BIOMEDE 517 - Neural Engineering - Labs 5 and 6 - Dr. Cindy Chestek
Interpretations of Neural Recordings
Kushal Jaligama

1 - Introduction

When dealing with prosthetics, correct replication of neural signals is important to maintain parity between human limbs and artificial limbs. Neural recordings are typically high dimensional in size and making the data more manageable through dimensionality reduction allows neuroengineers to properly replicate the signals needed to react with an artificial limb.

2 - Methods

2.1 - Model Assumptions and Design for Interpreting Neural Recordings

The first model utilized for this lab covered the impedance spectra of the neural implant from (Williams, 2007) in order to explore the effects of gliosis and edema on electrode recordings at the time of implantation and 7 days later. Next, a sample 128-channel ECoG dataset was utilized to simulate the findings of (Pistohl, et. al., 2012), in which three frequency bands were extracted from ECoG data, in order to decode grasp types. Then, spike waveform data was treated as high-dimensional data, and each sample along the voltage trace was treated as a separate variable that described the waveform. Since many data points are interrelated, this high-dimensional data can be reduced to a lower-dimensional space by way of a process called Principal Component Analysis, compressing the data. This facilitates visualization of the data and spike sorting. A final data analysis tool known as Independent Component Analysis was used to gather information from an EEG dataset, which was derived from a configuration of 6 antennae inside of a head and 128 electrodes placed on top of a fake scalp, all referenced to their average.

2.2 - Signal Conditioning Methods

To gather information from a complex dataset, there must be a proper degree of granularity. Too much information causes in difficulty what parts of the data are characteristic to the effects of neural recordings, whereas too little information can result in inconclusive analysis. In this laboratory, there were several instances oof signal conditioning. Signals from the 128-channel ECoG data set were averaged and used as a reference to the 29th channel. This data was also smoothed over a 100 ms timeframe, and these two processes helped mitigate artifacts and noise from the resultant waveform. Spike waveform data was normalized such that every point along the snippets of voltage traces had a mean of 0 and a standard deviation of 1, to give each sample equal importance when calculating principal components.

2.3 - Software Tools Used to Implement Signal Analysis

This entire laboratory was performed in MATLAB, which gives access to several signal analysis tools. MATLAB's filter designer tools were used to extract waveforms within designated frequency bands. MATLAB also has built in functions to gather the Principal Components of a data set and even perform Independent Component Analysis. MATLAB can also be

used to apply Fast Fourier Transforms and even gather Power Spectrums by way of Welch's Power Spectral Density Estimation.

2.4 - Assumptions of Clustering Algorithms

Utilizing Principal Component Analysis allows one to project a dataset onto a lower dimensional space. Essentially, the dataset that will be used with clustering algorithms is typically difficult and impractical to analyze directly. Clustering and Gaussian mixing can help sort spikes into different units for further interpretation.

2.5 - Implementations of Clustering Algorithms

To start clustering points, one must calculate the Principle Components of a dataset, pick K random points to act as the centroids of clusters, then iteratively recalculate centroids for each cluster. After projecting the points into this space, one can iteratively cluster them and determine which points are similar to each other based on their squared Euclidean distance to the centroid of a cluster. The objective function serves to minimize this distance for all of the points in the data set. The group to which a data point belongs to changes based on the number of clusters and centroids, denoted by k (apdx: Table 1, #3). The centroid of a cluster can be calculated using the mean of all the points in a cluster (apdx: Table 1, #5). These centroids would be continuously recalculated until the clustering reaches convergence (when none of the points change assignment). One can extend this algorithm by utilizing the median of a cluster of points rather than the mean to calculate a new centroid point. Yet another modification to this algorithm is to utilize the Mahalanobis distance between a centroid and a point in a cluster instead of Euclidean distance (apdx: Table 1, #4). The Gaussian Mixture model further builds on this by incorporating a probabilistic approach to clustering of points. Points are placed into the most likely cluster based on the underlying distribution calculated from the data. This entails calculating the covariance of each cluster in addition to the new centroids of clusters. This can be implemented using the MATLAB function gmdistribution.fit.

3 - Results

3.1 - Outcome of Electrode Interface Calculations

Gathering the total impedance of an electrode at the time of insertion and 7 days later by utilizing Nyquist samples of complex and real impedance, one can determine the magnitude of impedance of an electrode. The results showed that after 7 days, gliosis and edema significantly increased impedance at the site of electrode insertion.

3.2 - Extracted Features from ECoG Signals

Electrocorticography data can be broken into various frequency ranges, each of which correspond to different neural function. The frequency ranges expressed in (apdx: Figure 1) show that each range has a different electrical power response. This is useful to know because particular behaviors can be correlated with a power response in a certain frequency domain.

3.3 - Accuracy and Limitations of Dimensionality Reduction

Looking at a dataset in a lower dimensional space makes it easier to draw conclusions about what is happening. One can extract the Principal Components of a data set and use them to reconstruct data in a way that maintains a good representation of the data but is compressed in size. One can use a varying number of Principal Components for reconstruction of a spike snippet shown in (apdx: Figure 2), and the more components used increases the granularity of the data. Comparing two plots of principal components can reveal which components of the data are heavily correlated. One can use clustering algorithms to detect which points are similar to one another and represent something common in the original waveform (apdx: Figure 3).

3.4 - Independent Component Analysis

Analyzing the individual principal components themselves can help understand the situation in which data originated from. Creating a topographic plot of principal components results in a heat map of sorts that describes voltage intensities in visual space (apdx: Figure 4). Applying a Fast Fourier Transform to Principal Components reveals the frequencies that contribute most to the data (apdx: Figure 5).

3.5 - Clustering Algorithms

Applying algorithms such as K-Means reveals what parts of a dataset are similar to one another, ultimately revealing what sorts of data sets are correlated with certain behaviors. Clustering can be done with the mean and median of data points, or by utilizing Gaussian Mixing to examine the underlying distribution properties (apdx: FIgure 6).

3.6 - Limitations and Advantages of Clustering Algorithms

These algorithms can often overgeneralize if we have too few features that are being clustered, and can be inconclusive if there are too few features. It is important to extract a number of features that retain the characteristics of the original dataset and also makes the dataset something that can trivially be analyzed. Using the mean and median of a dataset to calculate the centroid of a cluster changes the points that belong to a certain group.

4 - Discussion

Once a neural dataset has been effectively analyzed, and one can determine the inner features that characterize a large waveform, one can create electrodes with configurations that result in artificial kinematic behaviors replicating that of human limbs.

References

Williams, Justin C, et al. “Complex Impedance Spectroscopy for Monitoring Tissue Responses to Inserted Neural Implants.” *Journal of Neural Engineering*, vol. 4, no. 4, 2007, pp. 410–423., doi:10.1088/1741-2560/4/4/007.

