NENS 230
Assignment #5
Good Coding Practices

Improving an Event-Triggered Average Function

Due: Tuesday, October 27th, 2015

Goals

- Practice reading and understanding code that someone else has written, including debugging code.
- Practice improving the readability of code using good style.
- Practice improving the efficiency of code by pre-allocating variables and "vectorizing" mathematical operations.
- Improve the modularity of code by breaking pieces out as separate functions.

Introduction

In this assignment you apply the MATLAB best practices that were covered in Lecture 5 to improve an almost-working, but very poorly-written, function that processes data from a neuroscience experiment. You will run the function with the provided data to see how it works, find and fix the one bug, and then go through the code and improve its clarity, modularity, and performance.

The goal of this assignment is to practice good coding style, using descriptive names and comments, and writing code that is reasonably efficient. By seeing the same function before and after you improve it, you should better appreciate what a difference these changes make both in clarity and performance. Furthermore, it is useful to learn how to read through bad code others have written and figure out how it works; this is unfortunately the situation you might frequently encounter. Now you can be part of the solution, not the problem or precipitate.

Brief neuroscience background about the data and the program's goal

In characterizing properties of neurons of sensory systems, a useful concept is that of the neuron's response function, which describes what kind of sensory stimuli best evoke a response in the neuron. In the early visual system, a type of neuron called a retinal ganglion cells (RGC) will fire action potentials ("spikes") in response to a time-varying change in light intensity. The particular time-varying pattern is important, and the response of a neuron to different time-varying patterns of stimulus activity is modeled by a response function, also known as a temporal receptive field. Intuitively, we can think of the response function as the pattern of light that the cell is best "tuned" for: when time-varying light intensity adheres closely to this response function, the neuron is most likely to spike. For example, the "monophasic OFF" RGCs whose response functions are shown in yellow in **figure 1** are most likely to spike when the light falling on the photoreceptors that input to these RGCs becomes briefly darker (~0.15 seconds of negative deflection in the stimulus).

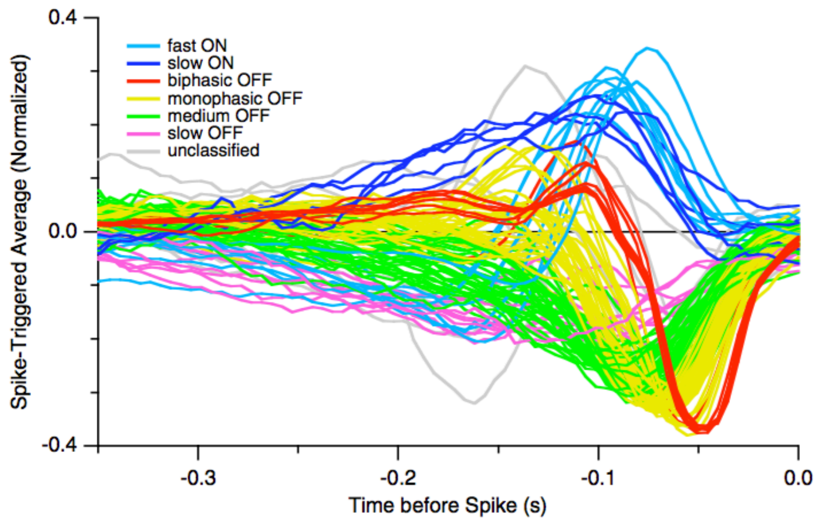

Figure 1: Response functions of several different retinal cells from the larval tiger salamander were estimated using an analytical technique called a spike-triggered average (STA). Each cells' response functions are colored according to their functional classification based on this STA. In this plot, the spike occurs at time = 0 s. Adapted from Segev et al., Journal of Neurophysiology, 2006

We can estimate the response function of a neuron using a simple analytical technique called event-triggered averaging. When applied to such data, this is called a **spike-triggered average (STA).** First, data is collected: a stimulus is applied to the experimental preparation (e.g., a recently dissected retina onto which light is projected) and the spiking activity of the neuron is recorded. Figure 2 demonstrates a snippet of such data, which is provided for this assignment. The light intensity projected onto the retina over the course of the experiment is shown in red, and the recorded spikes from the cell of interest are shown as black ticks above.

