Theoretical and Computational Neuroscience 2018

Problem Set 1: The Integrate-and-Fire Model

Due date: 9:00am on Thursday, January 25, 2018

1 Introduction

In this exercise, we want to simulate a simple model of a neuron called Integrate-and-Fire model (read Dayan and Abbott, p.162-165. For further information start at p.153). In this model, the dendritic tree is not simulated. We assume that all inputs from the dendritic tree (excitatory and inhibitory) are summed up in some way to build a total incoming current. The neuron accumulates this total incoming current in the soma until a voltage threshold is reached. This will trigger an action potential thereby resetting the membrane potential back to its resting potential. Because the cell membrane of the neuron is leaky, the neuron will lose its accumulated charge over time. Therefore the neuron will not fire if the accumulation of charge is too slow. We will make the simplified assumption that the resistance (leakiness) of the cell membrane is constant.

2 The basic equation

If we neglect for a moment that the neuron will fire an action potential after reaching a voltage threshold, the change in membrane voltage (written symbolically as $\frac{dV(t)}{dt}$) can be described by the following differential equation (Don't be scared if you do not know what a differential equation is. The meaning of the equation is very intuitive!):

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

Here $\tau_m = 0.010\ s$ is the so-called membrane time constant, describing how fast current is leaking through the membrane. E_L is the resting membrane potential (-0.065 V), while $V(t)$ is the actual potential as a function of time. R_m is the constant total membrane resistance (10^7 Ohms) and $I_e(t)$ is the fluctuating incoming current.

3 Numerical simulation

The equation tells us how much the membrane voltage is going to *change* depending on $V(t)$ and $I_e(t)$. To get the voltage at some later time, we have to *sum up* the small changes $dV(t)$ for the small time steps dt. For this we need to know $I_e(t)$. The procedure is:

1. Calculate $dV(t)$ by using the current values for $V(t)$ and $I_e(t)$.
2. Add $dV(t)$ to $V(t)$ to get the new voltage at $V(t + dt)$.
3. Check if the threshold voltage for producing an action potential, $V_{\text{th}} = -0.050\ V$, is reached. If so, set $V(t + dt)$ down to the reset voltage $V_{\text{reset}} = -0.065\ V$. We do not simulate the action potential itself. We simply treat the action potential as a discharge and reset the voltage.
4. Return to 1, but now use values $V(t + dt)$ and $I_e(t + dt)$.
5. Repeat until t_{final} is reached.

To calculate $dV(t)$, we rewrite formula (1) as:

$$dV(t) = \frac{1}{\tau_m}((E_L - V(t)) + R_m I_e(t)) \cdot dt \tag{2}$$

Our calculation scheme will be only accurate if we take dt small enough – smaller than 0.001 s. The external currents I_e will have to be very small (in the 10^{-9} Ampere range) to mimic actual conditions. Note that quantities in MATLAB don't have units (like Amperes and seconds) so you should simply input parameters in a consistent set of units and interpret the results according to that.

4 Questions

1: Describe equation (1) in words (p.163 may help).

2: Make yourself more familiar with arrays in MATLAB by reading the *Getting Started* part of the MATLAB product help. You have to know how to initialize an array and how to access the different elements. Also read about flow control commands in MATLAB, for example *if, for, continue....* You do not have to turn in anything for this part of the homework.

3: Now implement the Integrate-and-Fire model for constant input currents. Let $I_e(t)$ be a positive constant in a reasonable range and set the parameters as described above. Choose dt to be small enough to get smooth results (this will depend on your choice of $I_e(t)$). Plot the voltage as a function of time including a number of voltage resets as the threshold potential was reached. (If you cannot get resets, your I_e is probably too small.) Comment your code properly and add it, along with all later code, to the printout of your homework!

4: Try different constant values for I_e and produce a graph showing how the firing rate changes with I_e. (The firing rate is defined as the number of threshold crossings (action potentials) per second.) You can either do this by hand or write a MATLAB program doing it for you. What is approximately the minimum value for I_e that the cell starts to produce action potentials at all?

5: Try two other choices for $I_e(t)$ (A periodic function for example). Add the graphs showing the current and voltage traces to your notes.

Bonus problems:

6: Real neurons are noisy. Add Gaussian noise to the Integrate-and-Fire model with a constant current input. You can do this by generating a random number from a Gaussian distribution (MATLAB command: `randn`), and adding it to the voltage at each time step. Investigate how adding noise affects the timing precision of spikes (i.e. how much jitter there is in the time at which spiking thresholds are crossed).

7: Real neurons have a refractory period. This can be modeled in our integrate-and-fire model by adding an extra inhibitory current (p.165-166). The equation for the spike rate changes to:

$$\tau_m \frac{dV(t)}{dt} = (E_L - V(t)) - r_m g_{sra}(t)(V(t) - E_K) + R_m I_e(t) \tag{3}$$

The spike-rate adaptation conductance $g_{sra}(t)$ is given by the differential equation:

$$\tau_{sra} \frac{dg_{sra}(t)}{dt} = -g_{sra}(t) \tag{4}$$

given a time constant τ_{sra} (see eqs. 5.13 and 5.14 in Dayan and Abbott, and Fig. 5.6). Whenever the neuron fires a spike, g_{sra} is increased by an amount Δg_{sra}. You can set the initial value of g_{sra} to zero. Implement this improvement to the Integrate-and-Fire model without noise (using the values $r_m \Delta g_{sra} = 0.06$, $\tau_{sra} = 0.10\ s$, and $E_K = -0.07\ V$, as in caption to Fig. 5.6) and investigate how the refractory period depends on the parameters of the extra current.

For people new to MATLAB, one purpose of this first homework set is to get you used to this programming environment. You will probably have to spend some time reading the MATLAB help. Even if you cannot immediately apply the knowledge you gain to the homework problems, getting control over MATLAB will help you for later exercises.

Bonus Problems: You do not have to do the bonus problems, and they do not count for the grade on the homework. But we will keep track of the points you score on them, and can use these to "bump up" grades for students who wind up just below a grade boundary at the end of the semester.

Turning in the homework: Please upload the following files to the Canvas Assignment Homework 1:

1) Your legible and commented code. If you have multiple files (for example, for different problems), please indicate which can be run to reproduce the output for each problem in file name. Please be sure that your code runs! If it does not, it partial credit may not be awarded for incorrect outputs.

2) A word document containing your answers to each question, with your MATLAB output figures embedded where necessary. Please include a caption accompanying each figure. Be sure to label your axes.