Problem Set 1: Detecting Spikes

Problem Set 1 will give you the opportunity to write Python code that makes use some of the Python syntax and commands we have introduced to you in Unit 1. In Problem Set 1, you will write a **Spike Detector** which will return the times that an action potential occurred in the electrical signal recorded from a neuron.

Why do neuroscientists need spike detectors?

As we learned in Unit 1, neurons use electrical and chemical signals to communicate. As information is received by a neuron, typically via chemical synapses with receptors on the neuron's dendrites, changes to the neuron's electrical membrane potential occur. These changes are summed, both temporally and spatially. If the change in membrane potential is large enough at a particular instant in time, a threshold is crossed and the receiving neuron will sends an output signal down its axon. This output signal is sent via an action potential (also called a spike), originating at the axon hillock. Once the action potential reaches the axon terminal, neurotransmitter can be released across chemical synapses, thus allowing this neuron to convey its message to downstream cells. Therefore, if we know when a neuron is firing action potentials (spiking), and how often these spikes occur, we can try to interpret both the inputs that may be driving the cell to fire and the message this neuron is conveying within its neural circuit. Spike detectors allow us to isolate these characteristic changes in the electrical activity of the neuron so we can to start to analyze what the cell might be doing.

To complete this problem set, you will

- Write a function that locates action potentials in an electrical signal;
- Write a function to plot the shape of the action potentials you have located;
- Write a function to display the locations of the action potentials within a given signal.

At the end of this problem set, you will

- Complete a programming assignment submission which will evaluate that your function locates the action potentials at the correct times;
- Complete a peer assessment submission to confirm that you are accurately plotting the both the shape and times of your action potentials. (To receive credit for the peer assessment, you must complete your peer evaluations by the evaluation due date.)

To help you accomplish this task, we will provide you with the following

- A function that does a HORRIBLE job finding action potentials
- A function that lets you check the performance of your action potential locator (when we give you the correct answer)
- Two example electrical signals, with the correct spike times, so you can check your work.

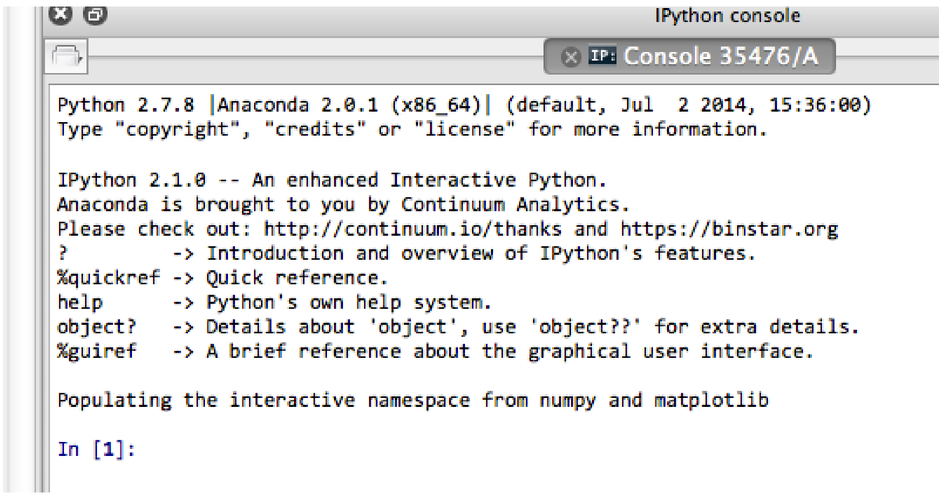
Things to keep in mind while you are coding

- It's really easy to make mistakes when you are writing code. It happens to everyone. Try not to get frustrated. Ask for help if you need it!
- Sometimes it is useful to print out the values of some of your variables to make sure things are going as you want them to (or you can use the debugger in Spyder – but printing is sometimes much easier!)
- It is often very useful to write out your logic with pencil and paper before you begin.

