BIOMEDE 517 - Neural Engineering - Labs 1 and 2 - Dr. Cindy Chestek
Implications of Single Neuron Simulations
Kushal Jaligama

1 - Introduction

As technology develops at an ever-increasing rate, people question the implications that powerful technological advances will have on the human race. There exists a hypothetical that hyper-advanced technology could potentially lead to the diminishment of human population. A solution to this is augmenting human function to have a symbiotic relationship with technology, and approaching this would require deep knowledge of neural circuitry. Specifically, one must address the conditions of a neuron firing, and build up from the individual neuron level to map sets of neurons to patterns, which eventually can be reverse-engineered to augment human function. An action potential represents the electrical signals that neurons generate. Neurons can be represented as electrical circuits and be modeled (with adaptations to a cellular context) computationally using circuitry concepts and equations. This is useful because these models can be utilized to know how a neuron will react to a certain stimulus. Simulating these action potentials provide insight into neuronal firing. Mainen et.al. have discovered that regardless of the site of electrical stimulation of a neuron, an action potential is always generated from the neuron's axon hillock. This means that neurons have unidirectional firing and will affect which other linked neurons will generate an action potential. Mainen's findings are only one part of the puzzle. This paper explores the metaphor of a neuron as a circuit, the mathematical models borrowed from electrical engineering, adapted to suit a neuron's intricate cellular structure, and how action potentials are propagated down axons, represented as electronic cables.

2 - Methods

2.1 - Calculating Initial Conditions

The concentrations of ions inside and outside of the cell membrane create an electrochemical resting potential. The Goldman-Hodgkin-Katz Equation (appendix, Table 1, #1) yields a numerical value for this resting potential (V_m) of a cell based on these circuit-like properties of a neuron, using the values given in (appendix, Table 3). According to the Hodgkin-Huxley model, ion channels, or gates, of neurons contain particles that probabilistically open or close the channel to ion flow. These gates have opening and closing reaction rates (appendix, Table 1, #4, #5) which can be used in tandem with the resting potential to calculate the initial gating probabilities (appendix, Table 1, #3) of channel particles.

2.2 - Simulation of Dynamic Variables

Differentials can be used to simulate the state of the dynamic variables involved in action potential generation, namely the probabilities of m, h, and n gates being open and the membrane potential. Changes in gated channels can be modeled with alpha and beta sigmoid models (appendix, Table 1, #4, #5) used in a differential (appendix, Table 1, #6) to calculate the initial probability of a gate being open to ion flow. The Gating Differential Equation can be used for all of m, h, and n gates. The membrane potential can be simulated with

a given stimulus current in MATLAB using these differentials, and an axon potential is generated (appendix, Figure 1).

2.3 - Critical Length of Myelinated Segment

The critical length of a myelinated segment represents how far the myelin sheath can extend down an axon and still enable an action potential to be generated downstream. Thinking of this axon as an electrical cable, we can calculate the critical length based on the axial and membrane resistance of a neuron. Every axon has a characteristic length constant that relates to the critical length, which can be calculated using this constant, the resting potential, and peak membrane potential (appendix, Table 1, #7).

2.4 - Assumptions of Neuron Models and Axon Design

The NEURON model utilized for this portion was a standard one compartment neuron that utilized Hodgkin-Huxley ion channels. Simulating an action potential with this model entails using the conductance of each channel, probabilities that a channel is open, and a standard stimulus current to gather a characteristic membrane potential over time. A one compartment axon had multiple action potentials due to a stimulus that was for a duration longer than the length of the refractory period of an action potential.

2.5 - Model Axons of Multiple Diameters

These models were coded in NEURON, with a characteristic diameter, length and resistivity which can affect the speed of action potential propagation along the neuron's axon. Two separate model axons were used, with identical resistivity and lengths, but the second axon had double the diameter of the first. An additional parameter for these axons was the number of compartments it contained. Multiple compartments entails that an action potential propagates down these compartments, whereas as in a single compartment model, action potentials occur repeatedly along the same compartment.

2.5 - Mainen Neuron

Mainen, et. al. modeled a layer 5 pyramidal cell that was used to clearly explain the way action potentials propagate through the neuron. Simulations of the Mainen neuron model lead to the finding that action potentials always begin from the axon hillock and propagate downstream to the end of the myelinated axon, regardless of the point of stimulation. This occurs because the axon hillock has the greatest density of ion channels than all other points of the cell. The NEURON model loaded from this study illustrates the change in action potentials when a stimulus is placed on two different key locations, the soma and the dendrite of the cell.

2.6 - Simplified Neuron Design

The Mainen neuron model is an extremely precise and complex model, and several key observations can be made from this. However, observing the effects of changing parameters becomes non-trivial due to this complexity, so a simplified model was

created to understand how scaling a neuron changes its characteristic action potentials and membrane currents. This neuron was designed as a simple soma with a dendrite and an axon with myelinated and non-myelinated components. Each component utilized Hodgkin-Huxley parameters with given conductances for both the membrane and channels.

3 - Results

3.1 - Resting State Values

Table 2 in the appendix lists the initial conditions of a single neuron simulation. Generally, neurons have a resting potential of approximately -70 mV. This is determined by the concentrations of intra and extracellular sodium and potassium ions and the state of the gated ion channels. The Hodgkin-Huxley model is one that represents the membrane and its channels as items in a circuit. The cell's membrane is represented as a capacitor that is in a parallel circuit with variable resistors that represent ion channels, one for each type of ion that can flow. The Goldman-Hodgkin-Katz Equation takes into account the "resistances" of these ion channels as membrane permeabilities (for each ion), as well as the concentrations of ions present. If the parameters are all known (as they were in this exercise), then the membrane voltage can be calculated. The m, h, and n particles are at a steady state, and since the equation for Initial Gating Probabilities (appendix, Table 1, #3) takes into account the resting membrane potential, the alpha and beta sigmoid models converge to nominal reaction rates which ultimately dictate steady state gate probabilities.

