University of Southern California EE511

Expectation Maximization

Project #7

Name: Wu Jiawei
USCID: 9600392575

Abstract

In the project of expectation maximization, four experiments are conducted using Matlab. The core method of the project is to simulate Bayesian iterative learning process by implementing the random number generators. The theories of expectation maximization algorithm, multivariate Gaussian distribution, the Gaussian mixture model and the k-means clustering methods are applied comprehensively in the lab. The experiments are repeated multiple times and the outcomes are shown in diagrams and calculations. The comparisons are conducted between the experimental results and the theoretical probability density functions.

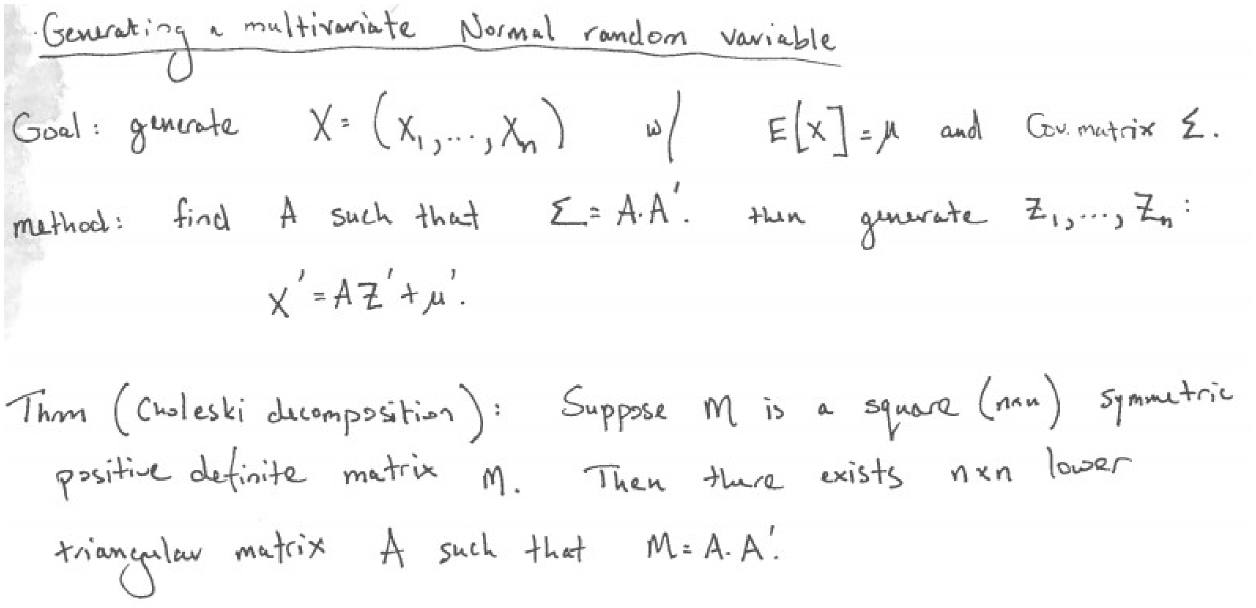
Introduction

Three experiments are conducted in the lab. All the samples are generated by random selection and the experiments are repeated multiple times. The goal of the first trial is to implement the random number generator to obtain a random vector with given multivariate Gaussian distribution. The aim of the second experiment is to generate samples following the mixture distribution using random number generator and compare with the theoretical probability density function. The objective of the third experiment is running simulation to generate samples for a Gaussian mixture model and estimate the probability density function using the expectation maximization algorithm. The aim of the fourth lab is to run the k-means clustering to process data and use the GMM-EM algorithm to fit the dataset.

Methodology & Results

Experiment No.1

Description of Algorithm:



Generating a multivariate Normal random variable

Goal: generate $X = (X_1, \dots, X_n)$ w/ $E[X] = \mu$ and Cov. matrix Σ.

method: find A such that $\Sigma = A \cdot A'$. then generate $Z_1, \dots, Z_n$:

$$X' = AZ' + \mu'.$$

Thm (Choleski decomposition): Suppose M is a square ($n \times n$) symmetric positive definite matrix M. Then there exists $n \times n$ lower triangular matrix A such that $M = A \cdot A'$.

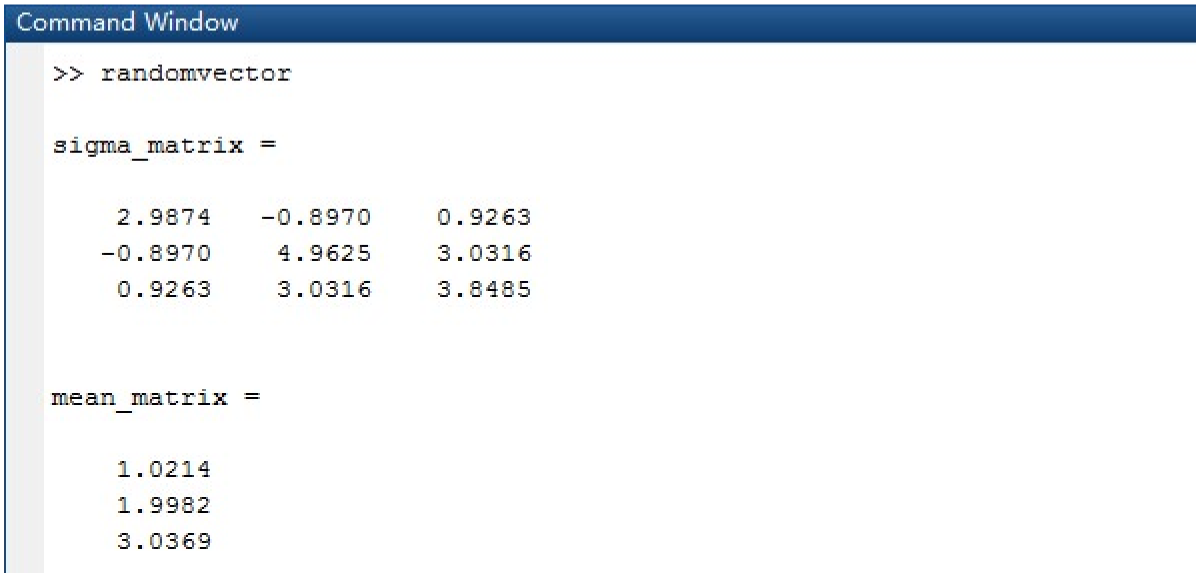
Description of Method:

The random vector is generated by random number generator. The cholesky decomposition is applied to the given covariance matrix. The sample covariance matrix and sample mean matrix are generated and compared with the theoretical probability density function parameters. The 3-D scatter plot is generated to show the result.

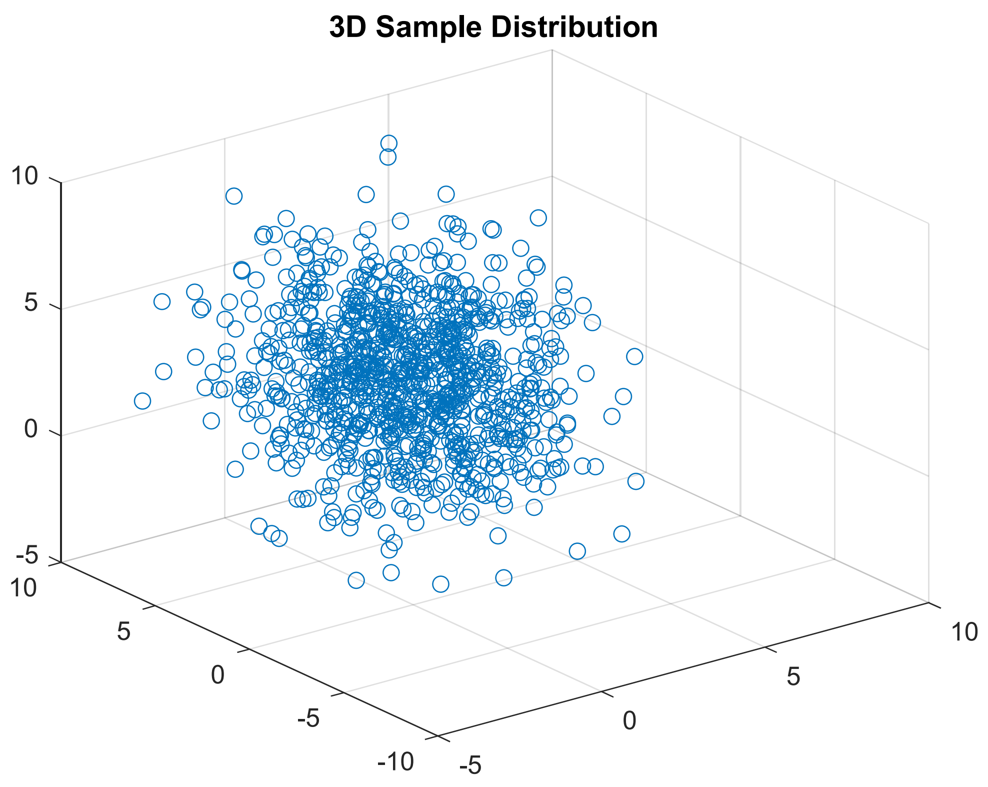
Code:

```
%Implement random number generator
for i=1:1000
sigma=[3 -1 1; -1 5 3; 1 3 4];
A=chol(sigma,'lower');%Applying the cholesky decomposition to given matrix
Z=randn(3,1);
mu=[1;2;3];
X(i,:)=A*Z+mu;
end

%Compute sample covariance matrix
for i=1:3
    for j=1:3
        cov1=cov(X(:,i),X(:,j));
        cov2(i,j)=cov1(1,2);
    end
end

%Compute sample mean matrix
for i=1:3
   mean1(i)=mean(X(:,i));
end

sigma_matrix=cov2
mean_matrix=mean1'
scatter3(X(:,1),X(:,2),X(:,3));
title('3D Sample Distribution');
```

Simulation Result:



```
>> randomvector

sigma_matrix =

    2.9874   -0.8970    0.9263
   -0.8970    4.9625    3.0316
    0.9263    3.0316    3.8485


mean_matrix =

    1.0214
    1.9982
    3.0369
```



Finding:

The results shows that the random vector $\mathrm{X} = [X_1, X_2, X_3]$ generated follows the given multivariate Gaussian distribution.

