BIOMEDE 517

Name: Diksha Anoop Kumar Zutshi
UMID:34050139
Date: 01/26/2024

Introduction:

The field of computational neuroscience has increasingly utilized detailed single neuron simulations to gain insights into the intricate workings of neuronal dynamics. Leveraging parameters derived from past experimental studies, such as ion concentrations, membrane resistances, and channel conductances, these simulations allow researchers to dissect the mechanisms underlying neuronal excitability and action potential propagation. This lab report will showcase the application of the Hodgkin-Huxley model to simulate action potentials in neurons of varying sizes and channel densities. These simulations not only reinforce our understanding of fundamental neuronal properties but also provide a framework for investigating pathological conditions and potential therapeutic targets in neurophysiology. This lab report aims to encapsulate the multifaceted utility of single neuron simulations in unraveling the nuances of neuronal function and behavior.

Methods:

Lab-1:

1. Initial Conditions:
   Here, we calculate all the initial state of the cell, when it is at rest.The Nernst potential equation is used to calculate the equilibrium potential (also known as the Nernst potential, E_ion) for a particular ion across a membrane.The resting membrane potential of a cell is the voltage (or electrical potential) difference across the cell membrane when the cell is in an unstimulated state. The resting point for dynamic channel variables in the Hodgkin-Huxley model refers to the steady-state values and the rate constants for the gating variables of the ion channels. The global values have been referred from Mainen.

Ion Concentrations:

Ion species	Intracellular concentration (mM)	Extracellular concentration (mM)
K^+	155	4
Na^+	12	145
Ca^{2+}	10^{-4}	1.5
Cl^-	4	120

Equations to calculate Nerst Potential:

$$E_{ion} = \frac{k_B T}{ze} \ln \frac{Co}{Ci} = \frac{25}{z} \ln \frac{Co}{Ci} (mV)$$

Membrane potential equation:

$$V_m = 25 \ln \frac{P_K[K_o] + P_{Na}[Na_o] + P_{Cl}[Cl_i]}{P_K[K_i] + P_{Na}[Na_i] + P_{Cl}[Cl_o]}$$

$P_K = 1.0 \qquad P_{Na} = 0.04 \qquad P_{Cl} = 0.45$

Resting point for dynamic channel variables for Sodium,Potassium:

$$\alpha(V) = \frac{A(V_m - V_{1/2})}{1 - e^{-(V_m - V_{1/2})/k}} \qquad \beta(V) = \frac{-A(V_m - V_{1/2})}{1 - e^{(V_m - V_{1/2})/k}} \qquad m_0 = m_\infty = \frac{\alpha_m(V_{rest})}{\alpha_m(V_{rest}) + \beta_m(V_{rest})}$$

2. Time Step and Equations: Neuron dynamics here has been simulated using the Hodgkin-Huxley model. The simulation runs over the time span (from 0 to 5 milliseconds), with a fine time step resolution of 0.01 milliseconds. The code defines a set of constants such as, membrane capacitance, maximum conductances for sodium and potassium ions, leak conductance, reversal potentials, and an externally applied current. The timing for when the current is turned on and off is also specified.The simulation pre allocates vectors to store the membrane potential (Vm_) and the gating variables. In the main simulation loop, the code calculates the rate of change for the gating variables using the forward Euler method. The total ionic currents (INa, IK, IL) through the sodium, potassium, and leak channels are calculated based on the gating

variables and their respective conductances. An external current (Iext) is applied for a brief period as specified by t_on1 and t_off1. The change in membrane potential (dV) is computed based on the total current and the membrane capacitance.

3. Critical length of myelinated axon:
   Voltage decays exponentially as a function of length of the axon.To determine how far does the voltage (V0) pass in the axon passively:

```
lambda=sqrt(rm/ri) cm     ;Length constant
rm = 40000; %ohm-cm2  ;Membrane resistance
ri = 200; %ohm-cm          ;Intracellular / axial resistance
V(x)=V(0)*exp(-(x/lambda));  V(0): Peak Voltage, V(x):Threshold Voltage
x=-lambda*real(log(Vthreshold/Vpeak)) cm
```

Lab 2:

1. Assumptions of the model:
   Hodgkin Huxley Properties:It represents the neuron's membrane as an electrical circuit with capacitors and voltage-gated ion channels.The model assumes that the ion channels open and close according to the membrane potential, which allows for the generation and propagation of action potentials.
   Passive Membrane Properties:The passive properties model, often referred to as a 'leak' channel, simulates the non-specific leakage of ions through the neuron's membrane. This property is used for simulating the resting membrane potential and the passive spread of electrical signals (like post-synaptic potentials).
   Extracellular Properties:This model allows for the simulation of extracellular potentials and fields. Often used in the study of extracellular recordings and in multi-compartment neuron models where interactions between cells and its external environment is relevant.
2. Parameters of the Axon Model:
   After adding Hodgkin Huxley, Passive Membrane & Extracellular properties to the axon, the default model parameter are given by figures 1,2,3 (appendix):
3. Structure of the Simplified Neuron and Experiments:
   The dendrite, soma, axon hillock, and non-myelinated axon are equipped with both active (Hodgkin-Huxley model, h.hh) and passive (leak channels, h.pas) membrane properties, as well as extracellular properties (h.extracellular). The myelinated axon, however, only has passive and extracellular properties, reflecting its role in rapid signal transmission rather than action potential generation. The axon hillock is uniquely defined with a tapering diameter, creating a funnel-like shape transitioning from the soma to the non-myelinated axon. Finally, these sections are connected in a sequential order reflecting the anatomical structure of a neuron: the dendrite connects to the soma, which connects to the axon hillock, leading to the non-myelinated axon and finally to the myelinated axon.
   Here, we decreased the capacitance of the myelinated axon to make sure of the high electrical resistance of the myelin layer, as it's meant to be. We added stimulus to make the cell barely fire and recorded the response in the middle of each of the sections. We increased the ion channel density and observed a negative current:which signifies an increased inflow of Na+ ions & outwards flow of K+ ions, which can potentially lead to rapid depolarization.