Getting started with Spyder!

Your first step is to open `problem_set1.py` in Spyder (or whatever Python environment you are choosing to use.)

Before we browse through the code in that file, you will also want to have a "console" started in the Spyder environment. The console allows you to type interactive commands. To ensure you have a console running, you can go select "Consoles->Open an IPython console" from the Spyder menu bar. This will usually be started automatically and you can recognize the IPython console by the prompt, which will look something like this:



```
Python 2.7.8 |Anaconda 2.0.1 (x86_64)| (default, Jul  2 2014, 15:36:00)
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

Populating the interactive namespace from numpy and matplotlib

In [1]:
```

OK, let's look back at the file. You'll see that it starts with some comments (indicated by the # signs). Python ignores this, but you should always comment your code so other people (or you, when you come back later to look at it) can understand your code.

Next you will see some lines of code that begin with `import`. These lines allow us to use some built in Python functions stored in separate modules. Particularly, we will use commands from the modules called *NumPy* and *matplotlib*.

Below that, you will find some functions (and some skeletons of functions). We will work through these as we go through the problem set. Below all the functions, you'll see some example code snippets.

Loading some real data

First, we've provided you with a function called `load_data` that you can use to read in the provided data files. We can see how to use this function by loading the file into the interpreter. To do this in Spyder, just press the green "go" key on the button bar. This will evaluate the code in the file we are currently looking at.

When you do this, the a command will automatically be pasted into the console that starts with `runfile(…)`. The code in our python file will be run.

If we do this for our `problem_set1.py` file we will define the functions in that file. Then, we can test these from the IPython console.

You can click your mouse over to the console and type:

```
t,v = load_data('spikes_example.npy')
```

This reads in the example data set and stores the data in two arrays. (Arrays work very similarly to the lists we introduced to you in the Python videos. They have some additional functionality as well, but you should be all set to work with them based on what you know about lists). The two arrays, assigned here to the variables t and v, contain the timestamps for the recorded signal and the recorded voltages, respectively.

You can explore the arrays that are returned from `load_data()`by, for example, seeing how long the arrays are. You can try things like:

```
len(t)
max(v)
t[0]
t[-1]
```

Each of these should return a result at the command line.

Plot the data

A great way to start to look at any kind of neural data is to plot what you have. Having run `load_data()`, you now have a bunch of times and a bunch of membrane voltages. Use the `plt.plot` function to see what we gave you. You should be able to clearly see the spikes. (Hint: If you don't remember how to use `plt.plot` you can either review the applicable Python videos, or you can enter `plt.plot` into the object inspector or try `help(plt.plot)`The help file is pretty long – because there are lots of ways to customize plots. Don't worry about that, the info you need is right towards the top.)

(Depending on your settings in Spyder, to see your figure, you might need to call `plt.show()`to make the figure pop up.)

Great – now you should be able to see the spikes you will ultimately be looking for!

Let's start with some more visualization.

Take a look at the `bad_AP_finder` function we have provided. If you read through the code, you should see why it is so bad at its job. But we gave it to you for a good reason – your function should be structured the same way (same inputs, same format for outputs). Except you need to find the right times (but not yet…). You can try out the bad Finder by typing:

```
APTimes = bad_AP_finder(t,v)
```

Now `APTimes` is an array of all the times that we think there may be an action potential.

Let's pretend for a bit that this function does its job. We can now think about how to plot our results.

We'd like you to put a red | (vertical tick) at the location of each action potential. Put them all at a fixed y value. (For example, at a y value of 500. Or even better, at y value bigger than the biggest voltage – check out `max`).

