**Boston University**

**Electrical & Computer Engineering**

**EC463 Senior Design Project**

Prototype Testing Plan

**Neuron Spike Identification with Machine Learning**



Chen Yang, PhD

Yang Research Laboratory

cheyang@bu.edu

by

Team 2

Spike Sorters

Team Members

Victoria Carlsten [carlsten@bu.edu](mailto:email1@bu.edu)

Hao Chen [ha0chen@bu.edu](mailto:email1@bu.edu)

Claire Cropper [ccropper@bu.edu](mailto:email1@bu.edu)

Shi Gu [bengushi@bu.edu](mailto:email1@bu.edu)

**Required Materials:**

Hardware:

* Personal Laptop

Software:

* Python, 2 scripts
* 2 data files
  + *DataImport.py*
    - Load recording data
    - Detect spikes
  + *spikesortingVTJason.py*
    - Load recording data
    - Detect spikes
    - Delete unnecessary data

**Setup:**

The setup only consists of one part: using our personal laptops to generate three graphs via two Python scripts. Firstly, running the DataImport.py using the data file ‘30 min\_0001.abf’ will produce the graph that depicts a bandpass filter. Then, running the spikesortingVTJason.py will produce 2 graphs. One is to generate the graph of Raw, LFP and Spike data after the raw data has been processed by the bandpass filters. The second one is to generate the graph of spikes that have been deleted unnecessary data (fake oscillation spikes, 3 ms spikes).

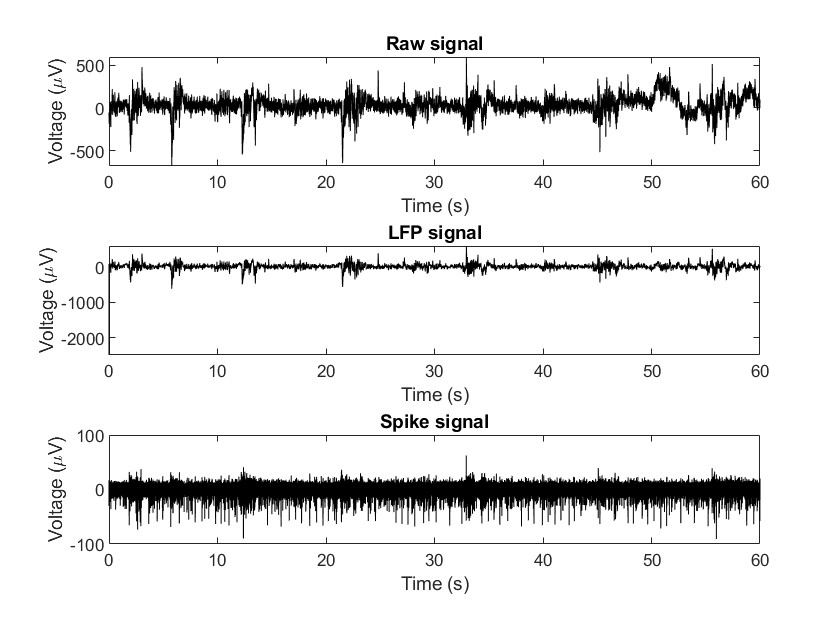
**Pre-testing Setup Procedure:**

DataImport file:

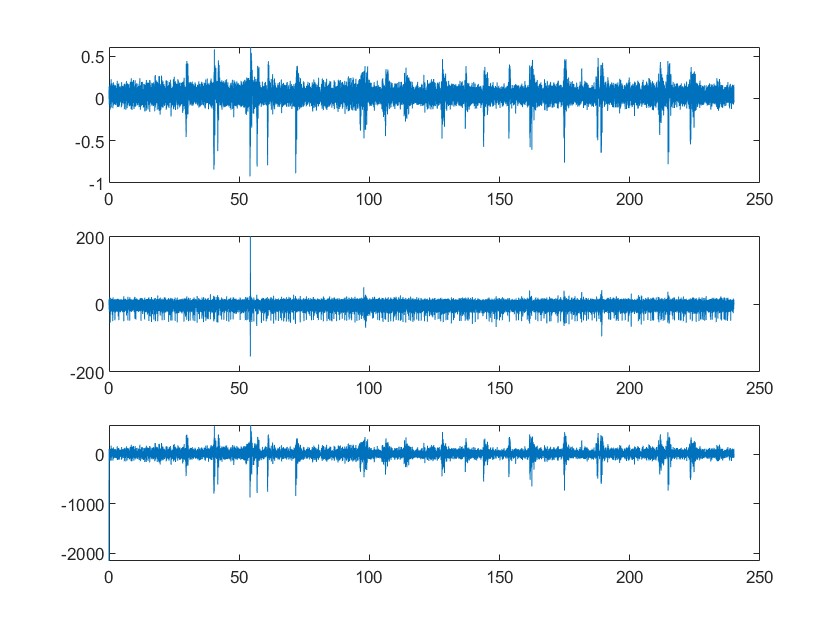
1. Make sure user is in the correct working directory
2. Make sure ‘30 min\_0001.abf’ is in the working directory
3. Run the python script, *DataImport.py*

spikesortingVTJason file:

