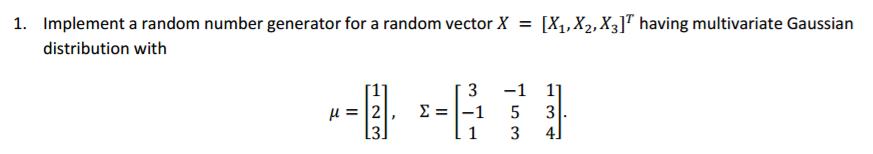
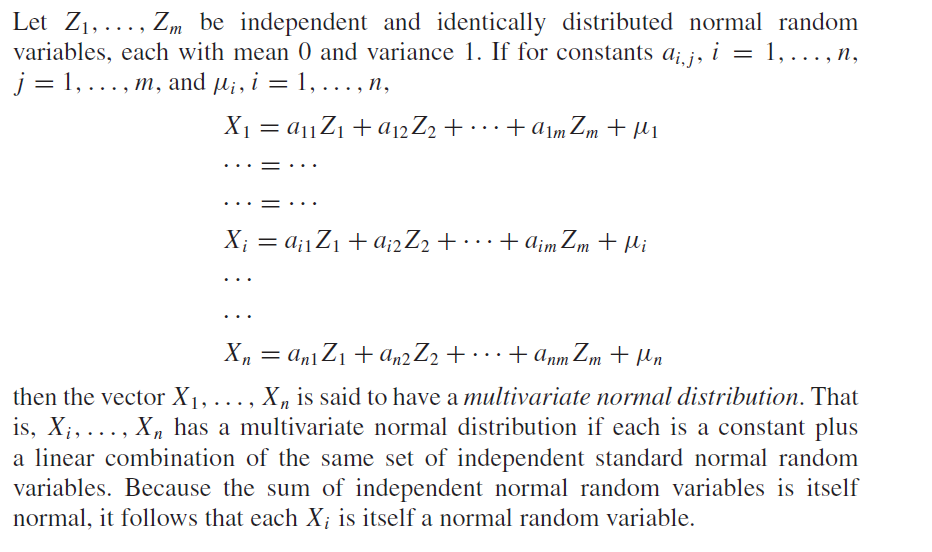
Project #7 – Expectation Maximization

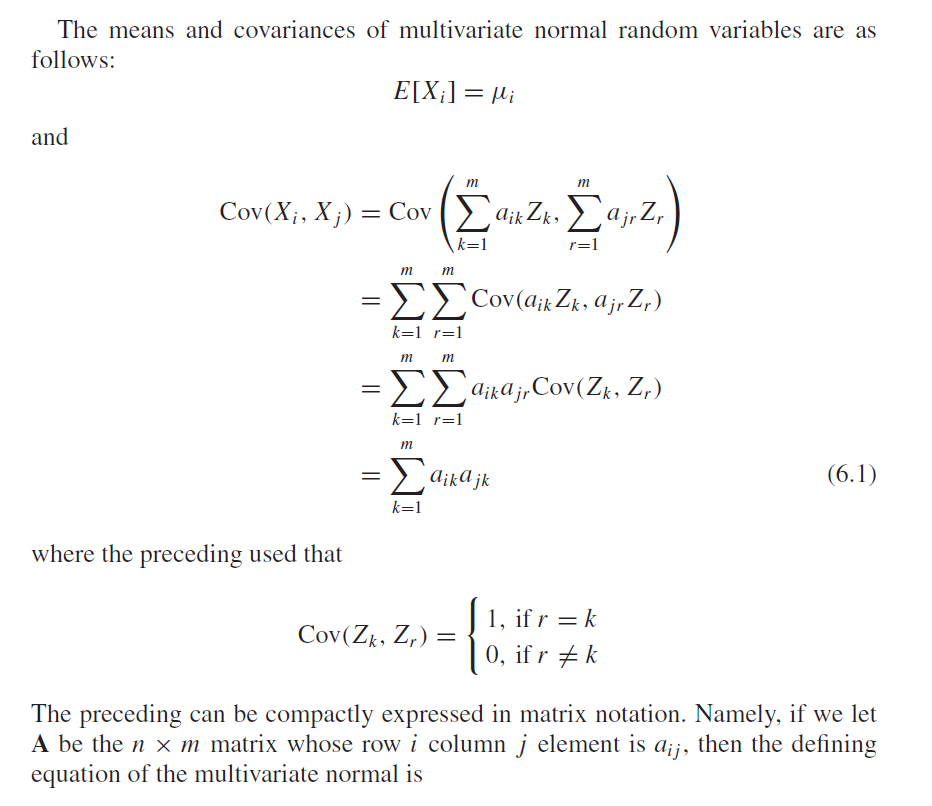
**Concept behind this problem:**

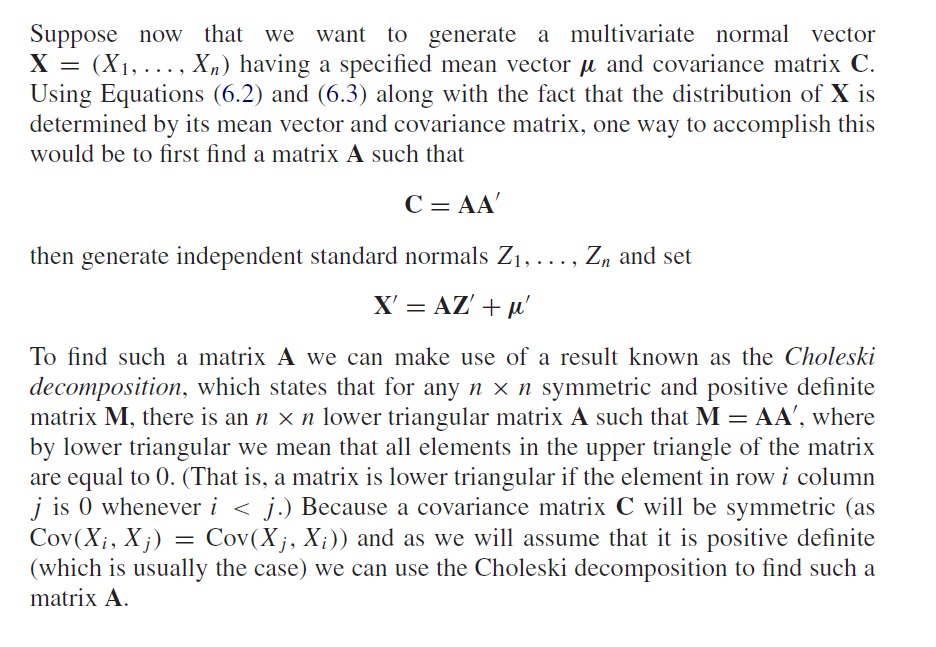
The question requires the knowledge of ***Multivariate Gaussian Distribution***.

The multivariate normal (MV-N) distribution is a multivariate generalization of the one-dimensional normal distribution. In its simplest form, which is called the "standard" MV-N distribution, it describes the joint distribution of a random vector whose entries are mutually independent univariate normal random variables, all having zero mean and unit variance. In its general form, it describes the joint distribution of a random vector that can be represented as a linear transformation of a standard MV-N vector.

