```
% Tutorial 2.2
% Created by Jonathan Lindbloom, 2/16/2019
% Email: jlindbloom@smu.edu

close all
clear all

% Define parameters.
global e_leak r_membrane c_membrane v_threshold v_reset
e_leak = -70e-3;
r_membrane = 100e6;
c_membrane = 0.1e-9;
v_threshold = -50e-3;   % Threshold for Q1.
v_reset = -65e-3;       % Reset for Q1.

% Create figure for mean firing rates vs. input current.

f1 = figure;

% Create figure for mean membrane potentials vs. input current.

f2 = figure;

% Create figure for mean membrane potentials vs. firing rate.

f3 = figure;

% Create vectors for storing plot data for plots created above.

I_var = (100e-12):(25e-12):(600e-12);  % Create vector for varying
 current.

q1_firing_rate = zeros(1, length(I_var));
q2_firing_rate = zeros(1, length(I_var));
q3_firing_rate = zeros(1, length(I_var));
q1_mean_membrane_potential = zeros(1, length(I_var));
q2_mean_membrane_potential = zeros(1, length(I_var));
q3_mean_membrane_potential = zeros(1, length(I_var));

% Create time vector.
global dt
dt = 0.0001;
t = 0:dt:2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q1 - Forced Voltage Clamp.

v = zeros(1, length(t));    % Create membrane potential vector.
v(1) = e_leak;              % Set initial value equal to the leak
 potential.

refractory_period = 2.5e-3;
```

```
for i = 1:(length(I_var))   % Vary I_app and compute mean firing rates
 & membrane potentials.
    I_app = (I_var(i))*ones(1,length(t));
    result = q1sim(v, t, refractory_period, I_app);     % Run
 simulation.
    q1_mean_membrane_potential(i) = mean(result);     % Store mean
 potential.
    spikes = fire_count(result, 49e-3);
    fire_rate = spikes/2;
    q1_firing_rate(i) = fire_rate;                      % Store firing
 rate.
end

I220 = (220e-12)*ones(1,length(t));     % Create current vector of
 I220 pA.
I600 = (600e-12)*ones(1,length(t));     % Create current vector of
 I600 pA.

v220 = q1sim(v, t, refractory_period, I220);
v600 = q1sim(v, t, refractory_period, I600);
f4 = figure;
figure(f4);
plot(t(1:2000), v220(1:2000));
hold on;
plot(t(1:2000), v600(1:2000));
ylim([-0.07 .055]);
xlabel('Time (s)');
ylabel('Membrane Potential (mV)');
title('Q1 - Forced Voltage Clamp');
legend('I_{app} = 220 pA', 'I_{app} = 600 pA');


%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q2 - Threshold increase.

v = zeros(1, length(t));    % Create membrane potential vector.
v(1) = e_leak;              % Set initial value equal to the leak
 potential.

for i = 1:(length(I_var))   % Vary I_app and compute mean firing rates
 & membrane potentials.
    I_app = (I_var(i))*ones(1,length(t));
    result = q2sim(v, t, I_app);                % Run simulation.
    q2_mean_membrane_potential(i) = mean(result);   % Store mean
 potential.
    spikes = fire_count(result, 49e-3);
    fire_rate = spikes/2;
    q2_firing_rate(i) = fire_rate;              % Store firing rate.
end

v220 = q2sim(v, t, I220);
v600 = q2sim(v, t, I600);
f5 = figure;
```

```
figure(f5);
plot(t(1:2000), v220(1:2000));
hold on;
plot(t(1:2000), v600(1:2000));
ylim([-0.07 .055]);
xlabel('Time (s)');
ylabel('Membrane Potential (mV)');
title('Q2 - Threshold Increase');
legend('I_{app} = 220 pA', 'I_{app} = 600 pA');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Q3 - Refractory conductance.

v = zeros(1, length(t));    % Create membrane potential vector.
v(1) = e_leak;              % Set initial value equal to the leak
 potential.


