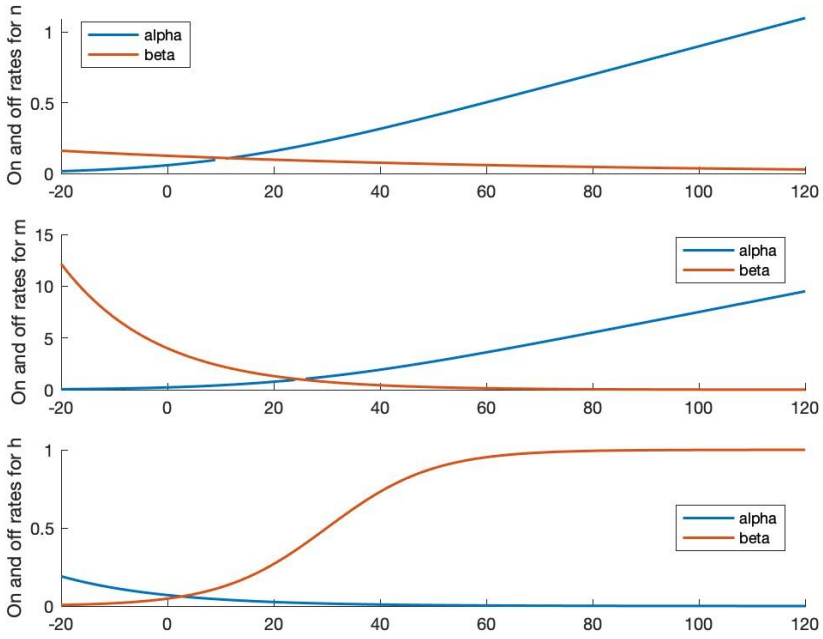
Robert Heeter
BIOE 446
3 November 2023

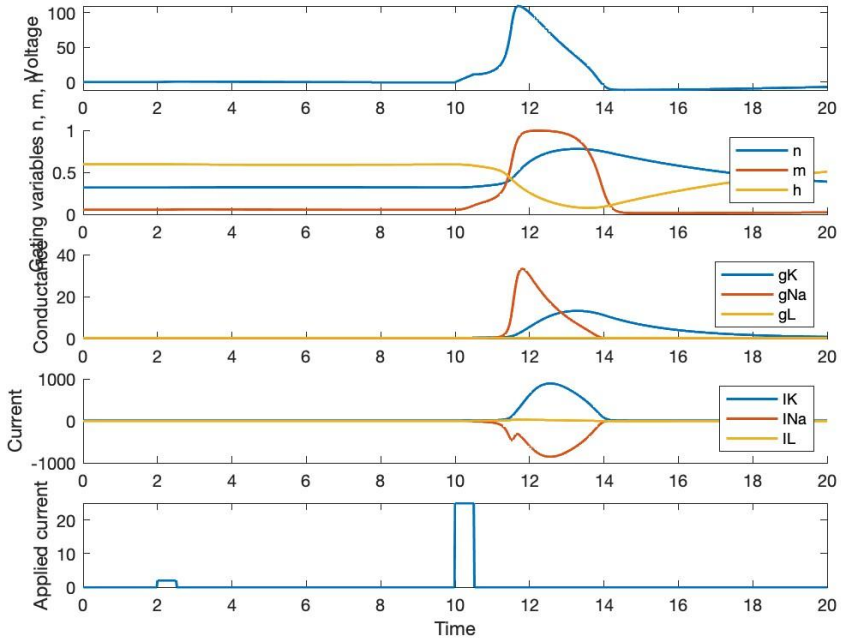
Lab 10: Signal Processing in Neurons

ANSWERS

1. **Simulating the Hodgkin-Huxley model (DEMO)**
   a. Figure



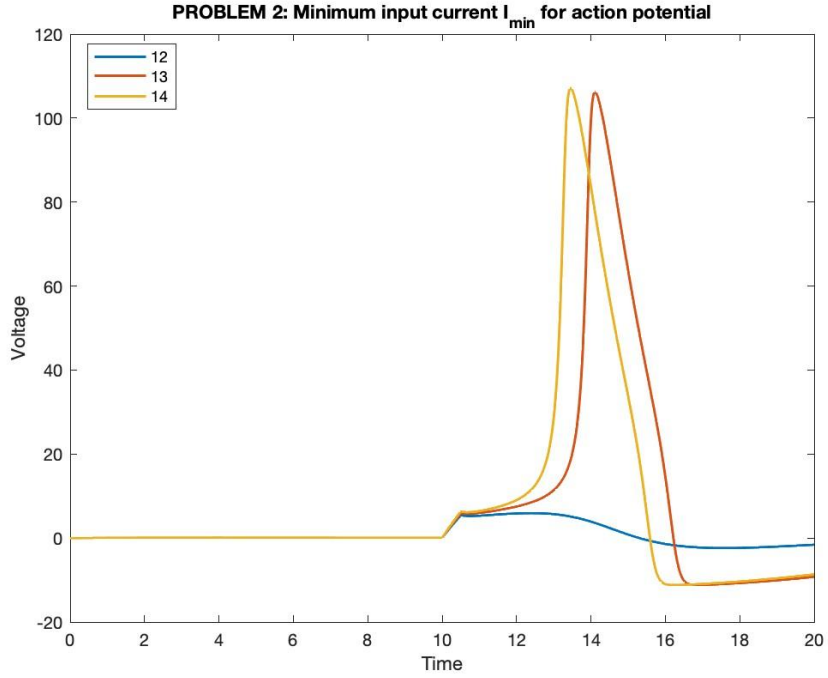
   b. Figure



2. Determining minimum stimulus to generate excitatory response

a. Minimum input current needed to stimulate an excitatory response in a neuron that is initially at rest = ~13 uA/cm^2

