Neurobiology 204 – Unit #8: Neuronal Variability Matlab Simulation

Overview of this assignment
The goal of this assignment is to familiarize you with different temporal patterns of spiking found in cortical neurons. First, read Shadlen and Newsome (1998) for a detailed introduction to the topic. The basic issue is that neurons in the cerebral cortex receive a large number of inputs (~10,000), many of which are excitatory and most of which are from nearby neurons that likely share response properties. In this “high input regime” how do neurons remain sensitive to changes in their inputs?

For the written component of the homework you will write a ~2-page narrative summarizing the results of your simulations. You will characterize the spiking statistics of a “probe neuron” that receives varying inputs whose properties you can control. We created two MATLAB classes, the probeNeuron class and the Ensemble class. Typically, you will model the probe neuron receiving inputs from one or more ensembles consisting of classes of excitatory or inhibitory neurons. Think of the ensembles as the pools of neurons that synapse onto the probe neuron and influence its membrane potential. The spiking of each ensemble is created internally in the following way: First, a seed spike train is generated, in which all the inter spike intervals (isi) are derived from a gamma distribution. You can change the shape of this underlying isi distribution by changing the regularity of the ensemble. Second, each ensemble neuron will derive its own spike train from this seed spike train. The asynchrony parameter governs how closely a given ensemble neuron follows the seed spike train; large values make the neurons very asynchronous.

A note on difficulty and collaboration:
Depending on your MATLAB comfort level, this simulation may seem daunting. We’ve tried to make it accessible, so you can play with the simulation without being a code warrior. Most problems below can be answered using minimal programming; you don’t need to write a lot of code to get a good grade on this assignment. We’ve suggested “novice”, “intermediate”, and “advanced” versions of the exercises below. Choose one that reflects your interest and comfort level, or, if you’ve got your own ideas, feel free to ignore our suggestions and do your own thing. We encourage you to discuss your findings with your classmates, but this exercise is only useful to the extent that you actually roll up your sleeves, get your hands dirty and dig into the simulation. To that end, as for the previous simulation exercise (unit #2) please collaborate in English, not by sharing code.

A note on grading criteria
Your narrative will be graded on the basis of several criteria. (1) Is your narrative organized and logical? Are your results displayed in a clear way in one or multiple figures? (2) Do you clearly state the interpretation of your results (i.e., why the model responded the way it did to your “experimental” manipulation)? Is this interpretation reasonable and have you given it some real thought? (3) Did you display creativity in designing your
“experiment”? It’s fine, if you just follow the instructions and make minimal parameter changes, but a really creative angle will also win you some points.

**Getting started**

0. Download the Shadlen and Newsome (1998) paper and read it. This is a long and technically challenging paper, so you should focus on Results sections 1 and 2 (consider section 3 optional).

1. Unzip the file VarSim2015.zip. It will create the base code for your simulation in your directory.

2. Start Matlab, navigate to VariabilitySimulation/ and, in the command window, type

   ```
   >> startup
   ```

   Matlab will respond with ‘Ready’.

3. You will be working with two classes, ProbeNeuron and Ensemble.

   a. **ProbeNeuron** constructs leaky integrate-and-fire neuron objects. These objects take spike train input and produce membrane potential and spike train output.

   b. **Ensemble** constructs objects containing spike train output from a neural population.

4. Let’s create a **ProbeNeuron** object and connect it to an **Ensemble** object. From the command window, create a **ProbeNeuron** object called **pn**

   ```
   >> pn = ProbeNeuron()
   ```

   The object’s default properties will be displayed. Note that the last four properties are empty []. We will soon populate them.

5. Create one **Ensemble** object. By default, it will have 100 neurons, each firing 50 spikes s⁻¹ on average.

   ```
   >> e = Ensemble()
   ```

6. Now, wire the **Ensemble** object to the **ProbeNeuron** object by calling one of its methods **addEnsemble()**. **Note** the **ensembleInput** property is no longer empty.
>> pn.addEnsemble(e)

7. We are ready to simulate one trial of pn by calling its method showResults(). This will get input spike trains from e, compute and plot the output membrane potential and spike train—both to be stored in the Vm and spikeTrain properties of pn.

