```matlab
% Tutorial 4.1
% Email: jlindbloom@smu.edu

clear all
close all

% Define parameters.
global g_leak g_na g_k e_na e_k e_leak c_membrane
g_leak = 30e-9;
g_na = 12e-6;
g_k = 3.6e-6;
e_na = 45e-3;
e_k = -82e-3;
e_leak = -60e-3;
c_membrane = 100e-12;

% Setup time vector.

global dt
dt = 0.0001e-3;
t = 0:dt:0.35;

% Create vector for applied current in part b.
I_app = zeros(1, length(t));
for i = 1:((100/dt)/1000)
   I_app(floor(((100/dt)/1000)+i)) = 0.22e-9;
end

% Run simulation.
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app);

% Generate plots for part b.
f1 = figure;
figure(f1);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time");
ylim([-0.2e-10, 2.5e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
title("Membrane Potential vs. Time");
ylim([-0.1, 0.05]);

% Part (c), run twice with two different ms delays.

% Create vector for applied current.
I_app = zeros(1, length(t));
delay = 16; % number in milliseconds.
```

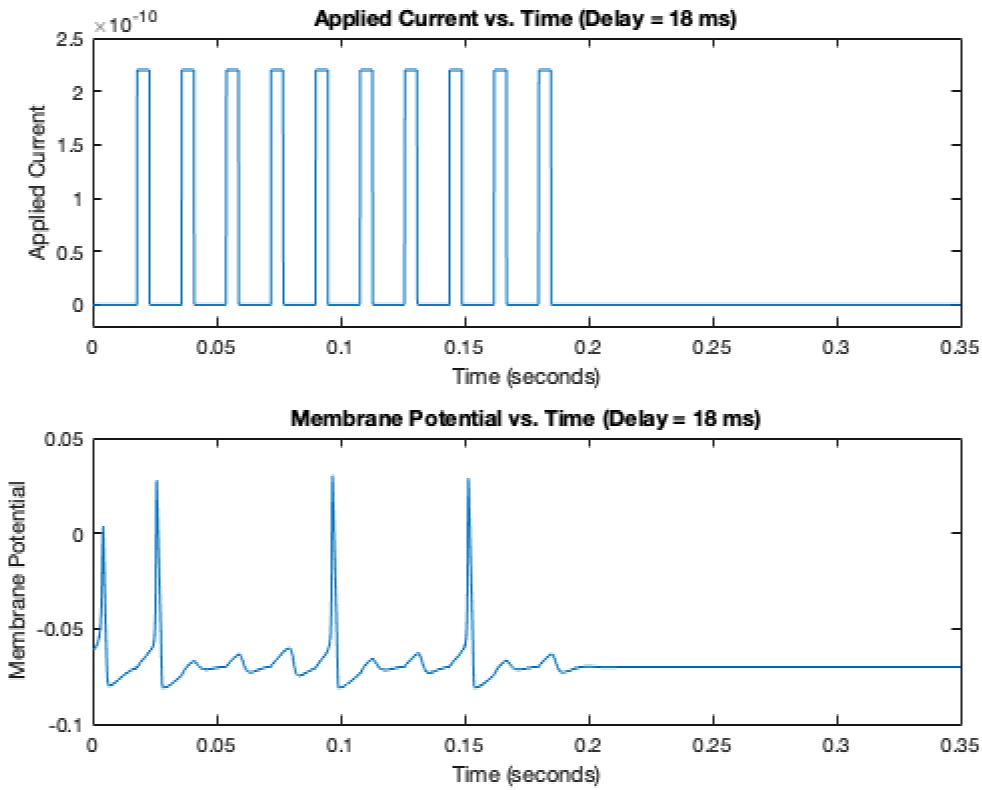
```
delay_index = floor((delay/dt)/1000);
for i = 1:10
   left = i*delay_index;
   right = left + floor((5/dt)/1000);
   I_app(left:right) = 0.22e-9;
end

% Run simulation/
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app);

% Generate plots for part c.
f2 = figure;
figure(f2);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time (Delay = 16 ms)");
ylim([-0.2e-10, 2.5e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
title("Membrane Potential vs. Time (Delay = 16 ms)");

% Create vector for applied current.
I_app = zeros(1, length(t));
delay = 18; % number in milliseconds.
delay_index = floor((delay/dt)/1000);
for i = 1:10
   left = i*delay_index;
   right = left + floor((5/dt)/1000);
   I_app(left:right) = 0.22e-9;
end

% Run simulation/
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app);

% Generate plots for part c.
f3 = figure;
figure(f3);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time (Delay = 18 ms)");
ylim([-0.2e-10, 2.5e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
title("Membrane Potential vs. Time (Delay = 18 ms)");
```

