BE 521: Homework 6

Spike sorting

Spring 2019

59 points

Due: Thursday, 03/5/2020 11:59pm

Objective: Detect and cluster spikes

John Talley
Collaborators: Joseph Iwasyk

Overview

In this homework, you will do some basic spike sorting using two different datasets. The first (`I521_A0006_D001`) is from a crayfish neuromuscular junction, a good model for human central nervous system synapses[1]. Specifically, the data contains two simultaneous recordings: an extracellular recording from the third nerve (channel `nerve`) of a crayfish abdominal ganglion, which contains six spontaneously active motor neurons, and an intracellular recording from the superficial flexor muscle (channel `muscle`) innervated by this nerve. You will attempt to discern relationships between the classes of spike waveforms you extract from the motor nerve trace and elicited potentials seen in the muscle fiber recording. Then, you will revisit a human intracranial EEG recording (`I521_A0006_D002`) and use some of the techniques you've learned in class to build a more automated spike sorter. Note: While spikes may have positive and negative deflections, we will only focus on positive spikes on this homework for simplicity.

1 Spike Detection and Clustering (38 pts)

In this section, you will explore some basic filtering and spike thresholding to ultimately compare spike clusters you pick out by eye to those selected by an automated algorithm.

1. You can assume that the nerve samples have already been low-pass filtered. Here you will high-pass filter in order to remove signals like slow local field potentials and 60 Hz power line noise. Create a 4th order *elliptic filter* with 0.1 dB of ripple in the passband, a stopband 40 dB lower than the peak value in the passband, and a passband edge frequency of 300 Hz (see Matlab's `ellip` function and make sure your give the edge frequency in the correct normalized form). The statement to create this filter (defined by the filter coefficients `b` and `a`) should look something like

```
[b,a]=ellip(n,Rp,Rs,Wp,'high')
```

   Clearly specify the denominator and numerator coefficients obtained for your filter function. (2pts)

[1] The sampling rate of this data is 2000 Hz, which is adequate for this homework's instructional purposes but usually inadequate for real spike sorting, which often uses sampling frequencies on the order of 20 kHz.

```
dataset_ID = 'I521_A0006_D001';
ieeg_id = 'jtalley';
ieeg_pw = 'jta_ieeglogin.bin';

session = IEEGSession(dataset_ID,ieeg_id,ieeg_pw);

Fs  = session.data.sampleRate;
durationInUSec1 = session.data.rawChannels(1).get_tsdetails.getDuration;
durationInSec1 = durationInUSec1/1e6;
duration1 = seconds(durationInSec1);
durationInUSec2 = session.data.rawChannels(2).get_tsdetails.getDuration;
durationInSec2 = durationInUSec2/1e6;
duration2 = seconds(durationInSec2);

muscle = getvalues(session.data,1:(durationInSec1*Fs + 1),1);
nerve = getvalues(session.data,1:(durationInSec2*Fs + 1),2);

n = 4;
Rp = 0.1;
Rs = 40;
Wp = (300/Fs)*2;

[b,a] = ellip(n,Rp,Rs,Wp,'high');
```

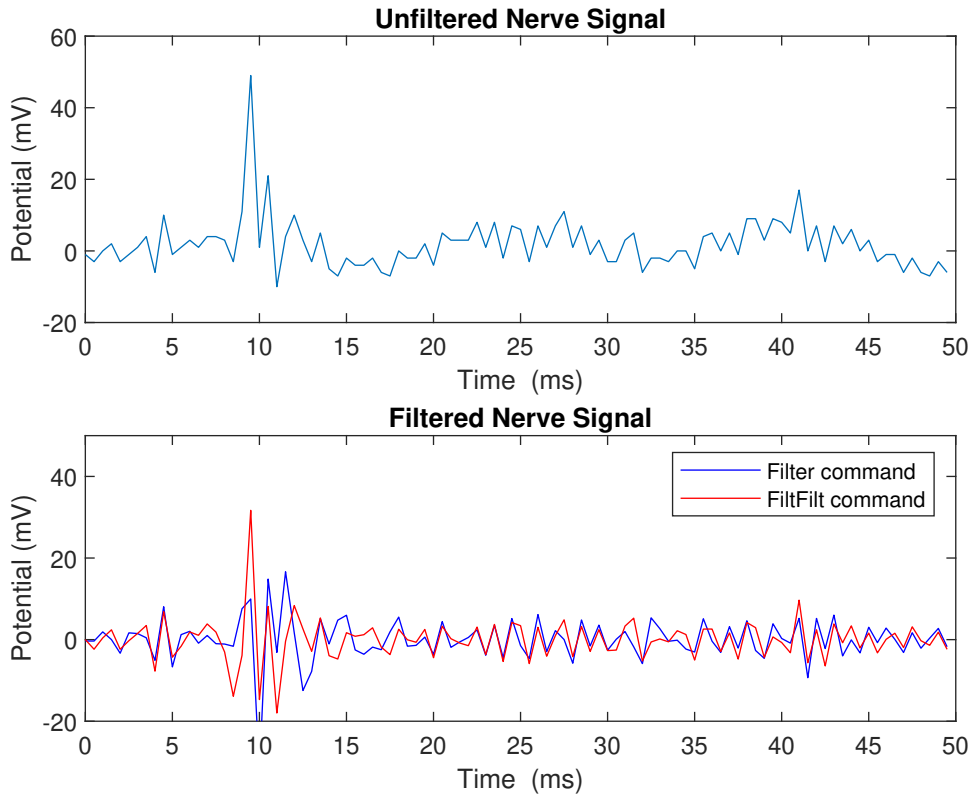
```
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesDetails class
exist — not clearing java
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesInterface class
exist — not clearing java
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: jtalley
Client password: ****
```

