Lab Report 5 & 6

Introduction

Understanding neuron spike data is an important step in comprehending the nervous system. Recordings from the brain must be interpreted properly to get useful information from the data. For instance, ECoG signals can be used as a control signal for brain-machine interfaces but must be filtered first. Principal component analysis (PCA) can be used for spike sorting to distinguish neurons in a dataset. PCA can produce a plot of the principal components (PC) which can be sorted by various spike sorting algorithms. This paper analyzes tissue response to electrodes, approaches to analyze local field potentials, dimensionality reduction of spike recordings, and various spike clustering algorithms.

Methods

The first portion of this paper estimated the effects of tissue response from impedance spectra. Figure 1 (Figure 3D in 'Complex impedance spectroscopy for monitoring tissue responses to inserted neural implants' by Justin C Williams et al. (2007)) has Nyquist plots of an electrode after implementation and 7 days later. This was used to find the values of $R_{en}+R_{ex}$ where if $R_{en}+R_{ex} < 0$ it was treated as 0. $R_{en}+R_{ex}$ was extrapolated to be the x - intercept of each line using a ruler from the straight portion at the top and was found to be 205,000 Ω for the 7-day implant and 150,000 Ω for the fresh impalnt. The alpha for each graph was estimated using the slope from the estimated top point, x-intercept, and Equation 1. The K values of each line was found using Equation 2 with a frequency of 100 Hz. Finally, the tissue related response and magnitude at 1kHz, Z_{total} ,was found using Equation 3. The total impedance was compared for the fresh and 7 day electrode. Most of the computations were done in MATLAB.

ECoG data was taken in and filtered to show meaningful features. The file ecogdatasnippet.mat was loaded into MATLAB and common average referencing (CAR) was done to channel 29, only using active channels. This was done by taking the average of all the samples that had a 1 in ecogdatasnippet.refChannels and subtracting that from channel 29's values. Bandpass filters with 3 ranges, 2 -6 Hz, 14-46 Hz, and 54 - 114 Hz (as seen in Pisthol et al (2011), 'Decoding natural grasp types from human ECoG) were used on channel 29's CAR data and plotted. The power of each filtered signal was obtained by squaring the voltage, smoothed, and plotted on Figure 2.

A PCA of the data was then performed to see how the PCs can be used to reconstruct the data. PCA assumes there is linearity in the data set. A file containing 41568 snippets of spikes each 32 samples long was imported into MATLAB. The data was first normalized and then the MATLAB PCA function was used to find the PC. The number of PCs necessary to capture 90% of the variance was identified as 9 by using Equation 4 and seeing how many PCs make the equation greater than .90 .The 9 PCs of spike 5 were then plotted in figures by using Equation 5 for each sample for each PC for Figure 3 - 11. Figure 12 and 13 show spike 5 normalized and the original spike respectively. Spike 5 was reconstructed using all PCs by multiplying the weight of the eigenvector by the transpose of the eigenvector and compared to the original waveform in Figure 14. Spike 5 was also approximated using 6 and 4 PCs and compared with the original

waveform in Figures 15 and 16. Finally the first PC was plotted against the second PC in Figure 17.

The k-means algorithm was used to cluster spikes from the same group together. This algorithm minimizes the objective function shown in Equation 6. This and k-medoids are partitional clustering algorithms that break up the dataset into distinct, non-overlapping groups without any underlying assumptions about the distribution of data. The PCs were first plotted together as in Figure 17 where 3 clusters were observed. In MATLAB, three random points were chosen as starting centroids. For each point in the data set, its distance from the centroid was calculated and assigned to the point it is closest to. The centroids were recalculated by taking the mean of the points in each cluster using Equation 7 and this was done in a while loop until the centroid did not change points. Figure 18 shows the first and second PCs plotted but each point is colored according to each cluster. The shadedErrorBar function was then used to plot the average spike for each cluster with an area showing the standard deviation as shown in Figure 19.

K-means was extended to k-medoids and using the Mahalanobis distance. The same iterative process in MATLAB for k-means was implemented in k-medians however, in every iteration the median was found to get the medoid and that was used to compare the distances of all the points. A graph using k-medoids is shown in Figure 20. The Mahlanobis distance is given by Equation 8. For the algorithm implementing the Mahalanobis distances Equation 9 shows the new objective function which is minimized. In the first iteration a regular k-means algorithm is used. After that the Mahalanobis distances were calculated (with MATLAB's mahal function) for each cluster. Other than that the algorithm was similar where each point was assigned to the smallest cluster but mahal was used to find distances. A graph using the Mahalanobis distances is shown in Figure 21.

Finally a Gaussian mixture model was used to cluster the points. In this model it is assumed that spikes within each cluster are normally distributed about the cluster centroid. This was implemented in MATLAB mostly with built-in functions. The data was first normalized, and the PCA function was used to get the first two PCs. Function options were defined with the function statset. 3 clusters were then chosen and the gmdistribuition.fit function was used. The cluster assignments were retrieved by the cluster function and the clusters, centroids, and contour plots around the centroids were plotted in Figure 22.

Results

In the first portion of this paper the difference between the total impedance of a fresh electrode and the electrode after 7 days was calculated. The magnitude of the total impedance for the fresh and 7 day electrode was found to be about 79,180 Ω and 253,160 Ω respectively.. These values make sense because after several days of the electrode being implanted scar tissue develops around the electrode. Glia scars can develop which increase the impedance of the electrode and make it more difficult to record signals from the area.

ECoG data was filtered to capture meaningful features. Channel 29 of the data was CAR and put through bandpass filters with different frequency ranges. It can be seen in Figure 2 that in the low frequency filter there is a spike at about 5000 samples. The medium frequency data

does not have any noticeable spikes and stay between abou 0 and 2 mV^2. The high frequency power stays around 11 mV^2 but has a little spike at 5000 samples as well. This means that at the time sample 5000 was taken the patient had movement or neuronal activity. This demonstrated how analysis with filtering of ECoG data usually must be done to get meaningful results and the low and high filters should be used when detecting activity from this ECoG position.

The next portion explored PCs and how they can be used to identify important features of neural signals and reconstruct original data. It was found that 9 PCs were necessary to capture 90% of the variance in data. The top 9 PCs were plotted for spike 5. Figure 3 shows the first PC, Figure 4 the second and so on. Figure 3, the graph with the first PC is fairly close to the original spike showing an initial rise, steep drop, rise higher than the beginning, and curve back down. The first PC should look most like the original spike because it represents the maximum variance direction in the data. Figure 4 looks less like the original but does have an initial rise, dip, rise, and dip again. Figure 5 is smoother and has a small rise and dip but it has a mostly accurate repolarization that shows a curved large hill. Figure 6 has the second large hill but is less smooth. Figure 7 shows that there should be 2 hills however the first one is much larger than the original. Figure 8 shows too many hills with 3. Figures 9, 10, and 11, show several jagged hills however they do not look very much like the original. The smaller the PC the less accurate the graph looked because it accounts for a smaller amount of the variance.

The original spike was then recreated with varying amounts of PCs. Figure 14 shows the spike with all the PCs compared to the original. The lines are identical since all the dimensions of variance are accounted for. Figure 15 shows a spike recreation with the first 6 PCs. The recreation is fairly accurate of the original spike. Figure 16 shows a spike recreation with the first 4 PCs. The recreation is less accurate than Figure 15 but is still fairly accurate. This shows that only the first first few PCs are necessary to estimate a recreation of the original spike. The less PCs used the less accurate the model will be, but even with 4 PCs the model is not far off.

Additionally, the first two PCs were plotted against each other in Figure 17. There are clusters of dots that can be organized into different spikes. These two show a large % of variability, more easily visualize data, and are easier to give to algorithms.

The k-means algorithm was then implemented to group the clusters seen in the plot of the two PCs. Figure 17 shows about 3 clusters so for this algorithm K = 3. Every time the k-means algorithm is run a different plot shows up because different initial random numbers are chosen and the plot converges to different points. Most of the graphs don't accurately cluster the data. Figure 18 shows a decent iteration where the data is split up into 3 clusters. Figure 19 shows the average results for each cluster plotted with the standard deviation. The lines and standard deviations overlap a lot which can contribute to why it is hard to differentiate the clusters. One of the major limitations of k- means is that you must initially say how many clusters there are. K-means will give the amount of clusters initialized so if the wrong number of clusters is initially given then an accurate output will not be given. K-means assumes spherical shapes of clusters and does not work well when clusters are in different shapes, since it mainly differentiates by the radius from the centroid. Additionally, k-means does not have an effective way of taking into

account that some clusters may be overlapping. Since k-means is mostly based on the mean, outliers can have a large impact on the mean and can distort the clustering results.

Another method looked at was k-medoids. K-medoids is also a cluster partitioning function that uses a similar technique as k-means but uses the medians to calculate the medoids instead of the means to calculate the centroids. The results are also different each run but one example is shown in Figure 20. An advantage of k-medoids is that outliers will not skew the data as much and distort the results. Another clustering technique investigated was doing k-means but using the Mahalanobis distance instead of the Euclidean distance. The Mahalanobis normalizes distances from each centroid k to the covariance of points within cluster k. The results of this algorithm are different every time but Figure 21 shows one result. The graphs with these iterations showed less "pie" graph-like partitions than k-means or k-medoids. With the Mahalanobis distance it allows for some flexibility in the shape of the clusters to not be circular and it takes into account variances and covariances amongst variables. One limitation of this method is that it may not converge so a maximum iteration limit may need to be included.