Results:

Lab-1:

Resting State Values:

From Table 1 (Appendix), activation gating variable of Na^+ (m), it suggests that at the resting membrane potential, there is a low probability of sodium channels being open. Inactivation gating variable of Na^+(h), it indicates that a substantial number of the sodium channel inactivation gates are open at rest. Activation gating variable of K^+(n) has its value pretty low, which would suggest that a very low number of potassium channels are open at this particular resting potential.

Time Course of Variables:

Figure 4 (Appendix), describes the behavior of membrane potential over time, giving us the peak voltage at 60.57mV. The image also describes the changing behavior of gating variables over time. Note that 'm' and 'h' gates act opposite to each other.

```
fprintf('Peak Voltage: %f\n', max(peaks));
Peak Voltage: 60.570956
```

How far does the action potential propagate down the axon?

Critical Threshold Voltage: -56.9mV
Peak Voltage: 60.57 mV

Figure 5 (Appendix) showcases the critical threshold voltage and peak voltage of the action potential. When a depolarization event raises the membrane potential above this critical threshold voltage, voltage-gated sodium channels open, leading to a rapid influx of Na+ ions and the subsequent rise in membrane potential to a peak.

Critical length of the axon (cm)

```
>> lengthconstant
Critical Length of Myelinated Axon (): 0.884169
```

Lab-2:

Effects of changing Diameter:

Axon 1: Diameter 10um, 5mm long, 1000 compartments.;Applied Current = 30nA
Axon 2: Diameter 20um, 5mm long, 1000 compartments.;Applied Current = 70nA

Action Potentials:

From figure 6 & 7 (appendix) we can observe that much lower applied current is required to make the Axon 1 fire, in comparison to Axon 2. Action Potential initiates at around 2ms in Axon 1 and at around 4ms in Axon 2. The higher surface area to volume ratio and lower capacitance allows Axon1 to reach the action potential threshold more quickly and with less current.

Length Constants:

```
##length constant
import math
Rm=40000 ##passive membrane resistance for calculation
Ri=200 ##internal resistance
lambda1=math.sqrt((((soma1.diam)/2)*Rm)/(2*Ri))
print('Length Constant Axon1:',lambda1)
```

```
Length Constant Axon1: 22.360679774997898
```

```
##length constant
Rm=40000 ##passive membrane resistance for calculation
Ri=200 ##internal resistance
lambda2=math.sqrt((((soma2.diam)/2)*Rm)/(2*Ri))
print('Length Constant Axon2:',lambda2)
```

```
Length Constant Axon2: 31.622776601683793
```

Diameter is directly proportional to the length constant (lambda), hence, Lambda2>Lambda.

Varying Ion Channel Density:From Figure 8,9,& 10, we can observe that the ion channel density has been increased for the unmyelinated axon; it has the largest negative peak current amongst all other sections & higher than unmyelinated axon without increased ion channel density.With increased Na^+ channel density, the peak inward current (negative current) during the initial phase of the action potential is larger and more rapid, reflecting the greater influx of Na^+ ions.

Discussions:

Lab1:

At rest fewer activation gates are open in comparison to the inactivation gates for Na+.Very few K+activation gates 'n' are open during rest, because K+ions act in response to depolarization. Activation and inactivation gates of Na+ ions, m & h respectively, have an opposing action to each other, which allows the sodium channels to participate in the generation of action potentials with a distinct rising and falling phase. Initially, upon a sufficient depolarizing stimulus, 'm' increases rapidly allowing Na+ influx, leading to the rapid upstroke of the action potential. As the membrane potential reaches its peak, 'h' starts to decrease, inactivating the sodium channels and contributing to the repolarization of the membrane. The distance that the electrical potential can travel before diminishing is the Critical Length of the axon. This depends on the length constant lambda, which inturn is governed by the internal and the membrane resistance. A higher length constant means that voltage changes can propagate further along the axon. Single neuron models are the building blocks of larger neural networks. Understanding single neuron dynamics is essential for simulating more complex brain functions and behaviors.Insights from neuron models can be applied to improve algorithms in AI and machine learning, particularly in developing more efficient and biologically inspired neural network architectures.