2. Using the `filter` function and `filtfilt` function, obtain two different filtered outputs of the nerve signal.

   (a) In a 2x1 subplot, plot the first 50 ms of the unfiltered nerve signal in the top subplot; in the bottom subplot, plot the `filter` output in blue and the `filtfilt` output in red. Use a potential range (y-axis) of -20 to 50 millivolts. (4 pts)

```
t = 1/Fs:1/Fs:durationInSec1+1/Fs;
t_ms = 1/Fs:1000/Fs:1000*durationInSec1;
t_50 = t_ms(1:(0.050*Fs));
nerve_mV = nerve/1000; % convert uV to mV
nerve_raw = nerve_mV(1:(0.050*Fs)); % first 50 ms

nerve_filter = filter(b,a,nerve_mV);
nerve_ff = filtfilt(b,a,nerve_mV);

figure();

subplot(2,1,1);
plot(t_50,nerve_raw);
xlabel('Time  (ms)');
ylabel('Potential (mV)');
title('Unfiltered Nerve Signal');

subplot(2,1,2);
plot(t_50,nerve_filter(1:(0.050*Fs)),'b');
```

```
hold on;
plot(t_50,nerve_ff(1:(0.050*Fs)),'r');
ylim([-20 50]); % -20 to 50 mV, originally in microvolts
xlabel('Time  (ms)');
ylabel('Potential (mV)');
title('Filtered Nerve Signal');
legend('Filter command','FiltFilt command');
```



(b) How is the unfiltered signal different from the filtered signal? What is different about the two filtered (red and blue) signals? (2 pts)

In comparing the raw, unfiltered nerve signal to the two filtered signal traces, we see that low freqeuncy noise components are removed. That is, slow oscillations in the data recording resulting from artifacts were removed, which removes the "humps" or "hills" observed around 20-30 ms 35-45 ms in the raw nerve signal. In comparing the two filtering methods, the red "filtfilt" processed signal matches the spike shape and temporal location of the original signal much more closely than the MATLAB "filter" function processed signal. There appears to be a delay of a couple ms in the spike events in the blue trace, and the trace dips below the -20 mV lower bound of the prescribed potential range. The background traces appear similar in the two filtering methods when spike events are not occuring, but the spike event in the red filtfilt signal more closely matches the temporal location and amplitude proportions seen in the original signal.

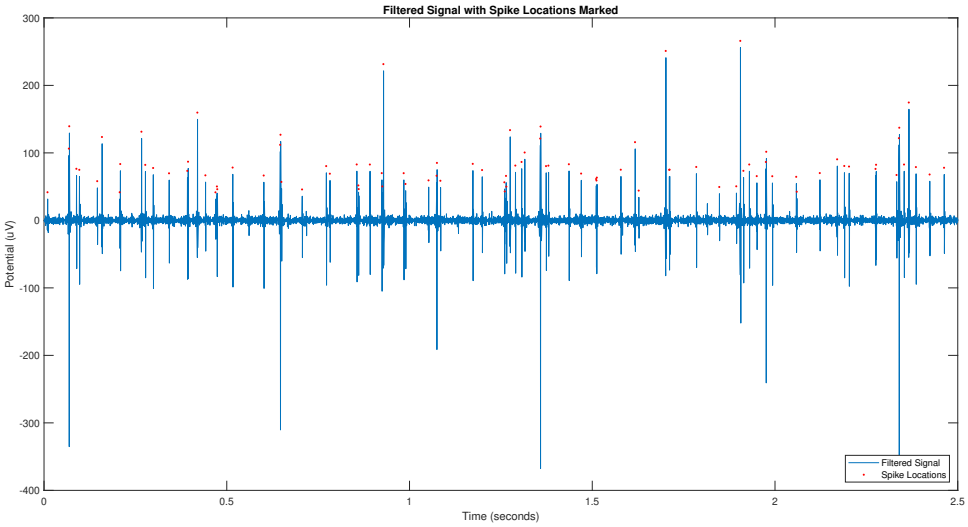
(c) Briefly explain the mathematical difference between the two filtering methods, and why one method might be more advantageous than the other in the context of spike detection? (5 pts)

Applying MATLAB's "filter" method introduces a time delay (phase shift) in the resulting signal. Thus, in the blue trace, both the pre-spike dip in potential and the two potential spikes observed occur about 2 ms later than in the raw signal. In the "filtfilt" method, the filter is applied in both forward and backward directions to result in a zero-phase shift in the resulting output signal, which is why we call this "zero-phase filtering". This is the advantage of using the "filtfilt" method over

the "filter" command. While "filter" only applies the transfer function with coefficients a b in the forward direction temporally, "filtfilt" applies the function in both directions to yield a zero-phase shifted output signal. The benefit here is that the spike events are accurate in the timespace in which they occur with the filtfilt method, and there is no time-delay introduced in the signal. Having an accurate understanding of the temporal characteristics of a spike or peak in potential is important in spike sorting to understanding how a nerve responds to stimuli introduced at spike timepoints. When different types of cells may have spikes of different amplitudes, response times, transients, and refractory periods, it is essential to use a zero-phase filtering method. Thus, we reason that the "filtfilt" command is better than "filter" in MATLAB for spike-sorting applications.

