PHY3907 Work Term Report

Computational Neuroscience and Fiber Loss Simulation at uOttawa

Mohammed A. Chamma - 6379153

University of Ottawa

September 12, 2014

Contents

Abstract

In this report I describe my time at the uOttawa Physics Department doing computational neuroscience under Professor Bela Joós during the summer of 2014. The purpose of my research project was to model the measured voltage signal of a bundle of nerves in an arm from an experiment where nerves in the elbow are stimulated and then the propagated signal is measured further down the nerves, around the wrist. In damaged nerves, the signal does not propagate properly down the arm, and can be a clear signal of some diseases. I describe different models of neurons used in developping our own model of the arm nerves and I describe a form of damage nerves can experience called "coupled left-shift sodium channels", modelled by Professor Bela's group. Finally, I describe my completed simulation model written in the Python programming language and give a brief overview of its use and some sample results.

Introduction

I spent my summer coop term at the University of Ottawa studying neuroscience under Professor Bela Joos. The term lasted four months, starting in May and ending in August. It was also my first time living on my own; I moved out of my home in Kanata and rented a room in a neighbourhood very close to the University with the goal of shaving off almost 2 hours spent commuting every day. I worked on the third-floor of Macdonald, the Physics building, in an office with two other students also doing research with Professor Bela. The environment was very relaxed and academic. I worked on my own laptop.

During my time there I learned the basics of neuroscience. Professor Bela and I talked about a project to model an experiment where nerve damage in an arm is measured by electrically stimulating one area of the arm (like the elbow) and then measuring the propagated signal through the nerves at another point (like the wrist). In arms with damaged nerves, this signal can arrive more slowly or more weakly than in arms with healthy nerves. Therefore, being able to detect and characterize signals that aren't propagating properly can lead to more accurate diagnoses.

Before I could model a bundle of damaged nerves in an arm I had to learn and understand how single neurons behaved. This meant learning the basic physiology of a neuron: about axons, myelin sheaths, and axon nodes. After that, I learned about the Hodgkin-Huxley model of an axon node: a mathematical model developped in the 1950s used to model voltage clamp experiments with a giant squid axon. After I could successfully model a single node, we found a paper by Donald Mcneal written in 1976 that presented a model of axon nodes connected to each other. With this model we could simulate signals propagating from node to node, and successfully simulated results that indicated propagation. After that, we added a model of damage to each node of the axon. The model for this damage was developped by Professor Bela's group. With all these pieces we were able to represent a bundle of nerve fibers as found in an arm; we could control the length of the nerve, the size of each nerve and the amount of damage in each nerve. This model lets us simulate the experiment with different kinds of arms, healthy and unhealthy ones.

This report will give an overview of the concepts necessary for the building of the model and a look at the simulation code and how it can be used to produce data on the arm signal experiment with a wide variety of parameters. In addition some results from the model will be shown and direction for the future will be explored.

Neurons

At the heart of the project is the behaviour of neurons. Neurons are cells capable of transmitting electrical signals. In human bodies they form the structure of our brains and control the movement of our limbs. In 1952 A. L. Hodgkin and A. F. Huxley measured spikes of potential in the cellular membrane of a giant squid axon. These spikes are referred to as action potentials, and are signals that can propagate down the length of a neuron and stimulate an adjacent neuron, thus transmitting the signal even further. Hodgkin and Huxley's innovation was in creating a mathematical model that attempted to explain the electrical activity of the neuron.

The neuron is composed of the cell body, dendrites, synapses, and axons. It is the axons that are critical in understanding the Hodgkin-Huxley model. Axons are long tendrils that jut out from the cell body; they are long, thin extensions of the cell that usually reach out to other neurons. In mature neurons the arm of the axon is protected by a myelin sheath. This sheath does not cover the entire axon, since it is the membrane of the axon and not the myelin sheath that allows ions to flow through the membrane. Instead, the sheath covers the axon in a way that leaves periodic gaps of axon membrane exposed. These axonal gaps are called nodes of Ranvier. In this report I refer to them as axon nodes.