Hints:

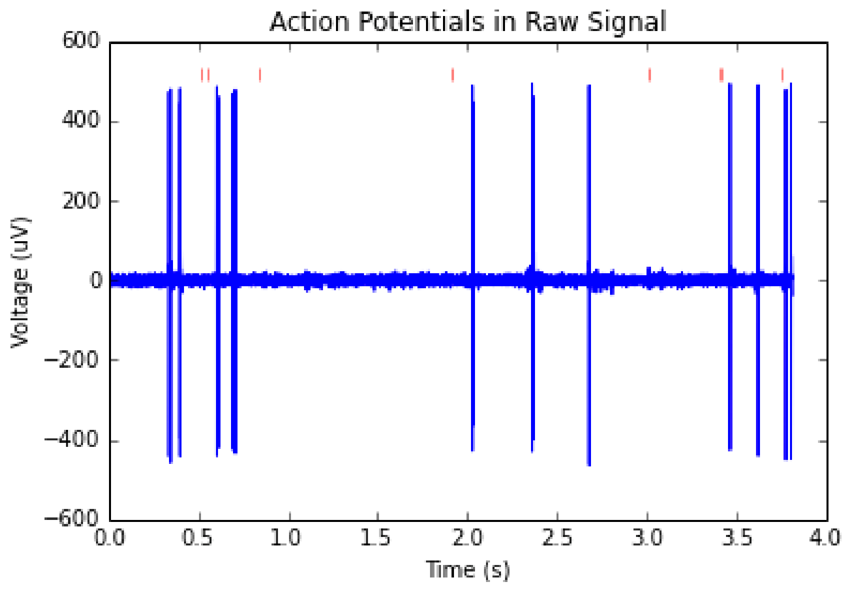
- You want to draw multiple layers within your figure. Check out the argument `hold` By using the plot command with the argument `hold=True`, your old plot won't be deleted the next time you call the plot function.
- You are interested in changing the color and shape of the marker. The help file for `plt.plot` tells you what to do.
- There is a function in NumPy called "ones", `np.ones` that takes a variable length and lets you make a vector of ones of that length:

  `p = np.ones(4)` gives you

  `p = [1,1,1,1]`

You also want to label your x and y axes appropriately and give your figure a title. (See `plt.xlabel, plt.ylabel, plt.title`)

When it works, you should see something that looks like this if you plot the data from the `spike_example.npy` set:



Notice that we have times on the x axis, not arbitrary array indices. Also notice that the tick marks are not in the right place (since the detector is bad!)

And you are going to want to draw a plot in pretty much the same way many times during this problem set. So it's best if we make a function for it! We started one for you called `plot_spikes.` It take 4 variables: time, voltage, APTimes, and the string you'd like to use for the plot title (because you may want to change this for each data set) and it should make that plot. You will be ultimately submitting some of these plots for part of your peer assessment. Remember to comment your code!

What about the characteristic shape of the action potential waveform?

We also want to plot the individual waveforms of the spikes. Depending on the electrode placement, and the type of neuron we are recording from, different neurons will have different waveform shapes, but these shapes should remain fairly consistent throughout our recording session.

The individual waveforms are just zoomed in versions of the voltage vs time plots that you just made, centered around the time of each spike. You want to plot all the spike waveforms on top of each other to make sure they look similar.

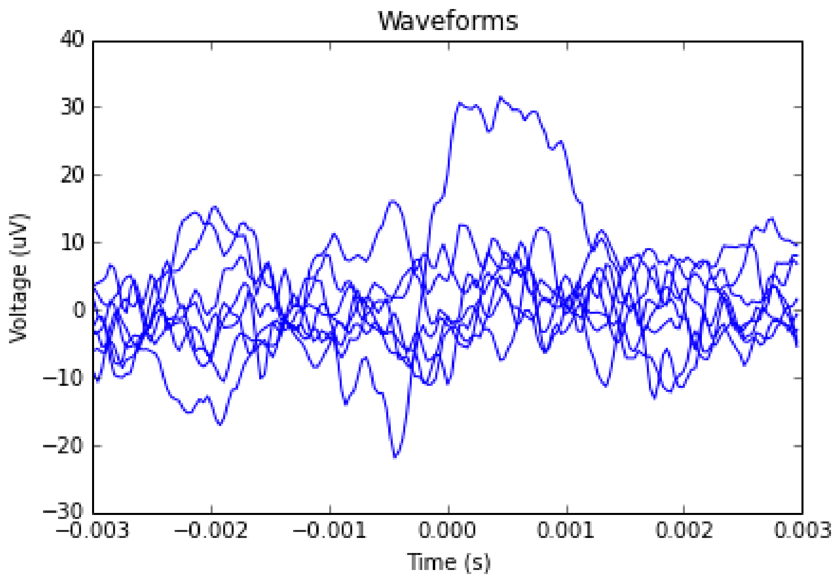
For this plot, you'll want your x-axis to go from -3 to 3ms. The y-axis should be in the same units as above.

