Amrita Shah AMS 332 Spring 2023

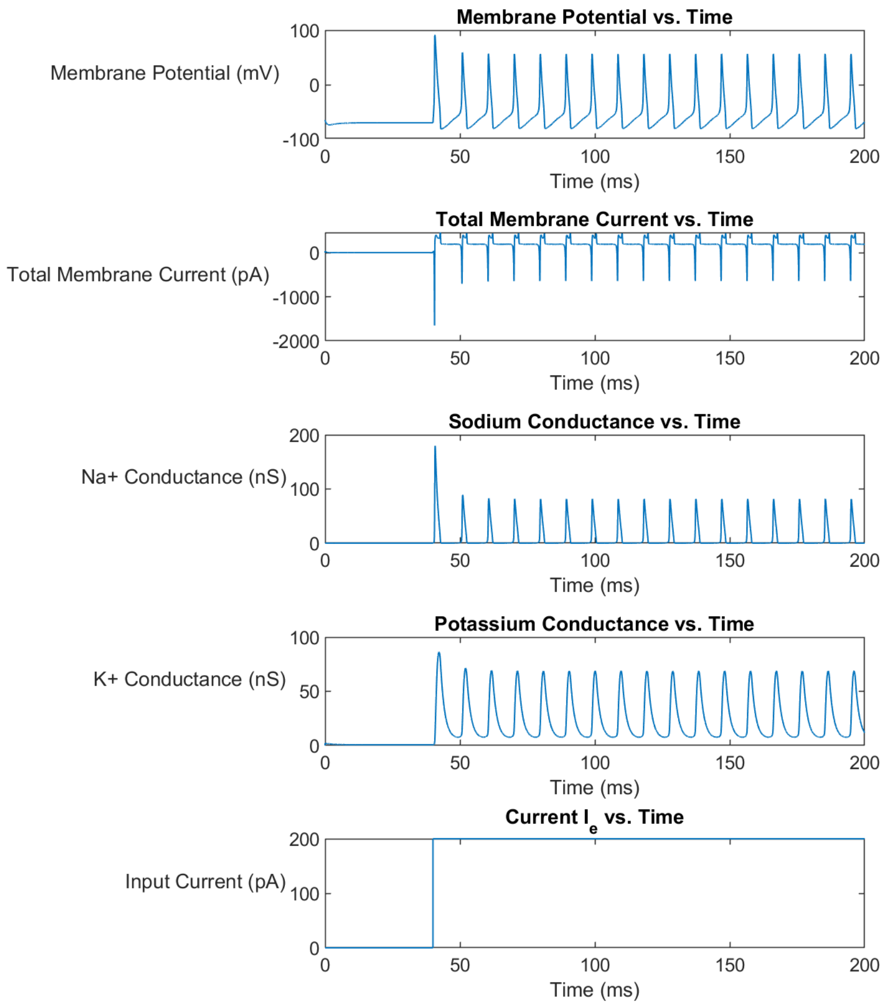
Unit 4: Modeling neurons, synapses, and cortical activity

Project 4 Report

Part A: Hodgkin-Huxley model

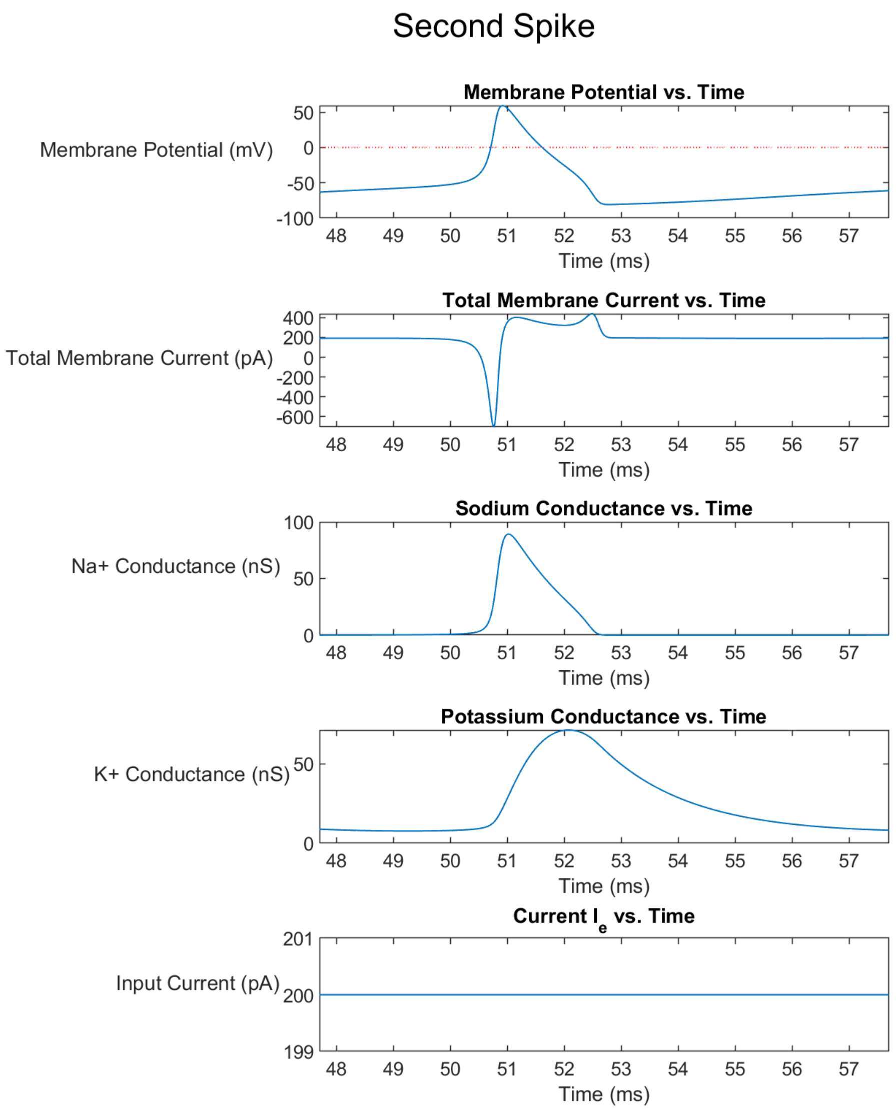
1. The evolution of one Hodgkin-Huxley neuron was simulated for 200 ms.

   Figure A1: Membrane potential in mV, total membrane current in pA, sodium conductance in nS, potassium conductance in nS, and current in pA were plotted over time in ms.



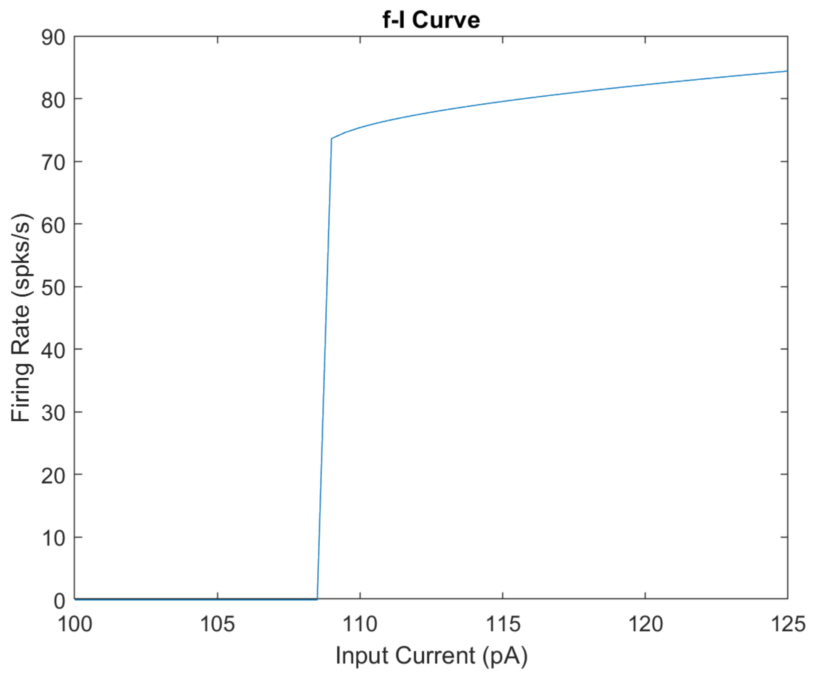
2. Spikes were detected and counted by determining a critical value V_{spk} of 0 mV. There were 17 spikes in this simulation over 200 ms.