3.2 - Time Course of Membrane Potential and Gate Particles

One can see in (appendix, Figure 1, Figure 2), that given a sufficient stimulus, the neuron depolarizes itself to a peak of approximately 60 mV. This is due to sodium ion influx into the cell since the m gate fully opens to allow ions to flow in as current is applied. Then, at the same time as the peak voltage is reached, potassium ions begin to flow out of the cell since the n gate begins to open up, and the h gate transitions to a not-inactivated state because sodium ions are still flowing into the cell, and the m gate begins to inactivate.

3.3 - Determine Threshold Voltage and Peak Voltage

In (appendix, Figure 1), one can see that as the m gates open fully (appendix, Figure 2), the membrane potential reaches a peak. When an insufficient stimulus is applied, the cell does not reach its threshold voltage level (appendix, Figure 3) because there are not enough ions flowing into the cell to cause the channel gates to open (appendix, Figure 4).

3.4 - Outcome of One Compartment NEURON Simulation

After creating a neuron with one compartment (a simple soma) in the NEURON language, one can see the effects of adding a stimulus for a set period of time (appendix, Figure 5). A stimulus current was placed at the midpoint of the soma.

This current caused an inflow of sodium ions to depolarize the cell until a peak voltage was reached, then a potassium ions begin to flow out of the cell, causing the neuron to repolarize to a negative voltage, stay in a refractory period for approximately 1-2 ms, and return to a resting voltage. The neuron encounters multiple action potentials, showing many voltage peaks and troughs in regular intervals as the cell continuously reaches its resting voltage after it repolarizes and immediately depolarizes due to a continuous stimulus current.

3.5 - Effects of Axon Diameter on Action Potential Propagation

The diameter of an axon directly affects the rate of action potential propagation. In this portion, the axon with a diameter of 10 um had an action potential traveling at a speed of 1.779 m/s, whereas the action potential for the axon with a 20 um diameter traveled at a speed of 2.527 m/s. These rates were calculated using NEURON, and the action potentials are shown in (appendix, Figure 6). Although the rate of action potential propagation did not double, it is still significantly higher in the larger diameter axon. This rate increase is also consistent along both the first and last compartments of the respective axons, showing that the rate of action potential propagation is increased throughout the entire axon when the characteristic diameter is increased.

3.6 - Key Results of Mainen Neuron Simulation

In (appendix, Figure 7), it is evident that when current is applied to the dendrite, the voltage of the dendrite slowly ramps up and rapidly peaks when the current causes a large enough electrochemical gradient to push the voltage above the threshold current. The voltage along the dendrite maintains a higher voltage for a longer period of time because the direct current causes enough ions to flow through the dendritic ion channels to keep the voltage above the threshold level. When current is applied to the soma, the action potential also begins from the axon hillock but the dendrite experiences a standard action potential due to an indirect current, one that is propagated rather than applied by a clamp.

3.7 - Outcome of Simplified Neuron Model

In the smallest scale of the simplified neuron model (12 um), the axon hillock generated a smaller current than that of the 24 um and 48 um models (appendix, Figure 8). The larger neurons experienced a larger current because there were more available ion channels at the hillocks of these neurons, and this causes a larger burst of current to propagate down the remainder of the axon.

4 - Discussion

The implications of understanding how action potentials are generated and propagate through neurons are far-reaching. There are plentiful types of neurons in the brain, and they can be modeled via software such as NEURON simply through modification of parameters

such as length, diameter, number of compartments, and what components they are connected to. Intricate structure of neurons can also be generated, and one can model how a stimulus applied to one point on these more complex structures results in action potentials propogating throughout the rest of the structure. This is useful because mapping the brain requires knowledge of interconnected complex structures, and more effective predictions can be made about the action potentials that propagate through the brain. This is necessary to know how to augment the brain, as complex processes do not have simple neuronal firing and must take into account how neurons fire in relation to one another.

References

Mainen ZF, Joerges J, Huguenard JR, Sejnowski TJ (1995) A model of spike initiation in neocortical pyramidal neurons. *Neuron* 15:1427-39

BIOMEDE 517 - Neural Engineering - Labs 1 and 2 - Dr. Cindy Chestek

Appendix

Kushal Jaligama

Table 1 - Equations Used for Single Neuron Modeling

#	Equation Name	Equation
1	Goldman-Hodgkin-Katz Equation	$V_m = 25 \ln \frac{P_K[K_o] + P_{Na}[Na_o] + P_{Cl}[Cl_i]}{P_K[K_i] + P_{Na}[Na_i] + P_{Cl}[Cl_o]}$
2	Nernst Equation	$E_{ion} = \frac{k_B T}{ze} \ln \frac{Co}{Ci} = \frac{25}{z} \ln \frac{Co}{Ci} (mV)$
3	Initial Gating Probabilities	$n_0 = n_\infty = \frac{\alpha_n(V_{rest})}{\alpha_n(V_{rest}) + \beta_n(V_{rest})}$
4	Alpha Sigmoid Model	$\alpha_n(V) = \frac{A_n(V_m - V_{1/2})}{1 - e^{-(V_m - V_{1/2})/k_n}}$
5	Beta Sigmoid Model	$\beta(V) = \frac{-A_n(V_m - V_{1/2})}{1 - e^{(V_m - V_{1/2})/k}}$
6	Gated Channel Differential	$\Delta n = \Delta t(\alpha_n(1-n) - \beta_n n)$
7	Membrane Potential Differential	$\Delta V_m = \Delta t(I_M - I_{Na} - I_K - I_L) / C_m$
7	Critical Length Equation	$V(x) = V_o e^{-x/\lambda}$

Table 2 - Resting State Values

Membrane Resting Potential (mV)	m Particle	h Particle	n Particle
-72.3406	0.0226	0.1529	3.4991e-04

Table 3 - Parameters for Goldman-Hodgkin-Katz Equation

P_K	P_{Na}	P_{Cl}
1.0	0.04	0.45
Na_o	K_o	Cl_o
145	4	120
Na_i	K_i	Cl_i
12	155	4

- These values come from Mainen, et. al.