3. Using a spike threshold of +30 mV, calculate the index and value of the peak voltage for each spike in the **filtered** nerve signal (select the best one). Use these values to plot the first 2.5 seconds of the nerve signal with a red dot above (e.g. 10 mV above) each spike. (Hint: Plot the entire length of the nerve signal with all the spikes marked but then restrict the x-axis using `xlim` to [0, 2.5] seconds) (4 pts)

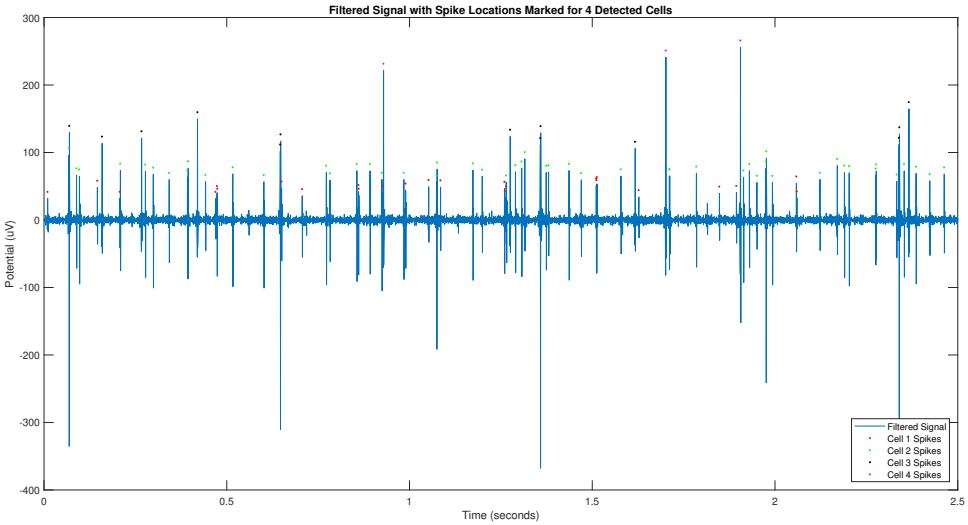
```
[pks, locs] = findpeaks(nerve_ff, Fs, 'MinPeakHeight', 30);
% dot_height = zeros(length(locs),1);
% for i = 1:length(locs)
%     dot_height(i) = pks(i) + 10000; % 10 mV above spike
% end

dot_height =  pks + 10;

figure('WindowState','Maximized');
plot(t,nerve_ff);
hold on;
scatter(locs,dot_height,'.r');
xlim([0 2.5]);
xlabel('Time (seconds)');
ylabel('Potential (uV)');
title('Filtered Signal with Spike Locations Marked');
legend('Filtered Signal','Spike Locations','Location','southeast');
```



4. Under the assumption that different cells produce different action potentials with distinct peak amplitudes, decide how many cells you think were recorded (some number between 1 and 6). You may find it helpful to zoom in and pan on the plot you made in question 1.3. You may also find it useful

to plot the sorted peak values to gain insight into where "plateaus" might be. (No need to include these preliminary plots in the report, though.) Use thresholds (which you well set manually/by eye) to separate the different spikes. Make a plot of the first 2.5 seconds similar to that in 1.3 except now color the spike dots of each group a different color (e.g., `'r.'`,`'g.'`,`'k.'`,`'m.'`).(6 pts)

```
% Thresholds determined via mnaual visual inspection of above plot
sort(pks); % to look at various thresholds
[pks1, locs1] = findpeaks(nerve_ff, Fs, 'MinPeakHeight', 30);
[pks2, locs2] = findpeaks(nerve_ff, Fs, 'MinPeakHeight', 55);
[pks3, locs3] = findpeaks(nerve_ff, Fs, 'MinPeakHeight', 100);
[pks4, locs4] = findpeaks(nerve_ff, Fs, 'MinPeakHeight', 200);

% Remove repeats
index1 = find(pks1 > 55);
pks1(index1) = [];
locs1(index1) = [];

index2 = find(pks2 > 100);
pks2(index2) = [];
locs2(index2) = [];

index3 = find(pks3 > 200);
pks3(index3) = [];
locs3(index3) = [];

% No repeats for cell 4 (highest threshold)


dot_height1 =  pks1 + 10;
dot_height2 =  pks2 + 10;
dot_height3 =  pks3 + 10;
dot_height4 =  pks4 + 10;

figure('WindowState','Maximized');
plot(t,nerve_ff);
hold on;
plot(locs1,dot_height1,'.r');
 hold on;
plot(locs2,dot_height2,'.g');
hold on;
plot(locs3,dot_height3,'.k');
hold on;
plot(locs4,dot_height4,'.m');
xlim([0 2.5]);
xlabel('Time (seconds)');
ylabel('Potential (uV)');
title('Filtered Signal with Spike Locations Marked for 4 Detected Cells');
legend('Filtered Signal','Cell 1 Spikes','Cell 2 Spikes','Cell 3 Spikes','Cell 4 Spikes','Location','southea
```



5. Use Matlab's k-means[2] function (`kmeans`) to fit k clusters (where k is the number of cells you think the recording is picking up) to the 1D data for each spike.

