**Summer research internship**

Computational Neuroscience Simulation

at the University of Ottawa

Martin Matte

Submitted to Dr Bela Joos

University of Ottawa

August 29, 2016

**Contents**

Summary **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 01

Introduction **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 02

Theoretical background **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 3-11

Code used **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 11-13

Result **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 13-15

Discussion **. . . . . .** **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 16

Conclusion **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 17

References **. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 18

**Summary**

# During the summer of 2016, I worked under the supervision of Dr. Bela Joos on a research project. The purpose of this project was to model the electrical signal of a bundle of myelinated neurons like the ones found in one’s forearm. The goal of this model was to produce data that closely matched what would be produced experimentally by the threshold tracking technique. Once the model was in place, I damaged the voltage-gated sodium channel of some nodes of Ranvier in order to mimic a condition in certain neuropathic disorder. The main goal of this project was to help us find clues that could reveal if a nerve is damaged even before the symptoms are present. In damaged nerves, the speed of propagation of the compound action potential and its duration become altered. In the Results section, I highlight some data from the simulation that suggested we could possibly determine if the damage is present in a real-world experiment. In conclusion, this internship was very useful in terms of the experience and knowledge acquired.I was able to challenge my problem-solving skills on various complex problems, which strengthened my desire to pursue a career in science and technology. I would like to thank, Dr. Joos for the opportunity of working with him and for the countless hours he spent helping me on various problems.

# Introduction

# During the summer of 2016, from May through August, I worked under the supervision of Dr. Bela Joos on a computational neuroscience research project. I worked on the third-floor of Macdonald, the physics department building at the University of Ottawa. Since this was my first time working on a neuroscience project. I spent the vast majority of the first two month learning as much as I could about the properties of neurons. The main goal of this project was to create a computational model of a nerve bundle that could simulate how the electrical signal travel through a nerve, mimicking the data that would be generated experimentally using the threshold tracking technic. Once I mastered basic knowledge, I set out to build the model step by step. The first step was to build the nodes of Ranvier, which is where the action potential is created. After consulting with Dr Joos, we decided to use the Hodgkin-Huxley model to simulate the action potential at the different nodes of Ranvier. The second step was to add the damage. This damage would increase the excitability of the nodes by left-shifting the voltage-gated sodium channel. The third step was to connect the nodes in order to have a complete axon. The last step was to connect axons of different sizes together in order to generate a nerve bundle.

# Theoretical background

# This model was created combining the results of three research papers: The Hodgkin–Huxley model created in 1952, the Coupled left-shift of the sodium channel model created by Professor Joos’ group in 2012 and finally the McNeal model created by Dr. Donald McNeal in 1976.

# Neurons

# In order to get a sense of how these different models fit together, it helps to look at the physical aspects of a neuron.

# C:\Users\martin\Desktop\python\images\neuron_axon.png

Figure : image of a myelinated neuron from google images

# A neuron is composed of dendrites, a cell body, and an axon. For the purpose of this research, we will focus solely on the axon. The axon is either myelinated or not. In our case, all the axons simulated are myelinated. The axon has two main parts: the nodes of Ranvier (where the action potential is generated), and the myelin sheath (which acts as an insulator in separating the nodes).

# Hodgkin–Huxley model (1952)

# The nodes of Ranvier are responsible for generating the action potential or the electrical nerve impulse. In physiology, an action potential is a short-lasting event in which the electrical membrane potential of a cell rapidly rises and falls, following a consistent trajectory. The action potential is caused by special voltage-gated ion channels present in the nodes. The Hodgkin-Huxley model is a mathematical model that describes how action potentials in neurons are initiated and propagated. As you can see in figure 2, the model can be represented by a simple electronic circuit. Cm,EL,GL,ENA,EK represent the membrane capacitance, the leak current resting potential, the leak current conductance, the sodium channel resting potential, and the potassium resting potential, respectively. GNA and GK are the sodium channel conductance and the potassium channel conductance respectively. The main aspect of the Hodgkin and Huxley model was the ability to determine how these two conductances changed with the voltage applied to the circuit. The model successfully produced action potential that closely matched experimental values.