for i = 1:(length(I_var))   % Vary I_app and compute mean firing rates
 & membrane potentials.
    I_app = (I_var(i))*ones(1,length(t));
    result = q3sim(v, t, I_app);                % Run simulation.
    q3_mean_membrane_potential(i) = mean(result);   % Store mean
 potential.
    spikes = fire_count(result, 49e-3);
    fire_rate = spikes/2;
    q3_firing_rate(i) = fire_rate;              % Store firing rate.
end

v220 = q3sim(v, t, I220);
v600 = q3sim(v, t, I600);
f6 = figure;
figure(f6);
plot(t(1:2000), v220(1:2000));
hold on;
plot(t(1:2000), v600(1:2000));
ylim([-0.09 .055]);
xlabel('Time (s)');
ylabel('Membrane Potential (mV)');
title('Q3 - Refractory Conductance');
legend('I_{app} = 220 pA', 'I_{app} = 600 pA');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make mean firing rate vs. input current plot.
figure(f1);
scatter(I_var, q1_firing_rate);
hold on;
scatter(I_var, q2_firing_rate);
hold on;
scatter(I_var, q3_firing_rate);
legend('Q1 - Forced Voltage Clamp', 'Q2 - Threshold Increase', 'Q3 -
 Refractory Conductance');
xlabel('Input Current (pA)');
```

```
ylabel('Mean Firing Rate (Hz)');
title('Mean Firing Rate vs. Input Current for Modified LIF Models');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make mean membrane potential vs. input current plot.

figure(f2);
scatter(I_var, q1_mean_membrane_potential);
hold on;
scatter(I_var, q2_mean_membrane_potential);
hold on;
scatter(I_var, q3_mean_membrane_potential);
legend('Q1 - Forced Voltage Clamp', 'Q2 - Threshold Increase', 'Q3 -
 Refractory Conductance');
xlabel('Input Current (pA)');
ylabel('Mean Membrane Potential (V)');
title('Mean Membrane Potential vs. Input Current for Modified LIF
 Models');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mean membrane potential vs. firing rate plot.
figure(f3);
scatter(q1_firing_rate, q1_mean_membrane_potential);
hold on;
scatter(q2_firing_rate, q2_mean_membrane_potential);
hold on;
scatter(q3_firing_rate, q3_mean_membrane_potential);
legend('Q1 - Forced Voltage Clamp', 'Q2 - Threshold Increase', 'Q3 -
 Refractory Conductance');
xlabel('Firing Rate');
ylabel('Mean Membrane Potential (V)');
title('Mean Membrane Potential vs. Firing Rate for Modified LIF
 Models');


%%%%%%%%%%%%%%%%%%%%%%%%
% Save all figures.
saveas(f1, 'Firing_Rate_vs_Input_Current.png');
saveas(f2, 'Mean_Membrane_Potential_vs_Input_Current.png');
saveas(f3, 'Mean_Membrane_Potential_vs_Firing_Rate.png');