Pistohl, Tobias, et al. “Decoding Natural Grasp Types from Human ECoG.” *NeuroImage*, vol. 59, no. 1, 2012, pp. 248–260., doi:10.1016/j.neuroimage.2011.06.084.

BIOMEDE 517 - Neural Engineering - Labs 5 and 6 - Dr. Cindy Chestek

Appendix

Kushal Jaligama

Table 1 - Equations Used for Electrostatic Modeling

#	Equation Name	Equation
1		
2		
3	Objective Function	$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\|\vec{x}_n - \vec{\mu}_k\|\|^2, \quad where\ r_{nk} = \begin{cases} 1, & if\ x_n\ is\ in\ group\ k \\ 0, & if\ x_n\ is\ not\ in\ group\ k \end{cases}$
4	Mahalanobis Distance	$d_M = \sqrt{(x-\mu_k)^T S_k^{-1}(x-\mu_k)}$
5	K-Means	$\vec{\mu}_k = \frac{1}{N_k}\sum_{n}^{N_k}\vec{x}_n$
6		

Figure 1 - Common Average Referencing Data

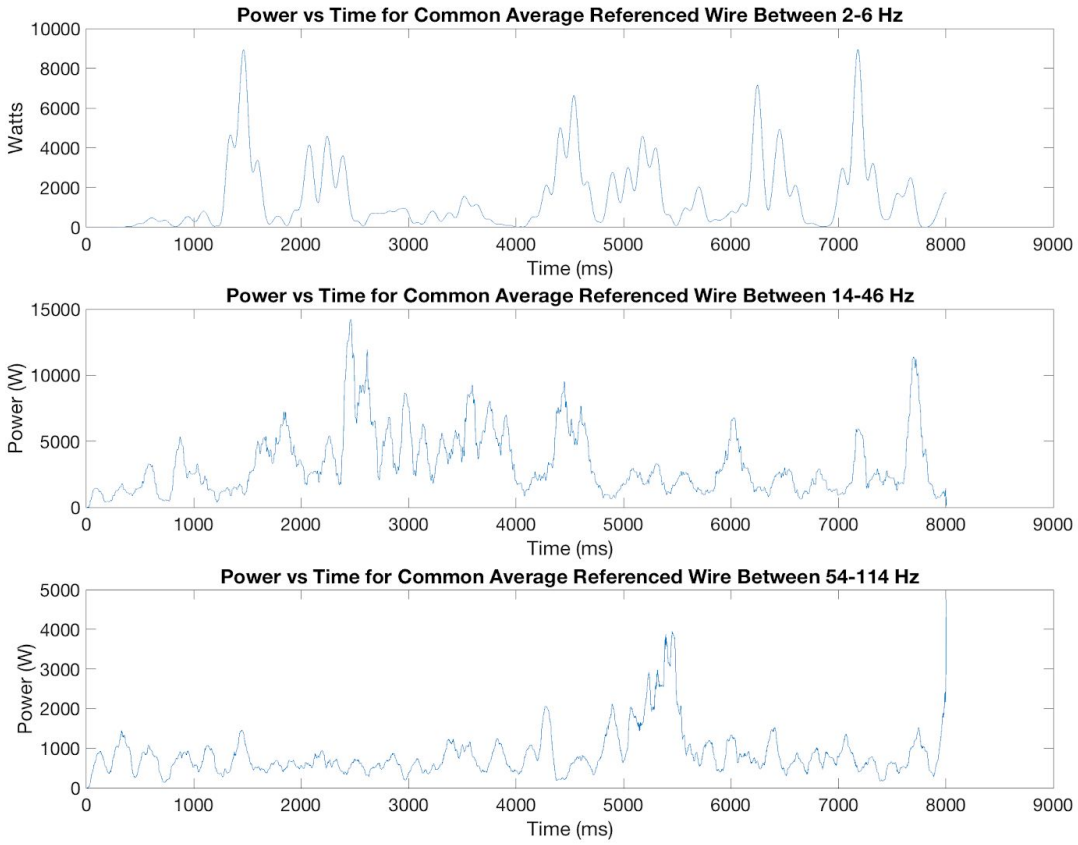


Figure 2 - Reconstructed Waveform with 9 PCs

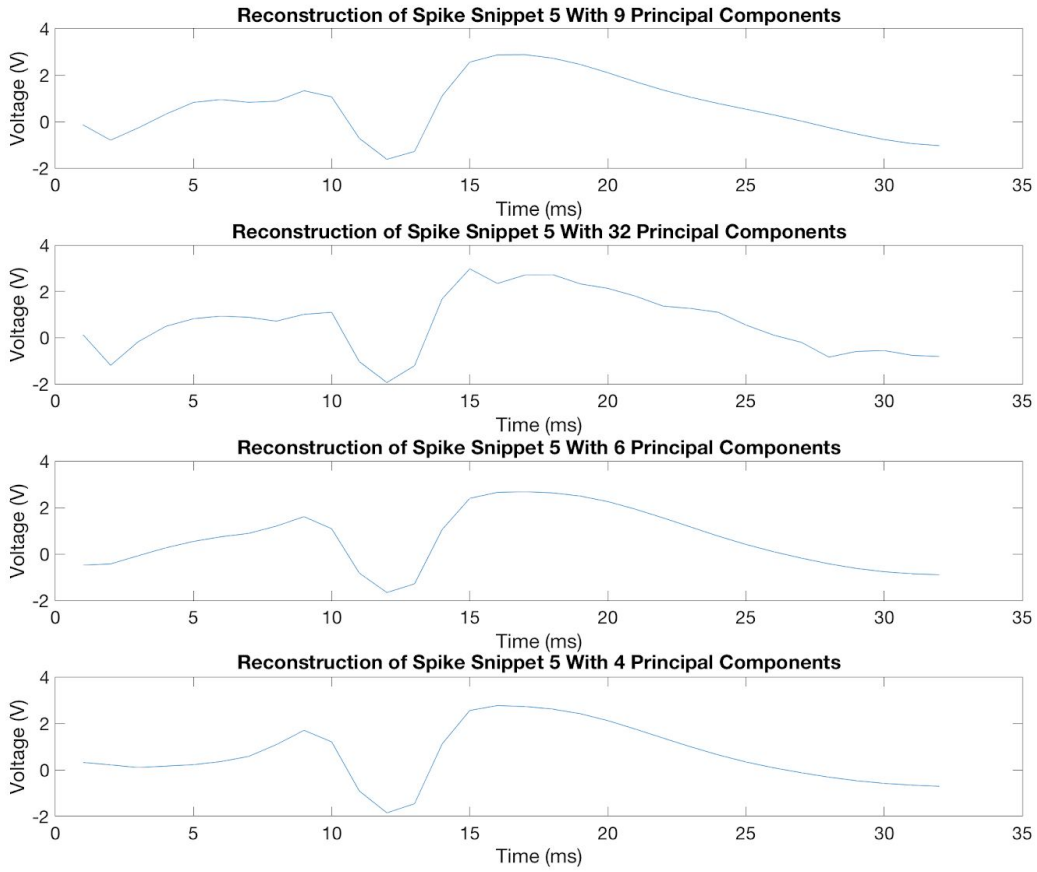


Figure 3 - Principal Component Comparison

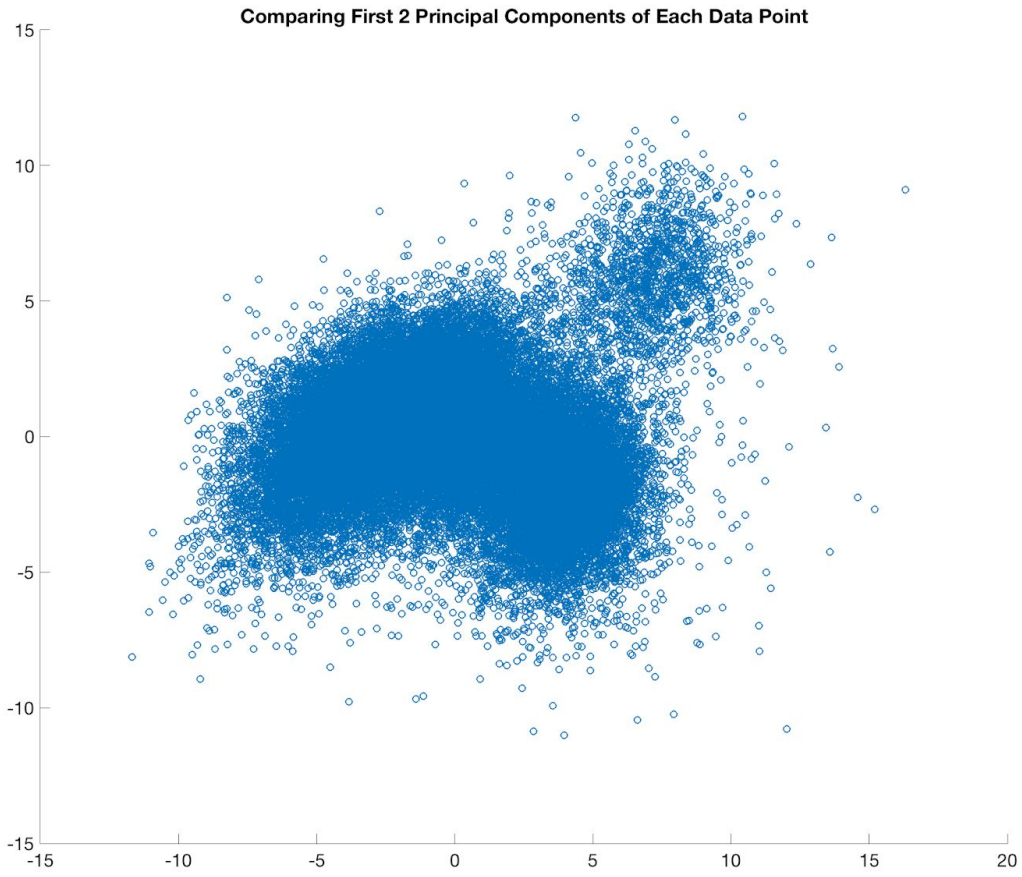
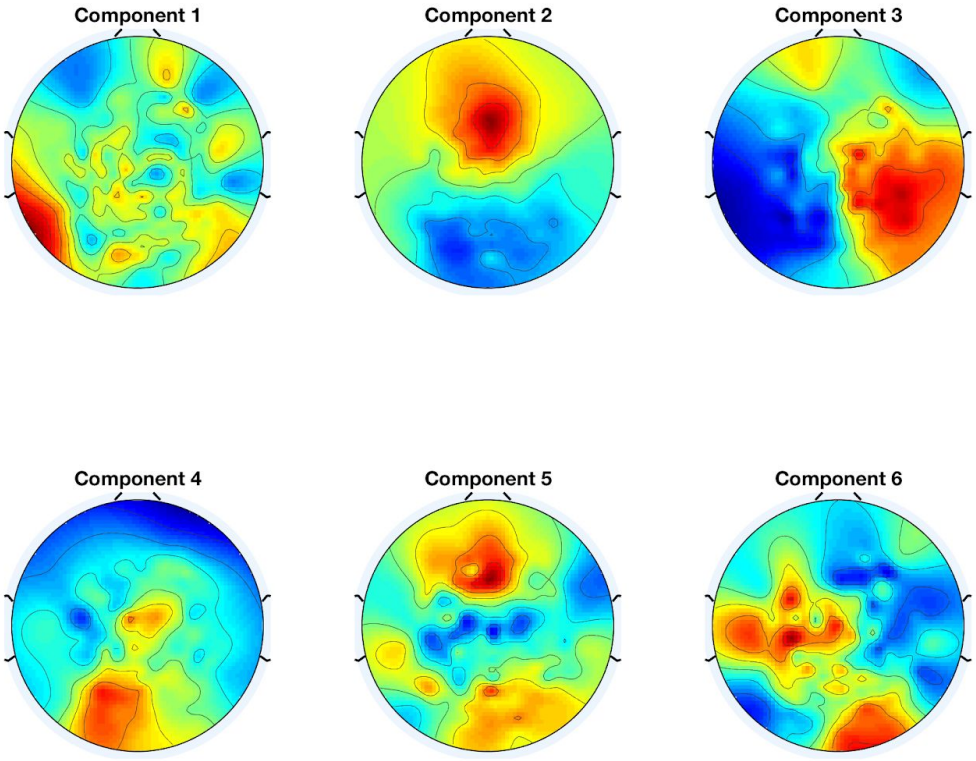