You'll want this to be a function, as well.

Hints:

- Loops will come in handy here
- You want to consider how often the voltage was recorded (i.e. what was the sampling rate of the signal?) Keep your units straight! (Be careful about seconds vs milliseconds)
- You may need to think a bit about how to convert from the time the spike occurred to the new x-axis. This won't necessarily be straightforward for you. You may need to experiment a bit to get it right.
- A helpful function is `plt.find.` It returns the index of each element in a vector that meet a condition you can define.
- Other functions you might want to use include `range, np.linspace, min, max`
- Use help or the object inspector to see how these functions work if you aren't familiar with them.
- There are multiple ways to approach this problem, but it is a bit harder than it looks. Ask for help if you need!

You should get something that looks like this:



Notice that I didn't find any action potentials (my detector is still bad!) Your figure may look much different at this point due to the random nature of the bad detector.

Now let's try to get the times right!

So far, we've been using our very, very bad spike time finder. We've actually provided you with a function called `detector_tester()` that will show you just how bad it is (or just how great yours is, once you write it!)

This function tells you:

- The percentage of spikes that were correctly detected (percentTrueSpikes). A spike time is considered correct if it is within 2.5ms of the actual time.

- How many extra spikes you found per second of data (falseSpikeRate). This is the number of spikes you reported per second that didn't actually occur.

If everything were perfect, you'd want percentTrueSpikes = 100% and falseSpikes = 0. But data are messy, so you don't need to be perfect. Try for values like percentTrueSpikes > 90% and falseSpikeRate < 2.5 spikes/s for each data set provided.

You can run the code like this:

```
detector_tester(APTimes, actualTimes)
```

It prints out the results, so you don't need to worry about return values.

So here's how you can test out your code with the example dataset:

```
t,v = load_data('spikes_example.npy')
APTimes = bad_AP_finder(t,v)
actualTimes = get_actual_times('spikes_example_answers.npy')
detector_tester(APTimes, actualTimes)
```

Of course you'll want to substitute `bad_AP_finder` with `good_AP_finder` when you are ready!

And now it's up to you!

We've started `good_AP_finder` for you. Now add in the code to make a real action potential detector.

There's no one best way to detect an action potential, so you can try anything that seems to make sense to you. It doesn't have to be perfect; it just has to meet the criteria set above.

Remember to plot your work to see how you're doing!

Hints:

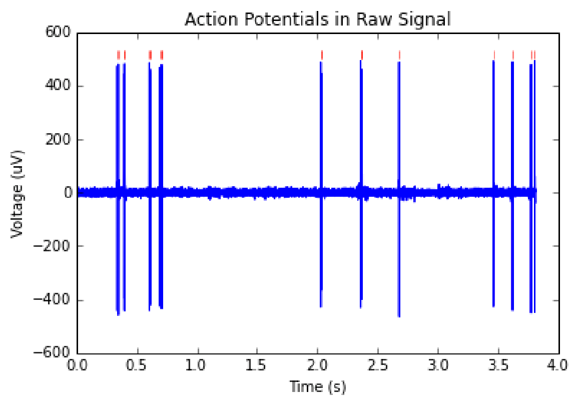
- Think about what aspects of the voltage signal indicate a spike has occurred.
- NumPy arrays can be flexibly indexed. After running `load_data()`, see, for example, what `t[v>450]` returns.
- `plt.find, plt.diff` and `np.insert` may come in handy as well (or not, there are lots of approaches).
- Run some sample lines of code in the console to see what these functions do.
- There are lots of valid approaches. All you need is one that works!