saveas(f4, 'Q1_Forced_Voltage_Clamp.png');
saveas(f5, 'Q2_Threshold_Increase.png');
saveas(f6, 'Q3_Refractory_Conductance.png');


%%%%%%%%%%%%%%%%%%%%%%%%%
% Comments on plots:

% Mean Firing Rate vs. Input Current - until the input current causes
 the
% membrane potential to reach the threshold, there are no spikes, and
 in
```

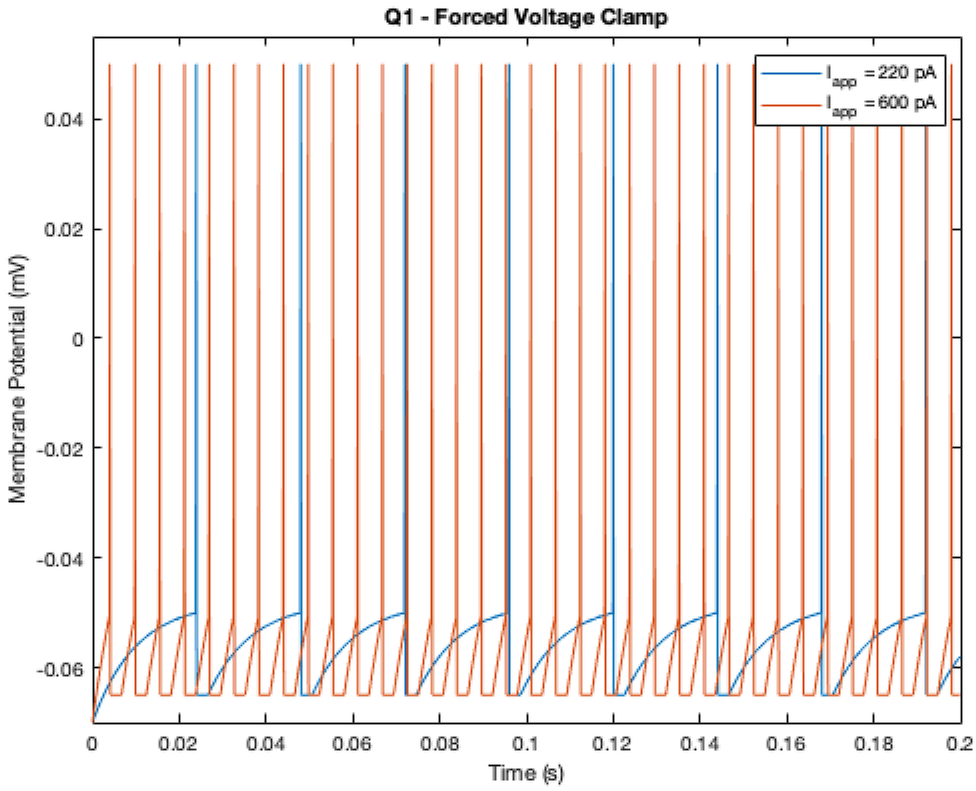
```
% turn the firing rate is necessarily zero. Once the input current is
 large
% enough to cause the membrane potential to reach the threshold, we
 see
% that an increase in the input current leads to an increase in the
 firing
% rate. This is to be expected.

% Mean Membrane Potential vs. Input Current - until the input current
% sauses the membrane potential to reach the threshold and produce
 spikes,
% the input current simply raises the mean membrane potential to
 values
% under the threshold. However, once the input current is high enough
 to
% induce spiking, we see a decrease in the mean membrane potentials.
% However, for the increasing threshold method we see the mean
 membrane
% potential quickly rise with an increase in input current unlike the
 other
% two methods - this agrees with the comments the textbook makes on
 page
% 74 ("The increasing threshold method allows the mean membrane
 potential to
% increase with the firing rate, while also preventing a spike during
 a
% refractory period.").

% Mean Membrane Potential vs. Firing Rate - we see that the forced
 voltage
% clamp method produces a curve that is decreasing, while the
 threshold
% increase method produces a curve that decreases quickyl and then
% increases as the firing rate increases. The reason this occurs is
 that as
% the firing rate increases, the membrane potential function produced
 by
% the voltage clamp spends a relatively longer time at the fixed reset
% value since the time spent at the reset value is fixed and
 independent of
% the input current or firing rate. However, with the raised threshold
% method there is no fixed time per spike spent at any reset value, so
 the
% mean potential is not affected as severaly as it is by the voltage
 clamp
% method. The refractory conductance method is similar to the voltage
 clamp
% in this regard, but the curve for this method is shifted much
 further
% down compared to the other two - this is because of the parameter
 E_k =
% -80 mV used in the model for the ODE for refractory conductance.
