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

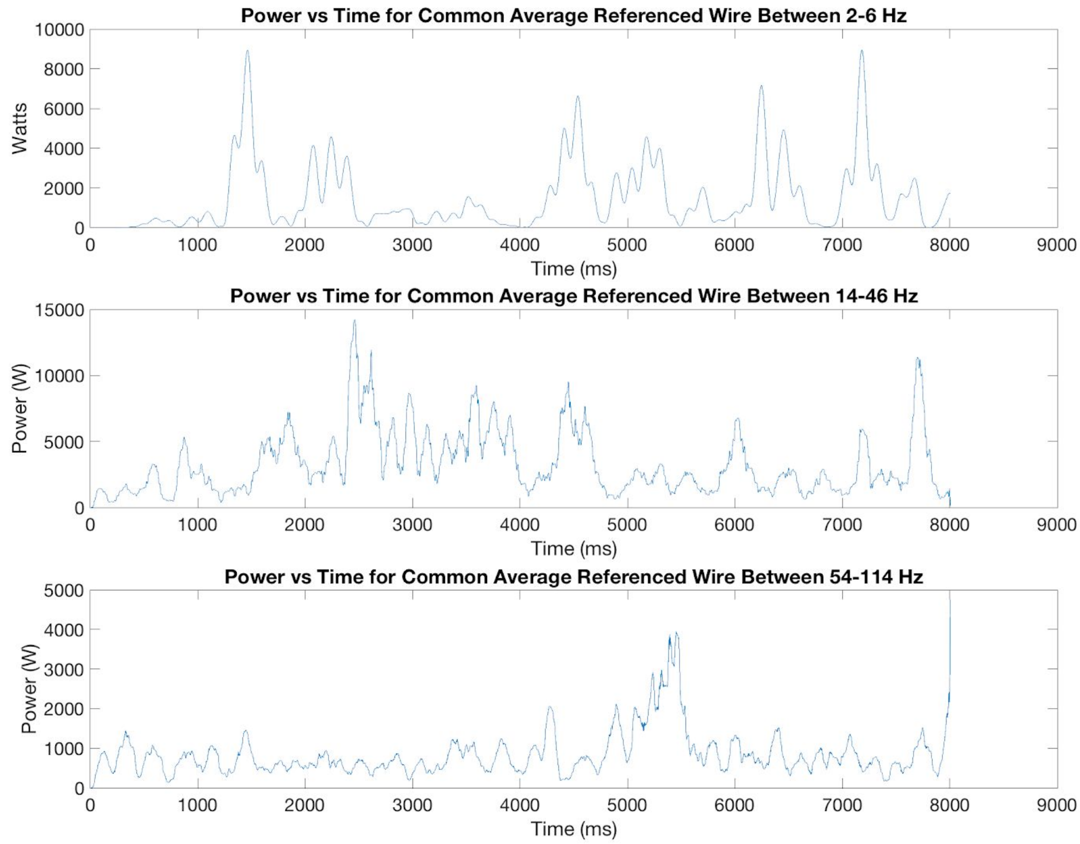
Appendix

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		