It is a generalization of the one-dimensional [normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) to higher dimensions. One possible definition is that a [random vector](https://en.wikipedia.org/wiki/Random_vector) is said to be k-variate normally distributed if every [linear combination](https://en.wikipedia.org/wiki/Linear_combination) of its k components has a univariate normal distribution. Its importance derives mainly from the [multivariate central limit theorem](https://en.wikipedia.org/wiki/Central_limit_theorem#Multidimensional_CLT). The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) [correlated](https://en.wikipedia.org/wiki/Correlation_(statistics)) real-valued [random variables](https://en.wikipedia.org/wiki/Random_variable) each of which clusters around a mean value







**Reference:**

**[1] Piazza**

**[2] Chp-6 Multivariate Normal Distribution and Copulas from Simulation by Sheldon Ross**

**Code & Code Description:**

* Firstly, with the given mean and covariance matrix, we have to calculate the value of X with the equation mentioned below.

C:\Users\Harish\Pictures\SnippingTool++\Uploads\upload(147).png

* Before using the above equation, we calculate the value of Z which is a standard normal matrix using normrnd function.

%%Initialize

given\_mean=[1 2 3];

given\_cov\_matrix=[ 3 -1 1

-1 5 3

1 3 4];

matrix\_A=chol(given\_cov\_matrix,'lower');%choleski function

for iter=1:No\_of\_samples

z1\_value=normrnd(0,1);

z2\_value=normrnd(0,1);

z3\_value=normrnd(0,1);

ind\_std\_norm\_Z=[z1\_value z2\_value z3\_value];

norm\_vec\_X=matrix\_A\*ind\_std\_norm\_Z.'+given\_mean.';

Vector\_X(iter,:)=norm\_vec\_X; end

* We then calculate the values of Mean and Covariance of X1,X2 and X3 for ‘n’ samples.

disp(['Statistics of ',num2str(No\_of\_samples),' samples'])

X1\_Mean=mean(Vector\_X(:,1));

X2\_Mean=mean(Vector\_X(:,2));

X3\_Mean=mean(Vector\_X(:,3));

disp(['The Sample mean of X1 = ',num2str(X1\_Mean),' X2 = ',num2str(X2\_Mean),' X3 = ',num2str(X3\_Mean)]);

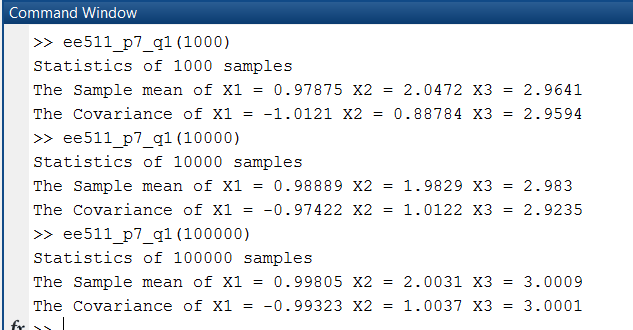
cov\_X12=cov(Vector\_X(:,1),Vector\_X(:,2));

cov\_X13=cov(Vector\_X(:,1),Vector\_X(:,3));

cov\_X23=cov(Vector\_X(:,2),Vector\_X(:,3));

disp(['The Covariance of X1 = ',num2str(cov\_X12(1,2)),' X2 = ',num2str(cov\_X13(1,2)),' X3 = ',num2str(cov\_X23(1,2))]);

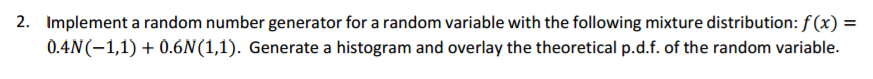
**Results and Theoretical Values:**



**Observations:**

* Compute the sample covariance matrix and sample mean shows that we are sampling for the correct distribution.
* Since the sample mean and covariance is almost the same, Choleski decomposition method is effective in generating in multivariate gaussian.
* Since the matrix matrix\_A is positive definite matrix, Choleski decomposition is real.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**Concept behind this problem:**

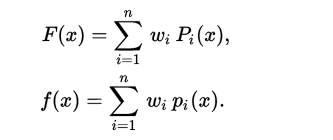
The problem uses the concept of **Mixture Distributions**.

A mixture distribution is the probability distribution of a random variable that is derived from a collection of other random variables as follows: first, a random variable is selected by chance from the collection per given probabilities of selection, and then the value of the selected random variable is realized. The underlying random variables may be random real numbers, or they may be random vectors (each having the same dimension), in which case the mixture distribution is a multivariate distribution.

The individual distributions that are combined to form the mixture distribution are called the mixture components, and the probabilities (or weights) associated with each component are called the mixture weights. The number of components in mixture distribution is often restricted to being finite, although in some cases the components may be countably infinite.

A distinction needs to be made between a random variable whose distribution function or density is the sum of a set of components (i.e. a mixture distribution) and a random variable whose value is the sum of the values of two or more underlying random variables, in which case the distribution is given by the convolution operator. The distributions can be made up of different distributions (e.g. a normal distribution and a t-distribution) or they can be made up of the same distribution with different parameters.

So, for a distribution f = p\*N(x1,y1) + (1-p)\*N(x2,y2), it means that the probability that the N(x1,y1) occurs is p, while the probability that N(x2,y2) occurs is (1-p). We can generate uniform variable (0,1) to decide which one we should choose.



**Reference:**

**[1] Piazza**

**Code and Description:**

* Firstly, generate random variable for selection between any of the two random variables.
* Sample it for ‘n’ times and store it in a vector (where n is 100,1000,10000 etc…).
* Since the given equation 0.4N(-1,1) + 0.6N(1,1) , the fx is either N(-1,1) or N(1,1)
* Then I plot the histogram of the vector and overlay it over the theoretical PDF of the equation (calculated via normrnd inbuilt function of Matlab).

yyaxis right

xaxis=-3:0.1:3;

theo\_norm\_pdf=0.4\*normpdf(xaxis,-1,1)+0.6\*normpdf(xaxis,1,1);

plot(xaxis,theo\_norm\_pdf);

ylabel('Theoretical Normal PDF of the mixture distribution');

end

yyaxis left

hist(mix\_pdf,25);

title('Theoretical pdf generated over the histogram for the mixture distribution f(x)');

xlabel('Range of X');

ylabel('Frequency');

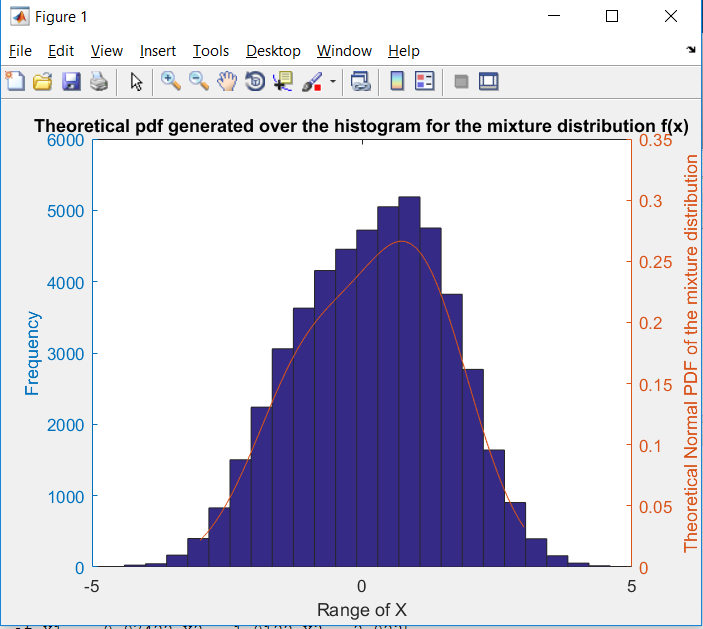
% -- Main logic

for iter=1:no\_of\_samples

r\_number = rand <= 0.4;

