NENS 230: Analysis Techniques for Neuroscience using MATLAB
Fall 2011

Assignment 2: Frequency tracking in opsin-expressing neuron
DUE DATE: 9am, Monday, October 10th

Please submit this assignment to nens230@gmail.com, with the subject line "hw2". Detailed assignment description is below, but in short, you will submit one function, *countSpikesByLightPulseFrequency.m* and two figures which should be in the .png format. You should send these files in a .zip file attached to your email called hw2_SUID where SUID is your SuNET ID (e.g. hw2_djoshea.zip).

This second assignment is intended to demonstrate the use of multidimensional arrays, positional indexing, conditional operators, logical indexing, and the find command.

Neuroscience Background

The patch clamp technique is a method that facilitates the study of electrochemical signaling and ion channel physiology in excitable tissue. Neuroscientists typically prepare live slices of brain tissue and study the intracellular dynamics of individual neurons. Using a very thin electrode (a glass pipette with a very narrow tip containing conductive saline solution) to create a high impedance seal against the cell membrane of the neuron and then applying suction to create a low impedance connection to the intracellular space, the patch clamp technique allows neuroscientists to record ionic currents flowing into and out of the cell and observe the effect of pharmacology or other manipulations on these currents.

In current clamp mode, the electrode is connected to a constant current source which injects a fixed amount of current into the cell (typically on the order of tens of picoamperes). Simultaneously, the experimenter can record the intracellular membrane voltage of the cell, that is, the voltage difference from the inside of the cell to the extracellular medium. When the current injected is zero, the current clamp effectively serves as a voltmeter which passively records the membrane voltage of the neuron as a function of time.

In this assignment, you will be provided with a data file in the Axon Binary Format (abf), as used by the pCLAMP electrophysiology data acquisition system [http://www.moleculardevices.com/Products/Software/pCLAMP.html]. We will load

the information found in this file into MATLAB using a utility called abf2load, which we have downloaded from Mathworks Central for you [http://www.mathworks.com/matlabcentral/fileexchange/22114-abf2load].

The neuron that was patched in this experiment was expressing ChR2, which is a light gated cation channel. When blue light illuminates this neuron's cell membrane, positive current is allowed to flow into the neuron and depolarizing it, leading to an action potential. The purpose of this experiment was to determine how well the neuron could track periodically delivered 2 ms pulse trains of light at various frequencies.

When this data was recorded, pCLAMP was setup to record two signal channels simultaneously. The first is the membrane voltage of the neuron; the second is the laser signal used to activate the light source. The experiment is divided into multiple sweeps, each having the same duration in time. For each sweep, the file records the value of the signal on both channels as a function of time. The laser signal is programmed to generate pulses at a specific frequency on each sweep, and this frequency is varied across sweeps.

Assignment Description

Your task is to count the number of spikes fired by the neuron on each sweep, as well as to compute the laser pulse frequency used on each sweep. Using this information, you will plot the number spikes evoked against the laser pulse frequency. However, in order to do this, you will need a mechanism to detect spikes in the membrane voltage signal, and pulses in the laser signal. We will accomplish this by setting a threshold for each channel, and then detecting the times when the signal crosses this threshold from below.

The code provided in countSpikesByLightPulseFrequency.m provides a good starting point. As you can see in the function signature (the specification of what the functions input arguments and returned output arguments are, found on the first line of the .m file), you can call this function like this:

```
abfname = 'recording.abf';
[nSpikesEvoked pulseFrequencyHz] = countSpikesByLightPulseFrequency(abfname)
```

For more information on the function's purpose and its inputs and outputs, run:

```
help countSpikesByLightPulseFrequency
```

```
[nSpikesEvoked pulseFrequencyHz] = countSpikesByLightPulseFrequency(abfname)
  Computes a frequency tracking curve for an light-sensitized neuron
  recorded in current clamp under periodic optical stimulation.

  This function accepts an .abf file name compatible with the abf2load
  utility that contains current clamp recordings (channel 1: membrane
  voltage) along with a laser activation signal (channel 2: laser state).
  Using a simple threshold, it detects how many action potentials the cell
  fires in a given sweep, then computes the light pulse frequency used on
  that sweep, and generates a plot of number of spikes evoked vs. light
  pulse frequency. It also generates a plot for each sweep in the file
  which shows visually where the detected spikes and laser pulses are
  located, allowing visual debugging of the detection algorithm.

  INPUTS:
  abfname - *.abf file name to be loaded by abfload()

  OUTPUTS:
  nSpikesEvoked - nSweeps x 1, how many spikes were evoked on a given sweep
  pulseFrequencyHz - nSweeps x 1, laser pulse frequency used on a given sweep
```

Notice that the help command simply echoes the long comment located at the top of the .m file for that function. For your own functions in the future, be sure to write a similarly helpful comment that indicates what the function does and how to use it, including what each of the inputs and outputs represent.

Essentially, the code includes comments which indicate what you're supposed to accomplish on the next line, and places where you need to edit the code are indicated by ???. I've chosen the variable names for most things for you, mainly so that I could refer to them by name in the comments, but feel free to change these if you prefer something else. Moreover, the structure of the code as written is only a suggestion. Feel free to accomplish this however you prefer. Many of the threshold crossing detection steps could be combined more efficiently onto one line of code if you so desire. The code is written for maximal clarity and instructive value, not for conciseness. Also, if you feel motivated to do a better job detecting spikes in the signal, say using a Schmitt trigger [http://en.wikipedia.org/wiki/Schmitt_trigger], feel free, though it's not necessary for the assignment.

Deliverables:

Please submit your completed countSpikesByLightPulseFrequency.m function. Save

the final light frequency tracking figure to tracking.png image [File -> Save as on the figure window]. Also, save the first individual sweep plot generated with the membrane voltage and laser state signal annotated with the detected threshold crossings. One of these plots is sufficient. Save it as sweep1.png. Zip up these three files to hw2_YourSUID.zip (e.g. hw2_djoshea.zip) and attach it to an email at nens230@gmail.com.

Good luck!