<u>Figure 1 - Common Average Referencing Data</u>



<u>Figure 2 - Reconstructed Waveform with 9 PCs</u>

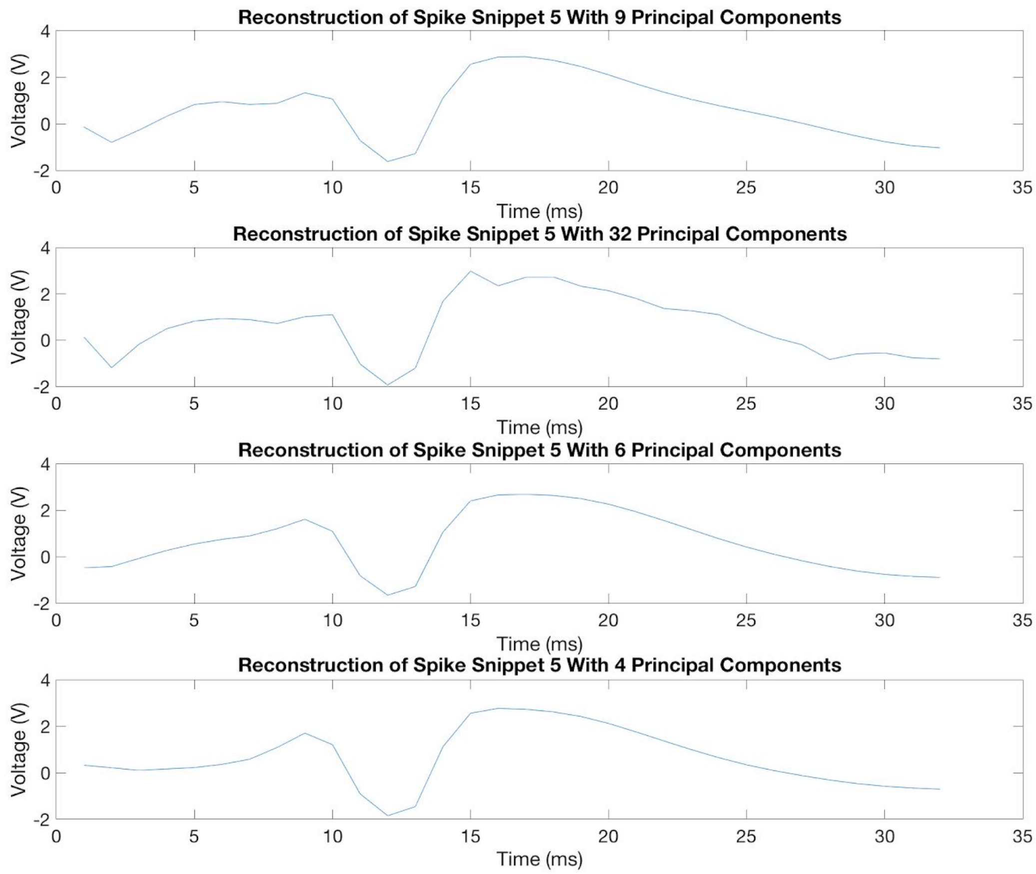


