**EE511 Project 7**

**Zhang Fan**

**USC ID: 1417-68-5115**

Contents

[EE511 proj. 7 Zhang Fan USC ID: 1417-68-5115 1](#_Toc480138489)

[Question 1 3](#_Toc480138490)

[Analysis 3](#_Toc480138491)

[Basic Theory: 3](#_Toc480138492)

[Our goal 3](#_Toc480138493)

[Algorithm 3](#_Toc480138494)

[Code 3](#_Toc480138495)

[Result 4](#_Toc480138496)

[Question 2 4](#_Toc480138497)

[Algorithm 4](#_Toc480138498)

[Code 4](#_Toc480138499)

[Result 5](#_Toc480138500)

[Question 3 5](#_Toc480138501)

[Analysis: 5](#_Toc480138502)

[Result 6](#_Toc480138503)

[Close: 6](#_Toc480138504)

[well-separated: 7](#_Toc480138505)

[ellipsoidal covariance 8](#_Toc480138506)

[sphere 8](#_Toc480138507)

[8](#_Toc480138508)

[Comment 9](#_Toc480138509)

[Question 4 9](#_Toc480138510)

[Analysis: 10](#_Toc480138511)

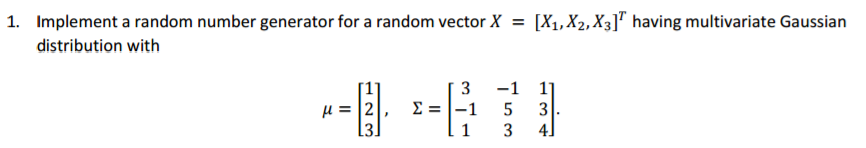
[Code for a: 10](#_Toc480138512)

[Code for b 10](#_Toc480138513)

[Result for a: 11](#_Toc480138514)

[Result for b: 12](#_Toc480138515)

# Question 1



Analysis:

### Basic Theory:

Generate a multivariate normal random variable:

Therefore, , where ;; , . , where i,j=1,2,3…n

Our goal

Generate and we have known and

## Algorithm

Firstly, generate three N~(0,1).

Secondly, apply Cholesky decomposition directly for .

Thirdly, According , we can easily generate X.

## Code

%----------------------

clear

Z=0;

Xsum=0

for i=1:1000

z1=normrnd(0,1);

z2=normrnd(0,1);

z3=normrnd(0,1);

mu = [1,2,3];