   Figure A2: To illustrate the dynamics of one single action potential, this figure is an enlarged version of Figure A1. The second action potential is visualized from 47.7 ms to 57.7 ms. The value of V_{spk} is represented with a red line at 0 mV, showing when the critical value was reached as it intersects with the membrane potential plot. Membrane potential in mV, total membrane current in pA, sodium conductance in nS, potassium conductance in nS, and current in pA were plotted over time in ms.



3. The threshold current was found by manipulating the simulation from (2). The simulation was run with various current values until only one action potential was fired. After that, the current was decreased until the smallest current value was found at which the neuron fires only one action potential then stops firing. The value of the current I_1 was 18.43 pA.
4. To better estimate the smallest value of the current above which repetitive firing ignites, the simulation was run over a longer period of time, 1000 ms. The current was then increased until repetitive firing ignited (at least for the first 1000 ms). The smallest value of the current I_{rh} which caused repetitive firing was 108.62 pA.
5. The 1000 ms simulation was run 51 times for current values from 100 pA to 125 pA in increments of 0.5 pA. For each run, the number of spikes and their times were recorded. The inter-spike-intervals (ISIs) between the spikes were calculated. The firing rate for each run with repetitive firing is equal to the inverse of the average ISI.

   Figure A5: The f-I curve was plotted as firing rate in spikes per second, as a function of input current in pA.



i) The f-I curve (Figure A5) exhibits type II firing because it is discontinuous in the change of the firing rate near the bifurcation point. The firing rate is zero below the bifurcation then jumps to finite positive values and continues to gradually increase.

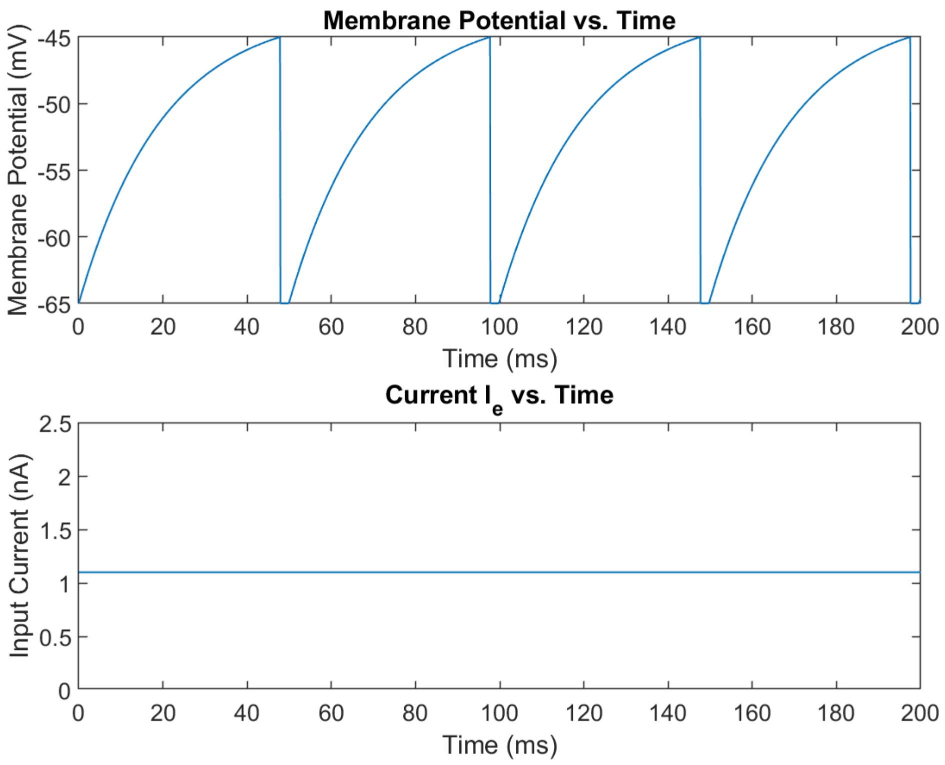
ii) The f-I curve can be traced in MATLAB to find the input current at which the firing rate jumps to positive values. From this, I_{rh} can be estimated to be 108.5 pA.

iii) The f-I curve suggests that using the model neuron to encode a graded input into the neuron’s output firing rate would not be efficient. This is because of the nonlinear, discontinuous relationship between firing rate and input current. For many input values, the firing rate is zero. The steep slope of the f-I curve near the bifurcation point also associates small changes in input current with large changes in firing rate. When varying the input current over a wide range, the output firing rate would not be an accurate representation of the input.

Part B: Leaky Integrate-and-Fire neuron model

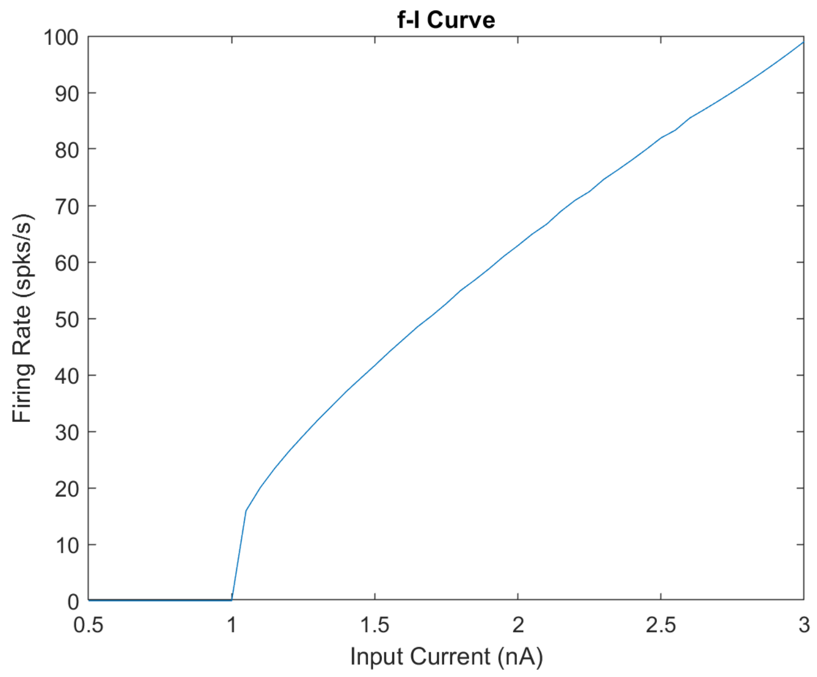
1. The evolution of a Leaky Integrate-and-Fire neuron was simulated for 200 ms. When V hit the V_{spk} threshold of -45 mV, a spike was said to occur. At each of these points in time, membrane potential was reset to V_r, the resting potential of -65 mV, for an absolute refractory period τ_{arp} of 2 ms before resuming LIF dynamics.

   Figure B1: Membrane potential in mV and external current in nA were plotted over time in ms.



2. The 200 ms simulation was run 51 times for current values from 0.5 nA to 3.0 nA in increments of 0.05 nA. For each run, the number of spikes and their times were recorded. The inter-spike-intervals (ISIs) between the spikes were calculated. The firing rate for each run is equal to the inverse of the average ISI.

Figure B2: The f-I curve was plotted as firing rate in spikes per second, as a function of input current in nA.



i) The f-I curve (Figure B2) exhibits type I firing because it is continuous in the change of the firing rate, unlike that in Figure A5. At larger current values (nA instead of pA), the firing rate of the neuron increases gradually from zero above the bifurcation point. Unlike type II firing, type I firing allows for repetitive firing to occur at arbitrarily small firing rates.

ii) It is not possible for the LIF neuron to have only a few action potentials and then stop firing. This is because a saddle-node bifurcation involves the coalescence and disappearance of a stable and an unstable node. This leaves the limit cycle, which describes repetitive firing, as the only possible dynamical behavior.