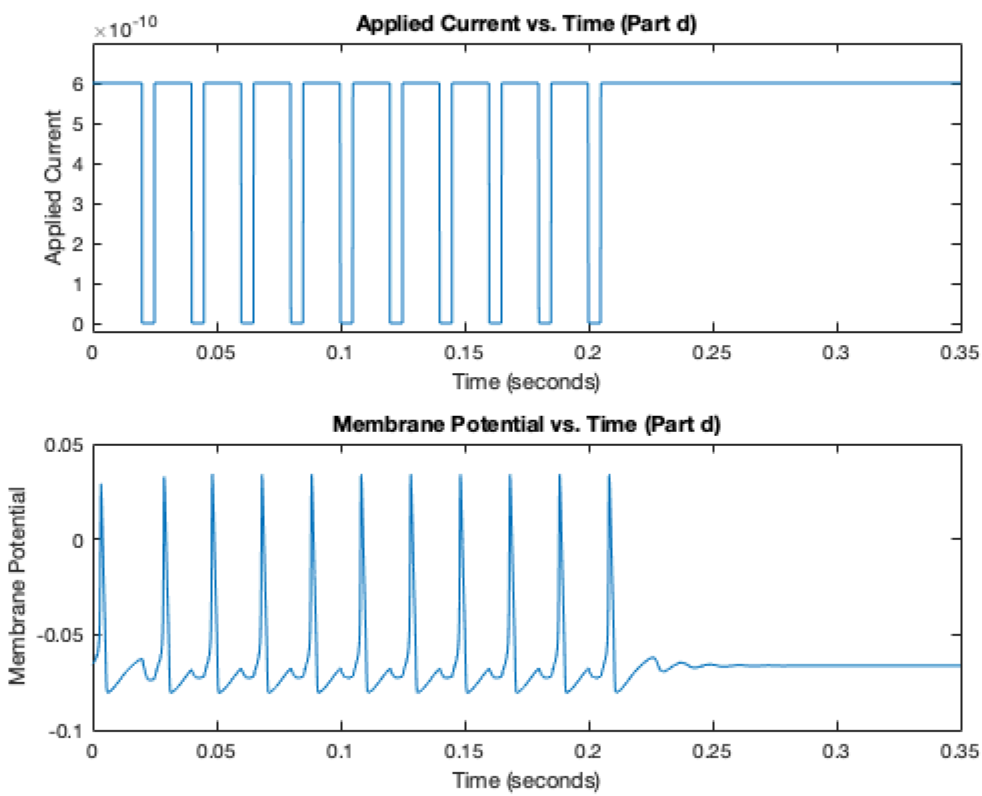
```
% Part d.

% Create vector for applied current.
I_app = zeros(1, length(t)) + 0.6e-9;
delay = 20; % number in milliseconds.
delay_index = floor((delay/dt)/1000);
for i = 1:10
   left = i*delay_index;
   right = left + floor((5/dt)/1000);
   I_app(left:right) = 0.0;
end

% Run simulation/
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app, -0.065, 0.05, 0.5,
 0.35);

% Generate plots for part d.
f4 = figure;
figure(f4);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time (Part d)");
ylim([-0.2e-10, 7e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
title("Membrane Potential vs. Time (Part d)");


% Part e.

% Create vector for applied current.
I_app = zeros(1, length(t)) + 0.65e-9;
delay = 20; % number in milliseconds.
delay_index = floor((delay/dt)/1000);
for i = 1:1
   left =  floor((100/dt)/1000);
   right = left + floor((5/dt)/1000);
   I_app(left:right) = 1e-9;
end

% Run simulation/
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app, -0.065, 0.05, 0.5,
 0.35);

% Generate plots for part e.
f5 = figure;
figure(f5);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
```