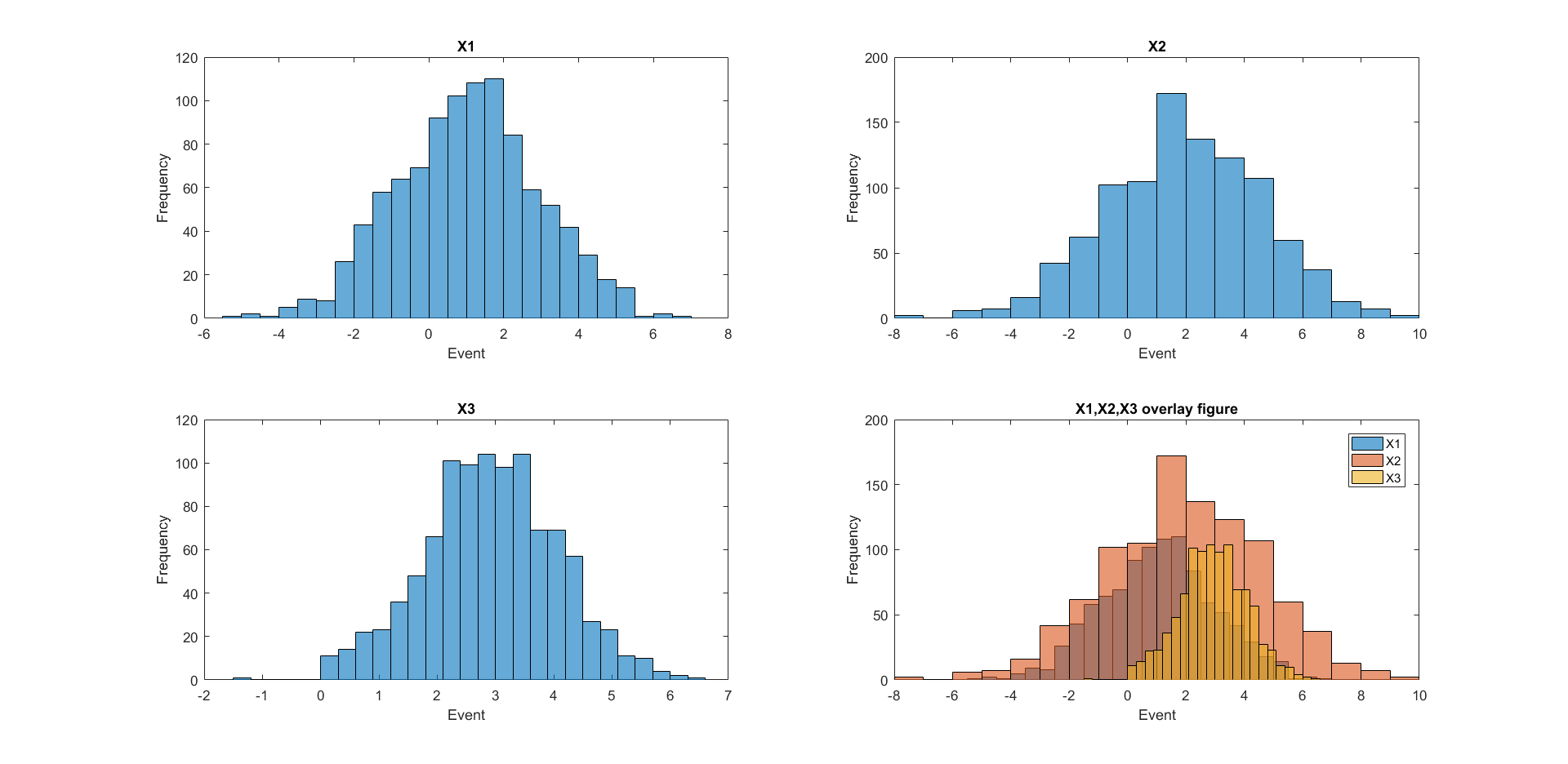
sigma = [3,-1,1;-1,5,3;1,3,4];

A=chol(sigma);

X(1:3,i)=A\*[z1,z2,z3]'+mu'

end

## Result



# Question 2



## Algorithm

Step1: generate N1~(1,1) and N2~(-1,1) and U~(0,1)

Step2: if U>0.4, then X=N2, else X=N1

Step3: repeat and generate histogram for X.

## Code

clear

for i=1:10000

x=rand;

n1=normrnd(-1,1);

n2=normrnd(1,1);

if x>0.4 X(i)=n2;

else X(i)=n1;

end

end

hold on

norm\_x=-5:0.1:5;

y=0.4\*normpdf(norm\_x,-1,1)+0.6\*normpdf(norm\_x,1,1);

plot(norm\_x,y);

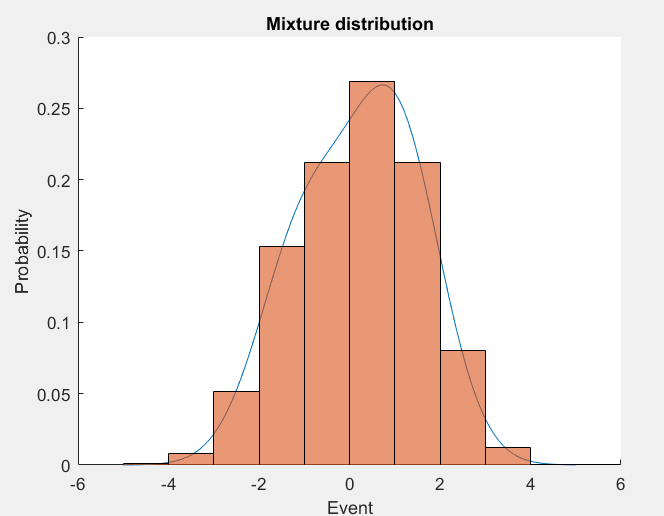
histogram(X,'Normalization','probability','Binwidth',1)

title('Mixture distribution')

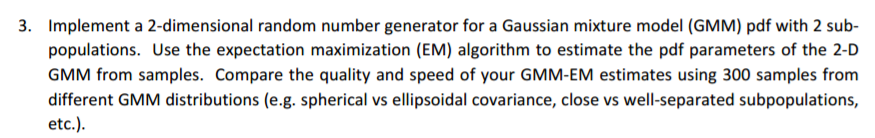
xlabel('Event');

ylabel('Probability')

## Result



# Question 3



## Analysis:

In this question, we can apply the method used in question 1 to generate multivariate Gaussian distribution and the method in question 2 to generate mixture distribution. And then apply EM algorithm to estimate the pdf parameters.