```

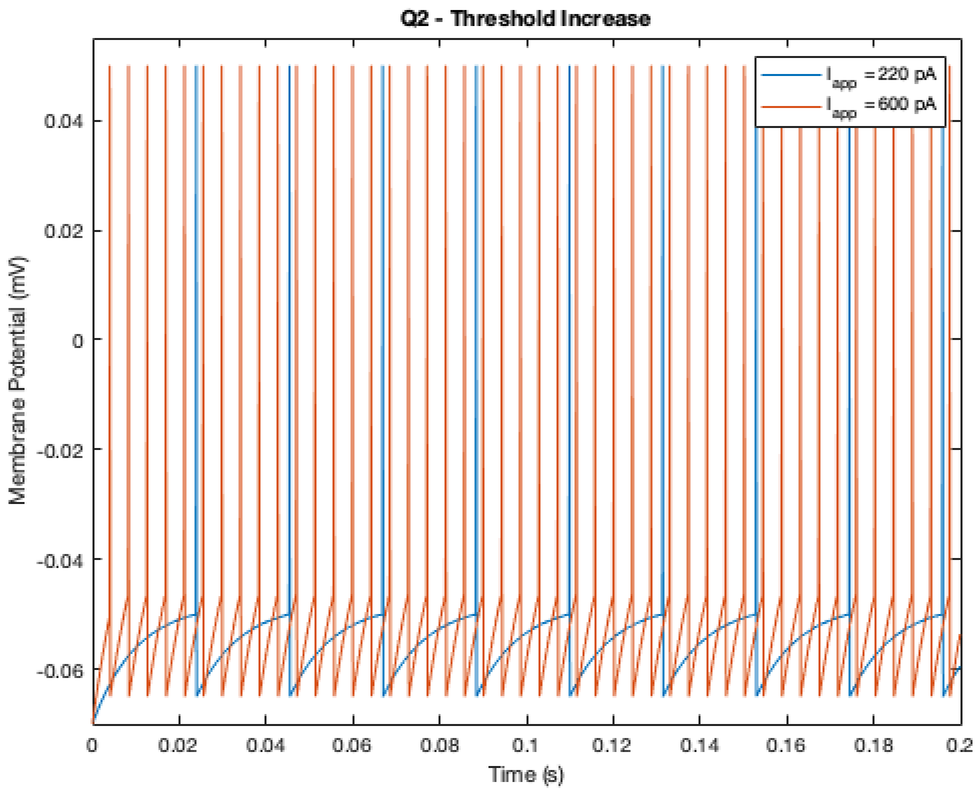
```
%%%%%%%%%%%%%%%%%%%%%%%
% Function Definitions:

function value = F(v, t, i_applied)
global e_leak r_membrane c_membrane
% Evaluates the RHS function of the ODE with given v, t, and I_app.
% Used for generating the RK4 solution.
value = ((e_leak - v)/(r_membrane*c_membrane)) + ((i_applied)/
c_membrane);
end

function value = Th(v, t)
% Evaluates the RHS function of the ODE for threshold with given v and
 t.
% Used for generating the RK4 solution.
value = (-50e-3 - v)/(1e-3);
end

function value = F2(v, t, i_applied, g)
global e_leak r_membrane c_membrane
% Evaluates the RHS function of the ODE with given v, t, I_app, and G
% (refractory conductance). Used for generating the RK4 solution.
value = ((e_leak - v)/(r_membrane*c_membrane)) + (g*(-80e-3 - v)/
c_membrane) + ((i_applied)/c_membrane);
end

function value = G(g, t)
% Evaluates the RHS function of the ODE for conductance with given g
 and t.
% Used for generating the RK4 solution.
value = -g/(0.2e-3);
end

function v_simulated = q1sim(v, t, refractory_period, i_applied)
% Simulates the membrane potential with a foced voltage clamp.