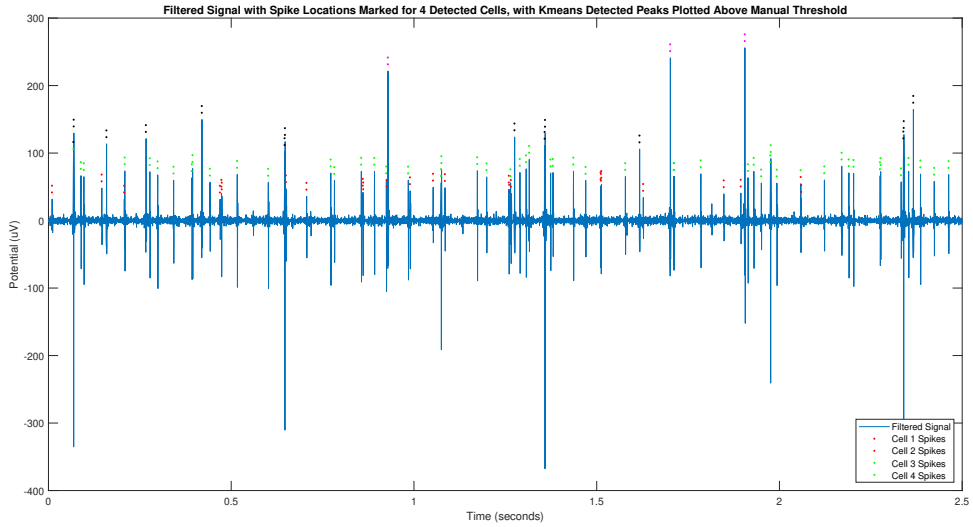
   (a) Using the same color order (for increasing spike amplitude) as you did for the thresholds in question 1.4, plot the spike cluster colors as little dots slightly above those you made for question 1.4. The final figure should be a new plot of the nerve voltage and two dots above each spike, the first being your manual label and the second your clustered label, which (hopefully/usually) should be the same color. (4 pts)

```
rng(1); % for reproducibility
k = 4;
[idx, C] = kmeans(pks, k); % Run kmeans clustering algorithm on all peaks

% Centroid values from kmeans for each cell type
C(1) % Cell 2 Centroid = 66.2488 (manual threshold was 30 mV)
C(2) % Cell 4 Centroid = 234.8149 (manual threshold was 55 mV)
C(3) % Cell 1 Centroid = 41.0998 (manual threshold was 100 mV)
C(4) % Cell 3 Centroid = 121.8080 (manual threshold was 200 mV)

% index by cell type from kmeans clusters

idx2 = find(idx == 1);
klocs2 = locs(idx2);
kpks2 = pks(idx2);
kdot2 = kpks2 + 20; % 20 mV above

idx4 = find(idx == 2);
klocs4 = locs(idx4);
kpks4 = pks(idx4);
kdot4 = kpks4 + 20;

idx1 = find(idx == 3);
klocs1 = locs(idx1);
kpks1 = pks(idx1);
kdot1 = kpks1 + 20;

idx3 = find(idx == 4);
klocs3 = locs(idx3);
kpks3 = pks(idx3);
```

[2]Clustering, like k-means you are using here, is a form of unsupervised learning.

```
kdot3 = kpks3 + 20;

figure('WindowState', 'Maximized');
plot(t,nerve_ff);
hold on;
plot(locs1,dot_height1,'.r');
plot(klocs1,kdot1,'.r');
hold on;
plot(locs2,dot_height2,'.g');
plot(klocs2,kdot2,'.g');
hold on;
plot(locs3,dot_height3,'.k');
plot(klocs3,kdot3,'.k');
hold on;
plot(locs4,dot_height4,'.m');
plot(klocs4,kdot4,'.m');
xlim([0 2.5]);
xlabel('Time (seconds)');
ylabel('Potential (uV)');
title('Filtered Signal with Spike Locations Marked for 4 Detected Cells, with Kmeans Detected Peaks Plo
legend('Filtered Signal','Cell 1 Spikes','Cell 2 Spikes','Cell 3 Spikes','Cell 4 Spikes','Location','so
```

```
ans =

   66.2488


ans =

  234.8149


ans =

   41.0998


ans =

  121.8080
```



(b) Which labeling, your manual ones or the ones learned by clustering) seem best, or do they both

seem just as good? (Again, panning over the entire plot may be helpful.) (2 pts)

Both kmeans and the manual thresholding used to label the spikes in this signal seem equally good. Using threshold manual threshold values of 30 mV, 55 mV, 100 mV, and 200 mV for the spikes of cell types 1-4 yielded, detection of 42, 83, 22, and 8 spikes for each cell, respectively. When kmeans clustering was run on the set of all peaks above 30 mV in the signal, 40, 83, 24, and 8 spikes were found for each cell type 1-4. When inspecting the plot above, we see this to be true in the consistency of colored dot labels above each detected spike, with kmeans detections plotted in the same color in nearly every instance above the spike detected via manual thresholding, indicating that the two methods were equally good at clustering the spikes by cell type.

6. In this question, you will test the hypothesis that the muscle potential responses are really only due to spikes from a subset of the cells you have identified in the previous two questions. First, plot the first 2.5 seconds of the muscle fiber potential and compare it with that of the nerve. Observe the relationship between spikes and the muscle fiber response. (No need to include this plot and observation in your report.) Now, calculate the maximum muscle fiber potential change[3] in the 25 ms[4] window after each spike (with the assumption that spikes without any/much effect on the muscle fiber potential do not directly innervate it).

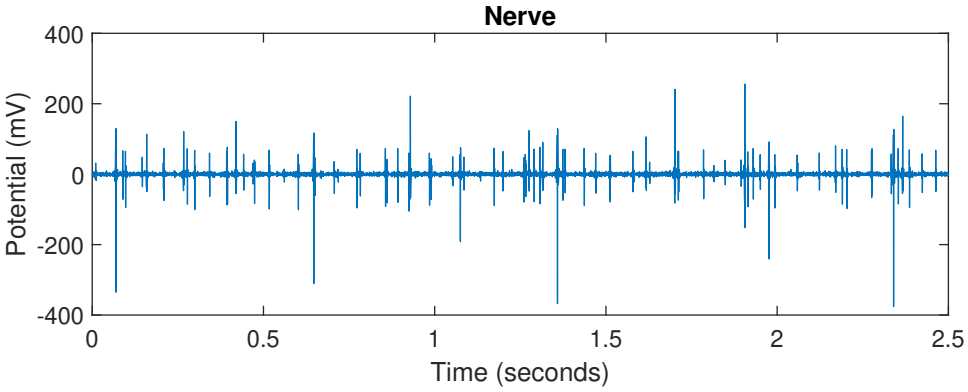
   (a) Using the cell groups you either manually defined or found via k-means clustering (just specify which you're using) again with different colors, plot a colored point for each spike where the x-value is the spike amplitude and the y-value is the muscle potential change. (6 pts)