The myelin sheath is important in that it changes the way signals propagate down the axon. Without the sheath, signals flow smoothly down the membrane of the axon, each spiking patch of membrane stimulating a patch of membrane right next to it. With the sheath, the signal jumps from exposed node to exposed node. This form of propagation is called saltatory propagation and is faster than propagation without the myelin sheath. Figure 1 shows a neuron depicting axon nodes, myelin sheaths and saltatory propagation.

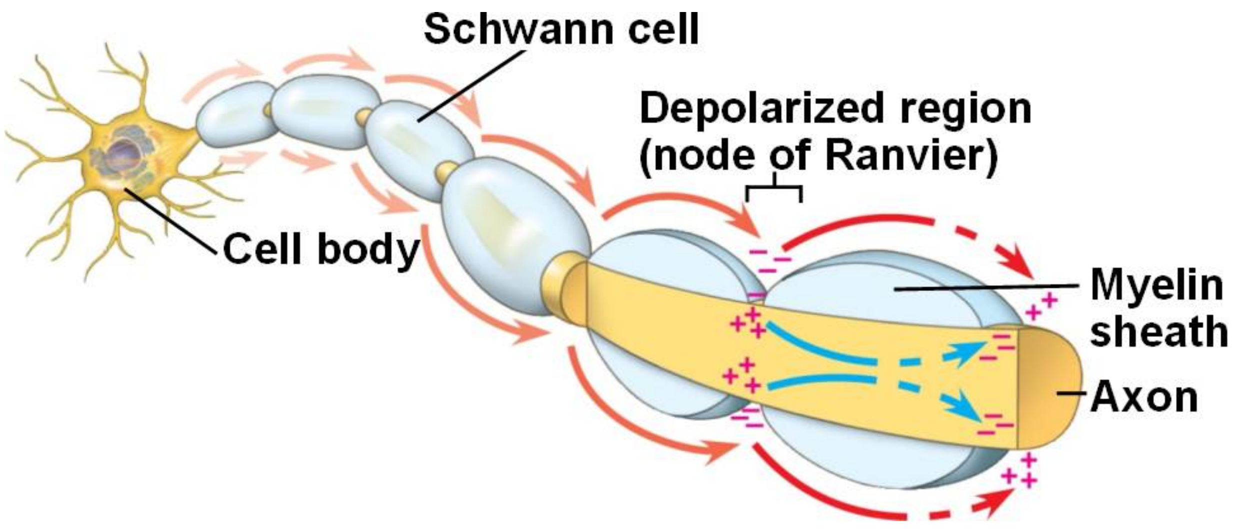

Figure 1: Depiction of a neuron with an axon and nodes seperated by myelin sheaths.

Hodgkin-Huxley Model (1952)

The wall of the axon, the cellular membrane, is polarized. A concentration gradient of ions is maintained between the inside and the outside of the membrane so that the inside of the cell is more negatively charged than the outside. Hodgkin and Huxley initiated action potentials in the squid axon by stimulating it at a point with an external potential. They explained that when an action potential occurs, positively charged sodium ions would flood into the cell through a point in the membrane of the axon, suddenly raising the membrane potential at that point, and disturbing the membrane potential a little further down the axon. After the flood of sodium ions into the axon, potassium ions leave through the membrane, the flood of sodium into the axon at the inital point slows and eventually stops, "resetting" the membrane potential of the axon at that point to its resting potential. Meanwhile at a point nearby on the axon, after being destabilized by it's neighbour, the same thing occurs and the potential spike is observed a little further down. This continues and the spike can be said to travel down the axon, and all the while the axon attempts to restore the concentration gradient and maintain a polarized potential all along the inside and outside of the cellular wall.

Though Hodgkin and Huxley didn't know it at the time, the concentration gradient is maintained by ion channels in the cellular membrane. These ion channels are proteins embedded in the membrane that can selectively allow ions in and out of the cell, controlling ionic currents through the membrane. Without knowledge of ion channels, Hodgkin and Huxley modelled the ionic currents as selective permeabilities in the membrane, controlled by "activation" and "inactivation" variables that depended on the voltage of the membrane. These would eventually be understood as voltage-gated ion channels that open and close depending on the current voltage of the membrane.