The final cluster algorithm used was the Gaussian mixture model (GMMs) . The GMM uses a probabilistic approach that assumes that spikes within each cluster are normally distributed about the cluster centroid. GMMs assign points to the most likely cluster instead of the nearest cluster like k-means. The GMM was implemented and is shown in Figure 22. The centroids are also plotted as 'X's with contour plots around them. There is no contour plot around the blue centroid because it has less points than the others and would not be as high Gaussian. Gaussian is usually more accurate than k-means but it is also usually slower and more computationally intensive. Additionally, a number of k clusters must be chosen for the GMM like k-means, making the initialization have a large effect on the result. It could be that the number of clusters given are not the number of neurons observed. For instance, if K = 5 like in Figure 23, then there are more clusters than neurons so post-processing would need to be done to combine clusters by changing K, visually validating and differentiating clusters or rerunning

Discussion

This paper analyzed tools to interpret experimentally acquired data from neurons. It was observed that electrodes after a couple days have a higher impedance which is important for understanding the effectiveness of electrodes to record. ECoG data was collected and filtered showing that frequencies between 2-6 Hz and 54-114 Hz are sufficient for displaying neuron activity and possibly can be used as control signals in devices. PCA was also shown as a way to reduce the dimensionality of spike recordings to represent the data easier, describe the data with fewer components, and is easier to give to algorithms to differentiate spike clusters.

Various spike sorting algorithms including k-means, k-medoid, Mahalanobis distance, and GMM were created to group signals and assign them to their source neurons. Sometimes these algorithms require post-processing because of overlapping spikes which cause the algorithms to produce inaccurate results. This can mean changing the K value, rerunning the algorithm, or running the algorithm in smaller chunks of the data. Knowing the source neuron of spikes can help characterize the data and predict movement based on signals and activation.

Appendix

Figures

Figure 1. Nyquist plots of an electrode initial implemented (blue) and the electrode 7 days later (red). Complex impedance spectroscopy for monitoring tissue responses to inserted neural implants' by Justin C Williams et al. (2007)

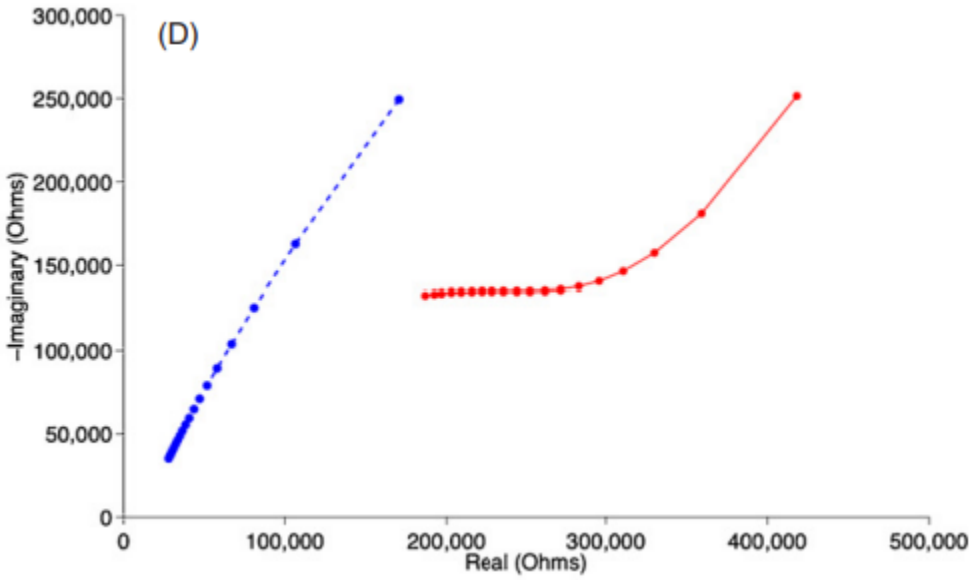


Figure 2. Power of filtered, common average reference, channel 29 ECoG data. Displays low, medium and high filters of channel.

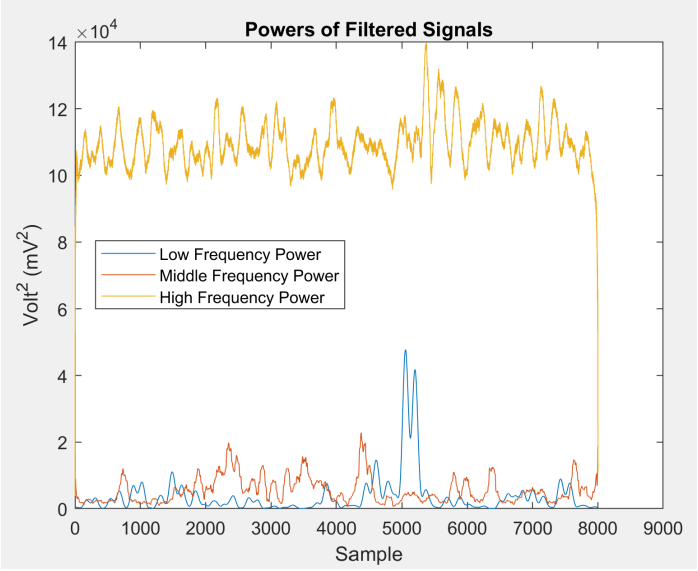


Figure 3. First principal component of spike 5.

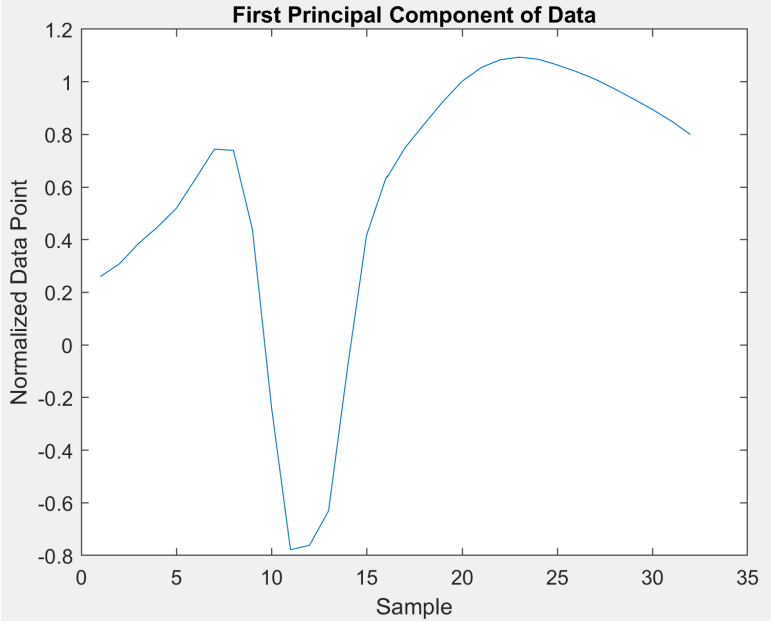


Figure 4. Second principal component of spike 5.

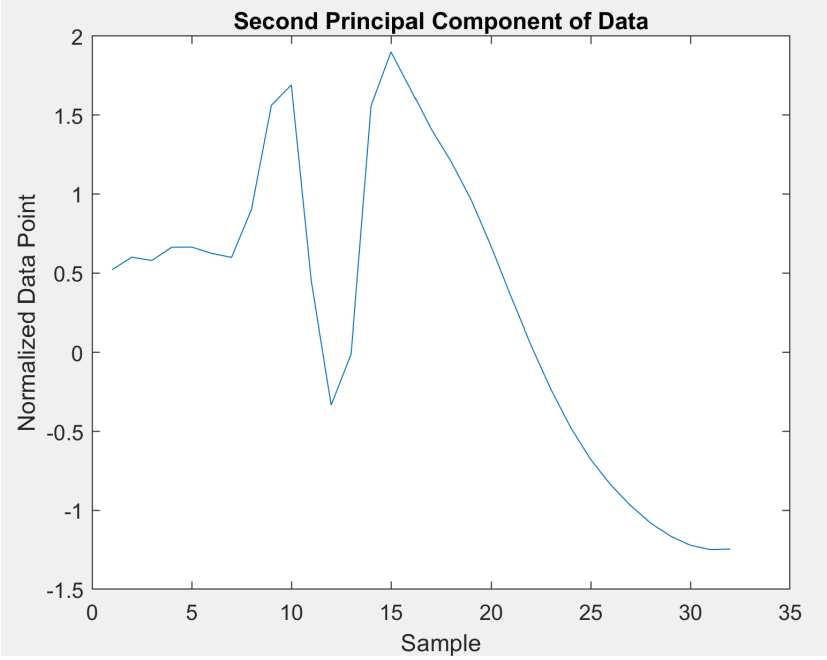


Figure 5. Third principal component of spike 5.

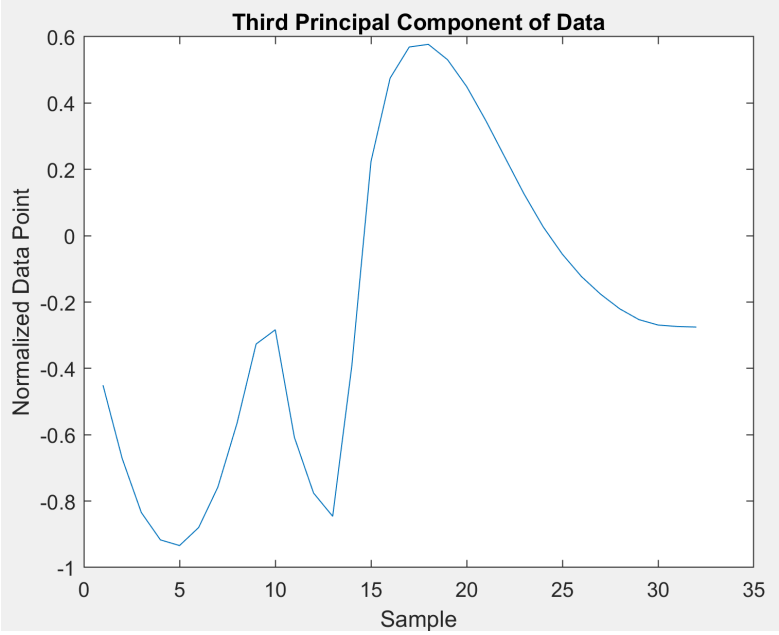


Figure 6. Fourth principal component of spike 5.