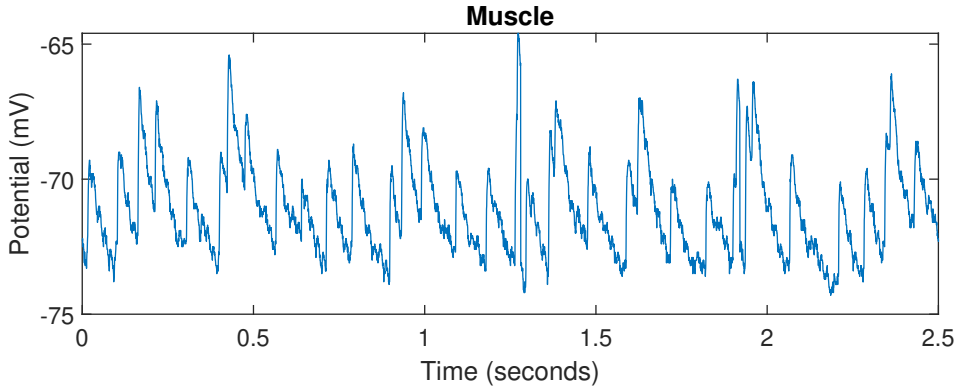
```
muscle_mV = muscle/1000; % Convert uV to mV

figure();

subplot(2,1,1);
plot(t,nerve_ff);
xlim([0 2.5]);
xlabel('Time (seconds)');
ylabel('Potential (mV)');
title('Nerve');

subplot(2,1,2);
plot(t,muscle_mV);
xlim([0 2.5]);
xlabel('Time (seconds)');
ylabel('Potential (mV)');
title('Muscle');

% Calculate maximum muscle fiber potential change in 25 ms window after
% each spike

muscle_response = zeros(length(locs),1);
for i = 1:length(locs)
    muscle_response(i) = range(muscle_mV(locs(i)*Fs:((locs(i)+0.025)*Fs)));
end

% Using kmeans idx to relate to each cell spike

muscle_response1 = muscle_response(idx1);
muscle_response2 = muscle_response(idx2);
muscle_response3 = muscle_response(idx3);
muscle_response4 = muscle_response(idx4);
```

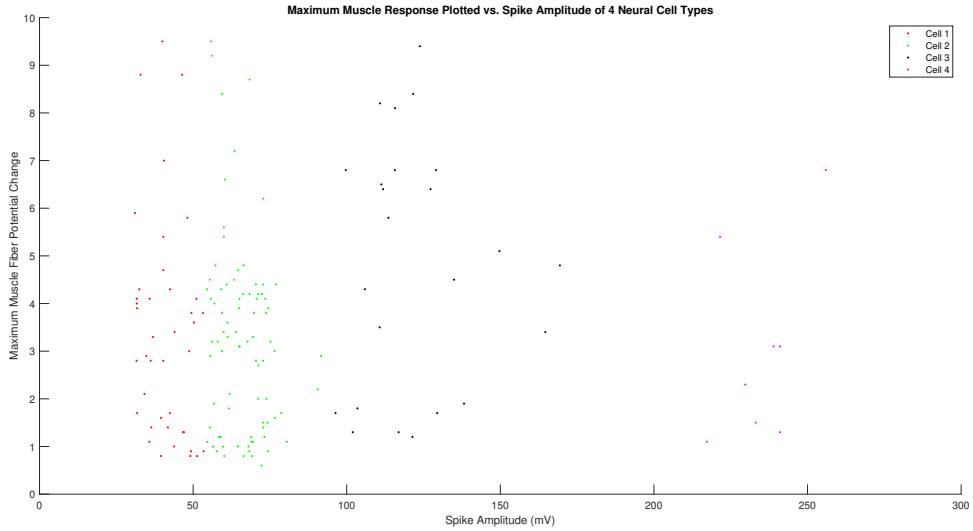
[3] max voltage - min voltage

[4] Note that this 25 ms window is somewhat ad hoc and is just what seems reasonable by eye for this data. It implies no underlying physiological time scale or standard.

```
% Plot spike amplitude fromkmeans on x, muscle response on y

figure('WindowState','Maximized');
scatter(kpks1, muscle_response1, '.r');
hold on;
scatter(kpks2, muscle_response2, '.g');
hold on;
scatter(kpks3, muscle_response3, '.k');
hold on;
scatter(kpks4, muscle_response4, '.m');
xlabel('Spike Amplitude (mV)');
ylabel('Maximum Muscle Fiber Potential Change');
title('Maximum Muscle Response Plotted vs. Spike Amplitude of 4 Neural Cell Types');
legend('Cell 1','Cell 2','Cell 3','Cell 4');
```







(b) Does this plot support the hypothesis that the muscle fiber responses are only due to a subset of the cells. Explain why or why not. (3 pts)

This plot does not support this hypothesis. All four cell types seem to yield muscle potential responses across the entire range of muscle fiber response potential changes from -9 mV (aside from cluster 4, but there are only 8 samples and they range from 1-7 mV in muscle responses). If some muscle fiber responses were due to only a subse of the cells, we would expect to see correlations where the cells clustered based on their spike amplitudes would yield muscle fiber potential changes clustered within a certain range. No such correlation seems to exist here, as all 4 cell clusters result in a wide range of muscular responses.