>> pn.showResults()

8. Figure 1 shows the membrane potential of the probe neuron pn. Figure 2 shows the spike raster of the input ensemble e. Every spike in figure 2 depolarized the membrane of the probe neuron a small amount (specified in the synaptic strength parameter). Adjust them on your screen so that they are vertically aligned. These figures are now dedicated to pn and e; don’t close them.

9. Note how the barrage of input from e is making pn fire, reset, and fire again repeatedly. Run another trial and observe how this property persists.

   >> pn.runTrial()
   >> pn.showResults()

10. Let’s change the spike rate of the ensemble object and observe the outcome

    >> e.meanRate = 20
    >> pn.runTrial()
    >> pn.showResults()

11. You can access and change any property of pn or e in this manner. For instance, note how the voltage decay between spikes speeds up when you change the leaky membrane’s time constant. Let’s freeze the input so that we can make a direct comparison (take a screen shot of figure 1 before it changes)

    >> pn.tau = 10
    >> newseeds = false;
    >> freeze = true;
    >> pn.runTrial(newseeds,freeze)
    >> pn.showResults()

This concludes the warm up. These classes can be used and reused in many ways. Acquaint yourself with the properties and methods in both class files ProbeNeuron.m and Ensemble.m. You can always get a list of the properties or methods for an object, for example, by typing properties(pn) or methods(pn). You can get more
information about the arguments for method with the `help` command. For example, `help ProbeNeuron.runTrial` will return the signature for `runTrial()`.

The basics covered here have been written into a simple function file called `testParameters.m` located in the `VariabilitySimulation/` directory. The code serves as example for preparing your own simulations and also demonstrates a few other functions that may help you analyse and display your data. These helper functions and others like them are located in the `helperfunctions/` subdirectory.

Final note: Be bold. Don’t let limitations of the code stop you from pursuing an idea. You can always download a fresh copy if things get too tangled.

Revisions:
- **CGL wrote simulation code and “Getting Started” notes**
- **TSH added introduction and created homework exercises (23 March 2015)**
Homework – Choose one exercise!

Exercise number one:
Explore the variability of the inter spike intervals (isi) (regularity of spiking, or coefficient of variance (c.v.)) of the probe neuron. Why is the spiking in some cases very regular and seemingly random in others?
Novice: Change the steps required for the probe neuron to fire (either by changing the spiking threshold of the probe neuron or the synaptic strength of the ensemble input). You probably will have to adjust the ensemble’s mean firing rate as well. What behavior does the probe neuron show? Does it spike regularly or irregularly?
Intermediate: In addition to the modifications in the novice exercise, change the time constant of the probe neuron’s membrane as well. How does it influence spiking variability?
Advanced: The input ensemble can be modified to create different patterns of firing. Try different regularity parameters of the ensemble. This parameter (e.g. regularity) controls the shape of the distribution underlying the interspike intervals in the ensemble (i.e. the shape parameter of the gamma distribution we use to choose the isi). How does the regularity of the ensemble’s spiking affect the probe neuron?

Exercise number two:
Another way to change the variability of the probe neuron spiking is by adding an inhibitory ensemble.
Novice exercise: Add a second ensemble consisting of inhibitory neurons (make the synapticStrength parameter negative). How does the spiking regularity in the probe neuron change?
Intermediate: As in the novice exercise, add an ensemble with inhibitory cells. How does the number of inhibitory cells change the spiking of the probe neuron?
Advanced: On top of the modification of the novice and intermediate exercise, change the synchrony of the input ensemble (change the e.asynchrony parameter, which is the shape parameter for the Laplace-distributed jitter around the seed spike train for each neuron (High synchrony: 0)). How does it affect the regularity of the probe neuron?

Exercise number three:
In this exercise you have to explore beyond a single response of the probe neuron and investigate how noisy spiking generated randomly (random noise) or with a given seed (frozen noise) affects the spiking across repeated trials. You will have to do some real coding for this exercise.
Advanced: Provide the seed used by the random number generator (rng) to create the interspike intervals for the seed spike train. The important code is in Ensemble class in its method gammaTimes(). Type ‘help rng’ to see how to use the same seed for each trial. What does c.v. tell you? Estimate the Fano factor for both cases (random seed and frozen seed), and contrast it to the c.v.
Tip: Look at the helper functions to calculate the Fano factor.

Have fun! Email Till (till@hms.harvard.edu) if you have any questions!