Figure 4 - Topoplots



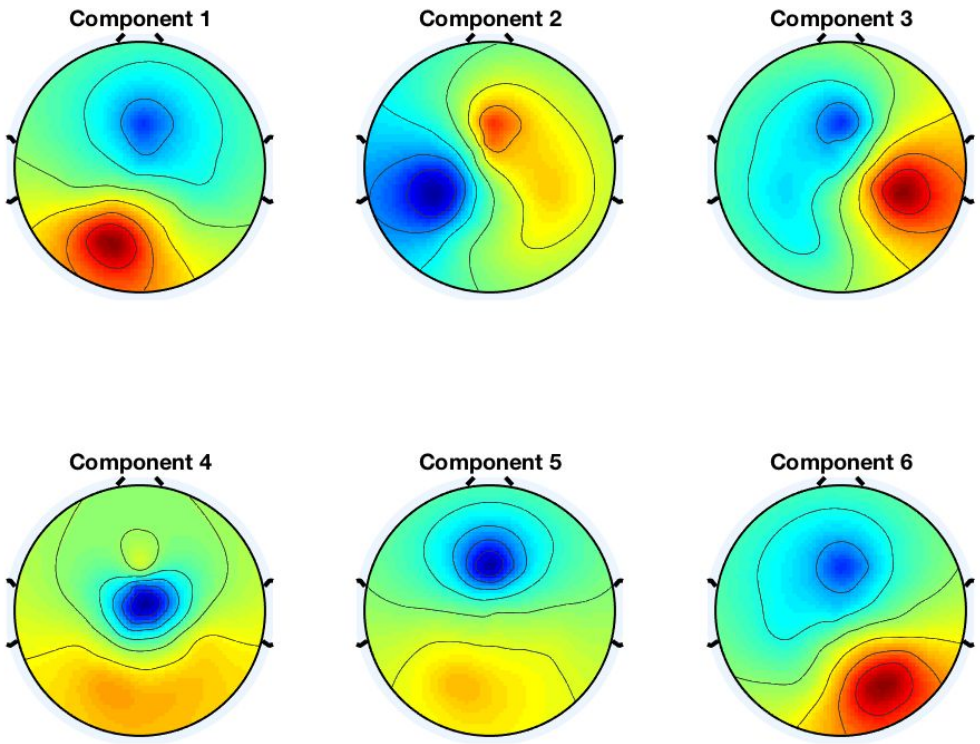
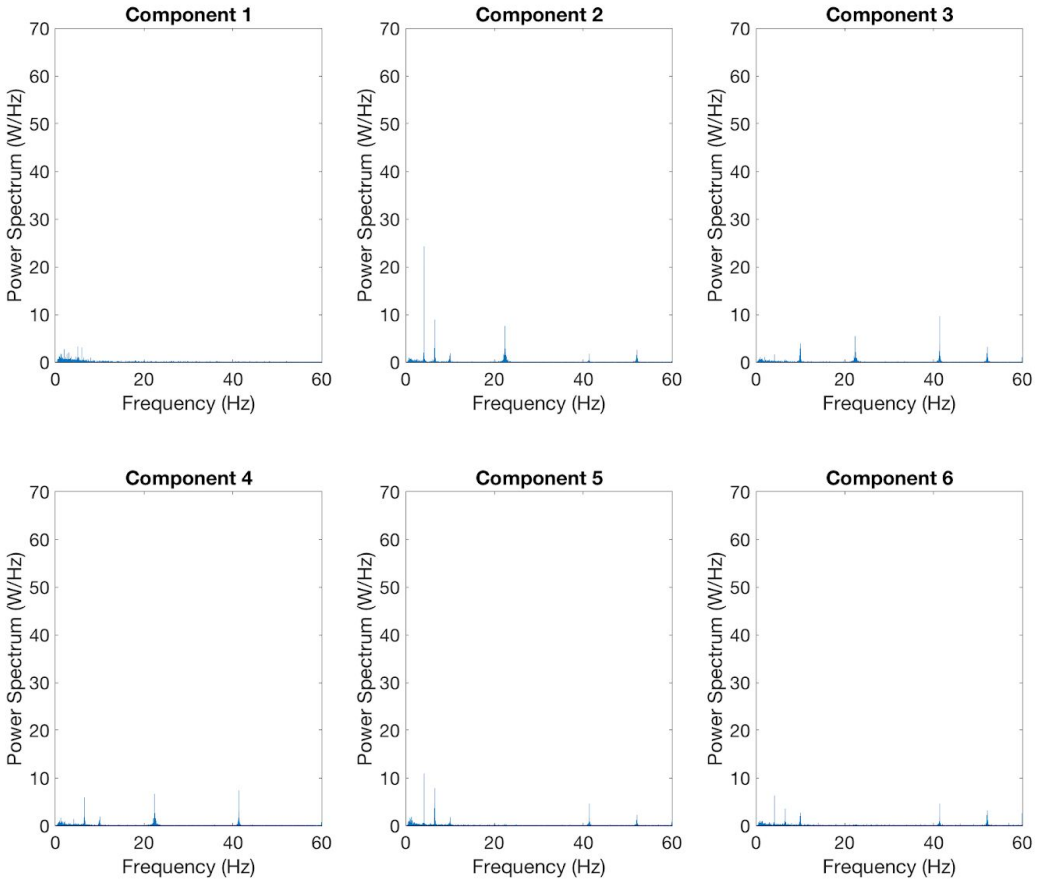