The Hodgkin-Huxley model is a set of four differential equations that can be solved in response to a stimulus for a solution of the membrane voltage. The model successfully produces action potentials and closely matches the experimental data acquired from the squid axon. Since then, the model has been used as the basis for more complicated models, and it's parameters can be tweaked for a wide variety of applications outside of the squid axon.

The equations in the Hodgkin-Huxley model that form the basis of our model are the following:

$$C_m \frac{dV_m}{dt} = -I_{Na} - I_K - I_L + I_{stim} \tag{1}$$

The equation is an application of the capacitance equation where the cellular membrane is a sort of capacitor because it separates charges from the inside and outside of the cell. The term C_m represents the membrane capacitance, and V_m represents the potential difference between the inside and outside of the cell. V_m is called the membrane voltage. The current terms I_{Na} and I_K represent the ionic currents due to sodium and potassium through the cellular membrane. Hodgkin and Huxley introduced an $I_L$term for what they called 'leakage' current, to account for current flow that wasn't caused by sodium or potassium. The current terms are treated using Ohm's law and in turn obey more equations:

$$\begin{aligned} I_{Na} =& \quad \bar{g}_{Na} m^3 h (V - E_{Na}) \\ I_K =& \quad \bar{g}_K n^4 (V - E_K) \\ I_L =& \quad \bar{g}_L (V - E_L) \end{aligned} \tag{2}$$

In these equations, the $\bar{g}$ terms are conductances (the inverse of resistance). The m, h, and n variables are dimensionless quantities between 0 and 1 and represent sodium channel activation, sodium channel inactivation, and potassium channel activation respectively. These variables capture the effect of the ion channels in the cellular membrane. The E_{Na}, E_K and E_L terms are constants called Nernst potentials.

The final pieces are the equations that govern the m, h, and n variables. These three variables are governed by first-order differential equations and by transition rate functions α and β. These functions are voltage dependent and represent the number of times per second that a closed gate opens (in the case of α) and the number of times per second that an open gate closes. These equations are given by:

$$\begin{aligned} \frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h \\ \frac{dn}{dt} &= \alpha_n(1-n) - \beta_n n \end{aligned} \tag{3}$$

The alpha-beta functions are found empirically, and each of the m, h and n variables have their own pair of alpha-beta functions:

$$\begin{aligned} \alpha_m &= 0.1\frac{V+40}{1-exp(-(V+40)/10)} & \alpha_h &= 0.07exp(-(V+65)/20) \\ \beta_m &= 4exp(-(V+65)/18) & \beta_h &= \frac{1}{exp(-(V+35)/10)+1} \end{aligned}$$

$$\begin{aligned} \alpha_n &= 0.01\frac{V+55}{1-exp(-(V+55)/10)} \\ \beta_n &= 0.125exp(-(V+65)/80) \end{aligned}$$

These equations can be solved for V_m using the the following scheme:

1. Set initial values for V_m, m, h, and n.

2. Calculate values for I_{Na}, I_K, I_L, and all the alpha-beta functions using the current values of V_m, m, h, and n.

3. Numerically integrate the differential equations on V_m, m, h, and n (equations (1)

and (2)).

4. Repeat from step 2.

Figure 2 shows an example solution of the membrane voltage with action potentials caused by a stimulus.

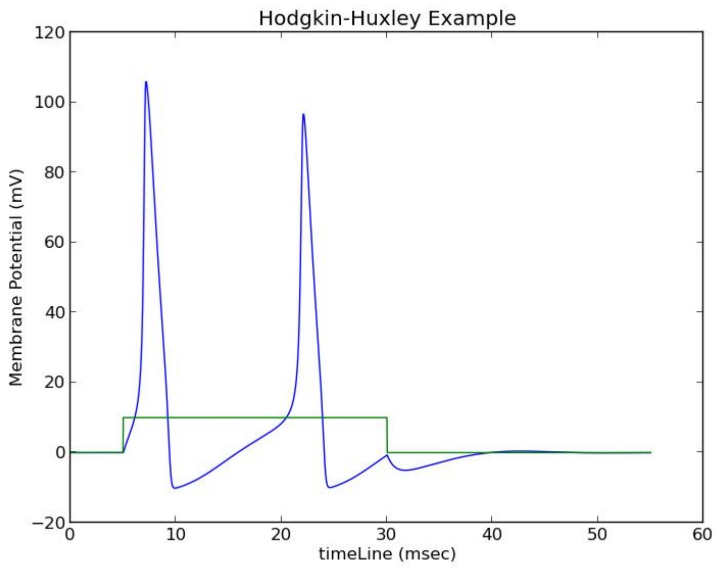

Figure 2: A solution of the Hodgkin-Huxley model (in blue) with an applied stimulus.

