EE 746 HW1-Spiking Neuron Models

Swadhin Dash (210020142) Shambhavi Shanker (21d070066) Parth Bansal (21d170028)

August 26, 2023

Contents

1 Leaky Integrate and Fire Model

(a) At steady state,$\frac{dV}{dt} = 0$

$$=> V(\infty) = \frac{I_o}{g_L} + E_L \tag{1}$$

Therefore, minimum Ic to initiate a spike $=> V(\infty) = V(t)$

$$I_c = g_L(V_t - E_L) = 2.7nA \tag{2}$$

(b) Code in the CODES section.

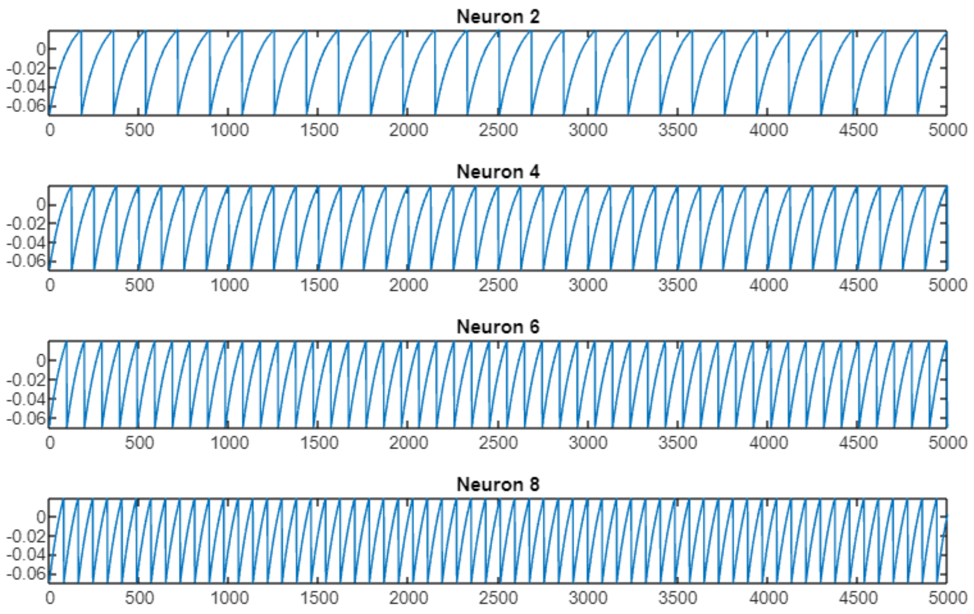

Figure 1: (c) Membrane Potential vs Time

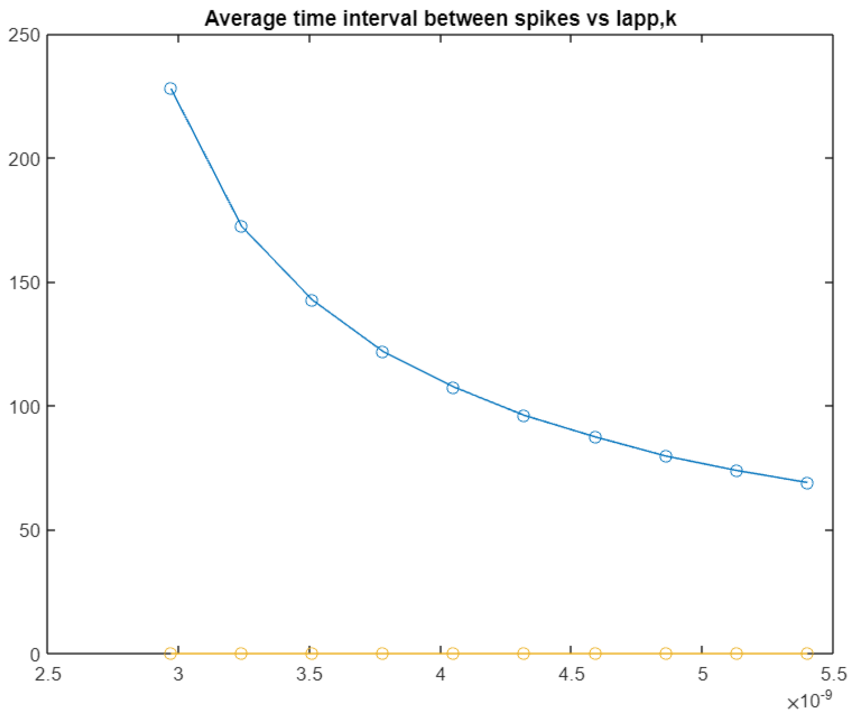

Figure 2: (d) Average time interval between spikes vs $I_{app,k}$

2 Izhikevich Model

(a) At steady state, $\frac{dV}{dt} = 0$, $\frac{dU}{dt} = 0$

$$=> V(\infty) = \frac{b}{k_Z} + E_t \tag{3}$$

$$=> U(\infty) = b(\frac{b}{k_Z} + E_t + E_r) \tag{4}$$

(b)
Equivalent difference Equations:

$$V(t+\delta T) = V(t) + \frac{1}{C}(k_z(V(t)-E_r)(V(t)-E_t) - U(t))\delta T \tag{5}$$

$$U(t+\delta T) = U(t) + a[b(V(t)-E_r) - U(t)]\delta T \tag{6}$$

(c)