Lab2:

Larger axons propagate action potentials faster due to lower internal resistance, the initiation of action potentials can be slower because it takes longer to depolarize the membrane to the threshold, as it requires more ionic current to change the membrane potential. It's easier to target with microelectrodes for intracellular recording in a larger axon. The larger size offers a bigger target, which is particularly important when using techniques like patch-clamping. Increased Na+ channel density, results in faster Na^+ influx, leading to quicker depolarization. Increased K+ channel density, results in enhanced efflux of K+ ions, during repolarization and hyperpolarization. This results in rapid depolarization, that is, decreased threshold for action potential. Shortening the action potential, as there is a quicker repolarization phase. By simulating how changes in axon size and ion channel density affect neuronal activity, these models can help in understanding the pathophysiology of various neurological disorders, such as epilepsy, where ion channel dysfunctions play a crucial role.

ModelDB:Bursting in dopamine neurons (Li YX et al 1996):The model is a computational framework focused on understanding burst firing in dopaminergic neurons of the substantia nigra pars compacta, a key region implicated in Parkinson's disease. It uniquely incorporates minimal ion currents, including a Na^+/K^+ ATPase pump sensitive to ouabain, to elucidate the mechanism of burst generation and interburst hyperpolarization, particularly in response to NMDA stimulation. This theoretical approach allows for testing hypotheses about neuronal behavior that are difficult to investigate experimentally.The model can be used to predict neuronal responses to new pharmacological agents, aiding in the development of medications targeting dopaminergic neurons.Incorporating these neuron models into larger network simulations can enhance our understanding of the substantia nigra's role in motor control and its dysfunction in neurological disorders.

Appendix:

Figures & Tables:

Lab 1:

Figure 1


```
{'point_processes': {},
 'density_mechs': {'pas': {'g': [0.001], 'e': [-70.0], 'i': [0.0]},
  'extracellular': {'xraxial': [[1000000000.0, 1000000000.0]],
   'xg': [[1000000000.0, 1000000000.0]],
   'xc': [[0.0, 0.0]],
   'e': [0.0],
   'i_membrane': [0.0],
   'vext': [[0.0, 0.0]]},
  'hh': {'gnabar': [0.12],
   'gkbar': [0.036],
   'gl': [0.0003],
   'el': [-54.3],
   'gna': [0.0],
   'gk': [0.0],
   'il': [0.0],
```


Figure 2


```
 'minf': [0.0],
 'hinf': [0.0],
 'ninf': [0.0],
 'mtau': [0.0],
 'htau': [0.0],
 'ntau': [0.0],
 'm': [0.0],
 'h': [0.0],
 'n': [0.0]}},
ions': {'na': {'ena': [50.0],
 'nai': [10.0],
 'nao': [140.0],
 'ina': [0.0],
 'dina_dv_': [0.0]},
'k': {'ek': [-77.0],
 'ki': [54.4],
 'ko': [2.5],
 'ik': [0.0],
 'dik_dv_': [0.0]}},
morphology': {'L': 100.0,
 'diam': [500.0],
```


Figure 3


```
 'pts3d': [],
 'parent': None,
 'trueparent': None},
'nseg': 1,
'Ra': 35.4,
'cm': [1.0],
'regions': set(),
'species': set(),
'name': 'axon',
'hoc_internal_name': '__nrnsec_0x58331c00f2f0',
'cell': None}
```


Table 1

Resting State Variables	Values
m0	0.02
h0	0.1529
n0	3.49*10^-4
Vm	-72.34

Figure 4

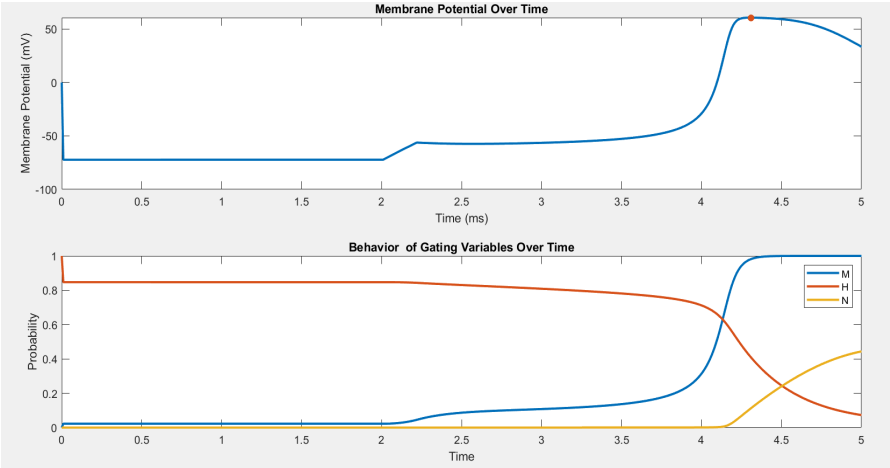
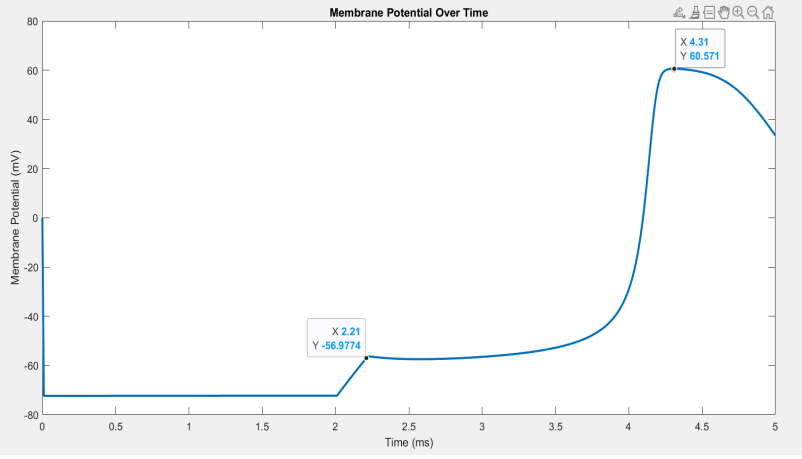