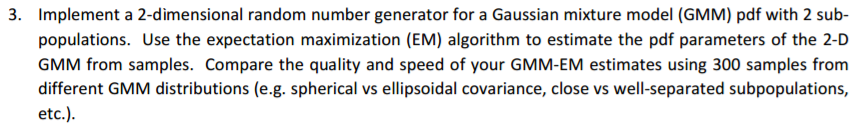
mix\_pdf(:,iter)=r\_number.\*normrnd(-1,1)+(1-r\_number).\*normrnd(1,1);end

**Results and Plots:**



**Observations:**

* The histogram plot is observed to be well fitted within the theoretical pdf curve.
* With increase in sample size as well as change in the histogram nbins, we see that the graph gets more and more fitted into the theoretical pdf.



**Concept behind this problem:**

The problem uses the concept of **Expectation Maximization and Gaussian Mixture**

**Models.**

**Expectation Maximization (EM):**

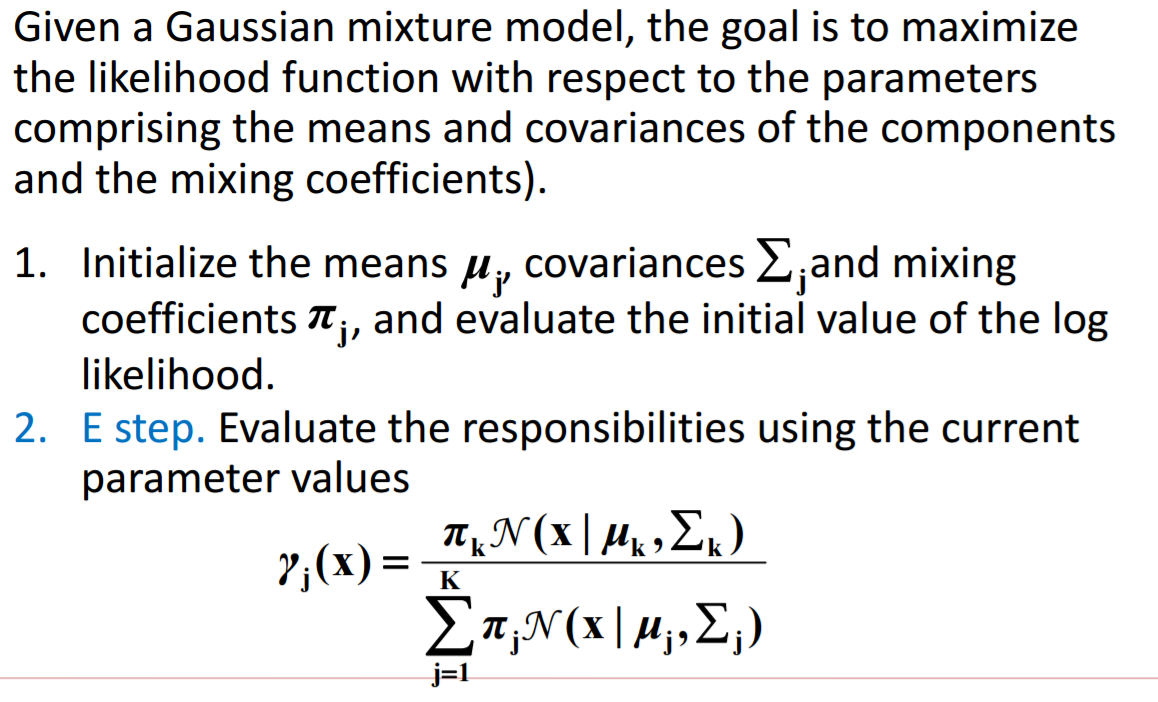
The expectation maximization algorithm is a refinement on this basic idea. Rather than picking the single most likely completion of the missing coin assignments on each iteration, the expectation maximization algorithm computes probabilities for each possible completion of the missing data, using the current parameters θ ˆ(t). These probabilities are used to create a weighted training set consisting of all possible completions of the data. Finally, a modified version of maximum likelihood estimation that deals with weighted training examples provides new parameter estimates, θ ˆ(t+1). By using weighted training examples rather than choosing the single best completion, the expectation maximization algorithm accounts for the confidence of the model in each completion of the data

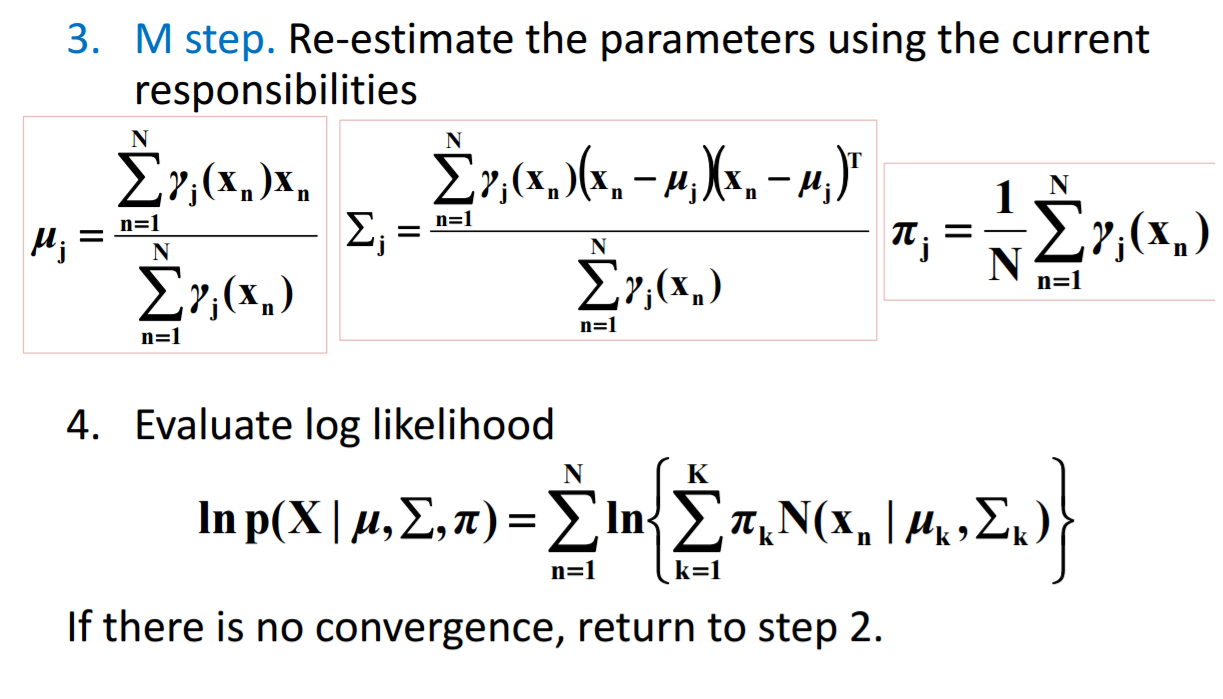
A mixture distribution is the probability distribution of a random variable that is derived from a collection of other random variables

**Gaussian mixture models:**

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data and the centers of the latent Gaussians. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model.