Figure

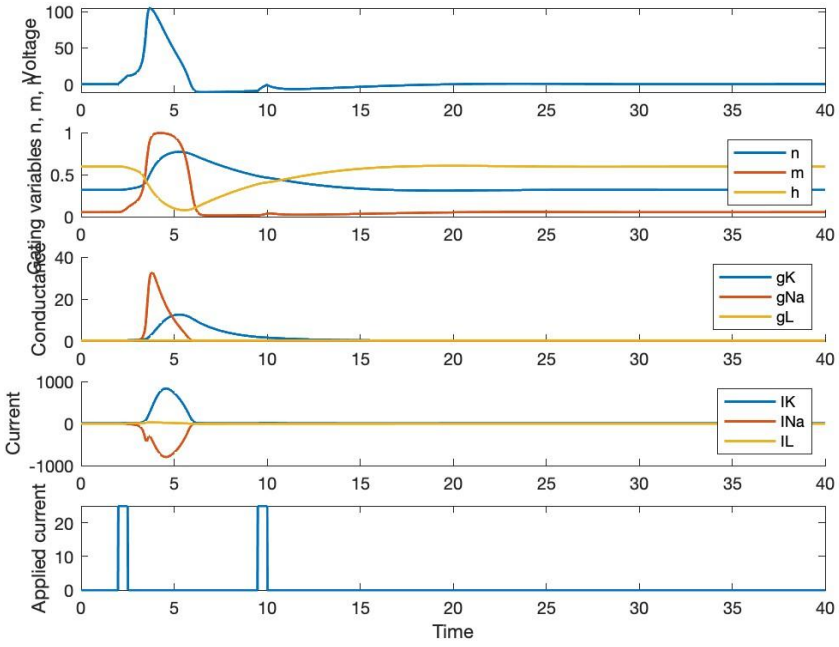


Action potentials are only generated beyond an input current of roughly 13 uA/cm^2 using the given parameters because the sodium channels that start an action potential are voltage-gated. This means that a large enough initial depolarization (from a significant input current) is required to cause an action potential (hence why they are "all or nothing."