Component 1
Component 2
Component 3
Component 4
Component 5
Component 6

Figure 5 - FFT Power Spectra



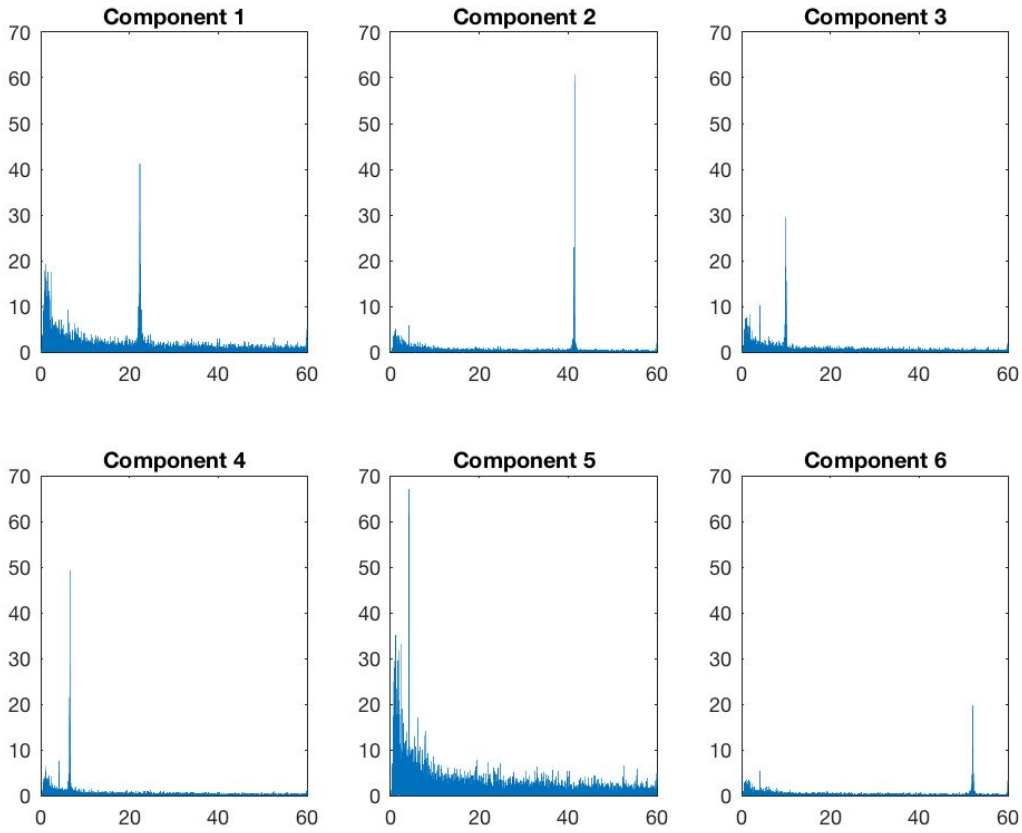