Figure 5



Lab 2:

Figure 6

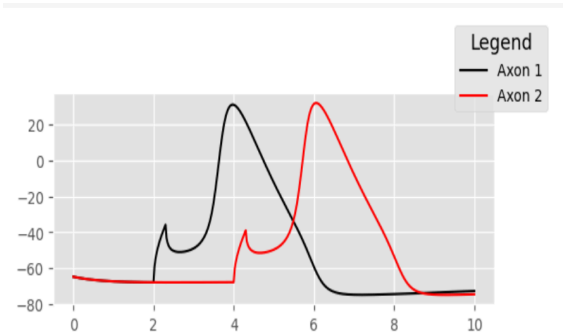


Figure 7

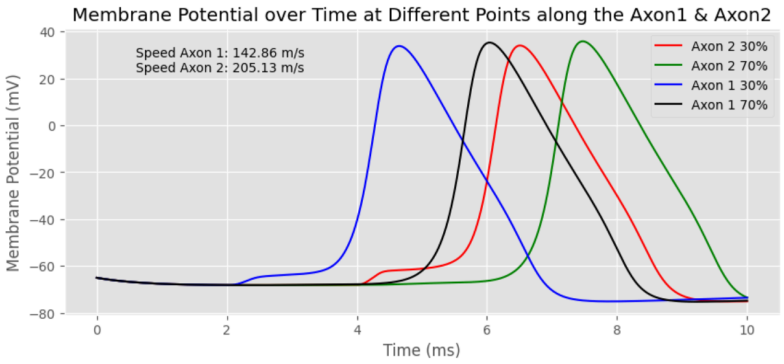
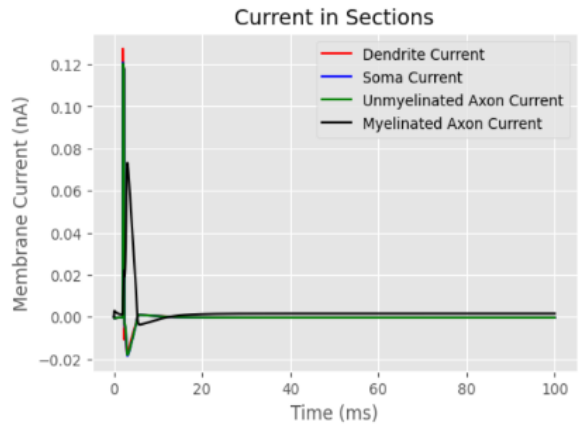


Figure 8

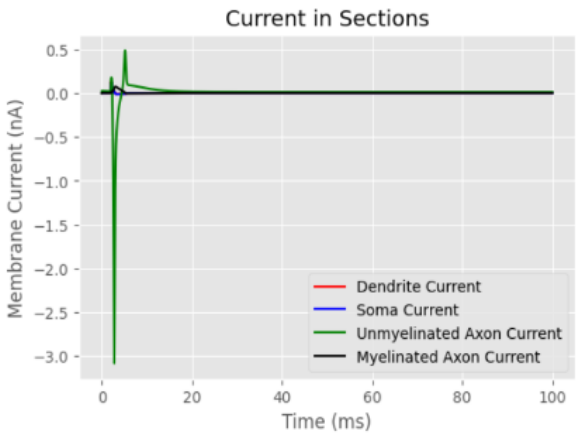
Without increasing the ion channel density




```
Difference in peak current between non-myelinated axon and soma: 0.00038452117747186584
```

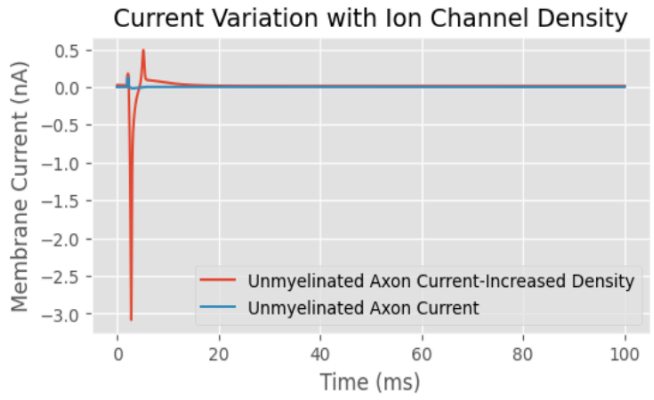
Figure 9

After increasing the ion channel density (10x)




```
Difference in peak current between non-myelinated axon and soma: -3.0699454230321335
```