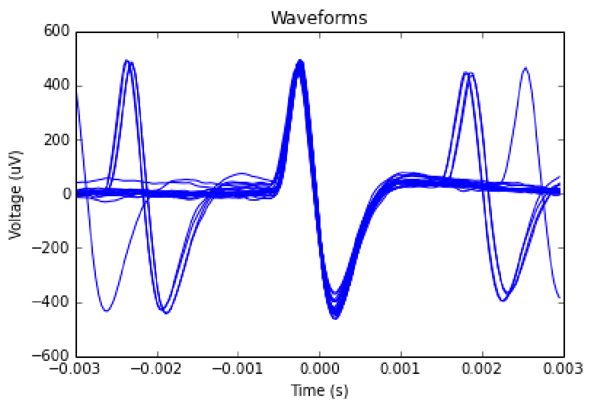
Additional hints:

- You may need to draw out your solution on paper or write things down before you start coding. For example, in this problem set, you may need to think a lot about converting between the index in an array (e.g. which element in the array am I referencing) and the time that a particular voltage occurs. Pictures can help you work through this.

- A common "setback" when working through the action potential detector code will be identifying the same spike more than once. If this happens to you, you are on the right track! Look at the hints above to see if you can get some ideas about what to do next!
- Make sure you try all the test data sets that we give you!

There are the plots we got once we implemented a better detector run on `spikes_example.npy`:





Why do you think the spikes in the waveform plot see to sometimes show spikes before and after the one we have centered? Can you see what might be going on by zooming in on the raw signal plot? Keep this in mind when peer reviewing waveform plots!

Assignment problems:

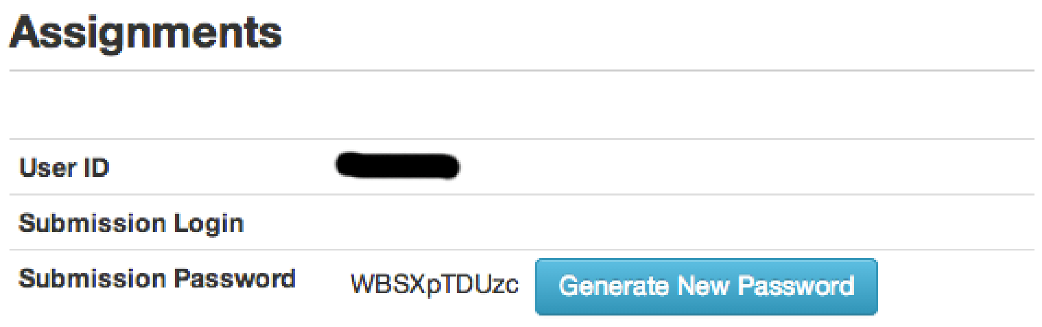
For the exercises, we want to you to try out your plotting functions and you spike finder using the `spikes_easy_practice` and `spikes_hard_practice` datasets. You will submit results of your spikefinder using the `submit_problem_set1.py` code (see **Submitting your code** below) and upload the results of your plotting routines using Peer Assignment tab in Coursera (see **Submitting your figures for Peer Assessment** below).

Optional challenge problem:

We've given you a challenge data set that shows you something that commonly occurs in the laboratory. You are welcome to try to extend your code to deal with this obstacle. The Advanced Topics: Spike Sorting video addresses this real life issue in more detail.

