Exercise: Leaky integrate and fire model of neural spike generation

This exercise investigates a simplified model of how neurons spike in response to current inputs, one of the most fundamental properties of neurons. The *leaky integrate and fire model* is based on the simple resistor-capacitor (RC) circuit that you will learn about here, with an extra rule: the model neuron spikes whenever the membrane potential surpasses a threshold. It is a simple and useful model, but ignores the detailed biophysical mechanisms behind spiking. In reality, neurons spike because of voltage-dependent ion channels. Besides these simplifications and apparent limitations the leaky integrate and fire is a useful model to understand how neurons respond to a current input.

Our intuition tells us that the rate at which spikes are produced is positively correlated with the strength of the input current delivered to the cell. We also understand that neurons cannot fire arbitrarily fast, so this correlation is limited by saturation and refractoriness.

In this exercise we will quantify the relation between input strength and firing rate. We will visualize this relation by plotting an *f-I* curve (also called an *activation function*). This curve describes the firing rate of a neuron (*f*, the number of spikes per second) as a function of injected input current *I*. Additionally, we will analyze the distribution of the *inter-spike intervals* (the timings between successive spikes) for different input currents.

Part 1: RC passive model of a cell: Building the model

In general we can model the membrane of a neuron as an RC circuit illustrated in Figure 1. There we depict the different electrical components, which are described in what follows.

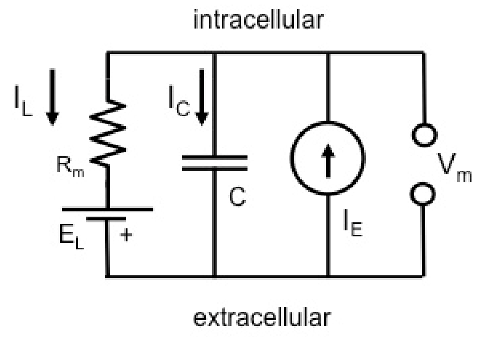

Figure 1: Equivalent electric (RC) circuit of a neurons' membrane

The cell membrane provides electrical insulation between the intracellular fluid and extracellular fluid that are two conductive media. This creates a capacitor, a device that accumulates charge across the membrane and therefore has the ability to *integrate* inputs over time. The input is represented by a current source (I_E). Remember that an integral is roughly a sum. In this particular case the capacitor has the ability to add up charge over time and therefore builds up a voltage difference across the membrane (V_m) that we refer as the *membrane potential*.

The capacitor has a capacitance proportional to the cell's surface area (A). A cell's total membrane capacitance (C_m) can be calculated as follows:

$$C_m = c_m A \tag{1.1}$$

where c_m is the *specific membrane capacitance*. Here, we assume c_m to be 10 nF/mm^2.

The current through the capacitor (I_C) is given by the following expression:

$$I_C = C_m \frac{dV_m}{dt} \tag{1.2}$$

The cell membrane is not a perfect insulator. Pores in the membrane I allow ions to move across it and therefore the charge that was built up across the capacitor has a way to *leak out*. This explains the name “leaky integrator.”

A cell’s total membrane conductance (G_m) represents how easily ions flow across the membrane. Recall that conductance and resistance are inversely related to each other ($R_m = 1/G_m$). Note that the total conductance is also proportional to the cell’s area, given by the following formula:

$$G_m = g_m A \tag{1.3}$$

where g_m is the *specific membrane conductance*. Here, we assume g_m to be 0.5 μS/mm^2.

You probably noticed that the resistor R_m is wired to a battery in Figure 1, which supplies a voltage E_L. This is to reflect the resting membrane potential of the neuron. In other words, neurons have a membrane potential difference across the membrane even when no electrical stimulation is applied. Therefore, E_L models this *resting potential*.

Next, Ohm’s law gives the leakage current (I_L) through the resistor R_m:

$$I_L = \frac{V_m - E_L}{R_m} \tag{1.4}$$

Finally, Kirchhoff’s current law states that sum of current flowing into a junction/node must be equal to the sum of currents flowing out of that node. By looking at Figure 1 we can see that:

$$I_C + I_L = I_E \tag{1.5}$$

Thus, applying equations 1.3 and 1.4 to 1.5 we obtain the following:

$$C_m \frac{dV_m}{dt} + \frac{V_m - E_L}{R_m} = I_E \tag{1.6}$$

which is a first-order, linear and autonomous ordinary differential equation.

Review questions:

Consider a spherical model neuron of radius 0.04 mm and the c_m and g_m values given in the preceding section. Write MATLAB® code to answer the following questions:

1. What is the total surface area of this cell? Recall that the surface area of a sphere is $A = 4\pi r^2$, where r is the sphere’s radius.
2. Calculate the total membrane capacitance (C_m).