Figure 3 - Principal Component Comparison

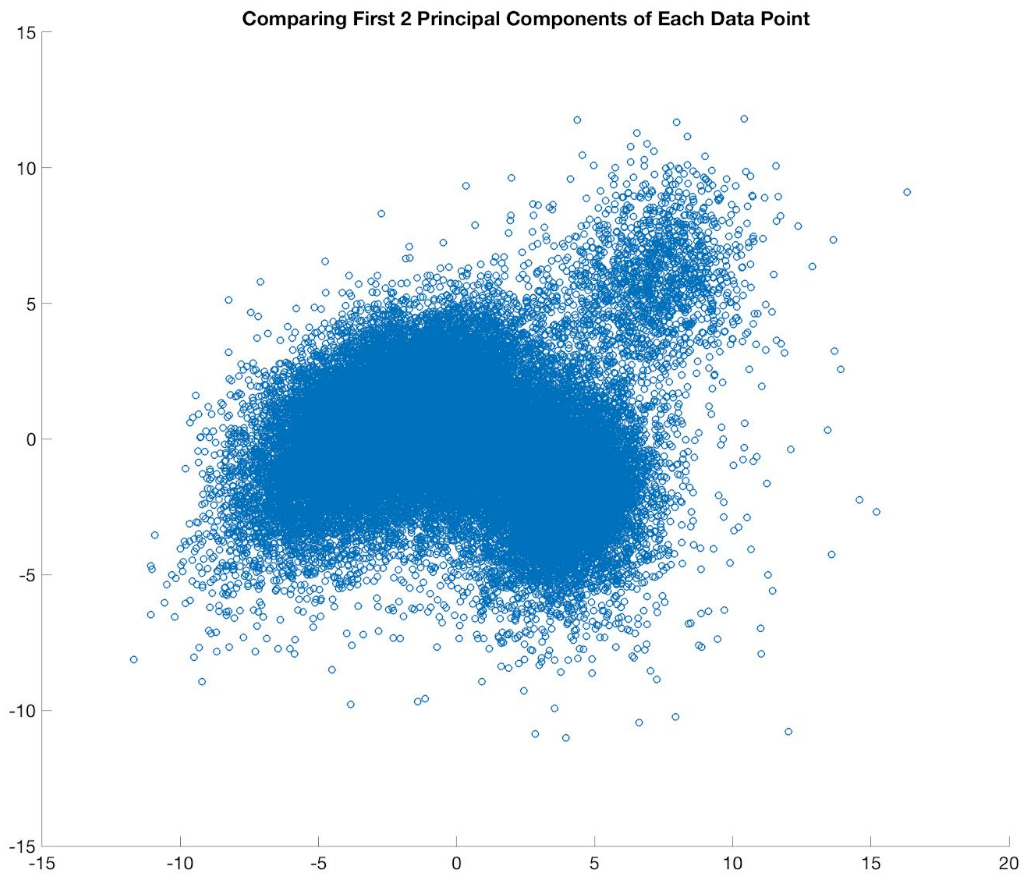
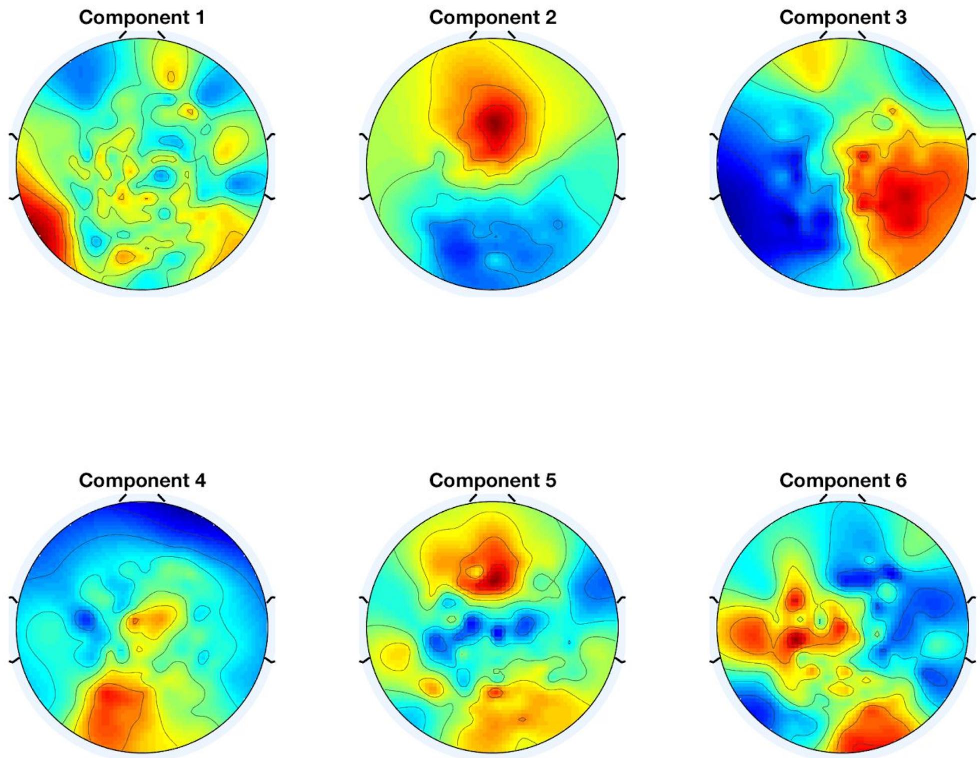


