre;

    K_out(i+1) = 7.8;

  E_K(i)  = nerst(tem,K_in,K_out(i),1);
  K = E_K(i);
```

Fast, Nicotinic Synaptic Input

```
=======================================================

   t_peak = 2.5;
   g_sym  = 2.90856e-5;
   u      = (time(i)./t_peak);
           g_syn  = time(i)*g_sym*exp(-u);


=======================================================
                  Current Calculation
=======================================================

   I_Na(i)  = (V(i)-E_Na).*(g_Na_bar.*(mNa(i)).*(mNa(i)).*(hNa(i)) );
   I_A(i)   = (V(i)-E_K(i)).*(g_A_bar.*(mK1(i)).*(hK1(i)) );
   I_K(i)   = (V(i)-E_K(i)).*(g_K_bar.*(mK2(i)).*(mK2(i)).*(hK2(i)) );
   I_M(i)   = (V(i)-E_K(i)).*(g_M_bar.*(mK3(i)));
   I_Ca(i)  = (V(i)-E_Ca(i)).*(g_Ca_bar.*(mCa(i)).*(hCa(i)) );
   I_C (i)  = (V(i)-E_K(i)).*(g_C_bar.*(mC (i)));
```

```
    I_AHP(i) = (V(i)-E_K(i)).*(g_AHP_bar).*(mAHP(i)).*(mAHP(i));
    I_L(i)   = (V(i)-E_L).*g_leak_bar;
    I_syn(i) = (V(i)-E_syn).*g_syn;

================================================================
                         Voltage Calculation
================================================================

    dV           = (1./C).*( I_ext(time(i))-(
I_Na(i)+I_A(i)+I_K(i)+I_M(i)+I_Ca(i)+I_C(i)+I_AHP(i)+I_L(i)+I_syn(i) ) );
    pre          = V(i)+dV.*dt;
    I_Na(i)  = (pre-E_Na).*(g_Na_bar.*(mNa(i)).*(mNa(i)).*(hNa(i)) );
    I_A(i)   = (pre-E_K(i)).*(g_A_bar.*(mK1(i)).*(hK1(i)) );
    I_K(i)   = (pre-E_K(i)).*(g_K_bar.*(mK2(i)).*(mK2(i)).*(hK2(i)) );
    I_M(i)   = (pre-E_K(i)).*(g_M_bar.*(mK3(i)));
    I_Ca(i)  = (pre-E_Ca(i)).*(g_Ca_bar.*(mCa(i)).*(hCa(i)) );
    I_C (i)  = (pre-E_K(i)).*(g_C_bar.*(mC (i)));
    I_AHP(i) = (pre-E_K(i)).*(g_AHP_bar).*(mAHP(i)).*(mAHP(i));
    I_L(i)   = (V(i) - E_L).*g_leak_bar;
    I_syn(i) = (V(i)-E_syn).*g_syn;
    dV_p_dt    = (1./C).*( I_ext(time(i))-(
I_Na(i)+I_A(i)+I_K(i)+I_M(i)+I_Ca(i)+I_C(i)+I_AHP(i)+I_L(i)+I_syn(i) ) );
    corr       = V(i) + (dt/2).*(dV_p_dt + dV);
    diff       = abs( corr - pre );
    EX(i)      = I_ext(time(i));
    I_K(i)   = I_A(i) + I_K(i) + I_M(i);