To compare the spherical vs ellipsoidal covariance, the difference of the sigma at one of the component should be relatively large to make the distribution look like a spherical. To compare the close vs well‐separated subpopulations, the expectation of two component should be tuned.

## Main code

clear;

%Implement a random number generator for a random(the method used in Q1)

mu1 = [-3,3];

sigma1 = [1 0;0 1];

mu2 = [3,-3];

sigma2 = [1 0;0 1];

%mixture distribution, method in Q2

p=0.5;

for i=1:30000

x=rand;

if x>p

r(i,:) = mvnrnd(mu1,sigma1,1);

else

r(i,:) = mvnrnd(mu2,sigma2,1);

end

end

%EM to estimate the paremeter

tic;

GMModel = fitgmdist(r,2)

toc;

plot(r(:,1),r(:,2),'.');

title(' sphere')

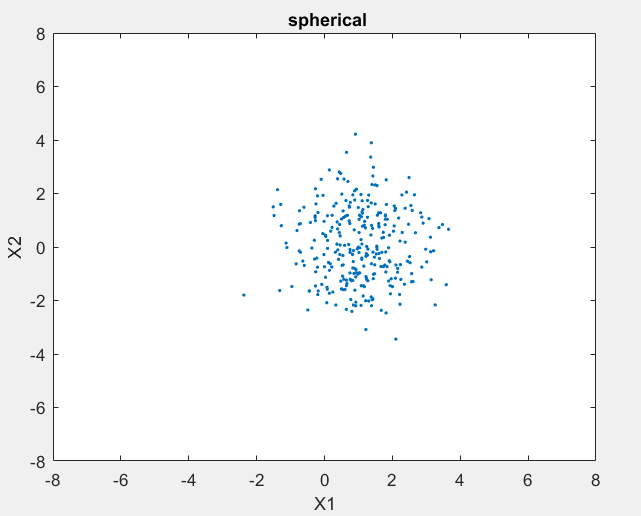
xlabel('X1');

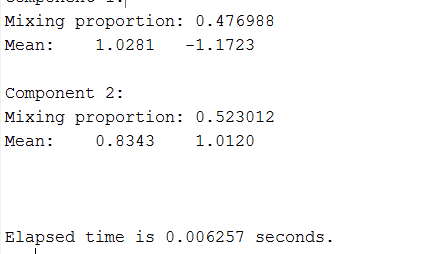
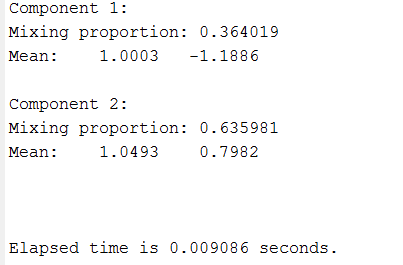
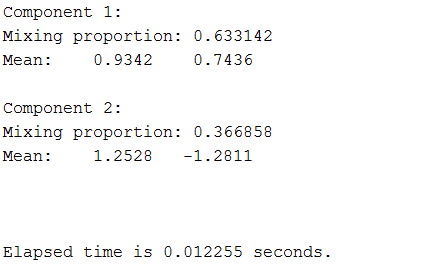
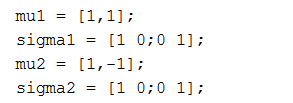
ylabel('X2')

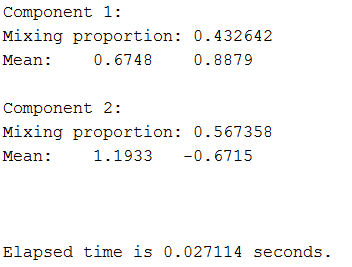
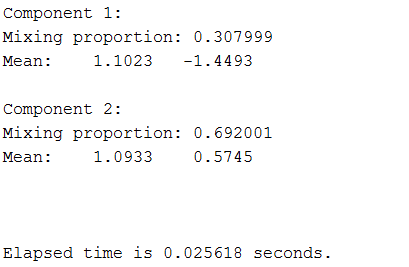
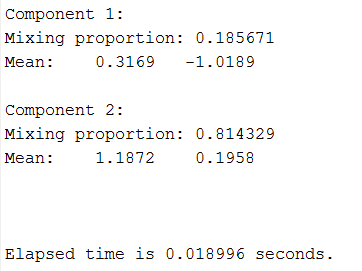
axis([-8 8 -8 8])