Figure 1 - Single Neuron Action Potential

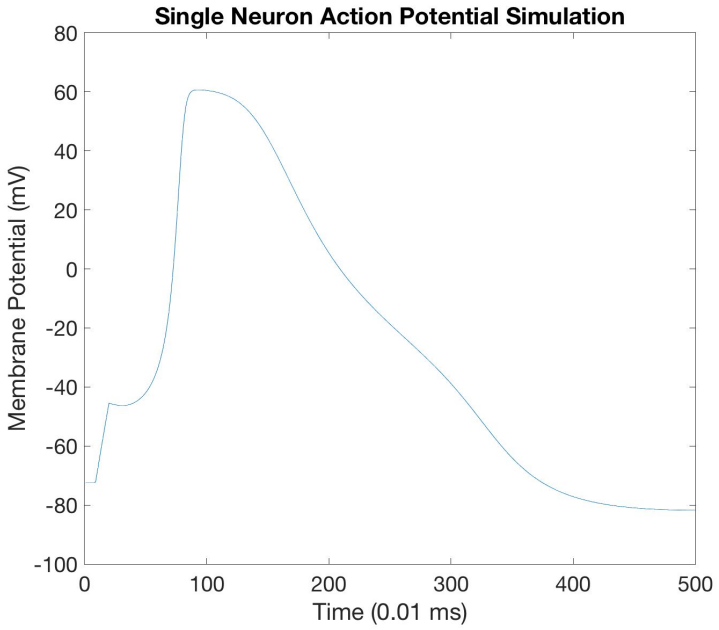


Figure 2 - Time Course of m, n, and h Gates

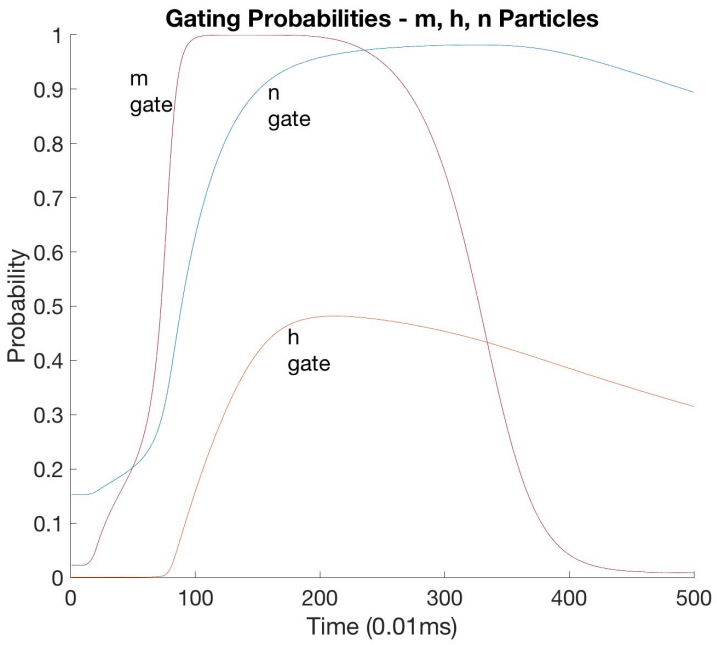


Figure 3 - Membrane Potential with Insufficient Current