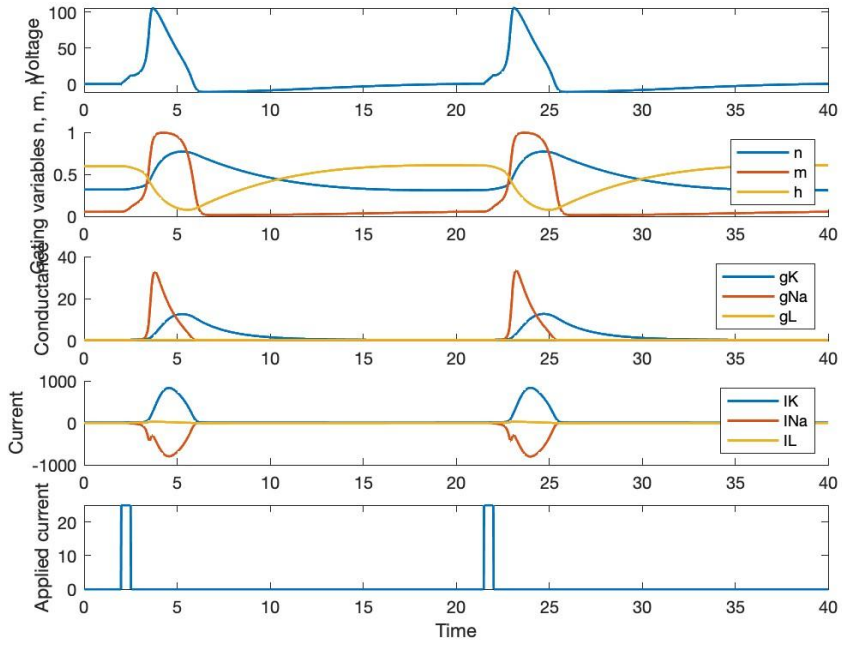
3. Determining refractory period in the Hodgkin-Huxley model

a. Minimum interval (refractory period) required to induce a second action potential in a neuron = ~13 ms