global v_threshold v_reset dt
v_simulated = zeros(1, length(v));
v_simulated(1) = v(1);
temp = length(v)-1;
refractory_cooldown = 0;
for n = 1:temp
    if refractory_cooldown > 0
        v_simulated(n+1) = v_reset;
        refractory_cooldown = refractory_cooldown - dt;
    else
        K0 = dt*(F(v_simulated(n), t(n), i_applied(n)));
        K1 = dt*(F(v_simulated(n) + K0/2, t(n) + dt/2, i_applied(n)));
        K2 = dt*(F(v_simulated(n) + K1/2, t(n) + dt/2, i_applied(n)));
        K3 = dt*(F(v_simulated(n) + K2, t(n) + dt, i_applied(n)));
        K4 = (K0 + 2*K1 + 2*K2 + K3)/6;

        if (v_simulated(n) + K4) >= v_threshold
            v_simulated(n+1) = v_reset;
```

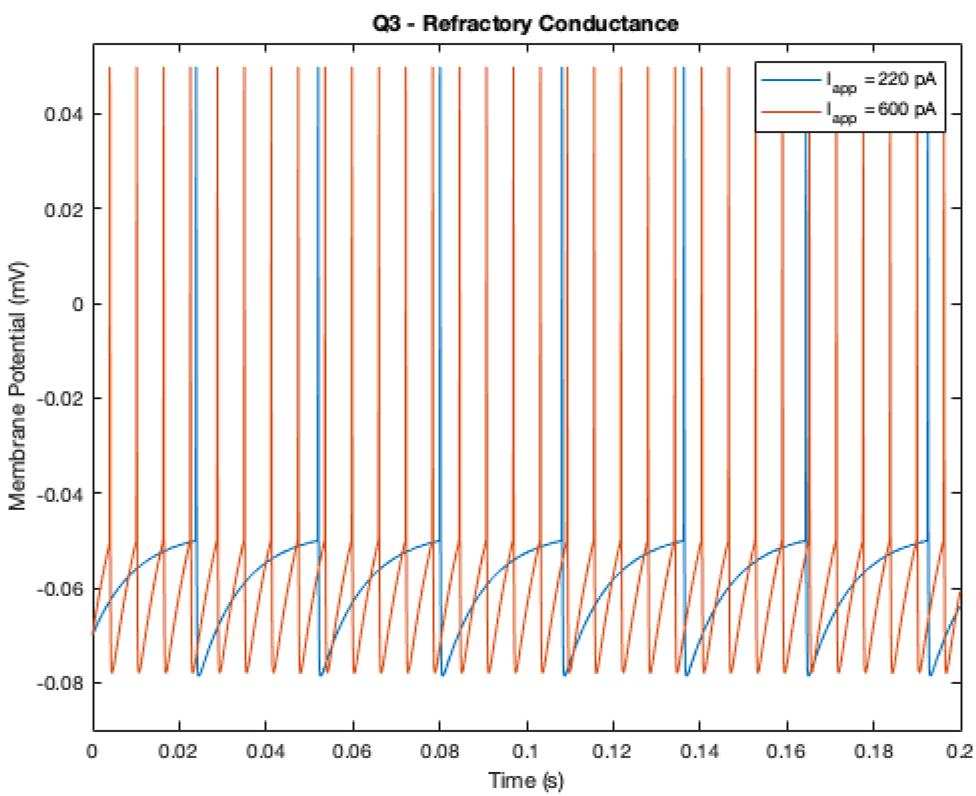
```
            refractory_cooldown = refractory_cooldown +
 refractory_period;
            v_simulated(n) = 50e-3;
        else
            v_simulated(n+1) = v_simulated(n) + K4;
        end
    end
end
end

function v_simulated = q2sim(v, t, i_applied)
% Simulates the membrane potential with threshold increase.
global v_threshold v_reset dt
v_simulated = zeros(1, length(v));
v_simulated(1) = v(1);

% Make threshold vector.
v_th = zeros(1, length(v));
v_th(1) = -50e-3;

temp = length(v)-1;
for n = 1:temp
    % Update membrane potential.