**EM Algorithm for GMM:**





**Reference:**

**[1] Piazza**

**[2]**<https://d1b10bmlvqabco.cloudfront.net/attach/ixp26i0vk6s5dg/itbtlqvcf3q1mk/j1peboio37hi/EM_Demystified_An_ExpectationMaximization_Tutorial.pdf>

**[3]** <http://mccormickml.com/2014/08/04/gaussian-mixture-models-tutorial-and-matlab-code/>

**[4]** <http://www.cse.iitm.ac.in/~vplab/courses/DVP/PDF/gmm.pdf>

**Code & Code Description:**

* First we have mean and sigma of two sub populations and generate a rand with 300 samples (given in question) providing arguments of gmdistribution . This provides for GMM model.
* Also we use few inbuilt Matlab functions in this code such as
  + **gmdistribution.fit :** uses an Expectation Maximization (EM) algorithm to construct an object obj of the gmdistribution class containing maximum likelihood estimates of the parameters in a Gaussian mixture model with k components for data in the n-by-d matrix X, where n is the number of observations and d is the dimension of the data.
  + **Ezcontour** : plots the contour lines of fun(x,y) using the contour function.
  + **Ezsurf:** Easy-to-use 3-D colored surface plotter on 2D plot

mean = [0 5;2 0];

sigma\_matrix = cat(3,[2 0;0 1],[1 0;0 1]);

weights = [0.75,0.25];

sample\_obj = gmdistribution(mean,sigma\_matrix,weights);

sample\_gen\_matrix = random(sample\_obj,300);

subplot(2,2,1);

hold on

opt = statset('Display','final');

%EM approach to calculate the minimum number of iteration it takes to converge to two Gaussian distributions

max\_expectation = gmdistribution.fit(sample\_gen\_matrix,2,'Options',opt);

* Basically the gmdistribution.fit will lead to the number of iterations it took to converge to two distributions.
* By use of scatter data points from ezcounter, we can predict how much the data fits into a particular distribution using EM approach
* We repeat the above procedure for Spherical and Ellipsoidal covariance with close and well spread subpopulations (distributions)

% Plots the contour plot of two cluster data points

ezcontour(@(x,y)pdf(max\_expectation,[x y]),[-4 10],[-4 10]);

title('Scatter plot after convergence');

hold off

%3D plot of the Gaussian distribution

subplot(2,2,2);

pdf(max\_expectation,sample\_gen\_matrix);

ezsurf(@(x,y)pdf(max\_expectation,[x y]),[-5 10],[-5 10])

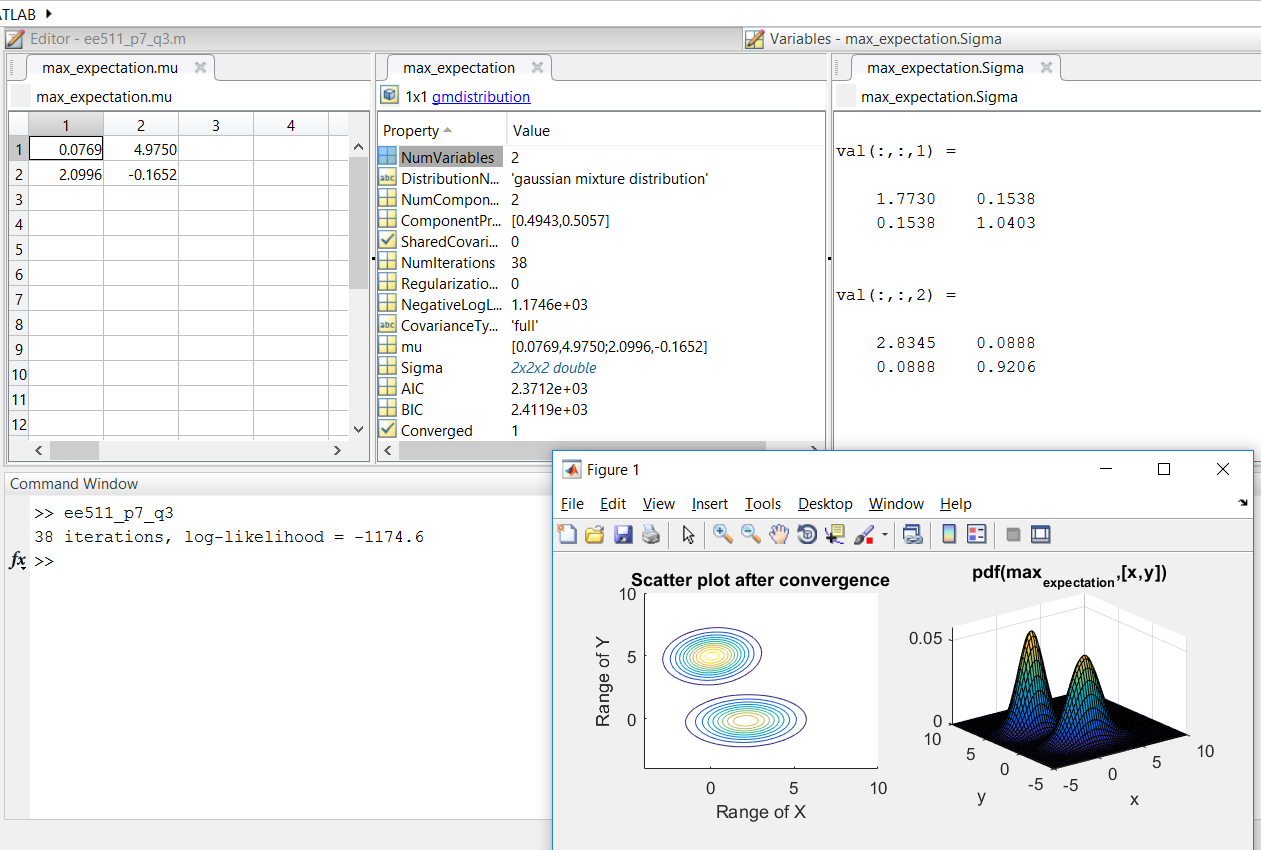
**Graphs and Plots**

1. **Ellipsoidal GMM (Symmetrical Weights) (CLOSELY SPACED)**

mean = [0 5;2 0];