b. Figure



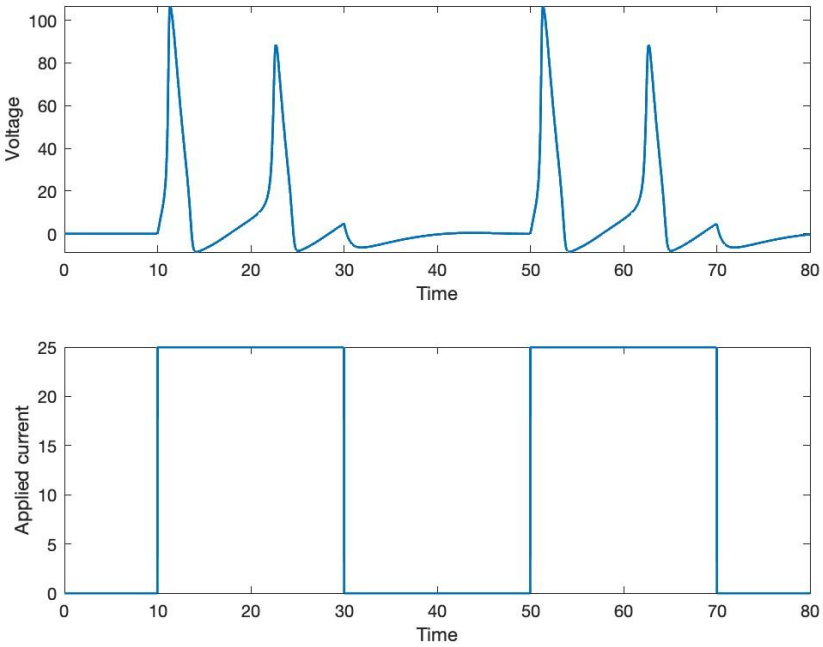
c. Figure



d. *The refractory period of about 13 ms in this simulation is roughly consistent with the time scale in humans (about 1-7 ms). For two stimuli that occur in a window shorter than this refractory period, only one action potential occurs (shown in part B, due to the first applied current) since there is not enough time for the neuron to repolarize/reset the electrochemical gradients across the membrane) before the second stimulus. For two stimuli that occur in a window longer than this refractory period, both action potentials occur (shown in part C).*

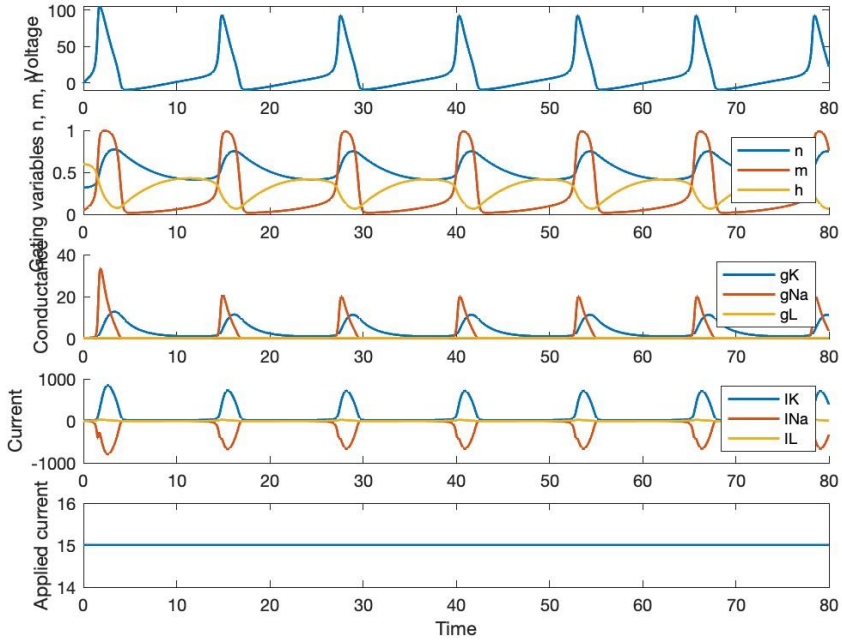
4. Controlling the beating of neurons

a. Figure



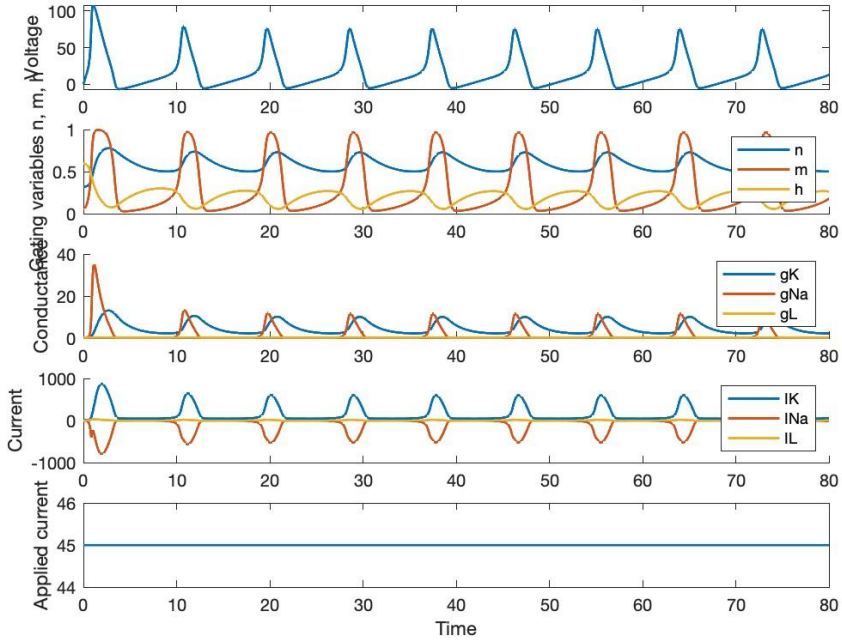
Subsequent action potentials (after initiating firing) occur before the voltage equilibrates from the prior action potential. This is possible because during the relative refractory period (i.e. after the absolute refractory period), an action potential can occur if there is a significantly large simulus (initial depolarization). As a result, the action potentials can occur with greater frequency. The subsequent action potentials have a slightly lower maximum voltage due to the residual ions left on either side of the membrane from the prior action potential.