    K0 = dt*(F(v_simulated(n), t(n), i_applied(n)));
    K1 = dt*(F(v_simulated(n) + K0/2, t(n) + dt/2, i_applied(n)));
    K2 = dt*(F(v_simulated(n) + K1/2, t(n) + dt/2, i_applied(n)));
    K3 = dt*(F(v_simulated(n) + K2, t(n) + dt, i_applied(n)));
    K4 = (K0 + 2*K1 + 2*K2 + K3)/6;

    % Update threshold.
    k0 = dt*(Th(v_th(n), t(n)));
    k1 = dt*(Th(v_th(n) + k0/2, t(n) + dt/2));
    k2 = dt*(Th(v_th(n) + k1/2, t(n) + dt/2));
    k3 = dt*(Th(v_th(n) + k2, t(n) + dt));
    k4 = (k0 + 2*k1 + 2*k2 + k3)/6;

    if (v_simulated(n) + K4) >= v_th(n)
        v_simulated(n+1) = v_reset;
        v_simulated(n) = 50e-3;
        v_th(n+1) = 200e-3;
    else
        v_simulated(n+1) = v_simulated(n) + K4;
        v_th(n+1) = v_th(n) + k4;
    end


end
end

function v_simulated = q3sim(v, t, i_applied)
% Simulates the membrane potential with refractory conductance
 increase.
global v_threshold v_reset dt
v_simulated = zeros(1, length(v));
```

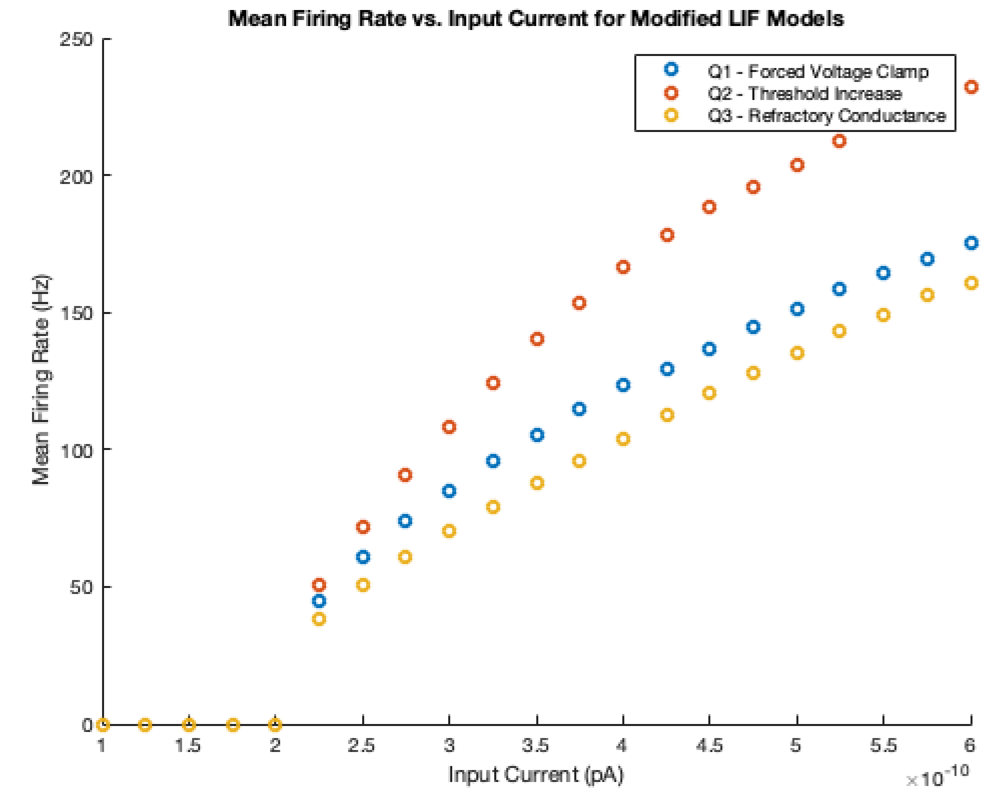
```
v_simulated(1) = v(1);

% Make threshold vector.
v_th = zeros(1, length(v));
v_th(1) = -50e-3;

% Make conductance vector.
g = zeros(1, length(v));

temp = length(v)-1;
for n = 1:temp
    % Update membrane potential.