2 Multivariate Clustering (21 pts)

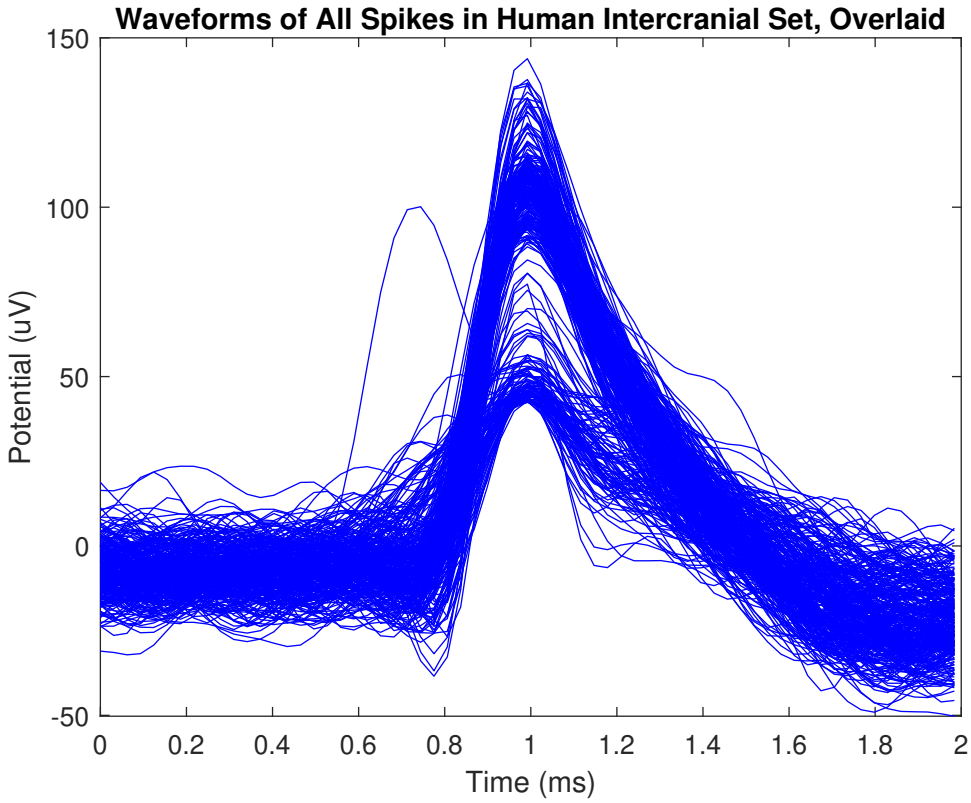
In this section, you will explore similar methods for spikes sorting and clustering but with a different dataset, the human intracranial data in `I521_A0006_D002`, which is a larger dataset of the same recording you saw in `I521_A0001_D001` of Homework 1.

1. Using a threshold six standard deviations above the mean of the signal, detect the spikes in the signal. In addition, extract the waveform from 1 ms before the peak to 1 ms after it with peak value in the middle. (You will end up with a matrix where each row corresponds to the number of data points in 2 ms of signal minus 1 data point. Use the closest integer number of data points for the $\pm$ 1 ms window.)

   (a) Plot the waveforms of all the spikes overlaid on each other in the same color. (4 pts)

```
dataset_ID = 'I521_A0006_D002';
ieeg_id = 'jtalley';
ieeg_pw = 'jta_ieeglogin.bin';

session2 = IEEGSession(dataset_ID,ieeg_id,ieeg_pw);

fs  = session2.data.sampleRate;
durInUSec = session2.data.rawChannels(1).get_tsdetails.getDuration;
durInSec = durInUSec/1e6;
dur = seconds(durInSec);

data = getvalues(session2.data,1:durInSec*fs+1,1);
threshold = mean(data) + 6*std(data);
```

```
[peaks,locations] = findpeaks(data, 'MinPeakHeight', threshold);

spikes = zeros(length(locations), ceil(0.002*fs));
for i = 1:length(locations)
    spikes(i,:) = data((locations(i)-floor(0.001*fs):(locations(i)+floor(0.001*fs))));
end

tspan = 1/fs:1000/fs:2;
figure;
for j = 1:length(locations)
    plot(tspan, spikes(j,:),'b');
    hold on;
end
xlabel('Time (ms)');
ylabel('Potential (uV)');
title('Waveforms of All Spikes in Human Intercranial Set, Overlaid');
```



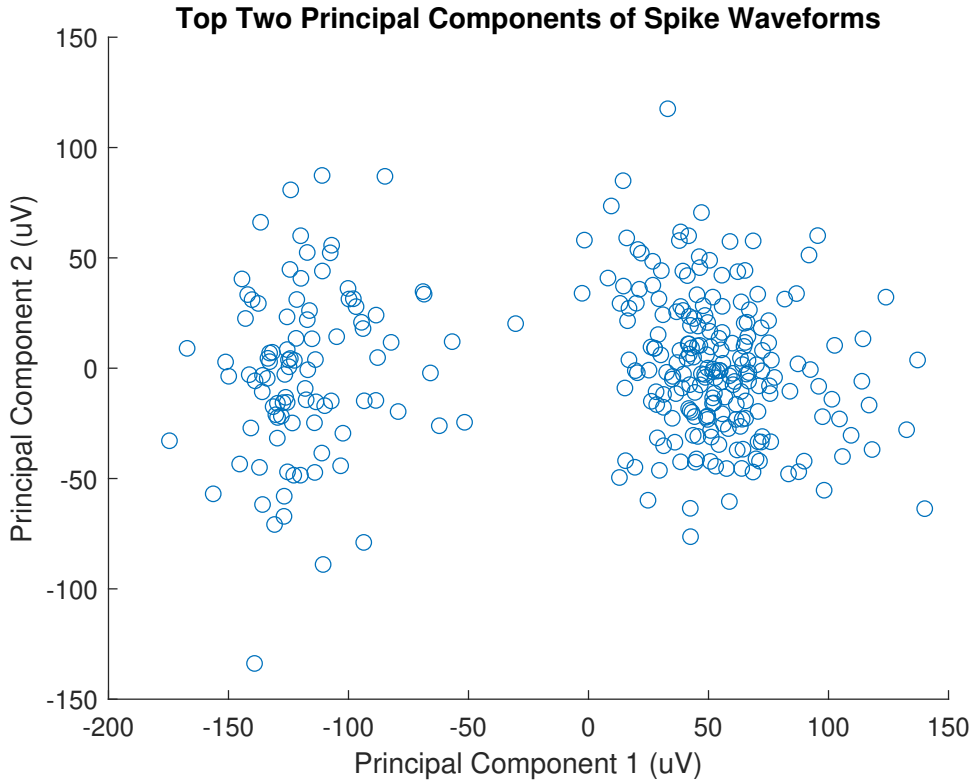
(b) Does it looks like there is more than one type of spike? (1 pt)

Yes, it looks like there is more than one type of spike. It looks like there are at least two clusters of spikes, one with peaks clustered around 50 mV and a second group with peaks clustered around 105 mV.