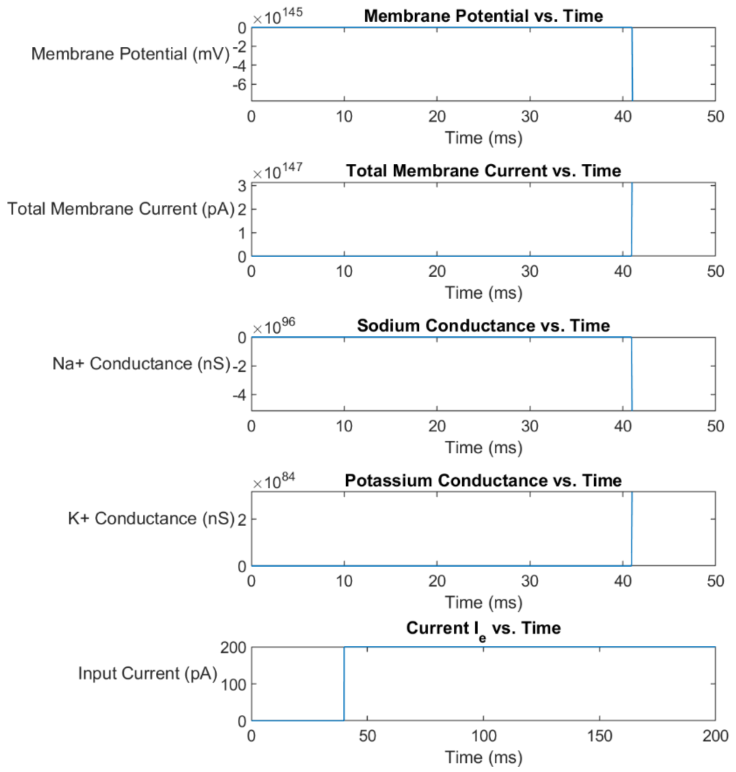
iii) The f-I curve can be traced in MATLAB to find the rheobase current. From this, ${I_{rh}}^{LIF}$ can be estimated to be 1.0 nA. The theoretical rheobase current can be calculated as $I_{rh}{}^{LIF} = G_L\left(V_{spk} - V_L\right) \rightarrow \frac{50}{1000}\,\mu\text{S}\,\left(-45\text{ mV} - (-65\text{ mV})\right) = 1.0\text{ nA}$.

The measured and theoretical values are equal.

3. Running time

i) The Hodgkin-Huxley model simulations do not work if using a timestep of 0.1 ms or 0.05 ms.

Figure B3: The evolution of a HH neuron simulated with a timestep of 0.05 ms. Membrane potential in mV, total membrane current in pA, sodium conductance in nS, potassium conductance in nS, and current in pA were plotted over time in ms.



ii) The Leaky Integrate-and-Fire neuron model simulation results did not change significantly when using a timestep of 0.2 ms instead of 0.1 ms. The number of spikes and their times was the same for both simulations.

iii) For a timestep of 0.01 ms, the HH model simulation takes 0.20 seconds, while the LIF model simulation takes 0.09 seconds. In that case the LIF simulation is roughly twice as fast as the HH simulation. For a timestep of 0.002 ms, the HH model simulation takes 0.25 seconds, while the LIF model simulation takes 0.08 seconds. In that case the LIF simulation is roughly thrice as fast as the HH simulation. The HH simulation consistently runs much slower than the LIF simulation, and the difference is apparent.

Part A: Hodgkin-Huxley model

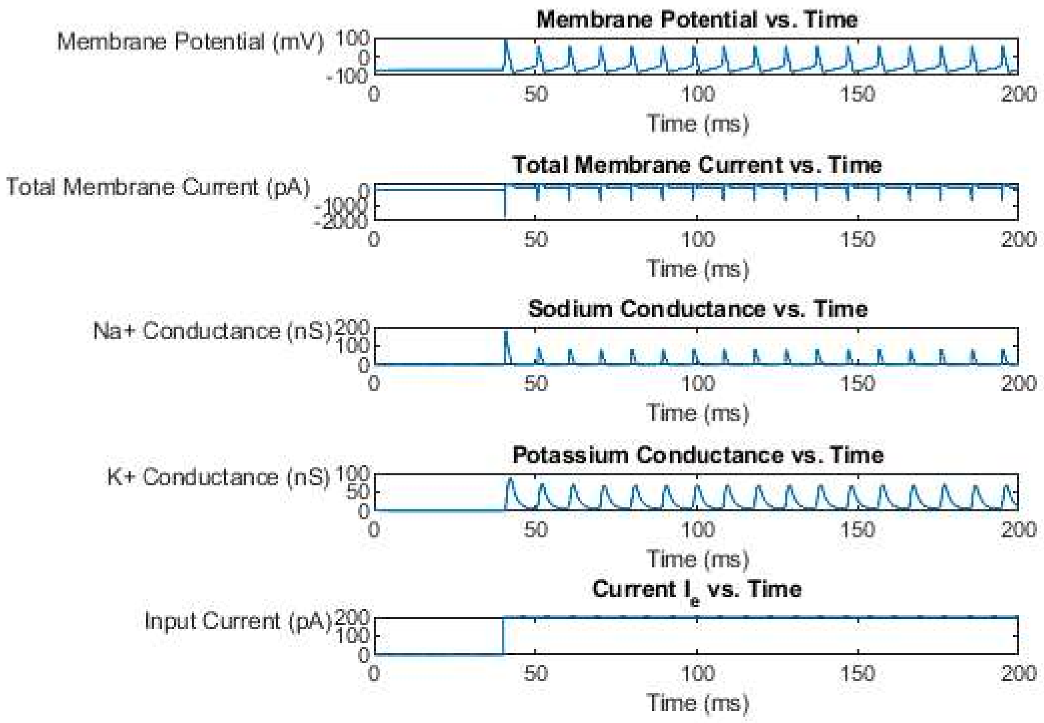
Contents

1. Main simulation

```
clear
close all

GNa = 400; % nS
GK = 200; % nS
GL = 2; % nS
ENa = 99; % mV
EK = -85; % mV
VL = -65; % mV
C = 2; % pF

maxT = 200; % ms
deltaT = 0.01; % ms
Tswap = 40;
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

V = zeros(size(time));
m = zeros(size(time));
h = zeros(size(time));
n = zeros(size(time));

V(1) = VL;

Ie = zeros(size(time));
Ie((Tswap/deltaT)+1:end) = 200;

Im = zeros(size(time));
Na_cond = zeros(size(time));
K_cond = zeros(size(time));

Im(1) = GL*(V(1)-VL) + GNa*(m(1)^3)*h(1)*(V(1)-ENa) + GK*(n(1)^4)*(V(1)-EK);
Na_cond(1) = GNa*(m(1)^3)*h(1);
K_cond(1) = GK*(n(1)^4);

% transition rates for V(1)
a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
a_h = 0.07 * exp(-0.05*(V(1)+65));
a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
b_m = 4 * exp(-0.0556*(V(1)+65));
b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
b_n = 0.125 * exp(-0.0125*(V(1)+65));
```

```
m_inf = a_m / (a_m + b_m);
h_inf = a_h / (a_h + b_h);
n_inf = a_n / (a_n + b_n);

m(1) = m_inf;
h(1) = h_inf;
n(1) = n_inf;

for i = 1:numiterations

    % transition rates for this V
    a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
    a_h = 0.07 * exp(-0.05*(V(i)+65));
    a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
    b_m = 4 * exp(-0.0556*(V(i)+65));
    b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
    b_n = 0.125 * exp(-0.0125*(V(i)+65));

    dm_dt = a_m*(1-m(i)) - b_m*(m(i));
    dh_dt = a_h*(1-h(i)) - b_h*(h(i));
    dn_dt = a_n*(1-n(i)) - b_n*(n(i));

    m(i+1) = m(i) + dm_dt*deltaT;
    h(i+1) = h(i) + dh_dt*deltaT;
    n(i+1) = n(i) + dn_dt*deltaT;

    dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;

    Im(i+1) = GL*(V(i+1)-VL) + GNa*(m(i+1)^3)*h(i+1)*(V(i+1)-ENa) + GK*(n(i+1)^4)*(V(i+1)-EK);
    Na_cond(i+1) = GNa*(m(i+1)^3)*h(i+1);
    K_cond(i+1) = GK*(n(i+1)^4);

end

fig1 = figure(1);
subplot(5,1,1)
plot(time, V)
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Membrane Potential vs. Time')

subplot(5,1,2)
plot(time, Im)
xlabel('Time (ms)')
ylabel('Total Membrane Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Total Membrane Current vs. Time')

subplot(5,1,3)
plot(time,Na_cond)
xlabel('Time (ms)')
ylabel('Na+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Sodium Conductance vs. Time')

subplot(5,1,4)
plot(time,K_cond)
xlabel('Time (ms)')
```

```
ylabel('K+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Potassium Conductance vs. Time')

subplot(5,1,5)
plot(time, Ie)
xlabel('Time (ms)')
ylabel('Input Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Current I_{e} vs. Time')
```



