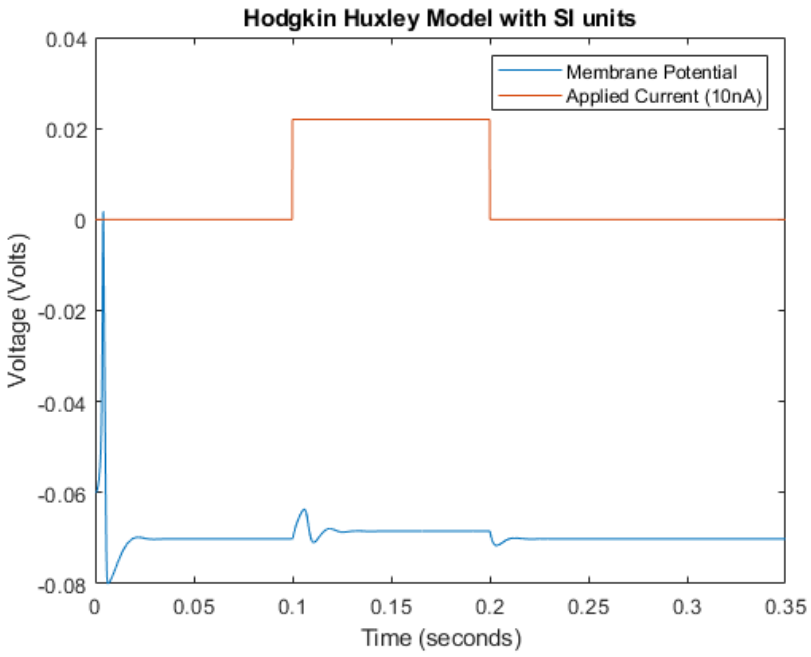
Problem 5

The objective of this lab is to simulate the system of ODEs of the Hodgkin Huxley model.

Contents

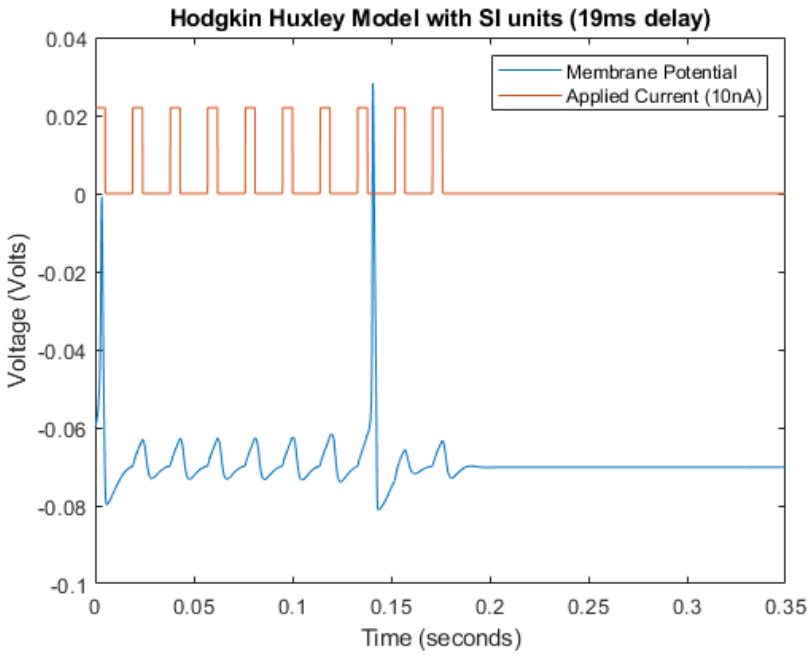
[Part A & Part B] Setting up the Model and Iapp protocol.