# C:\Users\martin\Desktop\python\HHcompt.gif

Figure : Hodgkin and Huxley electronic model from http://www.genesis-sim.org/UGTD/Tutorials/cnslecs/cns1.html

# The equations that form the basis of the model are:

Equation 1

# C:\Users\martin\Desktop\python\images\Eq1.png

# The first equation is a representation of Kirchhoff's current laws with Vm representing the voltage across the membrane. Note that Equation 1 is only valid for the node receiving the external current stimulus.

# 

Equation 2

Equation 2 is the current equation using Ohm’s law.

Equation 3

# C:\Users\martin\Desktop\python\images\Eq2.png

# The third set of the equations represents how the current changes with the voltage. Note that V is the voltage across the membrane. The gn {\displaystyle {\bar {g}}\_{n}}is the maximal value of the conductance, and *n*, *m*, and *h* are dimensionless quantities between 0 and 1 that are associated with potassium channel activation, sodium channel activation, and sodium channel inactivation, respectively:

Equation 4

# C:\Users\martin\Desktop\python\images\Eq3.png

# The fourth set of equations represents the derivatives of the coefficients m,h and n.

Equation 5

# C:\Users\martin\Desktop\python\images\Eq4.png

# The last set of equations contains an alpha-beta function for n,m, and h. These functions enable us to calculate the rate of change of our three coefficients. In order to use these 5 sets of equations, we need to set an initial voltage. I chose -65.4945, which is the resting membrane potential on most neuron.

# The Coupled left-shift model of Nav channels

# In order to simulate the damage to the nodes of Ranvier, Dr. Joos’ group left shifted the voltage-gated sodium channels. Remember that the sodium activation and inactivation coefficients are m and h, respectively. In order to left shift the sodium channels, I purposely increased the voltage value received by the alpha-beta of m and h in the fourth set of equations. This leads to an increase in the excitability of the affected node of Ranvier. We can easily customize the amount of damage to the node by increasing or decreasing the voltage added. The sodium current equation from the second set of equations becomes:

Equation 6

# 

# The AC value is binary: either 1 (damaged) or 0 (intact).

# The McNeal model

# This model was used to connect the nodes together to form a complete axon, and it also provided useful information needed to create the axon. In Figure 3 you can see the completed circuit diagram of 2 connected nodes.

Figure from google images

# C:\Users\martin\Desktop\python\images\complete_circuit.png

# To make the model simpler, I made 3 approximations. First, the exterior of the axon is at equipotential. Second, the membrane capacitance of the internode is zero. Third, the internode membrane resistance is infinite. To compute all the values required, we needed to know the values of two constants: the resistance between the node and the capacitance at the node. Once we knew the diameter of our axon, we could compute both values using:

Equation 7

Equation 8

In the equations above, d,pi,L,cm,l are the diameters, the internal resistivity, the internodal length, the membrane capacitance per unit area, and the nodal length, respectively.Once we knew Ga , we could calculate the current going through the internode using Ohm’s law. To have an expression for the current for every node, we needed to consider 4 different nodal situations: the first node, the last node, the node receiving the current pulse, and all the other nodes.

The first node is connected only to the second node and its current equation is:

Equation 9

The last node is connected only to its previous node and its current equation is:

Equation 10

The node receiving the pulse is connected to both adjacent nodes, however, it also receives external current and its equation is:

Equation 11

All the other nodes are connected to their adjacent nodes and therefore their current equations are:

Equation 12

Two important points about the last 4 equations: First, the d means the derivative, and second, the current going into the node is positive and the current leaving the node is negative.