McNeal Model (1976)

In 1976 Donald R. McNeal published a paper analyzing a model of interconnected axon nodes. The paper included a model for a simple stimulus to be applied at a single node and for the action potentials to propagate from one node to the next. We adapted this model for our purposes and used it to model an arbitrary number of axon nodes connected to each other. This model also takes into account the dimensions of the neuron, that is, its length, its diameter, and the width of the axon node gap between the myelin sheaths.

In this model the equation on the membrane voltage, in addition to the ionic current terms, has current terms for the currents leaving the node, the currents entering the node, as well as a stimulus current caused by an external potential.

For a given axon node n, the change in its membrane voltage V_n depends on an axial conductance G_a, which represents the conductance between the nodes, the membrane voltages of the node's neighbours V_{n-1} and V_{n+1}, and the external potential felt by the node and it's neighbours due to some stimulus $V_{e,n-1}$, $V_{e,n}$, and $V_{e,n+1}$. In addition to these new terms are the usual terms for the ionic currents through the membrane of the node. The equation is given by:

$$C_m \frac{dV_n}{dt} = G_a(V_{n-1} - 2V_n + V_{n+1} + V_{e,n-1} - 2V_{e,n} + V_{e,n+1}) - \pi dl(i_{Na} + i_K + i_L) \tag{4}$$

The current terms become lowerscript i's because they are current densities; current per unit area of cellular membrane. The πdl term multiplies by the sum of the current densities to form a term representing the total ionic current. The axon node is modelled as a cylinder and the wall of the cylinder (not the caps) serves as the part of the cellular membrane exposed to surrounding ions. The area of this wall is πdl where d is the diameter of the cylinder and l is its length.

The external potential felt by a node V_e is assumed to be due to a stimulus I from a monopolar spherical electrode r distance away from the node. Assuming an isotropic medium (where the permittivity ϵ and the permeability μ are uniform in all directions), the expression for V_e is given by

$$V_e = \frac{\rho_e I}{4\pi r} \tag{5}$$

Where ρ_e is the resistivity of the medium.

The McNeal model is solved by choosing initial values for V_m, m, h, and n, like in the Hodgkin-Huxley model, as well as a position r for the central node (where $n = 0$) and a stimulus I. The positions of the other nodes are calculated based on their index n and a value for the internodal length L. Essentially each node is located a multiple of L away from the central node. Each node's position is precisely nL units away from the central node. Given some position coordinates for I, the distance r between the stimulus and node n can be calculated from a little geometry and Pythagorean theorem.

The actual values for the dimensions are calculated in the following way. McNeal choose a value of $20\mu m$ for the fiber diameter D, which represents the diameter of the axon *and* the myelin sheath. McNeal then cites a source stating that the axonal diamter d is calculated from the ratio:

$$\frac{d}{D} = 0.7$$

The internodal length L is then calculated from

$$\frac{L}{D} = 100$$

Finally, the nodal gap width l is simply $l = 2.5\mu m$.

Nerve Damage Model (Boucher, Joós, Morris 2012)

With a group of axon nodes connected to each other able to experience propagating action potentials, the task was to be able to damage the nerve and to then see the effect this had on the propagation of signal and the measured compound action potential. The damage model used in this simulation was developped by Professor Bela's group and focuses on a specific type of damage related to the functioning of the sodium channels. In this model the voltage-gated sodium channels are much more excitable and respond to lower voltages than normal. Mathematically this corresponsds to a left shift in the sodium activation and inactivation variables m and h, caused by left-shifting the alpha-beta functions α_m, β_m, α_h, and β_h. This is done by simply replacing the function argument V with $V + LS_{AC}$ in the alpha-beta functions, where LS_{AC} is the left-shift (in volts) of affected channels. Each node has a property AC which represents the fraction of affected channels on the node. In addition to solving for m and h using the regular alpha-beta functions, we must now calculate values for m_{LS} and h_{LS} using the left-shifted alpha-beta functions. Accordingly, the expression for the total sodium current becomes the sum of the current due to healthy channels (with regular m and h) and the current due to unhealthy, affected, channels (with m_{LS} and h_{LS}):