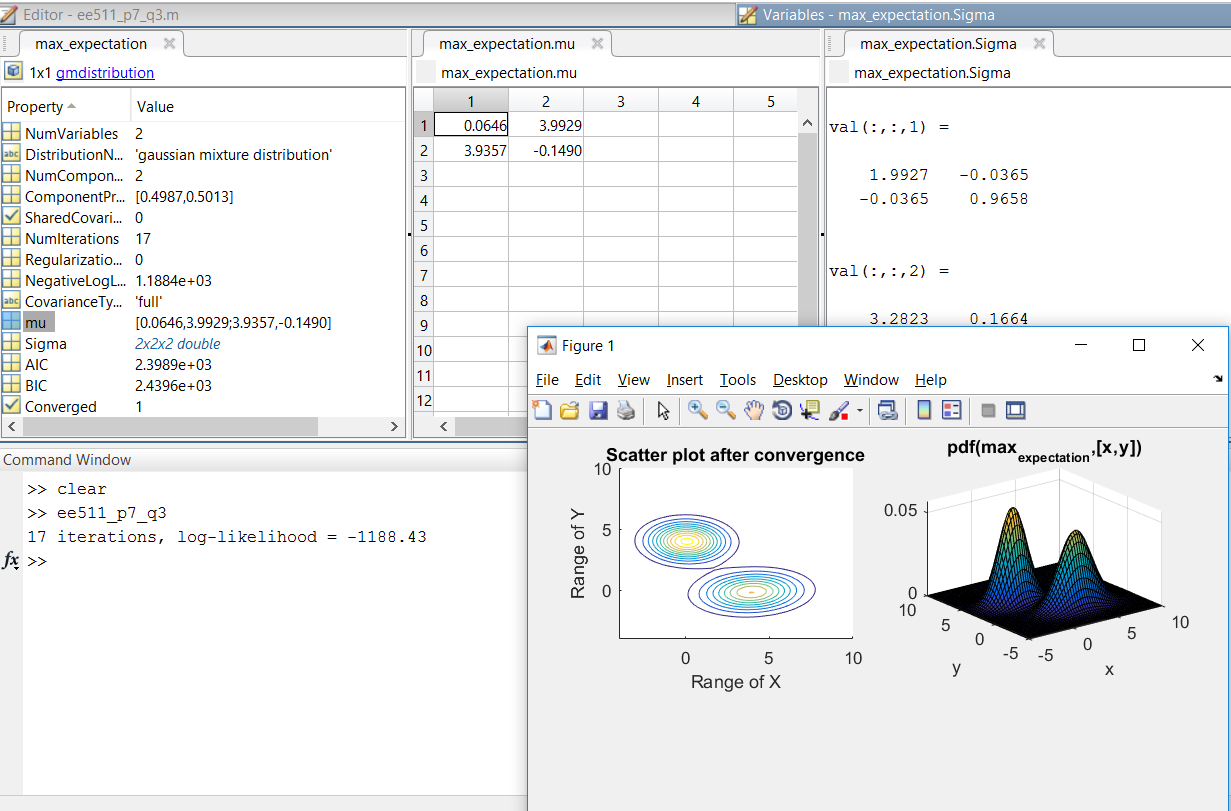
sigma\_matrix = cat(3,[2 0;0 1],[3 0;0 1]);

weights = [0.5,0.5];



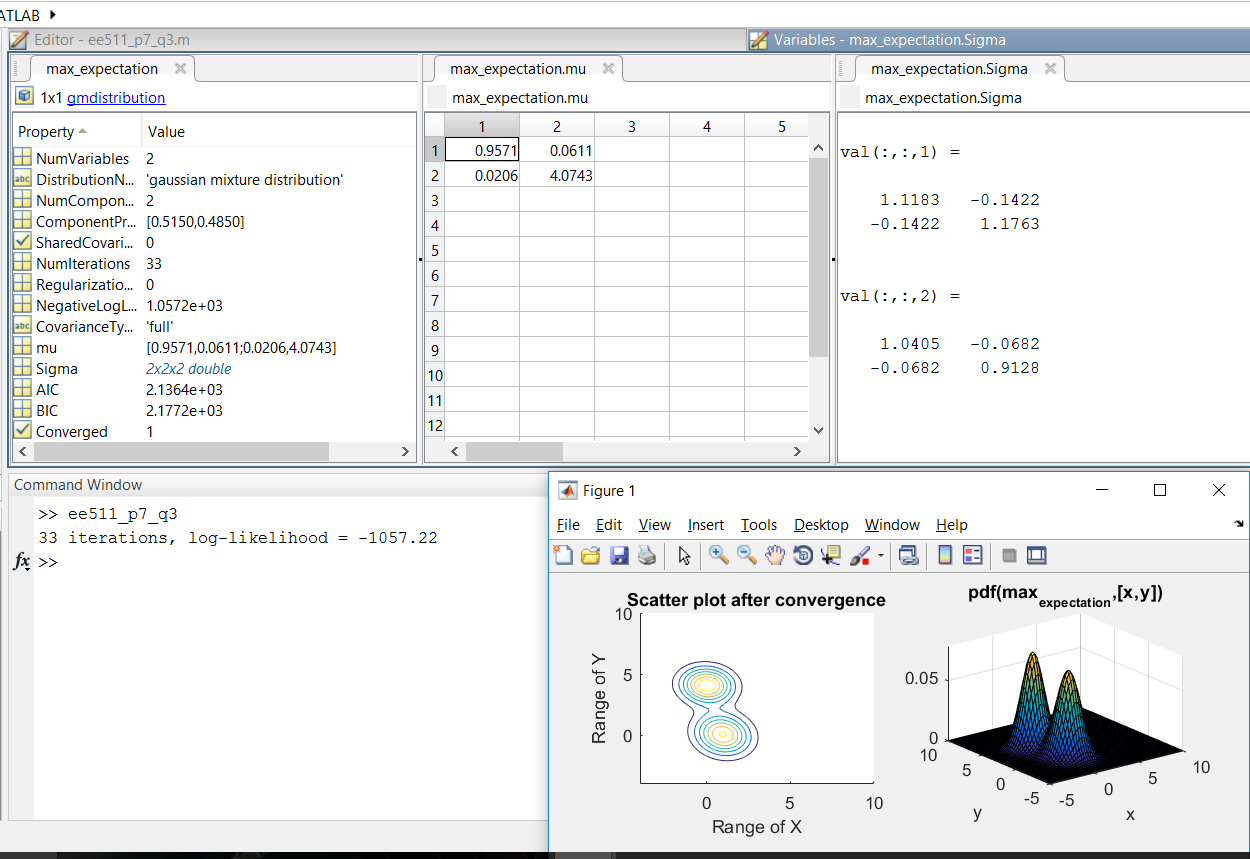
1. **Ellipsoidal GMM (Symmetrical Weights) (WIDELY SPACED)**

mean = [0 4;4 0];sigma\_matrix = cat(3,[2 0;0 1],[3 0;0 1]);weights = [0.5,0.5];