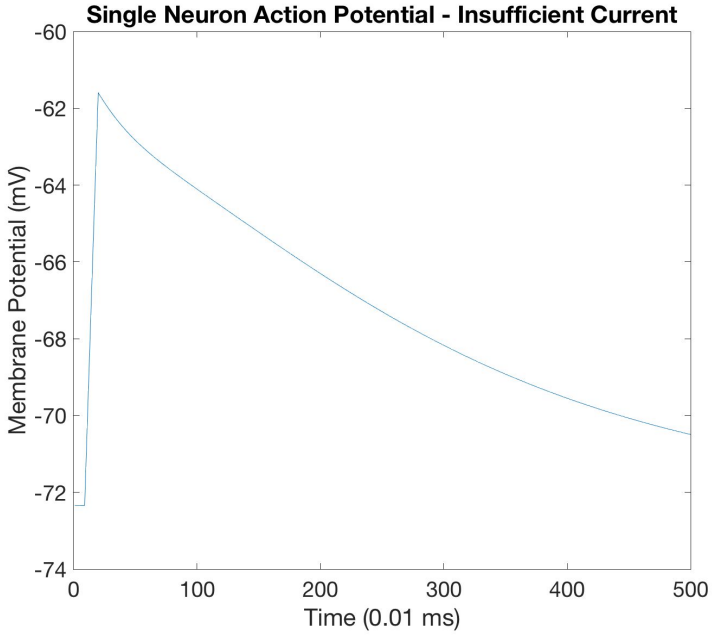


Figure 4 - Gating Probabilities with Insufficient Current

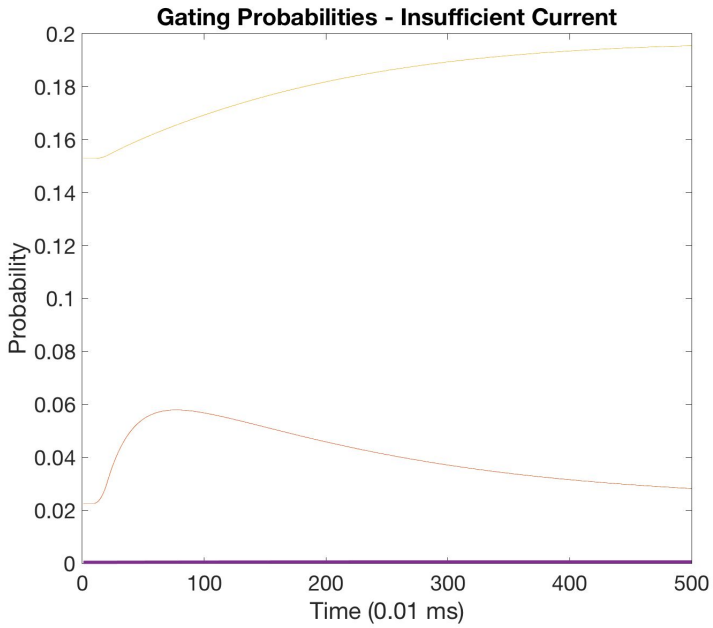


Figure 5 - Outcome of NEURON One Compartment Simulation

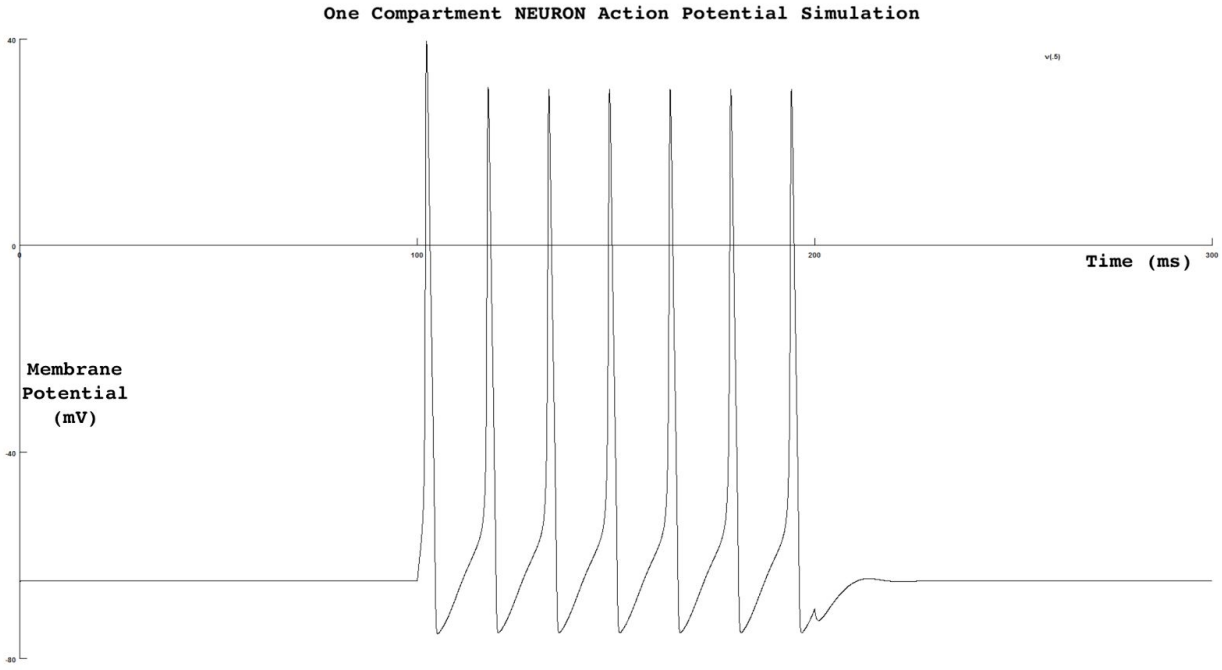