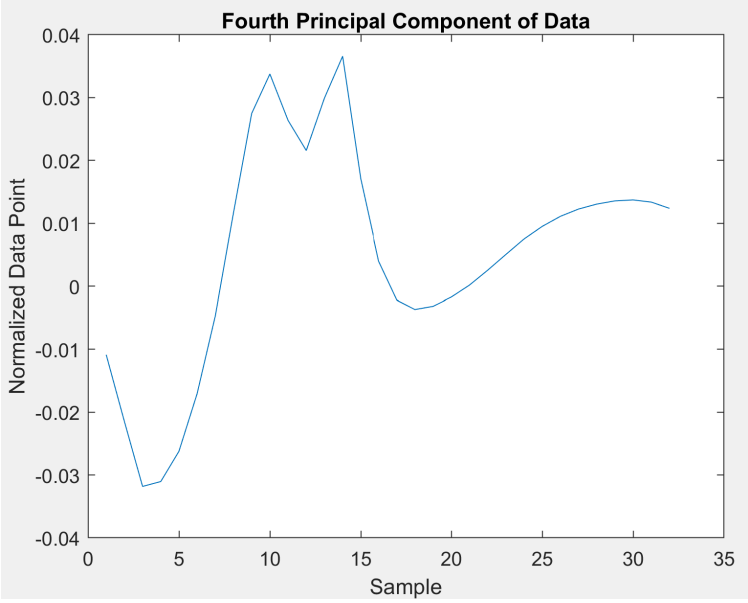


Figure 7. Fifth principal component of spike 5.

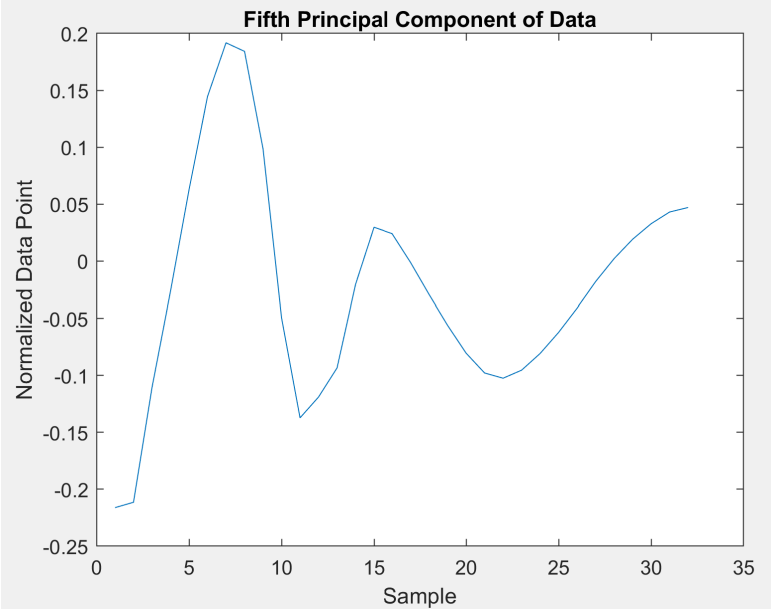


Figure 8. Sixth principal component of spike 5.

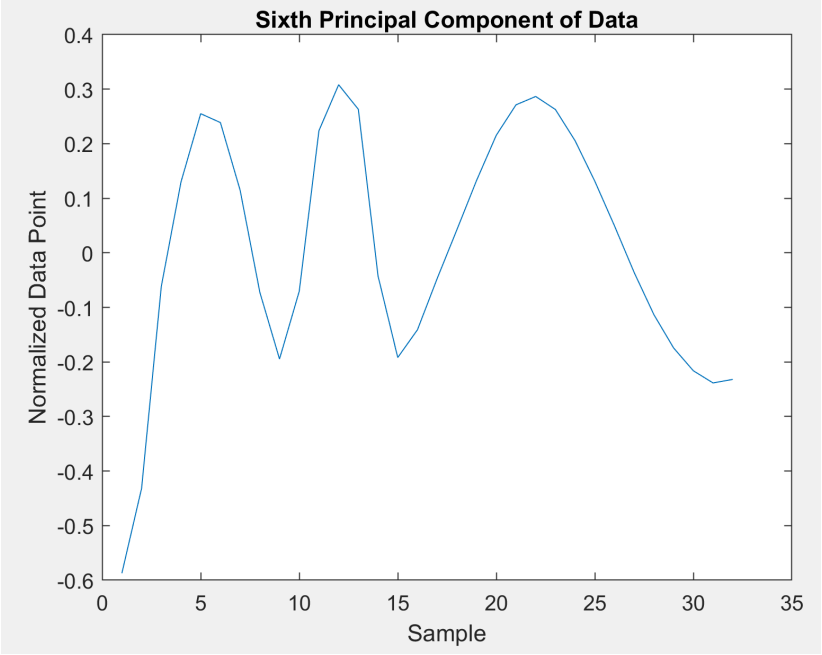


Figure 9. Seventh principal component of spike 5.

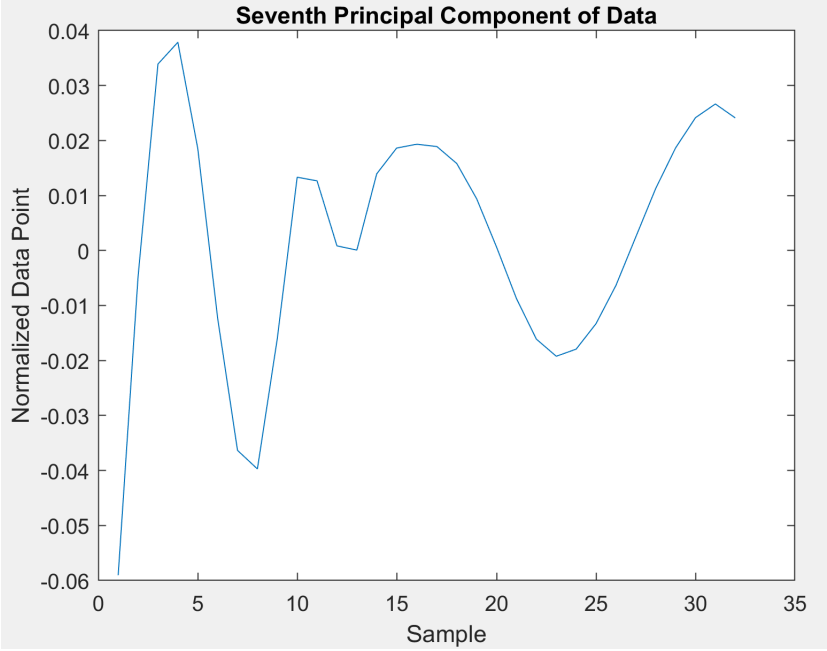


Figure 10. Eighth principal component of spike 5.

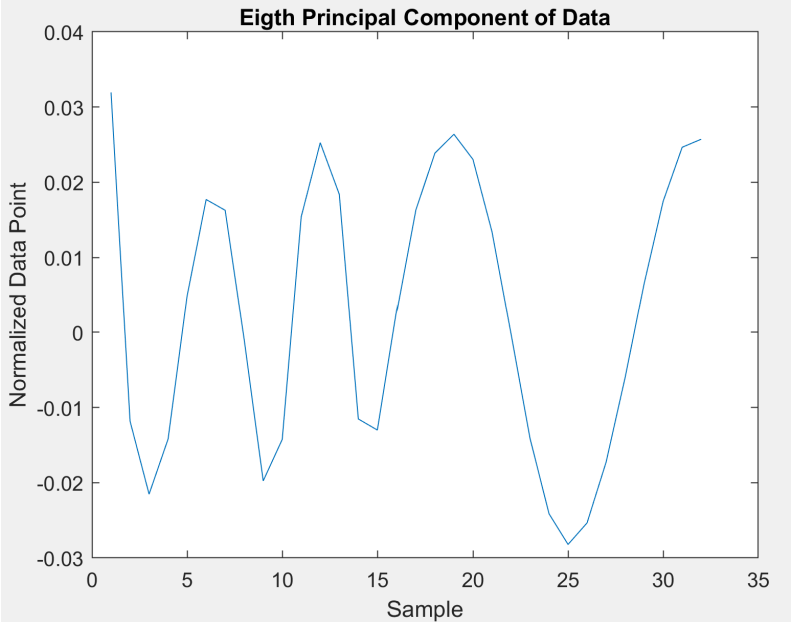


Figure 11. Ninth principal component of spike 5.