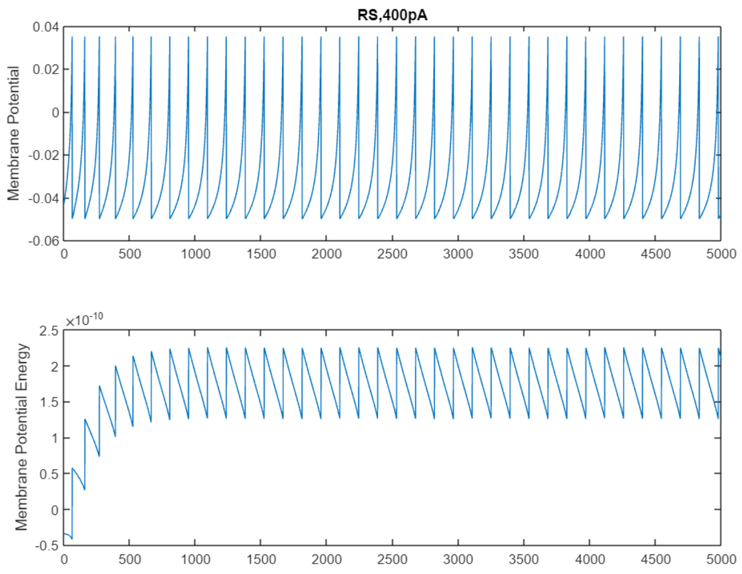

Figure 3: Membrane Potential and Energy(U) RS

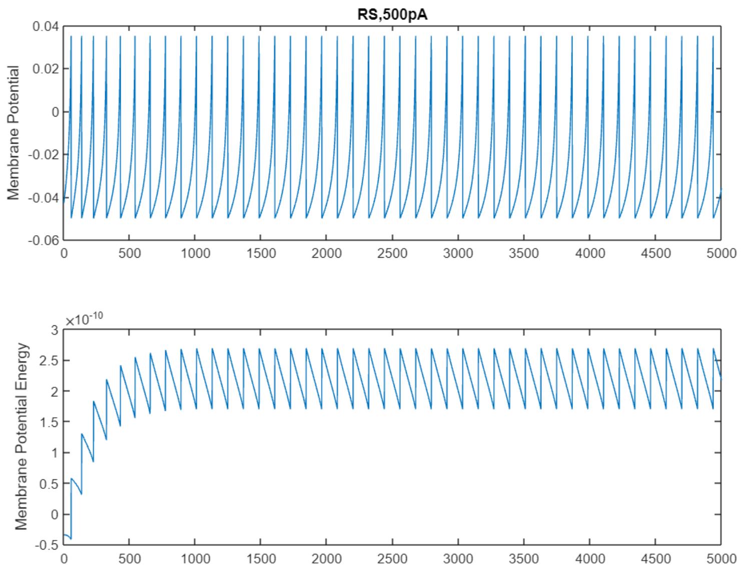

Figure 4: Membrane Potential and Energy(U) RS

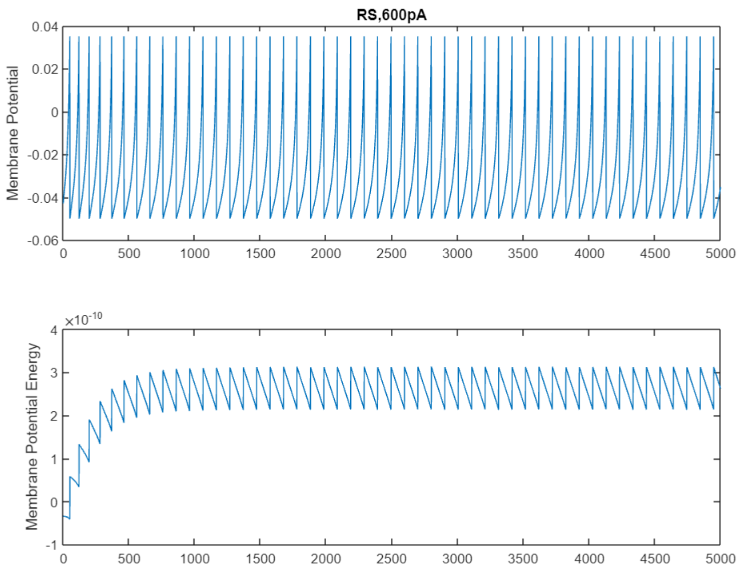

Figure 5: Membrane Potential and Energy(U) RS

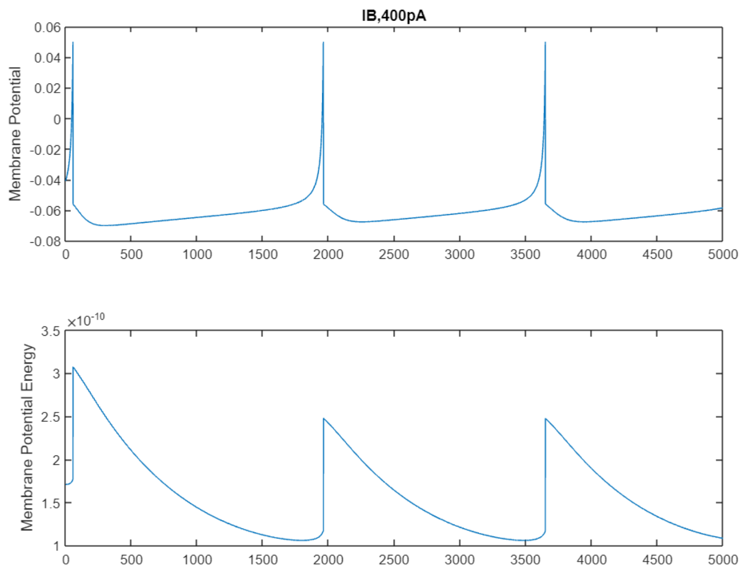

Figure 6: Membrane Potential and Energy(U) IB

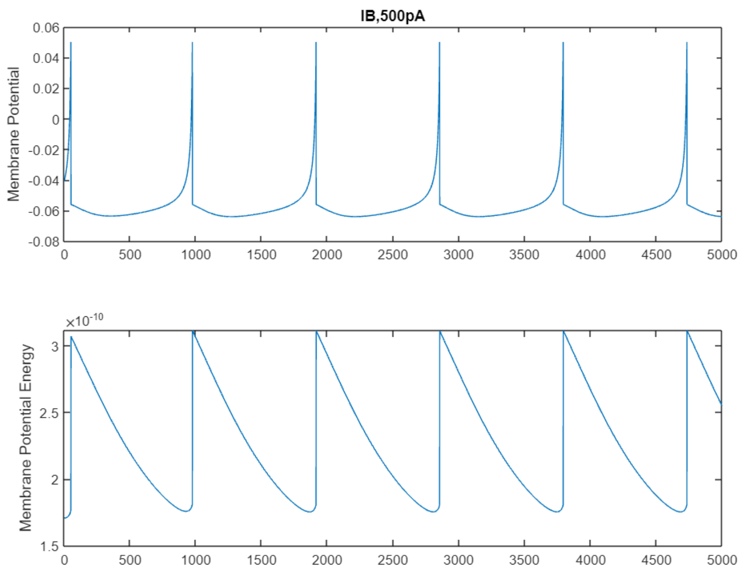

Figure 7: Membrane Potential and Energy(U) IB

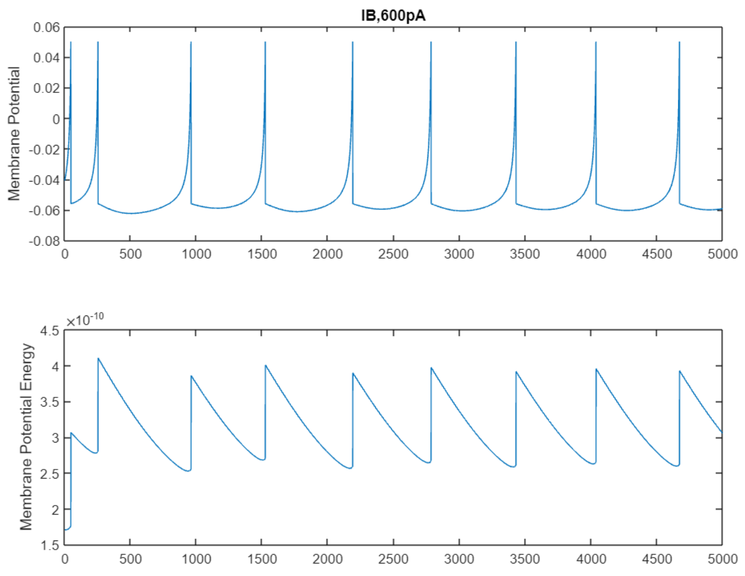

Figure 8: Membrane Potential and Energy(U) IB

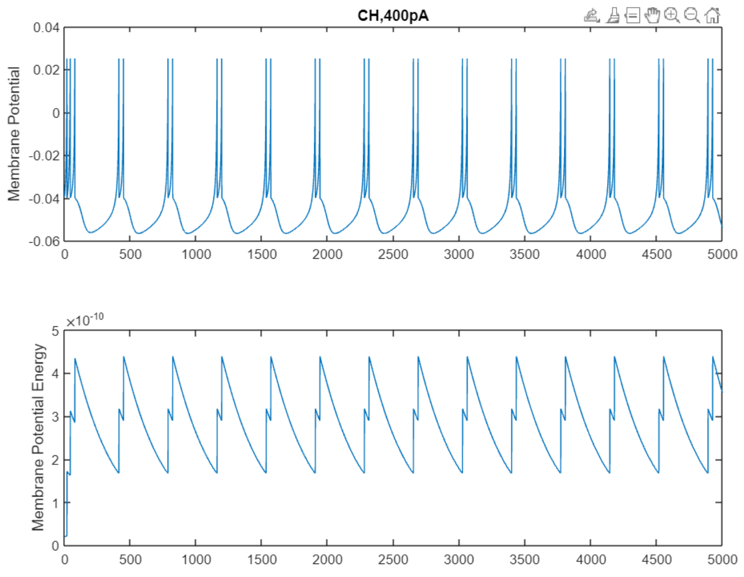

Figure 9: Membrane Potential and Energy(U) CH

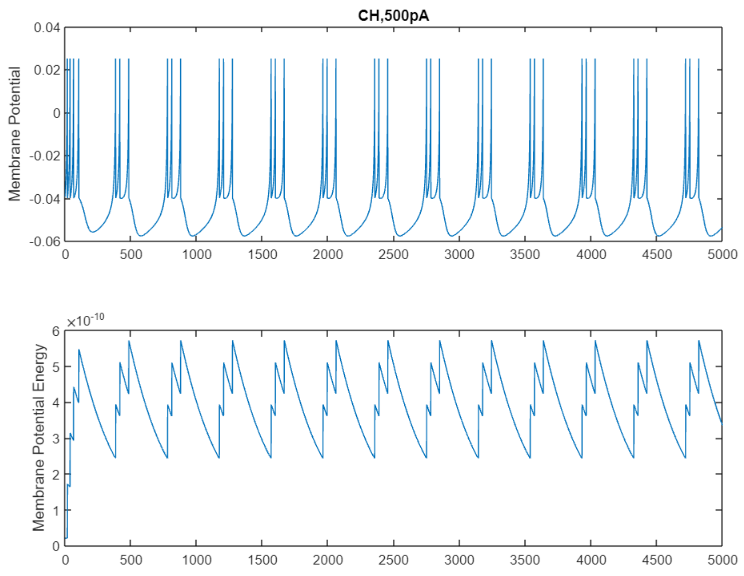