b. Figure



With a constant current, the action potentials are consistent beyond the initial firing as described in part A for the duration of the applied current. The maximum depolarization voltage remains constant, as well as the gating variables, conductances, and currents.

c. Figure



Compared to part B, with a higher constant applied current, the action potentials occur with a higher frequency and with slightly lower maximum voltages, conductances, gating variables, and currents.

CODE

```
close all
clc

%% Problem 1: Simulating the Hodgkin-Huxley model (DEMO)
disp('PROBLEM 1')
clear

%% Problem 1, Part A
% on and off rates for n, m, h
params.alpha_n = @(V) 0.01 * (10-V) ./ (exp(1 - V./10)-1);
params.beta_n = @(V) 0.125 * exp(-V/80);
params.alpha_m = @(V) 0.1 * (25-V) ./ (exp(2.5 - V./10)-1);
params.beta_m = @(V) 4 * exp(-V./18);
params.alpha_h = @(V) 0.07 * exp(-V./20);
params.beta_h = @(V) 1 ./ (exp(3- V./10)+1);

% range of V
vrange = -20:120;

% plot results
figure(1)
subplot(311)
hold on
plot(vrange, params.alpha_n(vrange), LineWidth=1.5)
plot(vrange, params.beta_n(vrange), LineWidth=1.5)
legend("alpha", "beta", Location='best')
ylabel('On and off rates for n')

subplot(312)
hold on
plot(vrange, params.alpha_m(vrange), LineWidth=1.5)
plot(vrange, params.beta_m(vrange), LineWidth=1.5)
legend("alpha", "beta", Location='best')
ylabel('On and off rates for m')

subplot(313)
hold on
plot(vrange, params.alpha_h(vrange), LineWidth=1.5)
plot(vrange, params.beta_h(vrange), LineWidth=1.5)
legend("alpha", "beta", Location='best')
ylabel('On and off rates for h')

%% Problem 1, Part B
tspan = 0:0.02:20;
steps = [2, 2.5, 2; 10, 10.5, 25];

% define model parameters
params.gKbar = 36;
params.gNabar = 120;
params.gLbar = 0.3;
```

```
params.EK = -12;
params.ENa = 120;
params.EL = 10.6;
params.Cm = 1;

% define initial parameters
V0 = 0;
n0 = params.alpha_n(V0) / (params.alpha_n(V0) + params.beta_n(V0));
m0 = params.alpha_m(V0) / (params.alpha_m(V0) + params.beta_m(V0));
h0 = params.alpha_h(V0) / (params.alpha_h(V0) + params.beta_h(V0));

y0 = [V0; n0; m0; h0];

% simulate HH
[t, y] = ode45(@(t,y) hh_model(t, y, params, @(t) step_current(t, steps)), tspan,
y0);

Vm = y(:, 1);

n = y(:, 2);
m = y(:, 3);
h = y(:, 4);

gK = params.gKbar*n.^4;
gNa = params.gNabar*m.^3.*h;
gL = params.gLbar*ones(size(t));

IK = gK .* (Vm - params.EK);
INa = gNa .* (Vm - params.ENa);
IL = gL .* (Vm - params.EL);

applied_I = step_current(t, steps);

% plot results
figure(2)
subplot(511)
plot(t, Vm, LineWidth=1.5)
ylabel('Voltage')

subplot(512)
hold on
plot(t, n, LineWidth=1.5)
plot(t, m, LineWidth=1.5)
plot(t, h, LineWidth=1.5)
legend('n', 'm', 'h')
ylabel('Gating variables n, m, h')

