NENS 230: Analysis Techniques for Neuroscience using MATLAB
Fall 2011

Assignment 3: Frequency tracking comparison between Opsins
DUE DATE: 9am, Monday, October 17th

Please submit this assignment to nens230@gmail.com, with the subject line "hw3". Detailed assignment description is below, but in short, you will submit two functions, *loadPatchingData.m* and *compareTrackingByOpsin.m*, and one figure which should be in the .png format. You should send these files in a .zip file attached to your email called hw3_SUID where SUID is your SuNET ID (e.g. hw3_djoshea.zip).

This third assignment is intended to demonstrate the use of functions to encapsulate code, MATLAB's file reading and parsing commands, using data structures to organize data, and then aggregating data to plot a comparison.

Neuroscience Background

This assignment builds off of the function you wrote in assignment 2. The idea is to analyze data recorded from several different neurons which were expressing a given opsin, and then to average their frequency tracking curves together. By repeating this process for several different opsins, we can get an idea of how one opsin performs relative to another, in the sense of how well expressing neurons follow a periodic train of light pulses at various frequencies.

However, it's worth highlighting that your function from last week simply counts the number of spikes that are evoked in a given trace. Therefore, neurons which fire multiple spikes (e.g. doublets, triplets) in response to each light pulse would tend to show a large number of evoked spikes, even if they trail off with later pulses due to the development of a plateau potential. In fact, this is common with ChR2-H134R[1]. Consequently, when interpreting these plots, larger numbers of spikes evoked may not indicate "better" frequency tracking. A better approach to this would be to divide the trace into small time windows following each light pulse and to count the number of these windows which have one or more spikes. Counting the number of windows which have two or more spikes would indicate how often doublets or spike multiples are evoked as well.

---

[1]See Gunaydin et al., Nature 2010. http://www.nature.com/neuro/journal/v13/n3/full/nn.2495.html

Furthermore, the spike detection mechanism used in the last assignment isn't the greatest, as it's very prone to false positives, especially for noisy traces. If you were to add noise to the traces, you'd expect an upwards going voltage signal to cross zero, then bounce very quickly above and below zero a few times simply because of noise, not because the cell actually fired multiple spikes. This actually happens in sweep 6. Visual inspection of the trace reveals 7 spikes, but correctly implemented solutions will detect 8 spikes in this sweep. Zooming in on the third spike revels why: the trace actually crosses zero twice at two very closely spaced samples. Perhaps the simplest way to prevent this is to impose a hard refractory period, that is, throw away spikes that follow another spike by < 2 ms or so. If you feel like attempting this, try looking at diff(spikeTimesMs) and throwing away spikes where this time delta is less than some threshold. Another solution is to require the voltage signal to fall below some second lower threshold (say, -10 mV) before recognizing further high threshold crossings as spikes (in other words, the voltage must go above 0, then below -10, then above 0 in order for two spikes to be detected). This method is known as a Schmitt trigger, and is used in electronic circuits to implement threshold crossing detection in the presence of noise (e.g. button debouncing)[2].

Assignment Description

Your task is to implement two functions. The first, loadPatchData.m, will load information stored in a csv file about a set of neurons which were patched, what opsins they contained, and what abf file each corresponds to. You could imagine generating a spreadsheet like this while you were patching, filtering it down to the data you wanted to analyze, and then exporting it as a CSV. The goal is to read this file using the textscan function and then to organize this data into a struct array, where each patched neuron is represented as a struct in the array, and each type of associated date is a field in that struct (e.g. date, neuron id number, opsin, and abf file name).

The second, compareOpsinTracking.m will accept this patchData struct array as an input. It will then figure out the set of opsins represented in this dataset. For each opsin that it finds, it will find all of the patched neurons expressing this opsin, run countSpikesEvokedByLightFrequency.m on the abf file corresponding to that neuron, and store the results. As it is doing this, it will ensure that all of the abf files use roughly the same light pulse frequencies. It will then average together the light tracking curves for that opsin, compute the mean tracking curve at standard error of the mean at each point. After doing this for each opsin, it will then generate a summary comparison plot which has one curve (with standard error bars) plotted for

[2]http://en.wikipedia.org/wiki/Schmitt_trigger

each opsin.

Use the provided runAnalysis.m script to run the code with correct arguments. This script assumes that the ABF files are located in a subfolder named Data located in the same folder as runAnalysis.m. Change dataDirectory in this script if this is not the case. Also, running your countSpikesEvokedByLightFrequency.m script on many abf files will generate a lot of figures. **Therefore, I recommend you change your countSpikesEvokedByLightFrequency.m function (from last week) such that it always uses the same two figures for everything. In other words, find the first call to `figure(iSweep)` and change it to `figure(1)`. Then, find the second call to figure; and change this to `figure(2);`.** This will prevent you from dealing with dozens of figures when debugging. Also, you can use the close all command to close all figures.

Remember, that since you're calling countSpikesEvokedByLightFrequency.m, which calls abfload.m, that both of these functions will need to be somewhere on your path. You can add the Assignment 2 folder that they live in to your path using File -> Set Path.

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.

Deliverables:

Please submit your completed loadPatchData.m and compareOpsinTracking.m functions. Save the final light frequency tracking comparison figure to an png image titled compare.png [File -> Save as on the figure window].

Good luck!