Submitting your code:

Once you have updated problem_set1.py to include your `good_AP_finder()` function, you can submit this part of your problem set by opening the file `submit_problem_set1.py` in Spyder. With that file open, you can run it (press the green play button or hit F5) and it will prompt you for your email address and a "one time password". You get this one time password from Coursera. On the Programming Assignments page, at the top, you should see your own *Submission Password*. This is NOT your Coursera password.



Assignments

User ID	[redacted]
Submission Login	
Submission Password	WBSXpTDUzc

If you have logged in correctly, the submit script will run your `good_AP_finder()` on new test data from the *Easy* and *Hard* test sets.

Submitting your figures for Peer Assessment

To complete the problem set, you will also need to complete the **Problem Set 1 Peer Assessment**. Here you will be asked to upload four files (the spike times plots and waveforms plots for the `spike_easy_test` and `spike_had_test` data sets). Please read the question prompts carefully to ensure that you submit the correct figure in the correct locations! You will need to save the figures (for inline figures, right click on the figure and select "Save As", for figures in their own window, just choose the save icon). You can save the files as png, jpg, gif or pdf for peer assessment.

The purpose of the peer assessment is to ensure that you are presenting your data in a meaningful way that can be easily interpreted by others. Therefore, a lot of the assessment is based on the format of your figure, including axes labels and the figure title. Each figure will be assessed in the following categories (2pts possible per category): *Please note, there are multiple ways to receive full credit – only examples are given:*

> The x-axis is properly labeled. The x-axis label should be "Time" or something similar. The units should be in either milliseconds (ms) or seconds (s), and the units given should match the numeric labels provided.
>
> > 0: The x-axis is unlabeled, or the labeling and units are both incorrect.

1: The x-axis is labeled but either the label name or the units are incorrect or not provided.

2: The x-axis is labeled properly (e.g. "Time") and units are correctly provided (e.g. ms).

The y-axis is properly labeled. The y-axis should be labeled as "Voltage" or something similar. The units should be in either microvolts (uV) or volts (V), and the units given should match the numeric labels provided.

0: The y-axis is unlabeled, or the labeling and units are both incorrect.

1: The y-axis is labeled but either the label name or the units are incorrect or not provided.

2: The y-axis is labeled properly (e.g. "Voltage") and units are correctly provided (e.g. uV).

The figure is properly titled. There are lots of options for the figure title - it just needs to make sense. For example, "Spike Waveforms" or "Action Potential Waveforms from Easy/Hard Test Data Set"

0: The figure is untitled, or the title provided is irrelevant.

2: The figure has a descriptive title.

Also, the spike time plots will be scored for clarity:

The data are clearly displayed. Both the voltage signal and the time of the spikes are clearly indicated. (Accuracy of the spike times is not required to receive full points for this criterion.) The spike times should be indicated by a red vertical tick mark (|).

0: Both the voltage signal and spike times are either missing or not presented clearly.

1: Either the voltage signal or the spike times are missing or not clearly presented.

2: The voltage signal and spike times are clearly indicated.

And for accuracy. These DO NOT have to be perfect (we allow for some spikes to be missed, or added incorrectly). We would like it to be mostly accurate. Points should not be deducted for minor errors.

The spike time markers are accurate.

0: The spike time markers do not consistently align with the action potentials.

2: The spike time markers appear to align with the action potentials.

Additionally, the Waveform plots will be evaluated for clarity:

The data are clearly displayed. Waveforms are clear. (Accuracy of the waveform shape is not required to receive full points for this criterion.)