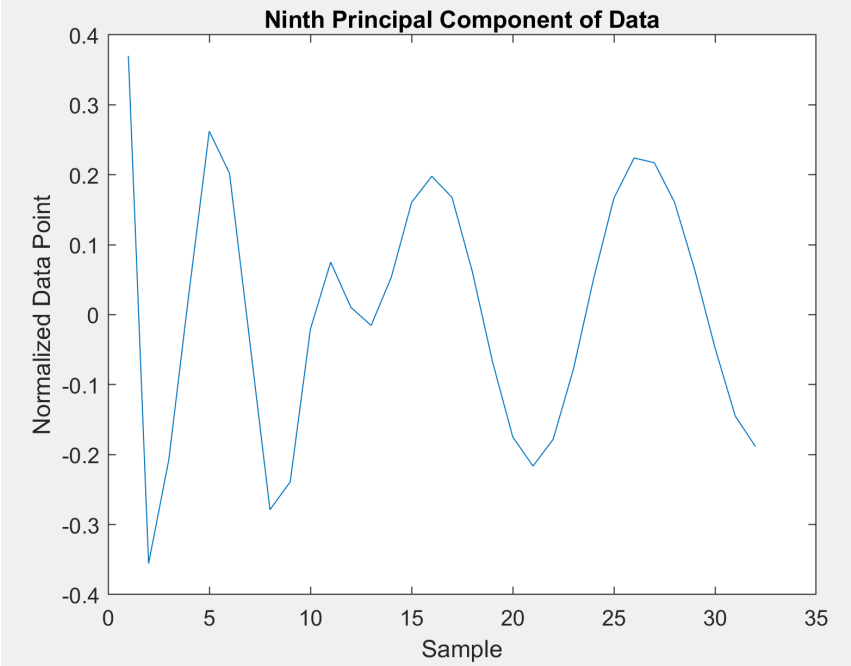


Figure 12. Original normalized spike 5 data.

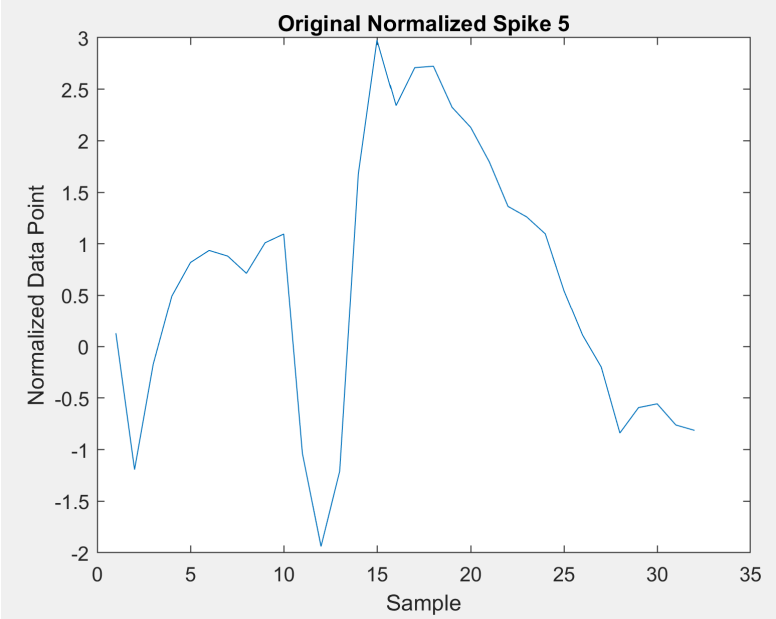


Figure 13. Original Spike 5.

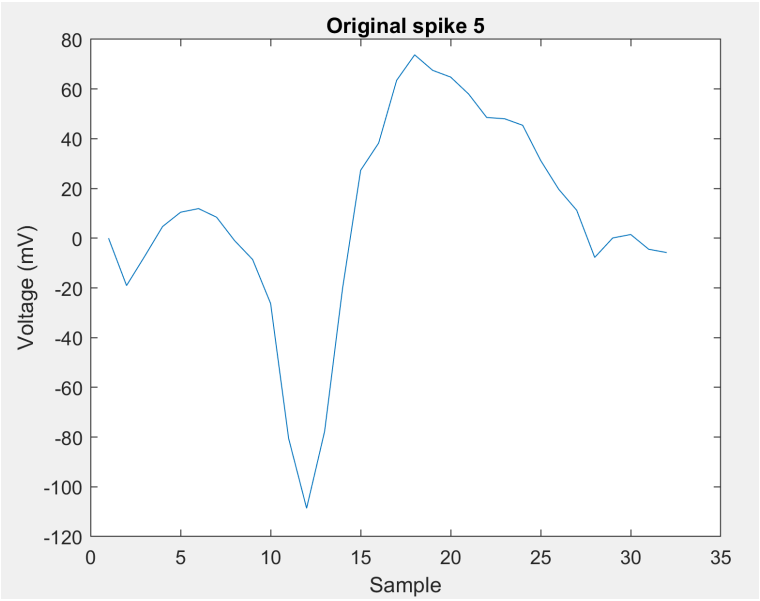


Figure 14. Original spike 5 plotted with the reconstructed spike 5 using all PC show exact same graphs.

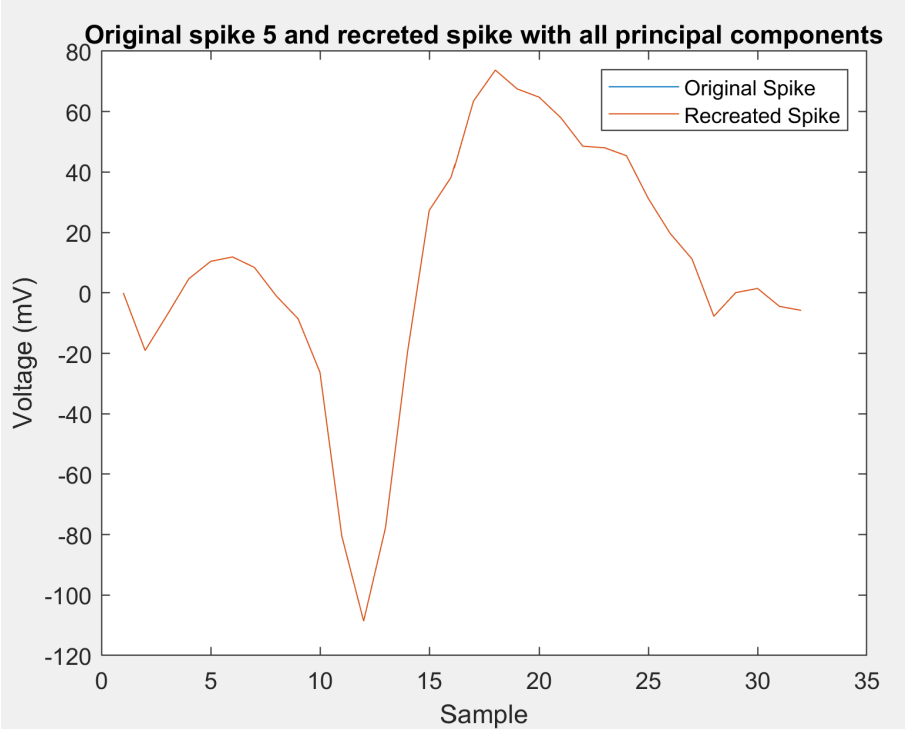


Figure 15. Graph of original spike 5 and reconstructed spike with 6 PC shows close match.

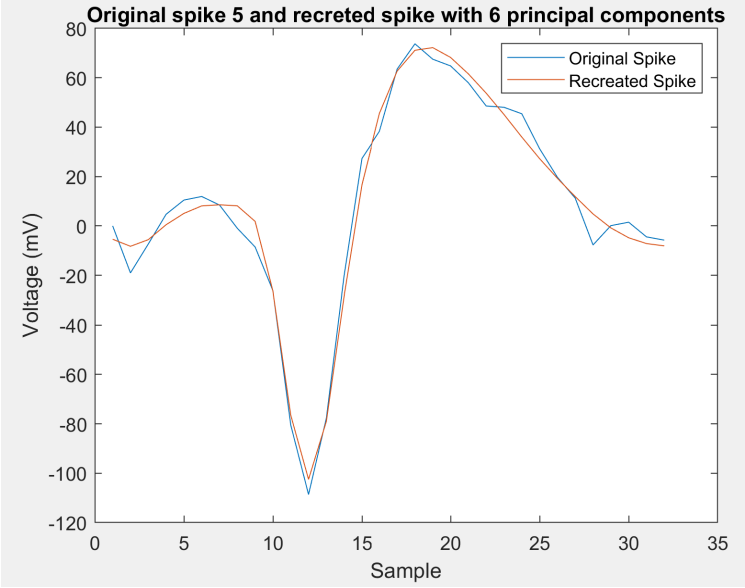


Figure 16.Graph of original spike 5 and reconstructed spike with 4 PC are very similar.

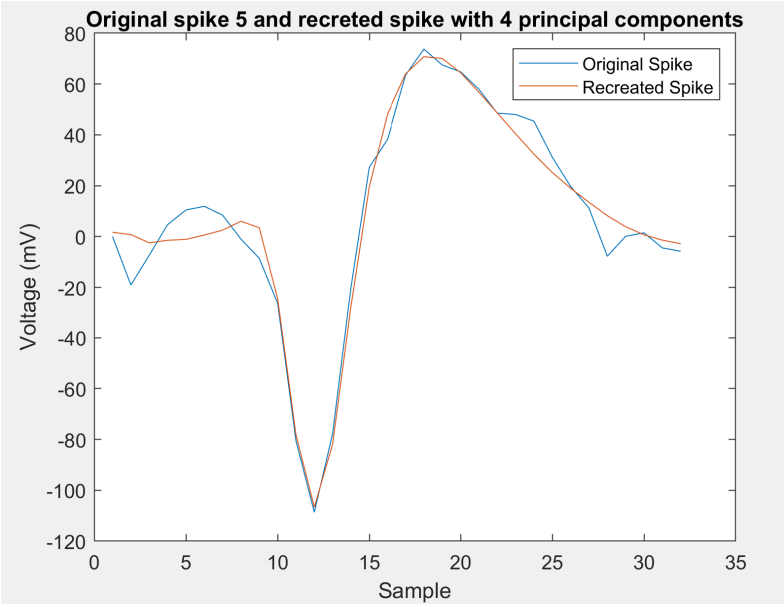


Figure 17. Principal component 1 vs principal component 2 shows data clusters.

