Model of a single neuron

The following matlab notebook has been written by Charles Burns in completion of the computational neuroscience course.

This document will follow the worksheet point by point for simplification. May be later rewritten as a short report.

The content is indirectly based on Dayan & Abbott (2001) Theoretical Neuroscience (Sections 5.1-5.4, 5.8-5.9).

The model of the membrane potential of a single neuron is derived from a simple system of equations:

$$V(0) = 0 \quad (1)$$
$$dV = (I - GV)dt \quad (2)$$

where $V(t)$ is the voltage at time t, $I(t)$ is an input current at time t, and G is the leak conductance (assumed constant), and dt is a small change in time. This system of equations can be solved by:

$$V(t+dt) = V(t) + dt(I - G \cdot V(t)) \quad (3)$$

1. A simple integrate and fire model

Below we include the code to implement an integrate and fire model, assuming no refractory period. We implement the following equation:

$$\tau_m \frac{dV}{dt} = E_L - V + R_m I_e \quad (1)$$

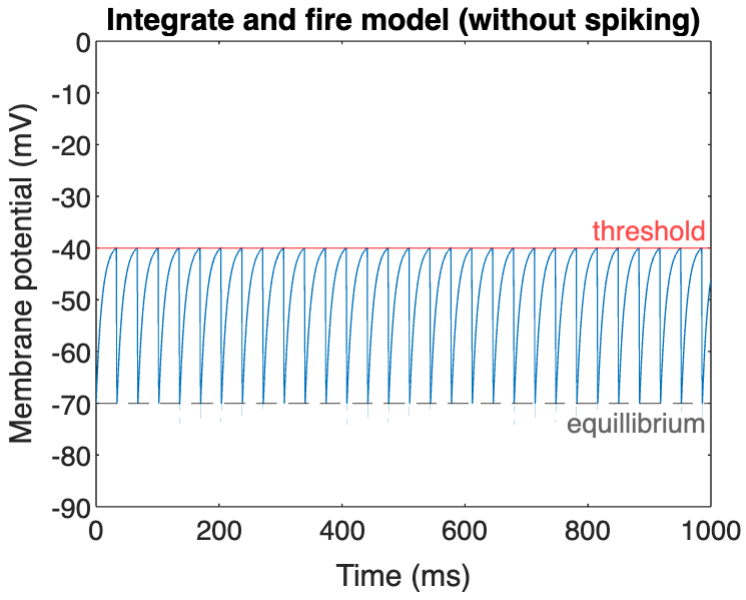
and its solution for the *subthreshold* membrane potential at time t, given a membrane resistance R_m, input current I_e and:

$$V(t) = E_L + R_m I_e + (V(0) - E_L - R_m I_e)\exp(-t/\tau_m) \quad (2)$$

therefore, when $V(t)$ reaches V, our code will reset to the equilibrium potential in the next timestep.

```
%Integrate and fire model of a single neuron.

%Parameters         Unit,     description
tau_m= 10*10^(-3);  %Seconds, membrane time constant
El = -70*10^(-3);   %Volt,    equillibrium potential due to leak channels
Vfire = -40*10^(-3);%Volt,    action potential threshold
Rm = 10*10^6;       %Ohm,     membrane resistance
I = 3.1*10^(-9);    %Ampere,  current for simulated input
dt = 10^(-3);       %Seconds, timestep for sampling voltage.
tMax = 1;           %Seconds, end of sampling period.

%We assume there is no refractory period and we do not plot spikes.
%we also assume that V(0) is the equillibrium potential.

times = 0:dt:tMax; %timepoints we sample at in seconds.
```

```
indices = 1:tMax/dt; %we index in integer steps.
V = zeros(length(indices),1); %initialise our voltage collected at each
timepoint
V(1) = El; %we include a constant to reset the *equation* of the membrane
potential after every spike.

for (i = indices) %translating timepoints to index
    %Integrate (update the membrane potential)

    V(i+1)=V(i)+dt*(El-V(i)+Rm*I)/tau_m; %implementing equation (3), with
di=1

    if V(i) > Vfire %if current timestep exceeds potential
      V(i+1) = El; %we reset V (going back to equillibrium).
    end

end

mV = V.*10^3; %rescaled into mV

plot(times*1000,mV); %adjusting the time interval, as the first entry is
V(t=0), and the last entry is techinically at V(t=999), t in ms.
xlabel('Time (ms)');
ylabel('Membrane potential (mV)');
ylim([-90,0]);
yline(Vfire*1000,'r-', 'threshold')
yline(-70,'--', 'equillibrium', 'LabelVerticalAlignment','bottom')
title('Integrate and fire model (without spiking)')
```



2. Analytical computation

Given the initial conditions in the code block above, what's the minimum input current for the cell to reach the threshold potential?