subplot(513)
hold on
plot(t, gK, LineWidth=1.5)
plot(t, gNa, LineWidth=1.5)
```

```
plot(t, gL, LineWidth=1.5)
legend('gK', 'gNa', 'gL')
ylabel('Conductance')

subplot(514)
hold on
plot(t, IK, LineWidth=1.5)
plot(t, INa, LineWidth=1.5)
plot(t, IL, LineWidth=1.5)
legend('IK', 'INa', 'IL')
ylabel('Current')

subplot(515)
plot(t, applied_I, LineWidth=1.5)
ylabel('Applied current')
xlabel('Time')


%% Problem 2: Determining minimum stimulus to generate excitatory response
disp('PROBLEM 2')
% clear

%% Problem 2, Part A
tspan = 0:0.02:20;

I = 2:25;
I_min = 0;

for i=1:length(I)
    steps = [10, 10.5, I(i)];
    [t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current(t, steps)),
tspan, y0);

    Vm = y(:, 1);

    num_act_pot = count_act_pot(Vm, 40);
    if num_act_pot >= 1
        I_min = I(i);
        break
    end
end

I_set = [I_min-1, I_min, I_min+1];
Vm_set = zeros(length(Vm), 3);

for i=1:length(I_set)
    steps = [10, 10.5, I_set(i)];
    [t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current(t, steps)),
tspan, y0);

    Vm = y(:, 1);
```

```
    Vm_set(:, i) = Vm;
end

disp(I_set)

figure(3)
plot(t, Vm_set, 'LineWidth', 1.5)
title('PROBLEM 2: Minimum input current I_{min} for action potential')
xlabel('Time')
ylabel('Voltage')
legend(string(I_set), Location='best');


%% Problem 3: Determining refractory period in the Hodgkin-Huxley model
disp('PROBLEM 3')
% clear

%% Problem 3, Part A
tspan = 0:0.02:40;
d = 0:60;
d_min = 0;

params.ENa = 115;

for i=1:length(d)
    [t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current_with_duration(t,
d(i))), tspan, y0);

    Vm = y(:, 1);
    num_act_pot = count_act_pot(Vm, 40);
    if num_act_pot >= 2
        d_min = d(i);
        break
    end
end

disp(d_min)

%% Problem 3, Part B
[t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current_with_duration(t,
d_min-6)), tspan, y0);

Vm = y(:, 1);

n = y(:, 2);
m = y(:, 3);
h = y(:, 4);

gK = params.gKbar*n.^4;
gNa = params.gNabar*m.^3.*h;
```

```
gL = params.gLbar*ones(size(t));

IK = gK .* (Vm - params.EK);
INa = gNa .* (Vm - params.ENa);
IL = gL .* (Vm - params.EL);

applied_I = step_current_with_duration(t, d_min-6);

figure(4)
subplot(511)
plot(t, Vm, LineWidth=1.5)
ylabel('Voltage')

subplot(512)
hold on
plot(t, n, LineWidth=1.5)
plot(t, m, LineWidth=1.5)
plot(t, h, LineWidth=1.5)
legend('n', 'm', 'h')
ylabel('Gating variables n, m, h')

subplot(513)
hold on
plot(t, gK, LineWidth=1.5)
plot(t, gNa, LineWidth=1.5)
plot(t, gL, LineWidth=1.5)
legend('gK', 'gNa', 'gL')
ylabel('Conductance')

subplot(514)
hold on
plot(t, IK, LineWidth=1.5)
plot(t, INa, LineWidth=1.5)
plot(t, IL, LineWidth=1.5)
legend('IK', 'INa', 'IL')
ylabel('Current')