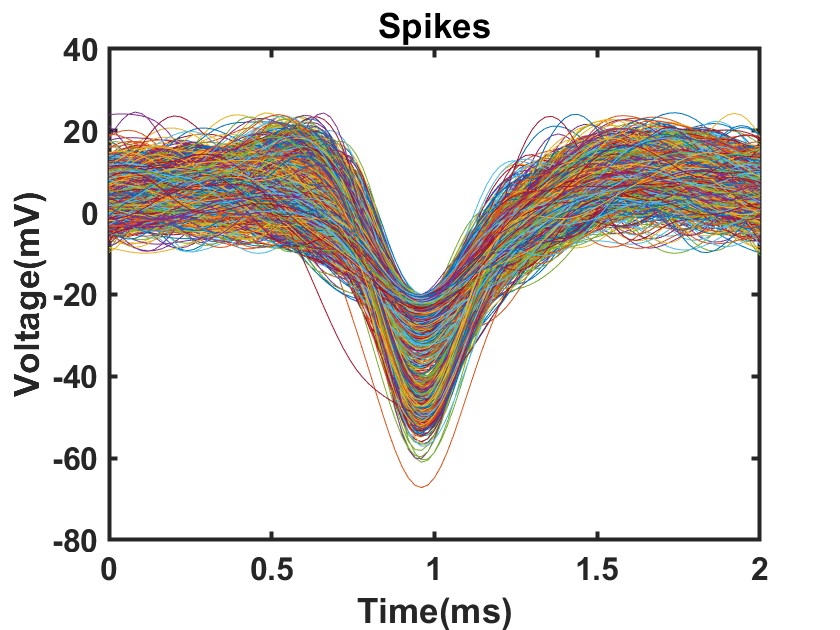
1. Make sure user is in the correct working directory
2. Make sure ‘10 min recording1.mat’ is in the working directory
3. Run lines 1 - 118 of the python script, *spikesortingVTJason.py*



*Figure 1. Bandpass filter for picking up the LFP and Spike signal in DataImport.m*

****

*Figure 2. Bandpass filter for Spikes and LFP in spikesortingVTJason.m*



*Figure 3. Spikes graph in spikesortingVTJason.m*

**Testing Procedure:**

1. Run DataImport.py
2. The computer will load a plot identical to Figure 1.
3. Run spikesortingVTJason.py
4. The computer will load a plot identical to Figures 2 and 3.