Figure 6 - Multiple Axons, Multiple Compartments

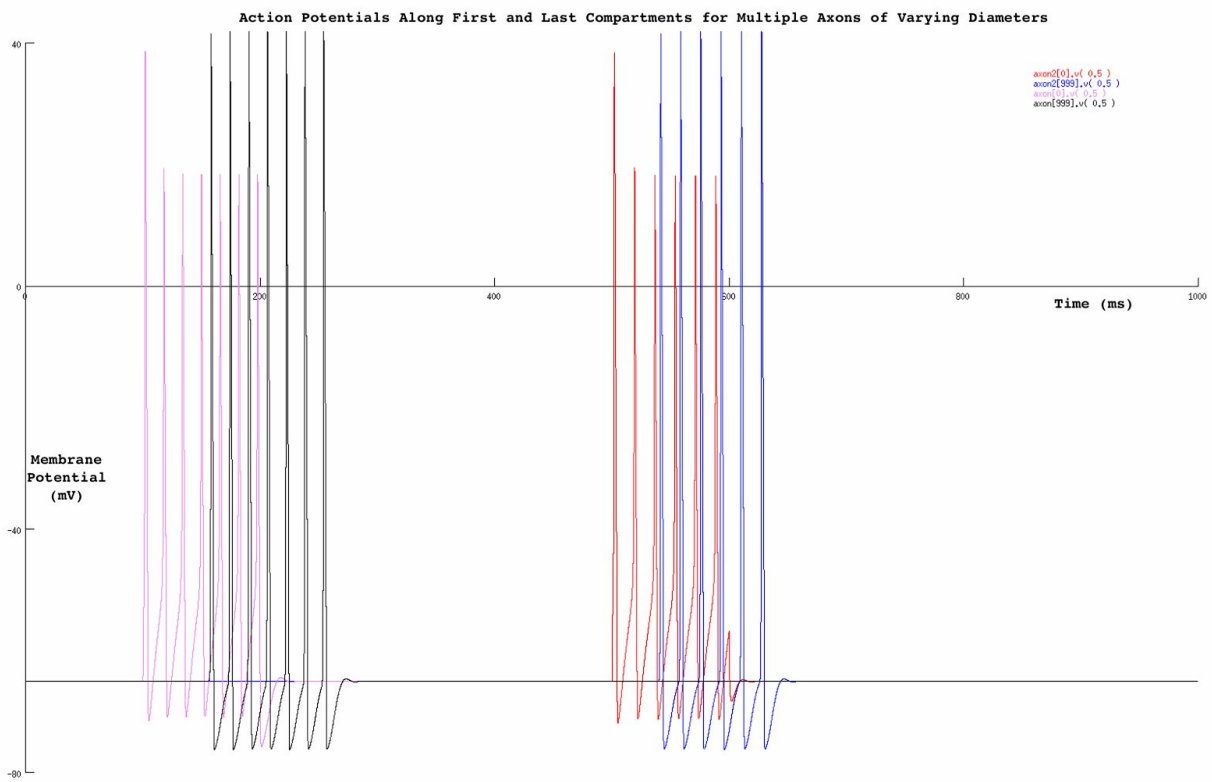
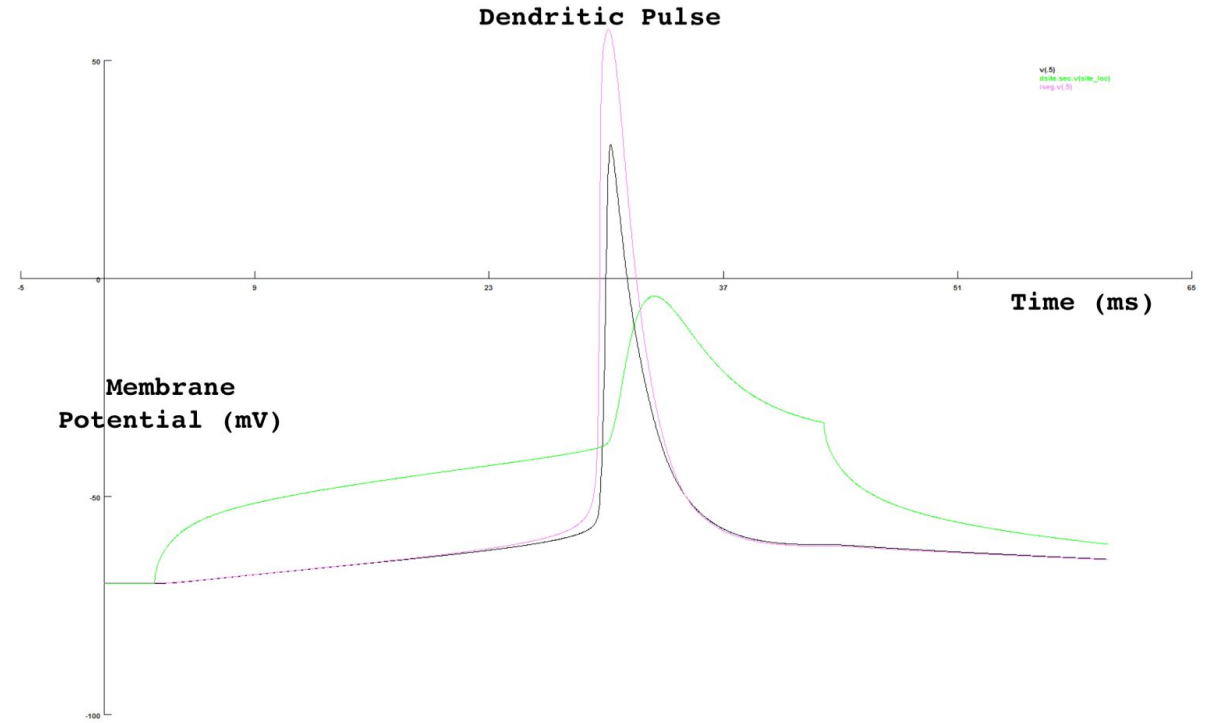


Figure 7 - Mainen Neuron Simulation



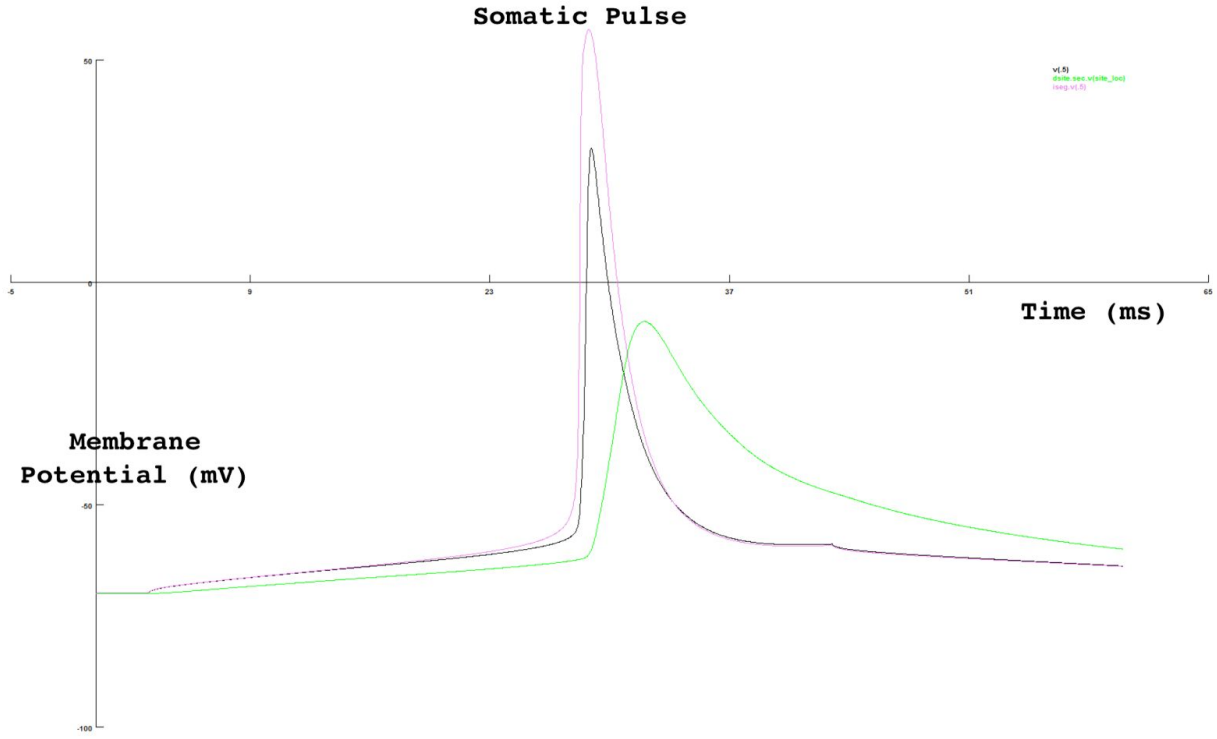
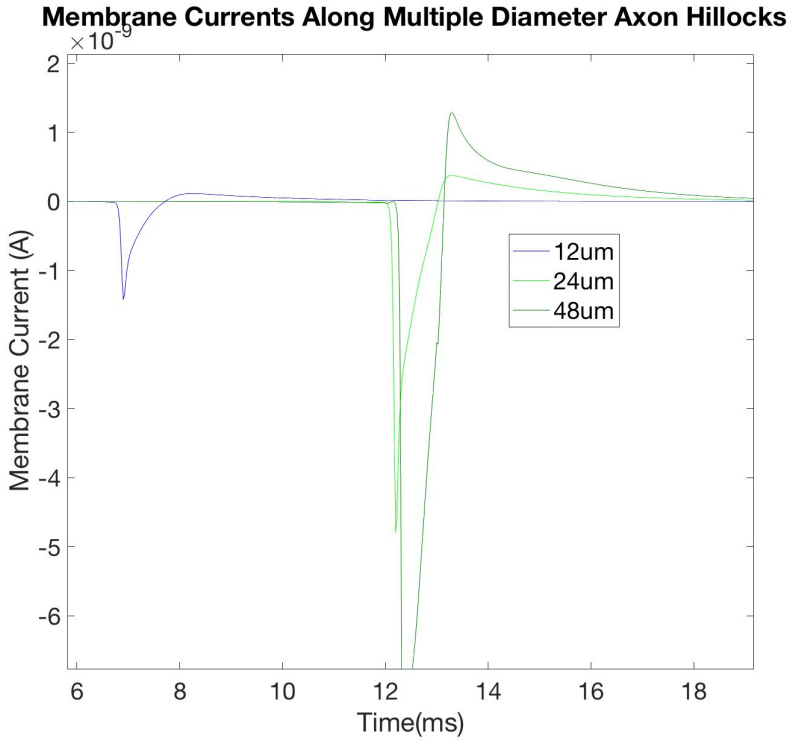