Figure 10




```
Difference in peak current between non-myelinated axons: -3.0634852646871886
```

Code:

Lab1:

Initial Conditions:

```
%%% INITIAL CONDITIONS%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% NERNST POTENTIAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Na
z1=1;
Co1=145;
Ci1=12;
%K
z2=1;
Co2=4;
Ci2=155;
%Cl
z3=-1;
Co3=120;
Ci3=4;
%%reversal potentials
ENa=25/z1*log(Co1/Ci1);
Ek=25/z2*log(Co2/Ci2);
Ecl=25/z3*log(Co3/Ci3);
%%%%%%%%%%%%%%%%%%%% MEMBRANE RESTING POTENTIAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%ION PERMEABILITIES
Pna=0.04;
Pk=1;
Pcl=0.45;
Vm=25*log((Pna*Co1+Pk*Co2+Pcl*Ci3)/(Pna*Ci1+Pk*Ci2+Pcl*Co3));
%%%%%%%%%%%%% STARTING/RESTING POINT FOR DYNAMIC CHANNEL VARIABLES %%%%%%%%
%%Vm @rest = m0,h0,n0
%%sodium gating variable = m
m_alpha_A=0.182;
m_beta_A=0.124;
m_alpha_Vhalf=-35;
m_beta_Vhalf=-35;
m_alpha_k=9;
m_beta_k=9;
m_alpha=m_alpha_A*(Vm-m_alpha_Vhalf)/(1-exp(-(Vm-m_alpha_Vhalf)/m_alpha_k));
m_beta=-m_beta_A*(Vm-m_beta_Vhalf)/(1-exp((Vm-m_beta_Vhalf)/m_beta_k));
m0=m_alpha/(m_alpha+m_beta);
%%sodium gating variable = h
h_alpha_A=0.024;
h_beta_A=0.0091;
h_alpha_Vhalf=-50;
h_beta_Vhalf=-75;
h_alpha_k=5;
h_beta_k=5;
%h_Vhalf=-65;
%h_k=6.2;
%h0=1/(1+((exp(Vm-h_Vhalf))/h_k));
%h0=1-h0;
h_alpha=h_alpha_A*(Vm-h_alpha_Vhalf)/(1-exp(-(Vm-h_alpha_Vhalf)/h_alpha_k));
h_beta=-h_beta_A*(Vm-h_beta_Vhalf)/(1-exp((Vm-h_beta_Vhalf)/h_beta_k));
h0=h_alpha/(h_alpha+h_beta);
%%potassium gating variable = n
n_alpha_A=0.02;
n_beta_A=0.002;
n_alpha_Vhalf=20;
n_beta_Vhalf=20;
n_alpha_k=9;
n_beta_k=9;
n_alpha=n_alpha_A*(Vm-n_alpha_Vhalf)/(1-exp(-(Vm-n_alpha_Vhalf)/n_alpha_k));
n_beta=-n_beta_A*(Vm-n_beta_Vhalf)/(1-exp((Vm-n_beta_Vhalf)/n_beta_k));
n0=n_alpha/(n_alpha+n_beta);
fprintf('m0: %f\n', m0);
```

```
fprintf('h0: %f\n', h0);
fprintf('n0: %f\n', n0);
```

Main Time Step:

```
% Simulation parameters
tspan = [0 5]; % Total simulation time in ms
dt = 0.01; % Time step in ms
time = tspan(1):dt:tspan(2); % Time vector
% Constants
Cm = 1; % Membrane capacitance, uF/cm^2
gNa_ = 100; % Sodium (Na) maximum conductances, mS/cm^2
gK_ = 50; % Potassium (K) maximum conductances, mS/cm^2
gL = 0.5; % Leak maximum conductances, mS/cm^2
Eleak=-72.5;%mV
Iapp = 80; % Applied current in uA/cm^2
t_on1 = 2; % time current pulse turns on (ms)
t_off1 = 2.2; % time current pulse turns off(ms)
%preallocation
Vm_ = zeros(1, length(time));
M = zeros(1, length(time));%%gating variables
H = zeros(1, length(time));
N = zeros(1, length(time));
% M(1)=m0; H(1)=h0; N(1)=n0;
%%main
for i= 1:length(time)-1

   m_alpha=m_alpha_A*(Vm-m_alpha_Vhalf)/(1-exp(-(Vm-m_alpha_Vhalf)/m_alpha_k));
   m_beta=-m_beta_A*(Vm-m_beta_Vhalf)/(1-exp((Vm-m_beta_Vhalf)/m_beta_k));
   h_alpha=h_alpha_A*(Vm-h_alpha_Vhalf)/(1-exp(-(Vm-h_alpha_Vhalf)/h_alpha_k));
   h_beta=-h_beta_A*(Vm-h_beta_Vhalf)/(1-exp((Vm-h_beta_Vhalf)/h_beta_k));
   n_alpha=n_alpha_A*(Vm-n_alpha_Vhalf)/(1-exp(-(Vm-n_alpha_Vhalf)/n_alpha_k));
   n_beta=-n_beta_A*(Vm-n_beta_Vhalf)/(1-exp((Vm-n_beta_Vhalf)/n_beta_k));
   dm = ((m_alpha*(1-m0)) - (m_beta*m0))*dt;
   dh = ((h_alpha*(1-h0)) - (h_beta*h0))*dt;
   dn = ((n_alpha*(1-n0)) - (n_beta*n0))*dt;
   %Current
   INa = gNa_*m0^3*(1-h0)*(Vm-ENa);%%%%%%
   IK = gK_*n0^4*(Vm-Ek);
   IL = gL*(Vm-Eleak);

   Iext = Iapp * (time(i) >= t_on1 && time(i) <= t_off1);
   dV = (Iext - (INa+IK+IL)) / Cm;

   I_total=Cm*(dV/dt)+INa+IK+IL+Iapp;

   % Store results
   Vm_(i+1) = Vm;
   M(i+1) = m0;
   H(i+1) = h0;
   N(i+1) = n0;
   %Update
   m0 = m0 + dm;
   h0 = h0 + dh;
   n0 = n0 + dn;
   Vm = Vm + dV*dt;
end
[peaks, indxs]=findpeaks(Vm_,'MINPEAKHEIGHT',-10);
spiketimes=time(indxs);
% Plot results
subplot(2,1,1)
plot(time,Vm_,spiketimes,peaks,'*','LineWidth',2)
title('Membrane Potential Over Time');
xlabel('Time (ms)');
ylabel('Membrane Potential (mV)');
hold on
subplot(2,1,2)
```

```
%%plots to capture behavior of gating variables
plot(time,M,time,1-H,time,N,'LineWidth',2)
title('Behavior  of Gating Variables Over Time');
xlabel('Time')
ylabel('Probability')
legend('M','H','N')
fprintf('Peak Voltage: %f\n', max(peaks));
% plot(Vm_,M,Vm_,1-H,Vm_,N,'LineWidth',2)
% xlabel('Voltage')
% ylabel('Probability')
% legend('M','H','N')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Critical Axon Length:

```
%intracellular / axial resistance (ri)
%membrane resistance(rm)
%length constant (lambda)
rm = 40000; %ohm-cm2
ri = 200; %ohm-cm
lambda=sqrt(rm/ri);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%axon length
Vthresh=-56.9;
Vpeak=max(peaks);  %%peak potential
%Vthresh=Vpeak*exp(-x/lambda)
%x=-lambda*log(Vthresh/Vpeak)   %% how far down does the potential propagate
x=-lambda*real(log(Vthresh/Vpeak));
fprintf('Critical Length of Myelinated Axon (): %f\n', x);
```

Lab2:

Varying Exon Diameter between 2 Axons:

Axon 1:

```
##AXON 1: dia 10um, 5mm long, 1000 compartments
soma1 = h.Section(name="soma1")
soma1.L=5*mm
soma1.diam=10*um
soma1.nseg=1000
soma1.insert(h.hh)  ##hodgkin huxley model
soma1.insert(h.pas)  ##passive memebrane properties
#inserting current
ic0=h.IClamp(soma1(0.1))
ic0.delay=2
ic0.dur=0.3
ic0.amp=30
#Recording
t0=h.Vector().record(h._ref_t)
v0=h.Vector().record(soma1(0.1)._ref_v)
#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)

##length constant
import math
Rm=40000 ##passive membrane resistance for calculation
Ri=200 ##internal resistance
lambda1=math.sqrt((((soma1.diam)/2)*Rm)/(2*Ri))
print('Length Constant Axon1:',lambda1)
```

Axon 2:

```
##AXON 2:  dia 20um, 5mm long, 1000 compartments
#Creating a cell
soma2 = h.Section(name="soma2")
#properties
soma2.L=5*mm
soma2.diam=20*um
soma2.nseg=1000
soma2.insert(h.hh)  ##hodgkin huxley model
soma2.insert(h.pas)  ##passive membrane properties
#current
ic1=h.IClamp(soma2(0.1))
ic1.delay=4
ic1.dur=0.3
ic1.amp=70
##recording
t1=h.Vector().record(h._ref_t)
v1=h.Vector().record(soma2(0.1)._ref_v)
#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)

##length constant
Rm=40000 ##passive membrane resistance for calculation
Ri=200 ##internal resistance
lambda2=math.sqrt((((soma2.diam)/2)*Rm)/(2*Ri))
print('Length Constant Axon2:',lambda2)
```

Plotting together:

```
from matplotlib import pyplot
import matplotlib.style as style
style.use('ggplot')

fig = pyplot.figure(figsize=(6,5))
ax1 = fig.add_subplot(2,1,1)
axon1 = ax1.plot(t0, v0, color='black', label='Axon 1')
axon2 = ax1.plot(t1, v1, color='red', label='Axon 2')
fig.legend(title='Legend', title_fontsize='13', loc='upper right')
fig.show()
```

Speed of Action Potential:

Axon1:

```
##Speed of action potential propogation
##AXON 1
recording_point_1 = h.Vector()  # For the first point on the axon
recording_point_2 = h.Vector()  # For the second point on the axon
t0=h.Vector().record(h._ref_t)  # For recording time
# Attach the vectors to specific points on the axon
# Let's say we record from 0.3 (30%) and 0.7 (70%) along the axon's length
recording_point_1.record(soma1(0.3)._ref_v)
recording_point_2.record(soma1(0.7)._ref_v)