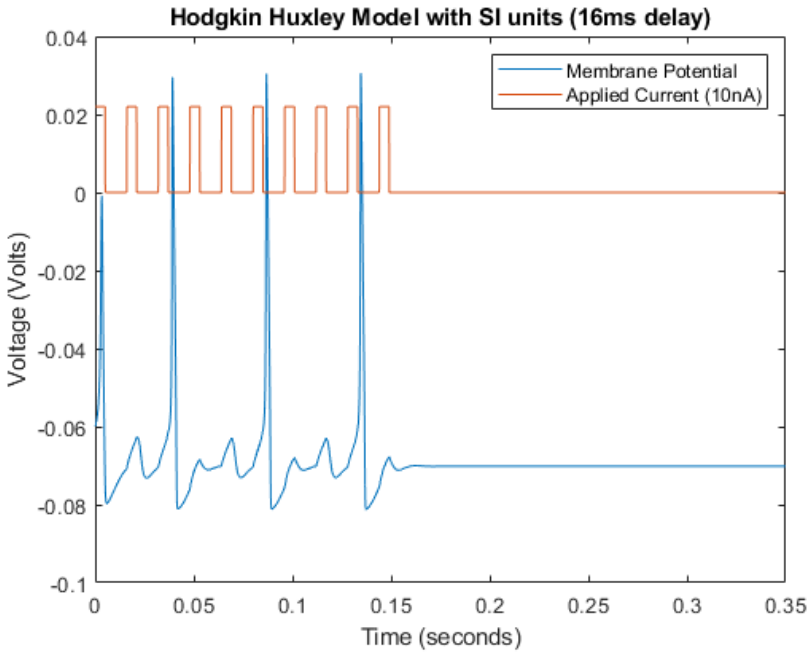
```
%Initial Conditions
Vm0 = -60*10^-3; % Membrane potential
n0 = 0; % initial m
m0 = 0; % initial n
h0 = 0; % initial h
y0 = [Vm0;n0;m0;h0]; % set intials to array
%Time vector span
tspan = [0,0.35]; % from 0 to 0.35s.
%Implementing ODE45 function
[t,y] = ode45(@(t,y) lab5partb(t,y),tspan,y0);
Vm=y(:,1); %extract Vm from the matrix

%setting up the iapp vector
iapp = zeros(1,3500);
for pos = 1:3500
    if pos >= 1000 && pos <= 2000
    iapp(pos) = 0.022; %set iapp vector
    else
    iapp(pos) = 0;
    end
end
tiapp = (0.0001:0.0001:0.35); %setting up corresponding tvec exclusive for iapp

%plotting the graphs
figure(1)
plot(t,Vm,'-')
hold on
plot(tiapp,iapp)
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
title ('Hodgkin Huxley Model with SI units')
legend({'Membrane Potential','Applied Current (10nA)'})
hold off
```