Figure 10: Membrane Potential and Energy(U) CH

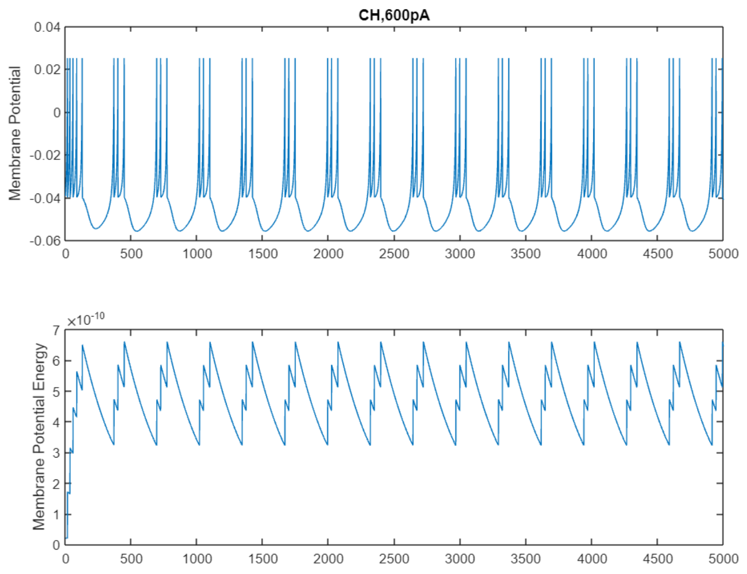

Figure 11: Membrane Potential and Energy(U) CH

3 Adaptive Exponential Integrate-and-Fire Model

(a) Equivalent Difference Eqs:

$$V_{n+1} = V_n + \frac{1}{C}[g_L(\triangle_T(\exp(\frac{V_n - V_T}{\triangle_T})) - (V_n - E_L)) - U_n + I_{app}]\triangle t$$

$$U_{n+1} = U_n + \frac{1}{\tau_\omega}[a(V_n - E_L) - U_n]]\triangle t$$

(b) At steady state, $\frac{dV}{dt} = 0$, $\frac{dU}{dt} = 0$

$$U_\infty = a[V(t) - E_L] \tag{7}$$

$$V_\infty = \frac{gl}{gl + a}(E_L + \triangle T(exp(\frac{V\infty - V_T}{\triangle_T})) \tag{8}$$

We solve the above equations of steady state values to get the resting potential.
(c)