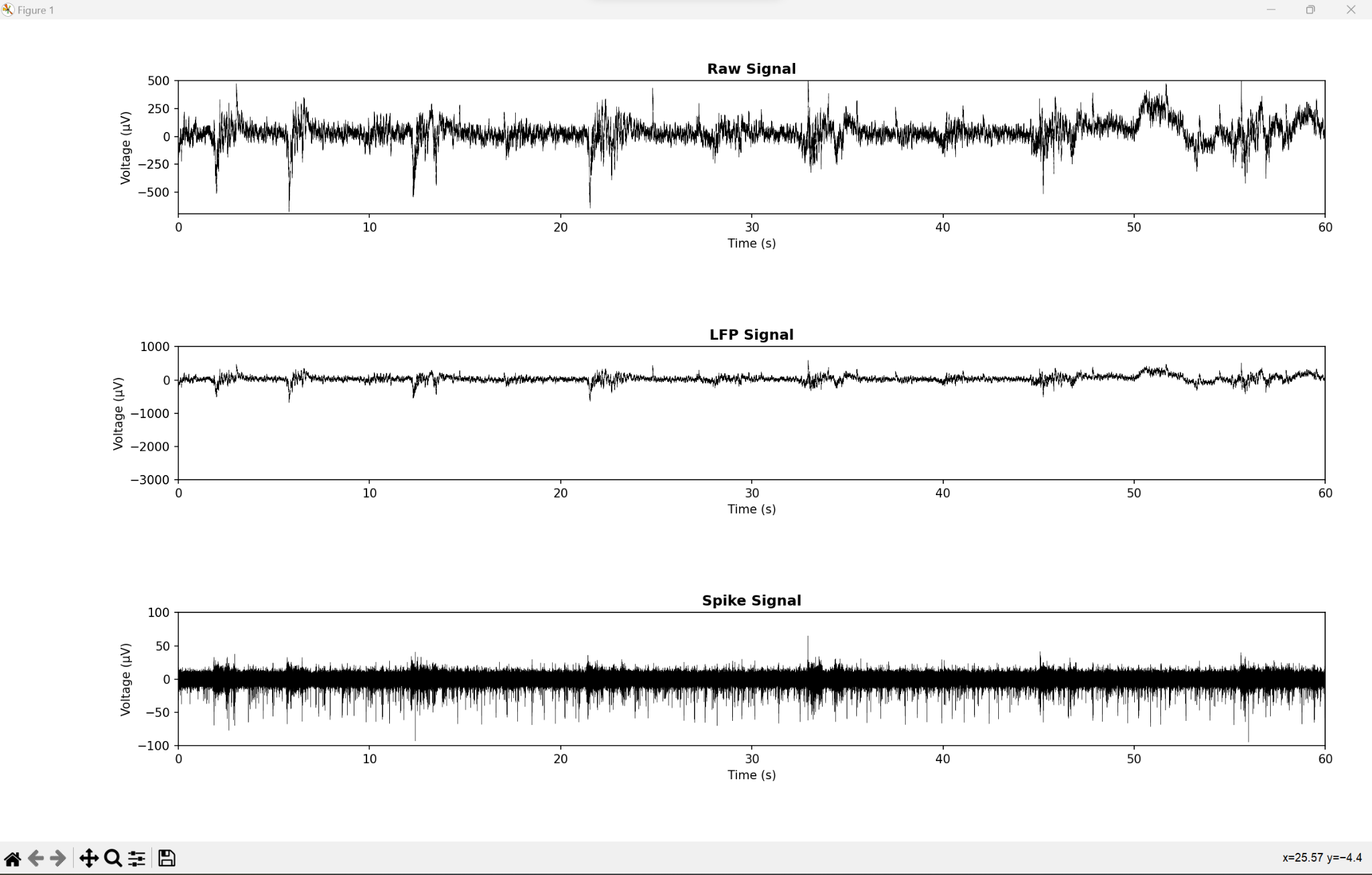
**Measurable Criteria:**

1. DataImport.py should successfully accept and load the recording data from the file ‘30 min\_0001.abf’, which is an .abf file of pre-recorded 30 minutes of Electrophysiological Data.
2. Spikes should be detected according to the threshold value and output a plot similar to Figure 1.
3. spikesortingVTJason.py (lines 1 - 118) should correctly load the recording data from the ‘10 min recording1.mat’, which is a mat file of pre-recorded 10 minutes of Electrophysiological Data, and then select the part of interest.
4. Spikes should be detected according to the threshold value and output a plot similar to Figure 2.
5. First step of PCA analysis of the spike array should output a plot similar to Figure 3.