Figure 4 - Topoplots



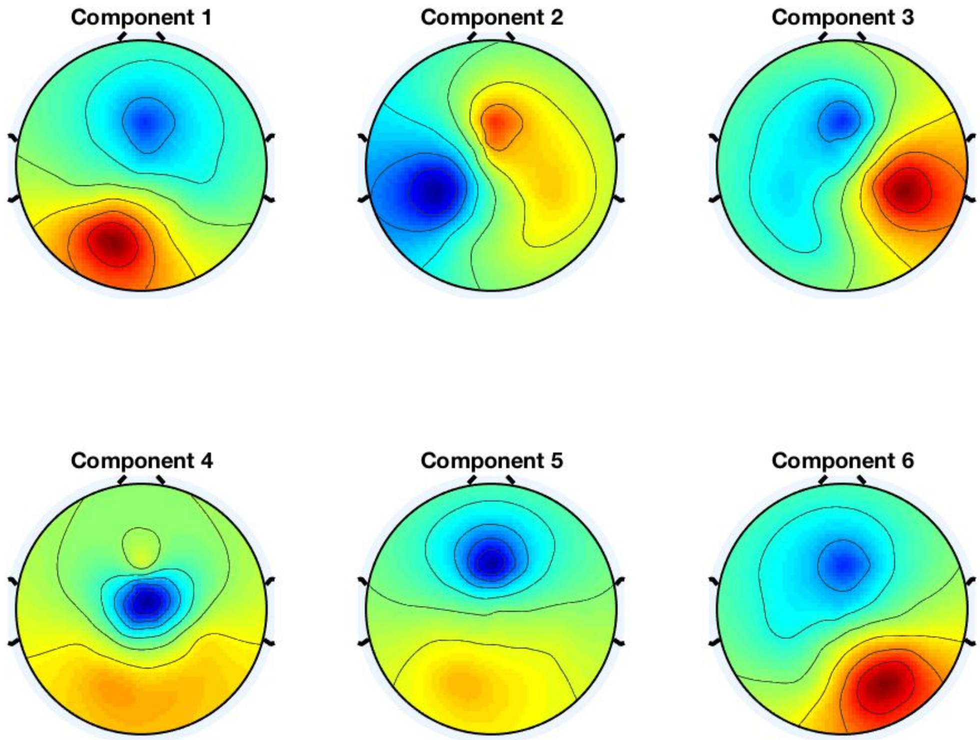
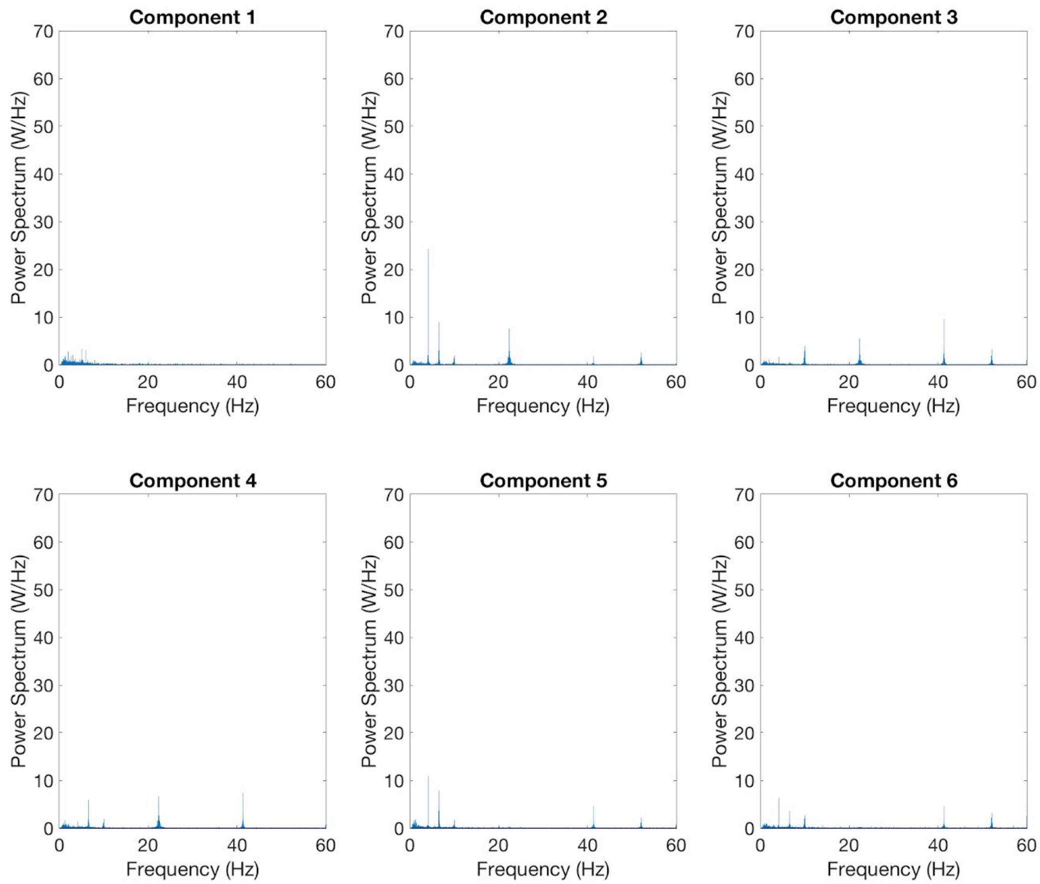