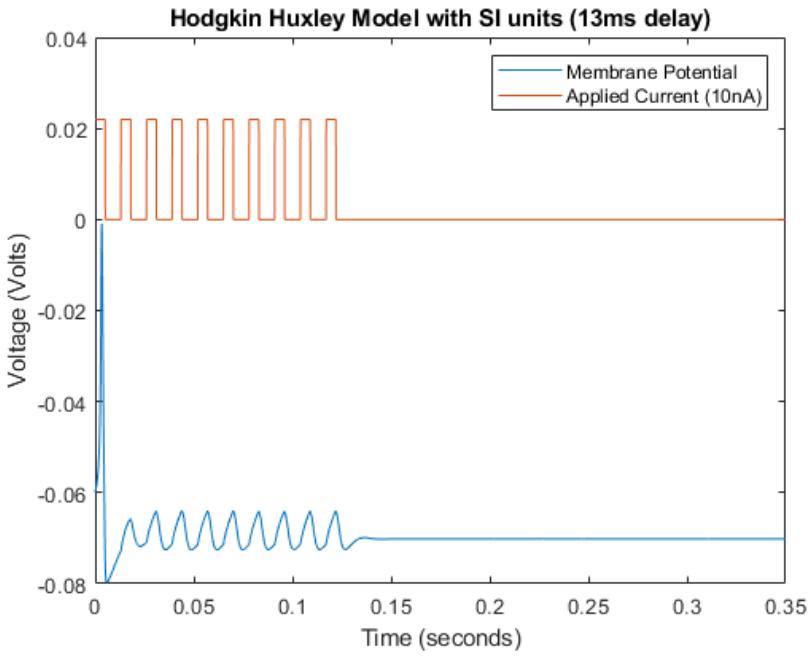


[Part C] Iapp Current-Clamp pulse Protocol

```
%Initial Conditions
Vm0 = -60*10^-3; % Membrane potential
n0 = 0; % initial m
m0 = 0; % initial n
h0 = 0; % initial h
y0 = [Vm0;n0;m0;h0]; % set intials to array
tspan = [0,0.35]; % from 0 to 0.35s.
I = 0.22*10^-9; % applied current
delay = [0.019,0.016,0.013]; %setup for the three cases
tiapp = (0.0001:0.0001:0.35); %setting up corresponding tvector for iapp plotting

%setting up cases
for cases = 1:3 %delay = 19ms,16ms,13ms in that order.
[t,y] = ode45(@(t,y) lab5partc(t,y,[delay(cases),I]),tspan,y0); %Use ODE45
Vm=y(:,1); %extract Vm from the matrix
%setting up the iapp vector for plot
iapp = zeros(1,3500); %precision of 0.1ms
pos = 1;
for j = 1:10
    for k = 1:50
        iapp(k + pos - 1) = 0.022;
    end
    pos = pos + delay(cases)*10000;
end
%plotting the graphs
figure (cases+1)
plot(t,Vm,'-')
hold on
plot(tiapp,iapp)
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
legend({'Membrane Potential','Applied Current (10nA)'})
if cases == 1
    title ('Hodgkin Huxley Model with SI units (19ms delay)')
elseif cases == 2
    title ('Hodgkin Huxley Model with SI units (16ms delay)')
elseif cases == 3
    title ('Hodgkin Huxley Model with SI units (13ms delay)')
end
hold off
% resetting iapp for next case
    for reset = 1:3500
    iapp(reset) = 0; % reset it to zero
    end
end
```


Hodgkin Huxley Model with SI units (19ms delay)
Membrane Potential
Applied Current (10nA)
Voltage (Volts)
Time (seconds)
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35


Hodgkin Huxley Model with SI units (16ms delay)
Membrane Potential
Applied Current (10nA)
Voltage (Volts)
Time (seconds)
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35

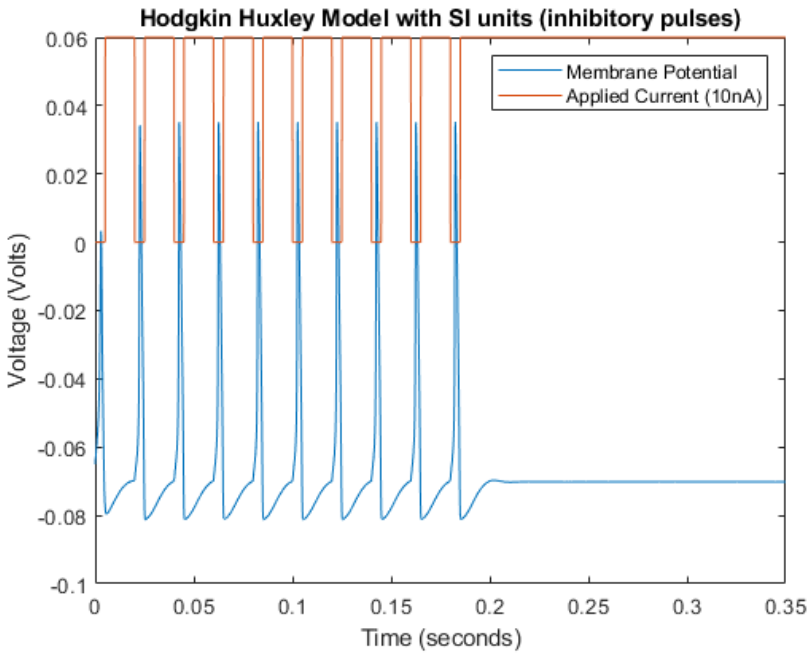


Explanation

As seen from the coupled graphs, there is an optimal delay time. From delay 13ms graph, the delay is too little that only little oscillations are present that are not enough to induce action potentials. However, delay time of 16ms has more action potentials than 19ms, which means that the optimal delay time is near to 15 and 16ms. This shows that at the right time the pulse can still superpostition with the aligned oscillation that induces an action potential, or it's not gonna work.

[Part D] Inhibitory pulses

```
%Initial Conditions
Vm0 = -65*10^-3; % Membrane potential
n0 = 0; % initial m
m0 = 0; % initial n
h0 = 0; % initial h
y0 = [Vm0;n0;m0;h0]; % set intials to array
tspan = [0,0.35]; % from 0 to 0.35s.
I = 0.65*10^-9; % applied current
delay = 0.02; %delay = 20ms = 0.02s
tiapp = (0.0001:0.0001:0.35); %setting up corresponding tvector for iapp plotting

%setting up ODE45
[t,y] = ode45(@(t,y) lab5partc(t,y,[0.02,I]),tspan,y0); %Use ODE45
%setting up iapp vector for plotting
iapp = 0.06*ones(1,3500); %precision of 0.1ms
pos = 1;
for j = 1:10
    for k = 1:50
        iapp(k + pos - 1) = 0;
    end
    pos = pos + delay*10000;
end
%plotting graphs
figure (5)
Vm=y(:,1); %extract Vm from the matrix
plot(t,Vm)
hold on
plot(tiapp,iapp)
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
legend({'Membrane Potential','Applied Current (10nA)'})
title ('Hodgkin Huxley Model with SI units (inhibitory pulses)')
hold off
```