$$I_{Na} = \bar{g}_{Na} m^3 h (1 - AC)(V - E_{Na}) + \bar{g}_{Na} m_{LS}^3 h_{LS} AC (V - E_{Na})$$

or simply,

$$I_{Na} = [m^3 h(1 - AC) + m_{LS}^3 h_{LS} AC]\ \bar{g}_{Na}(V - E_{Na})$$

Simulation

With the mathematical framework in place, a virtual arm is created for the simulation. The arm is a bundle of nerve fibers, and each fiber is a set of axon nodes connected to each other. Each of the nodes obey the equations explained above. Each fiber has a different diameter and is placed in a different position. A stimulus is placed some position away from the bundle of nerves and so each nerve fiber feels a slightly different stimulus. Many parts of the simulation can be tweaked for different virtual experiments: the number of fibers, the number of nodes each fiber has, the diameter distribution of the fibers, the radius of the whole bundle, the fraction of damaged sodium channels different nodes have, the position of the stimulus, the magnitude of the stimulus, and the amount of time to simulate.

The following is a table of the physical constants used in the simulation, mostly from McNeal's paper:

Table 1: Constants

Constant	Value	Description
V_r	-65.5 mV	resting voltage
c_m	1 ¯F/cm^2	membrane capacitance per unit area
$\bar{g}_{Na}$	120 mS/cm^2	sodium conductance per unit area
$\bar{g}_K$	36 mS/cm^2	potassium conductance per unit area
$\bar{g}_l$	0.25 mS/cm^2	leakage current conductance per unit area
E_{Na}	50 mV	Nernst potential of sodium
E_K	-77 mV	Nernst potential of potassium
E_l	-54.4 mV	Nernst potential of leakage stuffs
ρ_e	300 $\Omega \cdot$ cm	external resistivity
ρ_i	110 $\Omega \cdot$ cm	internal resistivity
LS	35 mV	left-shift
D	0.0019-0.021 cm	fiber diameter
l	$2.5 * 10^{-4}$cm	axon node length
I	-2.59 ¯A	magnitude of stimulus current

Some results will be shown after the code and the details of the simulation and its implementation are examined more closely.

Code

The code for the simulation is available open-source and online at github.com/mef51/FiberLoss. This section will give a brief overview of how the code is structured and a look at some of the algorithms used.

Classes

Several classes are defined that describe and implement the different components of the simulation.

- `AxonPositionNode:` This class defines the concept of the axon node. Specifically it creates variables that will track the node's index, it's position, the fraction of affected channels it has and its dimensions. In addition, the solution variables are stored in this class: each node stores its own V_m, m, h, and n solutions. Moreover it is here that the alpha-beta functions and the method of numerical integration is defined. All the mathematical calculations happen within instances of this class. Each node in the simulation creates an instance of this class. This class also defines somes plotting functions that will generate images of plots. This is the primary visualisation tool in the simulation.

- `NerveFiber:` The NerveFiber class represents a single nerve and essentially stores a set of nodes that are connected to each other. When the fiber is instatiated it creates AxonPositionNode classes for each node and assigns it it's position, index, and its damage. A NerveFiber class is instantiated for each nerve fiber in the simulation.

- `NerveBundleSimulation:` This class manages the simulation. It tracks the timeline, the attributes of nerve bundle (including how many fibers there are and how many nodes to create per fiber), and implements the main simulation loop. When this class is instantiated it will create and place the NerveFibers. If it succeeds, the simulation can be started. The main simulation loop iterates over every moment of time. In each moment it will iterate over each nerve fiber in the bundle, and for each fiber it will iterate over each node in the fiber. It computes the distance between the node and the external stimulus and then steps the node forward in time by calling the node's integration method. This loop continues until every node has been stepped, in every fiber, and for every moment of time. When the simulation is complete the data is stored in memory under each AxonPositionNode class. From this point the plotting functions can be called to produce a plot for each node in the simulation, or the entirety of the data generated can be dumped into some machine readable format for future analysis.