## Results

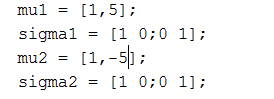
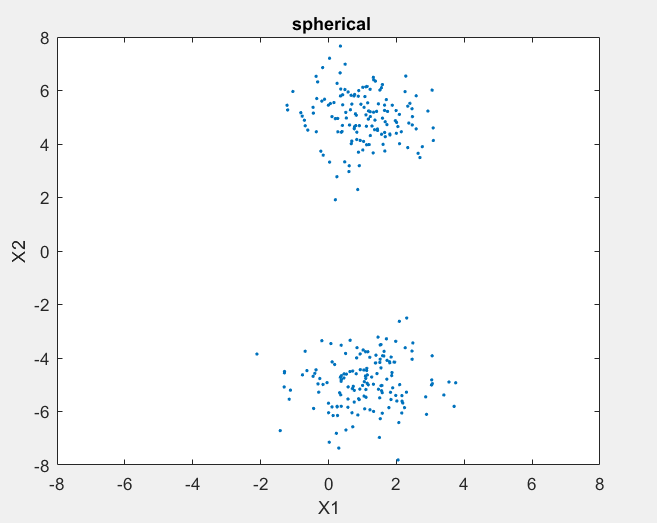
### Close:

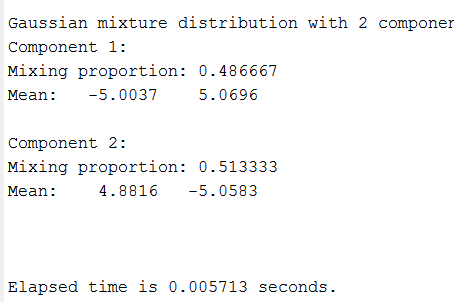
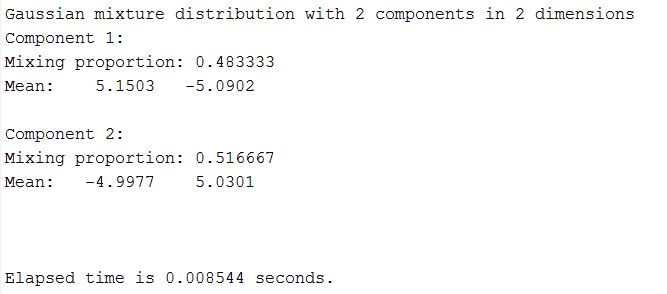
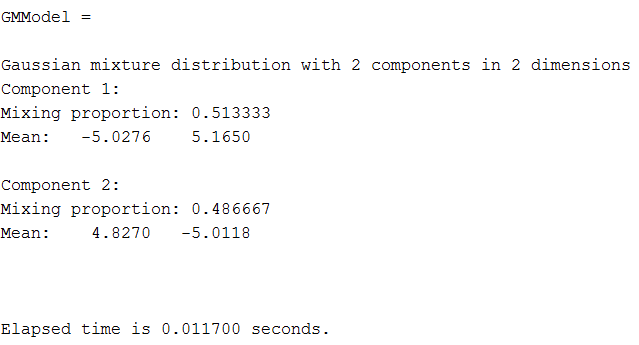
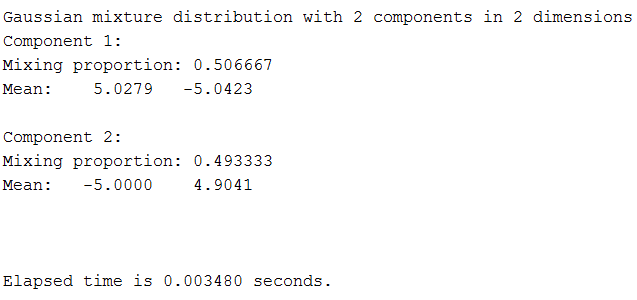
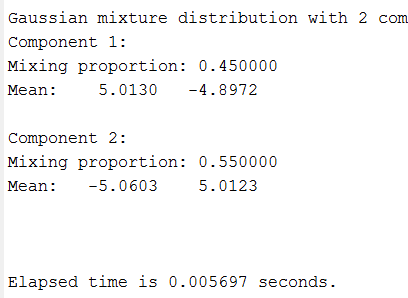
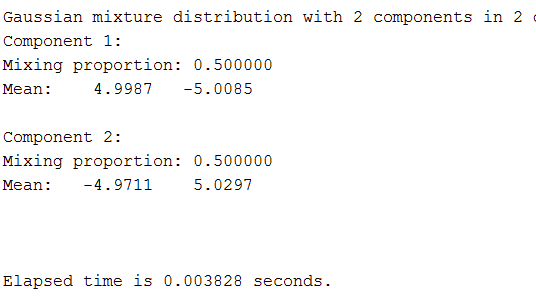