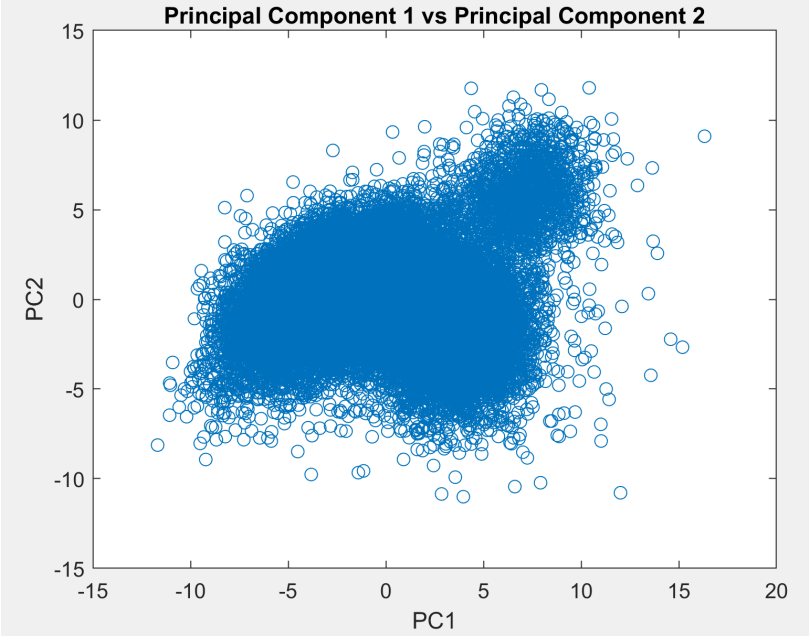


Figure 18. PC1 vs PC2 plot using K-means algorithm to separate into 3 clusters.

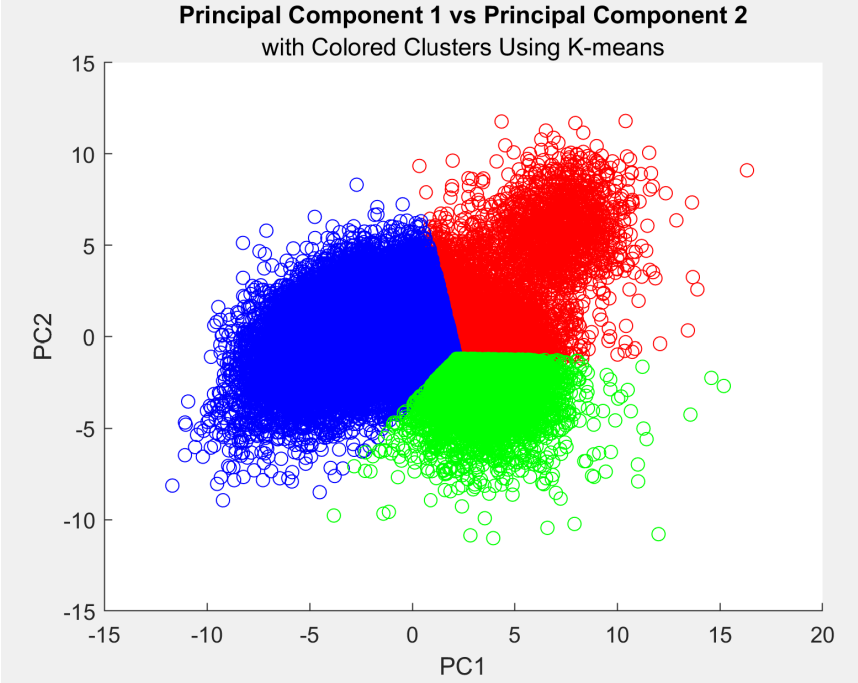


Figure 19. Average spike for each cluster with standard deviation show overlapping standard deviation.

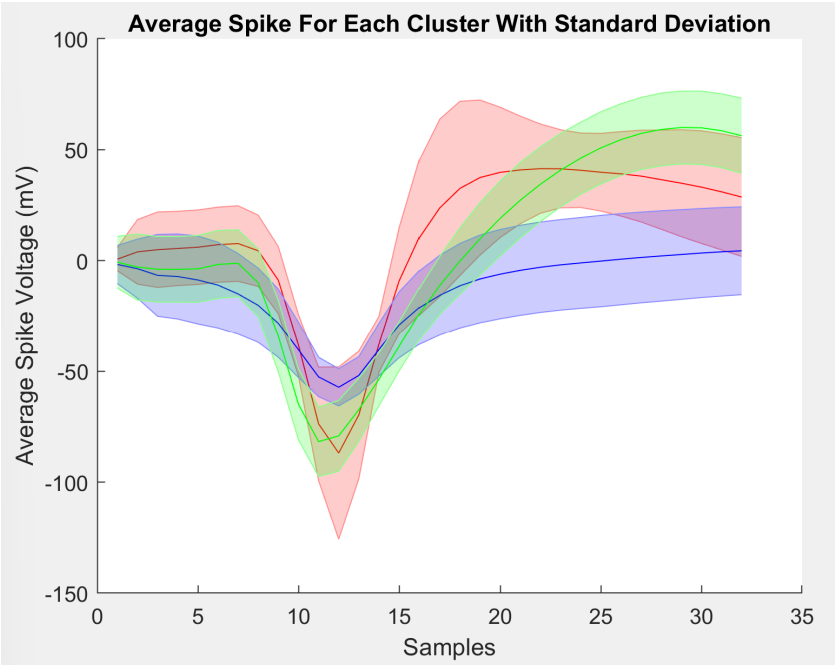


Figure 20. PC1 vs PC2 plot using K-mediods algorithm to separate into 3 clusters.

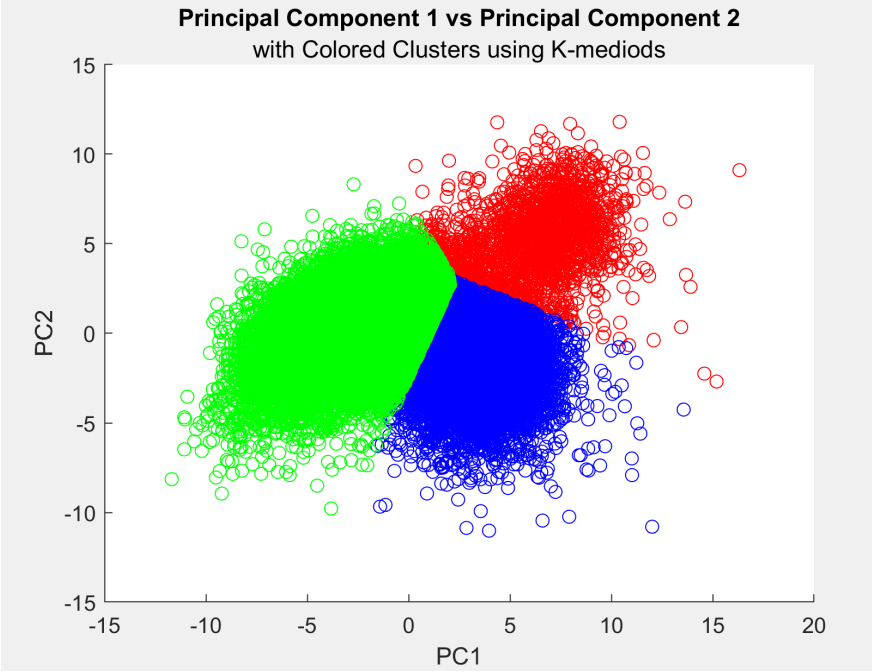


Figure 21. PC1 vs PC2 plot using mahalanobis algorithm to separate into 3 clusters.

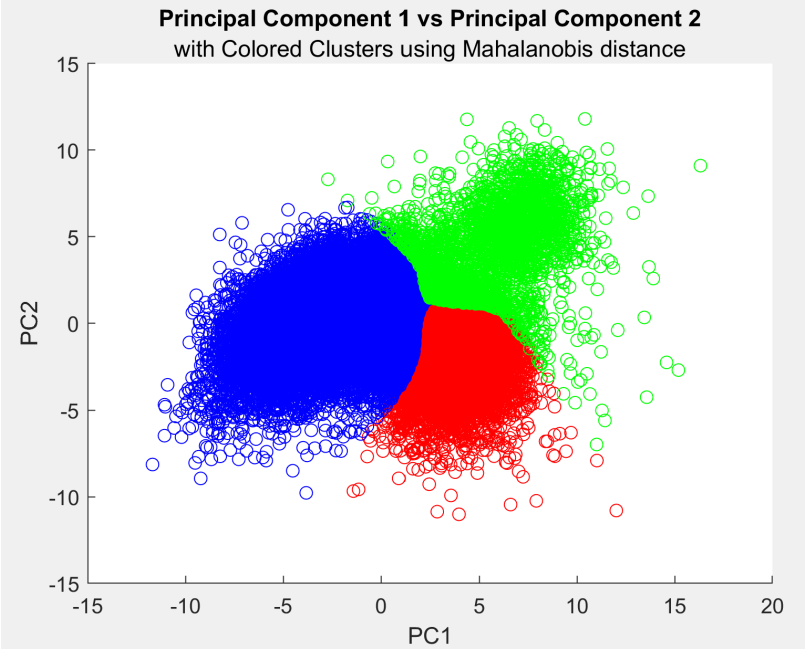


Figure 22. PC1 vs PC2 plot using GMM algorithm to separate into 3 clusters. There are also the centroids labeled with x and contour plots around the centroids.