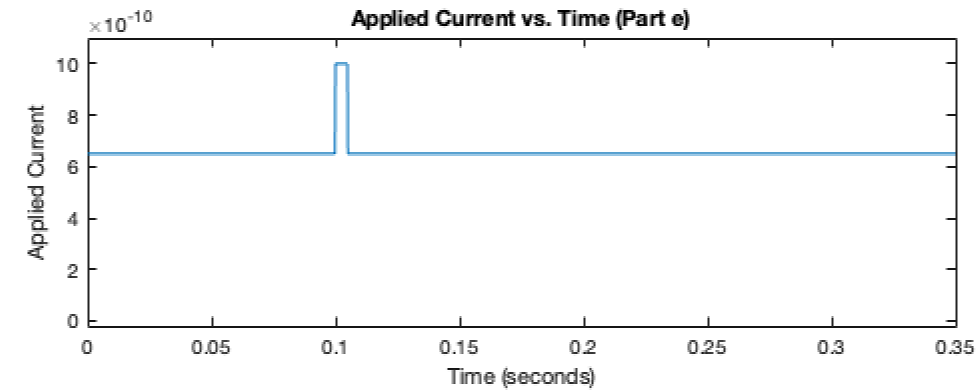
```
ylabel("Applied Current");
title("Applied Current vs. Time (Part e)");
ylim([-0.2e-10, 11e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
title("Membrane Potential vs. Time (Part e)");

f5half = figure;
figure(f5half);
plot(t, m_sim);
hold on
plot(t, h_sim, '-');
hold on
plot(t, n_sim, '--');
xlabel("Time (seconds)");
ylabel("Gating Variables");
title("Gating Variables");
legend("Sodium Activation", "Sodium Inactivation", "Potassium
 Activation");
xlim([-0.05, 0.4]);
saveas(f5half,"Part_e_activation_variables.png");


% Part f.

% Create vector for applied current.
I_app = zeros(1, length(t)) + 0.7e-9;
delay = 20; % number in milliseconds.
delay_index = floor((delay/dt)/1000);
for i = 1:1
   left =  floor((100/dt)/1000);
   right = left + floor((5/dt)/1000);
   I_app(left:right) = 1e-9;
end

% Run simulation/
[v_sim, m_sim, h_sim, n_sim] = hhsim(t, I_app, -0.065, 0.00, 0.00,
 0.00);

% Generate plots for part f.
f6 = figure;
figure(f6);
subplot(2,1,1);
plot(t, I_app);
xlabel("Time (seconds)");
ylabel("Applied Current");
title("Applied Current vs. Time (Part f)");
ylim([-0.2e-10, 11e-10]);
subplot(2,1,2);
plot(t, v_sim);
xlabel("Time (seconds)")
ylabel("Membrane Potential");
```

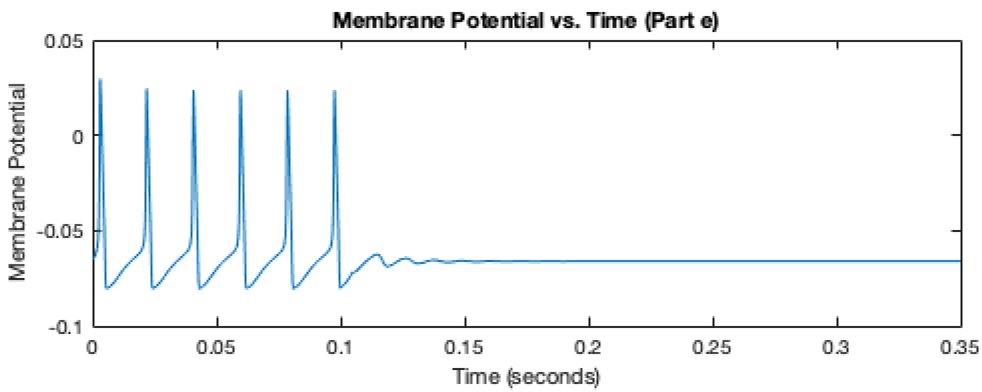
```
title("Membrane Potential vs. Time (Part f)");

f7 = figure;
figure(f7);
plot(t, m_sim);
hold on
plot(t, h_sim, '-');
hold on
plot(t, n_sim, '--');
xlabel("Time (seconds)");
ylabel("Gating Variables");
title("Gating Variables");
legend("Sodium Activation", "Sodium Inactivation", "Potassium
 Activation");
xlim([-0.05, 0.4]);
saveas(f7,"Part_f_activation_variables.png");




%%%%%%%%%%%%%%%%%%%%%%%
% Function Definitions:

function [v_sim, m_sim, h_sim, n_sim] = hhsim(t, i_applied, v_init,
 m_init, h_init, n_init)
% Simulates the Hodgkin-Huxley model given the input time vector and
% applied current.

global dt g_leak g_na g_k e_na e_k e_leak c_membrane

% Default parameters if not inputted.
if (~exist('v_init'))
    v_init = -61e-3;
end
if (~exist('m_init'))
    m_init = 0;
end
if (~exist('h_init'))
    h_init = 0;
end
if (~exist('n_init'))
    n_init = 0;
end

% Setup vectors.
v_sim = zeros(1, length(t));
m_sim = zeros(1, length(t));
h_sim = zeros(1, length(t));
n_sim = zeros(1, length(t));

v_sim(1) = v_init;
m_sim(1) = m_init;
h_sim(1) = h_init;
n_sim(1) = n_init;
```