Figure 8 - Simplified Neuron Model at Various Scales



Code Utilized for MATLAB Simulations in Part 1 (organized by file name, execute Lab1.m)

```
% Lab1.m
% Part 1
% Problem 1
z_K = 1;
Co_K = 4;
Ci_K = 155;

E_K = 25/z_K*log(Co_K/Ci_K);

z_Na = 1;
Co_Na = 145;
Ci_Na = 12;

E_Na = 25/z_Na*log(Co_Na/Ci_Na);

% Problem 2
P_K = 1.0;
P_Na = 0.04;
```

```
P_Cl = 0.45;

Co_Cl = 120;
Ci_Cl = 4;

Vm_rest = 25*log((P_K * Co_K + P_Na * Co_Na + P_Cl * Ci_Cl) / (P_K * Ci_K + P_Na * Ci_Na +
P_Cl * Co_Cl))

% Problem 3
% Calculate the resting point for all dynamic channel variables
% These will be simulated over time to generate the action potential,
% m(inf), n(inf), and h(inf)
% V_rest corresponds to Vm (resting potential) of cell membrane

% Sodium Gate Probabilities
m_gate = gate_open_probability(Vm_rest, 0.182, -35, 9, 0.124, -35, 9)
h_gate = gate_open_probability(Vm_rest, 0.024, -50, 5, 0.0091, -75, 5)

% Potassium Gate Probability
n_gate = gate_open_probability(Vm_rest, 0.02, 20, 9, 0.002, 20, 9)

% Part 2 - Outputs the graphs for action potential, m, h, n gate
% probabilities
action_potential(Vm_rest, m_gate, h_gate, n_gate, E_Na, E_K)

% Part 3
% Problem 1 - threshold voltage = -45.48mV @ current of 250mA (obtained by
% looking at the graph and noticing where the voltage begins to spike)

% Problem 2 - calculate length constant associated with myelinated part of
% axon compartment
axon_radius = 1.5e-6 % m
R_m = 40000e-4 % ohm-m^2
R_i = 200e-2 % ohm-m
r_m = R_m / (pi * (axon_radius*2)) % ohm-m
r_i = R_i / (pi * (axon_radius)^2) % ohm/m
length_constant = sqrt(r_m / r_i) % in m (lambda)
length_constant = length_constant*10^(6) % um
```

```
% Problem 3 - how far can myelin extend down cable and still enable action
% potential to be generated downstream
% Normalize the action potential voltages
peak_voltage = 60
V_0 = peak_voltage - Vm_rest
V_thresh = -45.8
V_x = V_thresh - Vm_rest
critical_length = log(V_x / V_0) * -length_constant % um
critical_length = critical_length * 10^(-3) % mm

% Action Potential.m
% Part 2 of Lab 1
% Get the actional potential curve based on stimulus current and gate probs
function action_potential(V_m, m_gate, h_gate, n_gate, E_Na, E_K)
% Set up constants
C_m = 1; % uF/cm^2
g_bar_Na = 100; % mS/cm^2
g_bar_K = 50; % mS/cm^2
g_leak = 0.5; % ms/cm^2
E_leak = -72.5; % mV
time_step = 0.01;

total_time_steps = 500;

I_M = zeros(total_time_steps, 1);
I_M(10:20) = 100
for t=1:total_time_steps
    % Sodium
    g_Na = g_bar_Na * m_gate^3 * (1-h_gate);
    I_Na = g_Na * (V_m - E_Na);

    % Potassium
    g_K = g_bar_K * n_gate^4;
    I_K = g_K * (V_m - E_K);

    % Leakage
    I_L = g_leak * (V_m - E_leak);

    % Update your deltas
```

```
    % Voltage
    delta_V_m = time_step * (I_M(t) - I_Na - I_K - I_L) / C_m;
    % Sodium
    delta_m_gate = gate_differential(time_step, m_gate, alpha(V_m, 0.182, -35, 9), beta(V_m,
0.124, -35, 9));
    delta_h_gate = gate_differential(time_step, h_gate, alpha(V_m, 0.024, -50, 5), beta(V_m,
0.0091, -75, 5));
    % Potassium
    delta_n_gate = gate_differential(time_step, n_gate, alpha(V_m, 0.02, 20, 9), beta(V_m,
0.002, 20, 9));

    % Update the actual values
    V_m = V_m + delta_V_m;
    m_gate = m_gate + delta_m_gate;
    h_gate = h_gate + delta_h_gate;
    n_gate = n_gate + delta_n_gate;

    % Put into vectors for plotting purposes
    Voltage_Vector(t) = V_m;
    M_Gate_Vector(t) = m_gate;
    H_Gate_Vector(t) = h_gate;
    N_Gate_Vector(t) = n_gate;
end

figure(1)
plot(Voltage_Vector)
figure(2)
hold on
plot(M_Gate_Vector)
% figure(3)
plot(H_Gate_Vector)
% figure(4)
plot(N_Gate_Vector)

% gate_differential.m
function diff = gate_differential(timestep, gate_prob, alpha, beta)
diff = timestep * (alpha * (1 - gate_prob) - (beta * gate_prob));
end
```

```
% beta.m
function beta_out = beta(V_m, A, V_half, k)
beta_out = (-A * (V_m - V_half)) / (1 - exp((V_m - V_half) / k));
end

% alpha.m
function alpha_out = alpha(V_m, A, V_half, k)
alpha_out = (A * (V_m - V_half)) / (1 - exp((-(V_m - V_half) / k)));
end

% gate_open_probability.m
function gate_prob = gate_open_probability(V_rest, A_alpha, V_half_alpha, k_alpha, A_beta,
V_half_beta, k_beta)
gate_prob = alpha(V_rest, A_alpha, V_half_alpha, k_alpha) / (alpha(V_rest, A_alpha,
V_half_alpha, k_alpha) + beta(V_rest, A_beta, V_half_beta, k_beta));
end
```