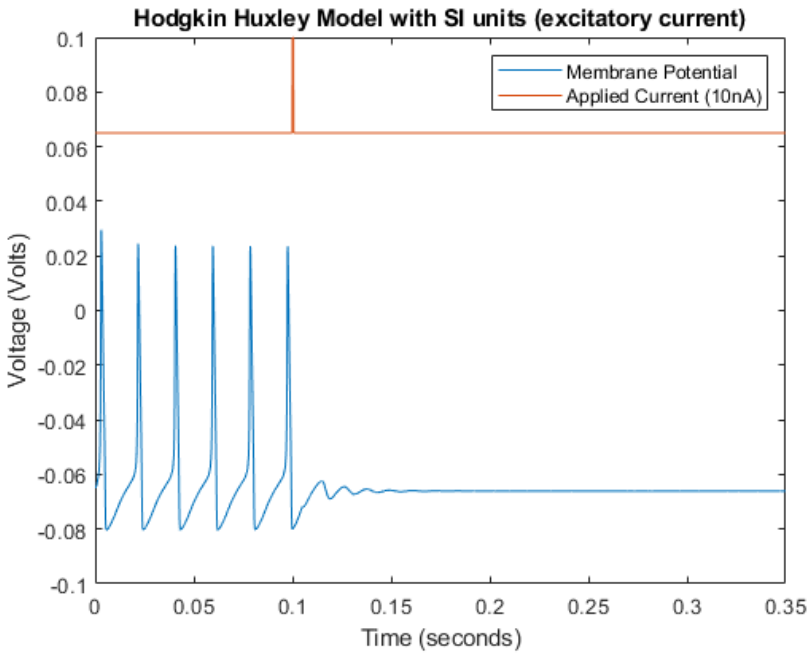


Explanation

As seen from the graph, from the time when there is a change in current, the inhibitory pulse induces the action potential which its peak aligns with the heart of the duration of the pulse. From the n,m variables we can also see the sudden increase of the two activation gates of the inhibitory pulse as well. It is the change of current that matters.

[Part E] Increase Excitatory Current

```
%Initial Conditions
Vm0 = -65*10^-3; % Membrane potential
m0 = 0.05; % initial m
n0 = 0.35; % initial n
h0 = 0.5; % initial h
y0 = [Vm0;n0;m0;h0]; % set intials to array
tspan = [0,0.35]; % from 0 to 0.35s.
tiapp = (0.0001:0.0001:0.35); %setting up corresponding tvector for iapp plotting

%setting up ODE45
[t,y] = ode45(@(t,y) lab5parte(t,y),tspan,y0); %Use ODE45
%setting up iapp vector for plotting
iapp = 0.065*ones(1,3500); %precision of 0.1ms
for pos  = 1000:1005
    iapp(pos) = 0.1;
end
%plotting graphs
figure (6)
Vm=y(:,1); %extract Vm from the matrix
plot(t,Vm,'-')
hold on
plot(tiapp,iapp)
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
legend({'Membrane Potential','Applied Current (10nA)'})
title ('Hodgkin Huxley Model with SI units (excitatory current)')
hold off
```