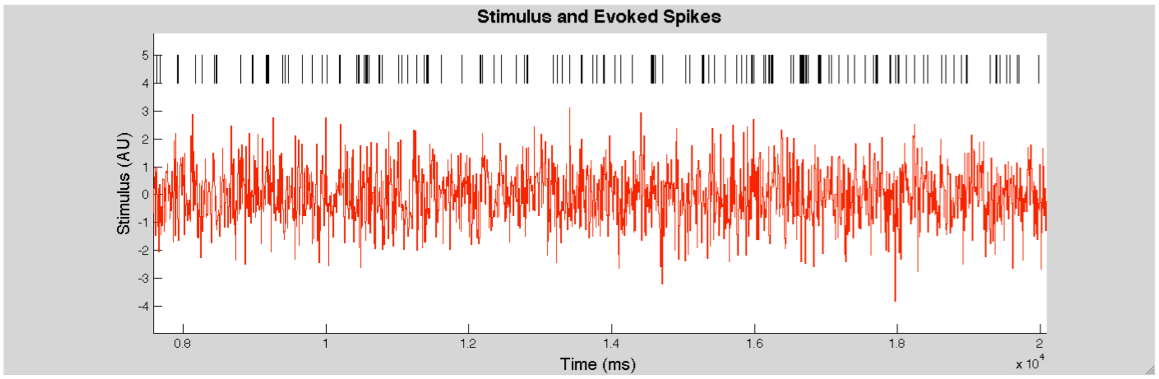

Figure 2: Snippet of the data used in this assignment. The light stimulus projected onto the retina is shown in red. The simultaneously recorded spikes of one retinal ganglion cell are marked with black ticks.

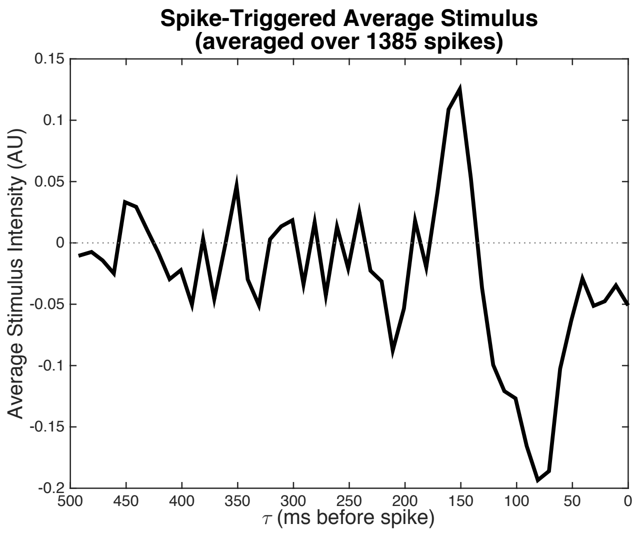

Figure 3: The code you'll improve in the assignment should have a final output that look like this.

To compute the spike-triggered average, we just average what the stimulus was immediately preceding every spike. Although there is considerable noise and non-linearity in the system, by averaging over what the stimulus was during the snippets of time preceding thousands of individual spikes, we get an accurate estimate of the neuron's response function. This is because spikes tend to happen when the preceding stimulus was somewhat similar to the "optimal" stimulus, i.e. the response function. It is very unlikely that the *exact* response function-like stimulus ever preceded a spike, but this event-triggered average approaches this optimal stimulus (pretty cool, right?).

It is also important to consider how far before each spike we want to look at when computing the STA. We refer to the delay between the time of a spike and the preceding stimulus as tau (τ). So τ=0 ms is the time of the spike, and τ = 10 ms refers to 10 ms preceding the spike. While it is possible that whether or not a neuron spikes is dependent on what the stimulus was many seconds beforehand, in the retina this is not the case. Thus, if we look at the STA more than 300ms before each spike, the average stimulus tends to approach zero. This is what we expect to see if the stimulus that far back does not affect the likelihood of a spike: there would be approximately equal number of spikes with positive and negative stimulus intensity 300+ ms before the spike.

Note that while this example comes from neurophysiology, the idea behind an event-triggered average is applicable whenever you measure two variables, and you want to see whether one of the variables (here, the light intensity) has some characteristic behavior immediately before/after some particular "event" that occurs in the second variable (here, when spikes happen).