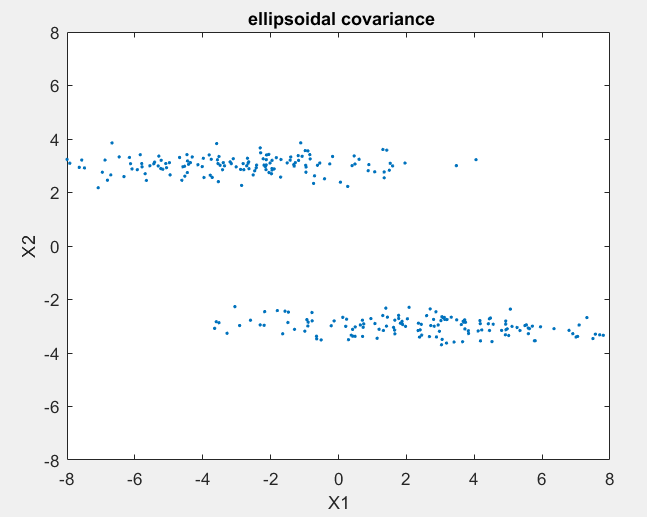
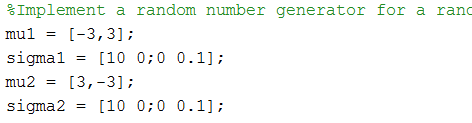


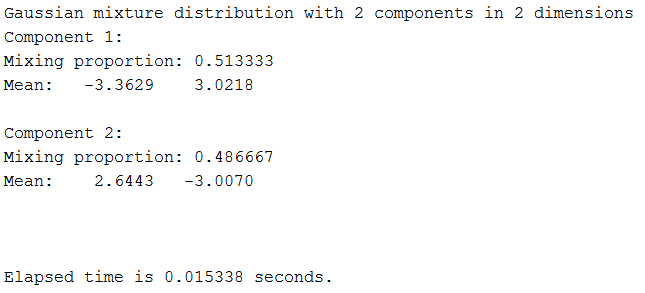
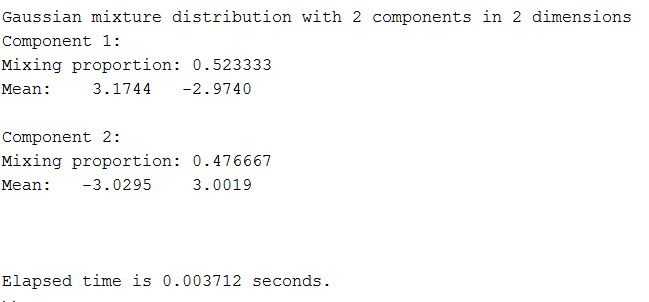
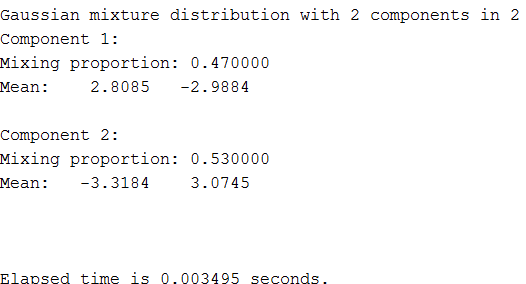
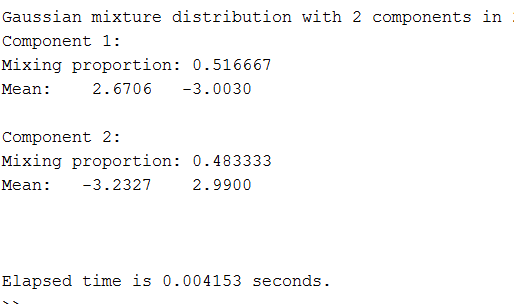
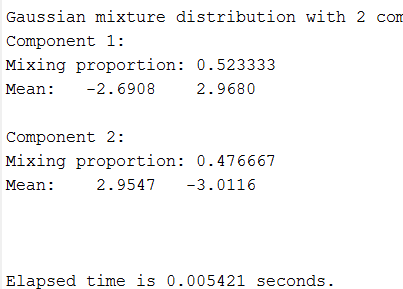
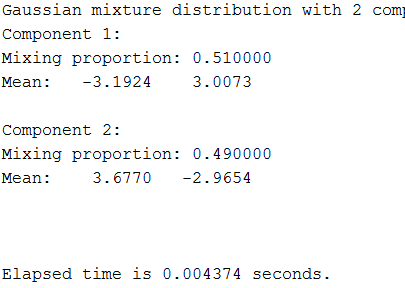
### well-separated:





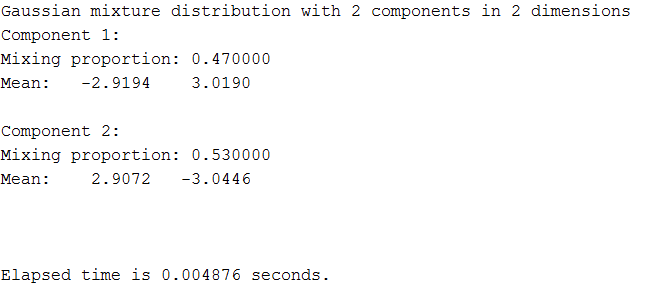
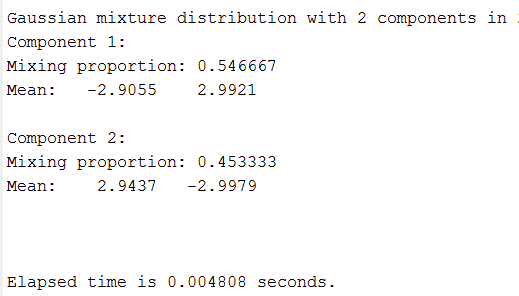
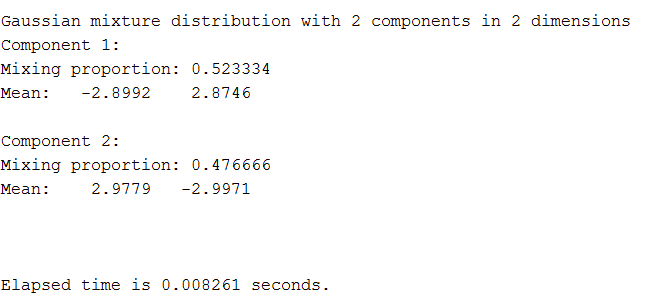
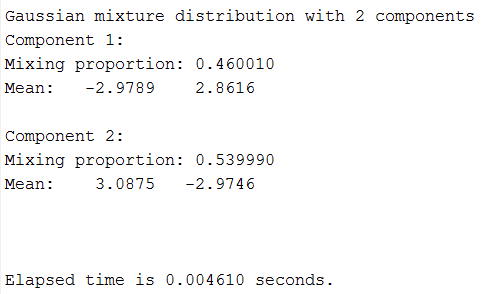
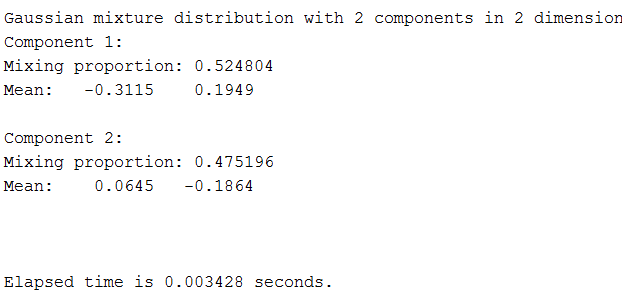
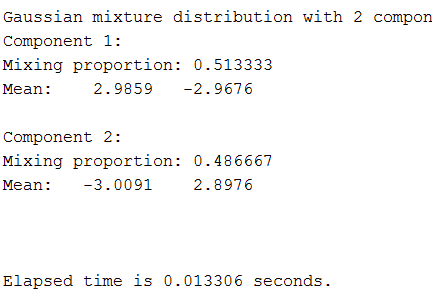
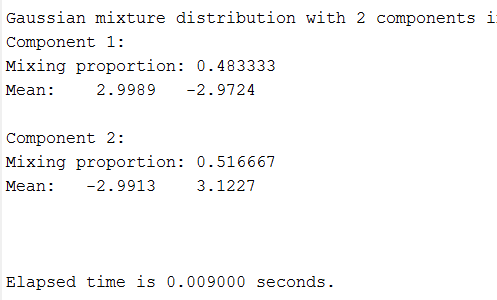
### ellipsoidal covariance