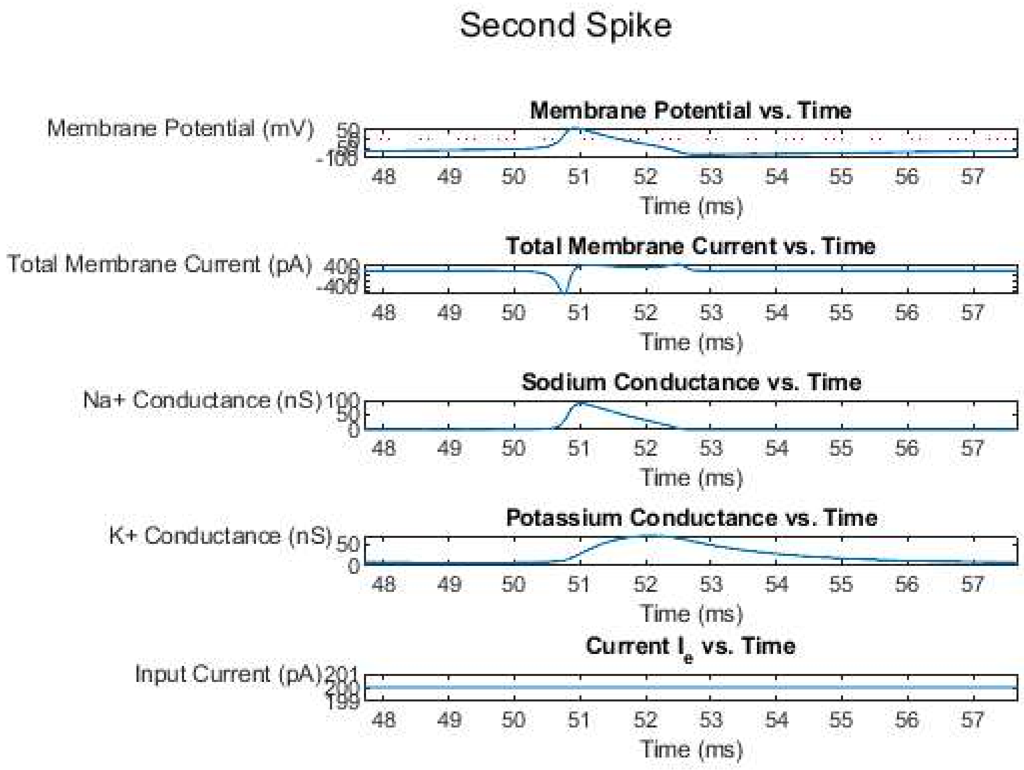
2. Spike detection

```
clear
close all

GNa = 400; % nS
GK = 200; % nS
GL = 2; % nS
ENa = 99; % mV
EK = -85; % mV
VL = -65; % mV
C = 2; % pF
Vspk = 0; % mV

maxT = 200; % ms
deltaT = 0.01; % ms
Tswap = 40;
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

spikes = 0;
Tspikes = [];
```

```
V = zeros(size(time));
m = zeros(size(time));
h = zeros(size(time));
n = zeros(size(time));

V(1) = VL;

Ie = zeros(size(time));
Ie((Tswap/deltaT)+1:end) = 200;

Im = zeros(size(time));
Na_cond = zeros(size(time));
K_cond = zeros(size(time));

Im(1) = GL*(V(1)-VL) + GNa*(m(1)^3)*h(1)*(V(1)-ENa) + GK*(n(1)^4)*(V(1)-EK);
Na_cond(1) = GNa*(m(1)^3)*h(1);
K_cond(1) = GK*(n(1)^4);

% transition rates for V(1)
a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
a_h = 0.07 * exp(-0.05*(V(1)+65));
a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
b_m = 4 * exp(-0.0556*(V(1)+65));
b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
b_n = 0.125 * exp(-0.0125*(V(1)+65));

m_inf = a_m / (a_m + b_m);
h_inf = a_h / (a_h + b_h);
n_inf = a_n / (a_n + b_n);

m(1) = m_inf;
h(1) = h_inf;
n(1) = n_inf;

for i = 1:numiterations

    % transition rates for this V
    a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
    a_h = 0.07 * exp(-0.05*(V(i)+65));
    a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
    b_m = 4 * exp(-0.0556*(V(i)+65));
    b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
    b_n = 0.125 * exp(-0.0125*(V(i)+65));

    dm_dt = a_m*(1-m(i)) - b_m*(m(i));
    dh_dt = a_h*(1-h(i)) - b_h*(h(i));
    dn_dt = a_n*(1-n(i)) - b_n*(n(i));

    m(i+1) = m(i) + dm_dt*deltaT;
    h(i+1) = h(i) + dh_dt*deltaT;
    n(i+1) = n(i) + dn_dt*deltaT;

    dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;
    if V(i+1) > Vspk && V(i) < Vspk
        spikes = spikes+1;
        Tspikes(end+1) = i/100;
    end

    Im(i+1) = GL*(V(i+1)-VL) + GNa*(m(i+1)^3)*h(i+1)*(V(i+1)-ENa) + GK*(n(i+1)^4)*(V(i+1)-EK);
    Na_cond(i+1) = GNa*(m(i+1)^3)*h(i+1);
    K_cond(i+1) = GK*(n(i+1)^4);

end

fig2 = figure(2);
sgtitle('Second Spike')

subplot(5,1,1)
plot(time, V)
xlim([47.7 57.7])
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Membrane Potential vs. Time')
hold on
plot(time, zeros(size(V)), ':r')
hold off

subplot(5,1,2)
plot(time, Im)
xlim([47.7 57.7])
xlabel('Time (ms)')
ylabel('Total Membrane Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Total Membrane Current vs. Time')

subplot(5,1,3)
plot(time,Na_cond)
xlim([47.7 57.7])
xlabel('Time (ms)')
ylabel('Na+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Sodium Conductance vs. Time')

subplot(5,1,4)
plot(time,K_cond)
xlim([47.7 57.7])
xlabel('Time (ms)')
ylabel('K+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Potassium Conductance vs. Time')

subplot(5,1,5)
plot(time, Ie)
xlim([47.7 57.7])
xlabel('Time (ms)')
ylabel('Input Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Current I_{e} vs. Time')

spikes
Tspikes
```

```
spikes =

    17
```