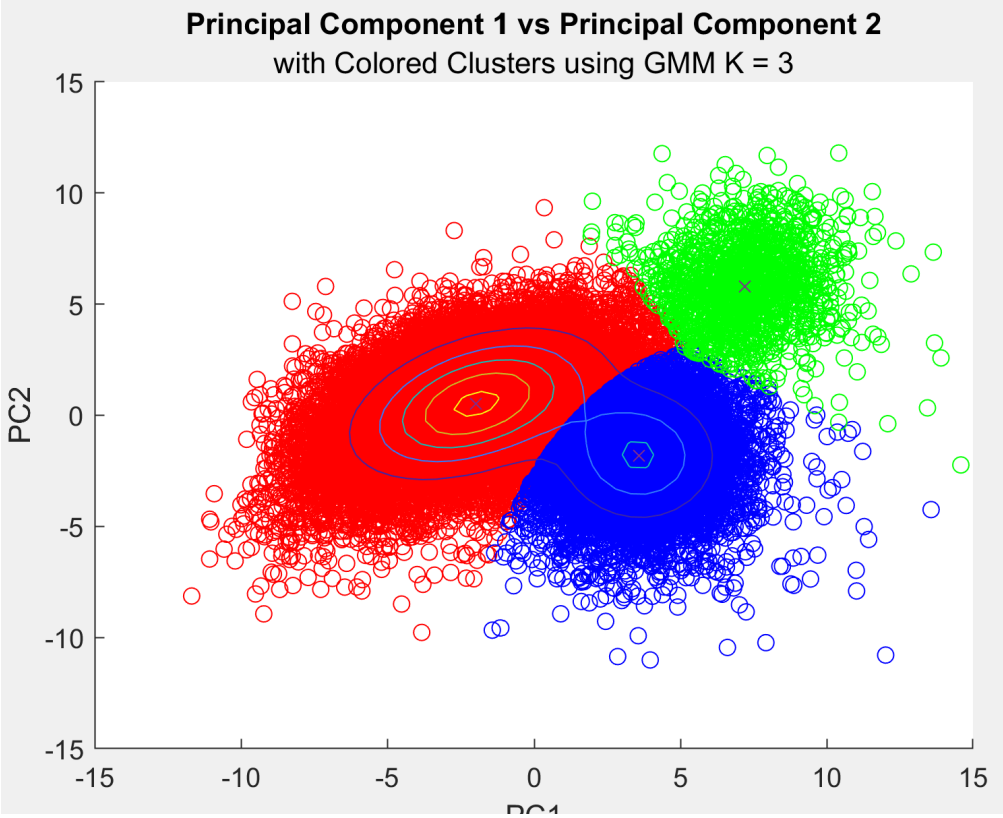
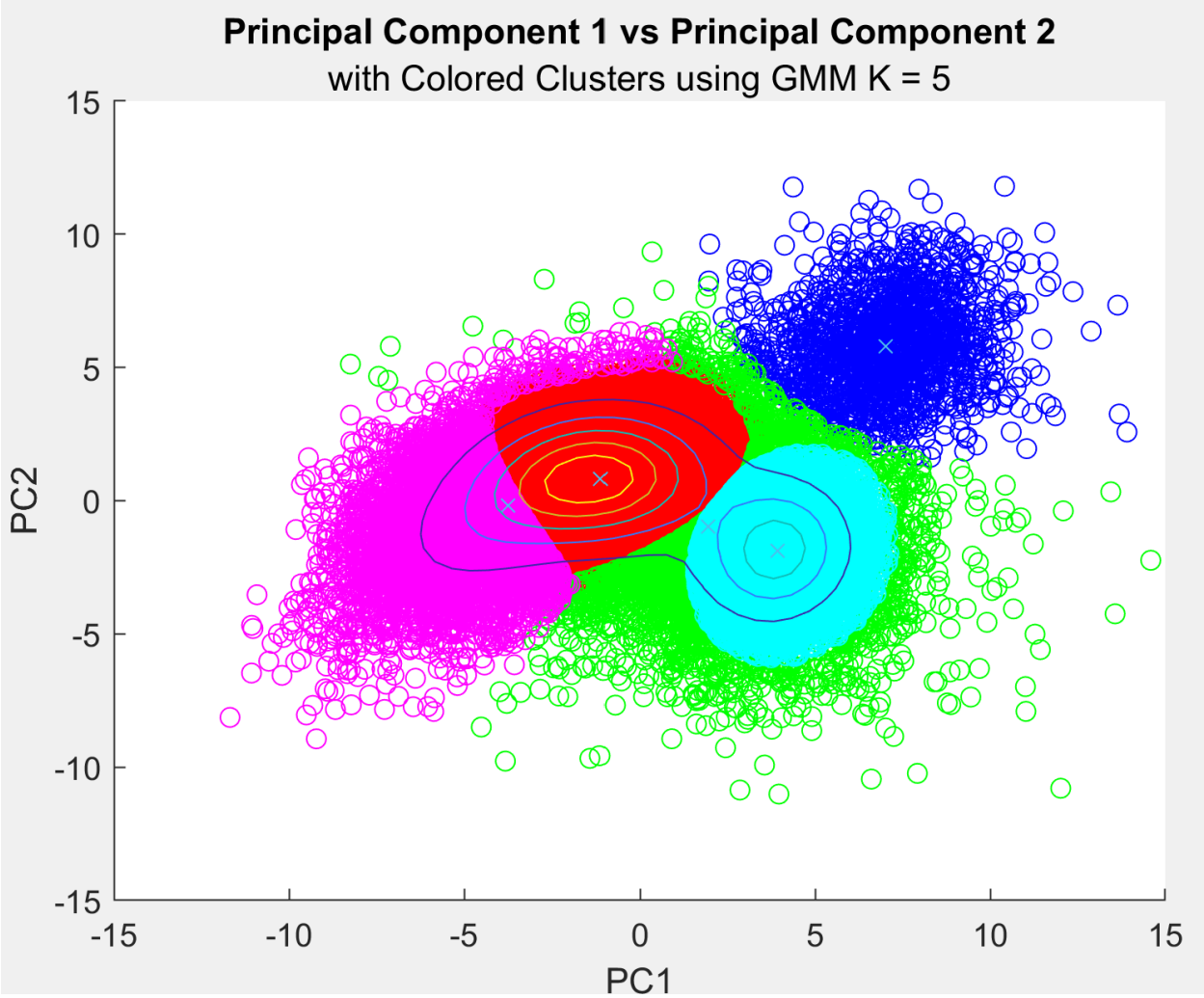


Figure 23.PC1 vs PC2 plot using GMM algorithm to separate into 5 clusters. There are also the centroids labeled with x and contour plots around the centroids.



Equations

Equation 1

$$\alpha = \frac{2}{\pi}\arctan(slope)$$

Equation 2

$$K = (Z_{re}(\omega_0) - jZ_{im}(\omega_0) - (R_{en} + R_{ex})) \cdot (j\omega)^{\alpha}$$

Equation 3

$$Z_{Tot} = R + \frac{K}{(j\omega)^{\alpha}}$$

Equation 4

$$\frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{D} \lambda_i}$$

Equation 5

$$\vec{x}_i \cong \sum_{j}^{k} w_{ij} \vec{u}_j$$

Equation 6

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left|\left|\vec{x}_n - \vec{\mu}_k\right|\right|^2, \quad where\ r_{nk} = \begin{cases} 1, & if\ x_n\ is\ in\ group\ k \\ 0, & if\ x_n\ is\ not\ in\ group\ k \end{cases}$$

Equation 7

$$\vec{\mu}_k = \frac{1}{N_k} \sum_{n}^{N_k} \vec{x}_n$$

Equation 8

$$d_M = \sqrt{(x - \mu_k)^T S_k^{-1} (x - \mu_k)}$$

Equation 9

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} d_M^2, \quad where\ r_{nk} = \begin{cases} 1, & if\ x_n\ is\ in\ group\ k \\ 0, & if\ x_n\ is\ not\ in\ group\ k \end{cases}$$

Code

Lab 5 Part 1

```
% Blue line = 20,000 Imaginary Ohms or 0 real Ohms
% Red line = 205,000 Real Ohms
ZredReal = 410000;
ZredIm = 250000;
RenRexRed = 205000;
alphaRed = .637;
Kred = (ZredReal - ZredIm*j - RenRexRed)*(1j*2*pi*100)^(alphaRed);
KredMag = abs(Kred);
ZredTot = RenRexRed + KredMag/((j*2*pi*1000)^alphaRed);
ZredTotAbs = abs(ZredTot)

ZblueReal = 180000;
ZblueIm = 250000;
RenRexBlue = 0;
alphaBlue = .590;
Kblue = (ZblueReal - ZblueIm*j - RenRexBlue)*(1j*2*pi*100)^(alphaBlue);
KblueMag = abs(Kblue);
ZblueTot = RenRexBlue + KblueMag/((j*2*pi*1000)^alphaBlue);
ZblueTotAbs = abs(ZblueTot)
```