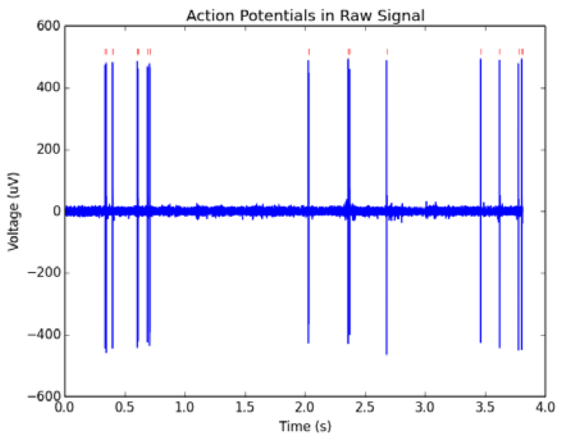
> 0: The waveforms are absent or unclear.
>
> 2: Waveforms are clearly presented.

And the Waveform plots will be evaluated for accuracy. There may be a small number of incorrect waveforms displayed in the figure. Some waveforms may not appear at zero (as in the example we showed you above.) All of this s expected, and points should not be deducted.

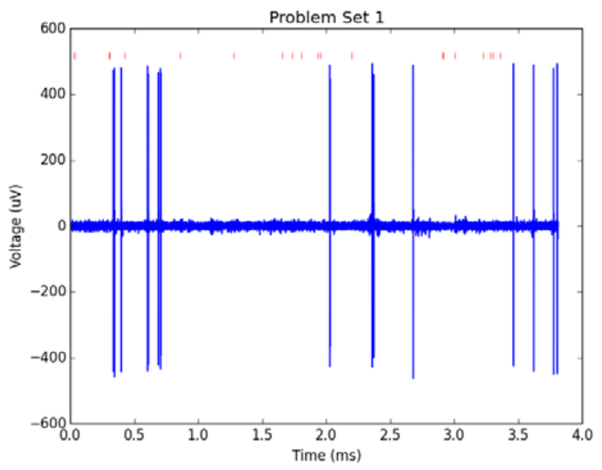
> The waveforms are accurate.
>
> > 0: The waveforms shown do not appear to be the same shape and are not occurring at time zero.
> >
> > 1: Either the waveforms do not have the same, characteristic shape, or they are not properly occurring at time zero.
> >
> > 2: Most waveforms have approximately the same characteristic shape of an action potential and are most are properly displayed as occurring at time zero.

Each figure may earn up to 10 points total, for 40 possible points of peer assessment. You must all perform 5 peer assessments, and assess your own figures (by the peer evaluation due date) to receive these points.

Here are some examples (these show data from the practice files, so your data will look different!):



This figure would receive full credit. The x-axis is properly labeled (i.e. the units shown match the axis label). The y-axis is similarly appropriate. The title is descriptive. The data are clear and the action potentials appear to align with the | marks.



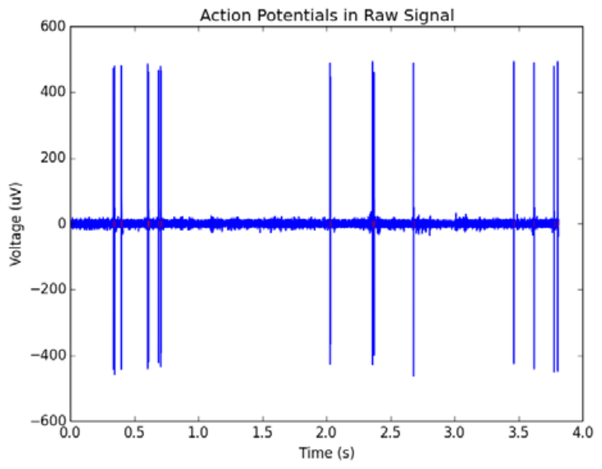
This figure would be scored as follows:
x-axis: 1 pt. The units (ms) and the labels (s) do not match.

y-axis: 2 pts

Title: 0 pts. The title is not descriptive.

Clarity: 2 pts.

Accuracy: 0 pts.
Total Score: 5 pts.