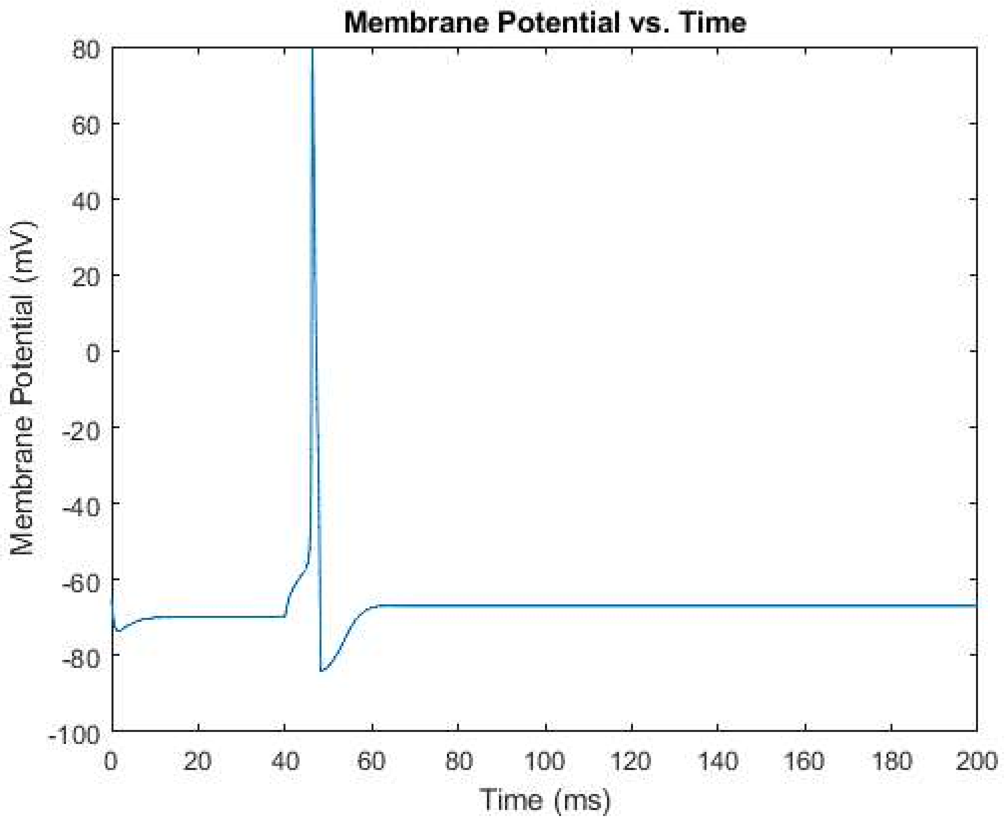
```
Tspikes =

  Columns 1 through 7

   40.5200   50.7100   60.3700   69.9700   79.5700   89.1700   98.7700

  Columns 8 through 14

  108.3700  117.9600  127.5600  137.1600  146.7500  156.3500  165.9500

  Columns 15 through 17

  175.5500  185.1400  194.7400
```



3. Action potential threshold

```
clear
close all

GNa = 400; % nS
GK = 200; % nS
GL = 2; % nS
ENa = 99; % mV
EK = -85; % mV
VL = -65; % mV
C = 2; % pF

maxT = 200; % ms
deltaT = 0.01; % ms
Tswap = 40;
```

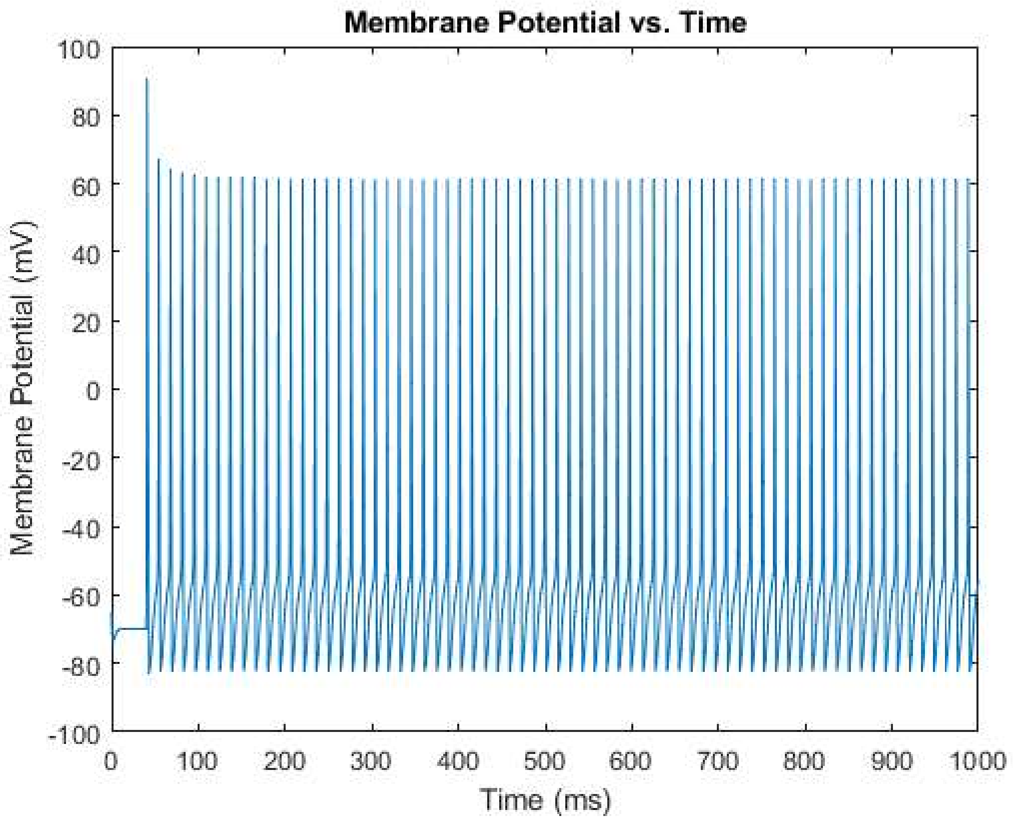
```matlab
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

V = zeros(size(time));
m = zeros(size(time));
h = zeros(size(time));
n = zeros(size(time));

V(1) = VL;

Ie = zeros(size(time));
Ie((Tswap/deltaT)+1:end) = 18.43;

% transition rates for V(1)
a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
a_h = 0.07 * exp(-0.05*(V(1)+65));
a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
b_m = 4 * exp(-0.0556*(V(1)+65));
b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
b_n = 0.125 * exp(-0.0125*(V(1)+65));

m_inf = a_m / (a_m + b_m);
h_inf = a_h / (a_h + b_h);
n_inf = a_n / (a_n + b_n);

m(1) = m_inf;
h(1) = h_inf;
n(1) = n_inf;

for i = 1:numiterations

    % transition rates for this V
    a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
    a_h = 0.07 * exp(-0.05*(V(i)+65));
    a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
    b_m = 4 * exp(-0.0556*(V(i)+65));
    b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
    b_n = 0.125 * exp(-0.0125*(V(i)+65));

    dm_dt = a_m*(1-m(i)) - b_m*(m(i));
    dh_dt = a_h*(1-h(i)) - b_h*(h(i));
    dn_dt = a_n*(1-n(i)) - b_n*(n(i));

    m(i+1) = m(i) + dm_dt*deltaT;
    h(i+1) = h(i) + dh_dt*deltaT;
    n(i+1) = n(i) + dn_dt*deltaT;

    dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;

end

fig3 = figure(3);
plot(time, V)
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)')
title('Membrane Potential vs. Time')
```



4. Rheobase current

```
clear
close all

GNa = 400; % nS
GK = 200; % nS
GL = 2; % nS
ENa = 99; % mV
EK = -85; % mV
VL = -65; % mV
C = 2; % pF

maxT = 1000; % ms
deltaT = 0.01; % ms
Tswap = 40;
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

V = zeros(size(time));
m = zeros(size(time));
h = zeros(size(time));
n = zeros(size(time));

V(1) = VL;

Ie = zeros(size(time));
Ie((Tswap/deltaT)+1:end) = 108.62;

% transition rates for V(1)
a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
```

```
a_h = 0.07 * exp(-0.05*(V(1)+65));
a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
b_m = 4 * exp(-0.0556*(V(1)+65));
b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
b_n = 0.125 * exp(-0.0125*(V(1)+65));

m_inf = a_m / (a_m + b_m);
h_inf = a_h / (a_h + b_h);
n_inf = a_n / (a_n + b_n);

m(1) = m_inf;
h(1) = h_inf;
n(1) = n_inf;

for i = 1:numiterations

    % transition rates for this V
    a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
    a_h = 0.07 * exp(-0.05*(V(i)+65));
    a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
    b_m = 4 * exp(-0.0556*(V(i)+65));
    b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
    b_n = 0.125 * exp(-0.0125*(V(i)+65));

    dm_dt = a_m*(1-m(i)) - b_m*(m(i));
    dh_dt = a_h*(1-h(i)) - b_h*(h(i));
    dn_dt = a_n*(1-n(i)) - b_n*(n(i));

    m(i+1) = m(i) + dm_dt*deltaT;
    h(i+1) = h(i) + dh_dt*deltaT;
    n(i+1) = n(i) + dn_dt*deltaT;

    dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;

end

fig4 = figure(4);
plot(time, V)
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)')
title('Membrane Potential vs. Time')
```