subplot(515)
plot(t, applied_I, LineWidth=1.5)
ylabel('Applied current')
xlabel('Time')

%% Problem 3, Part C
[t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current_with_duration(t,
d_min+6)), tspan, y0);

Vm = y(:, 1);

n = y(:, 2);
m = y(:, 3);
h = y(:, 4);
```

```
gK = params.gKbar*n.^4;
gNa = params.gNabar*m.^3.*h;
gL = params.gLbar*ones(size(t));

IK = gK .* (Vm - params.EK);
INa = gNa .* (Vm - params.ENa);
IL = gL .* (Vm - params.EL);

applied_I = step_current_with_duration(t, d_min+6);

figure(5)
subplot(511)
plot(t, Vm, LineWidth=1.5)
ylabel('Voltage')

subplot(512)
hold on
plot(t, n, LineWidth=1.5)
plot(t, m, LineWidth=1.5)
plot(t, h, LineWidth=1.5)
legend('n', 'm', 'h')
ylabel('Gating variables n, m, h')

subplot(513)
hold on
plot(t, gK, LineWidth=1.5)
plot(t, gNa, LineWidth=1.5)
plot(t, gL, LineWidth=1.5)
legend('gK', 'gNa', 'gL')
ylabel('Conductance')

subplot(514)
hold on
plot(t, IK, LineWidth=1.5)
plot(t, INa, LineWidth=1.5)
plot(t, IL, LineWidth=1.5)
legend('IK', 'INa', 'IL')
ylabel('Current')

subplot(515)
plot(t, applied_I, LineWidth=1.5)
ylabel('Applied current')
xlabel('Time')

%% Problem 3, Part D
% No MATLAB code required.


%% Problem 4: Controlling the beating of neurons
disp('PROBLEM 4')
```

```
% clear

%% Problem 4, Part A
tspan = 0:0.02:80;
steps = [10, 30, 25; 50, 70, 25];

% simulate HH
[t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current(t, steps)), tspan,
y0);

Vm = y(:, 1);

applied_I = step_current(t, steps);

figure(6)
subplot(211)
plot(t, Vm, 'LineWidth', 1.5)
ylabel('Voltage')
xlabel('Time')

subplot(212)
plot(t, applied_I, 'LineWidth', 1.5)
ylabel('Applied current')
xlabel('Time')

%% Problem 4, Part B
tspan = 0:0.02:80;
steps = [0, 80, 15];

[t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current(t, steps)), tspan,
y0);

Vm = y(:, 1);

n = y(:, 2);
m = y(:, 3);
h = y(:, 4);

gK = params.gKbar*n.^4;
gNa = params.gNabar*m.^3.*h;
gL = params.gLbar*ones(size(t));

IK = gK .* (Vm - params.EK);
INa = gNa .* (Vm - params.ENa);
IL = gL .* (Vm - params.EL);

applied_I = step_current(t, steps);

figure(7)
subplot(511)
plot(t, Vm, LineWidth=1.5)
```

```
ylabel('Voltage')

subplot(512)
hold on
plot(t, n, LineWidth=1.5)
plot(t, m, LineWidth=1.5)
plot(t, h, LineWidth=1.5)
legend('n', 'm', 'h')
ylabel('Gating variables n, m, h')

subplot(513)
hold on
plot(t, gK, LineWidth=1.5)
plot(t, gNa, LineWidth=1.5)
plot(t, gL, LineWidth=1.5)
legend('gK', 'gNa', 'gL')
ylabel('Conductance')

subplot(514)
hold on
plot(t, IK, LineWidth=1.5)
plot(t, INa, LineWidth=1.5)
plot(t, IL, LineWidth=1.5)
legend('IK', 'INa', 'IL')
ylabel('Current')

subplot(515)
plot(t, applied_I, LineWidth=1.5)
ylabel('Applied current')
xlabel('Time')