The covariance matrix for the random vector is

$$\sum = \begin{bmatrix} 2.9874 & -0.8970 & 0.9263 \\ -0.8970 & 4.9625 & 3.0316 \\ 0.9263 & 3.0316 & 3.8485 \end{bmatrix}$$

Which is close to the theoretical covariance matrix.

$$\sum = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 5 & 3 \\ 1 & 3 & 4 \end{bmatrix}$$

The mean matrix for the random vector is

$$\mu = \begin{bmatrix} 1.0214 \\ 1.9982 \\ 3.0369 \end{bmatrix}$$

Which is close to the theoretical covariance matrix.

$$\mu = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Thus, the random vector generated by the random number generator follows the multivariate Gaussian distribution.

Experiment No.2

Description of Algorithm:

The mixture distribution is the probability distribution of a random variable derived from a collection of other random variables.

The mixture component is the individual distributions that are combined to form the mixture distribution.

The mixture weight is the probability associated with each component.

The Gaussian mixture model (GMM) is to find mean and covariances of Gaussians and weight of each.

For the distribution $f(x) = 0.4N(-1,1) + 0.6N(1,1)$

It means that the probability that $N(-1,1)$ occurs is 0.4, while the probability that $N(1,1)$ occurs is 0.6.

Description of Method:

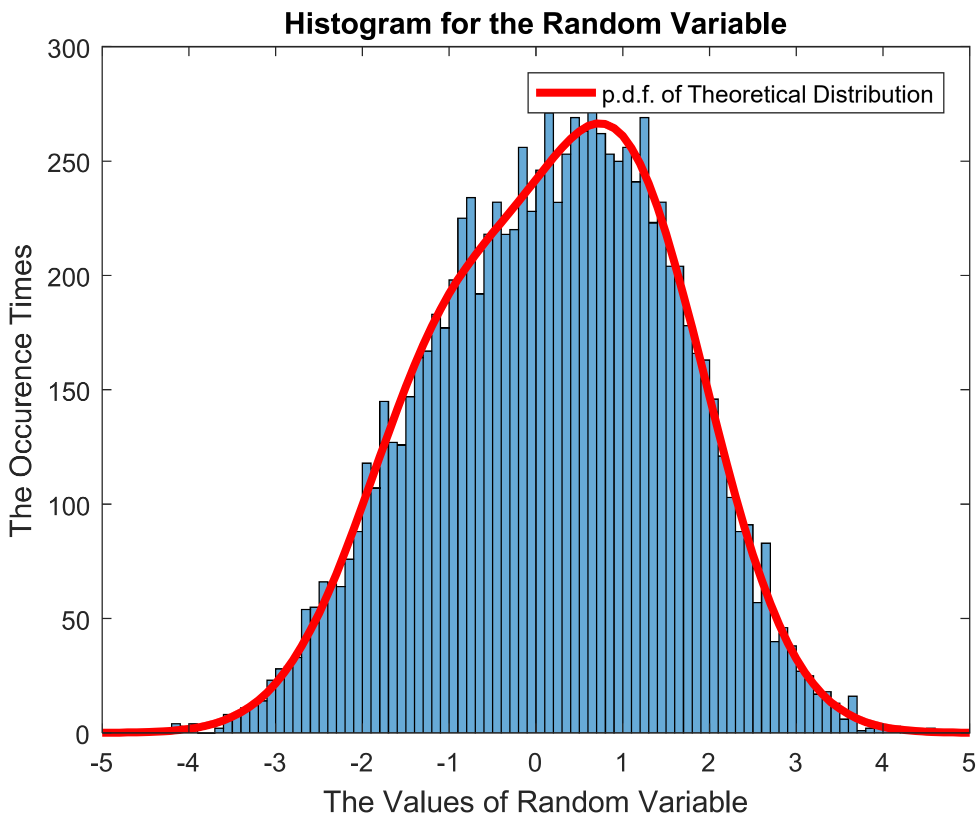
In the lab2, the for loop is utilized to generated samples follow the N(-1,1) 40% of the time when the random number is less than 0.4 and generated samples follow the N(1,1) 60% of the time when the random number is more than 0.4. The histogram is generated to show the result and the theoretical pdf of the mixture distribution is overlay to the histogram.

Code:

```
N=10000;%N represents the sample size
A=zeros(10000,1);
```

```
%Generate mixture distribution by looping and comparing random numbers
%with P=0.4
for i=1:10000
u=rand;
if u<0.4
    A(i)=normrnd(-1,1);
else
    A(i)=normrnd(1,1);
end
end
%Generate theoretical nromal distribution
x=-5:0.1:5;
B=0.4*normpdf(x,-1,1)+0.6*normpdf(x,1,1);

%Plot the histogram
histogram(A,'binwidth',.1);
hold on;
T1=plot(x,B*.1*N,'-r','LineWidth', 3);
title('Histogram for the Random Variable');
xlabel('The Values of Random Variable');
ylabel('The Occurence Times');
legend([T1],'p.d.f. of Theoretical Distribution');
hold off;
```

Simulation Result:



Finding:

The experimental distribution is shown in the result that the theoretical mixture distribution fits the experimental data distribution. It looks like a single bell curve because the means of two normal distributions are too close to each other.

Experiment No.3

Description of Algorithm:

The summary of the EM algorithm procedure is that

EM algorithm for estimating GMM parameters

1. **Initialization:** Choose the initial estimates $w_j^{(0)}$, $\mu_j^{(0)}$, $\Sigma_j^{(0)}$, $j = 1, \dots, k$, and compute the initial log-likelihood

$$L^{(0)} = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{j=1}^{k} w_j^{(0)} \phi(y_i \,|\, \mu_j^{(0)}, \Sigma_j^{(0)})\right).$$

2. **E-step:** Compute

$$\gamma_{ij}^{(m)} = \frac{w_j^{(m)} \phi(y_i \,|\, \mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \phi(y_i \,|\, \mu_l^{(m)}, \Sigma_l^{(m)})}, \; i = 1, \dots, n, \; j = 1, \dots, k,$$

and

$$n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)}, \; j = 1, \dots, k.$$

3. **M-step:** Compute the new estimates

$$w_j^{(m+1)} = \frac{n_j^{(m)}}{n}, \; j = 1, \dots, k,$$

$$\mu_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i, \; j = 1, \dots, k,$$

$$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} \left(y_i - \mu_j^{(m+1)}\right)\left(y_i - \mu_j^{(m+1)}\right)^T, \; j = 1, \dots, k.$$

4. **Convergence check:** Compute the new log-likelihood

$$L^{(m+1)} = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{j=1}^{k} w_j^{(m+1)} \phi(y_i \,|\, \mu_j^{(m+1)}, \Sigma_j^{(m+1)})\right).$$

Return to step 2 if $|L^{(m+1)} - L^{(m)}| > \delta$ for a preset threshold δ; otherwise end the algorithm.

Description of Method:

In the lab3, 2-D GMM random numbers with two subpopulations are generated. The GMM-EM algorithm is used to iteratively update the estimates of weights and distribution parameters. This resides in a while loop that continues as long as the change in the log likelihood continues to change by more than the threshold. Different GMM distributions are considered and the histograms are generated to show the results.

Code:

```
clc; clear; close all;
```

```
N=300;%Sample size
%case one-close subpopulations
sigma1=[2 0; 0 2]; sigma2=[1 0; 0.5 5];

mu1=[0;3];
mu2=[-3;0];
weight1=0.5;
weight2=0.5;
weight=[0.5,0.5];

%case two-well-seperated subpopulations
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.5;
% weight2=0.5;
% weight=[0.5,0.5];

%case three-spherical subpopulations
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.25;
% weight2=0.75;
% weight=[0.25,0.75];

%case four-spherical subpopulations
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.15;
% weight2=0.85;
% weight=[0.15,0.85];

%case five-ellipsoidal subpopulations
% sigma1=[1 -2; -2 6];
% sigma2=[4 0; 0 1];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.5;
% weight2=0.5;
% weight=[0.5,0.5];


%Generating samples with multivariate Gaussian distribution
for i=1:N
    u=rand;

    A1=chol(sigma1,'lower');
    A2=chol(sigma2,'lower');

    Z=randn(2,1);
    if u<weight1
        X=A1*Z+mu1;
    else
        X=A2*Z+mu2;
    end
    m(i,1)=X(1);
    m(i,2)=X(2);
end

%Initialize with k-means algorithm.
[y,C]=kmeans(m,2);
miu1=C(1,:);
miu2=C(2,:);
%Obtain the centers of two subpopulations and set as the first estimation
%of mean.
miu=zeros(2,1,2);
miu(:,:,1)=miu1';
miu(:,:,2)=miu2';
%Set the initial guess of covariance matrix.
covE=zeros(2,2,2);
covE(:,:,1)=[1 0;0 1];
covE(:,:,2)=[1 0;0 1];
SH=0;
%Compute the initial log-likelihood.
for i=1:N

SH=SH+log(weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m
(i,:),miu(:,:,2)',covE(:,:,2)));
end
L=SH/N;

%Draw the diagram for the first step
A=reshape(covE(:,:,1),[2 2]);
B=reshape(covE(:,:,2),[2 2]);
C = reshape(miu,[2 2]);
covEE=cat(3,A,B);
obj = gmdistribution(C,covEE,weight);
figure;
ezcontourf(@(x,y) pdf(obj,[x y]),[-10 10],[-10 10]);
title('2D Initial Diagram');
figure;
ezsurf(@(x,y)pdf(obj,[x y]),[-10 10],[-10 10]);
title('3D Final Diagram');

%E-steps
Nor=zeros(N,2);
n=zeros(1,2);
Lnew=0;
counter=0;
while abs(L-Lnew)>0.001
    L=Lnew;
    for j=1:2
        for i=1:N
            SP=0;
            for z=1:2
```