As mentioned earlier, the first and last nodes are connected only to one adjacent node, which is not a realistic approximation. To solve this problem, I created a nerve of 30 centimeters and cut the first and last 5 centimeters.

**Nerve Bundle**

The last eleven equations required a set of constants given in Table1. Once we plugged in these constants, we had enough data to create many computational axons of different sizes. I decided to create a nerve 20 centimeters long (cut from 30 centimeters). This nerve contained 20 axons ranging linearly in size from 15 micrometers to 24.5 micrometers. This is all we needed to start the simulation.

Table 1: Constants used

|  |  |  |
| --- | --- | --- |
| Symbol | Value | Description |
| Vr | -65.4945 mV | Resting potential |
| Cm | 1 uF/cm2 | Membrane capacitance per unit area |
| gNa | 120 mS/cm2 | Sodium conductance per unit area |
| gk | 36 mS/cm2 | Potassium conductance per unit area |
| gL | 0.25 mS/cm2 | Leakage current per unit area |
| ENa | 50 mV | Nernst potential for sodium |
| Ek | -77 mV | Nernst potential for potassium |
| EL | -54.4 mV | Nernst potential for leakage current |
| pi | 110 000 mΩ\*cm | Internal resistivity |
| L/d | 142.86 | Ratio of internodal length on diameter |
| l | 1 uM | Nodal length |

**The Code**

The software I developed for the simulation is available open source at github.com/mart31/Computational-Neuroscience. In this section, I will give a brief overview of the different functionalities of the program I created. The software was written in python. I wrote three different modules, each with a unique set of functionalities. The first is the nerve simulation module, the second is the data analysis module, and the last is the sauvegarde graphique module.

**Nerve simulation module**

As the name implies, the nerve simulation module is where the nerve is simulated. There are two main classes in this module. The first is the axon class, which creates a virtual axon, sets a time interval, initiates its values, sets the damage on the designated nodes of Ranvier, and finally starts the simulation. The axon potential function of the class Axon is the heart of the simulation. Using the formulas from the theoretical background section, it calculates the membrane potential of all the nodes of Ranvier during the time interval given. The second is the nerve class, which enables the creation of a bundle of axons and saves the data into CSV files.

**Data analysis module**

This module downloads the data generated during the simulation and extracts useful information using the Use\_data class. This class contains 16 different functions to help visualize the data. We cut the first and last 5 centimeters of the nerve in order to remove the edge effect. Since the threshold tracking technique will only reveal what the compound action potential looks like, the function that calculates the compound action potential is the central part of this module. I created the calculate\_nerve\_ap function in order to calculate the compound action potential from the data generated. To do so, at every point on the nerve we calculated the average potential from all the nodes located at that point. Once the compound action potential was created, we could analyze its duration and speed at every point on the nerve using the plot\_nerve\_ap\_speed and show\_ap\_width functions.

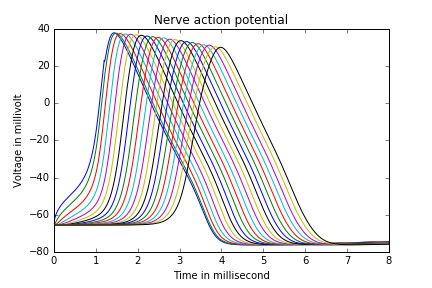
**Sauvegarde graphique module**

This module was copied from a program available online. Its sole purpose is to save the graphic generated by the code.

**Results**

In this section, I discuss the results generated through the simulation. In the discussion section, I highlight the weakness of the program and the possible solutions to these weaknesses.

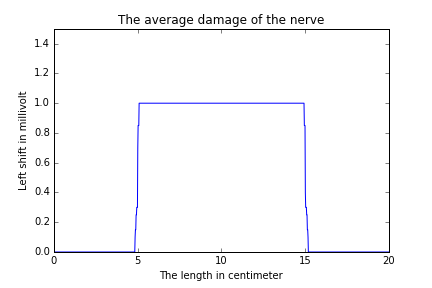
Figure



In Figure 4, we can see the compound action potential at a different point and a different time on the nerve. The curve to the left is where the current pulse is injected and the curve to the right is at the end of the nerve.