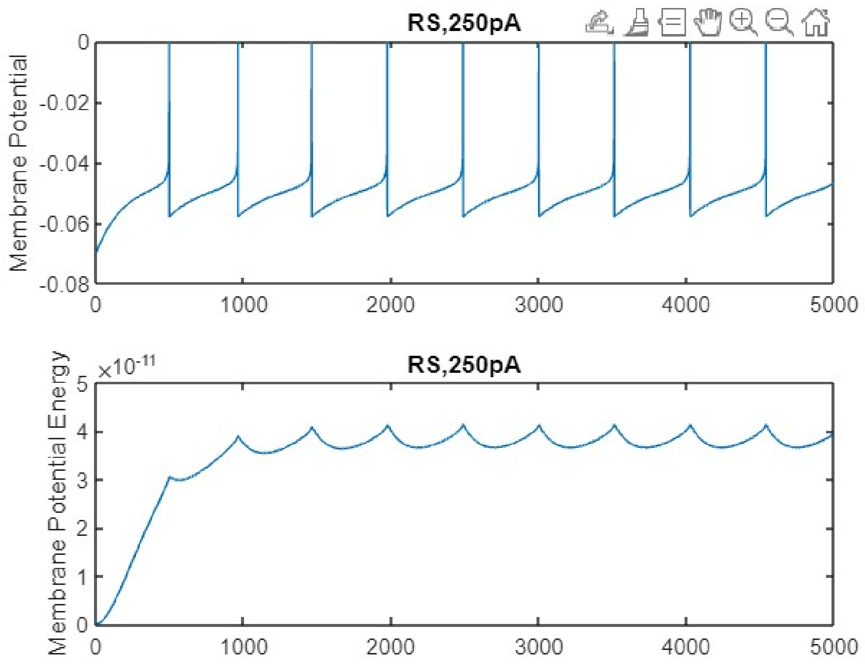

Figure 12: Membrane Potential and Energy(U) RS

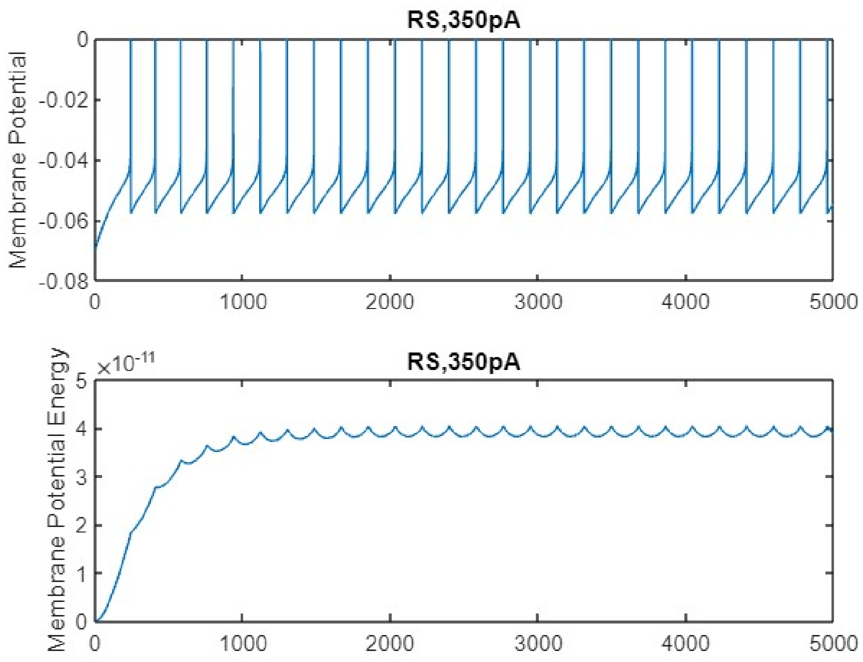

Figure 13: Membrane Potential and Energy(U) RS

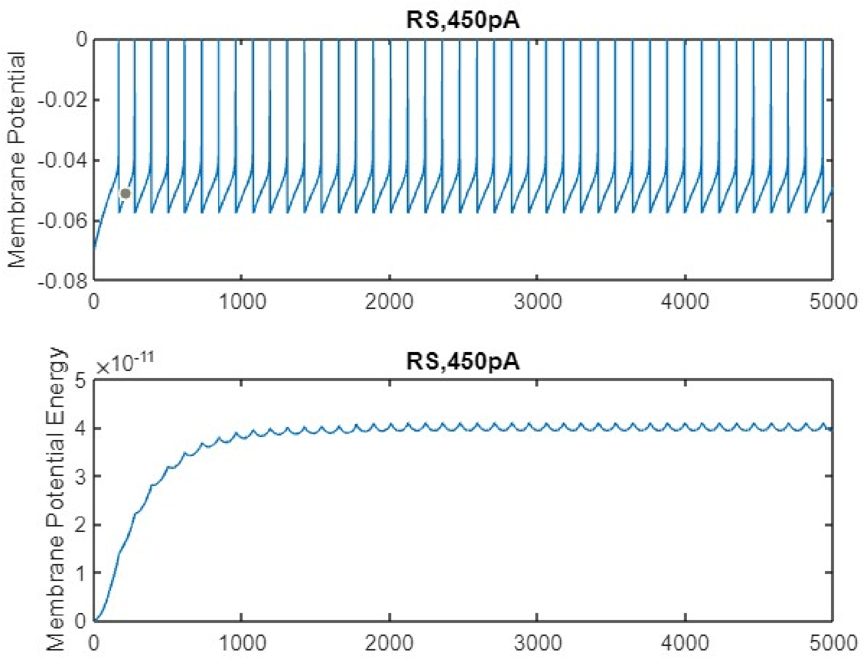

Figure 14: Membrane Potential and Energy(U) RS

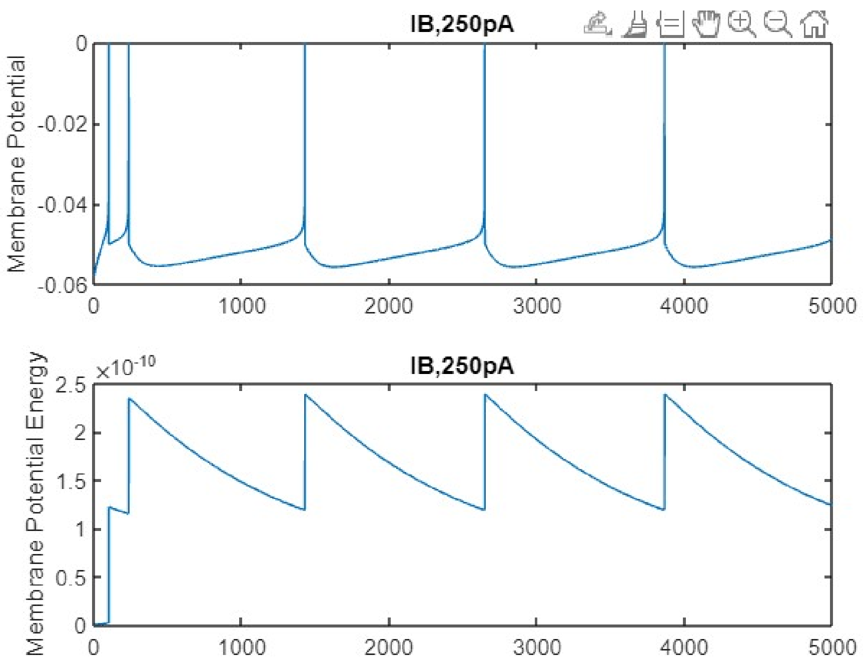

Figure 15: Membrane Potential and Energy(U) IB

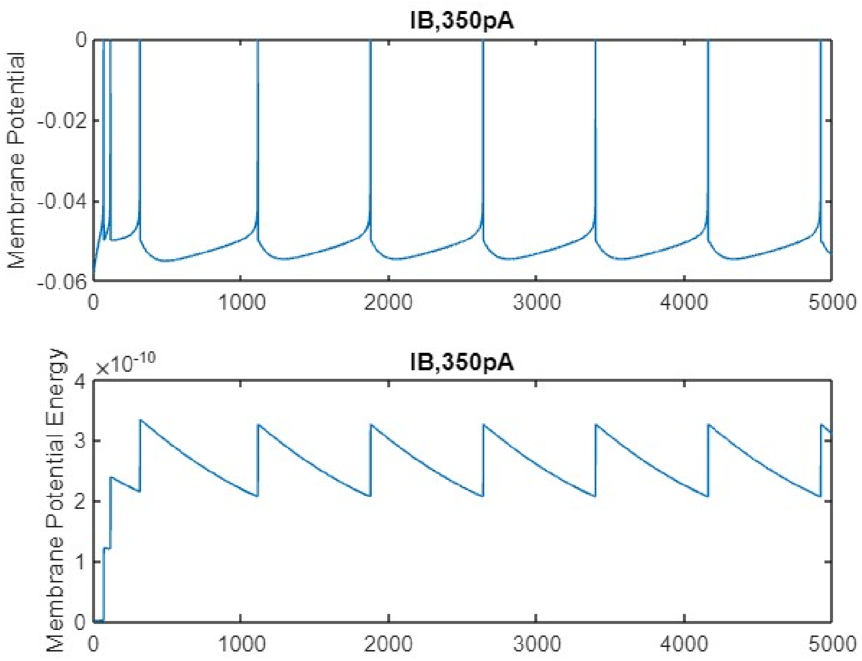