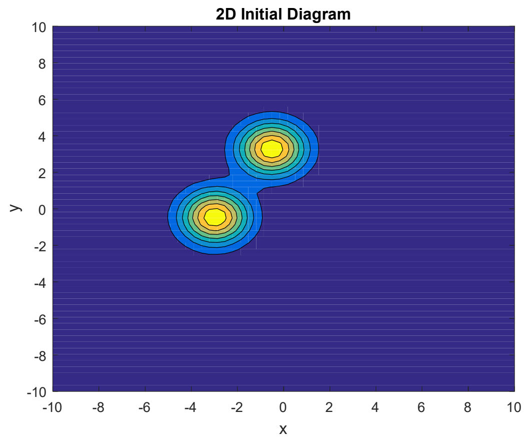
```
SP=weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m(i,:),m
iu(:,:,2)',covE(:,:,2));
            end
            Nor(i,j)=weight(j)*mvnpdf(m(i,:),miu(:,:,j)',covE(:,:,j))/SP;
        end
    end
    n(1)=sum(Nor(:,1));
    n(2)=sum(Nor(:,2));
    %M-steps
    weight=n/N;
    for j=1:2
        miu(:,:,j)=(Nor(:,j)'*m/n(j))';
        Acov=zeros(2,2);
        for i=1:N
            Acov=Acov+Nor(i,j)*(m(i,:)'-miu(:,:,j))*(m(i,:)'-miu(:,:,j))';
        end
        covE(:,:,j)=Acov/n(j);
    end

    SH=0;
    for i=1:N

SH=SH+log(weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m
(i,:),miu(:,:,2)',covE(:,:,2)));
    end
    Lnew=SH/N;
    counter=counter+1;

end
%Display the quality GMM-EM estimations
covE
miu
%Display the speed of GMM-EM estimations
speed=counter;
speed
%Display the final weight of GMM-EM estimations
Finalweight=weight;
Finalweight
%Draw the diagram for the last step
A=reshape(covE(:,:,1),[2 2]);
B=reshape(covE(:,:,2),[2 2]);
C = reshape(miu,[2 2]);
covEE=cat(3,A,B);
obj = gmdistribution(C,covEE,weight);
figure;
ezcontourf(@(x,y) pdf(obj,[x y]),[-10 10],[-10 10]);
title('2D Initial Diagram');
figure;
ezsurf(@(x,y)pdf(obj,[x y]),[-10 10],[-10 10]);
title('3D Final Diagram');
```

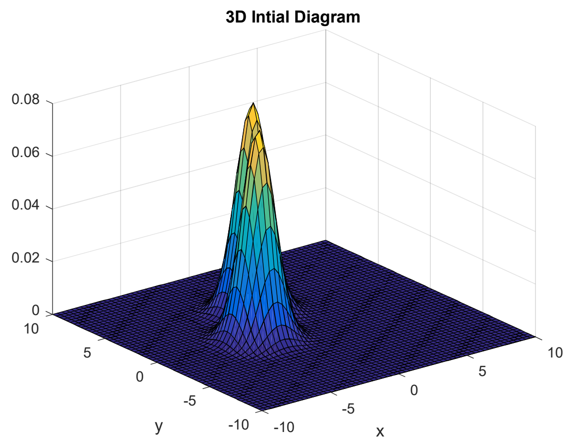
Simulation Result:

1). Case-one: close subpopulations

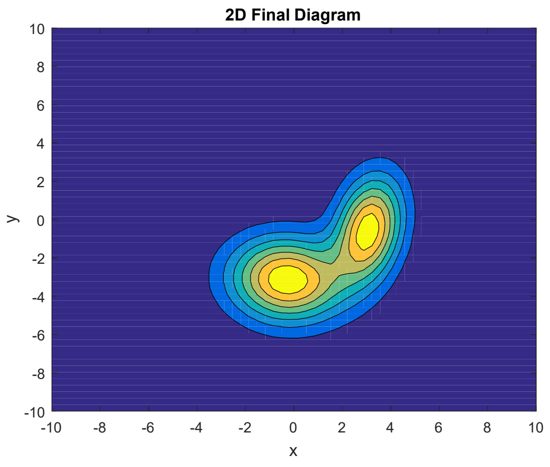
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;3];
% mu2=[-3;0];
% weight1=0.5;
% weight2=0.5;
% weight=[0.5,0.5];

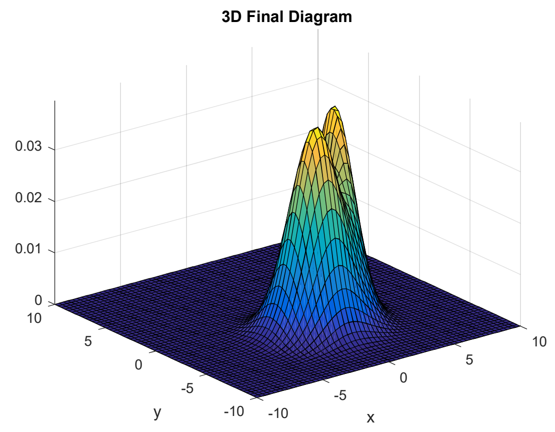
Initial Plot:

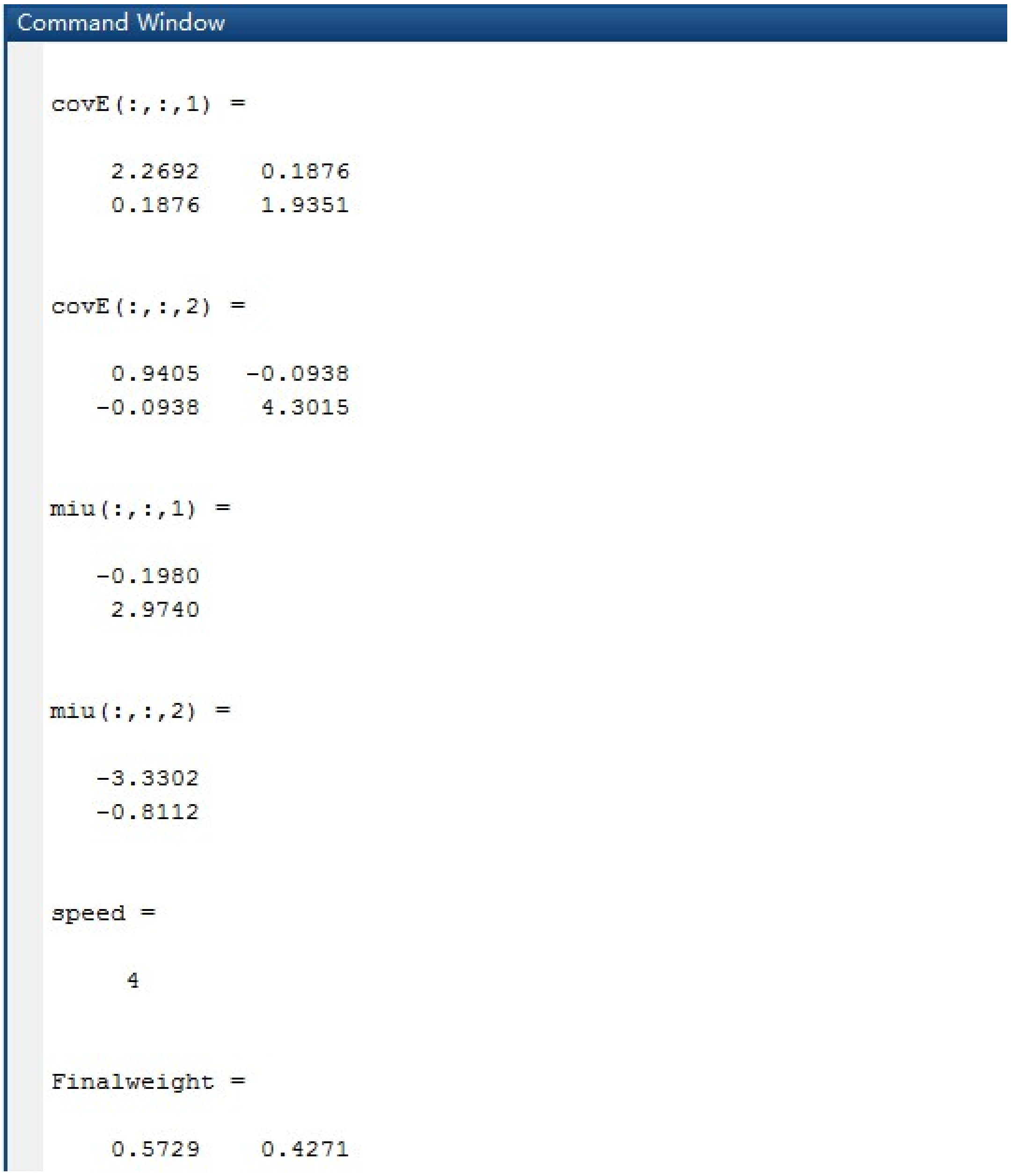




Final Plot:







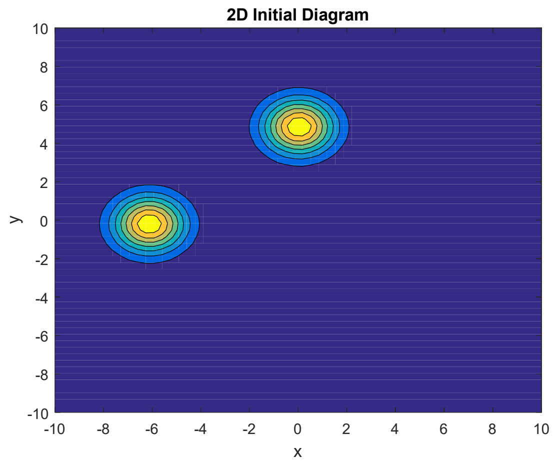
Command Window

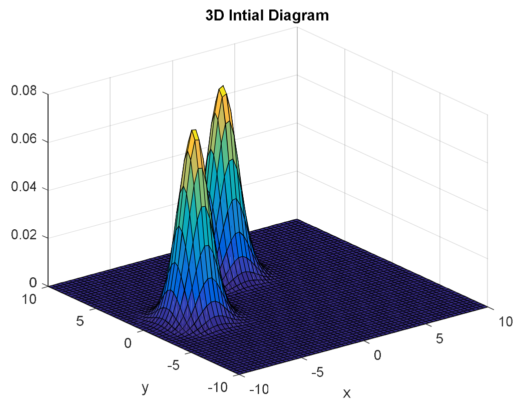
```
covE(:,:,1) =

    2.2692    0.1876
    0.1876    1.9351


covE(:,:,2) =

    0.9405   -0.0938
   -0.0938    4.3015


miu(:,:,1) =

   -0.1980
    2.9740


miu(:,:,2) =

   -3.3302
   -0.8112


speed =

     4


Finalweight =

    0.5729    0.4271
```

2). Case-two: well-separate subpopulations

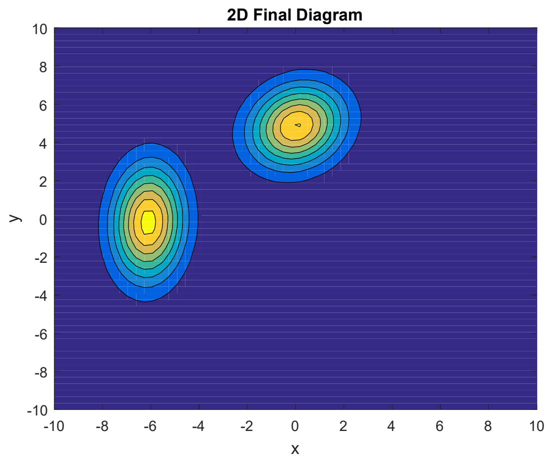
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.5;
% weight2=0.5;
% weight=[0.5,0.5];

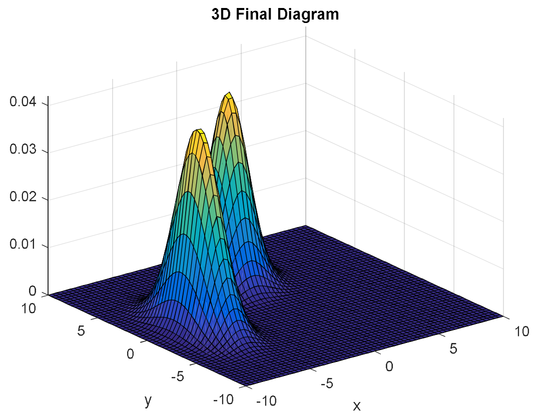
Initial Plot:

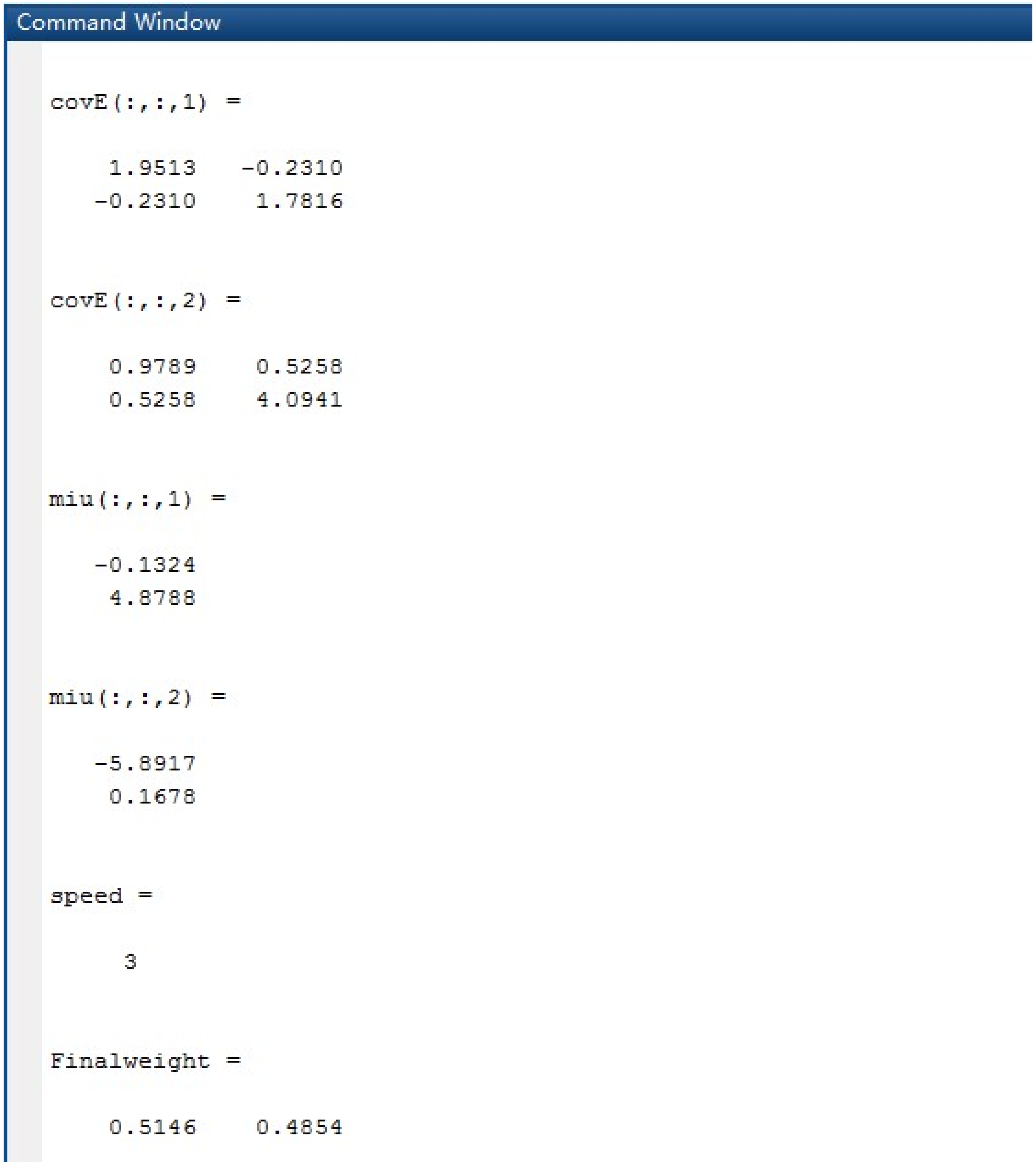




Final Plot:





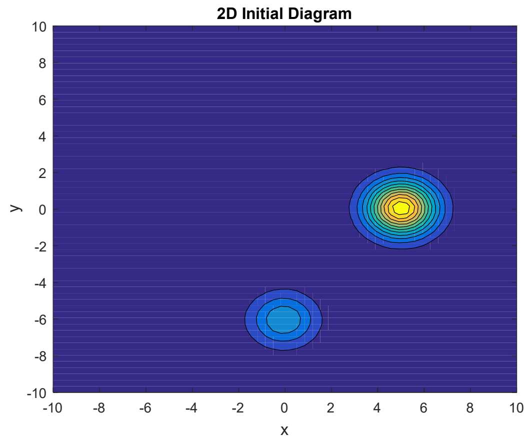

Command Window

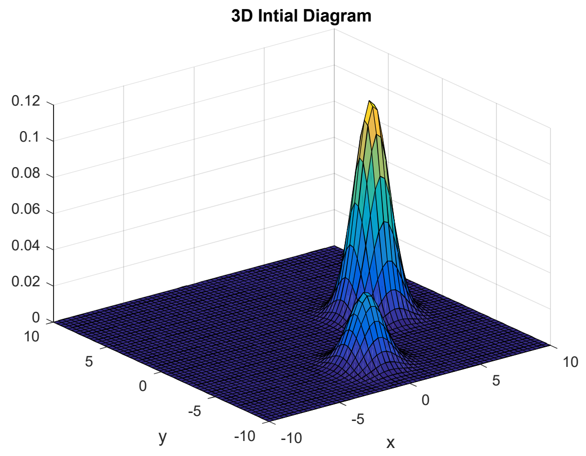
```

covE(:,:,1) =

    1.9513   -0.2310
   -0.2310    1.7816


covE(:,:,2) =

    0.9789    0.5258
    0.5258    4.0941


miu(:,:,1) =

   -0.1324
    4.8788


miu(:,:,2) =

   -5.8917
    0.1678


speed =

     3


Finalweight =

    0.5146    0.4854
```

3). Case-three: spherical subpopulations with weight=[0.25 , 0.75]

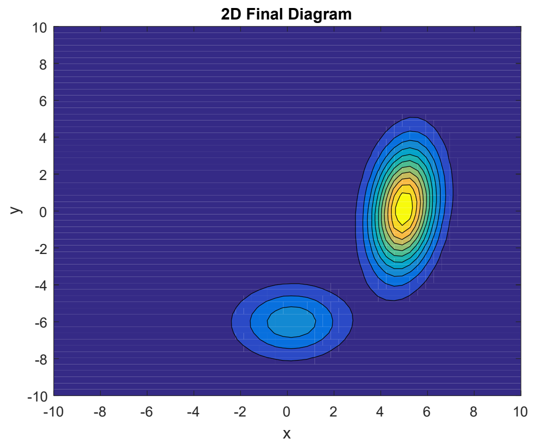
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.25;
% weight2=0.75;
% weight=[0.25,0.75];