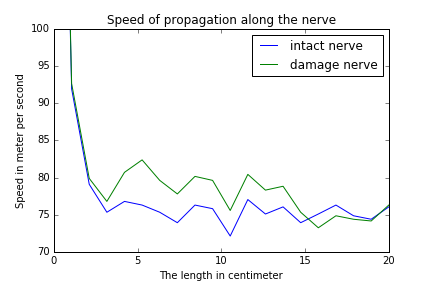
As the signal travels along the nerve, the max value of the action potential becomes lower and its duration becomes longer. This is due to the different traveling speeds among axons of different sizes. In the next three figures, we look at the relationship between the damage added and the traveling speed.

Figure

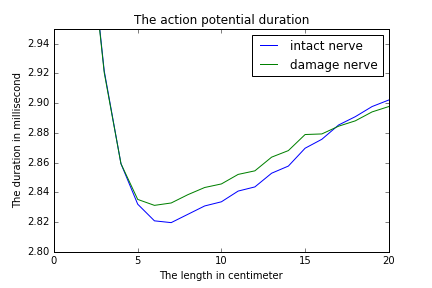


In this simulation, I added constant damage on all the axons starting at 5 centimeters and ending at 15 centimeters. On the next page in Figure 6 and 7, we can see that when the compound action potential enters the damaged zone, it speeds up and its durations change almost immediately. Two important points to note: first the speed of propagation in the first few centimeters is very high, which is due to the current being injected (which boosts the speed at which the action potential is being triggered), and second, the line representing the speed changes abruptly, which is due to the discrete nature of the nodes.

Figure



Figure



**Discussion**

By measuring the compound action potential at different points on the nerve, we could potentially determine if the damage is present just by analyzing the speed of propagation. Since in the real world we cannot compare the data measured with the “not-damaged” version of this data, we could look for a sudden jump in the speed or duration of the compound action potential possibly indicating if the damage is present.

However, we must take into account the weakness of this model. The many approximations used in this model, notably the Hodgkin-Huxley model, the myelin sheath being a perfect insulator and cutting the left and right side of the nerve in order to remove the edge effect, which all add some error to the model. When we inject the current into the nerve, only one node per axon gets influenced by it, when in reality nearby nodes would feel this sudden increase in voltage to different degrees. The biggest problem with the model is the way I calculated the compound action potential. When we calculated the average voltage at a point on the nerve, we discovered which node on every axon was located at that point and then calculated an average of their potential. This is the reason why, in Figure 6, the speed of the action potential change abruptly. In reality, the adjacent nodes would affect the measurement nearly as much. In my opinion, this problem creates the biggest error but is potentially the easiest to solve. If we knew the distance between the sensor and the nerve, we could compute the effect of all the nodes of all the axons in the simulated nerve by simple trigonometry. This would certainly generate more precise data.

**Conclusion**

By combining the Hodgkin–Huxley model, Mc Neal model, and the coupled left shift model from Dr. Joos’ group. I was able to simulate a nerve bundle like the bundle in one’s forearm. From the data generated by this model, we saw that the speed of propagation of the compound action potential is modified when it enters a damaged region. Using this information, we could potentially determine if the nerve is damaged before the symptoms appear. I highlighted some of the weaknesses, notably the way the compound action potential is calculated. In the future, our goal is to refine the simulation model in order to gain better insight into the electrical signal of the nerve, perhaps making it possible to diagnose some neuropathic diseases before the symptoms appear.

I would like to thank, Dr. Joos for the opportunity to work with him on this project. It was a very stimulating experience which taught me a great deal about research, and it strengthened my desire to solve large and complex problems.