Figure 16: Membrane Potential and Energy(U) IB

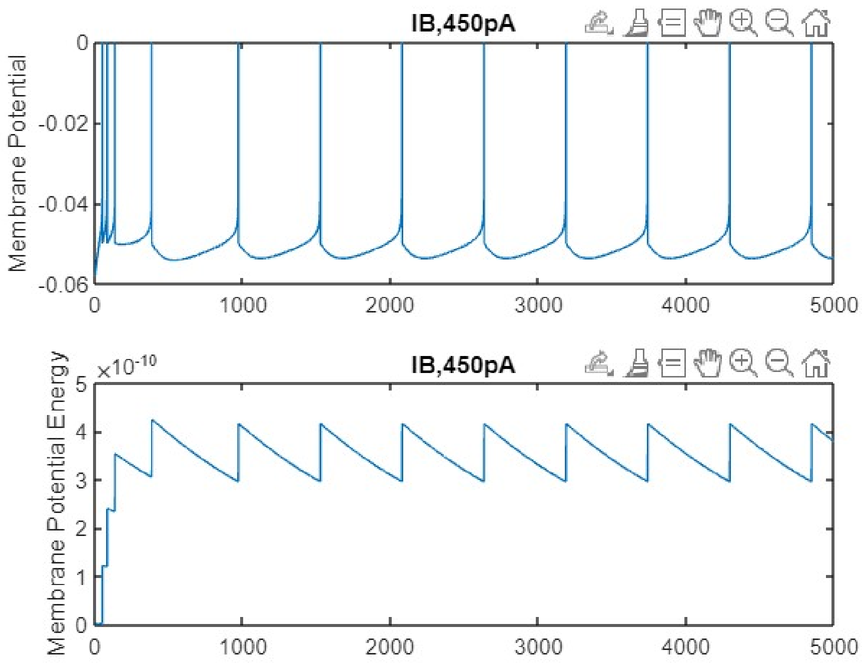

Figure 17: Membrane Potential and Energy(U) IB

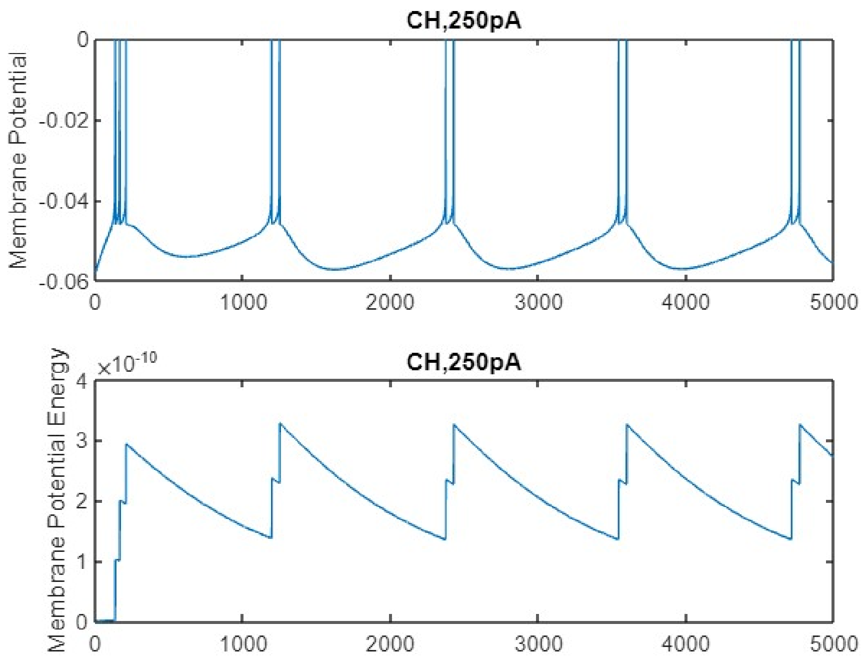

Figure 18: Membrane Potential and Energy(U) CH

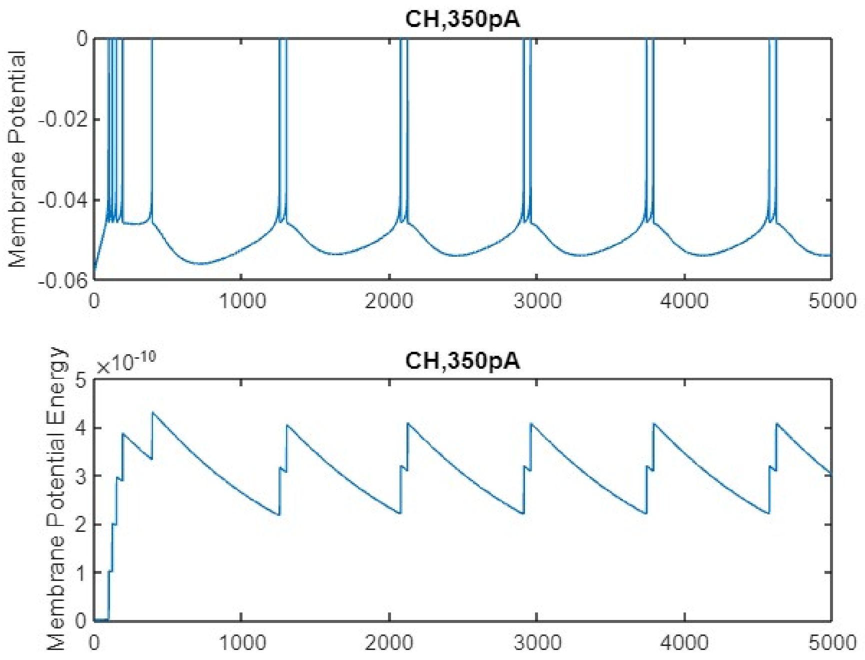

Figure 19: Membrane Potential and Energy(U) CH

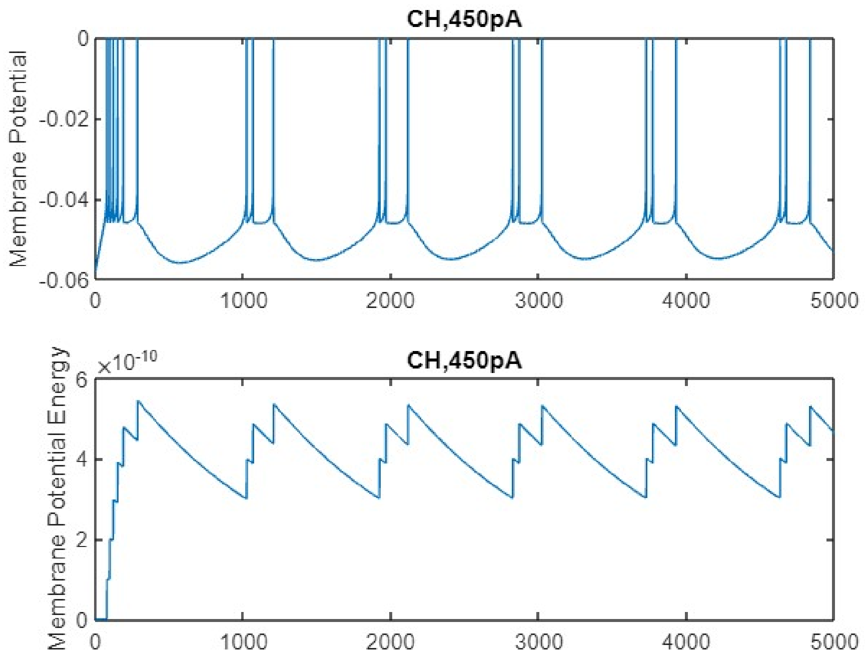