We can consider the convergence of the membrane potential in an attractor state, namely a local minima of the gradient of the membrane potential, V^*. Rearranging equation (1) above and setting $\frac{dV}{dt}=0$, we obtain the following:

$$\frac{dV}{dt}=\frac{1}{\tau_m}(E_L-V+R_mI_e)$$
$$\Rightarrow 0=\frac{1}{\tau_m}(E_L-V^*+R_mI_e)$$
$$\Rightarrow V^*=E_L+R_mI_e$$

Inputting the conditions from the codeblock in part one will result in $V^*=-39mV$, confirming that our neuron does reach the threshold in our simulations.

More generally, the neuron fires when $V^*>V_{th}$, which let's us rearrange the above for a condition on the input current, I_e:

$$I_e>\frac{V_{th}-E_L}{R_m}$$

Which, given our initial conditions will result in $I_e>3nA$.

```
minimumCurrent = (Vfire-El)/Rm
```

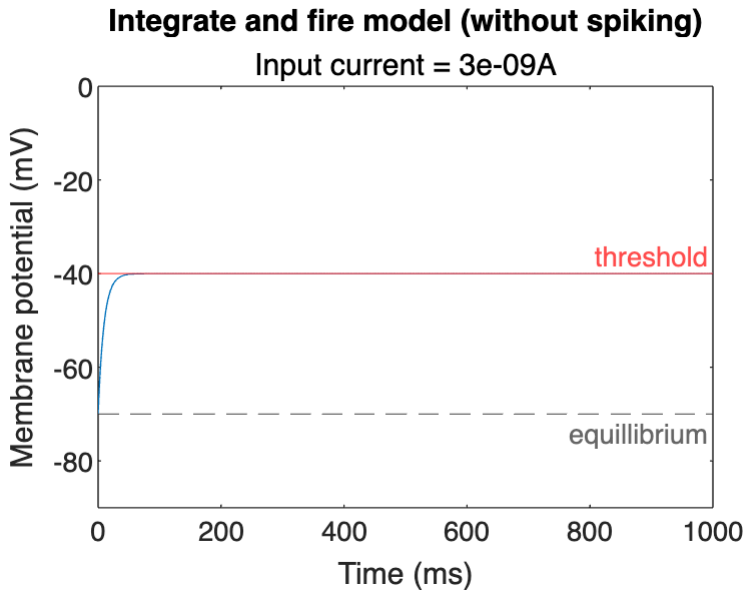
```
minimumCurrent = 3.0000e-09
```

3. Rerunning integrate and fire model

Here we re-run the integrate and fire model but *just below* the analytically derived minimum current for firing. From the figure below, we see that this neuron never fires.

```
%Parameters from step 1 apply here.

V2 = zeros(length(times),1); %initialising new data array
V2(1)=El; %resetting after above.

I2 = minimumCurrent; %0.1 nA lower than before.

for (i = indices) %see indices as defined in step 1.

    V2(i+1)=V2(i)+dt*(El-V2(i)+Rm*I2)/tau_m; %Integrate: update v(t)

    if V2(i) > Vfire %Fire:
      V2(i+1) = El; %we reset V (going back to equillibrium).
    end

end

mV2 = V2.*10^3; %rescaled into mV
```

```
plot(times*1000,mV2); %adjusting the time interval, as the first entry is
V(t=0), and the last entry is techinically at V(t=999), t in ms.
xlabel('Time (ms)');
ylabel('Membrane potential (mV)');
ylim([-90,0]);
yline(Vfire*1000,'r-', 'threshold')
yline(-70,'--', 'equillibrium','LabelVerticalAlignment','bottom')
title('Integrate and fire model (without spiking)')
subtitle('Input current = ' + string(minimumCurrent) +'A')
```