#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
```

```
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)
###SPEED
# Find the indices of the peak action potential (e.g., the first occurrence of the maximum value)
recording_point_1=list(recording_point_1)
recording_point_2=list(recording_point_2)
peak_index_axon1_30 = recording_point_1.index(max(recording_point_1))
peak_index_axon1_70 = recording_point_2.index(max(recording_point_2))
# Calculate the time difference
time_diff_axon1 = t0[peak_index_axon1_70] - t0[peak_index_axon1_30]
##distance= 40%: 0.04 , 5mm(total length),
distance= 0.04*soma1.L
speed_axon1 = distance / time_diff_axon1
```

Axon 2:

```
##Speed of action potential propogation
##AXON 2

recording_point_11 = h.Vector()  # For the first point on the axon
recording_point_22 = h.Vector()  # For the second point on the axon
t1=h.Vector().record(h._ref_t)  # For recording time
# Attach the vectors to specific points on the axon
# Let's say we record from 0.3 (10%) and 0.7 (70%) along the axon's length
recording_point_11.record(soma2(0.3)._ref_v)
recording_point_22.record(soma2(0.7)._ref_v)

#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)
###SPEED
# Find the indices of the peak action potential (e.g., the first occurrence of the maximum value)
recording_point_11=list(recording_point_11)
recording_point_22=list(recording_point_22)
peak_index_axon2_30 = recording_point_11.index(max(recording_point_11))
peak_index_axon2_70 = recording_point_22.index(max(recording_point_22))
# Calculate the time difference
time_diff_axon2 = t1[peak_index_axon2_70] - t1[peak_index_axon2_30]
##distance= 60th: 0.04 , 5mm(total length),
distance1= 0.04*soma2.L
speed_axon2 = distance1/time_diff_axon2
```

Plotting Action Potentials and Speeds:

```
plt.figure(figsize=(10, 4))
plt.plot(t1, recording_point_11, label='Axon 2 30%',color='red')
plt.plot(t1, recording_point_22, label='Axon 2 70%',color='green')
plt.plot(t0, recording_point_1, label='Axon 1 30%',color='blue')
plt.plot(t0, recording_point_2, label='Axon 1 70%', color='black')
plt.annotate(f'Speed Axon 1: {speed_axon1:.2f} m/s', xy=(0.1, 0.9), xycoords='axes fraction')
plt.annotate(f'Speed Axon 2: {speed_axon2:.2f} m/s', xy=(0.1, 0.85), xycoords='axes fraction')
plt.xlabel('Time (ms)')
plt.ylabel('Membrane Potential (mV)')
```

```
plt.title('Membrane Potential over Time at Different Points along the Axon1 & Axon2')
plt.legend()
plt.show()
```

Dendrite→Soma→Axon Hillock→Non-Myelinated Axon→Myelinated Axon

```
##Dendrite
dend = h.Section(name='dendrite')
dend.L=50*um
dend.diam=12*um
dend.nseg=222
dend.insert(h.hh)  ##hodgkin huxley model
dend.insert(h.pas)  ##passive memebrane properties
dend.insert(h.extracellular)  ##extracellular properties

##Soma
soma3 = h.Section(name='soma')
soma3.L=24*um
soma3.diam=21*um
soma3.nseg=100
soma3.insert(h.hh)  ##hodgkin huxley model
soma3.insert(h.pas)  ##passive memebrane properties
soma3.insert(h.extracellular)   ##extracellular properties

##Axon hilloc
hilloc = h.Section(name='axon hilloc')
hilloc.L=16*um
hilloc.nseg=9
##setting diameters
diameters = [21, 19, 16, 14, 11, 8, 5, 3, 1]
# Ensure that the length of the diameters list matches nseg
if len(diameters) != hilloc.nseg:
    raise ValueError("Number of diameters must match nseg")
# Assign the diameters to each segment
for seg, diam in zip(hilloc, diameters):
    seg.diam = diam * um
hilloc.insert(h.hh)  ##hodgkin huxley model
hilloc.insert(h.pas)  ##passive memebrane properties
hilloc.insert(h.extracellular)   ##extracellular properties


#Unmyelinated axon
unmyelin = h.Section(name='non myelinated axon')
unmyelin.L=16*um
unmyelin.diam=1*um
unmyelin.nseg=100
unmyelin.insert(h.hh)  ##hodgkin huxley model
unmyelin.insert(h.pas)  ##passive memebrane properties
unmyelin.insert(h.extracellular)   ##extracellular properties

#myelinated axon
myelin = h.Section(name='myelinated axon')
myelin.L=300*um
myelin.cm=0.04
myelin.diam=1*um
myelin.nseg=100
```

```
myelin.insert(h.pas)  ##passive memebrane properties
myelin.insert(h.extracellular)   ##extracellular

##building connection
dend.connect(soma3(0))
soma3.connect(hilloc(0))
hilloc.connect(unmyelin(0))
unmyelin.connect(myelin(0))
```

Adding Current at the most distant end of the Dendrite:

```
#current : Add a current stimulus to make the cell fire at the most distant end of the dendrite.
ic1=h.IClamp(dend(0.9))
ic1.delay=2
ic1.dur=0.3
ic1.amp=6
##recording
t1=h.Vector().record(h._ref_t)
v1=h.Vector().record(dend(0.9)._ref_v)
#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)
```

Recording Voltage at middle sections:

```
import matplotlib.pyplot as plt
from neuron import h

# Set up recording vectors
time1 = h.Vector()
dend_current = h.Vector()
soma_current = h.Vector()
unmyelin_current = h.Vector()
myelin_current = h.Vector()
# Record membrane currents and time
time1.record(h._ref_t)
dend_current.record(dend(0.5)._ref_i_membrane)
soma_current.record(soma3(0.5)._ref_i_membrane)
unmyelin_current.record(unmyelin(0.5)._ref_i_membrane)
myelin_current.record(myelin(0.5)._ref_i_membrane)
# Run the simulation
#By default, the NEURON h module provides the low level fadvance function for advancing one time step.
h.load_file("stdrun.hoc")
#initialize our simulation such that our cell has a resting membrane potential of -65 mV
h.finitialize(-65*mV)
#simulation from the current time (0) until 10 ms
h.continuerun(10*ms)
# Convert NEURON Vectors to Python lists for plotting
time_list1 = list(time1)
dend_current_list = list(dend_current)
soma_current_list = list(soma_current)
unmyelin_current_list = list(unmyelin_current)
myelin_current_list = list(myelin_current)
# Plotting
plt.figure(figsize=(6, 4))
plt.plot(time_list1, dend_current_list, color='red',label="Dendrite Current")
```

```
plt.plot(time_list1, soma_current_list, color='blue',label="Soma Current")
plt.plot(time_list1, unmyelin_current_list, color='green',label="Unmyelinated Axon Current")
plt.plot(time_list1, myelin_current_list, color='black',label="Myelinated Axon Current")
plt.xlabel('Time (ms)')
plt.ylabel('Membrane Current (nA)')
plt.title("Current in Sections")
plt.legend()
plt.show()
# Calculate and print the difference in peak currents
soma_peak = min(soma_current_list)
unmyelin_peak = min(unmyelin_current_list)
difference = unmyelin_peak - soma_peak
print("Difference in peak current between non-myelinated axon and soma:", difference)
```

Increasing Ion Channel Density:

```
#Now make this model slightly more realistic by increasing the density of the ion channels (found
#in 'gnabar', 'gkbar') by a factor of 10 in the non-myelinated section of the axon.
for seg in unmyelin:
    seg.hh.gnabar *= 10  # Increase Na+ channel density
    seg.hh.gkbar *= 10   # Increase K+ channel density
```

Currents after increasing the ion channel density:

```
# Set up recording vectors
time2 = h.Vector()
dend_current = h.Vector()
soma_current = h.Vector()
unmyelin_current1 = h.Vector()
myelin_current = h.Vector()

# Record membrane currents and time
time2.record(h._ref_t)
dend_current.record(dend(0.5)._ref_i_membrane)
soma_current.record(soma3(0.5)._ref_i_membrane)
unmyelin_current1.record(unmyelin(0.5)._ref_i_membrane)
myelin_current.record(myelin(0.5)._ref_i_membrane)
v2=h.Vector().record(dend(0.9)._ref_v)
# Run the simulation
h.tstop = 10  # for example, 100 ms
h.run()

# Convert NEURON Vectors to Python lists for plotting
time_list2 = list(time2)
dend_current_list = list(dend_current)
soma_current_list = list(soma_current)
unmyelin_current_list1 = list(unmyelin_current1)
myelin_current_list = list(myelin_current)

# Plotting
plt.figure(figsize=(6, 4))
plt.plot(time_list2, dend_current_list, color='red',label="Dendrite Current")
plt.plot(time_list2, soma_current_list, color='blue',label="Soma Current")
plt.plot(time_list2, unmyelin_current_list1, color='green',label="Unmyelinated Axon Current")
plt.plot(time_list2, myelin_current_list, color='black',label="Myelinated Axon Current")
plt.title("Current in Sections")
plt.xlabel('Time (ms)')
plt.ylabel('Membrane Current (nA)')
plt.legend()
```

```
plt.show()
# Calculate and print the difference in peak currents
soma_peak = min(soma_current_list)
unmyelin_peak = min(unmyelin_current_list1)
difference = unmyelin_peak - soma_peak
print("Difference in peak current between non-myelinated axon and soma:", difference)
```

Difference in currents between non myelinated axons before and after increasing ion channel density:

```
# Plotting
plt.figure(figsize=(6, 3))
plt.plot(time_list2, unmyelin_current_list1, label="Unmyelinated Axon Current-Increased Density")
plt.plot(time_list1, unmyelin_current_list, label="Unmyelinated Axon Current")
plt.xlabel('Time (ms)')
plt.ylabel('Membrane Current (nA)')
plt.title("Current Variation with Ion Channel Density")
plt.legend()
plt.show()
# Calculate and print the difference in peak currents
unmyelin_peak1 = min(unmyelin_current_list1)
unmyelin_peak = min(unmyelin_current_list)
difference = unmyelin_peak1 - unmyelin_peak
print("Difference in peak current between non-myelinated axons:", difference)
```