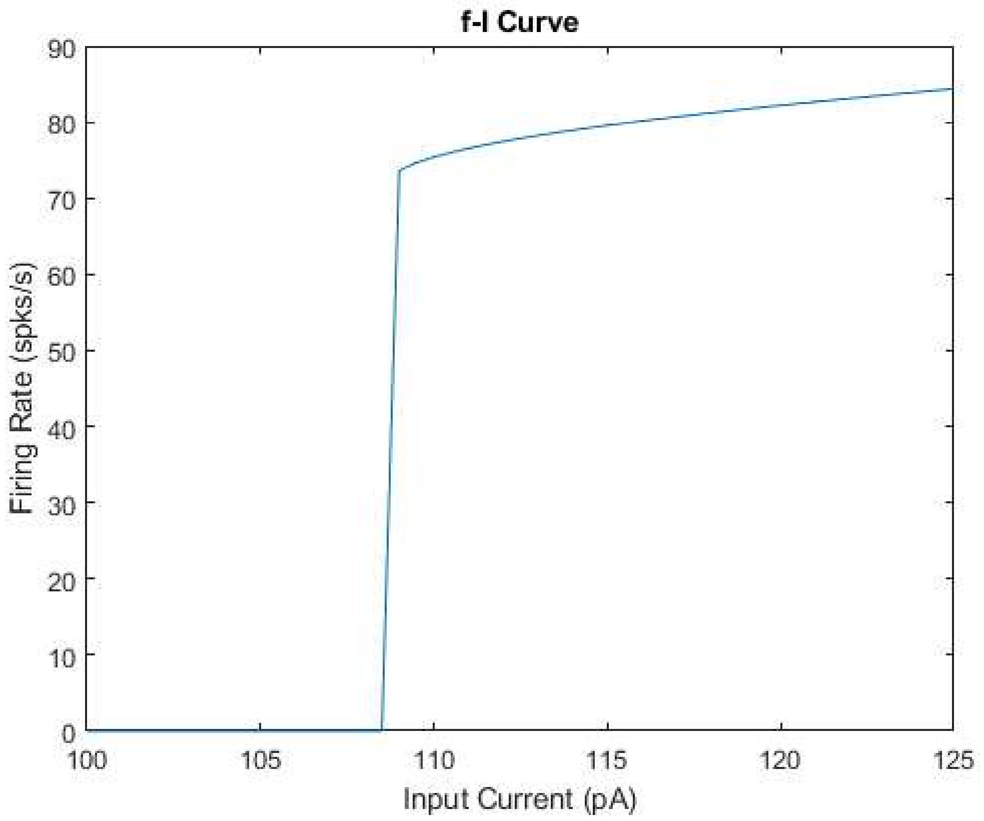


5. f-I curve

```
clear
close all

runs = 100:0.5:125;
fr = zeros(1, length(runs));

for r = 1:length(runs)

    GNa = 400; % nS
    GK = 200; % nS
    GL = 2; % nS
    ENa = 99; % mV
    EK = -85; % mV
    VL = -65; % mV
    C = 2; % pF
    Vspk = 0; % mV

    maxT = 1000; % ms
    deltaT = 0.01; % ms
    Tswap = 40;
    numiterations = maxT/deltaT;
    time = 0:deltaT:maxT;

    spikes = 0;
    Tspikes = [];

    V = zeros(size(time));
    m = zeros(size(time));
    h = zeros(size(time));
```

```
        n = zeros(size(time));

        V(1) = VL;

        Ie = zeros(size(time));
        Ie((Tswap/deltaT)+1:end) = runs(r);

        % transition rates for V(1)
        a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
        a_h = 0.07 * exp(-0.05*(V(1)+65));
        a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
        b_m = 4 * exp(-0.0556*(V(1)+65));
        b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
        b_n = 0.125 * exp(-0.0125*(V(1)+65));

        m_inf = a_m / (a_m + b_m);
        h_inf = a_h / (a_h + b_h);
        n_inf = a_n / (a_n + b_n);

        m(1) = m_inf;
        h(1) = h_inf;
        n(1) = n_inf;

        for i = 1:numiterations

            % transition rates for this V
            a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
            a_h = 0.07 * exp(-0.05*(V(i)+65));
            a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
            b_m = 4 * exp(-0.0556*(V(i)+65));
            b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
            b_n = 0.125 * exp(-0.0125*(V(i)+65));

            dm_dt = a_m*(1-m(i)) - b_m*(m(i));
            dh_dt = a_h*(1-h(i)) - b_h*(h(i));
            dn_dt = a_n*(1-n(i)) - b_n*(n(i));

            m(i+1) = m(i) + dm_dt*deltaT;
            h(i+1) = h(i) + dh_dt*deltaT;
            n(i+1) = n(i) + dn_dt*deltaT;

            dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
            V(i+1) = V(i) + dV_dt*deltaT;
            if V(i+1) > Vspk && V(i) < Vspk
                spikes = spikes+1;
                Tspikes(end+1) = i/100;
            end
        end

        if spikes < 15
            fr(r) = 0;
        else
            fr(r) = 1/(mean(diff(Tspikes)));
        end
    end

    fig5 = figure(5);
    plot(runs, fr*1000)
```

```
xlabel('Input Current (pA)')
ylabel('Firing Rate (spks/s)')
title('f-I Curve')
```



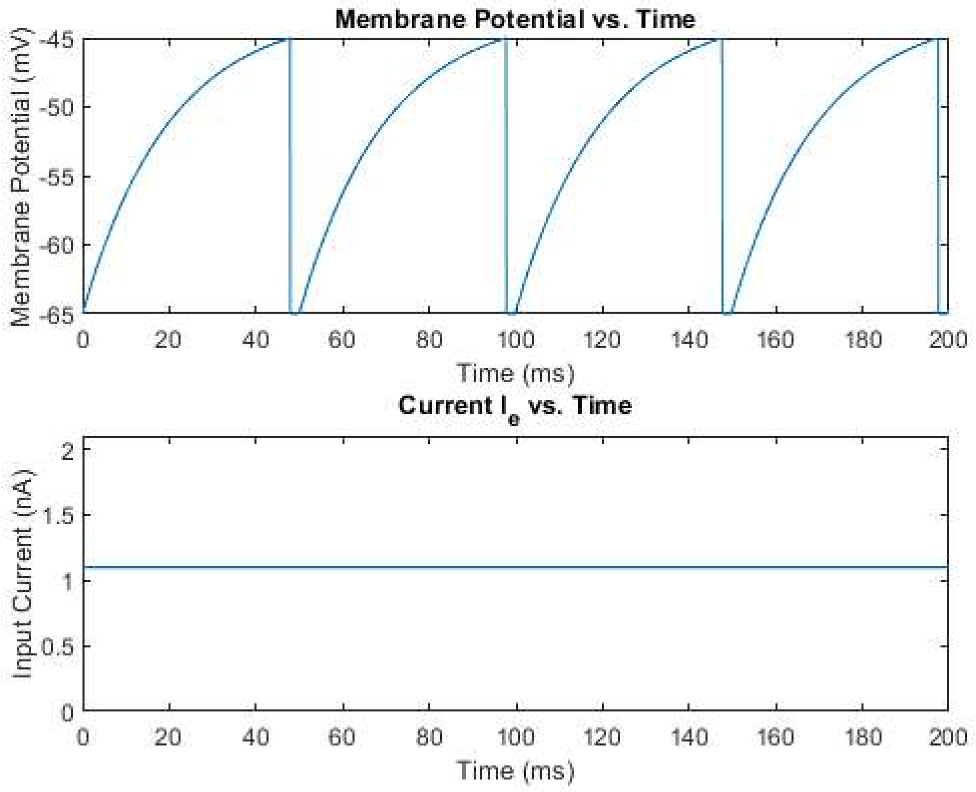
Part B: Leaky Integrate-and-Fire neuron model