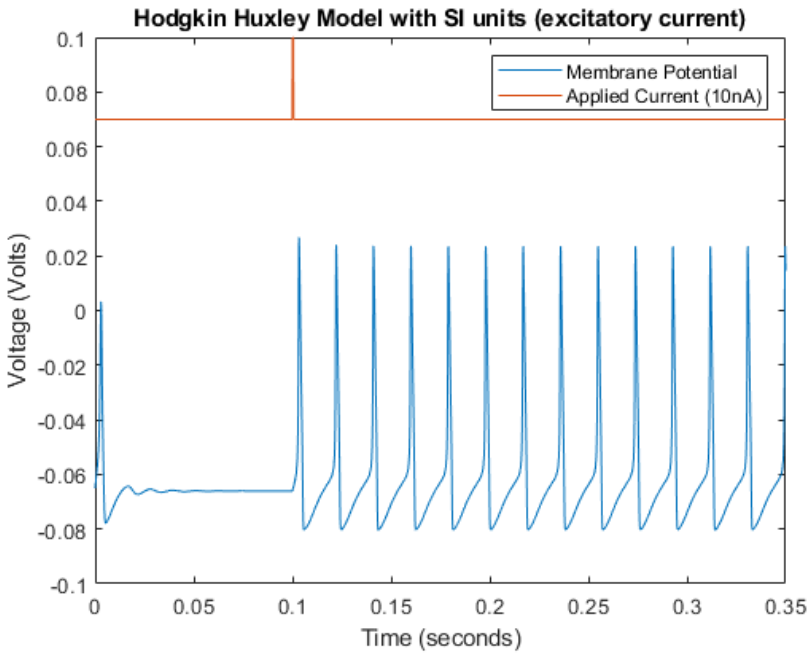


Explanation

before the excitatory current takes place, action potentials were firing. After the excitatory current, the membrane potential is stuck. The change is due to the values of n,m,h are nonzero where the channels are already activating. When there is an excitatory higher than the initial threshold, the neurons is unable to fire more action potential since it's used to the excitedness of newly input current.

[Part F] Increase Excitatory Current II

```
%Initial Conditions
Vm0 = -65*10^-3; % Membrane potential
m0 = 0; % initial m
n0 = 0; % initial n
h0 = 0; % initial h
y0 = [Vm0;n0;m0;h0];
tspan = [0,0.35]; % from 0 to 0.35s.
tiapp = (0.0001:0.0001:0.35); %setting up corresponding tvector for iapp plotting

%setting up ODE45
[t,y] = ode45(@(t,y) lab5partf(t,y),tspan,y0); %Use ODE45
%setting up iapp vector for plotting
iapp = 0.07*ones(1,3500); %precision of 0.1ms
for pos  = 1000:1005
    iapp(pos) = 0.1;
end
%plotting graphs
figure (7)
Vm=y(:,1); %extract Vm from the matrix
plot(t,Vm,'-')
hold on
plot(tiapp,iapp)
xlabel('Time (seconds)')
ylabel('Voltage (Volts)')
legend({'Membrane Potential','Applied Current (10nA)'})
title ('Hodgkin Huxley Model with SI units (excitatory current)')
hold off
```



Explanation

The excitatory pulse allows the neurons to fire lots of actions potentials than its initial subthreshold state before. This is because m,n,h are equal to zero, constant current doesn't affect the membrane potentials that much. The sudden excitatory pulse activates the m,n,h for the activation and inactivation of channels to start working, which causes the continous firing of action potentials on the graph after 100ms.

POD SUMMARY

```
% A) The diagram of the shape of b should be no pulse and one oscillation
% at the start and the end of the applied current when it changes. The
% shape of c should be the less delay the more action potentials until a
% minimal delay time before it stops producing action potentials. The shape
% of d should have action potentials each at the heart of the individual
% inhibitory pulse. The shape of e should be having a lot a action
% potentials before the exitatory pulse. After that it should not have any
% action potentials. The shape of f should be the opposite, where the
% action potentials started happening after the exitatory pulse.

% B) i)Pseudocode for one time current clamp function (e.g part b)
% * Set function taking inputs 2 inputs as time t and four column matrix y.
% * Set Vm, n, m, h as taking entries from respecting columns 1,2,3,4.
% * Set parameters (GL,GNa,GK,ENa,EK,EL,Cm) as SI Units
% * Set conditions for setting applied current when time >= to starting time and time <= ending to
% preferred value. Otherwise, set it back to initial current.
% * Set up gating variables for alpha and beta.
% * Set up ODEs for Vm, n, m, h and take values from them.
% * Set another column 4 matrix by taking [Vm,n,m,h] in that column order
% at the same time the ODEs process

% ii) Psudocode for recurring current clamp (e.g part c)
% * Same process. The only that's different is the time conditions for
% applied current.
% * let the function take in another input such that [delay time, clamp current]
% * Create 2 variables for delay time and input current.
% * Create start time and end time as lengths (1,10), and initialize them as start = 0, and end time = duration of clamp.
% * Setup the vectors by a for loop such that:
% * Each current element is the addition of the previous element and delay
% time
% * Setup another for loop that repeats 10 times such that:
% * if t>=start time and t<= end time set applied current to clamp current and break out the loop, otherwise set to initial current.
% * then it progresses to the second element.
% Note that the for loop is effective of the range of the current
% clamp times each case and will ignore all other current clamps when the
% time is within the range of one current clamp.

% C) Transitioning from using Euler's method to using ODE45 is very hard,
% so we looked through tutorials on youtube to make sure we knew what
% parameters to define for ODE45. We also read the MATLAB manual together.
```

Published with MATLAB® R2022b