    K0 = dt*(F2(v_simulated(n), t(n), i_applied(n), g(n)));
    K1 = dt*(F2(v_simulated(n) + K0/2, t(n) + dt/2, i_applied(n),
 g(n)));
    K2 = dt*(F2(v_simulated(n) + K1/2, t(n) + dt/2, i_applied(n),
 g(n)));
    K3 = dt*(F2(v_simulated(n) + K2, t(n) + dt, i_applied(n), g(n)));
    K4 = (K0 + 2*K1 + 2*K2 + K3)/6;

    % Update threshold.
    k0 = dt*(Th(v_th(n), t(n)));
    k1 = dt*(Th(v_th(n) + k0/2, t(n) + dt/2));
    k2 = dt*(Th(v_th(n) + k1/2, t(n) + dt/2));
    k3 = dt*(Th(v_th(n) + k2, t(n) + dt));
    k4 = (k0 + 2*k1 + 2*k2 + k3)/6;

    % Update refractory conductance.
    g0 = dt*(G(g(n), t(n)));
    g1 = dt*(G(g(n) + g0/2, t(n) + dt/2));
    g2 = dt*(G(g(n) + g1/2, t(n) + dt/2));
    g3 = dt*(G(g(n) + g2, t(n) + dt));
    g4 = (g0 + 2*g1 + 2*g2 + g3)/6;

    if (v_simulated(n) + K4) >= v_th(n)
        v_simulated(n+1) = v_simulated(n) + K4;
        v_simulated(n) = 50e-3;
        v_th(n+1) = 200e-3;
        g(n+1) = g(n) + 2e-6;
    else
        v_simulated(n+1) = v_simulated(n) + K4;
        v_th(n+1) = v_th(n) + k4;
        g(n+1) = g(n) + g4;
    end


end
end

function count = fire_count(v, v_exceed)