### sphere

### 

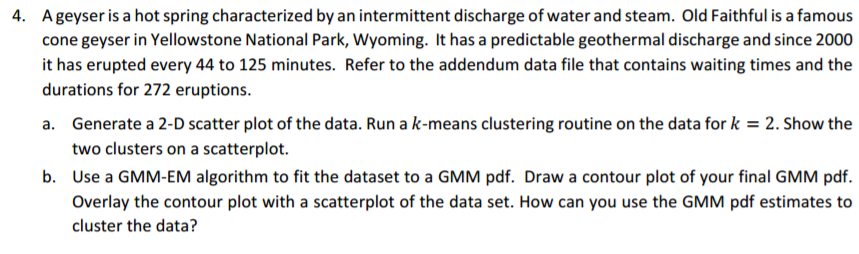
 

## Comment

From above results(Actually lots of trials have been implemented) we can see that the closer the two components are, the higher quality and faster the algorithm is.

And the closer that the distribution to sphere, the higher quality and faster of the GMM algorithm for estimating the parameters of pdf. However, it seems that the speed of the GMM algorithm is not highly influenced by the shape of distribution.

# Question 4



## Analysis:

In this question, we mainly apply K-means cluster method and GMM-EM algorithm to fit the dataset. For K-means algorithm, Clustering is a famous unsupervised learning problem with the following setup:

Given the n-dimensional data points {xi}ni=1 and predetermined number of clusters k, we want to find an assignment for every data points, i.e.

yi=A(xi)∈{1,2,⋯,k},∀i

k-means algorithm aims to partition the data points {xi}ni=1 with centers (means) of each cluster cj,j∈{1,2,⋯,k}. The main idea is to run the 2-step procedure iteratively: First, assign a point xi to the cluster with the closest center cj via yi=argminj∥xi−cj∥2. Then, after assigning all the points with a cluster index yi,∀i, the algorithm updates the center of each cluster by calculating the mean i.e.

cj=1nj∑i∈{i|yi=j}xi,

where nj denotes the number of data points in cluster j. The algorithm runs the assign-and-recalculate-mean steps repeatly until the the assignment remains unchanged.

k-means algorithm is a heuristic way to minimize the squared error objective

J=∑kj=1∑i∈{i|yi=j}∥xi−cj∥2,

which can be also viewed as the sum of squared distances from data point xi to the mean of its cluster cj.

(Reference: <https://piazza.com/class/ixp26i0vk6s5dg?cid=64>).

For GMM model, the algorithm is as the following shows:

Step 1: find the complete-data likelihood function for multivariate normal distribution.

Step 2: do E step.

Step 3: do M step.

Step 4: convergence check and terminate.

## Code for a:

%% Generate data for kmeans

clear

X= load('data.txt');

figure(1)

X=X(:,2:3);

plot(X(:,1), X(:,2), 'o');

title('Data Points without Labels')

%% kmeans and scatter plot

[y C] = kmeans(X,2); % Find the assignment y and the means C of each cluster

figure(2)

plot(X(y==1,1),X(y==1,2), 'x');

hold on

plot(X(y==2,1),X(y==2,2), 'o');

plot(C(1,1),C(1,2), 'rx','LineWidth',2);

plot(C(2,1),C(2,2), 'ro','LineWidth',2);

legend('Points of cluster 1','Points of cluster 2')

title('Data Points with Labels by K-means Clustering')

hold off

## Code for b

%% Generate data for kmeans

clear

X= load('data.txt');

X=X(:,2:3);

title('Data Points without Labels')

GMModel = fitgmdist(X,2);