2. For each spike, represent the waveform by its principal components. Use the `pca` command in Matlab. Intuitively, principal component analysis finds the coordinate system that most reduces the variability in your data.

   (a) Run principal component analysis on all the spike waveforms and represent your data with the top two principal components. Make a scatterplot of your data in this principal component (PC) space. (3 pts)

```
figure;
[coeff,score,latent,tsquared,explained,mu] = pca(spikes);
scatter(score(:,1),score(:,2));
xlabel('Principal Component 1 (uV)');
ylabel('Principal Component 2 (uV)');
title('Top Two Principal Components of Spike Waveforms');
```

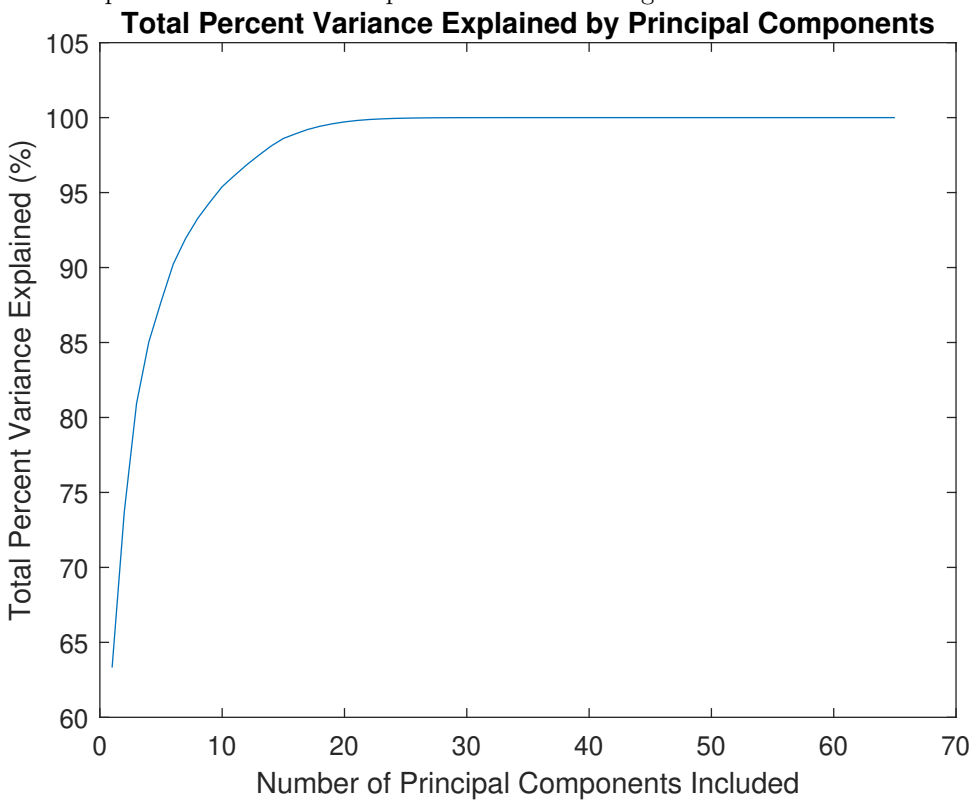


(b) Each PC also has an associated eigenvalue, representing the amount of variance explained by that PC. This an output of the `PCA` command. Plot the principal component vs the total (cumulative) percent variance explained. What is the percent variance explained if you include the top two principal components? (3 pts)

```
cum_var = zeros(length(explained),1);
for i = 1:length(explained)
    cum_var(i) = sum(explained(1:i));
end

figure;
plot(cum_var);
xlabel('Number of Principal Components Included');
ylabel('Total Percent Variance Explained (%)');
ylim([60 105]);
title('Total Percent Variance Explained by Principal Components');

% Including only top two principal components
cum_var(2)

% 73.7349% of variance is explained when including first two PC's
```

```
ans =

   73.7349
```