Detailed Instructions

You are going to fix up a buggy and poorly-written function that computes the spike-triggered average stimulus from data containing the stimulus and the spike times. The program is supposed to display the STA after averaging over 10 spikes, 500 spikes, and finally after all the available data (this lets you see how averaging more data improves the estimate of the neuron's response).

Unzip the Assignment5 file and copy its contents to an Assignment5 directory within whatever directory you keep work for this class. You are provided the STA-computing and results plotting function `SpikeTriggeredAverageStimulus.m` as well as a "start" script `MAKESTA.m` that just calls the function with the appropriate arguments. `MAKESTA.m` also times how long execution took, which can help you "benchmark" your progress. The spike time data is contained in the file `RGCspikes.mat`, and the stimulus is described in the file `VisualStim.mat`.

Start by running `MAKESTA.m`. You'll notice that the code breaks somewhere in `SpikeTriggeredAverageStimulus.m`. Your first task is to debug this function. **Hint:** You might want to put a breakpoint right before the line generating the error is executed, so that you can see the values and sizes of variables and make sure it's doing what it's supposed to be doing. A small fix in just this one line will make the code work correctly.

Once you've identified the bug, run `MAKESTA.m` again see the spike-triggered average be computed and displayed. Notice how it takes a long time to run? When you are finished with the assignment, running `MAKESTA.m` should give the same result, but much faster.

You should now edit `SpikeTriggeredAverageStimulus.m` and also break out oft-repeated pieces of it into separate functions (contained in separate .m files) called by `SpikeTriggerAverageStimulus.m`. Follow the best practices lecture to improve the following aspects of the code:

1. Add proper documentation, headers, and comments
2. Use sensible argument, variable, and function names.
3. Use good style; make the code neat and readable (e.g., indent if then statements, add blank lines between sections of code).
4. Simplify the `SpikeTriggerAverageStimulus.m` function by moving some of its code into one or more separate functions. If there's a block of code that seems to be repeated almost verbatim in several places (**hint**: the STA is plotted 3 different times), maybe it should be a separate function?
5. Improve the code's performance. For example, not constantly displaying text to the console, preallocation, and vectorization, are the low-hanging fruit.
6. Another hint on how to speed things up: do you need to even look at all the spikes if there isn't stimulus accompanying some of them? If you just threw

out those without corresponding stimulus, would you need the conditional check that's currently in the main loop?

7. The last two input arguments to `SpikeTriggeredAverageStimulus.m`, <`tau`> and <`ignoreFirstNms`>, should be optional arguments. You can make up reasonable defaults for them. The idea behind the latter argument is that sometimes we want to not analyze data from the first few seconds of the experiment if we believe that the retina is still adapting to being exposed to light.
8. If you feel ambitious, there is a slight change in the algorithm that can further speed things up and reduce how much memory is needed. Hint: do you really need all the intermediate results? Note that this makes only a marginal performance improvement.

Through optimization you can considerably speed up the execution time of this program. On a 2013 MacBook Pro 13", the poorly-written code runs in 46 seconds using the data and input parameters provided, whereas the improved solution code runs in <3 seconds.

Note that the `plotStimAndSpikes.m` function is not part of the intentionally badly-written code in this assignment. **You do not need to improve `plotStimAndSpikes.m`.** It's just there to help you see what the raw data you're working with looks like; when you run `MAKESTA.m` it will plot the stimulus intensity over time and the recorded spikes for you. You can zoom in using the plot tools to see what's going on. Seeing this raw data should reinforce that that taking an event triggered averaged over thousands of events has a seemingly magic effect: you can pull out a meaningful response function of a neuron from a seemingly random jumble of stimulus and spike times.

Submission

When you are finished, you will submit everything in your Assignment 5 directory. We will be looking at your improved `SpikeTriggeredAverageStimulus.m` function as well as any additional function(s) you wrote that `SpikeTriggeredAverageStimulus.m` calls. You should send these files in a single (zipped) folder attached to your email called hw5_SUID, where SUID is your SuNET ID (e.g. `hw5_sstavisk`). Email this zipped folder to nens230@gmail.com with subject line `[assignment 5]`.

Extra Credit

Whoever writes the SpikeTriggeredAverageStimulus program that executes the fastest when run on the grader's machine will win fame, glory, and an LKSC pastry.