Figure 20: Membrane Potential and Energy(U) CH

4 Spike energy based on Hodgkin-Huxley neuron model

(a) Ion currents and the membrane potentials: We observe 3 action potentials over the 30 ms span.

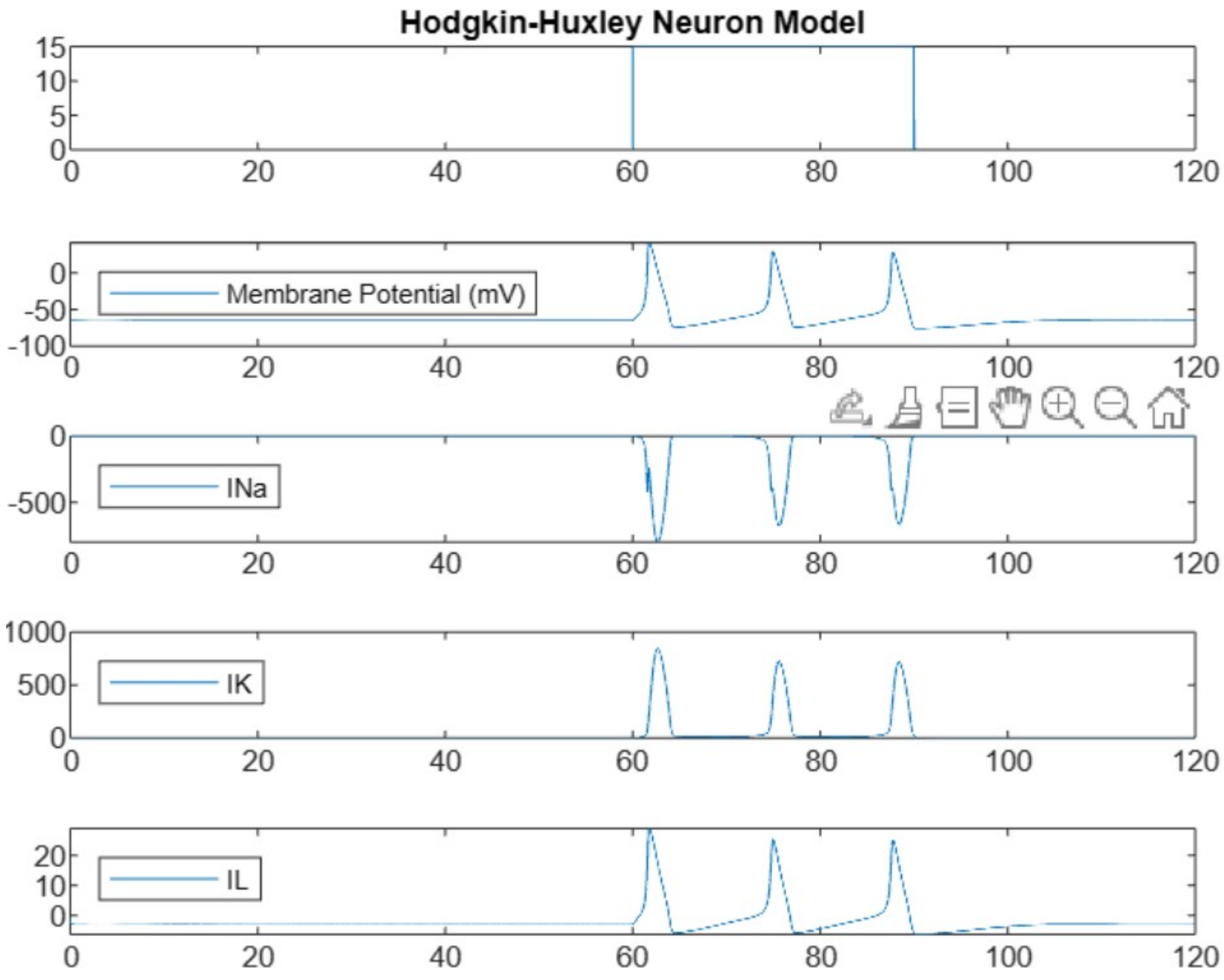

Figure 21: Membrane Potential CH

(b) Instantaneous Power dissipated (per unit area) in the ion channels and membrane capacitance: Figure 22