%% Problem 4, Part C
tspan = 0:0.02:80;
steps = [0, 80, 45];

[t, y] = ode45(@(t, y) hh_model(t, y, params, @(t) step_current(t, steps)), tspan,
y0);

Vm = y(:, 1);

n = y(:, 2);
m = y(:, 3);
h = y(:, 4);

gK = params.gKbar*n.^4;
gNa = params.gNabar*m.^3.*h;
gL = params.gLbar*ones(size(t));

IK = gK .* (Vm - params.EK);
INa = gNa .* (Vm - params.ENa);
IL = gL .* (Vm - params.EL);
```

```
applied_I = step_current(t, steps);

figure(8)
subplot(511)
plot(t, Vm, LineWidth=1.5)
ylabel('Voltage')

subplot(512)
hold on
plot(t, n, LineWidth=1.5)
plot(t, m, LineWidth=1.5)
plot(t, h, LineWidth=1.5)
legend('n', 'm', 'h')
ylabel('Gating variables n, m, h')

subplot(513)
hold on
plot(t, gK, LineWidth=1.5)
plot(t, gNa, LineWidth=1.5)
plot(t, gL, LineWidth=1.5)
legend('gK', 'gNa', 'gL')
ylabel('Conductance')

subplot(514)
hold on
plot(t, IK, LineWidth=1.5)
plot(t, INa, LineWidth=1.5)
plot(t, IL, LineWidth=1.5)
legend('IK', 'INa', 'IL')
ylabel('Current')

subplot(515)
plot(t, applied_I, LineWidth=1.5)
ylabel('Applied current')
xlabel('Time')


%% Functions

% Problem 1
function dydt = hh_model(t, y, p, stim_fn)
    % current conditions
    V = y(1);
    n = y(2);
    m = y(3);
    h = y(4);

    % applied stimulus
    I = stim_fn(t);
```

```
    % ODEs
    V_dot = (1/p.Cm) * (I - p.gKbar*n^4*(V-p.EK) - p.gNabar*m^3*h*(V-p.ENa) -
p.gLbar*(V-p.EL));
    n_dot = p.alpha_n(V)*(1-n) - p.beta_n(V)*n;
    m_dot = p.alpha_m(V)*(1-m) - p.beta_m(V)*m;
    h_dot = p.alpha_h(V)*(1-h) - p.beta_h(V)*h;

    dydt = [V_dot; n_dot; m_dot; h_dot];
end

function I = step_current(t, steps)
    % t: time
    % steps: matrix where columns are start_time, stop_time, and I, rows
    % are steps
    n_steps = length(steps(:, 1));
    I = zeros(size(t));

    for j = 1:n_steps
        step = steps(j,:);
        start_time = step(1);
        stop_time = step(2);
        stim = step(3);
        idx = (start_time <= t & t <= stop_time);
        I(idx) = stim;
    end
end

% Problem 2
function num_act_pot = count_act_pot(Vm, threshold)
    num_act_pot = 0;
    act_pot = false;

    for i = 1:length(Vm)
        if Vm(i) >= threshold && ~act_pot
            num_act_pot = num_act_pot + 1;
            act_pot = true;
        elseif Vm(i) < threshold
            act_pot = false;
        end
    end
end

% Problem 3
function I = step_current_with_duration(t, d)
    steps = [2, 2.5, 25; 2.5+d, 3+d, 25];

    n_steps = length(steps(:, 1));
    I = zeros(size(t));

    for j=1:n_steps
        step = steps(j, :);
```

```
        start_time = step(1);
        stop_time = step(2);
        stim = step(3);
        idx = (start_time <= t & t <= stop_time);
        I(idx) = stim;
    end
end

% Problem 4
```