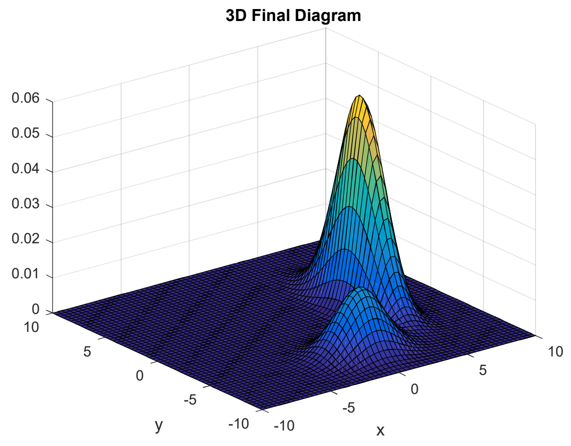
Initial Plot:

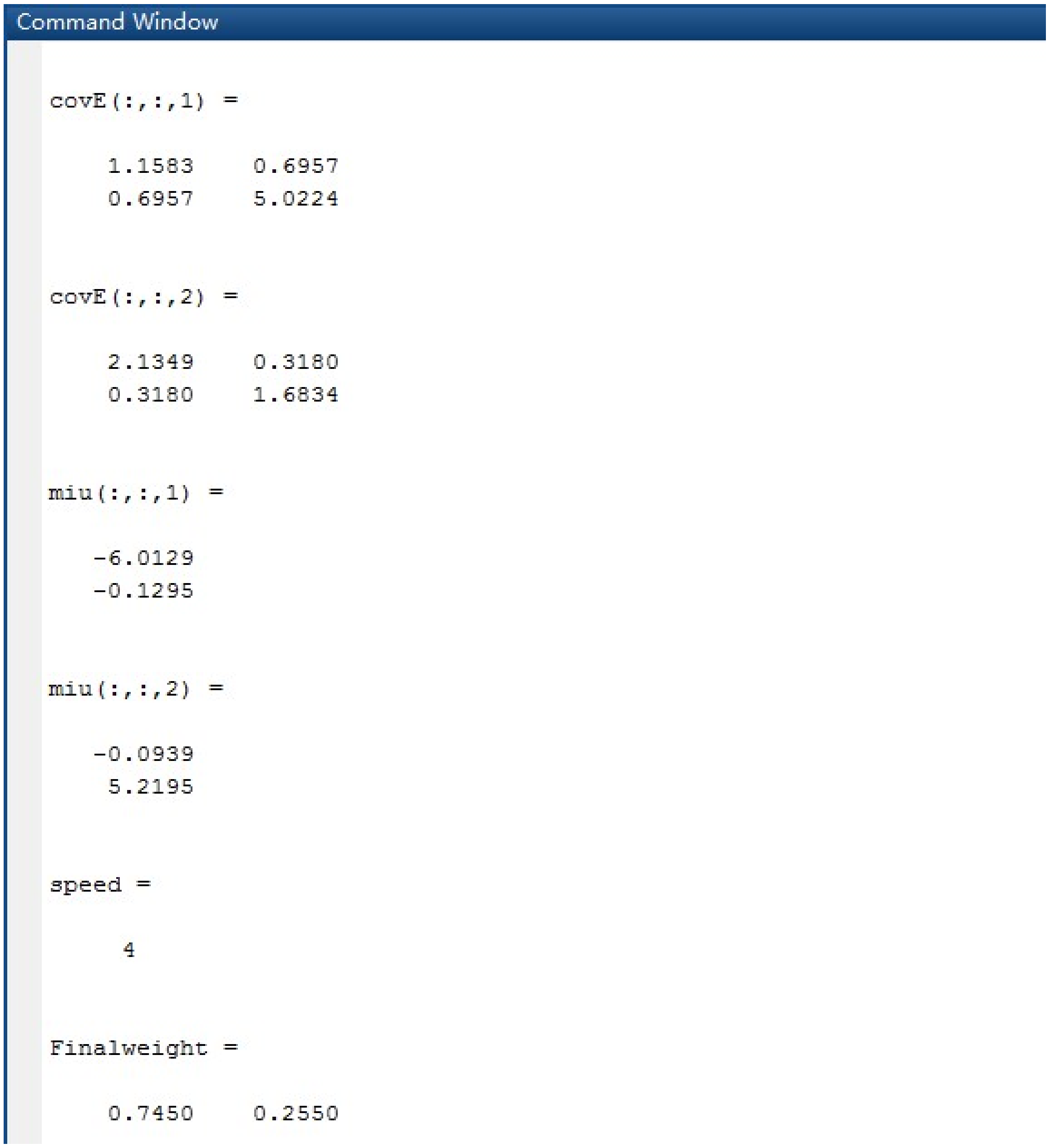




Final Plot:





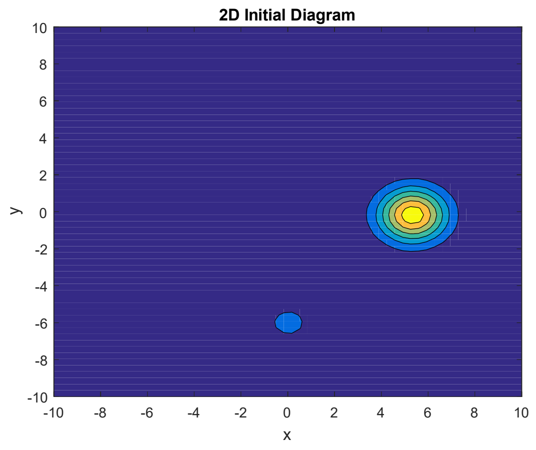

Command Window

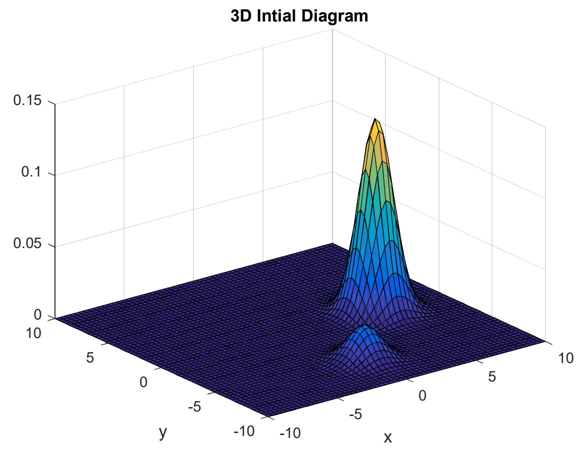
```

covE(:,:,1) =

    1.1583    0.6957
    0.6957    5.0224


covE(:,:,2) =

    2.1349    0.3180
    0.3180    1.6834


miu(:,:,1) =

   -6.0129
   -0.1295


miu(:,:,2) =

   -0.0939
    5.2195


speed =

     4


Finalweight =

    0.7450    0.2550
```

4). Case-four: spherical subpopulations with weight=[0.15 , 0.85].

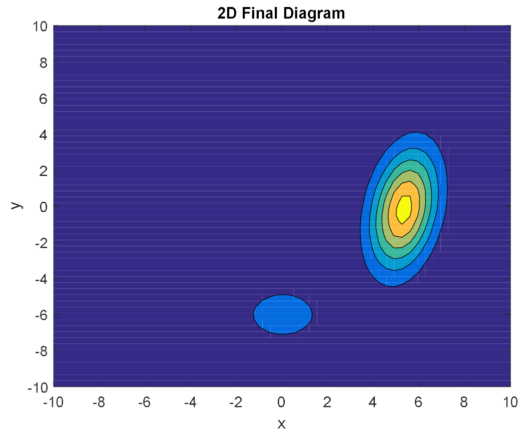
% sigma1=[2 0; 0 2];
% sigma2=[1 0; 0.5 5];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.15;
% weight2=0.85;
% weight=[0.15,0.85];