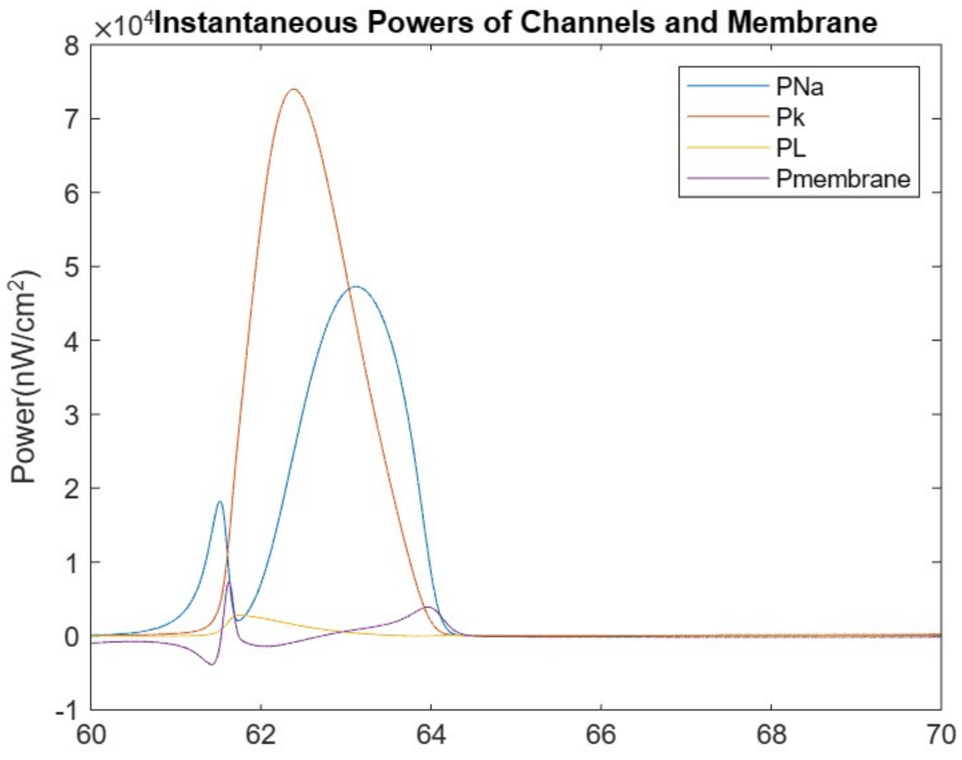

Figure 22: Energy(U) RS

(c) Total energy dissipated in one cycle of the action potential in membrane = Add powers of each point for 1 action potential.
The 3 action potentials are divided over the 30 ms time period = 1 action potential =10ms.
Total Energy(1 um^2) = $2.213 * 10^{-16}$ J

5 MATLAB Codes

5.1 Leaky Integrate and Fire Model

```
function [V_,I_,avg_]=LIF(n,m)

h=0.1e-3;
a=0.1;
I_c=2.7e-9;
V_T = 20e-3;
E_L = -70e-3;

V=ones(n,m)*E_L;
I=ones(n,m);
for i = 1:n
  I(i,:)=I_c*(1+i*a);
end

V(:,1)=RK2(V(:,1), h,I(:,1));
c = zeros(n);
avg = zeros(n);
prev = zeros(n);
for a = 2:m
   V(:,a)=RK2(V(:,a-1), 0.1e-3,I(:,a));
   for j=1:n
       if(V(j,a) > V_T)
           %V(j,a-1) = V(j,a);
           V(j,a) = E_L;
           if(c(j)>0)
               avg(j) = avg(j) + (a-prev(j)-avg(j))/(c(j)+1);
           end
           c(j) = c(j) + 1;
           prev(j) = a;
       end
   end

end
V_=V;
I_=I;
avg_=avg;
end

function k2 = RK2(V_vec, h, I_app)
k1 = V_vec + f_(V_vec, I_app) * (h/2);
k2 = k1 + f_(k1, I_app) * (h/2);
```

```
end

function deriv = f_(V, I_app)
C = 300e-12;
E_L = -70e-3;
G = 30e-9;
deriv = (1/C) * (I_app - G*(V - E_L));
end
```

5.2 Izhikevich Model

```
function [V_, U_] = Izhikevich(N,M,Neuron_Str,Iapp)
    RS = [100.0, 0.7, -60.0, -40.0, 0.03, -2.0, -50.0, 100.0, 35.0];
    IB = [150, 1.2, -75, -45, 0.01, 5, -56, 130, 50];
    CH = [50, 1.5, -60, -40, 0.03, 1, -40, 150, 25];
    Neuron =zeros(N,9);
    for i=1:N
    if(Neuron_Str(i)=="RS")
        Neuron(i,:) = RS;
    end
    if(Neuron_Str(i)=="IB")
        Neuron(i,:) = IB;
    end
    if(Neuron_Str(i)=="CH")
        Neuron(i,:) = CH;
    end
    end

    V_ss=(Neuron(:,4)+Neuron(:,6)./Neuron(:,2))*1e-3;
    U_ss=Neuron(:,6) .* 1e-12 .* (Neuron(:,4)+Neuron(:,6)./Neuron(:,2)-Neuron(:,3));
    V = ones(N,M);
    U = ones(N,M);


    V(:,1)=V_ss;
    U(:,1)=U_ss;
    dt = 1e-4;
    %Iapp = [400; 500; 600]*1e-12;

    %[V(:,1), U(:,1)] = RK4_VU(V(:,1), U(:,1), dt, Neuron, Iapp);
    for i = 2:M
```

```
        [V(:,i), U(:,i)] = RK4_VU(V(:,i-1), U(:,i-1), dt, Neuron, Iapp);
        for j = 1:N
           if(V(j,i) >= Neuron(j,9)*1e-3)
                V(j,i-1)= Neuron(j,9)*1e-3;
                V(j,i) = Neuron(j,7)*1e-3; % c
                U(j,i) = U(j,i) + Neuron(j,8)*1e-12; % d
           end
        end
    end
    V_=V;
    U_=U;
end

function [V_, U_] = RK4_VU(V, U, h, Neuron, Iapp)
    V1 = fV(V, U, Iapp, Neuron);
    V2 = fV(V+V1*(h/2), U, Iapp, Neuron);
    V3 = fV(V+V2*(h/2), U, Iapp, Neuron);
    V4 = fV(V+V3*h, U, Iapp, Neuron);
    V_ = V + h*((1/6)*V1 + (1/3)*V2 + (1/3)*V3 + (1/6)*V4);

    U1 = fU(V_, U, Neuron);
    U2 = fU(V_, U+U1*(h/2), Neuron);
    U3 = fU(V_, U+U2*(h/2), Neuron);
    U4 = fU(V_, U+U*h, Neuron);

    U_ = U + h*((1/6)*U1 + (1/3)*U2 + (1/3)*U3 + (1/6)*U4);
end

function der = fV(V, U, Iapp, Neuron)
C_ = (Neuron(:,1)*1e-12);
kz = Neuron(:,2)*1e-6;
Er = Neuron(:,3)*1e-3;
Et = Neuron(:,4)*1e-3;

%(V-Er).*(V-Et).*kz - U + Iapp

%x = kz .* (V-Er).*(V-Et);
der = 1./C_ .* ( kz.*((V-Er).*(V-Et)) - U + Iapp);
%disp("hi")
%der = V-Et
end


function der = fU(V, U, Neuron)
Er = Neuron(:,3)*1e-3;
```

```
a = Neuron(:,5)*1e3;
b = Neuron(:,6)*1e-9;
%c = Neuron(7)*1e-3;
%d = Neuron(8)*1e-12;
%v_peak = Neuron(9)*1e-3;

der = a.* ( b.*(V-Er) - U );
end
```