    while diff > tol

        I_Na(i)  = (corr-E_Na).* (g_Na_bar.*(mNa(i)).*(mNa(i)).*(hNa(i)) );
        I_A(i)   = (corr-E_K(i)).*(g_A_bar.*(mK1(i)).*(hK1(i)) );
        I_K(i)   = (corr-E_K(i)).*(g_K_bar.*(mK2(i)).*(mK2(i)).*(hK2(i)) );
        I_M(i)   = (corr-E_K(i)).*(g_M_bar.*(mK3(i)));
        I_Ca(i)  = (corr-E_Ca(i)).*(g_Ca_bar.*(mCa(i)).*(hCa(i)) );
        I_C (i)  = (corr-E_K(i)).*(g_C_bar.*(mC (i)));
        I_AHP(i) = (corr-E_K(i)).*(g_AHP_bar).*(mAHP(i)).*(mAHP(i));
        I_L(i)   = (corr-E_L).*g_leak_bar;
        dV       = (1./C).*( I_ext(time(i))-(
I_Na(i)+I_A(i)+I_K(i)+I_M(i)+I_Ca(i)+I_C(i)+I_AHP(i)+I_L(i)+I_syn(i) ) );
        pre      = corr + dV;
        I_Na(i)  = (pre-E_Na).*(g_Na_bar.*(mNa(i)).*(mNa(i)).*(hNa(i)) );
        I_A(i)   = (pre-E_K(i)).*(g_A_bar.*(mK1(i)).*(hK1(i)) );
        I_K(i)   = (pre-E_K(i)).*(g_K_bar.*(mK2(i)).*(mK2(i)).*(hK2(i)) );
        I_M(i)   = (pre-E_K(i)).*(g_M_bar.*(mK3(i)));
        I_Ca(i)  = (pre-E_Ca(i)).*(g_Ca_bar.*(mCa(i)).*(hCa(i)) );
        I_C (i)  = (pre-E_K(i)).*(g_C_bar.*(mC (i)));
        I_AHP(i) = (pre-E_K(i)).*(g_AHP_bar).*(mAHP(i)).*(mAHP(i));
        I_L(i)   = (pre-E_L).* g_leak_bar;
        dV_p_dt  = (1./C).*( I_ext(time(i))-(
I_Na(i)+I_A(i)+I_K(i)+I_M(i)+I_Ca(i)+I_C(i)+I_AHP(i)+I_L(i)+I_syn(i) ) );
        corr     = V(i)+(dt/2).*(dV_p_dt + dV);
        diff     = abs( corr-pre );
        I_K(i)   = I_A(i)+I_K(i)+I_M(i);

    end

    V(i+1)    = corr;
%    [V(i+1),mNa(i+1),hNa(i+1),mK1(i+1),hK1(i+1)] =
voltage(V(i),I_ext(time(i)),dt,mNa(i),hNa(i),mK1(i),hK1(i));

==========================================================
                        Calcium Update
==========================================================
tem = V(i);
 start = 0;

%  Ca =  Ca_update(V(i),dt,Ca,B,I_Ca(i));
%  Ca_in(i+1) =  Ca(11);

 if tem>=-25
    Ca =  Ca_update(V(i),dt,Ca,B,I_Ca(i));
    Ca_in(i+1) =  Ca(11);
 %   start = 1;
 else
   Ca_in(i+1) = 0.00005;
   % K_out(i+1) = 1.8;
 end

   %Ca_in(i+1) =  0.00005;
   i = i+1;

end

%plot(time,V)
end
```

Alpha_hNa

```
function y = alpha_hNa(v)

    sum = v+55.0;
    y   = ( -0.10.*sum ) ./ (1.0 - exp(sum./6.0));

end
```

I_ext

```
function y = I_ext(t)

 if t>00 && t<02

        y = 5;

 else

 y = 0;

end

end
```

h_inf_Na

```
function y = h_inf_Na(v)

y = alpha_hNa(v) ./ ( alpha_hNa(v) + beta_hNa(v));

end
```

h_inf_K2

```
function y = h_inf_K2(v)

sum = v+25;

y   = 1 ./ (exp(sum./4));

end
```

h_inf_K1

```
function y = h_inf_K1(v)

sum = v+110.0;

y   = ( 1 ) ./ (1.0 + exp(sum./18.0));

end
```

getCA_at_N

```
function [X] = getCA_at_N(X,no_shells,m_CA1,V,Parameter)

format long

=========================================================

     Concentrations here are in mu M
     Parameter (1) = rsize;
     Parameter (2) = rcore;
     Parameter (3) = delta_t;
     Parameter (4) = forward binding rate;
     Parameter (5) = backward binding rate;
     Parameter (6) = Diffusion Constant;
     Parameter (7) = B_iT;

     Binding Concent ration below
     level no_shells

     Parameter (8) = B_nT;
```

```
          Binding Concentration at
          level no_shells

          Parameter (9) = C_i;

          Ca Concentration below
          level no_shells

          Parameter (10) = C_n;

          Ca Concentration at
          level n
=========================================================


N                     = no_shells;
rsize                 = Parameter(1);
rcore                 = Parameter(2);
delta_t               = Parameter(3);
forward_binding_rate    = Parameter(4);
backward_binding_rate   = Parameter(5);
Diffusion_Constant      = Parameter(6);
B_iT                  = Parameter(7);
B_nT                  = Parameter(8);
C_i                   = Parameter(9);
C_n                   = Parameter(10);