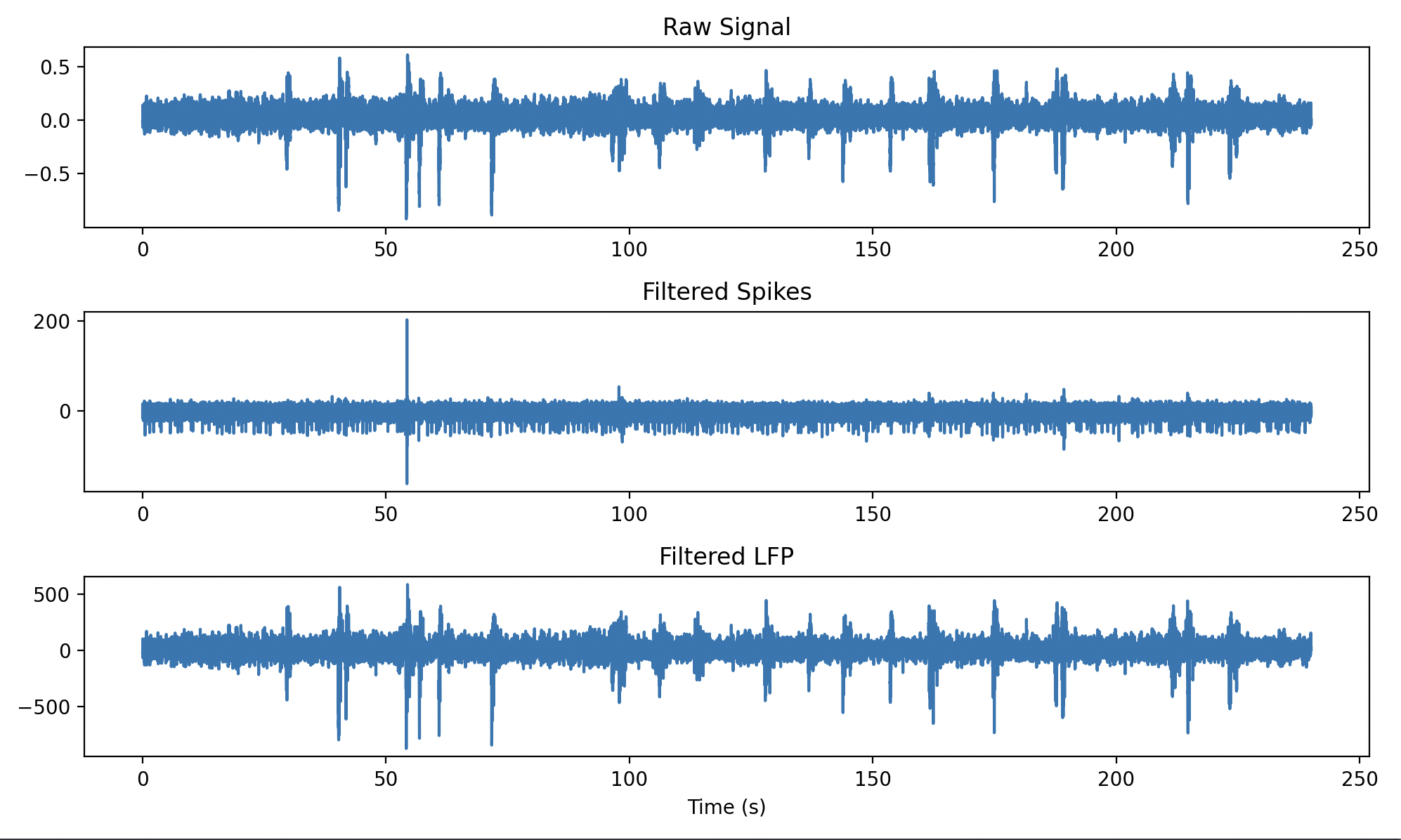
**Score Sheet**

| **Object** | **Category** | **Correct? (Y/N)** |
| --- | --- | --- |
| Bandpass filter for picking up the LFP and Spike signal in DataImport.m | Script | Y |
| Bandpass filter for Spikes and LFP in spikesortingVTJason.m | Script | Y |
| Filtered pikes graph in spikesortingVTJason.m | Script | Y |
| **Result →** Accurate Graphs | | |