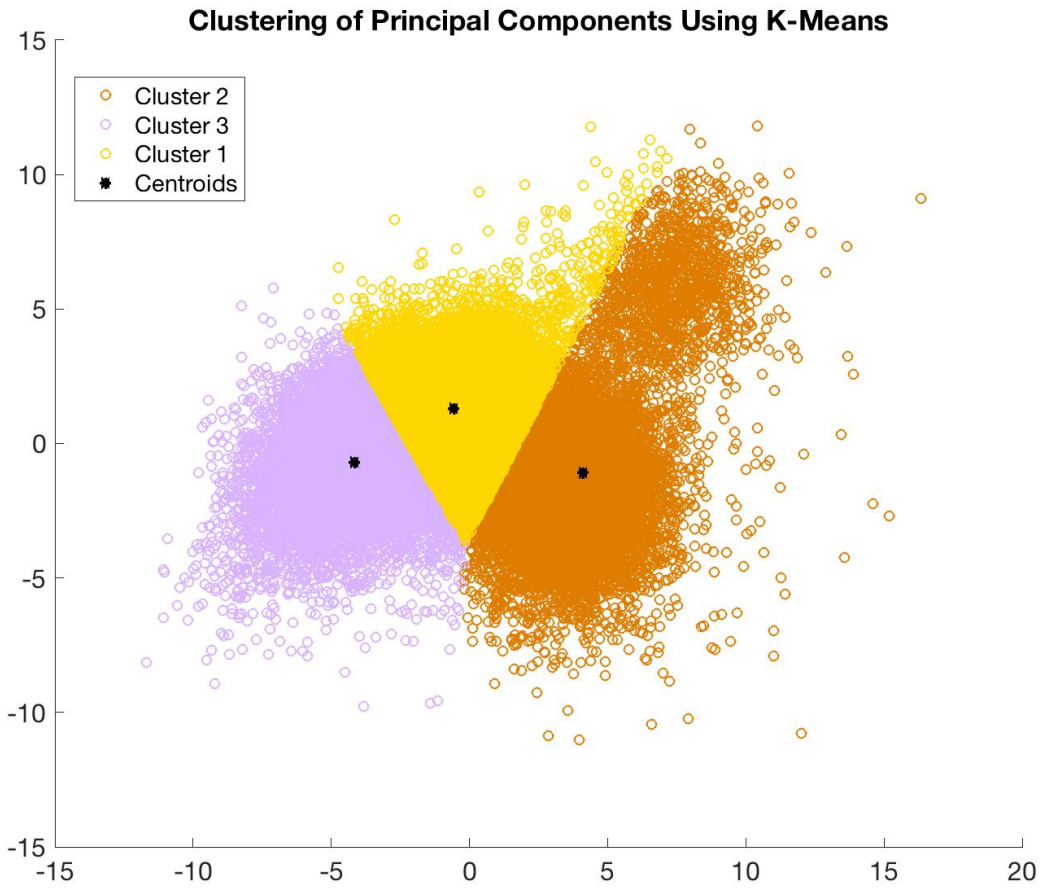
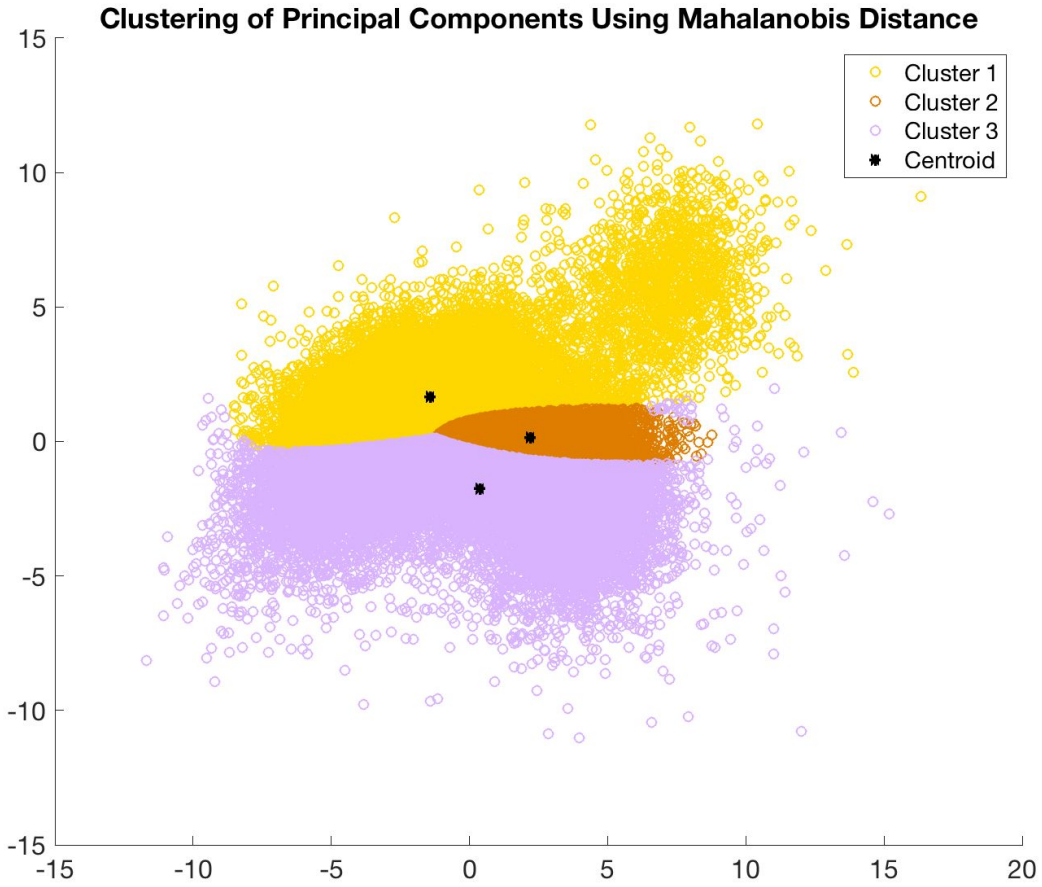
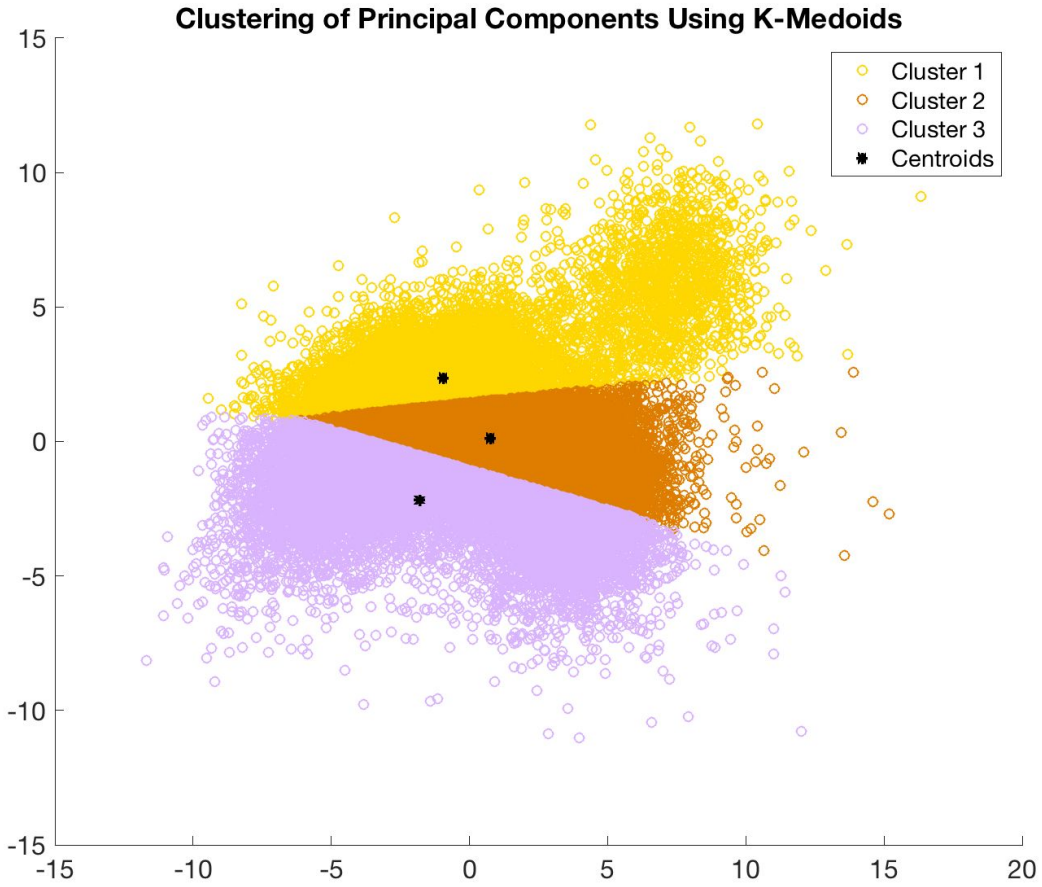