Contents

1. Main simulation

```
clear
close all

GL = 50/1000; % uS
VL = -65; % mV
C = 1; % nF
Vspk = -45; % mV
Vr = -65; % mV
Tau = 2; % ms

maxT = 200; % ms
deltaT = 0.1; % ms
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

spikes = 0;
Tspikes = [];
V = zeros(size(time));
V(1) = VL;

Ie = 1.1*ones(size(time));

for i = 1:numiterations
    if V(i+1) == Vr
        continue
    end
    dV_dt = ((-GL*(V(i)-VL)) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;
    if V(i+1) > Vspk && V(i) < Vspk
        spikes = spikes+1;
        Tspikes(end+1) = i/10;
        V((i+1):(i+1+(Tau/deltaT))) = Vr;
    end
end

fig1 = figure(1);
subplot(2,1,1)
plot(time, V(1:numiterations+1))
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)')
title('Membrane Potential vs. Time')

subplot(2,1,2)
plot(time, Ie)
xlabel('Time (ms)')
```

```
ylabel('Input Current (nA)')
title('Current I_{e} vs. Time')
```



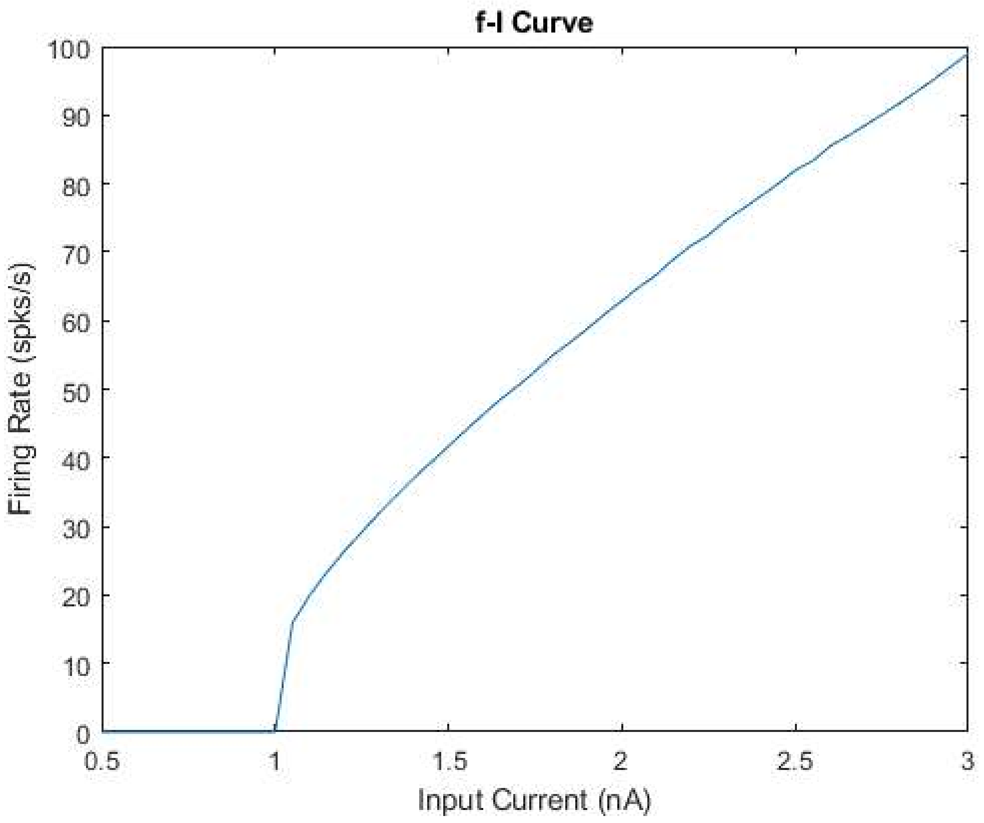
2. f-I curve

```
clear
close all

runs = 0.5:0.05:3;
fr = zeros(1, length(runs));

for r = 1:length(runs)
    GL = 50/1000; % uS
    VL = -65; % mV
    C = 1; % nF
    Vspk = -45; % mV
    Vr = -65; % mV
    Tau = 2; % ms

    maxT = 1000; % ms
    deltaT = 0.1; % ms
    numiterations = maxT/deltaT;
    time = 0:deltaT:maxT;

    spikes = 0;
    Tspikes = [];
    V = zeros(size(time));
    V(1) = VL;

    Ie = runs(r);
    for i = 1:numiterations
        if V(i+1) == Vr
            continue
        end
        dV_dt = ((-GL*(V(i)-VL)) + Ie) / C;
        V(i+1) = V(i) + dV_dt*deltaT;
        if V(i+1) > Vspk && V(i) < Vspk
            spikes = spikes+1;
            Tspikes(end+1) = i/10;
            V((i+1):(i+1+(Tau/deltaT))) = Vr;
        end
    end
    spikes
    if spikes < 1
        fr(r) = 0;
    else
        fr(r) = 1/(mean(diff(Tspikes)));
    end
end

fig5 = figure(5);
plot(runs, fr*1000)
xlabel('Input Current (nA)')
ylabel('Firing Rate (spks/s)')
title('f-I Curve')
```

```
spikes =

     0


spikes =

     0


spikes =

     0


spikes =

     0


spikes =

     0


spikes =

     0


spikes =
     0


spikes =

     0


spikes =

     0


spikes =

     0


spikes =

     0


spikes =

    15


spikes =

    20


spikes =

    23


spikes =

    26


spikes =

    29


spikes =

    32


spikes =

    34
```

```
spikes =

    37


spikes =

    39


spikes =

    41


spikes =

    44


spikes =

    46


spikes =

    48


spikes =

    50


spikes =

    52


spikes =

    55


spikes =

    56


spikes =

    58


spikes =
    61


spikes =

    63


spikes =

    65


spikes =

    66


spikes =

    69


spikes =

    71


spikes =

    72


spikes =

    74


spikes =

    76


spikes =

    78


spikes =

    80


spikes =

    82
```

```
spikes =

    83


spikes =

    85


spikes =

    87


spikes =

    88


spikes =

    90


spikes =

    91


spikes =

    93


spikes =

    95


spikes =

    97


spikes =

    99
```



Published with MATLAB® R2022b

Part B (#3)