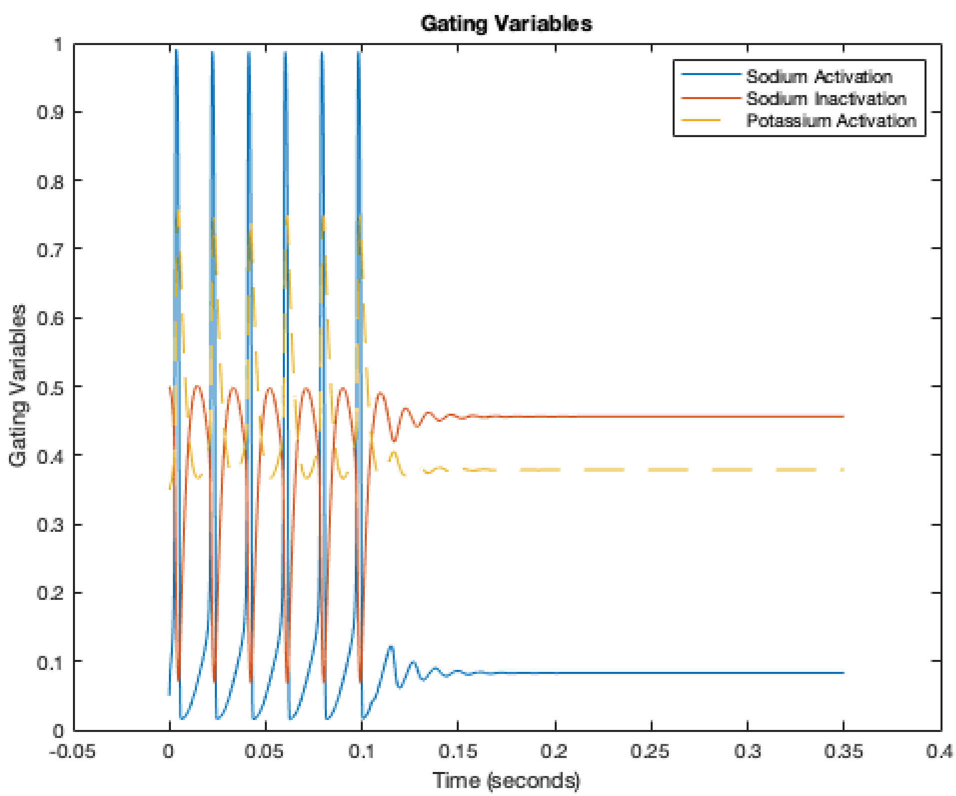
```
% March forward in time.
for n = 1:(length(t)-1)
    % Update v_sim.
    term1 = g_leak*(e_leak-v_sim(n));
    term2 = g_na*(m_sim(n)^3)*(h_sim(n))*(e_na - v_sim(n));
    term3 = g_k*(n_sim(n)^4)*(e_k - v_sim(n));
    v_sim(n+1) = v_sim(n) + (dt*((term1 + term2 + term3 +
 i_applied(n))/c_membrane));

    % Update m_sim.
    alpha = (((10^5)*(-v_sim(n)-0.045))/(exp(100*(-
v_sim(n)-0.045))-1));
    beta = (4*(10^3))*exp((-v_sim(n) - 0.07)/(0.018));
    term1 = alpha*(1-m_sim(n));
    term2 = -beta*m_sim(n);
    m_sim(n+1) = m_sim(n) + (dt*(term1 + term2));

    % Update h_sim.
    alpha = 70*exp(50*(-v_sim(n)-0.07));
    beta = ((10^3) / (1 + exp(100*(-v_sim(n)-0.04))));
    term1 = alpha*(1-h_sim(n));
    term2 = -beta*h_sim(n);
    h_sim(n+1) = h_sim(n) + (dt*(term1 + term2));

    % Update n_sim.
    alpha = (((10^4)*(-v_sim(n)-0.06))/(exp(100*(-v_sim(n)-0.06))-1));
    beta = 125*exp(((-v_sim(n)-0.07)/(0.08)));
    term1 = alpha*(1-n_sim(n));
    term2 = -beta*n_sim(n);
    n_sim(n+1) = n_sim(n) + (dt*(term1 + term2));
end

end
```