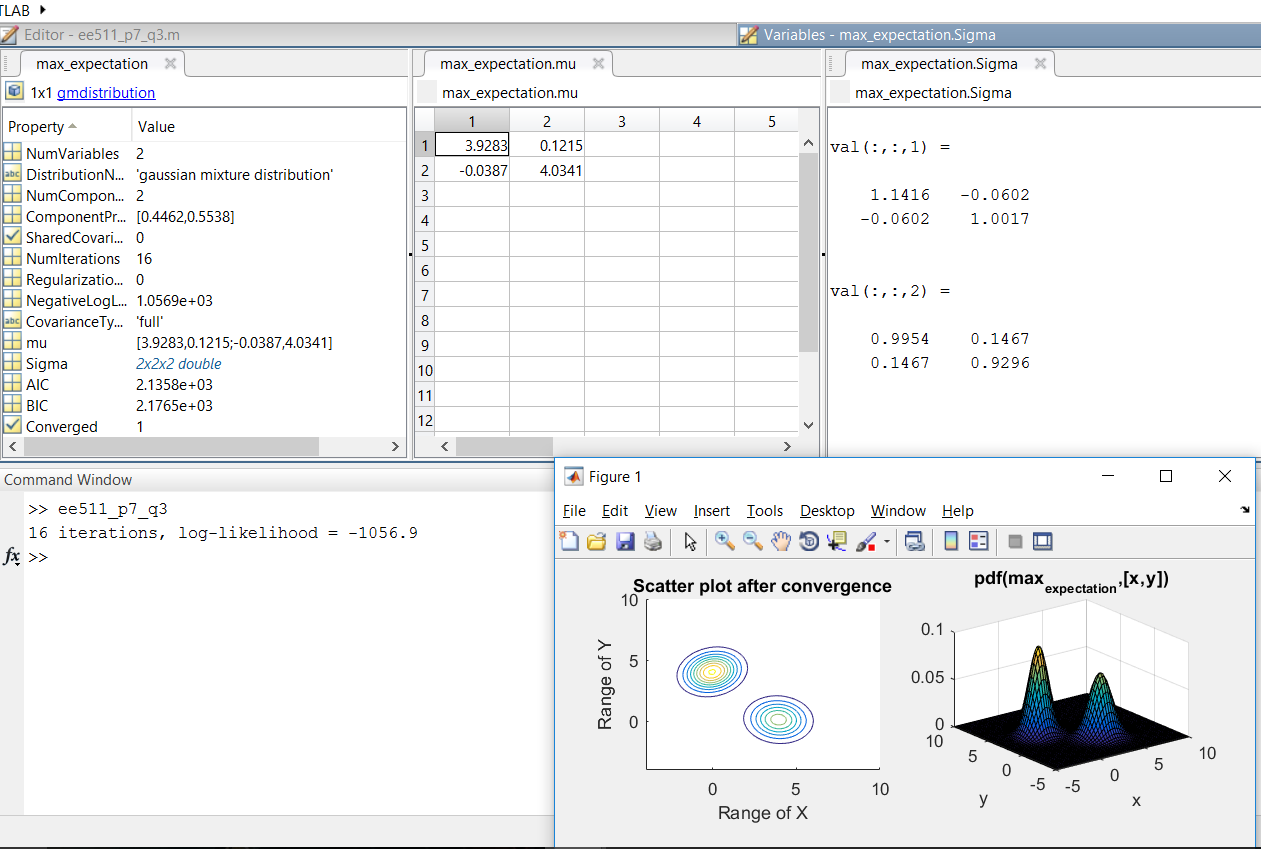
1. **Spherical GMM (Closely spaced) (Equal Weights)**

mean = [0 4;1 0]; sigma\_matrix = cat(3,[1 0;0 1],[1 0;0 1]); weights = [0.5,0.5];



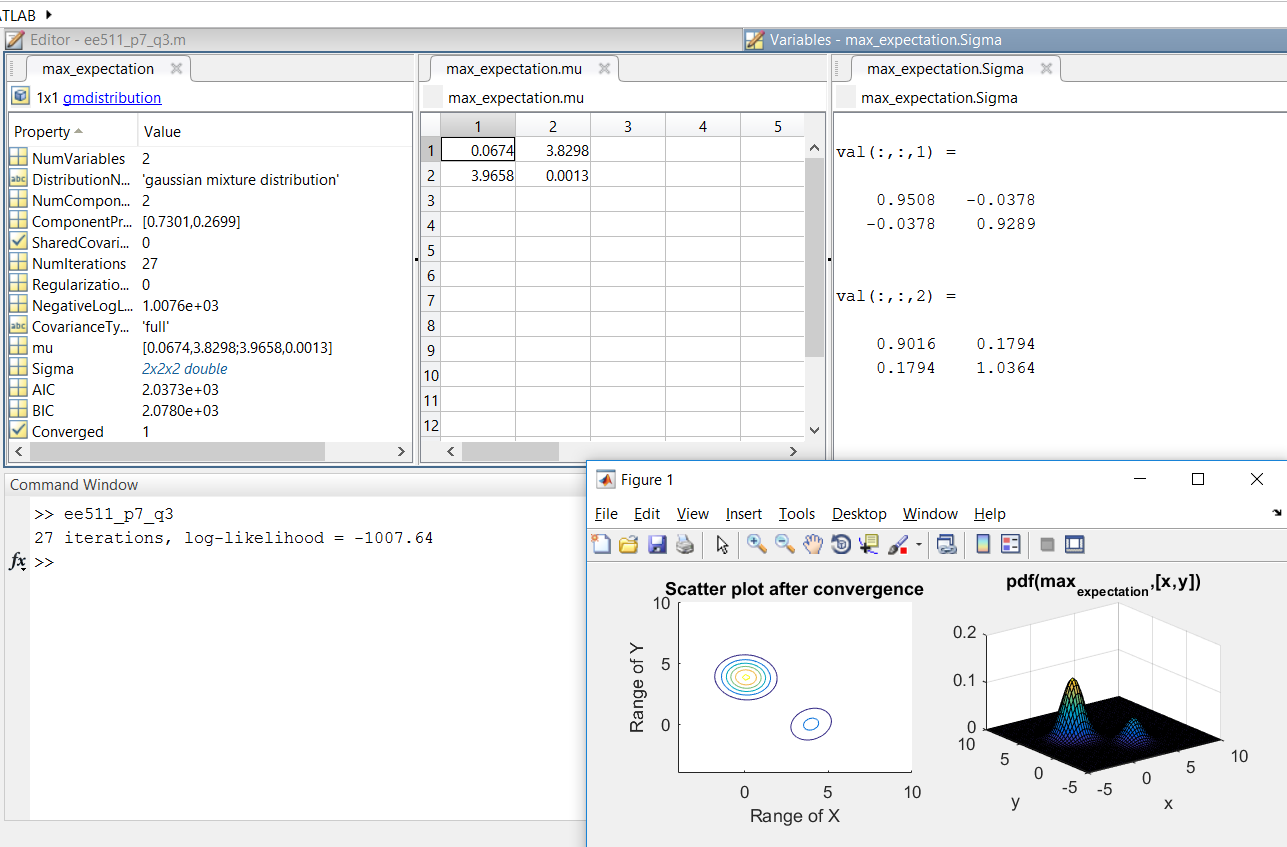
1. **Spherical GMM (Widely spaced) (Equal Weights)**

mean = [0 4;4 0];sigma\_matrix = cat(3,[1 0;0 1],[1 0;0 1]);weights = [0.5,0.5];