73.7349 percent of variance is explained when including first two PC's



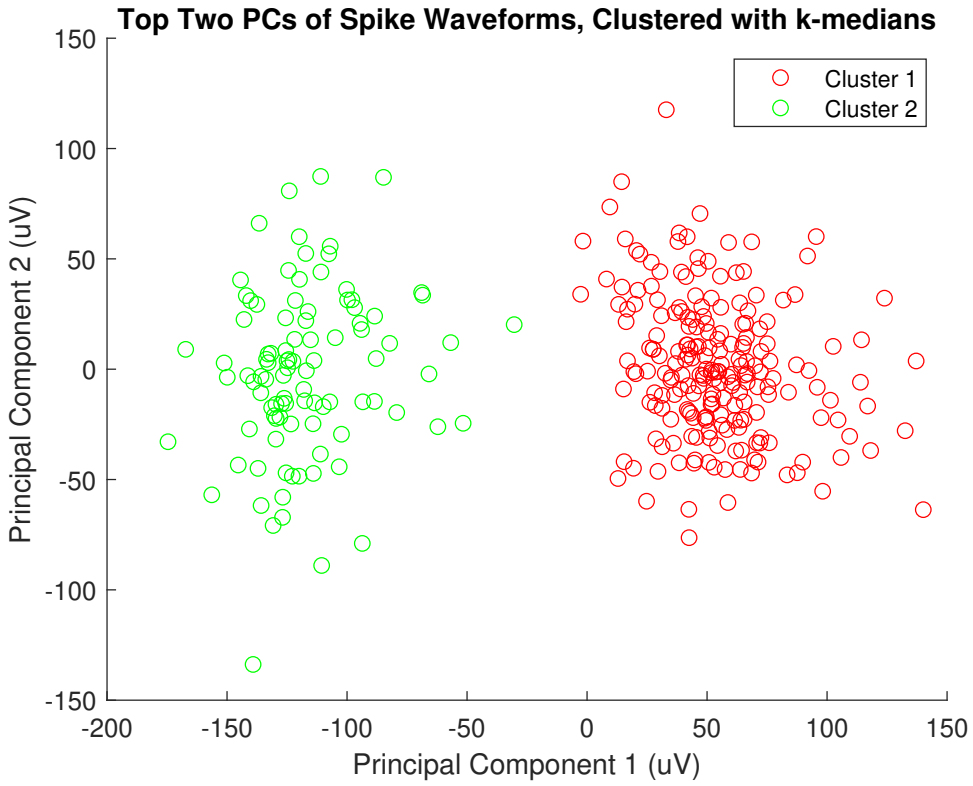
(c) Does it look like there is more than one cluster of spikes? (1 pt)

Yes, it looks like there are at least two clusters of spikes based on the two plots above. In plotting the top two principal components of the spike waveforms, there are two clear clusters within the space of the first two principal components. Additionally, only 74 percent of the total variance is explained by inclusion of the first two principal components, and only 63 percent is explained by the first PC, so it seems like there are at least two clusters of spikes.

3. Use the same `kmeans` function as you used before to cluster the spikes based on these two (normalized) features (the waveforms represented by the top two PCs). You will use a slight twist, though, in that you will perform k-medians (which uses the medians instead of the mean for the cluster centers) by using the `'cityblock'` distance metric (instead of the default `'sqEuclidean'` distance). Make a plot similar to that in 2.2.a but now coloring the two clusters red and green. (3 pts)

```
k = 2; % Two clusters of spikes
[indices,c] = kmeans(spikes,k,'Distance','cityblock');
indices1 = find(indices == 1);
indices2 = find(indices == 2);

score1 = score(:,1);
score2 = score(:,2);

cluster1_1 = score1(indices1);
cluster1_2 = score2(indices1);
```

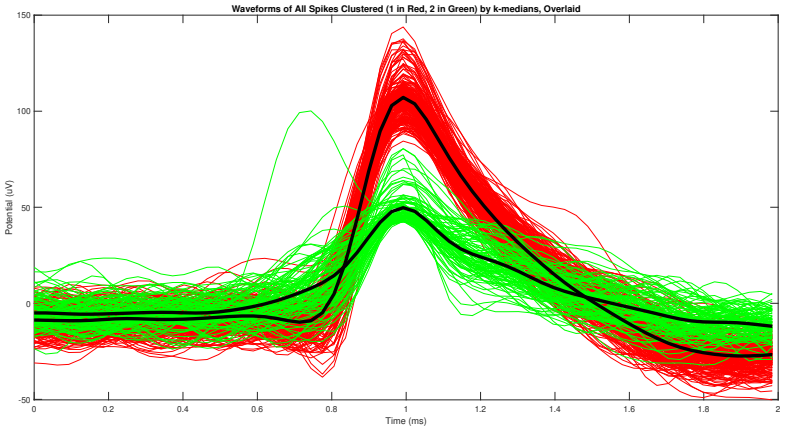
```
cluster2_1 = score1(indices2);
cluster2_2 = score2(indices2);

figure;
scatter(cluster1_1, cluster1_2,'r');
hold on;
scatter(cluster2_1, cluster2_2,'g');
xlabel('Principal Component 1 (uV)');
ylabel('Principal Component 2 (uV)')
legend('Cluster 1', 'Cluster 2');
title('Top Two PCs of Spike Waveforms, Clustered with k-medians');
```



4. Make a plot similar to 2.1 but now coloring the traces red and green according to which cluster they are in. Overlay the mean of the waveforms in each cluster with a thick black line (use the parameter `'LineWidth'` and value `'4'`). (3 pts)

```
spikes1 = spikes(indices1,:);
spikes2 = spikes(indices2,:);

spikes1_mean = zeros(1,65);
for i = 1:65
    spikes1_mean(i) = mean(spikes1(:,i));
end

spikes2_mean = zeros(1,65);
for j = 1:65
    spikes2_mean(j) = mean(spikes2(:,j));
end

figure('windowState','Maximized');
```

```
for i = 1:length(indices1)
    plot(tspan, spikes1(i,:),'r');
    hold on;
end

for j = 1:length(indices2)
    plot(tspan, spikes2(j,:),'g');
    hold on;
end

plot(tspan, spikes1_mean, 'k', 'LineWidth', 4);
plot(tspan, spikes2_mean, 'k', 'LineWidth', 4);

xlabel('Time (ms)');
ylabel('Potential (uV)');
title('Waveforms of All Spikes Clustered (1 in Red, 2 in Green) by k-medians, Overlaid');
```



5. What are some dangers of using the clustering techniques in this homework? (List 3) (3 pts)

   Some dangers of using the clustering techniques in this homework are: 1. k-means creates cluster of circular, uniform cluster sizes with strict thresholds 2. In PCA, it is often difficult to determine what the PC's actually mean from the data, which makes it difficult to choose the correct number of PC's to include. 3. Both PCA and k-means (and when used in conjunction) run the risk of overfitting if an incorrect cluster size is chosen.