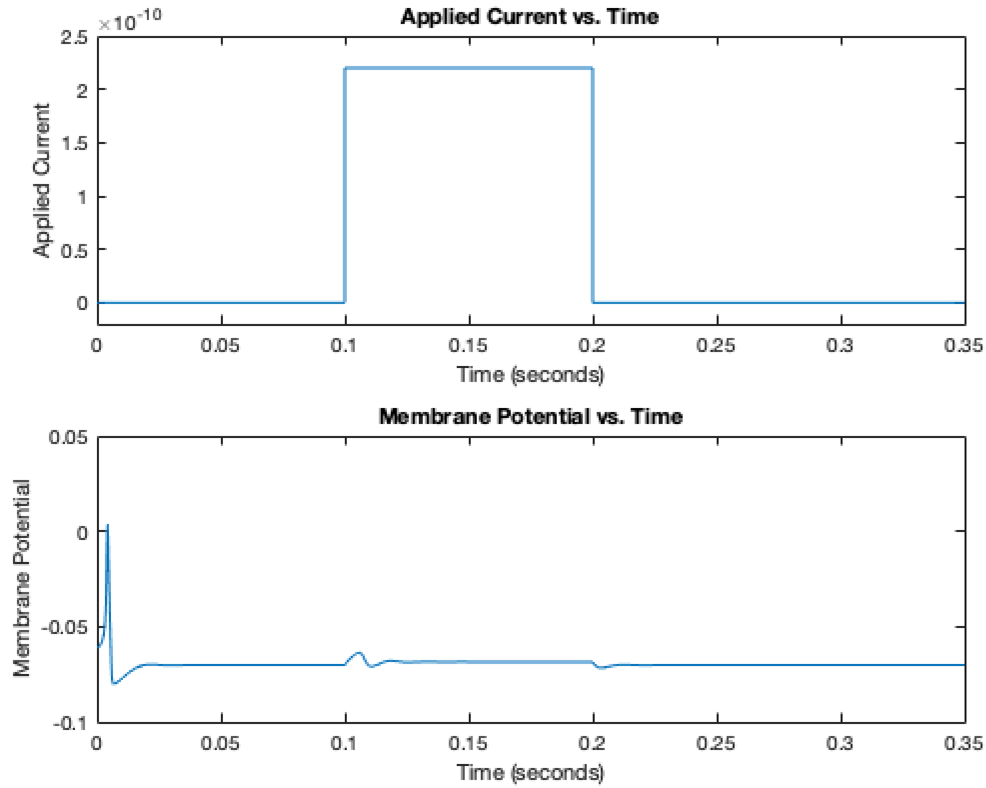

Applied Current vs. Time (Delay = 18 ms)
×10^-10
Applied Current
Time (seconds)
Membrane Potential vs. Time (Delay = 18 ms)
Membrane Potential
Time (seconds)