Code Utilized for NEURON Simulations - Part 1

```
// Kushal Jaligama
// BIOMEDE 517 - Neural Engineering
// Lab 2, Part 1
// Creating a single-compartment neuron model

create soma // Make a soma
access soma // Set soma as default object for editing
nseg = 1 // Soma modeled using just one segment
pt3dadd(0, 0, 0, 18.8) // Set start of soma at coordinate (0, 0, 0) with diameter 18.8 um
pt3dadd(18.8, 0, 0, 18.8) // Set end of soma at coordinwate (18.8, 0, 0)
Ra = 123.0 // Resistivity in ohm-cm (cm!!!)
insert hh // Add default Hodgkin-Huxley

objectvar mystim // This is a variable name
soma mystim = new IClamp(0.5) // Put halfway along this soma
                    // Can also VClamp
mystim.del = 100 // ms, Delay until stim turns on
mystim.dur = 100 // ms, How long stim stays on
mystim.amp = 0.1 // nA, from inside to outside membrane
tstop = 300 // ms, defaults to 5
```

Code Utilized for NEURON Simulations - Part 2

```
// Kushal Jaligama
// BIOMEDE 517 - Neural Engineering
// Lab 2, Part 2
// Multiple axons, multiple compartments

// First Axon
numParts = 1000
create axon[numParts]
for (i = 0; i < numParts; i = i + 1) {
        axon[i] {
                // define properties
                L = 100
                diam = 10
                insert hh
        }
}

for (i = 0; i < numParts - 1; i = i + 1) {
        connect axon[i](1), axon[i + 1](0)
}

objectvar mystim
// Technically have default section as axon, so no need for "axon mystyim = new...",
// can be "mystim = new..."
axon mystim = new IClamp(0) // Put at the start of the axon (start of first compartment)
                // Can also VClamp
mystim.del = 100 // ms, Delay until stim turns on
mystim.dur = 100 // ms, How long stim stays on
mystim.amp = 4 // nA, from inside to outside membrane

// // Second Axon

create axon2[numParts]
for (i = 0; i < numParts; i = i + 1) {
        axon2[i] {
                // define properties
                diam = 20
                L = 100
```

```
                insert hh
        }
}

for (i = 0; i < numParts - 1; i = i + 1) {
        connect axon2[i](1), axon2[i + 1](0)
}

objectvar mystim
axon2 mystim = new IClamp(0) // Put at the start of the axon (start of first compartment)
                // Can also VClamp
mystim.del = 500 // ms, Delay until stim turns on
mystim.dur = 100 // ms, How long stim stays on
mystim.amp = 10 // nA, from inside to outside membrane
// Graph will stop printing at this time step:
tstop = 1000 // ms, defaults to 5
```