3. Calculate the total membrane conductance G_m and the total membrane resistance R_m.

Part 2: Dynamics of the RC model

So far we have a differential equation (1.6) that relates changes in applied external current to changes in the membrane potential. This section focuses on how to numerically solve this equation to understand how the neurons' membrane potential changes over time. The idea is to find a function $V_m(t)$ that solves equation 1.6 for any arbitrary applied current.

The RC neuron response to an injected input current I_E is prescribed by the differential equation 1.6. After some rearrangement equation 1.6 can be rewritten as:

$$V_m(t) + \tau_m \frac{dV_m}{dt} = E_L + R_m I_E \qquad (2.1)$$

where $V_m(t)$ is the membrane potential at time t, R_m is the membrane resistance, C_m is the membrane capacitance, $I_E(t)$ is the current injected, and $\tau_m = R_m C_m$ is the membrane time constant. E_L is the resting membrane potential also known as the reversal potential of the leakage current (the current through resistor R_m). We call this the *resting potential* because when there is no input current and the voltage is not changing, the membrane voltage is equal to E_L. You can verify this by setting dV_m/dt and $I_E(t)$ to 0.

In the absence of input current this passive model of a cell relaxes exponentially to membrane potential E_L. Alternatively, when input current is supplied this modeled cell relaxes exponentially towards $E_L + R_m I_E(t)$.

So far we have a qualitative understanding on how this model responds to input currents. However, we would like to track the membrane potential as a function of time for any arbitrary input current. One way to do so is to approximate this differential equation by a difference equation, and use the computer to numerically integrate and find the solution for the membrane potential. To do so, we will rewrite dV_m as $V_m(t+\Delta t) - V(t)$ and replace dt by Δt, which after some rearranging leads to the following difference equation:

$$V_m(t+\Delta t) = V_m(t) + \frac{\Delta t\left(E_L - V_m(t) + R_m I_E(t)\right)}{\tau_m} \qquad (2.2)$$

where Δt is the integration time-step, i.e. the time elapsed from t to $t+\Delta t$. If the time-step is sufficiently small, the difference equation 2.2 provides a faithful representation of the differential equation 2.1.

If we know the parameters and the membrane potential at time 0 (known as the initial condition V_0 $V(t=0)$) we can find the membrane potential trajectory for any given input current.

As described this model is unable to generate spikes. In the following section we build on this model to generate a simplified version of a spiking cell.

Part 3: The leaky integrate and fire model

The RC model alone cannot produce action potentials as we have discussed in Parts 1 and 2. Nonetheless we can mimic spiking behavior by setting an additional rule: that the modeled neuron emits a spike whenever the membrane potential crosses a threshold value V_{th}. After each spike, we will reset the membrane voltage to a value V_{reset} below threshold, and the model will continue to integrate the input from there. In this model there is no mechanistic/biophysical description of the action potentials. We just introduced a fire and reset rule after threshold crossing. This constitutes the leaky integrate and fire model.

The membrane potential of this model neuron is described by the same differential equation as the RC passive model of a cell described in Part 1 and therefore can be approximated by equation 2.2. In the following we will study this model via numerical simulations. During simulations it is convenient to keep track of the membrane potential and of spiking behavior. The latter can be tracked by introducing a vector that is zero whenever the membrane potential is below threshold and gets a value 1 when the membrane potential crosses threshold. This representation is called a *binary spike train*.

For our simulations we will consider a leaky integrate and fire model with the following parameters:

$$R_m = 100\,\mathrm{M\Omega} \quad C_m = 200\,\mathrm{pF} \quad E_L = -70\,\mathrm{mV}$$
$$V_{th} = -60\,\mathrm{mV} \qquad V_{reset} = E_L$$

We will set the initial condition and integration time-step for equation 1.4 as follows:

$$V_0 = E_L \qquad \Delta t = 0.01\,\mathrm{msec}$$

4. Implement the leaky integrate and fire model as a MATLAB function named `myLIF`. This function takes as input: (1) the parameters of the model, (2) the time-step and initial condition[1], and (3) a scalar that sets the amplitude of the injected current. This function produces two outputs: (1) the membrane potential obtained by iteratively solving equation 1.4 with the explained fire and reset rules, and (2) a binary spike train.

5. Let's try the function you just wrote. Examine the output of the model neuron to a constant current of 150 pA that is applied for 0.5 seconds. Plot the membrane potential as a function of time. What is the firing rate of this neuron (i.e. the number of spikes per second)?

6. Compute the *inter-spike intervals* (the time between two subsequent spikes). The MATLAB functions `find` and `diff` may be helpful. Is this an irregular or regular firing neuron?