Lab 5 Part 2

```
ecogData = load('ecogdatasnippet.mat');

index = find(ecogData.refChannels == 1);
carSignal = mean(ecogData.ecogData(index,:));
channel29Mean = zeros(1,8003);
for i = 1:8003
   channel29Mean(i) = ecogData.ecogData(29,i) - mean(carSignal);
end

low = bandpass(channel29Mean, [2 6], 1000);
middle = bandpass(channel29Mean, [14 46], 1000);
high = bandpass(channel29Mean, [54 114], 1000);

figure(1)
plot(channel29Mean)
```

```
hold on
plot(low)
legend("Channel 29 CAR ","Channel 29 CAR low filter")
title("Channel 29 CAR orginal and with low filter")
xlabel("Sample")
ylabel("Volage (mV)")
hold off

figure(2)
plot(channel29Mean)
hold on
plot(middle)
legend("Channel 29 CAR ","Channel 29 CAR middle filter")
title("Channel 29 CAR orginal and with middle filter")
xlabel("Sample")
ylabel("Volage (mV)")
hold off

figure(3)
plot(channel29Mean)
hold on
plot(high)
legend("Channel 29 CAR ","Channel 29 CAR high filter")
title("Channel 29 CAR orginal and with high filter")
xlabel("Sample")
ylabel("Volage (mV)")
hold off

lowPower = zeros(1,8003);
middlePower = zeros(1,8003);
highPower = zeros(1,8003);

for i=1:8003
   lowPower(i) = low(i).^2;
   middlePower(i) = middle(i).^2;
   highPower(i) = high(i).^2;
end

figure(4)
plot(smooth(lowPower, 100))
```

```
hold on
plot(smooth(middlePower, 100))
plot(smooth(highPower,100))
title("Powers of ")
xlabel("Sample")
ylabel("Volt^2 (mV^2)")
legend("Low Frequency Power", "Middle Frequency Power", "High Frequency Power")
```

Lab 5 Part 3

```
data = load('spikes.mat');
meanData = mean(data.spikes, 1);
standardD = std(data.spikes);
normalize = (data.spikes - meanData)./ standardD;

[coeff, score, latent, tsquared, explained, mu] = pca(normalize);
%idx = find(cumsum(explained)> .90, 1);
result = cumsum(latent)./sum(latent);
k_90 = find(result == min(result(result > .90)));
%cumsum(latent)
%sum(latent)
%result
result == min(result(result > .90));

%{
figure(1)
plot(score(5,1)*coeff(:,1))
title("First Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(2)
plot(score(5,2)*coeff(:,2))
title("Second Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(3)
plot(score(5,3)*coeff(:,3))
```

```
title("Third Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(4)
plot(score(5,4)*coeff(:,4))
title("Fourth Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(5)
plot(score(5,5)*coeff(:,5))
title("Fifth Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(6)
plot(score(5,6)*coeff(:,6))
title("Sixth Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(7)
plot(score(5,7)*coeff(:,7))
title("Seventh Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(8)
plot(score(5,8)*coeff(:,8))
title("Eigth Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")

figure(9)
plot(score(5,9)*coeff(:,9))
title("Ninth Principal Component of Data")
xlabel("Sample")
ylabel("Normalized Data Point")
```

```
figure(10)
plot(normalize(5,:))
title("Original Normalized Spike 5")
xlabel("Sample")
ylabel("Normalized Data Point")
hold on

figure(11)
plot(data.spikes(5,:))
title("Original  Spike 5")
xlabel("Sample")
ylabel("Voltage")
hold on
%}

sp1_rec = score(5,:)*coeff(:,:)';
%plot(sp1_rec);
hold off
%figure(11)
unnormal = sp1_rec .*standardD + meanData;
plot(data.spikes(5,:))
hold on
plot(unnormal)
title("Original spike 5 and recreted spike with all principal components")
xlabel("Sample")
ylabel("Voltage (mV)")
legend("Original Spike", "Recreated Spike")
hold off

figure(12)
plot(score(:,1),score(:,2),'o')
title("Principal Component 1 vs Principal Component 2")
xlabel("PC1")
ylabel("PC2")
```

Lab 6 - Part 1

```
data = load('spikes.mat');
meanData = mean(data.spikes, 1);
```

```
standardD = std(data.spikes);
normalize = (data.spikes - meanData)./ standardD;

[coeff,score] = pca(normalize);

first = score(:,1);
second = score(:,2);
size(first);
%figure(1)
%plot(first, second,'o')

k = 3;
r1 = floor(1 + (41568-1)*rand());
r1x = first(r1)
r1y = second(r1)
r2 = floor(1 + (41568-1)*rand());
while r1 == 2
   r2 = floor(1 + (41568-1)*rand());
end
r2x = first(r2);
r2y = second(r2);
r3 = floor(1 + (41568-1)*rand());
while r3 == r1 || r3 == r2
   r3 = floor(1 + (41568-1)*rand());
end
r3x = first(r3);
r3y = second(r3);
distance = zeros(1, 41568);
cluster = zeros(1, 41568);

r1new = 0;
r1newx = 0;
r1newy = 0;
r2new = 0;
r2newx = 0;
r2newy = 0;
r3new = 0;
r3newx = 0;
r3newy = 0;
```

```
while (r1newx ~= r1x) && (r1newy ~= r1y) && (r2newx ~= r2x) && (r2newy ~= r2y) &&
(r3newx ~= r3x) && (r3newy ~= r3y)
   r1newx = r1x;
   r1newy = r1y;
   r2newx = r2x;
   r2newy = r2y;
   r3newx = r3x;
   r3newy = r3y;
   for i = 1:41568
    distance1 = (r1newx - first(i))^2 + (r1newy - second(i))^2;
    distance2 = (r2newx - first(i))^2 + (r2newy - second(i))^2;
    distance3 = (r3newx - first(i))^2 + (r3newy - second(i))^2;
    if(distance1 < distance2 && distance1 < distance3)
        cluster(i) = 1;
        distance(i) = distance1;
    end
    if(distance2 < distance1 && distance2 < distance3)
        cluster(i) = 2;
        distance(i) = distance2;
    end
    if(distance3 < distance1 && distance3 < distance2)
        cluster(i) = 3;
        distance(i) = distance3;
    end
   end
   mean1x = 0;
   mean1y = 0;
   mean2x = 0;
   mean2y = 0;
   mean3x = 0;
   mean3y = 0;
   count1 = 0;
   count2 = 0;
   count3 = 0;
   for k = 1:41568
     if cluster(k) == 1
        mean1x = mean1x + first(k);
        mean1y = mean1y + second(k);
        count1 = count1 + 1;
     end
```

```
    if cluster(k) == 2
      mean2x = mean2x + first(k);
      mean2y = mean2y + second(k);
      count2 = count2 + 1;
    end
      mean3x = mean3x + first(k);
      mean3y = mean3y + second(k);
      count3 = count3 + 1;
  end
  %disp("r1 end of while loop")
  %mean1
  %count1
  r1x = mean1x/count1;
  r1y = mean1y/count1;
  r2x = mean2x/count2;
  r2y = mean2y/count2;
  r3x = mean3x/count3;
  r3y = mean3y/count3;
end
disp("Out of while loop")
%{
cluster(3)
figure(2)
scatter(first(1), second(1),"red")
hold on
scatter(first(3), second(3),"green")
%}

redElements = find(cluster == 1);
greenElements = find(cluster == 2);
blueElements = find(cluster == 3);
figure(1)
scatter(first(redElements), second(redElements), 'red')
hold on
scatter(first(blueElements), second(blueElements), 'blue')
scatter(first(greenElements), second(greenElements), 'green')
title("Principal Component 1 vs Principal Component 2", "with Colored Clusters Using
K-means")
xlabel("PC1")
ylabel("PC2")
```

```
hold off

time = 1:32;
%figure(2)
%size(mean(data.spikes(redElements,:)))
%hold on
redMean = mean(data.spikes(redElements,:));
redStd = std(data.spikes(redElements,:));
blueMean = mean(data.spikes(blueElements,:));
blueStd = std(data.spikes(blueElements,:));
greenMean = mean(data.spikes(greenElements,:));
greenStd = std(data.spikes(greenElements,:));
disp("Green Stuff")
size(greenMean)
size(greenStd)
%plot(time, redMean, 'red')
%plot(time, blueMean, 'blue')
%plot(time, greenMean, 'green')
%legend('Red','Blue','Green');
%hold off

figure(3)
shadedErrorBar(time,redMean,redStd, 'lineProps', 'r');
hold on
shadedErrorBar(time,blueMean,blueStd,'lineProps', 'b');
shadedErrorBar(time,greenMean,greenStd,'lineProps', 'g');
xlabel("Samples")
ylabel("Average Spike Voltage (mV)")
title("Average Spike For Each Cluster With Standard Deviation")


%red1Mean = mean(data.spikes(redElements));
%redstd = std(data.spikes(redElements));
%size(red1Mean)
%x = red1Mean
%y = redstd;
%time = 1:32;
%plot(time, x)
%shadedErrorBar(x,y,'lineProps','g');
```