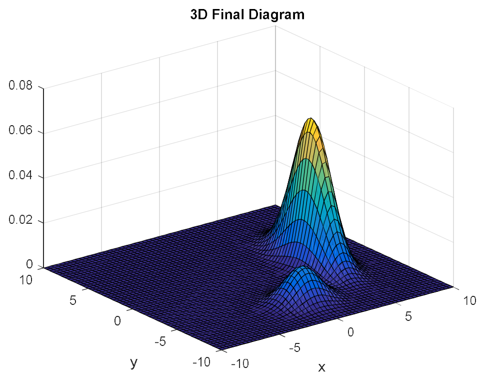
Initial Plot

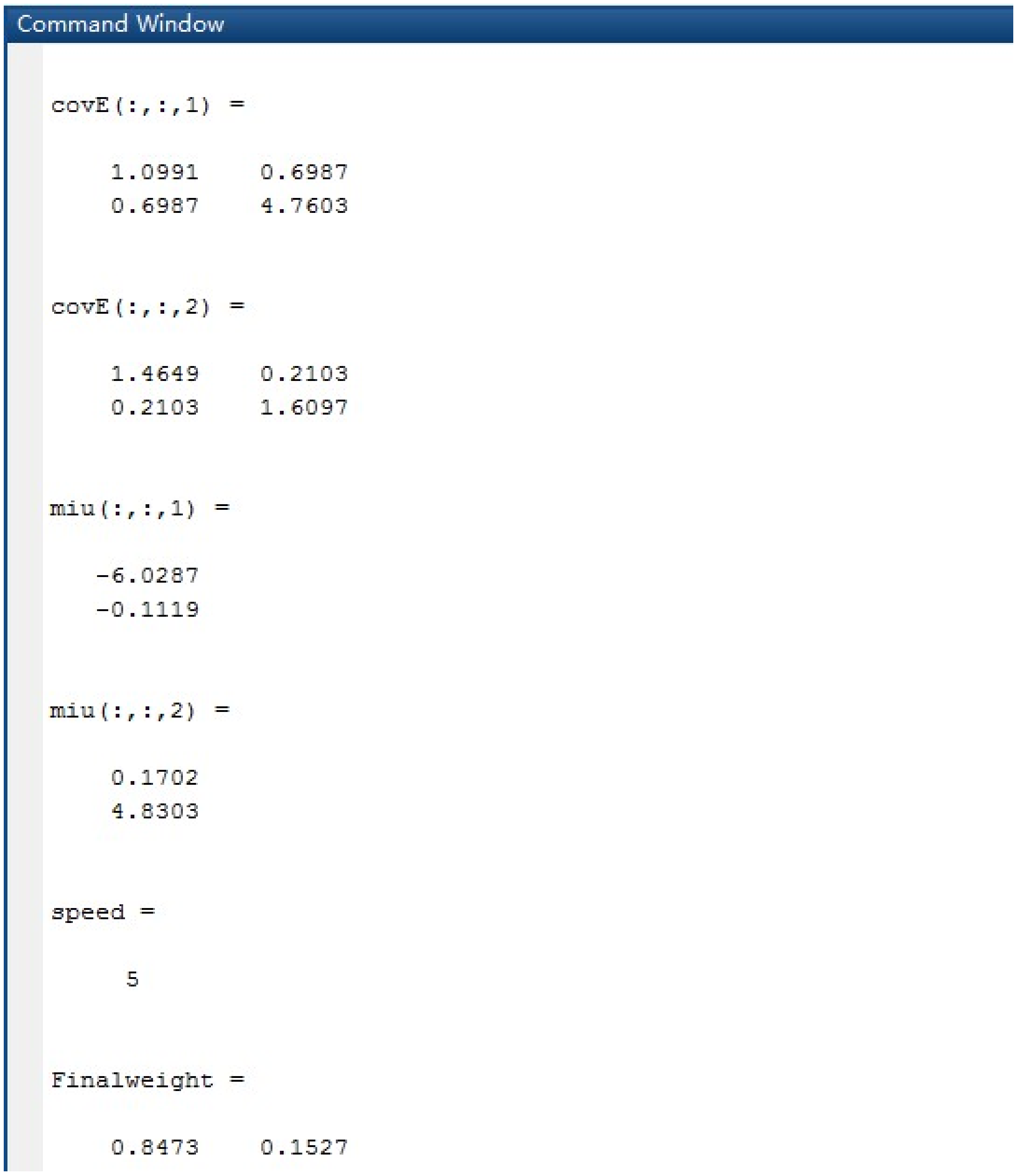




Final Plot





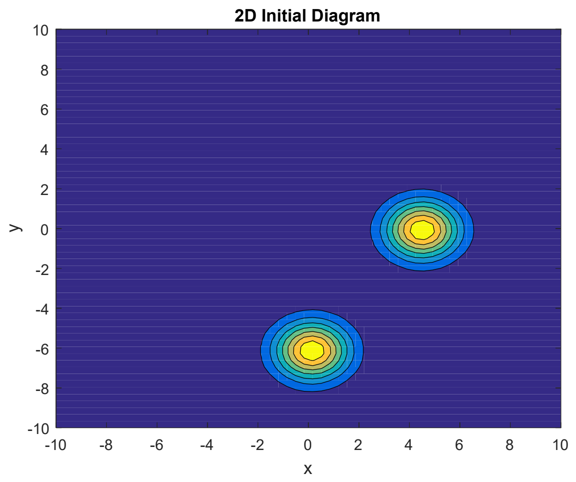

Command Window

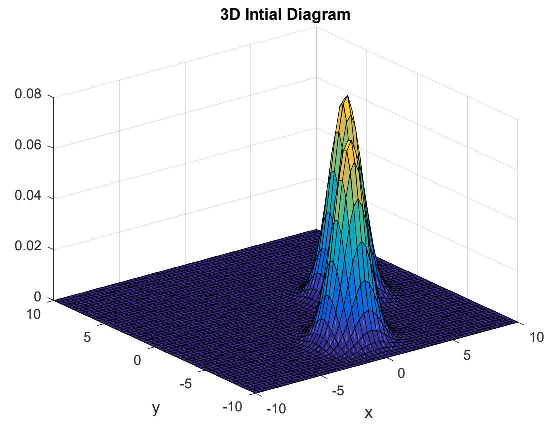
```

covE(:,:,1) =

    1.0991    0.6987
    0.6987    4.7603


covE(:,:,2) =

    1.4649    0.2103
    0.2103    1.6097


miu(:,:,1) =

   -6.0287
   -0.1119


miu(:,:,2) =

    0.1702
    4.8303


speed =

     5


Finalweight =

    0.8473    0.1527
```

5). Case-five: ellipsoidal subpopulations

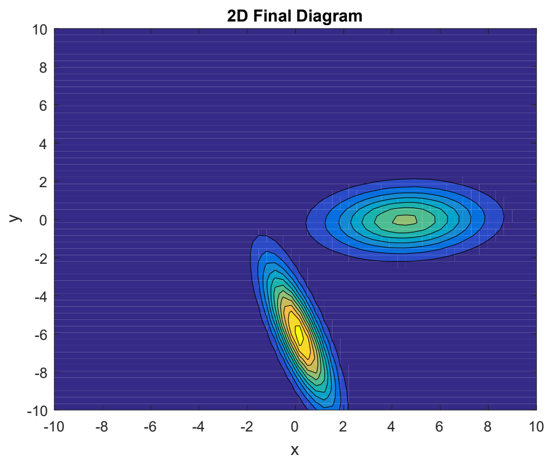
% sigma1=[1 -2; -2 6];
% sigma2=[4 0; 0 1];
% mu1=[0;5];
% mu2=[-6;0];
% weight1=0.25;
% weight2=0.75;
% weight=[0.25,0.75];

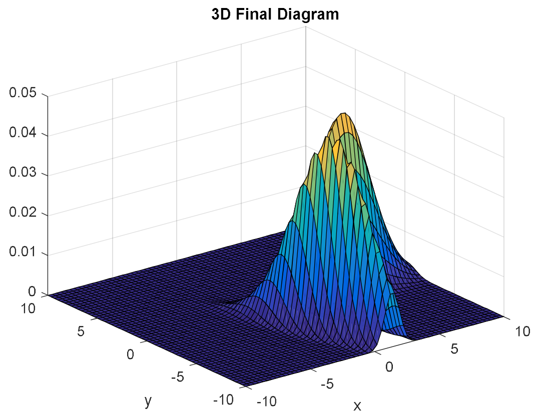
Initial Plot

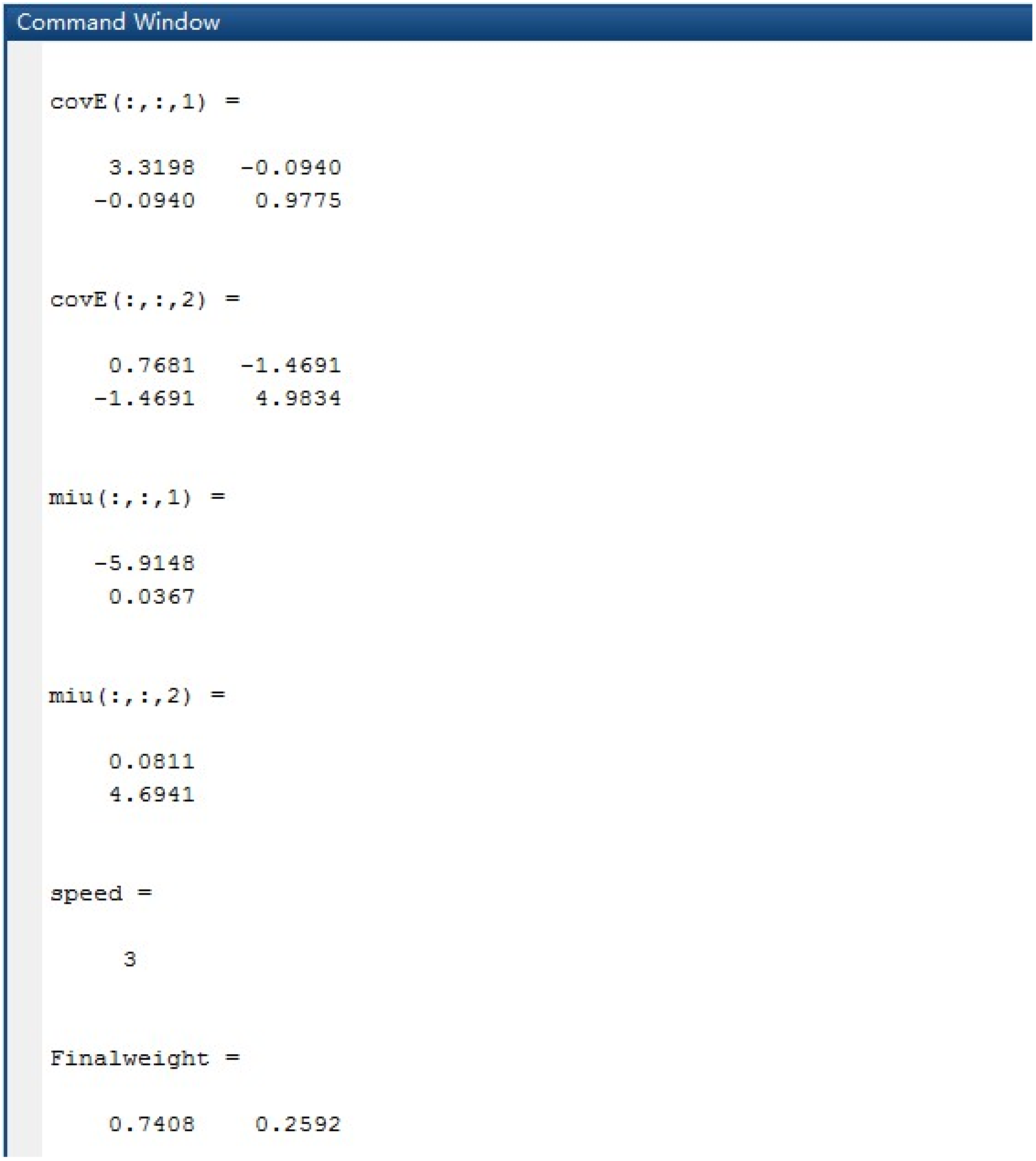




Final Plot







```

covE(:,:,1) =

    3.3198   -0.0940
   -0.0940    0.9775


covE(:,:,2) =

    0.7681   -1.4691
   -1.4691    4.9834


miu(:,:,1) =

   -5.9148
    0.0367


miu(:,:,2) =

    0.0811
    4.6941


speed =

     3


Finalweight =

    0.7408    0.2592
```

Finding:

1). It is shown in the results of case-one and case-two that given the same value of covariance matrix, weight but different value of mean matrix.

