Neural Computation (VS 265), Problem Set 2 - Neuron models

Due date: October 1, 3:30pm

Fall 2024

General guidelines:

- We are grading problem sets anonymously. **Include your student ID in the submission, but do not include your name.**
- You may work in small groups of 2-3. Note that you are responsible for writing up and submitting your submission individually.
- You are expected to attach any code you used for this assignment but will be evaluated primarily on the writeup.

Part 1: The Membrane Equation

i. *Membrane nonlinearity.* First let's examine how the membrane voltage at equilibrium behaves in response to specific ion channels opening. To understand the membrane's nonlinear behavior, we will first look at how the membrane voltage changes in response to opening specific ion channels, of one type at a time, and then in combination. Assume $V_{\text{rest}} = -70\,\text{mV}$ and $G_{\text{leak}} = 5\,\text{nS}$.

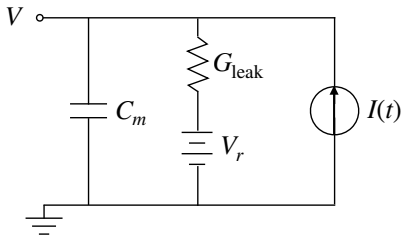
(a) Examine the effect of a single synaptic input that opens a set of sodium channels (ΔG_{Na}). Sweep ΔG_{Na} from 0 nS to 25 nS and plot the resulting equilibrium membrane potential (by solving for V at $\frac{dV}{dt} = 0$) over this range. You should notice a regime where the membrane voltage can be reasonably approximated as a linear function of ΔG_{Na} – what is that regime and why?

(b) Now do the same for an inhibitory synaptic input that opens a set of potassium channels, by varying ΔG_{K} over the same range and superimposing on the plot above.

(c) Next, examine how the membrane responds *jointly* to both synaptic inputs. Show this as a contour plot and choose a range of values for ΔG_{Na} and ΔG_{K} that allows you to see the linear vs. nonlinear regimes for combined input. Explain why this is happening with reference to your plots above.

(d) Finally, in a third plot, show the effect of shunting inhibition by simulating an inhibitory synaptic input that causes chloride channels to open by some amount (say $\Delta G_{\text{Cl}} = 10\,\text{nS}$) and now sweep ΔG_{Na} over the same range as above. (You may assume $V_{\text{Cl}} = V_{\text{rest}}$.) How does this compare to what you would expect from a linear superposition? (plot as a dashed line). Explain your results.

ii. *Membrane dynamics.* Now let's examine the membrane dynamics by simulating the membrane voltage in response to a time-varying input current, $I(t)$. Here we can aggregate the background level of open sodium, potassium and chloride channels into a single conductance, G_{leak}, leading to the following equivalent circuit:



As discussed in class, the resulting time-varying membrane voltage, V, is given by the leaky-integrator equation:

$$\tau \dot{V} + V(t) = V_r + \frac{1}{G_{\text{leak}}} I(t) \tag{1}$$

which may be simulated in discrete-time via

$$\texttt{V[n+1] = } \alpha * \texttt{(V\_r + I[n]/G\_leak) + (1-}\alpha\texttt{) * V[n]}$$

where `n` denotes the (integer) time-step of the simulation and $\alpha = \Delta t/\tau$. `n` is related to t in equation (1) via $t = \texttt{n}\Delta t$. (There are certainly better methods that yield more accurate results, which you are free to use, but this will suffice for our purposes. See the handout on Simulating Differential Equations for further details).

Now let's run this equation within a for-loop to compute the membrane voltage for a duration of 500 milliseconds using the following parameters:

- $C_m = 100\,\text{pF}$
- $G_{\text{leak}} = 5\,\text{nS}$.
- Initial condition: $V(0) = V_r = -70\,\text{mV}$
- $I(t) = \begin{cases} 0\,\text{pA} & 0 \le t < 100\,\text{ms} \\ 100\,\text{pA} & t \ge 100\,\text{ms} \end{cases}$

Try different values of G_{leak} and C_m to explore how these parameters affect the rise time and resulting membrane voltage. Plot the results of your simulation and interpret your findings.

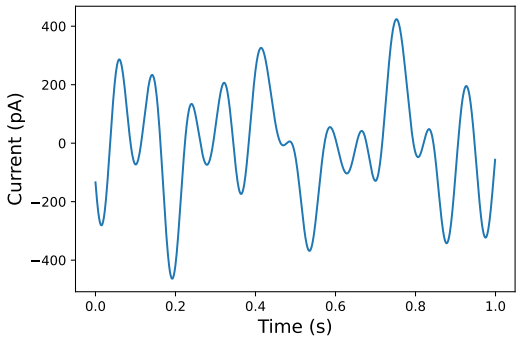
Part 2: The Leaky-Integrate-and-Fire (LIF) model

Now let's build upon the model in Part 1 by adding a spiking mechanism. A simple but useful way of doing this is via the LIF model. Within the for-loop for the leaky-integrator model above, we now check at each time-step `n` to see if the membrane voltage, V, exceeds the threshold voltage, V_{thresh}. If so, a spike (+55mv, but its exact value is arbitrary) is inserted at that time-step. The membrane voltage is then reset to V_r at the next time-step and held at that value for a duration given by the refractory period, t_{ref}, after which we continue the for-loop, with the time-step now advanced to $\texttt{n}+t_{\text{ref}}/\Delta t$. (See Eliasmith & Anderson, chapter 4, for further details.)

i. Run the LIF model in response to a step input current. Some typical parameters are $V_{\text{thresh}} = -50$ mV and $t_{\text{ref}} = 5$ ms, but you should experiment with others. Experiment with different levels of input current to see how it affects the firing rate.

ii. For a given setting of the parameters, sweep over a range of input currents and plot the firing-rate as a function of input current. Explain the shape of the firing-rate curve in terms of the leaky-integrator dynamics and the parameters you chose.

iii. Now consider a time-varying input current as shown below.



Drawing upon the methods described in Eliasmith & Anderson, chapter 4, show how this input could be nonlinearly encoded into spikes using two LIF neurons: one of the on-type and the other of the off-type. Simulate these neurons in response to the input current, and plot their spikes along with the signal. (*Hint:* The `plt.eventplot` function from `matplotlib` is useful for plotting spikes.) You may need to experiment with the parameters of your leaky-integrator and spike generator to find a regime where the spikes capture the transitions in the signal.

iv. Show how the LIF encoded signal can be decoded by downstream post-synaptic processes by convolving with the reconstruction kernel given in the collab, and plot the reconstruction alongside the input current. Note this kernel is only approximate as it was derived from a specific setting of LIF parameters that may not match yours. You can derive derive a more optimal kernel (optional) by following the methods described in Eliasmith & Anderson, chapter 4.