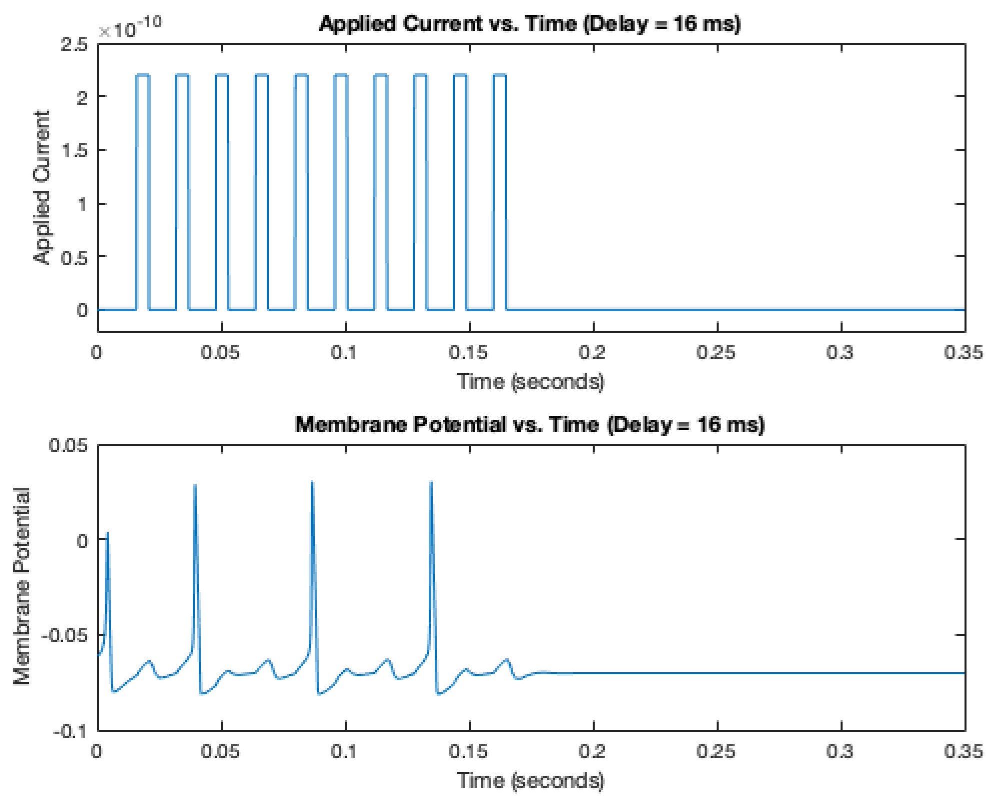

Applied Current vs. Time (Part d)
×10^-10
Applied Current
Time (seconds)
Membrane Potential vs. Time (Part d)
Membrane Potential
Time (seconds)

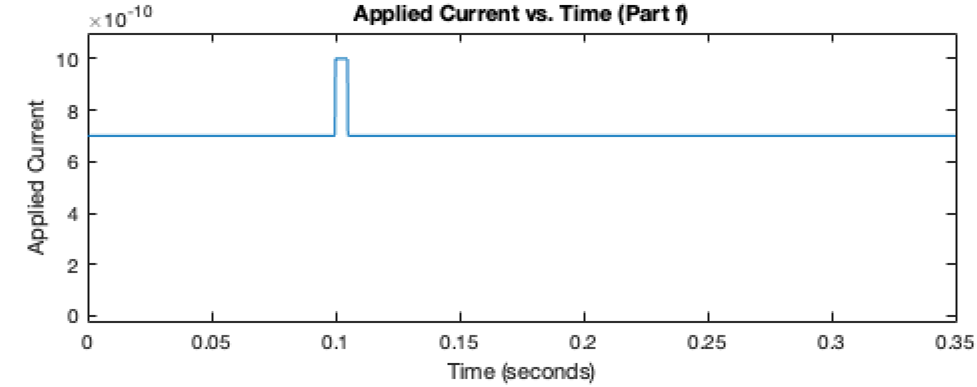

Applied Current vs. Time (Part e)
×10^-10
Applied Current
Time (seconds)
0
2
4
6
8
10
0.05
0.1
0.15
0.2
0.25
0.3
0.35