Integration

The equations on V_m, m, h, and n are solved numerically using a very simple scheme. The equations are solved using what's referred to as the Euler-forward method, which is highly prone to error if dt is not small enough but is very easy to implement and worked well for our purposes. In the future it should be straightforward to replace the current integration scheme with a more robust Runge-Kutta method or similar.

The Euler-forward method can be reached through a simple manipulation of the differential equation. Let $f(x)$ be some differentiable function that obeys the differential equation:

$$\frac{df}{dx} = g(f)$$

where $g(f)$ is some arbitrary function. Recalling the definition of the derivative, substitute $df = f_2 - f_1$, $f = f_1$, and $dx =$ some arbitrarily small number. Then,

$$\frac{df}{dx} = \frac{f_2 - f_1}{dx} = g(f_1)$$

Now solving for f_2 we arrive at:

$$f_2 = dx \cdot g(f_1) + f_1 \tag{6}$$

Using this expression, we can pick an initial value for f_1 and a small number for dx (for example $dx = 0.025$) knowing that the smaller dx is, the more accurate the solution, we then calculate the "next" value f_2. To move forward in time, we just set $f_1 = f_2$ and repeat the process, calculating the "next, next" value using the equation on f_2. Storing all these values, we ultimately get at a solution of f with arbitrary precision and arbitrary length. In highly intensive applications this method is not very efficient, but for this project it serves just fine.

User Interface

The current interface for a researcher wishing to study the model is hardly ideal –it is simply a few lines of code stuck at the bottom of the file. In the future it will be much more pleasant to have some sort of visual interface to set parameters. In the meantime however, the current method still provides lots of control.

The stimulus current is specified with a simple object. This object specifies the x, y, and z position of the stimulus, and its magnitude. The code for that looks like this:

```
stimulusCurrent = {
    "magnitude" : -2.59, # uA. the current applied at the surface
    "x"         : 0   # cm
    "y"         : 0.3 # cm
    "z"         : 0   # cm
}
```

The nerve is specified with a similar object. The nerve object represents the bundle as a whole and specifies the number of fibers there are, the number of nodes each fiber has, the x, y, and z position of the bundle and the length of each axon node. In addition, the minimum and maximum diameter of a fiber is specified and the algorithm attempts to randomly create and place fibers so that they are not overlapping. The diameter of each fiber is randomly generated and forced to be in the range specified by the minimum and maximum diameter. That code is written as:

```
nerve = {
    "numFibers" : 2,
    "numNodes"  : 11,
    "fibers"    : [], # initialize an empty list
    "radius"    : 0.0, # cm
    "x"         : 0.0, # cm
    "y"         : 0.0, # cm
    "z"         : 0.0, # cm
    "minFiberDiam" : 0.0019, # cm
    "maxFiberDiam" : 0.0021, # cm
    "axonalLength" : 2.5e-4 # cm
}
```

The final property of the nerve object is a "damage map". This object specifies the proportion of damaged channels in each node by index. It is a set of key-value pairs where the key is the index of a node and the value is a number from 0 to 1 representing the proportion. For example, attaching the following property to the nerve object:

```
damageMap : {
    11 : 0.9 # heavily damaged node at n = 11
}
```

will cause the node with index $n = 11$ to have an AC of 0.9. Currently this affects all nodes with $n = 11$ in all fibers.

With all this setup out of the way the simulation can be started, the plots can be created, and the data dumped to a file.

```
T = 55 # ms
dt = 0.025 # ms
simulation = NerveBundleSimulation(T, dt, nerve, stimulusCurrent)
simulation.simulate() # modifies the nerve object
plotInfoOfNodes(simulation)
simulation.dumpJSON("data.json")
print "Done."
```

The complete source code is available free and open-source online at
github.com/mef51/FiberLoss