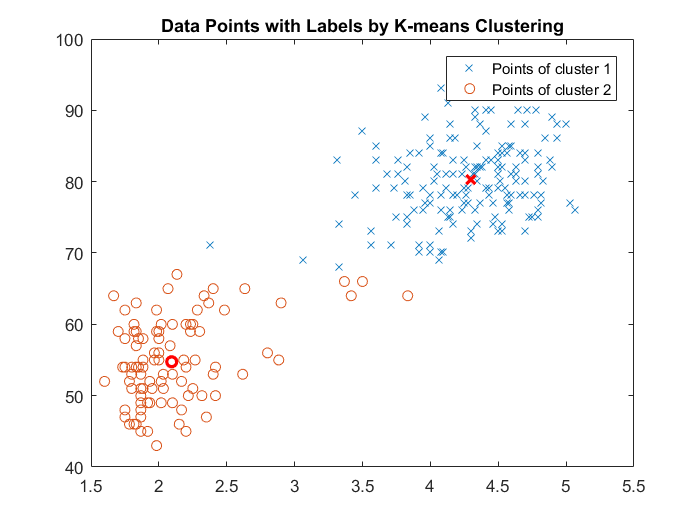
% 2D projection

ezcontourf(@(x,y) pdf(GMModel,[x y]),[1.5,5.5],[40,100]);

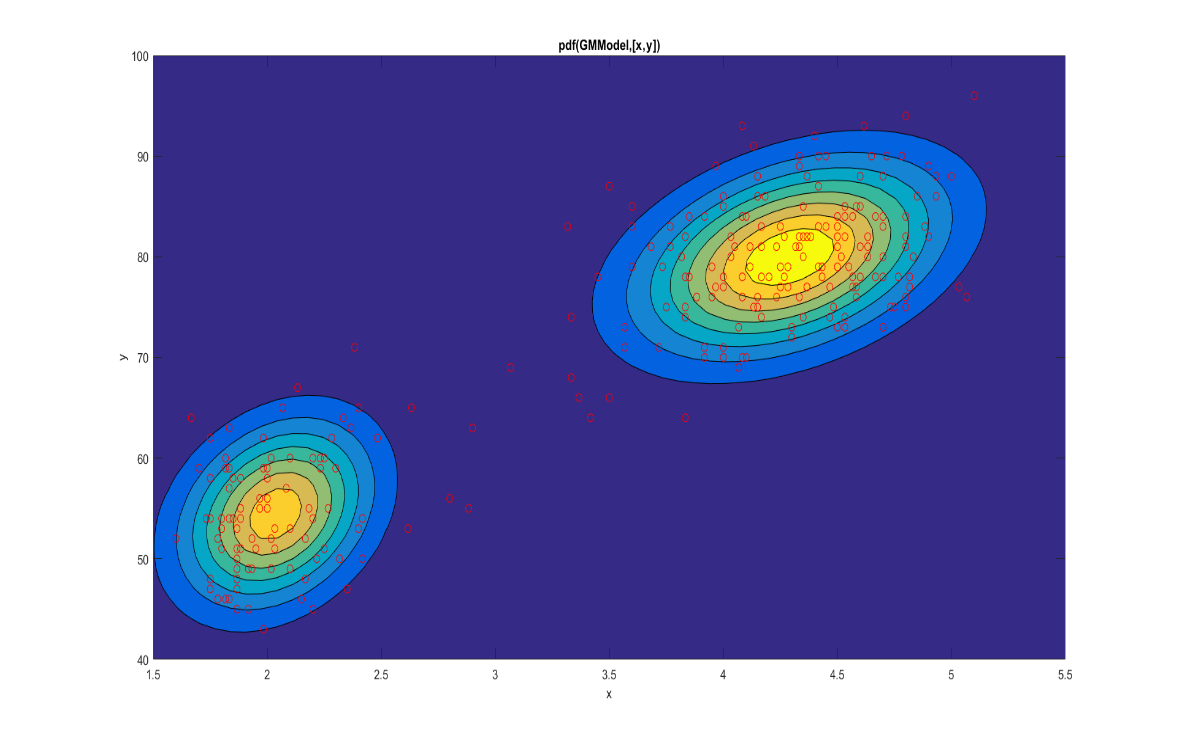
hold on

plot(X(:,1), X(:,2), 'or');

## Result for a:



## Result for b:



The yellow zone from the above figure means it has higher probability that the data belongs to correspond component. Therefore, we can just set a threshold of probability for the data: if the probability of the data that belongs to one component is higher than this threshold, then accept it. Or reject it.

## Comment

K-means method and GMM method both can be used for clustering. But there are still a little difference between them. For example, GMM method is according to the probability while K-means is according to distance between.