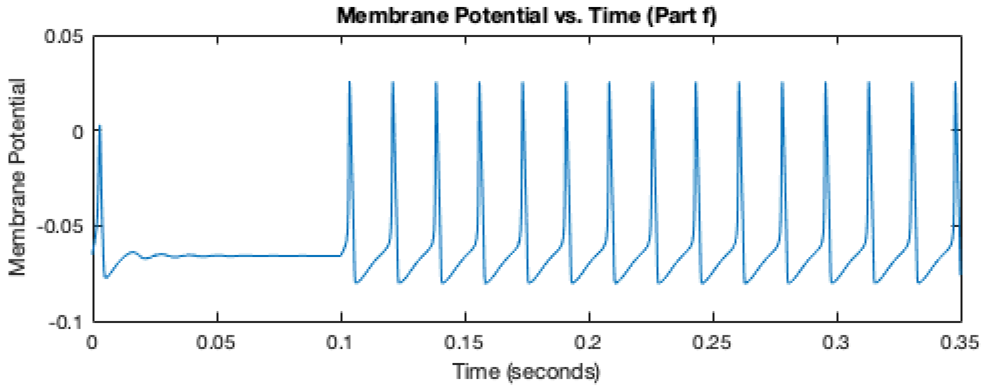

Membrane Potential vs. Time (Part e)
Membrane Potential
Time (seconds)
0.05
0
-0.05
-0.1
0.1
0.15
0.2
0.25
0.3
0.35

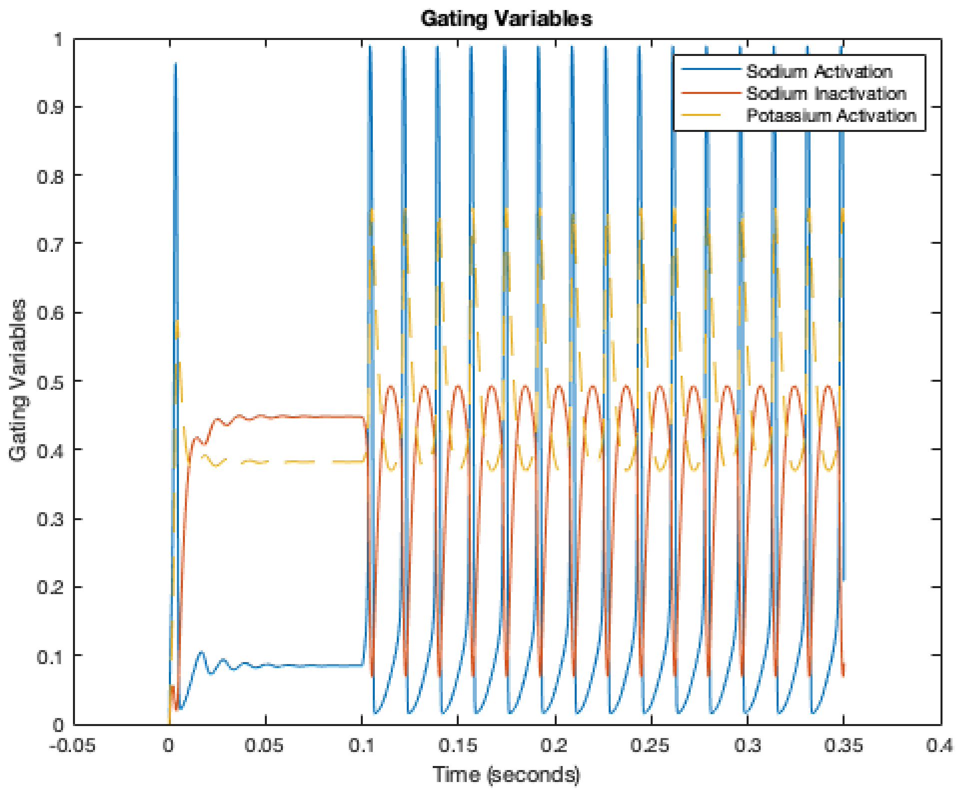

Gating Variables
Sodium Activation
Sodium Inactivation
Potassium Activation
Gating Variables
Time (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-0.05
0.05
0.15
0.25
0.35


Applied Current vs. Time
×10^-10
Applied Current
Time (seconds)
Membrane Potential vs. Time
Membrane Potential
Time (seconds)


Applied Current vs. Time (Delay = 16 ms)
×10^-10
Applied Current
Time (seconds)
Membrane Potential vs. Time (Delay = 16 ms)
Membrane Potential
Time (seconds)


Applied Current vs. Time (Part f)
×10^-10
Applied Current
Time (seconds)


Membrane Potential vs. Time (Part f)
Membrane Potential
Time (seconds)


Gating Variables
Sodium Activation
Sodium Inactivation
Potassium Activation
Gating Variables
Time (seconds)

Published with MATLAB® R2018a