Component 1
Component 2
Component 3
Component 4
Component 5
Component 6
0
10
20
30
40
50
60
70
0
20
40
60

Figure 6 - Clustering Algorithms




Clustering of Principal Components Using Mahalanobis Distance
Cluster 1
Cluster 2
Cluster 3
Centroid
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20


Clustering of Principal Components Using K-Medoids
Cluster 1
Cluster 2
Cluster 3
Centroids
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
20

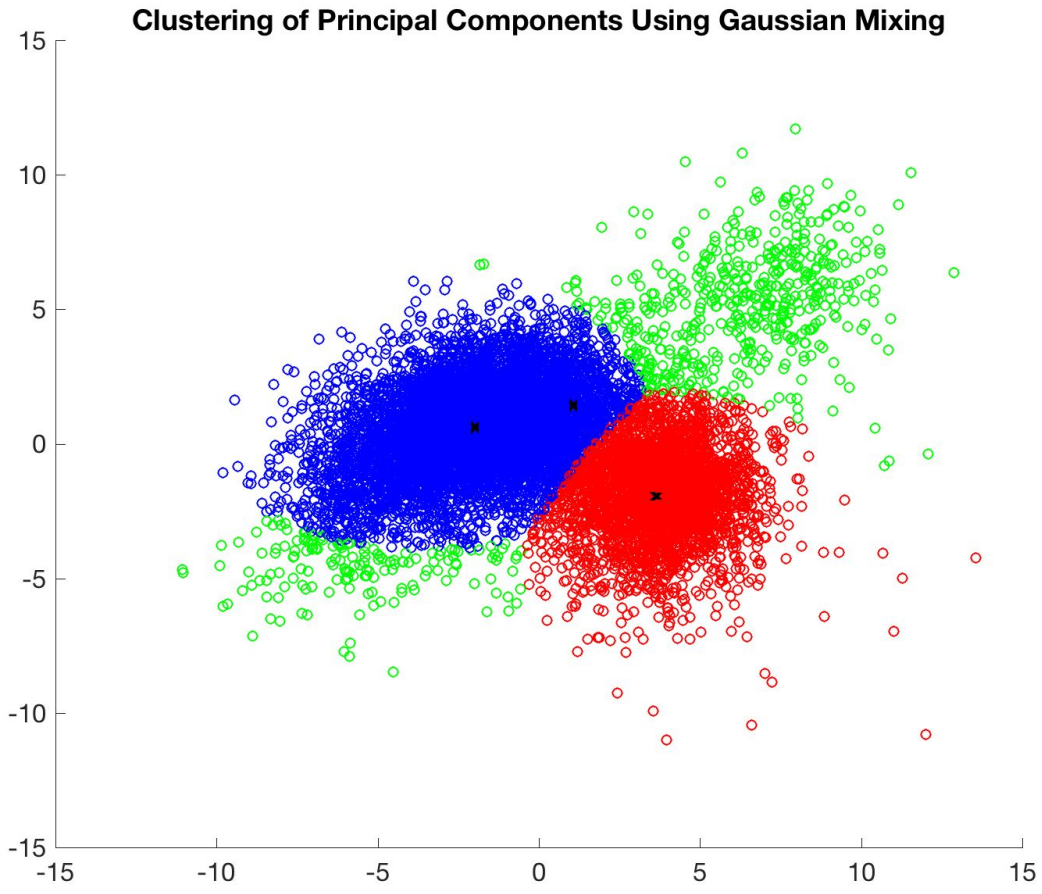
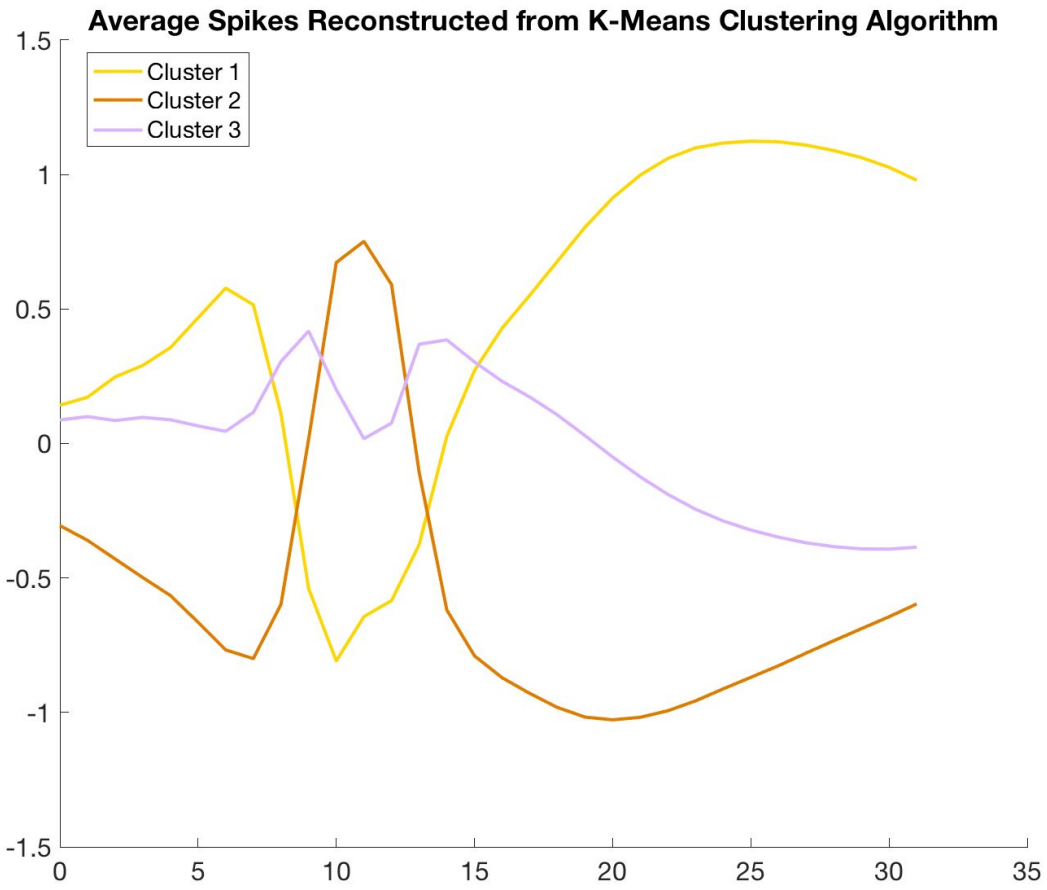

Clustering of Principal Components Using Gaussian Mixing
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15