Code Utilized for NEURON Simulations - Part 4

```
// Kushal Jaligama
// BIOMEDE 517 - Neural Engineering
// Lab 2, Part 4
// Custom simplified neuron model

// Use this to scale the model
scale_factor = 0.5

// Define parameters for components here
soma_length = 24  * scale_factor // um
soma_diameter = 21 * scale_factor // um
soma_nseg = 100

dendrite_length = 50 * scale_factor // um
dendrite_diameter = 12 * scale_factor // um
dendrite_nseg = 222

non_myelinated_axon_length = 16 * scale_factor // um
non_myelinated_axon_diameter = 1 * scale_factor // um
non_myelinated_axon_nseg = 100
```

```
myelinated_axon_length = 300 * scale_factor // um
myelinated_axon_diameter = 1 * scale_factor // um
myelinated_axon_nseg = 100 // um

axon_hillock_length = 9 * scale_factor // um
axon_hillock_nseg = 9

create soma, dendrite, axon_hillock[axon_hillock_nseg], non_myelinated_axon,
myelinated_axon
access soma

soma {
  nseg = soma_nseg
  pt3dadd(0, 0, 0, soma_diameter)
  pt3dadd(soma_length, 0, 0, soma_diameter)
  // Parameters from lab manual
  insert pas
  Ra=150
  cm=0.75
  g_pas=1/30000
  e_pas=-70
  insert na
  gbar_na=20
  insert kv
  gbar_kv=200
  insert km
  gbar_km=0.1
  insert kca
  gbar_kca=3
  insert ca
  gbar_ca=0.3
  insert cad
}

access dendrite
dendrite {
  nseg = dendrite_nseg
  pt3dadd(-dendrite_length, 0, 0, dendrite_diameter)
  pt3dadd(0, 0, 0, dendrite_diameter)
```

```
  // Parameters from lab manual
  insert pas
  Ra = 150
  cm = 0.75
  g_pas = 1/30000
  e_pas = -70
  insert na
  gbar_na = 20
  insert km
  gbar_km  = 0.1
  insert kca
  gbar_kca = 3
  insert ca
  gbar_ca = 0.3
  insert cad
}

access non_myelinated_axon
non_myelinated_axon {
  nseg = 100
  pt3dadd(soma_length + axon_hillock_length, 0, 0, non_myelinated_axon_diameter)
  pt3dadd(soma_length + axon_hillock_length + non_myelinated_axon_length, 0, 0,
non_myelinated_axon_diameter)
  // Parameters from lab manual
  insert pas
  Ra=150
  cm=0.75
  g_pas=1/30000
  e_pas=-70
  insert na
  gbar_na=30000
  insert kv
  gbar_kv=2000
}

access myelinated_axon
myelinated_axon {
  nseg = 100
```

```
  pt3dadd(soma_length + axon_hillock_length + non_myelinated_axon_length, 0, 0,
myelinated_axon_diameter)
  pt3dadd(soma_length + axon_hillock_length + non_myelinated_axon_length +
myelinated_axon_length, 0, 0, myelinated_axon_diameter)
  // Parameters from lab manual
  insert pas
  Ra=150
  cm=0.04
  g_pas=1/30000
  e_pas=-70
  insert na
  gbar_na=20
}

access axon_hillock

for (i = 0; i < axon_hillock_nseg; i = i + 1) {
   axon_hillock[i] {
      nseg = 1
      pt3dadd(soma_length + i, 0, 0, soma_diameter - i * ((soma_diameter -
non_myelinated_axon_diameter) / axon_hillock_nseg))
      insert pas
      Ra=150
      cm=0.75
      g_pas=1/30000
      e_pas=-70
      insert na
      gbar_na=30000
      insert kv
      gbar_kv=2000
   }
}

// Connect the compartments of the neuron together
connect dendrite(1), soma(0)
connect soma(1), axon_hillock[0](0)

for (i = 0; i < axon_hillock_nseg - 1; i = i + 1) {
   connect axon_hillock[i](1), axon_hillock[i + 1](0)
```

```
}

connect axon_hillock[axon_hillock_nseg - 1](1), non_myelinated_axon(0)
connect non_myelinated_axon(1), myelinated_axon(0)

// Apply the current stimulation

objectvar mystim
axon_hillock[4] mystim = new IClamp(0.5) // Place it at the midpoint of the 4th compartment
mystim.del = 5 // ms
mystim.dur = 5 // ms
mystim.amp = 0.62 // nA

// Graphing variables
tstop = 30 // ms

// ------------------------------------------------------------
// Exports Currents and Geometry
// ------------------------------------------------------------
forall {
  insert extracellular
  insert xtra
  }

load_file("interpxyz.hoc")    // only interpolates sections that have extracellular
load_file("setpointers.hoc")  // automatically calls grindaway() in interpxyz.hoc

// RECORD SECTION POSITIONS
objref f2
f2=new File()
f2.wopen("coordinates") // coordinate file name

f2.printf("name\tx\ty\tz\n")

forall{
  for (x) if(x!=0 && x!=1){
    f2.printf("%s(%g)\t%f\t%f\t%f\n", secname(), x, x_xtra(x), y_xtra(x), z_xtra(x))
  }
}
```

```
f2.close()

// RECORD MEMBRANE CURRENTS
objref f1
f1 = new File()
f1.wopen("currents") // current file name

finitialize()
fcurrent()

// write 'time' and section names (ms)
f1.printf("time (ms)\t")

forall {
  for (x) if(x!=0 && x!=1){
    f1.printf("%s(%g)\t",secname(),x)
  }
}

f1.printf("\n")

// write x values (um)
f1.printf("-1\t")
forall {
  for (x) if(x!=0 && x!=1){
    f1.printf("%f\t",x_xtra(x))
  }
}

f1.printf("\n")

// write y values (um)
f1.printf("-1\t")
forall {
  for (x) if(x!=0 && x!=1){
    f1.printf("%f\t",y_xtra(x))
  }
}
```

```
f1.printf("\n")

// write z values (um)
f1.printf("-1\t")
forall {
  for (x) if(x!=0 && x!=1){
    f1.printf("%f\t",z_xtra(x))
  }
}

f1.printf("\n")

// write currents (in Amps)
// note i_membrane is in mA/cm2 & area is in um2
proc advance() {
  f1.printf("%f\t", t)

  forall {
     for(x) if (x!=0 && x!=1){
       f1.printf("%e\t", i_membrane(x)*area(x)*(1e-11))  //current in Amps
     }
    }
    f1.printf("\n")
  fadvance()
}
run()
```

Code Utilized for Current Analysis - Lab 2 Part 4

```
% Lab 2 Part 4 Model Graphs
twelve = importdata('12um/currents')
twentyfour = importdata('24um/currents')
fourtyeight = importdata('48um/currents')

timeaxis = twelve.data(100:1203, 1);
hillock12 = twelve.data(100:1203, 205);
hillock24 = twentyfour.data(100:1203, 205);
hillock48 = fourtyeight.data(100:1203, 205);
```

```
plot(timeaxis, hillock12)
hold on
plot(timeaxis, hillock24)
plot(timeaxis, hillock48)
```