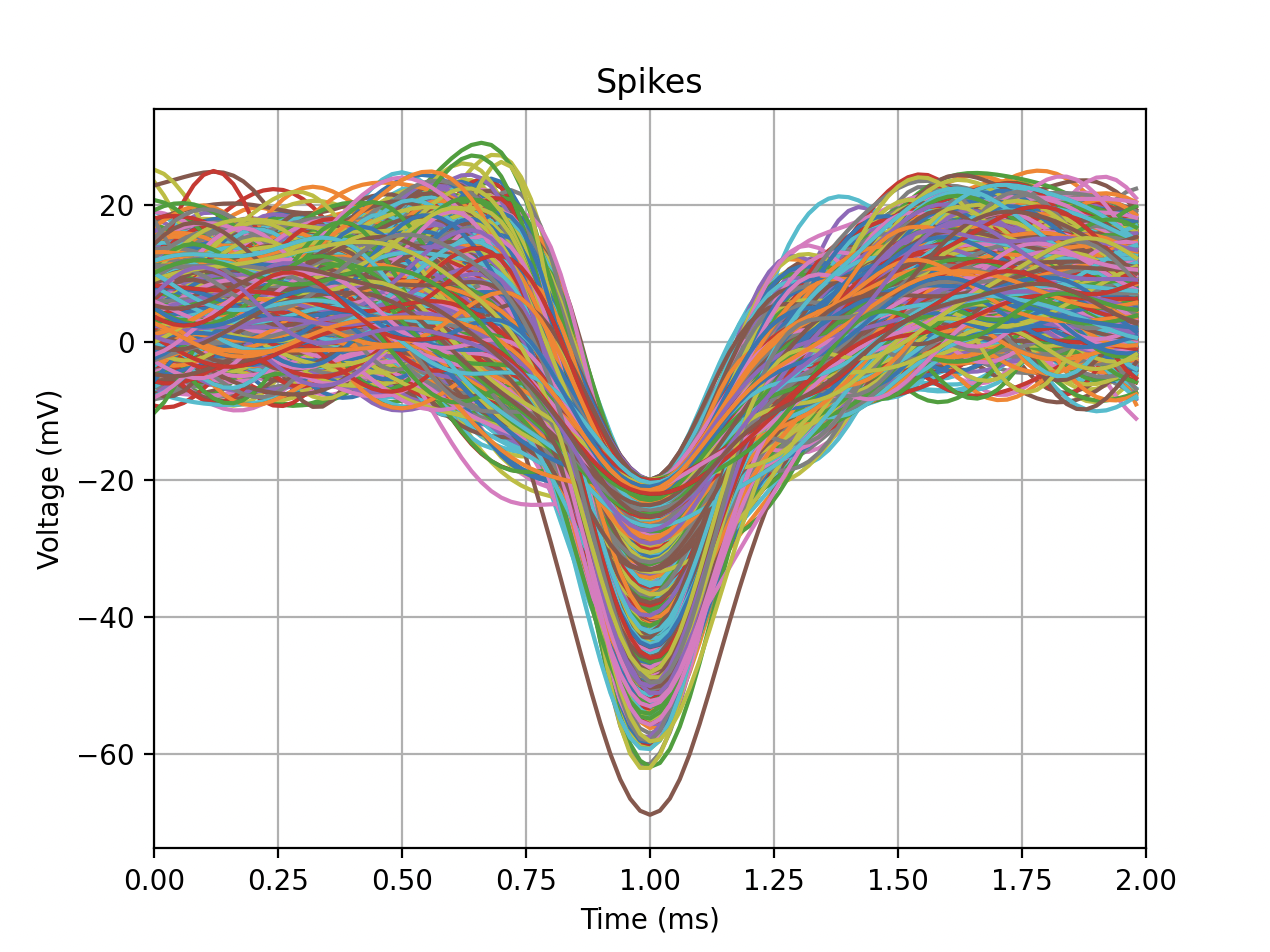
**Test Results in Detail**



*Figure 4. Bandpass filter for picking up the LFP and Spike signal in DataImport.py*

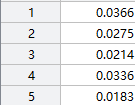


*Figure 5. Bandpass filter for Spikes and LFP in spikesortingVTJason.py*



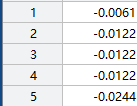
*Figure 6. Spikes graph in spikesortingVTJason.py*

We had identical looking graphs, however there were some discrepancies in the number of spikes detected in our spikesortingVTJason.py and spikesortingVTJason.m files once we printed the number of data points. We found a few hundred more spikes in our Python code than in our Matlab code, and we’re unsure as to what could be causing the difference.



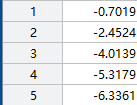


*Figure 7. The contrast between the first 5 data points of recording1 in Matlab and python*





*Figure 8. The contrast between the first 5 data points of rawsignal in Matlab and python*





*Figure 9. The contrast between the first 5 data points of spikes in Matlab and python*

*Figure 10. The contrast between the final numbers of spikes in Matlab and python*

As we can see in Figures 7 and 8, the data points are basically the same except that Python holds more decimal points compared to that in Matlab. After the raw signal has been processed by the bandpass filter, the discrepancies first appear between the spikes results shown in Figure 9. There might be 2 possible explanations for the discrepancies. The first reason could be the data points of Python holding more decimal points and thus affecting the calculations of the bandpass filter. The second reason could be that the bandpass algorithms are different between Python and Matlab. Reserving the discrepancies and doing the following process of deleting unnecessary data, we finally get the results in Figure 10. There are about 200 more data points found in the Python codes.

**Conclusion**

While our graphs are nearly identical, there are differences in the number of data points. In the future, we’re going to investigate the potential reasons for this, and determine what could be causing this. Some of the ways we’re going to try to solve this issue are creating scatter plots of both the Matlab and Python data, going through the documentation for every function used in both Matlab and Python to see if there are differences in their functions, and performing a t-test to compare the results from our Matlab vs Python code to further investigate the possibility of a significant difference between them. Our to-do list for the next week is to determine and fix the issue with our code concerning the amount of spikes detected, as well as to finish the PCA analysis portion.