Lab 6 Part 2.1

```
data = load('spikes.mat');
meanData = mean(data.spikes, 1);
standardD = std(data.spikes);
normalize = (data.spikes - meanData)./ standardD;

[coeff,score] = pca(normalize);

first = score(:,1);
second = score(:,2);
size(first);
%figure(1)
%plot(first, second,'o')

k = 3;
r1 = floor(1 + (41568-1)*rand());
r1x = first(r1)
r1y = second(r1)
r2 = floor(1 + (41568-1)*rand());
while r1 == 2
    r2 = floor(1 + (41568-1)*rand());
end
r2x = first(r2);
r2y = second(r2);
r3 = floor(1 + (41568-1)*rand());
while r3 == r1 || r3 == r2
    r3 = floor(1 + (41568-1)*rand());
end
r3x = first(r3);
r3y = second(r3);
distance = zeros(1, 41568);
cluster = zeros(1, 41568);

r1new = 0;
r1newx = 0;
r1newy = 0;
r2new = 0;
r2newx = 0;
r2newy = 0;
```

```
r3new = 0;
r3newx = 0;
r3newy = 0;

while (r1newx ~= r1x) && (r1newy ~= r1y) && (r2newx ~= r2x) && (r2newy ~= r2y) &&
(r3newx ~= r3x) && (r3newy ~= r3y)
   r1newx = r1x;
   r1newy = r1y;
   r2newx = r2x;
   r2newy = r2y;
   r3newx = r3x;
   r3newy = r3y;
   for i = 1:41568
    distance1 = (r1newx - first(i))^2 + (r1newy - second(i))^2;
    distance2 = (r2newx - first(i))^2 + (r2newy - second(i))^2;
    distance3 = (r3newx - first(i))^2 + (r3newy - second(i))^2;
    if(distance1 < distance2 && distance1 < distance3)
        cluster(i) = 1;
        distance(i) = distance1;
    end
    if(distance2 < distance1 && distance2 < distance3)
        cluster(i) = 2;
        distance(i) = distance2;
    end
    if(distance3 < distance1 && distance3 < distance2)
        cluster(i) = 3;
        distance(i) = distance3;
    end
   end
   r1x = median(first(find(cluster == 1)));
   r1y = median(second(find(cluster == 1)));
   r2x = median(first(find(cluster == 2)));
   r2y = median(second(find(cluster == 2)));
   r3x = median(first(find(cluster == 3)));
   r3y = median(second(find(cluster == 3)));
end
disp("Out of while loop")


redElements = find(cluster == 1);
```

```
greenElements = find(cluster == 2);
blueElements = find(cluster == 3);
figure(2)
scatter(first(redElements), second(redElements), 'red')
hold on
scatter(first(blueElements), second(blueElements), 'blue')
scatter(first(greenElements), second(greenElements), 'green')
title("Principal Component 1 vs Principal Component 2", "with Colored Clusters using
K-mediods")
xlabel("PC1")
ylabel("PC2")
hold off

time = 1:32;
%figure(3)
%size(mean(data.spikes(redElements,:)))
hold on
redMean = mean(data.spikes(redElements,:));
redStd = std(data.spikes(redElements,:));
blueMean = mean(data.spikes(blueElements,:));
blueStd = std(data.spikes(blueElements,:));
greenMean = mean(data.spikes(greenElements,:));
greenStd = std(data.spikes(greenElements,:));
%{
disp("Green Stuff")

%size(greenMean)
%size(greenStd)
%plot(time, redMean, 'red')
%plot(time, blueMean, 'blue')
%plot(time, greenMean, 'green')
legend('Red','Blue','Green');
hold off

figure(4)
shadedErrorBar(time,redMean,redStd, 'lineProps', 'r');
hold on
shadedErrorBar(time,blueMean,blueStd,'lineProps', 'b');
shadedErrorBar(time,greenMean,greenStd,'lineProps', 'g');
%}
```

```
%red1Mean = mean(data.spikes(redElements));
%redstd = std(data.spikes(redElements));
%size(red1Mean)
%x = red1Mean
%y = redstd;
%time = 1:32;
%plot(time, x)
%shadedErrorBar(x,y,'lineProps','g');
```

Lab 6 Part 2.2

```
data = load('spikes.mat');
meanData = mean(data.spikes, 1);
standardD = std(data.spikes);
normalize = (data.spikes - meanData)./ standardD;

[coeff,score] = pca(normalize);

first = score(:,1);
second = score(:,2);
size(first);
figure(1)
plot(first, second,'o')

k = 3;
r1 = floor(1 + (41568-1)*rand());
r1x = first(r1)
r1y = second(r1)
r2 = floor(1 + (41568-1)*rand());
while r1 == 2
   r2 = floor(1 + (41568-1)*rand());
end
r2x = first(r2);
r2y = second(r2);
r3 = floor(1 + (41568-1)*rand());
while r3 == r1 || r3 == r2
   r3 = floor(1 + (41568-1)*rand());
end
```

```
r3x = first(r3);
r3y = second(r3);
distance = zeros(1, 41568);
cluster = zeros(1, 41568);

r1new = 0;
r1newx = 0;
r1newy = 0;
r2new = 0;
r2newx = 0;
r2newy = 0;
r3new = 0;
r3newx = 0;
r3newy = 0;
firstLoop = 0;
numIterations = 0;

while (r1newx ~= r1x) && (r1newy ~= r1y) && (r2newx ~= r2x) && (r2newy ~= r2y) &&
(r3newx ~= r3x) && (r3newy ~= r3y) && (numIterations < 5)
  numIterations = numIterations + 1;


  r1newx = r1x;
  r1newy = r1y;
  r2newx = r2x;
  r2newy = r2y;
  r3newx = r3x;
  r3newy = r3y;
  if(firstLoop == 0)
    for i = 1:41568
     distance1 = (r1newx - first(i))^2 + (r1newy - second(i))^2;
     distance2 = (r2newx - first(i))^2 + (r2newy - second(i))^2;
     distance3 = (r3newx - first(i))^2 + (r3newy - second(i))^2;
     if(distance1 < distance2 && distance1 < distance3)
       cluster(i) = 1;
       distance(i) = distance1;
     end
     if(distance2 < distance1 && distance2 < distance3)
       cluster(i) = 2;
       distance(i) = distance2;
```

```
  end
  if(distance3 < distance1 && distance3 < distance2)
    cluster(i) = 3;
    distance(i) = distance3;
  end
end

mean1x = 0;
mean1y = 0;
mean2x = 0;
mean2y = 0;
mean3x = 0;
mean3y = 0;
count1 = 0;
count2 = 0;
count3 = 0;
for k = 1:41568
  if cluster(k) == 1
    mean1x = mean1x + first(k);
    mean1y = mean1y + second(k);
    count1 = count1 + 1;
  end
  if cluster(k) == 2
    mean2x = mean2x + first(k);
    mean2y = mean2y + second(k);
    count2 = count2 + 1;
  end
    mean3x = mean3x + first(k);
    mean3y = mean3y + second(k);
    count3 = count3 + 1;
end

r1x = mean1x/count1;
r1y = mean1y/count1;
r2x = mean2x/count2;
r2y = mean2y/count2;
r3x = mean3x/count3;
r3y = mean3y/count3;
firstLoop = 1;
else
```

```
firstIndex = find(cluster == 1);
secondIndex = find(cluster == 2);
thirdIndex = find(cluster == 3);
dM1x = mahal(first, first(firstIndex));
dM1y = mahal(second, second(firstIndex));

dM2x = mahal(first, first(secondIndex));
dM2y = mahal(second, second(secondIndex));

dM3x = mahal(first, first(thirdIndex));
dM3y = mahal(second, second(thirdIndex));
for i = 1:41568
  distance1 = sqrt(dM1x(i)^2 + dM1y(i)^2);
  distance2 = sqrt(dM2x(i)^2 + dM2y(i)^2);
  distance3 = sqrt(dM3x(i)^2 + dM3y(i)^2);
  if(distance1 < distance2 && distance1 < distance3)
    cluster(i) = 1;
    distance(i) = distance1;
  end
  if(distance2 < distance1 && distance2 < distance3)
    cluster(i) = 2;
    distance(i) = distance2;
  end
  if(distance3 < distance1 && distance3 < distance2)
    cluster(i) = 3;
    distance(i) = distance3;
  end
end
for k = 1:41568
  if cluster(k) == 1
    mean1x = mean1x + first(k);
    mean1y = mean1y + second(k);
    count1 = count1 + 1;
  end
  if cluster(k) == 2
    mean2x = mean2x + first(k);
    mean2y = mean2y + second(k);
```

```
      count2 = count2 + 1;
    end
      mean3x = mean3x + first(k);
      mean3y = mean3y + second(k);
      count3 = count3 + 1;
  end

  r1x = mean1x/count1;
  r1y = mean1y/count1;
  r2x = mean2x/count2;
  r2y = mean2y/count2;
  r3x = mean3x/count3;
  r3y = mean3y/count3;

 end

end
disp("Out of while loop")
%{
cluster(3)
figure(2)
scatter(first(1), second(1),"red")
hold on
scatter(first(3), second(3),"green")
%}

redElements = find(cluster == 1);
greenElements = find(cluster == 2);
blueElements = find(cluster == 3);
figure(2)
scatter(first(redElements), second(redElements), 'red')
hold on
scatter(first(blueElements), second(blueElements), 'blue')
scatter(first(greenElements), second(greenElements), 'green')
title("Principal Component 1 vs Principal Component 2", "with Colored Clusters using
Mahalanobis distance")
xlabel("PC1")
ylabel("PC2")
hold off
```

```
%{
figure(3)
red1Mean = mean(first(redElements));
red2Mean = mean(second(redElements));
redstd = std()
%}
```

Lab 6 Part 3

```
data = load('spikes.mat');
meanData = mean(data.spikes, 1);
standardD = std(data.spikes);
normalize = (data.spikes - meanData)./ standardD;
[coeff,score] = pca(normalize);
first = score(:,1);
second = score(:,2);
options = statset('display','final');
obj = gmdistribution.fit(score(:,1:2),3,'options',options);
clusters = cluster(obj, score(:,1:2));

redElements = find(clusters == 1);
greenElements = find(clusters == 2);
blueElements = find(clusters == 3);
%cyanElements = find(clusters == 4);
%magentaElements = find(clusters == 5);

figure(2)
scatter(first(redElements), second(redElements), 'red')
hold on
scatter(first(blueElements), second(blueElements), 'blue')
scatter(first(greenElements), second(greenElements), 'green')
%scatter(first(cyanElements), second(cyanElements), 'cyan')
%scatter(first(magentaElements), second(magentaElements), 'magenta')

centroids = obj.mu;
size(centroids)
plot(centroids(:,1),centroids(:,2), 'x')

ezcontour(@(x,y)pdf(obj, [x y]), [-15 15], [-15 15])
```

```
title("Principal Component 1 vs Principal Component 2", "with Colored Clusters using GMM K
= 3")
xlabel("PC1")
ylabel("PC2")
```