5.3 Adaptive Exponential Integrate-and-Fire Model

```
function[V,U] = AEF(N,M,Neuron_Str,Iapp)
RS = [200, 10, -70.0, -50.0, 2.0, 2.0, 30.0, 0.0, -58.0];
IB = [130, 18, -58, -50.0, 2, 4, 150, 120, -50];
CH = [200, 10, -58, -50, 2, 2, 120, 100, -46];

Neuron =zeros(N,9);
Vs = zeros(N,1);
Us = zeros(N,1);
for i=1:N
    if(Neuron_Str(i)=="RS")
        Neuron(i,:) = RS;
    end
    if(Neuron_Str(i)=="IB")
        Neuron(i,:) = IB;
    end
    if(Neuron_Str(i)=="CH")
        Neuron(i,:) = CH;
    end
    [Vs(i), Us(i)] = solve_ss(Neuron(i,:));
end

V = ones(N,M);
V(:, 1) = Vs;
U = ones(N,M);
U(:, 1) = Us;

dt = 1e-4;
%Iapp = [250; 350; 450]*1e-12;

for i = 2:M

    [V(:,i), U(:,i)] = RK1(V(:,i-1), U(:,i-1), dt, Neuron, Iapp);

    for j = 1:N
```

```
        if(V(j,i) >= 0)
            V(j,i-1)= 0;
            V(j,i) = Neuron(j,9)*1e-3; % c

            U(j,i) = U(j,i) + Neuron(j,8)*1e-12; % d

        end

    end



end
end
function [V_, U_] = RK1(V, U, h, Neuron, Iapp)
    V_ = V + fV(V,U,Iapp,Neuron)*h;
    U_ = U + fU(V,U,Neuron)*h;
end

function der = fV(V, U, Iapp, Neuron)
C = Neuron(:,1)*1e-12;
Gl = Neuron(:,2)*1e-9;
El = Neuron(:,3)*1e-3;
Vt = Neuron(:,4)*1e-3;
del= Neuron(:,5)*1e-3;
der = (-1*Gl.*(V-El)+Gl.*del.*exp((V-Vt)./del)-U+Iapp)./C;

end


function der = fU(V, U, Neuron)
El = Neuron(:,3)*1e-3;
a = Neuron(:,6)*1e-9;
tao= Neuron(:,7)*1e-3;

der = (a.*(V-El)-U)./tao;
end

function [Vs,Us] = solve_ss(Neuron)
syms x y
El = Neuron(3)*1e-3;
Vt = Neuron(4)*1e-3;
gl = Neuron(2)*1e-9;
a = Neuron(6)*1e-9;
DT = Neuron(5)*1e-3;
```

```
[Vss, Uss] = solve(x - El == y/a, gl*(x - El) + y == gl*DT*exp((x - Vt)/DT) );
Vs=double(Vss);
Us=double(Uss);
end
```

5.4 Spike energy based on Hodgkin-Huxley neuron model

```
% Constants
Cm = 1.0;       % Membrane capacitance (uF/cm^2)
gNa = 120;      % Sodium conductance (mS/cm^2)
ENa = 50;      % Sodium reversal potential (mV)
gK = 36;        % Potassium conductance (mS/cm^2)
EK = -77;       % Potassium reversal potential (mV)
gL = 0.3;       % Leak conductance (mS/cm^2)
EL = -55;      % Leak reversal potential (mV)
P = 0;
% Initial conditions
V0 = -65;       % Initial membrane potential (mV)
m0 = 0.05;      % Initial m gate value
h0 = 0.6;       % Initial h gate value
n0 = 0.32;      % Initial n gate value

% Time parameters
T = 30;         % Total simulation time (ms)
dt = 0.01;      % Time step (ms)
t = 0:dt:4*T;     % Time vector

% External current
I0 = 15;        % Amplitude of the current step (uA/cm^2)
t_on = 2*T;
t_off = 3*T;
Iext = I0 * (t >= t_on & t < t_off);

% Initialize vectors to store variables over time
V = zeros(size(t));
m = zeros(size(t));
h = zeros(size(t));
n = zeros(size(t));
INa = zeros(size(t));
IK = zeros(size(t));
IL = zeros(size(t));
PNa = zeros(size(t));
PK = zeros(size(t));
PL = zeros(size(t));
```

```
% Set initial values
V(1) = V0;
m(1) = m0;
h(1) = h0;
n(1) = n0;

% Simulate the Hodgkin-Huxley model
for i = 1:length(t) - 1
    % Compute gating variable derivatives
    alpha_m = (0.1 * (V(i) + 40)) / (1 - exp(-(V(i) + 40) / 10));
    beta_m = 4 * exp(-(V(i) + 65) / 18);
    alpha_h = 0.07 * exp(-(V(i) + 65) / 20);
    beta_h = 1 / (1 + exp(-(V(i) + 35) / 10));
    alpha_n = (0.01 * (V(i) + 55)) / (1 - exp(-(V(i) + 55) / 10));
    beta_n = 0.125 * exp(-(V(i) + 65) / 80);

    % Update gating variables
    m(i + 1) = m(i) + dt * (alpha_m * (1 - m(i)) - beta_m * m(i));
    h(i + 1) = h(i) + dt * (alpha_h * (1 - h(i)) - beta_h * h(i));
    n(i + 1) = n(i) + dt * (alpha_n * (1 - n(i)) - beta_n * n(i));

    % Compute membrane current
    INa(i+1) = gNa * m(i)^3 * h(i) * (V(i) - ENa);
    IK(i+1) = gK * n(i)^4 * (V(i) - EK);
    IL(i+1) = gL * (V(i) - EL);

    % Update membrane potential
    V(i + 1) = V(i) + dt * ((Iext(i) - INa(i) - IK(i) - IL(i)) / Cm);
end

%instantaneous power
PNa=INa.*(V-ENa);
PK=IK.*(V-EK);
PL=IL.*(V-EL);
P_membrane=((Iext - INa - IK - IL)).*V;

%c part- power for membrane
range=t>60 & t<70;
for i=6000:7000
    P=P+P_membrane(i);
end
-P*1e-15

% Plot the membrane potential with spikes
figure;
```

```
tiledlayout(5,1)
nexttile
plot(t, Iext)
title('Hodgkin-Huxley Neuron Model');
nexttile
plot(t, V);
legend('Membrane Potential (mV)','Location','northwest');
nexttile
plot(t, INa)
legend('INa','Location','northwest');
nexttile
plot(t, IK)
legend('IK','Location','northwest');
nexttile
plot(t, IL)
legend('IL','Location','northwest');

figure;
tiledlayout(1,1)
%nexttile
%plot(t(range), Iext(range))
%title('Hodgkin-Huxley Neuron Model');
nexttile
plot(t(range), PNa(range));
legend('PNa','PL','Location','northeast');
hold on
%nexttile
plot(t(range), PK(range))
legend('PK','Location','northeast');
%nexttile
plot(t(range), PL(range))
%nexttile
plot(t(range), P_membrane(range))
legend('PNa','Pk','PL','Pmembrane','Location','northeast');
title('Instantaneous Powers of Channels and Membrane');
ylabel('Power(nW/cm^2)');
hold off
%legend('Pmembrane','Location','northeast');
```