7. How would you estimate the mean firing rate from the ISIs?

A key characteristic of neurons is their *refractory period*. This refers to a brief rest period after an action potential is fired, when the neuron is unable to fire again. We can add a refractory period to the leaky integrate and fire model by holding the membrane potential at V_{reset} for the duration of the refractory period after each spike.

[1] The parameters, initial condition and time-step can be incorporated into a *structure* to make the function call shorter.

8. Make a copy of your `myLIF` function and call it `myLIFref`. Revise this new function to include a refractory period (t_{ref}) of 3 msec.
9. Examine the output of the model neuron to a constant current of 150 pA that is applied for 0.5 seconds. Plot the membrane potential as a function of time and check that your refractory period is properly working (zoom into your plot to verify this).
10. What is the firing rate of this neuron? How does it compare to the model without a refractory?

For the reminder of this exercise we will continue to use the LIF model with a refractory period of 3 msec.

Next, we will compute the *f-I* curve for this neuron. Remember that this curve describes the firing rate of a neuron (*f*, the number of spikes per second) as a function of injected input current *I*. To do so we need to examine the behavior of our model cell with currents of different amplitudes. We will test currents in the interval 0 to 500 pA in increments of 10 pA. Each current step will be applied for 1 second.

11. Use your `myLIFref` function to compute the *f-I* curve of this neuron. Plot the firing rate (number of spikes per second) as a function of current amplitude. What is the minimum current that will elicit a spike?

In the case of the leaky integrate and fire model the *f-I* curve has a closed analytical expression:

$$f(I_E)=\begin{cases}0 & \text{if } I_E \le \dfrac{V_{th}-E_L}{R_m}\\ \left[t_{ref}+\tau_m \ln\left(\dfrac{R_m I_E+E_L-V_{reset}}{R_m I_E+E_L-V_{th}}\right)\right]^{-1} & \text{if } I_E > \dfrac{V_{th}-E_L}{R_m}\end{cases} \qquad (2.3)$$

12. Plot the simulated and theoretical curves on the same figure. (Use the MATLAB commands `hold on` and `hold off` to do so.) Use a line for the theoretical curve and squares for the results of your simulation. To plot square markers, add ‘s’ to your plotting command, e.g. `plot(X,Y,'s')`.
13. Do the simulations match the analytical expression?
14. Repeat your simulations with input currents in the range 0 to 10000 pA in increments of 100 pA. Redo the plot for the simulated and theoretical curves. What is the theoretical maximum firing rate this neuron can achieve? How is it related to the refractory period of your neuron?
15. Include this maximum theoretical value as a black dashed line asymptote in your previous plot. To do so you can type in MATLAB `plot(X,Y,'--k')`.

Part 4: Introducing variability in the spike times

So far our model is able to produce regular spiking, as all the ISI sequence values are identical. A simple way to make our model more realistic is to inject a noisy current so that the times of threshold crossing are not so easily predictable. This will introduce variability in the ISIs that can be captured in an ISI distribution histogram.

A simple way to introduce a noisy current is by randomly selecting at each time-step a current value from a Gaussian distribution with mean I_e and standard deviation σ_{noise}. The MATLAB function `randn` will be helpful for generating noisy currents.

Let's visualize the effect of this noisy current by running a simulation example

16. Make a copy of your `myLIFref` function and name it `myLIFnoise`. Modify the new copy to use random stimulation. This function takes an extra input argument that sets the standard deviation of the noisy current. Use the `myLIFnoise` function to stimulate your cell for 10 seconds with a noisy current that has a Gaussian distribution with mean I_E = 200pA and standard deviation σ_{noise} 200pA. Plot a histogram (commands `hist`, `histc` and `bar` will be helpful) of the ISI distribution. Compute the mean and standard deviation of the ISI distribution.

Next, let's explore the effect of σ_{noise} more systematically. To do so we will run a set of simulations of 10 seconds each where the mean of the noise is fixed but the standard deviation is increased. We will use Gaussian noise with mean I_e = 200pA and will test values of standard deviation in the range 0 to 400pA in increments of 50 pA.

17. Using `subplot`, make a nine-panel figure window. From left to right plot the ISI distribution for increasing values of σ_{noise}. To facilitate visual comparison of the plots make sure that the scales on the x-axes are the same across panels (the command `xlim` will be helpful) and that you use the same binning for all your histograms.
18. Let us quantify the effect of noise on firing rate using simple statistics. Make a two-panel figure window. In the left panel plot the mean ISI as a function of the magnitude of σ_{noise}. On the right panel plot the standard deviation of the ISI as a function of σ_{noise}. Interpret these plots.
19. Can you predict how the noise in the input current will affect the shape of the *f-I* curve?
20. To test your prediction compute and plot the *f-I* curve as described in point 11 but set σ_{noise} to 400 pA. What has changed with respect to the noiseless case? Why do you think people say this curve has a "soft threshold"?