A1      = zeros((2*N) +2,(2*N)+2);
DV1     = zeros((2*N)+2,1);

A2      = zeros((2*N)+2,(2*N)+2);
DV2     = zeros((2*N)+2,1);

A3      = zeros((2*N)+2,(2*N)+2);
DV3     = zeros((2*N)+2,1);

A4      = zeros((2*N)+2,(2*N)+2);
DV4     = zeros((2*N)+2,1);

A       = zeros((2*N)+2,(2*N) +2);
Acopy   = zeros((2*N)+2,(2*N) +2);

DV      = zeros((2*N)+2,1);
```

```
%       construct buffer discretization matrices

[A1,DV1] = set_buffers(X,A1,DV1,N,Parameter);
A = A1;
DV = DV1';

%       construct diffusion discretization matrices

[A2,DV2] = set_diffusion (X,A2,DV2,N,Parameter);
A   = A + A2;
DV  = DV + DV2;

%       construct ICA current matrices

[A3,DV3] = set_ICA (X,A3,DV3,N,m_CA1,V,Parameter);
A   = A + A3;
DV  = DV + DV3;

%       construct Ca pump matrices

[A4, DV4] = set_Pump (X,A4,DV4,N,V,Parameter);
A   = A + A4;
DV  = DV + DV4;



X = (A\DV)';


end
```

Alpha_mK2

```
function y = alpha_mK2(v)

   sum = v+12;
   y   = (-0.0047.*sum) ./ (-1 + exp(-sum./12.0));

end
```

Ca_update

This function calculates the Calcium concentrations in the neuron after the action potential has fired.

```
function x = Ca_update(V,dt,Ca,B,I_Ca)

% Ca is Calcium Concentration Layer wise
% B is the buffer Concentration

A  = zeros(11,11);
DV = zeros(11,1);

F  = 96490;
D  = 6.0e-3;          %cm2sec-1
n  = 10;
f  = 1.0e2;
b  = 1e-1;


r_core   = 19;    % um micrometer
r_i      = zeros(11,1);
delta_r  = 0.1;      % um micrometer
B_it     = zeros(11,1);
Ca_equil = 50e-3;
Ca_n     = Ca(n+1,1);
Km       = 0.02;
tau_pump = 17.7*exp(V*1e-3/35);
K_rest   = 2.5;

% Setting up the radius of the Shells

for i=0:10

    r_i(i+1) = r_core + (i*delta_r);

end

r_out   = power(r_i(n+1),3);
```

```
r_in    = power(r_i(n),3);
r_diff  = (r_out-r_in);
V_n     = (4*pi/3)*r_diff ;

% Setting up the [B]i,total Concentrations

B_it(n+1) = 30e-0;
for i=1:n
   B_it(i)= 3.0e-0;
end


A(1,1) = ( 1+((3*D*dt)/(2*r_core*r_core))+(f*dt*B(1)*0.5) );
A(1,1) = A(1,1)-( (f*dt*dt*B(1)*(b+(f*Ca(1))))/(2*(2+(b*dt)+(f*dt*Ca(1)))) );

A(1,2) = -(3*D*dt)/(2*r_core*r_core);

for i=2:n

   A(i,i-1)= ( (-D*dt*(r_i(i)-delta_r))/(2*r_i(i)*delta_r*delta_r) );

   A(i,i)  = 1+( (D*dt)/(delta_r*delta_r) )+( f*dt*B(i)*0.5 );
   A(i,i)  = A(i,i)-( (dt*dt*f*B(i)*(b+(f*Ca(i))))/(2*(2+(b*dt)+(f*dt*Ca(i)))) );

   A(i,i+1)= ( (-D*dt*(r_i(i)+delta_r))/(2*r_i(i)*delta_r*delta_r) );

end

A(11,10)   = -( (D*dt*(r_i(11)-delta_r))/(2*r_i(11)*delta_r*delta_r) );

A(11,11) = 1+( (D*dt*(r_i(11)-delta_r)) / (2*r_i(11)*delta_r*delta_r)
)+(f*dt*B(11)*0.5)+(dt/(2*tau_pump));
A(11,11) = A(n+1,n+1)-(
(dt*dt*f*B(11)*(b+(f*Ca(11))))/(2*(2+(b*dt)+(f*dt*Ca(11)))) );