Figure 7 - Average Spike from Clusters




```
% BIOMEDE 517 - Neural Engineering
% Lab 5 Part 1
% Kushal Jaligama

% Part 1
% In this part, we use data from Williams 2007
% The frequencies of the points on the plot are taken from 100 Hz to 2 kHz
% in 100 Hz increments

% Estimate Ren + Rex - extrapolate the first few points
% Point = [Z_real, Z_imaginary]
Blue1 = [175000, 250000]
Blue2 = [110000, 160000]
slopeBlue = (Blue1(2) - Blue1(2)) / (Blue2(1) - Blue2(1));
blue_x_intercept = (slopeBlue * Blue1(1) - Blue1(2)) / slopeBlue;
RenplusRexBlue = blue_x_intercept; % Ohms
ZreBlue = Blue1(1); % Ohms
ZimBlue = Blue1(2); % Ohms
% Calculate the alpha value
```

```
alphaBlue = 2/pi()*atan(slopeBlue);

Red1 = [420000, 250000]
Red2 = [375000, 175000]
slopeRed = (Red1(2) - Red2(2)) / (Red1(1) - Red2(1));
red_x_intercept = (slopeRed * Red1(1) - Red1(2)) / slopeRed;
RenplusRexRed = red_x_intercept; % Ohms
ZreRed = Red1(1); % Ohms
ZimRed = Red1(2); % Ohms
% Calculate the alpha value
alphaRed = 2 / pi() * atan(slopeRed);

% By looking at values, RenplusRexBlue is NaN so make it zero
RenplusRexBlue = 0;

% In the paper the points range from 2Khz to 100Hz in increments of 100
% Z_real and Z_re for calculating K were gathered at 100 Hz
w_naught = j * 2 * pi() * 100;

% Calculate K for both lines
% Magnitude of a complex number (a + bi) is sqrt(a^2 + b^2)
% Magnitude of a complex number is also the absolute value of it
% K = (Zre(w_naught) - jZim(w_naught) - (Ren+Rex)) * (j*w_naught)^alpha
% The parenthesis in Zre(w_naught) are NOT multiplication, rather functions
% to grab the Zre at the 0th frequency (which is 100 in this paper).
KBlue = (ZreBlue - j*ZimBlue - RenplusRexBlue) * (j*w_naught)^alphaBlue
KRed = (ZreRed - j*ZimRed - RenplusRexRed) * (j*w_naught)^alphaRed

% Calculate magnitude of tissue related response (Ztotal) at 1000 Hz
w = j * 2 * pi() * 1000;
ZtotalBlue = abs(RenplusRexBlue + KBlue / (j * w)^alphaBlue)
ZtotalRed = abs(RenplusRexRed + KRed / (j * w)^alphaRed)

% ZtotalBlue and ZtotalRed are of the form (a + bi)

% What conclusions can you make about the electrode performance and the tissue response at the time of implantation and seven
days later?

% BIOMEDE 517 - Neural Engineering
% Lab 5 Part 2
% Kushal Jaligama

% Part 2
% Apply common average referencing to channel 29, only using channels
% in refChannels
refIndex = 1;
refzero = 1;

% refEcogData vector will contain all of the channels that are 1 in
% refChannels
for i=1:128
    if (refChannels(i) == 1)
        row = ecogData(i,:);
        refEcogData(refIndex, :) = row;
```

```
        refIndex = refIndex + 1;
    end
    if (refChannels(i) == 0)
        refzero = refzero + 1;
    end
end

% Gather the common average of the channels
average = mean(refEcogData);
% Then subtract the average from channel 29 to reference it
CAR_row = ecogData(29, :) - average;

% Now create filters that will grab the 3 bands of data specified in
% Pistohl et. al. 2011
low_freq = 2; % Hz
high_freq = 6; % Hz
low_band = designfilt('bandpassiir','FilterOrder',20,'HalfPowerFrequency1',low_freq,'HalfPowerFrequency2',high_freq,'SampleRate',1000);
low_freq = 14; % Hz
high_freq = 46 % Hz
intermediate_band = designfilt('bandpassiir','FilterOrder',20,'HalfPowerFrequency1',low_freq,'HalfPowerFrequency2',high_freq,'SampleRate',1000);
low_freq = 54; % Hz
high_freq = 114; % Hz
high_band = designfilt('bandpassiir','FilterOrder',20,'HalfPowerFrequency1',low_freq,'HalfPowerFrequency2',high_freq,'SampleRate',1000);

% Apply the filters to gather the waveforms of each frequency range
low_out = filter(low_band, CAR_row);
inter_out = filter(intermediate_band, CAR_row);
high_out = filter(high_band, CAR_row);

% Plot the data
% We have 8003 snippets of data recorded at 1000Hz (each snippet is 1 ms)
figure(1);
subplot(3, 1, 1);
plot(smooth(low_out.^2, 100));
subplot(3, 1, 2);
plot(smooth(inter_out.^2, 100));
subplot(3, 1, 3);
plot(smooth(high_out.^2, 100));

% BIOMEDE 517 - Neural Engineering
% Lab 5 Part 3
% Kushal Jaligama

% Part 3
% Dimensionality Reduction of Spike Recordings
close all
% Step 1, normalize the data to have mean of 0 and standard deviation of 1
normalized_spikes = spikes;
for i=1:size(spikes,2)
    normalized_spikes(:,i) = (spikes(:,i)-mean(spikes(:,i))) / std(spikes(:,i));
end
```

```
figure(5)
plot(linspace(1, 124, 41568), spikes)

% Time axis for the reconstructed spike snippets
time_axis = linspace(0, 3, 32); % Each snippet is 3 ms long and has 32 samples

% Get the PCA components, u is eigenvectors (these represent the principal components of the dataset)
% w is scores, latent is eigenvals
[u, w, eigenvals] = pca(normalized_spikes);

% Determine how many of the princip components capture 90% of variance
covariance = cumsum(eigenvals)/sum(eigenvals);
% The first covariance component that has .9 is covariance(9)
K = 9;
% Pick a representative spike and plot the top k eigenvectors of the data
% These eigenvectors correspond to the k largest eigenvalues
spike_num = 5; % We are asked to analyze spike number 5
% Perform a matrix multiplication of the scores and the first 9 PC vectors
spike_one = w(1, 1:K)*transpose(u(:,1:K)); % Grab 9 columns of eigenvectors and transpose
figure(1);
plot(0:1:31, spike_one);

% Plot reconstructed spikes based on different numbers of prinicpal
% components

subplot_x = 4; % How many rows there are
subplot_y = 1; % How many columns there are
subplot_num = 1;

num_pcas = [9 32 6 4];
figure(2);
for figs=1:4
    K = num_pcas(figs);
    reconstructed_spike = w(spike_num, 1:K) * transpose(u(:, 1:K));
    subplot(subplot_x, subplot_y, subplot_num);
    title(sprintf('Reconstruction of Spike Using %f Principal Components', num_pcas));
    plot(reconstructed_spike);
    subplot_num = subplot_num + 1;
end

% Extract first two principal components for all the data
first_two_princips = w(:, 1:2);

figure(3)
scatter(w(:, 1), w(:, 2));
title('Comparing First 2 Principal Components of Each Data Point')


%% Part 1: Run ICA

%Set path
addpath('extraFunctions'); %add path to topoplot and ICA functions
load('eegPhantomDataSnippet.mat'); %Load 5 minutes of 128-channel EEG data
```

```
%Run ICA with PCA reduction down to 60 channels
tic
[wts,sph] = runica(eegData,'extended',0,'pca',60,'maxsteps',512);
toc

%extended off: average step time = 0.148 sec
%extended on: average step time = 22 sec

%ICA_activations = wts * sph * data;
W=wts * sph;
ICs= W* eegData;

%% Part 2: Look at weights for first 6 components
load('phantomDataChanlocs.mat');
invW=pinv(W);
figure('name','Component topoplots');
for i=1:6
  subplot(2,3,i);
  topoplot(invW(:,i), chanlocs);
  title(['Component ' num2str(i)]);
end


%% Part 3: Run FFT on components
Fs=256; %sampling rate (Hz)
figure('name','Power spectra');
for i=1:6
  subplot(2,3,i);
  % Run Fast Fourier Transform (FFT) on first 6 components
  waveform = fft(ICs(i, :));
  %  plot(waveform);
  %Use pwelch with hamming window (see matlab documentation for pwelch)
  [spectrum, f] = pwelch(ICs(i,:), hamming(length(ICs)), [],[],Fs);
  plot(f, spectrum);
  xlim([0 60]); %only look at frequencies below 60 Hz
  ylim([0 70]); %optional y-axis setting
  title(['Component ' num2str(i)]);
end
figure('name', 'Fourier')
plot(waveform)

%% Part 4: Look at AMICA results (compare to ICA run performed)
ICs_orig=ICs;
load('icaweights_amica.mat');
load('icasphere_amica.mat');
W=icaweights*icasphere;
ICs=W*eegData;


%Run FFT and topoplot on the resulting components and tell which antenna
%corresponds to each frequency
load('phantomDataChanlocs.mat');
invW=pinv(W);
```

```
figure('name','Component topoplots');
for i=1:6
    subplot(2,3,i);
    topoplot(invW(:,i), chanlocs);
    title(['Component ' num2str(i)]);
end

Fs=256; %sampling rate (Hz)
figure('name','Power spectra');
for i=1:6
    subplot(2,3,i);
    % Run Fast Fourier Transform (FFT) on first 6 components
    waveform = fft(ICs(i, :));
    %     plot(waveform);
    %Use pwelch with hamming window (see matlab documentation for pwelch)
    [spectrum, f] = pwelch(ICs(i,:), hamming(length(ICs)), [],[],Fs);
    plot(f, spectrum);
    xlim([0 60]); %only look at frequencies below 60 Hz
    ylim([0 70]); %optional y-axis setting
    title(['Component ' num2str(i)]);
end
```