The well-separated and close subpopulations both generate the experimental covariance and mean matrix that are close the initial Gaussian mixture model pdf parameter. However, the speed of spherical subpopulations are more than ellipsoidal subpopulations.

2). It is shown in the results of the case-three and case-five that given the same value of mean matrix, weight but different value of covariance matrix.

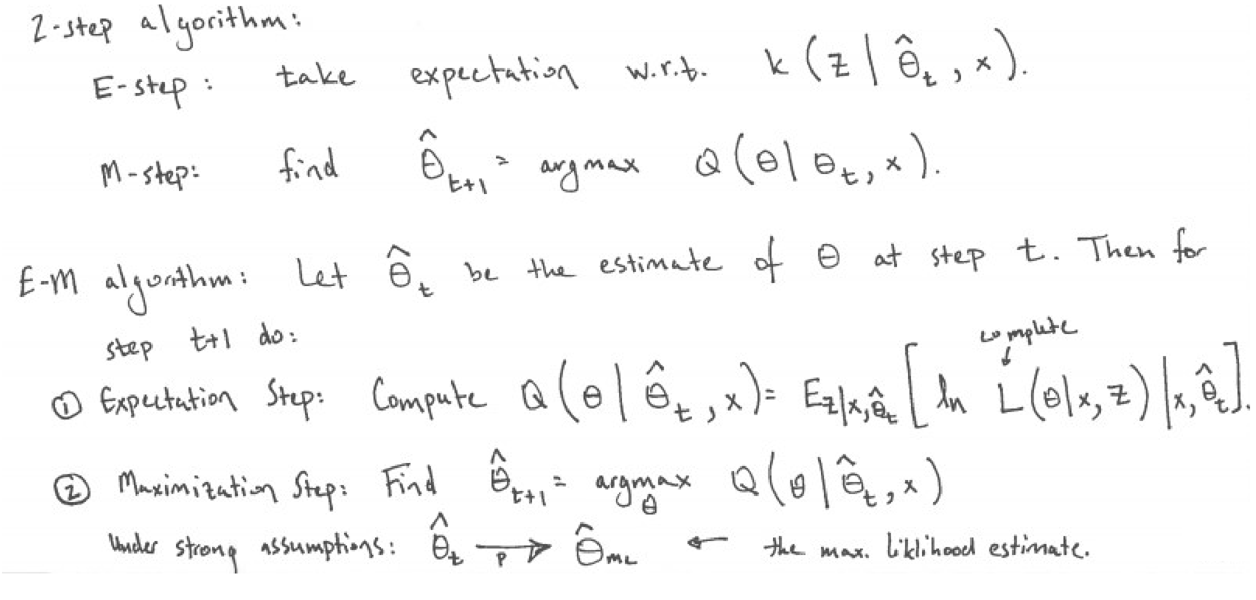
The close and well-separated subpopulations both generate the experimental covariance and mean matrix that are close the initial Gaussian mixture model pdf parameter. However, the speed of well-separated subpopulations is less than close subpopulations.

3). It is shown in the results of the case-three and case-four that given the same value of mean matrix and different value of covariance matrix, but different values of weights. The spherical subpopulations generate the experimental covariance and mean matrix that are close the initial Gaussian mixture model pdf parameter. However, the speed increase when the difference of each weight increases.

Experiment No.4

Description of Algorithm:

1). Expectation maximization algorithm (EM) for Iterative estimization θ_t of θ.



2-step algorithm:

E-step: take expectation w.r.t. $k(z \mid \hat{\theta}_t, x)$.

M-step: find $\hat{\theta}_{t+1} = \arg\max \; Q(\theta \mid \theta_t, x)$.

E-M algorithm: Let $\hat{\theta}_t$ be the estimate of θ at step t. Then for step $t+1$ do:

① Expectation Step: Compute $Q(\theta \mid \hat{\theta}_t, x) = E_{z|x,\hat{\theta}_t}\left[\ln \overset{\text{complete}}{L}(\theta \mid x, z) \,\middle|\, x, \hat{\theta}_t\right]$.

② Maximization Step: Find $\hat{\theta}_{t+1} = \underset{\theta}{\arg\max} \; Q(\theta \mid \hat{\theta}_t, x)$

Under strong assumptions: $\hat{\theta}_t \xrightarrow{P} \hat{\theta}_{ML}$ ← the max. liklihood estimate.

2). The k-means clustering method.

Given the n-dimensional date points $\{x_i\}_{i=1}^{n}$, and the predetermined number of cluster k,

$y_i = A(x_i) \in \{1,2,\ldots,k\}, \forall \mathrm{i}$

In order to partition the date point with means of each cluster $c_j, j \in \{1,2,\dots,k\}$.

First, assign a point x_i to the cluster with the closest center c_j.

$$y_i = argmin_j \left\| x_i - c_j \right\|^2$$

Second, update the center of each cluster by calculating the mean

$$c_j = \frac{1}{nj} \sum\nolimits_{i \in \{y_i | j\}} x_i$$

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

$$J = \sum\nolimits_{j=1}^{k} \sum\nolimits_{i \in \{y_i | j\}} \left\| x_i - c_j \right\|^2$$

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

Description of Method:

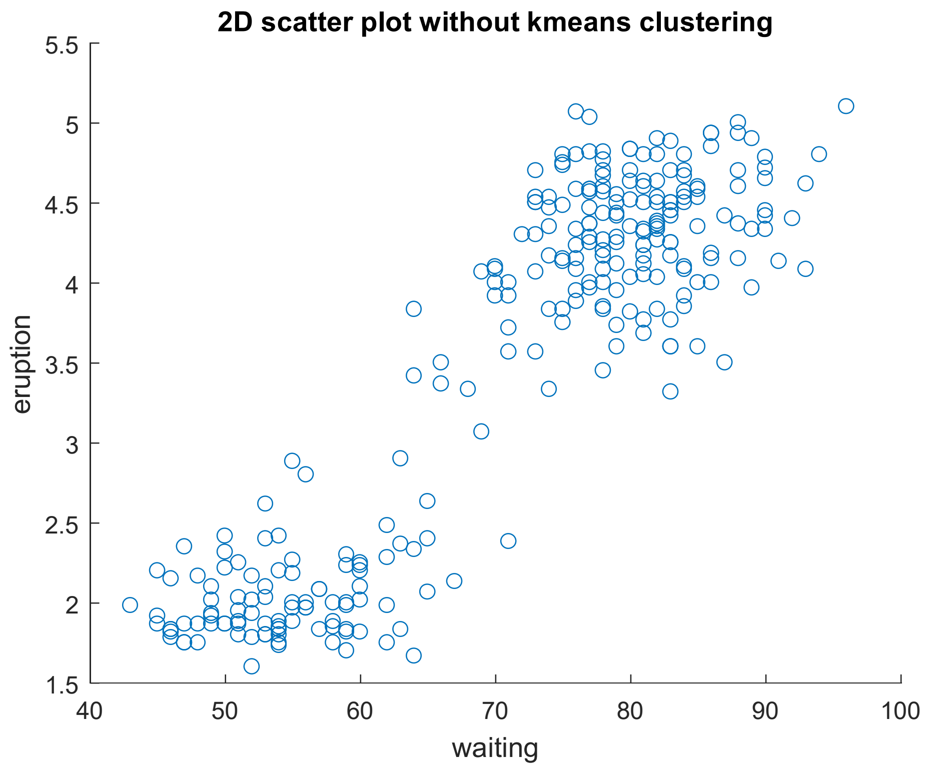
The k-means cluster method is used in the lab4 that process the data of geyser. The GMM-EM algorithm is used and the scatterplot of the data set is generated to show the sample. The contour plot of the final GMM pdf is overly the plot to estimate the cluster of data.

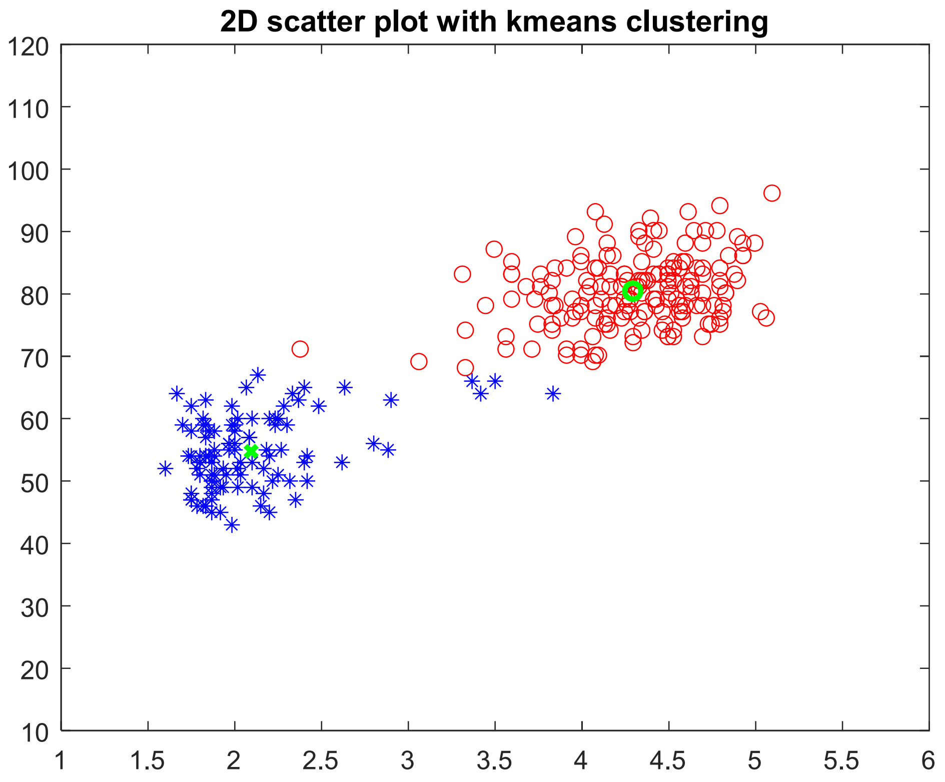
Code:

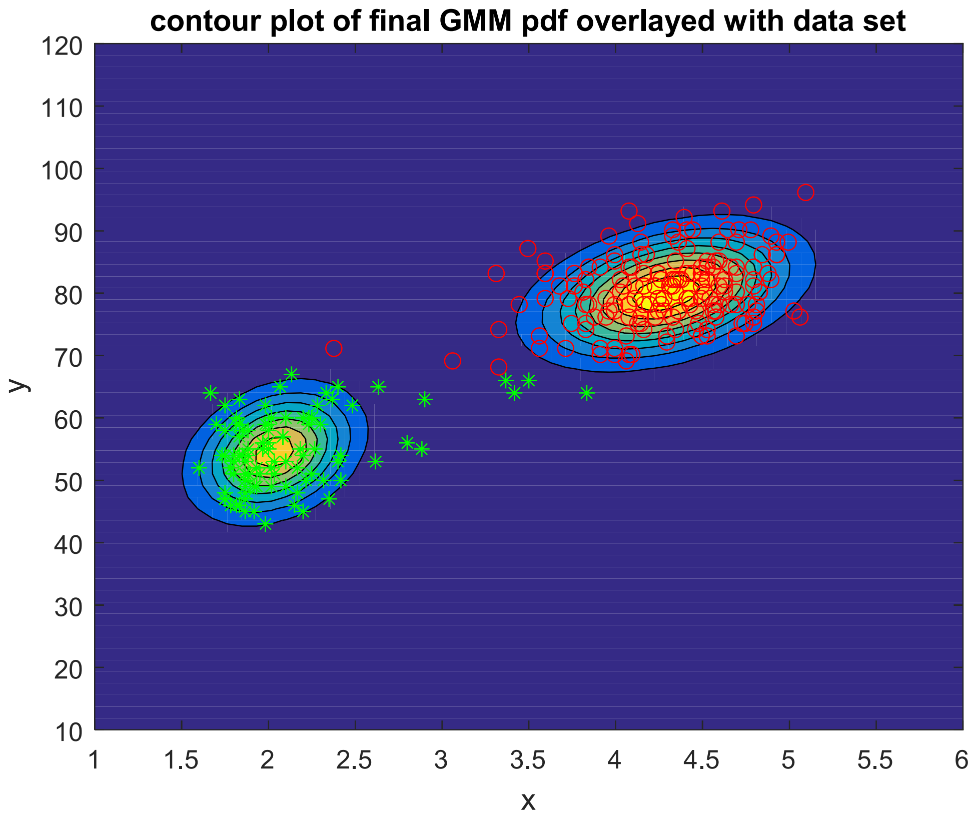
```
clc; clear; close all;

%Import the data txt
s = importdata('faithful.dat1.txt');
waiting = s(:,3);
eruption = s(:,2);
%Generate figure without kmeans clustering
scatter(waiting,eruption);
xlabel('waiting');
ylabel('eruption');
title('2D scatter plot without kmeans clustering');
%Generate figure with kmeans clustering
m=[eruption,waiting];
N = size(m,1);
[y,C]=kmeans(m,2);
figure;
plot(m(y==1,1),m(y==1,2), 'b*');
hold on;
plot(m(y==2,1),m(y==2,2), 'ro');
plot(C(1,1),C(1,2), 'gx','LineWidth',2);
plot(C(2,1),C(2,2), 'go','LineWidth',2);
title('2D scatter plot with kmeans clustering');
axis([1 6 10 120]);
hold off;
miu1=C(1,:);
miu2=C(2,:);
%Obtain the centers of two subpopulations and set as the first estimation
```

```
%of mean.
miu=zeros(2,1,2);
miu(:,:,1)=miu1';
miu(:,:,2)=miu2';
%Set the initial guess of covariance matrix.
covE=zeros(2,2,2);
covE(:,:,1)=[1 0;0 1];
covE(:,:,2)=[1 0;0 1];
weight = [.5 .5];
SH=0;
%Compute the initial log-likelihood.
for i=1:N

SH=SH+log(weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m
(i,:),miu(:,:,2)',covE(:,:,2)));
end
L=SH/N;

%E-steps
Nor=zeros(N,2);
n=zeros(1,2);
Lnew=0;
counter=0;
while abs(L-Lnew)>0.001
    L=Lnew;
    for j=1:2
        for i=1:N
            SP=0;
            for z=1:2

SP=weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m(i,:),m
iu(:,:,2)',covE(:,:,2));
            end
            Nor(i,j)=weight(j)*mvnpdf(m(i,:),miu(:,:,j)',covE(:,:,j))/SP;
        end
    end
    n(1)=sum(Nor(:,1));
    n(2)=sum(Nor(:,2));
    %M-steps
    weight=n/N;
    for j=1:2
        miu(:,:,j)=(Nor(:,j)'*m/n(j))';
        Acov=zeros(2,2);
        for i=1:N
            Acov=Acov+Nor(i,j)*(m(i,:)'-miu(:,:,j))*(m(i,:)'-miu(:,:,j))';
        end
        covE(:,:,j)=Acov/n(j);
    end

    SH=0;
    for i=1:N

SH=SH+log(weight(1)*mvnpdf(m(i,:),miu(:,:,1)',covE(:,:,1))+weight(2)*mvnpdf(m
(i,:),miu(:,:,2)',covE(:,:,2)));
    end
    Lnew=SH/N;
    counter=counter+1;

end

%Draw the diagram for the final result
A=reshape(covE(:,:,1),[2 2]);
B=reshape(covE(:,:,2),[2 2]);
C = reshape(miu,[2 2]);
covEE=cat(3,A,B);
obj = gmdistribution(C',covEE,weight);
figure;
ezcontourf(@(x,y) pdf(obj,[x y]),[1 6],[10 120]);
title('contour plot of final GMM pdf overlayed with data set');
hold on;
%Overlay the contour plot with data set
plot(m(y==1,1),m(y==1,2), 'g*');
plot(m(y==2,1),m(y==2,2), 'ro');
hold off;
figure;
ezsurf(@(x,y)pdf(obj,[x y]),[1 6],[10 120]);
title('3D Final Diagram');
```

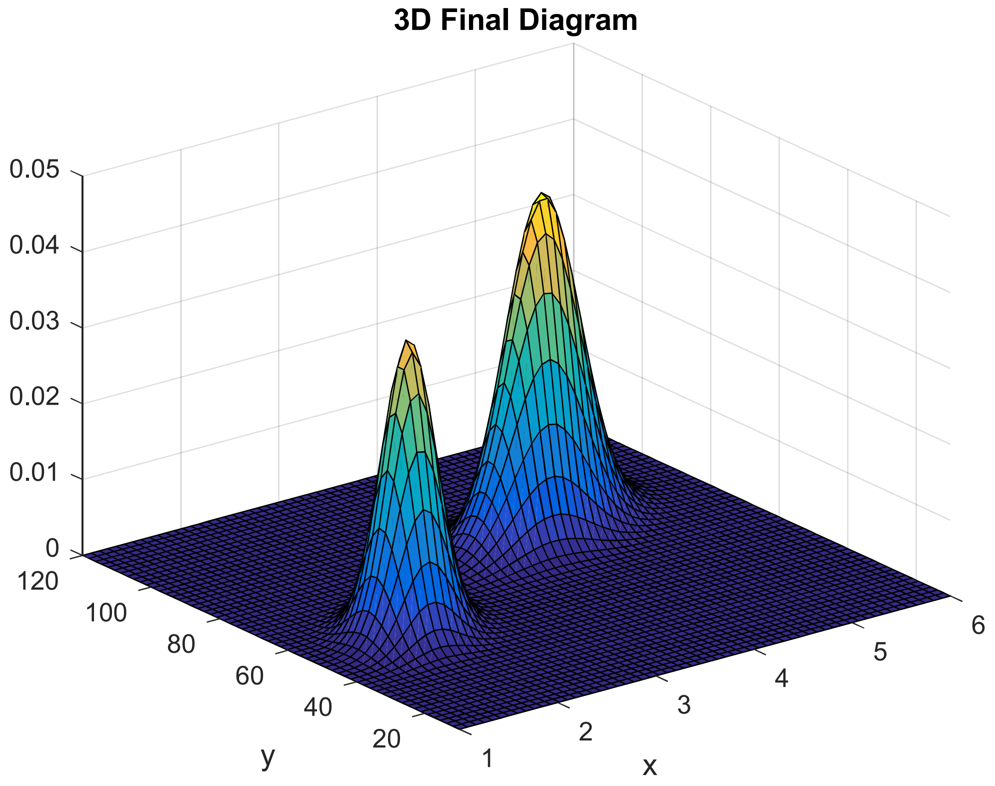
Simulation Result:







contour plot of final GMM pdf overlayed with data set
y
x



3D Final Diagram
y
x

Finding:

It is shown in the results 2-D scatter plots that the data of eruption and waiting time are separated using K-means clustering method. It can be seen from the result that the contour plot of final GMM pdf matches the scatterplot of the data set well. In conclusion, the GMM probability density function can be used to estimate the cluster of data.

Conclusion

Overall, the simulations of three experiments focus on expectation maximization theory. The random number generator is used to generate random vectors follows the multivariate Gaussian distribution. The second trial generates random variables with mixture distribution and the comparison is conducted with the theoretical probability density function. The third experiment simulates different Gaussian mixture models and the probability parameters are analyzed using expectation maximization method. The last experiment is finished by using k-means clustering method theory to generate 2-D scatter data figure and fit the dataset using GMM-EM algorithm. In conclusion, the project uses mathematical methods to process random samples and solve problems with different probability distributions.