Component 1
Component 2
Component 3
Component 4
Component 5
Component 6

Figure 5 - FFT Power Spectra



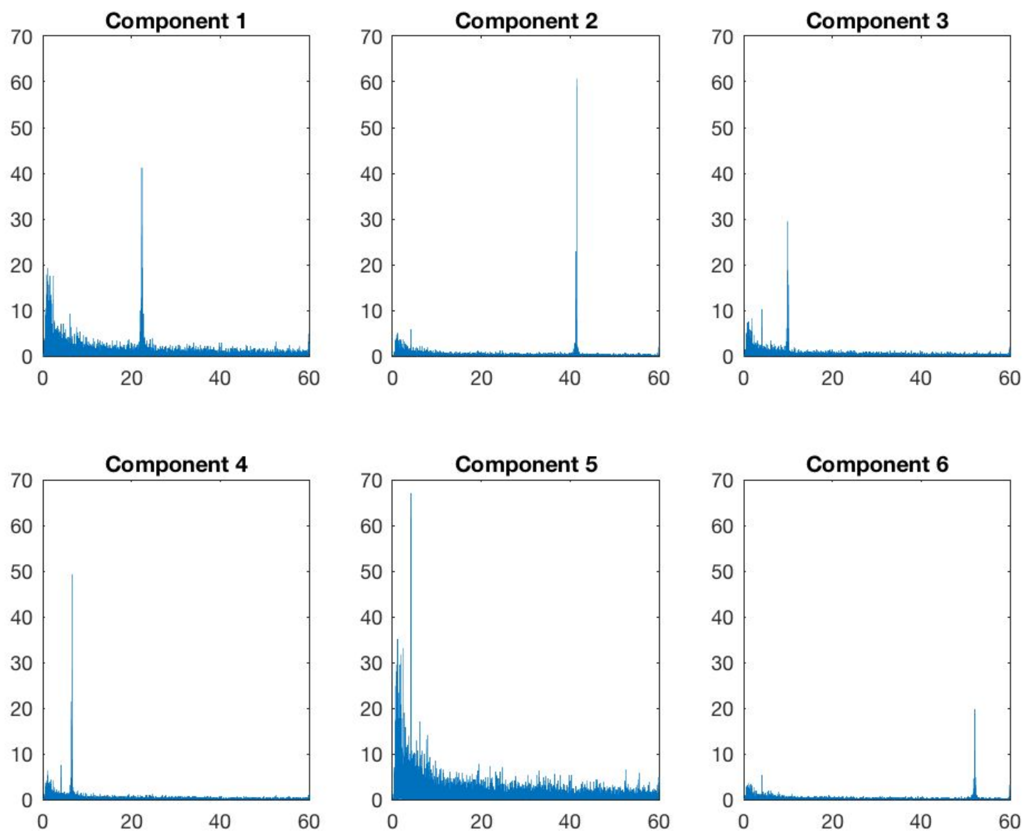
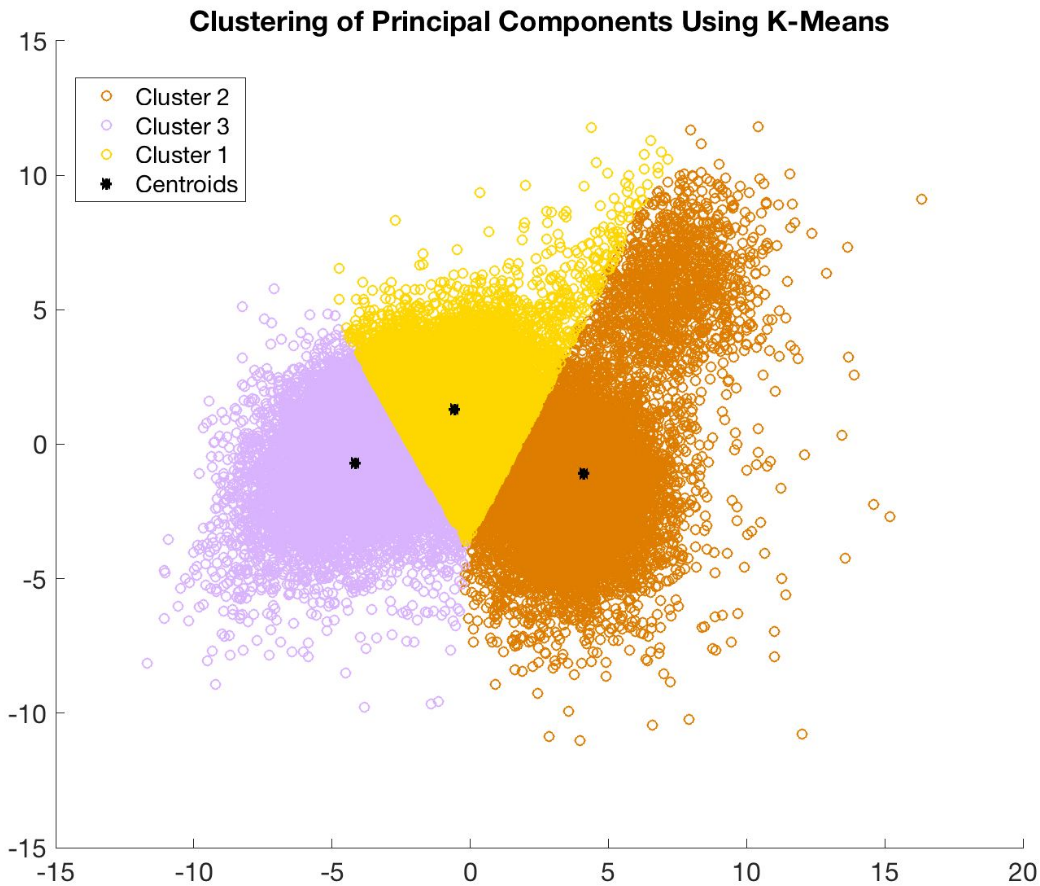