Contents

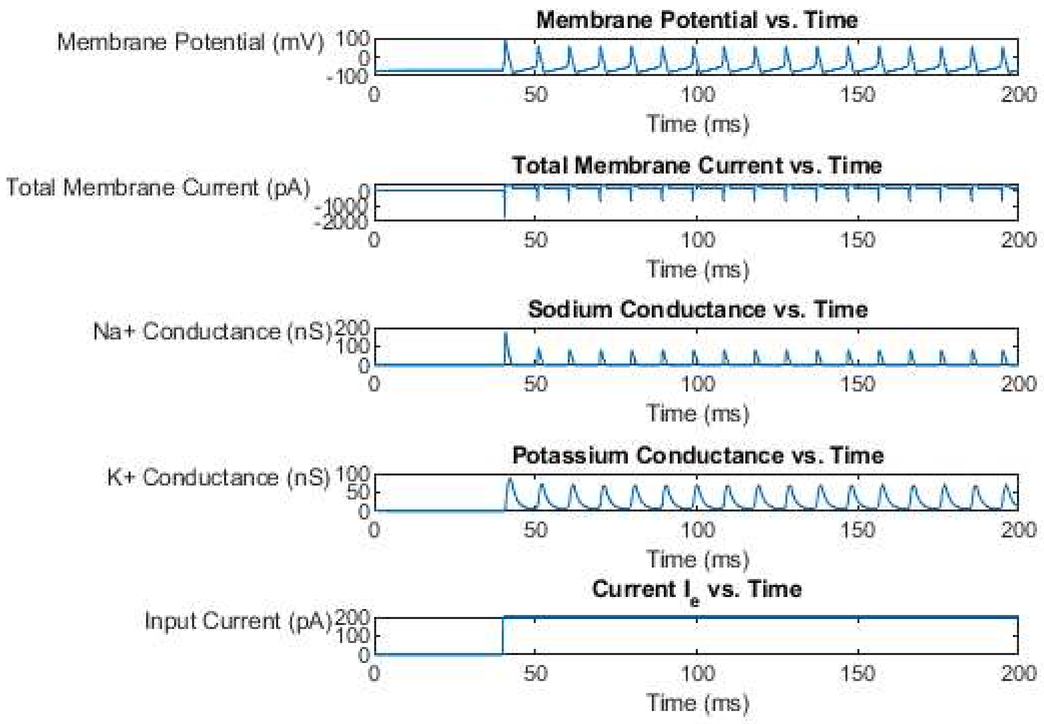
Hodgkin-Huxley

```
clear
close all

tic

GNa = 400; % nS
GK = 200; % nS
GL = 2; % nS
ENa = 99; % mV
EK = -85; % mV
VL = -65; % mV
C = 2; % pF

maxT = 200; % ms
deltaT = 0.002; % ms
Tswap = 40;
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;

V = zeros(size(time));
m = zeros(size(time));
h = zeros(size(time));
n = zeros(size(time));

V(1) = VL;

Ie = zeros(size(time));
Ie((Tswap/deltaT)+1:end) = 200;

Im = zeros(size(time));
Na_cond = zeros(size(time));
K_cond = zeros(size(time));

Im(1) = GL*(V(1)-VL) + GNa*(m(1)^3)*h(1)*(V(1)-ENa) + GK*(n(1)^4)*(V(1)-EK);
Na_cond(1) = GNa*(m(1)^3)*h(1);
K_cond(1) = GK*(n(1)^4);

% transition rates for V(1)
a_m = (0.1*(V(1)+40)) / (1- exp(-0.1*(V(1)+40)));
a_h = 0.07 * exp(-0.05*(V(1)+65));
a_n = (0.01*(V(1)+55)) / (1- exp(-0.1*(V(1)+55)));
b_m = 4 * exp(-0.0556*(V(1)+65));
b_h = 1 / (1+ exp(-0.1*(V(1)+35)));
b_n = 0.125 * exp(-0.0125*(V(1)+65));

m_inf = a_m / (a_m + b_m);
```

```
h_inf = a_h / (a_h + b_h);
n_inf = a_n / (a_n + b_n);

m(1) = m_inf;
h(1) = h_inf;
n(1) = n_inf;

for i = 1:numiterations

    % transition rates for this V
    a_m = (0.1*(V(i)+40)) / (1- exp(-0.1*(V(i)+40)));
    a_h = 0.07 * exp(-0.05*(V(i)+65));
    a_n = (0.01*(V(i)+55)) / (1- exp(-0.1*(V(i)+55)));
    b_m = 4 * exp(-0.0556*(V(i)+65));
    b_h = 1 / (1+ exp(-0.1*(V(i)+35)));
    b_n = 0.125 * exp(-0.0125*(V(i)+65));

    dm_dt = a_m*(1-m(i)) - b_m*(m(i));
    dh_dt = a_h*(1-h(i)) - b_h*(h(i));
    dn_dt = a_n*(1-n(i)) - b_n*(n(i));

    m(i+1) = m(i) + dm_dt*deltaT;
    h(i+1) = h(i) + dh_dt*deltaT;
    n(i+1) = n(i) + dn_dt*deltaT;

    dV_dt = ( -GL*(V(i)-VL) - GNa*(m(i)^3)*h(i)*(V(i)-ENa) - GK*(n(i)^4)*(V(i)-EK) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;

    Im(i+1) = GL*(V(i+1)-VL) + GNa*(m(i+1)^3)*h(i+1)*(V(i+1)-ENa) + GK*(n(i+1)^4)*(V(i+1)-EK);
    Na_cond(i+1) = GNa*(m(i+1)^3)*h(i+1);
    K_cond(i+1) = GK*(n(i+1)^4);

end

fig1 = figure(1);
subplot(5,1,1)
plot(time, V)
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Membrane Potential vs. Time')

subplot(5,1,2)
plot(time, Im)
xlabel('Time (ms)')
ylabel('Total Membrane Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Total Membrane Current vs. Time')

subplot(5,1,3)
plot(time,Na_cond)
xlabel('Time (ms)')
ylabel('Na+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Sodium Conductance vs. Time')

subplot(5,1,4)
plot(time,K_cond)
xlabel('Time (ms)')
ylabel('K+ Conductance (nS)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Potassium Conductance vs. Time')
```

```
subplot(5,1,5)
plot(time, Ie)
xlabel('Time (ms)')
ylabel('Input Current (pA)', 'Rotation', 0, 'HorizontalAlignment','right')
title('Current I_{e} vs. Time')

toc
```

```
Elapsed time is 0.242156 seconds.
```

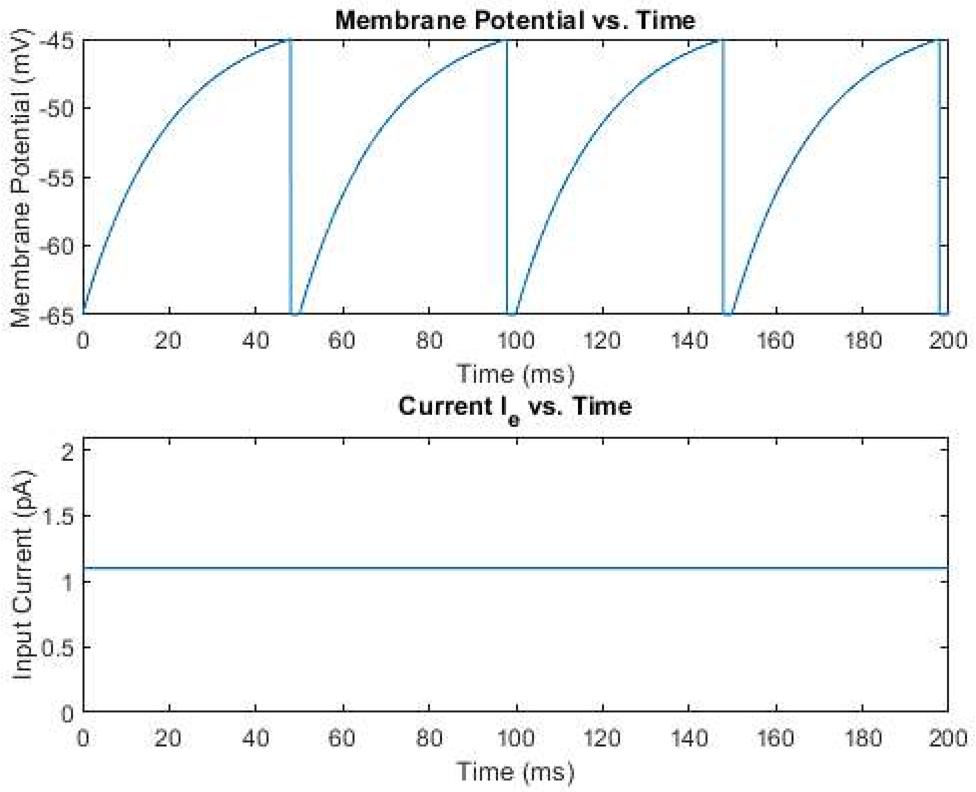


LIF

```
clear
close all

tic

GL = 50/1000; % uS
VL = -65; % mV
C = 1; % nF
Vspk = -45; % mV
Vr = -65; % mV
Tau = 2; % ms

maxT = 200; % ms
deltaT = 0.002; % ms
numiterations = maxT/deltaT;
time = 0:deltaT:maxT;
```

```
spikes = 0;
Tspikes = [];
V = zeros(size(time));
V(1) = VL;

Ie = 1.1*ones(size(time));

for i = 1:numiterations
    if V(i+1) == Vr
        continue
    end
    dV_dt = ((-GL*(V(i)-VL)) + Ie(i)) / C;
    V(i+1) = V(i) + dV_dt*deltaT;
    if V(i+1) > Vspk && V(i) < Vspk
        spikes = spikes+1;
        Tspikes(end+1) = i/10;
        V((i+1):(i+1+(Tau/deltaT))) = Vr;
    end
end

fig1 = figure(1);
subplot(2,1,1)
plot(time, V(1:numiterations+1))
xlabel('Time (ms)')
ylabel('Membrane Potential (mV)')
title('Membrane Potential vs. Time')

subplot(2,1,2)
plot(time, Ie)
xlabel('Time (ms)')
ylabel('Input Current (pA)')
title('Current I_{e} vs. Time')

toc
```

```
Elapsed time is 0.096215 seconds.
```



Published with MATLAB® R2022b