Results

The following is an example simulation run using the model described above. In this simulation, a single fiber with 41 nodes is stimulated. The eleventh node in the fiber, that is, the node with $n = 11$ is damaged completely with $AC = 1.0$. By our convention, the node that is closest to the stimulus is given $n = 0$, with the nodes on the right given $n = 1, 2, 3, 4...$ and the nodes on the left given $n = -1, -2, -3, -4...$ and so on. In the graphs you'll see information on the membrane voltage of the node, the time-course of the m, h, and n variables, and the time-course of the ionic currents. We can see healthy, action potentials at $n = 1$, and at $n = 11$, where all the channels are damaged, no action potentials at all, and only a slight response to the stimulus. Moreoever, past the damaged $n = 11$, node, we see only a single action potential has survived at the healthy $n = 12$, unlike the multiple spikes seen at $n = 1$.

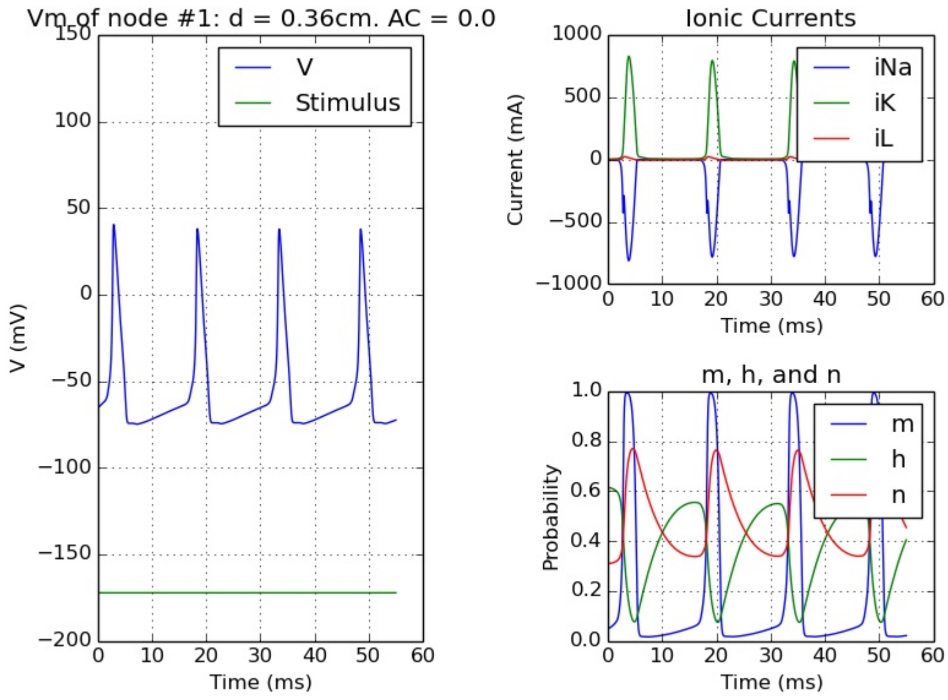

Figure 3: Multiple action potentials caused by a stimulus

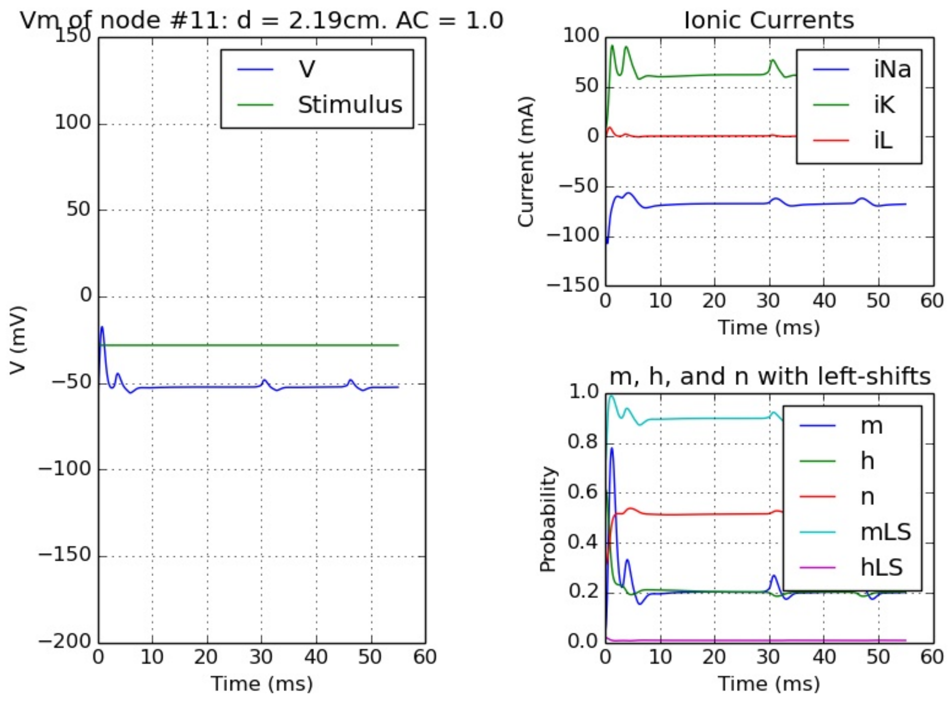

Figure 4: Response of a damaged node

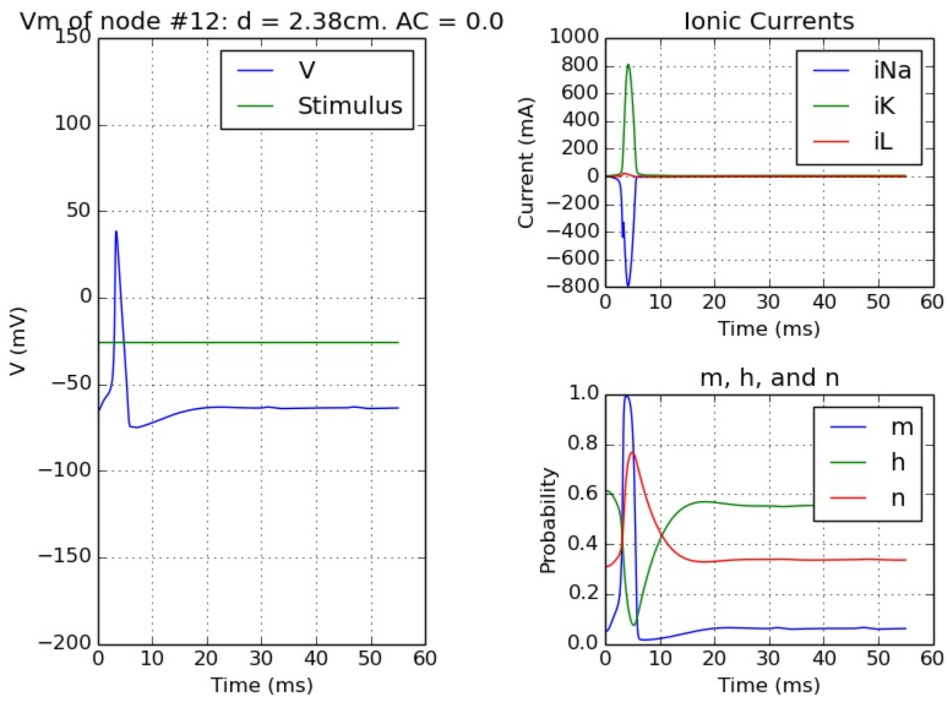

Figure 5: Faulty signal transmission past the damaged node

These results are promising with regard to the model's usefulness for studying these types of situations. Much more simulations will be needed to produce concrete results!

Conclusion

During my summer at the uOttawa Physics Department I worked with Professor Bela Joós to create a simulation model of an experiment whereby nerve damage in an arm is assessed by stimulating near the elbow and measuring the propagated signal down the arm around the wrist. We succeeded in completing a preliminary Python script that simulates a bundle of nerve fibers with axon nodes connected to each other, and were able to observe effects like signal propagation and signal loss due to damage. Our model was built up from earlier models of the neuron, specifically the famous Hodgkin-Huxley model from 1952 and the McNeal model from 1976. In the future our goal is to refine the simulation model and to use it to answer specific questions about nerve damage. Can we characterize neural pathologies by the measured signal? What kind of damage is completely destructive to signal propagation? If a single node is completely damaged is the whole fiber useless? These are questions we should be able to answer with our model. The answers to these questions can open doors to diagnosis and treatment of patients.