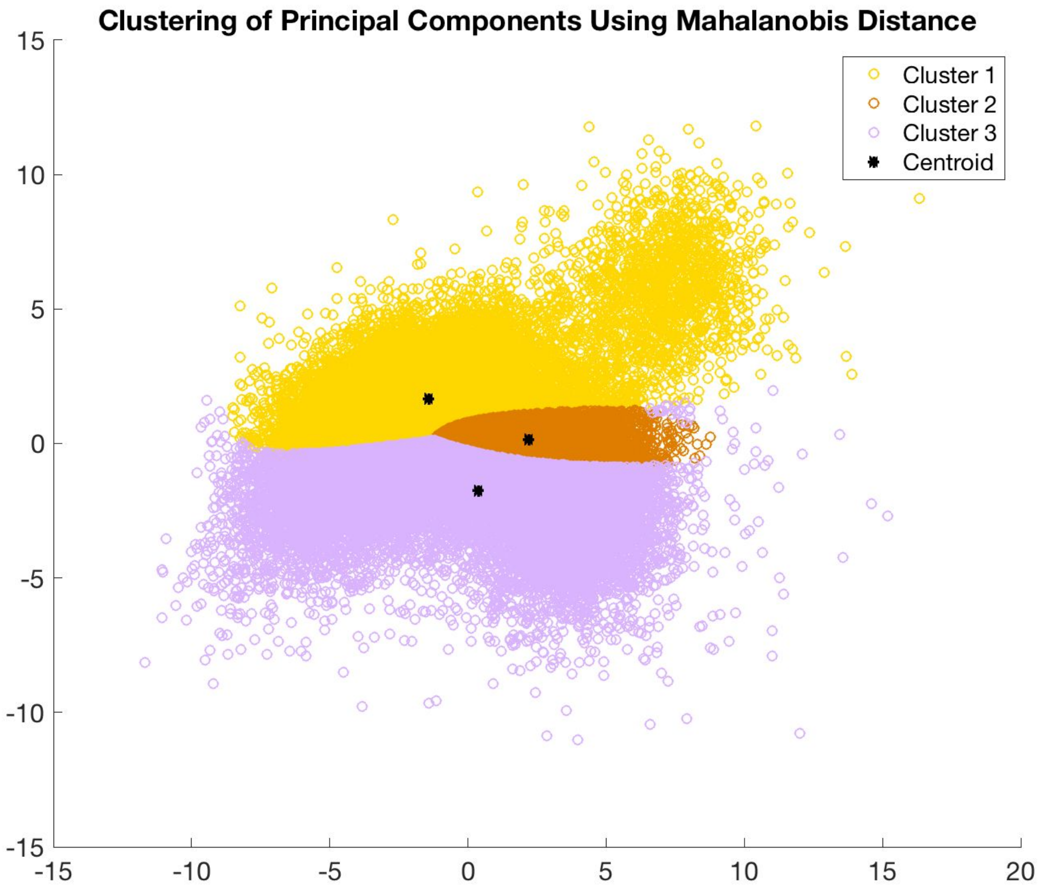
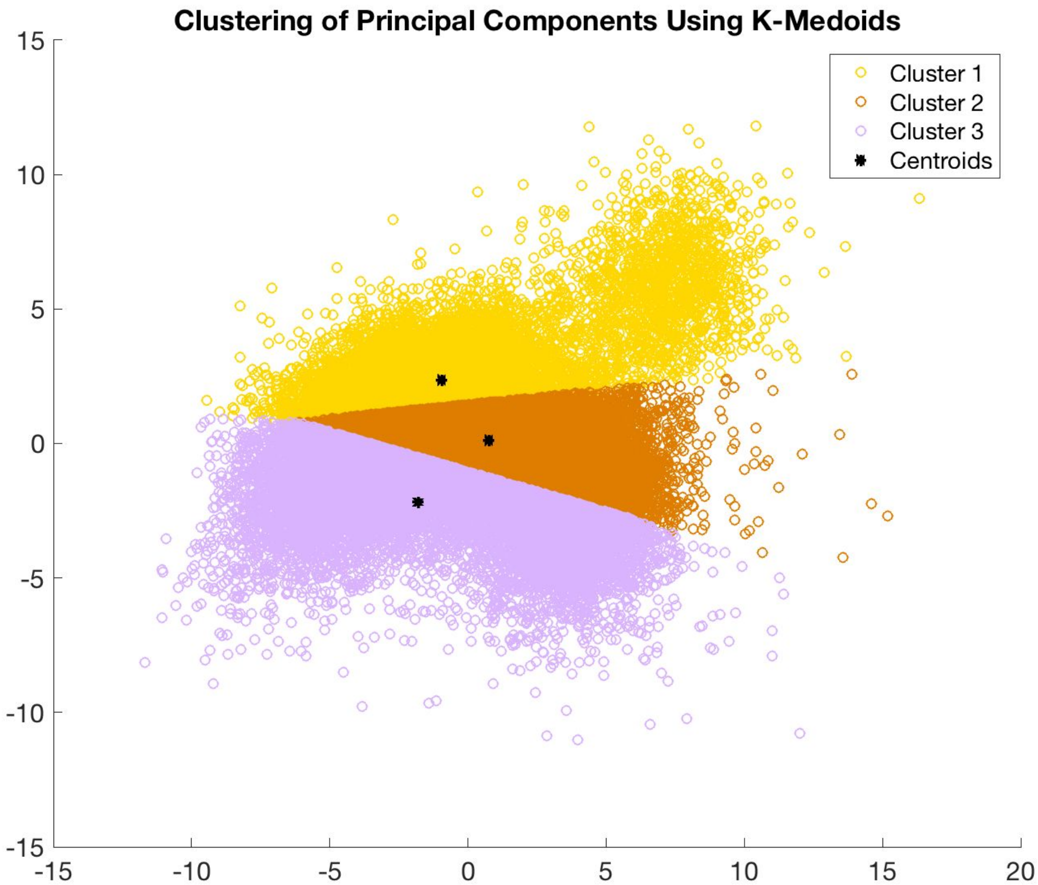