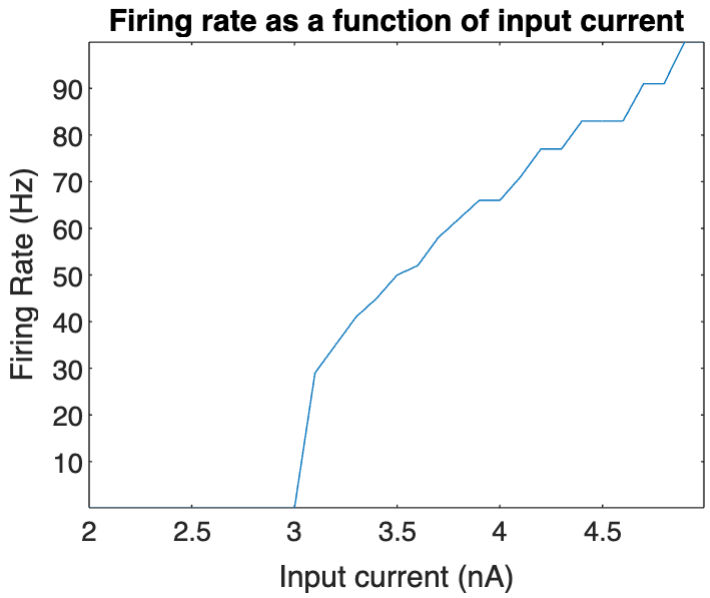


4. Firing rate as a function of input current

Following the worksheet instructions, we run another integrate and fire model, but this time we vary the input current and plot the firing rates.

```
%Parameters from step 1 apply here.

currentBin = [2:0.1:5]*10^(-9); %Amperes, the input current binned in 0.1
steps.
V3 = zeros(length(times),length(currentBin)); %initialise our voltage
collected at each timepoint
firingRate = zeros(length(currentBin),1); %initialise firing rate

for k = 1:length(currentBin)

    V3(1,k) = El; %reset the membrane potential for each current.
    Ik = currentBin(k);

 for (i = indices) %translating timepoints to index
    %Integrate (update the membrane potential)

    V3(i+1,k)=V3(i,k)+dt*(El-V3(i,k)+Rm*Ik)/tau_m; %implementing equation
(3), with di=1
```

```
    if V3(i,k) > Vfire %Fire:

      V3(i+1,k) = El; %we reset V (going back to equillibrium).

    end

end

    firingRate(k) = sum(V3(:,k)>Vfire)/tMax; %Firing rate in Hz.

end

plot(currentBin*10^9,firingRate) %scaling x-axis to nA
xlabel('Input current (nA)')
ylabel('Firing Rate (Hz)')
title('Firing rate as a function of input current')
```



5. Synaptic connections in neurons

This is our first step real step towards attractor dynamics in recurrent neural networks. We make basic assumptions about the neuronal parameters and see how their voltages will dynamically change.

a) Excitatory synapses

b) Inhibitory synapses

```
%Parameters - the same for both neurons
tau_m = 20*10^(-3);     %Seconds, membrane time constant
El = -70*10^(-3);       %Volt, equilibrium potential due to leak channels
Vreset = -80*10^(-3);   %Volt, the reset potential
Vth = -54*10^(-3);      %Volt, the threshold potential
RI = 18*10^(-3);        %Membrane resistance times input
RG = 0.15;              %Synapse resistance and conductance
```

```
Pmax = 0.5;              %probability, max synaptic opening probability
tau_s = 10*10^(-3);      %time constant for the synapse 'closing'.

Es = [0,-80*10^(-3)];    %Here we set the equilibrium of synapses in an array.

times = 0:dt:tMax;       %an array to index out the time passed
indices = 1:tMax/dt;     %an array for indexing

%initialising arrays
V1 = zeros(length(times),2); %membrane potential first neuron
V2 = zeros(length(times),2); %membrane potential second neuron

for c = 1:2 %for each case, either excitatory Es(1) or inhibitory Es(2)

%Reset before starting:
tf1 = 0; %assume we start by firing
tf2 = 0; %as above
V1(1,c) = Vreset + (Vth-Vreset)*rand(); %start between Vth and Vreset
V2(1,c) = Vreset + (Vth-Vreset)*rand(); %as above

    for i = indices

    Ps1 = Pmax*exp(-(times(i)-tf1)/tau_s);%probability synapse of 1st neuron
opens
    Ps2 = Pmax*exp(-(times(i)-tf2)/tau_s);%as above fore 2nd neuron synapse

        V1(i+1,c)=V1(i,c)+dt*(El-V1(i,c)-RG*Ps2*(V1(i,c)-Es(c))+RI)/tau_m;
%euler integration equation 3
        V2(i+1,c)=V2(i,c)+dt*(El-V2(i,c)-RG*Ps1*(V2(i,c)-Es(c))+RI)/tau_m;
%euler integration equation 3


        if V1(i,c) > Vth
            V1(i+1,c)=Vreset;
            tf1 = times(i);
        end

        if V2(i,c) > Vth
            V2(i+1,c)=Vreset;
            tf2 = times(i);
        end

    end

end

figure;
subplot(2,1,1);
hold on;
plot(times,V1(:,1)*1000,'r')
```

```
plot(times,V2(:,1)*1000,'b')
xlabel('Time (s)');
xlim([0,times(end)]);
ylabel('Membrane potential (mV)');
ylim([-100,-20]); %rescaling y-axis manually
yline(Vth*1000,'r-', 'threshold')
yline(Vreset*1000,'--', 'reset', 'LabelVerticalAlignment','bottom')
title('Excitatory Synapses')
hold off


subplot(2,1,2); hold on;
plot(times,V1(:,2)*1000,'r')
plot(times,V2(:,2)*1000,'b')
xlabel('Time (s)');
xlim([0,times(end)]);
ylabel('Membrane potential (mV)');
ylim([-100,-20]); %rescaling y-axis manually
yline(Vth*1000,'r-', 'threshold')
yline(Vreset*1000,'--', 'reset', 'LabelVerticalAlignment','bottom')
title('Inhibitory Synapses')
hold off;
```