This figure would be scored as follows:

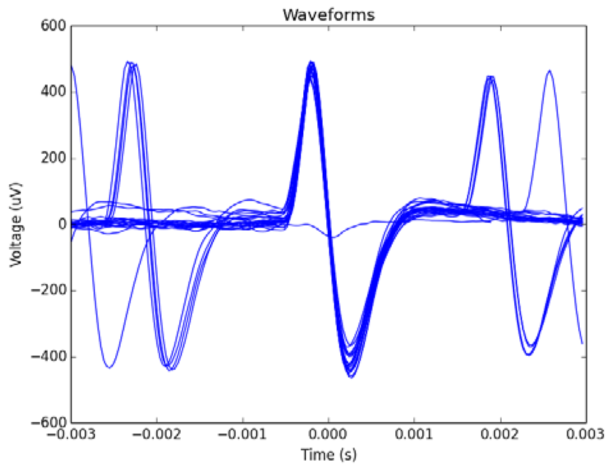
x-axis: 2 pts.

y-axis: 2 pts

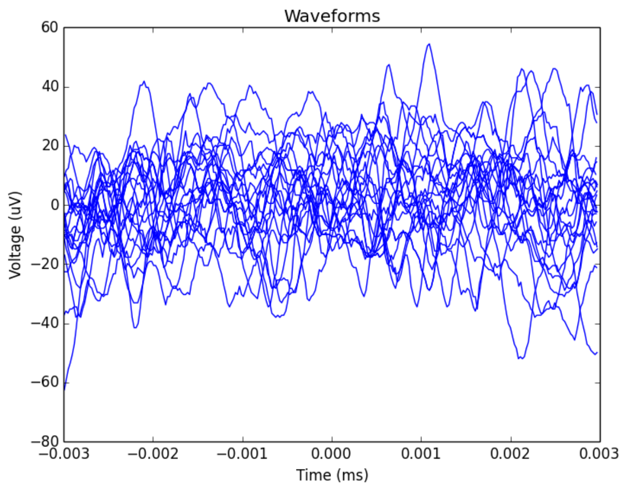
Title: 2 pts

Clarity: 1 pt. The spike times are not clear when presented in this way. (The tick marks are hard to see – they are at y = 0, but are accurate)

Accuracy: 2pts
Total Score: 9 pts.



This figure would receive full credit. The x-axis is properly labeled (i.e. the units shown match the axis label). The y-axis is similarly appropriate. The title is descriptive. The data look like one waveform, although there is an example of a false waveform in their data set, and some example of more than one waveform appearing in the time window.



This figure would be scored as follows:
x-axis: 1pt. The units (ms) and the labels (s) do not match.

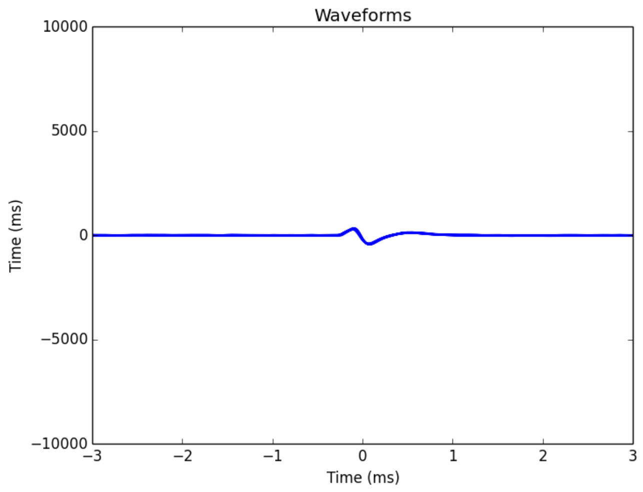
y-axis: 2 pts.

Title: 2 pts.

Clarity: 2 pts

Accuracy: 0 pts.

Total Score: 7 pts



This figure would be scored as follows:
x-axis: 2 pts

y-axis: 0 pts (Wrong label and units).

Title: 2 pts.

Clarity: 0 pts. (You can't see the waveforms in detail!)

Accuracy: 2 pts

Total Score: 6 pts