% Counts the number of spikes by counting the number of times the
% membrane potential exceeds a given value. v is a vector, v_exceed
% is a scalar.
count = 0;
```

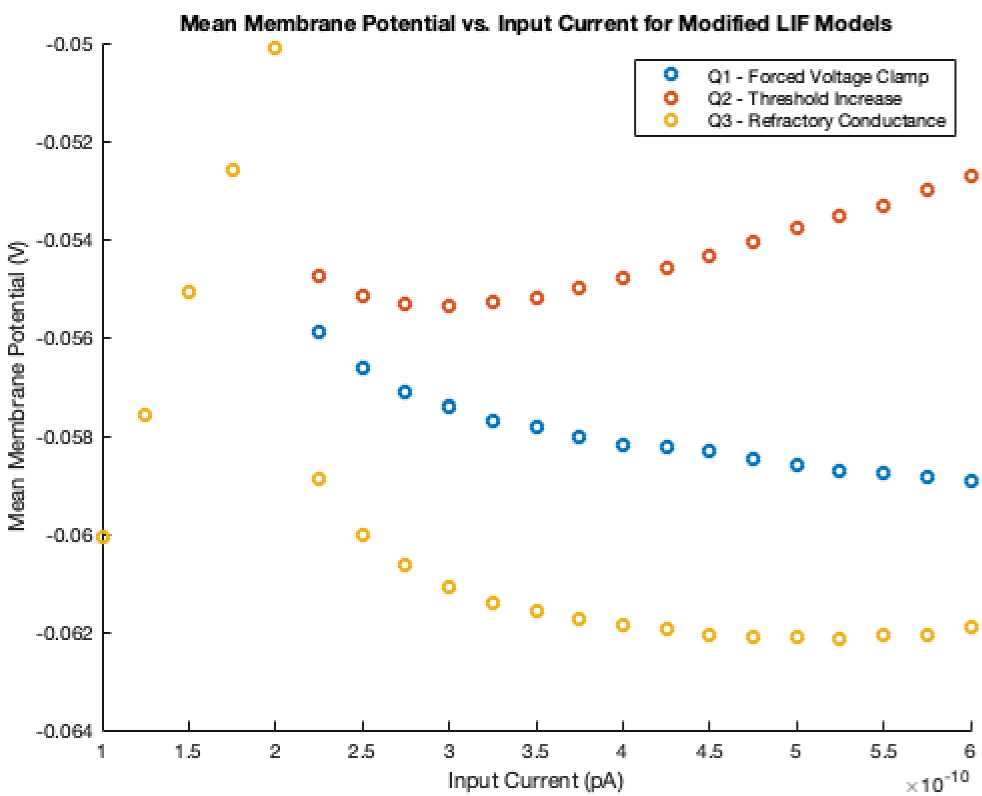
```
for n = 1:length(v)
    if v(n) > v_exceed
        count = count + 1;
    end
end
end
```

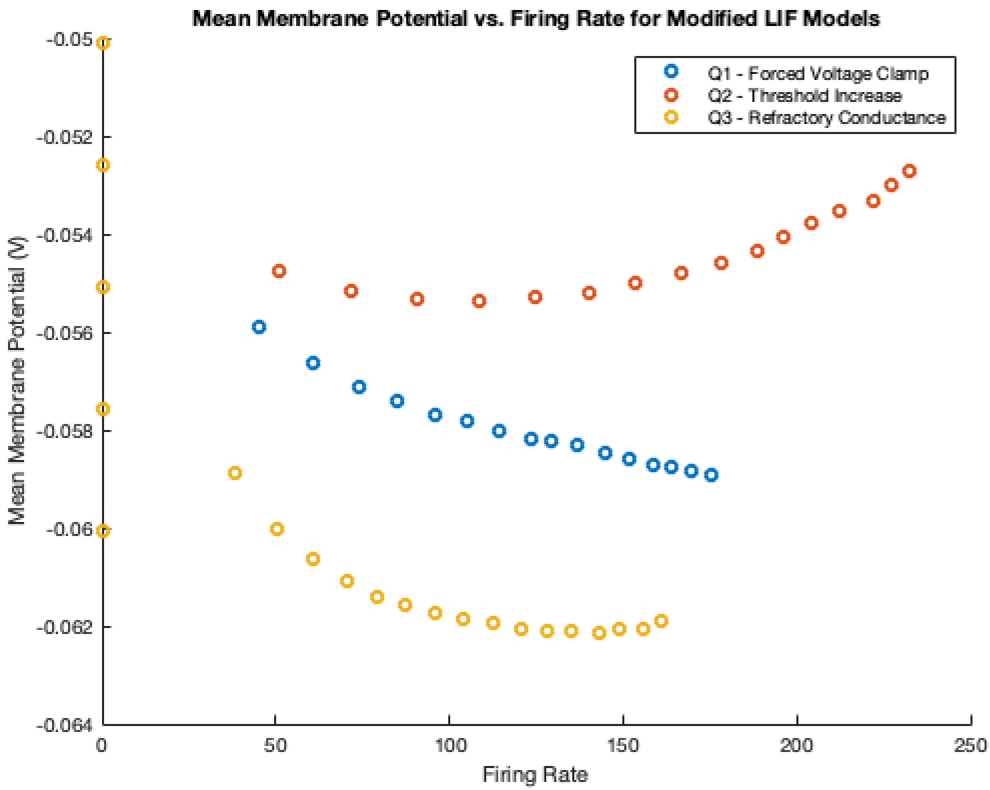



Q2 - Threshold Increase
I_{app} = 220 pA
I_{app} = 600 pA
Membrane Potential (mV)
0.04
0.02
0
-0.02
-0.04
-0.06
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Time (s)


Q3 - Refractory Conductance
I_{app} = 220 pA
I_{app} = 600 pA
Membrane Potential (mV)
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Time (s)


Mean Firing Rate vs. Input Current for Modified LIF Models
Q1 - Forced Voltage Clamp
Q2 - Threshold Increase
Q3 - Refractory Conductance
Mean Firing Rate (Hz)
Input Current (pA)
$\times 10^{-10}$


Mean Membrane Potential vs. Input Current for Modified LIF Models
Q1 - Forced Voltage Clamp
Q2 - Threshold Increase
Q3 - Refractory Conductance
Mean Membrane Potential (V)
Input Current (pA)
$\times 10^{-10}$



Published with MATLAB® R2018a