Component 1
Component 2
Component 3
Component 4
Component 5
Component 6
70
60
50
40
30
20
10
0
0
20
40
60

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

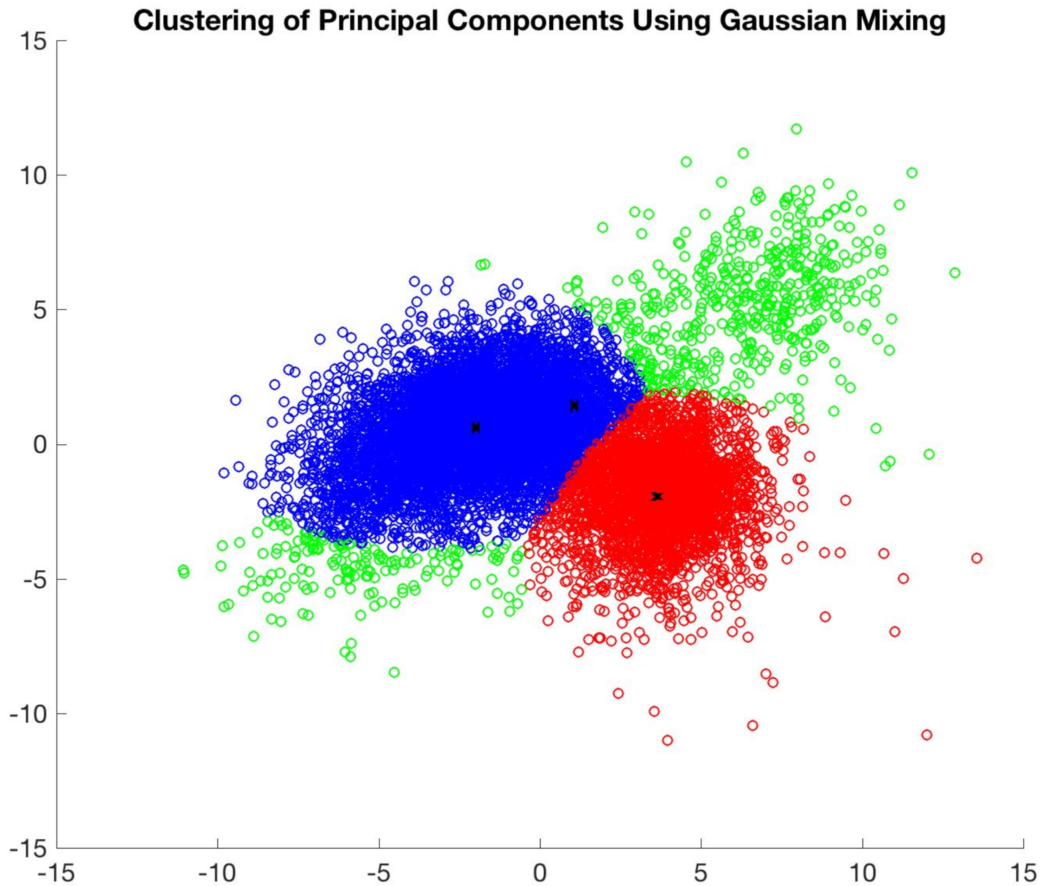
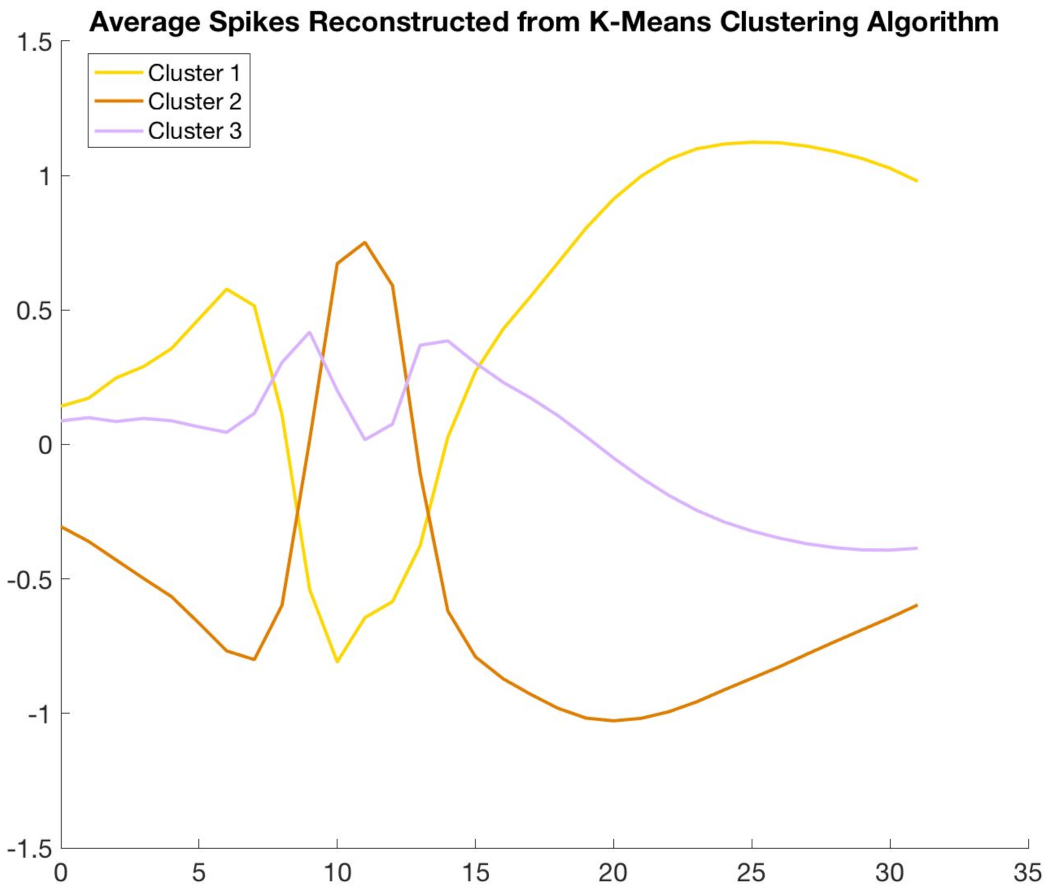

Clustering of Principal Components Using Gaussian Mixing
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15

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
```