DV(1,1) = (Ca(1)*(1+((3*D*dt)/(2*r_core*r_core)))) +
((3*D*dt*Ca(2))/(2*r_core*r_core));
DV(1,1) = DV(1,1)+(b*dt*B_it(1))-(b*dt*0.5);
DV(1,1) = DV(1,1)-(
(2*b*dt*dt*B_it(1)*(b+(f*Ca(1))))/(2*(2+(b*dt)+(f*dt*Ca(1)))) );
DV(1,1) = DV(1,1)-( (B(1)*((2*b*dt)+(dt*dt*b*b)+(dt*f*Ca(1))-
(dt*dt*f*b*Ca(1))))/(2*(2+(b*dt)+(f*dt*Ca(1)))) );
```

```
for i=2:n

   DV(i,1) = ( (D*dt*(r_i(i)+delta_r)*Ca(i+1))/(2*r_i(i)*delta_r*delta_r) )+(
Ca(i)*(1+((D*dt)/(delta_r*delta_r))) );
   DV(i,1) = DV(i,1)+( (D*dt*Ca(i-1)*(r_i(i)-delta_r))/(2*r_i(i)*delta_r*delta_r)
)+(2*dt*B_it(i))-(0.5*b*dt*B(i));
   DV(i,1) = DV(i,1)-( (dt*dt*B_it(i)*(b+(f*Ca(i))))/(2+(b*dt)+(f*dt*Ca(i))) );
   DV(i,1) = DV(i,1)-( (dt*B(i)*((2*b)-(b*b*dt)+(Ca(i)*((2*f)-
(f*b*dt)))))/(2*(2+(b*dt)+(f*dt*Ca(i)))) );

end

DV(11,1) = ( (D*dt*(r_i(11)-delta_r)*Ca(10))/(2*r_i(11)*delta_r*delta_r) )-(
(D*dt*Ca(11)*(r_i(11)-delta_r))/(2*r_i(11)*delta_r*delta_r) );
DV(11,1) = DV(11,1)-( (dt*I_Ca)/(2*F*V_n) )+(b*dt*B_it(11))-(b*dt*B(11))+(
(dt*Ca_equil)/(tau_pump) )-( (dt*Ca(11))/(2*tau_pump) );
DV(11,1) = DV(11,1)-( (dt*dt*B_it(11)*(b+(f*Ca(11))))/(2+(b*dt)+(f*dt*Ca(11)))
);
DV(11,1) = DV(11,1)-( (dt*B(11)*((2*b)-(b*b*dt)+(Ca(11)*((2*f)-
(f*b*dt)))))/(2*(2+(b*dt)+(f*dt*Ca(11)))) );


a = zeros(11,1);
b = zeros(11,1);
c = zeros(11,1);
d = DV;

for i=2:(n+1)

   a(i,1)= A(i,(i-1));
   c(i,1)= A((i-1),i);

end

for i=1:(n+1)

   b(i,1)= A(i,i);


end
```

```
a, b, c are the column vectors for the compressed tridiagonal matrix, d is the right
vector
n = length(b); % n is the number of rows

% Modify the first-row coefficients
c(1) = c(1) / b(1);    % Division by zero risk.
d(1) = d(1) / b(1);    % Division by zero would imply a singular matrix.

for i = 2:n-1
    temp = b(i) - a(i) * c(i-1);
    c(i) = c(i) / temp;
    d(i) = (d(i) - a(i) * d(i-1))/temp;
end

d(n) = (d(n)-(a(n)*d(n-1)))/( b(n)-a(n)*c(n-1));

% Now back substitute.
x(n) = d(n);
for i = n-1:-1:1
    x(i) = d(i) - c(i) * x(i + 1);
end

x = x';
for i=1:11
    if x(i)<0
        x(i)=0;
    end
end
end
```

beta_mNa

```
function y = beta_mNa(v)

    sum = v+42.0;
    y   = ( -0.40.*sum ) ./ (1.0 - exp(sum./20.0));

end
```

beta_mK2

```
function y = beta_mK2(v)

    sum = v+147.0;
    y   = exp(-sum./30.0);

end
```

beta_hNa

```
function y = beta_hNa(v)

    sum = v+0.0;
    y   = ( 4.5 ) ./ (1.0 + exp(-sum./10.0));

end
```

alpha_mNa

```
function y = alpha_mNa(v)

    sum = v+33.0;
    y   = ( 0.36.*sum ) ./ (1.0 - exp(-sum./3.0));

end
```