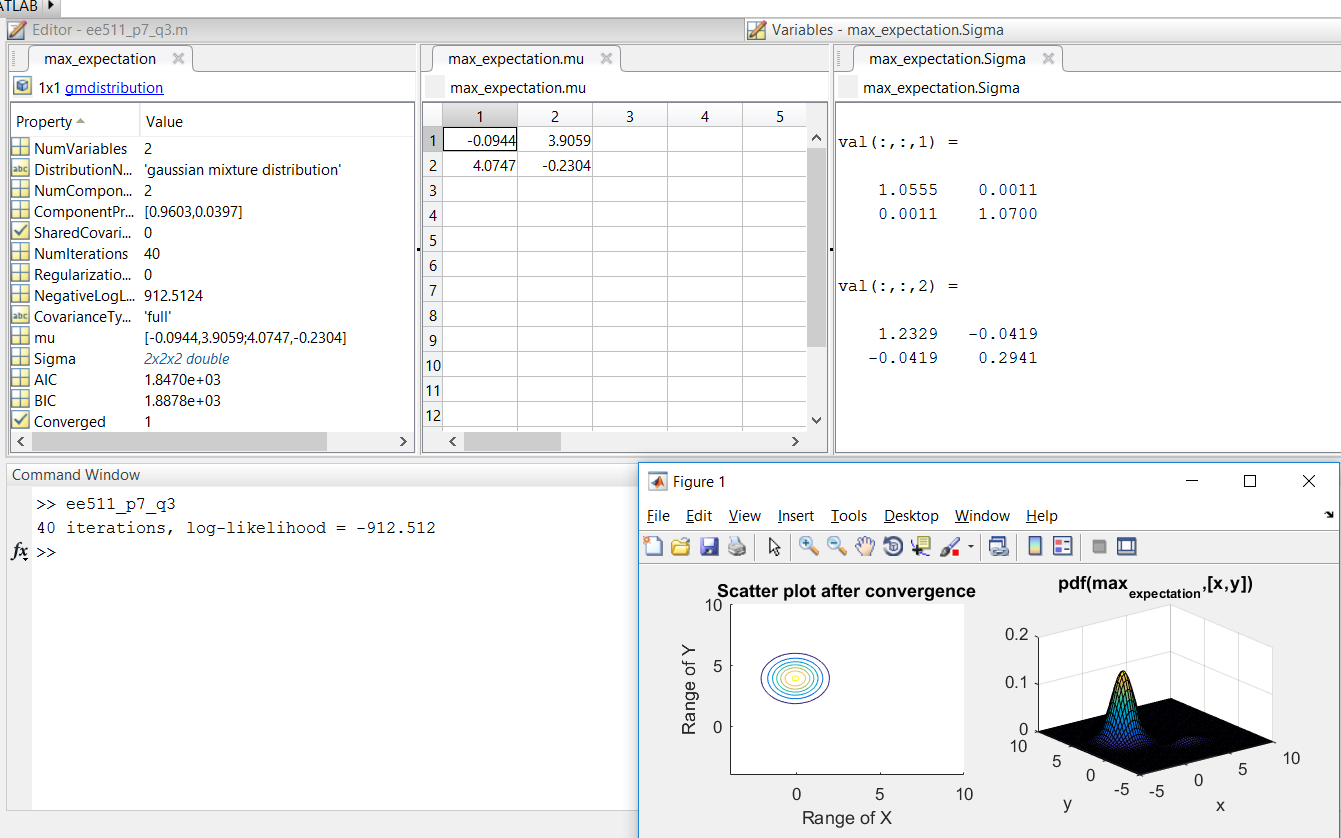
1. **Spherical GMM (Widely spaced) (Unequal Weights-Asymmetrical)**

mean = [0 4;1 0];sigma\_matrix = cat(3,[1 0;0 1],[1 0;0 1]);weights = [0.75,0.25];



1. **Spherical GMM (Widely spaced) (Unequal Weights-Highly Asymmetrical)**

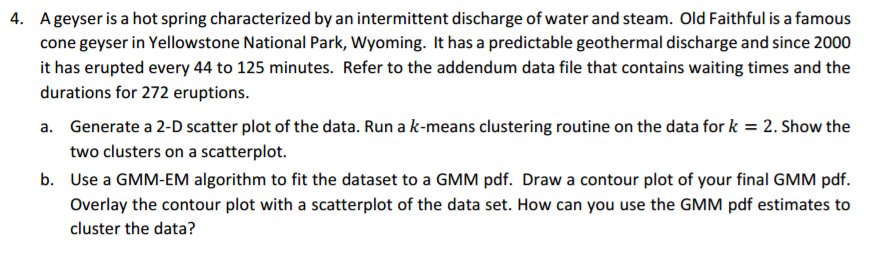
mean = [0 4;1 0];sigma\_matrix = cat(3,[1 0;0 1],[1 0;0 1]);weights = [0.95,0.05];



**Observations:**

|  |  |
| --- | --- |
| **Case** | **Number of iterations** |
| **Ellipsoidal GMM (Symmetrical Weights) (CLOSELY SPACED)** | **38** |
| **Ellipsoidal GMM (Symmetrical Weights) (WIDELY SPACED)** | **17** |
| **Spherical GMM (Closely spaced) (Equal Weights)** | **33** |
| **Spherical GMM (Widely spaced) (Equal Weights)** | **16** |
| **Spherical GMM (Widely spaced) (Unequal Weights-Asymmetrical)** | **27** |
| **Spherical GMM (Widely spaced) (Unequal Weights-Highly Asymmetrical)** | **40** |

* We can see the quality of the EM through the pdf parameters to the initial mean and sigma value we provided in the cluster.
* The 2D plots better show the fact that the initial cluster of points and the PDF of two sub populations fit into the Gaussian distribution.
* The calculated PDF parameters via EM method is close to the mean and sigma of the distribution used to generate the data points.
* By varying the spacing of data points, we observe that the number of iterations EM takes to reach convergence is changing.
* Widely spaced data points take less iterations to converge than closely spaced data points as shown in the above plots.

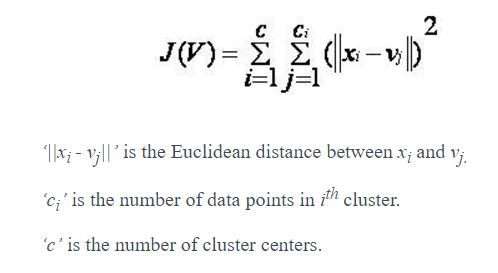


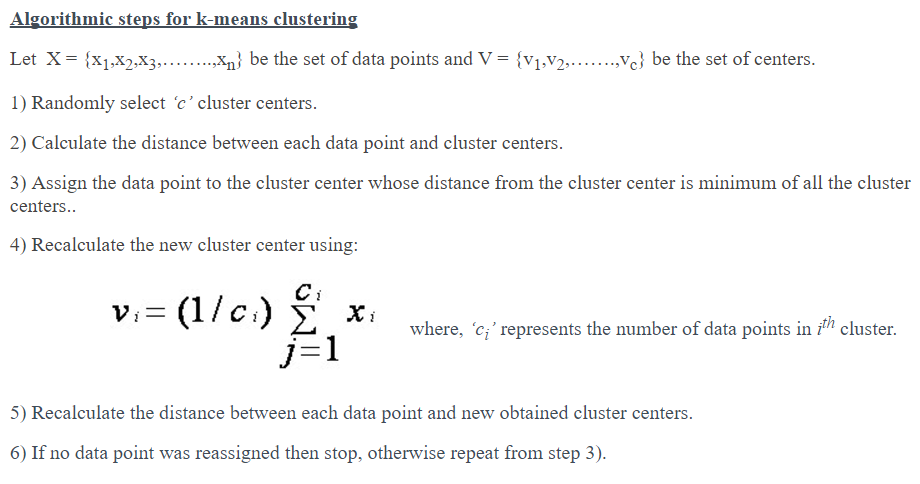
**Concept behind this problem:**

The problem uses the concept of **k-means clustering.**

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

k-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:





**Reference:**

**[1] Piazza**

**[2]** <https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm>

**[3]** <https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html>

**Code and Description:**

* Firstly, I load the data of eruption waiting times from a .txt file present in a local folder using **fullfile** builtin function.

max\_data = 272;

%Loading faithful.txt file into the matlab

fullname = fullfile('D:\Usc\Sem1\511\Project7','faithful.txt');

%Initialize

loaded\_data= load(fullname);

eruption\_time\_272 = loaded\_data(:,2);

eruption\_waiting\_time\_272(1,:) = loaded\_data(:,3);

eruption\_time\_272= eruption\_time\_272.';

* Next I store the eruption time and eruption waiting time in 2 columns of a matrix and plot their raw data point.

for iter = 1:max\_data

merged\_eruption\_waiting\_time(iter,1)= eruption\_time\_272(iter);

merged\_eruption\_waiting\_time(iter,2)= eruption\_waiting\_time\_272(iter);

end

figure(1);

plot(merged\_eruption\_waiting\_time(:,1), merged\_eruption\_waiting\_time(:,2), 'o');

[y,C]= kmeans(merged\_eruption\_waiting\_time,2);

title('Data Point without labels');

xlabel('waiting time since last eruption(min)');

ylabel('Duration of eruption(min)');

* Along with the above, I also plot the scattered plot of two clusters provided specifying the centroids.

figure(2)

plot(merged\_eruption\_waiting\_time(y==1,1),merged\_eruption\_waiting\_time(y==1,2), 'x');

hold on

plot(merged\_eruption\_waiting\_time(y==2,1),merged\_eruption\_waiting\_time(y==2,2), 'o');

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

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

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

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

hold off

* At this point of time, I have data points with and without labels plotted via K-means clustering.
* The next task is to overlay this plot with the plot attained via EM approach (theory and steps mentioned in the previous problem).
* Finally, a 3D plot of the Gaussian Mixture Model is calculated.

for iter = 1:max\_data

if(y(iter,1)==1)

X1(X1\_count,1)= merged\_eruption\_waiting\_time(iter,1);

X1(X1\_count,2)= merged\_eruption\_waiting\_time(iter,2);

X1\_count= X1\_count+1;

elseif(y(iter,1)==2)

X2(X2\_count,1)= merged\_eruption\_waiting\_time(iter,1);

X2(X2\_count,2)= merged\_eruption\_waiting\_time(iter,2);

X2\_count= X2\_count+1;

End end

GMM\_of\_X= [X1;X2];

figure(3);

hold on

scatter(X1(:,1),X1(:,2),10);

scatter(X2(:,1),X2(:,2),10);

opt = statset('Display','final');

max\_expectation = gmdistribution.fit(GMM\_of\_X,2,'Options',opt);

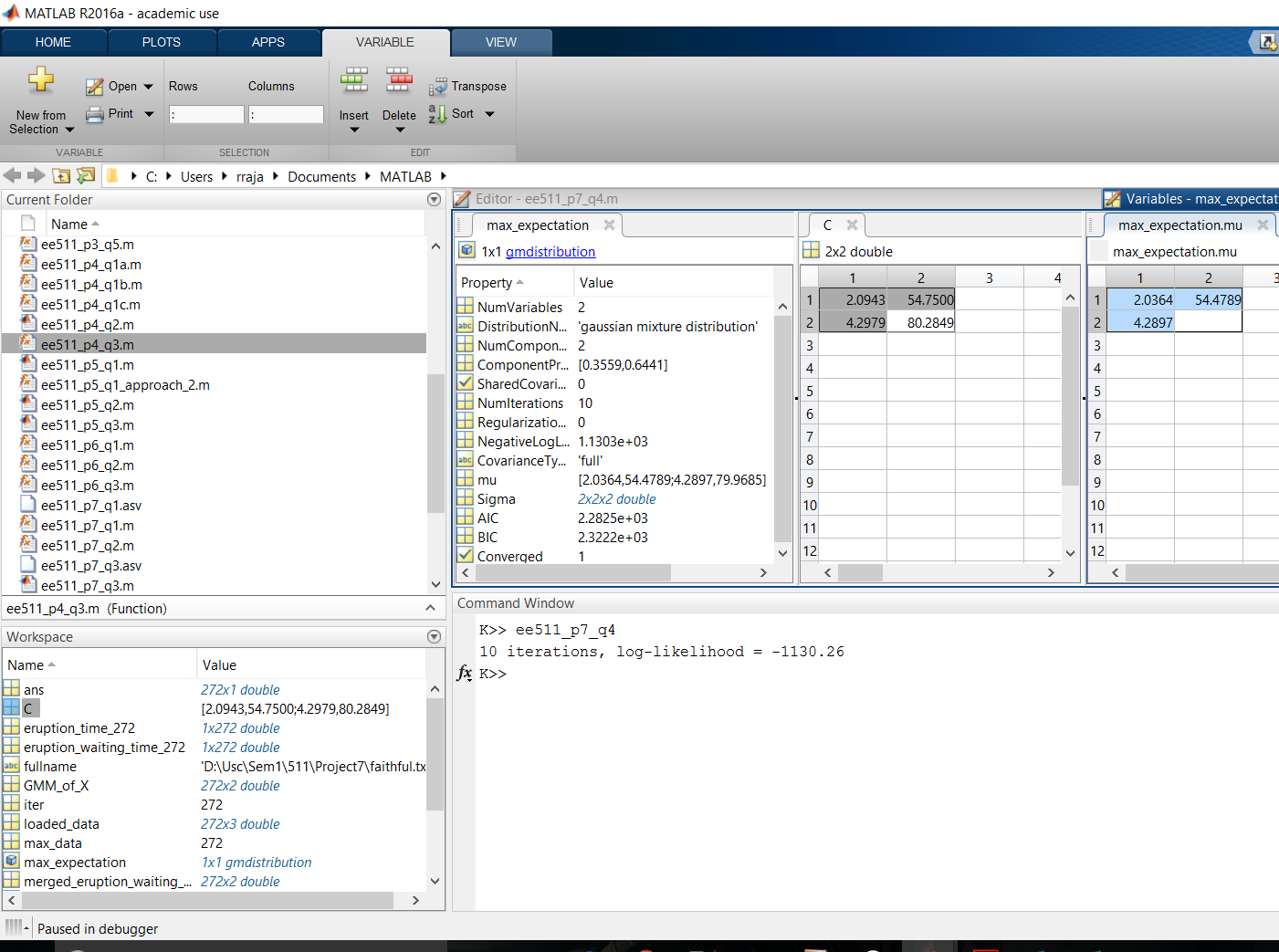
ezcontour(@(x,y)pdf(max\_expectation,[x y]),[0,7],[40,100]);

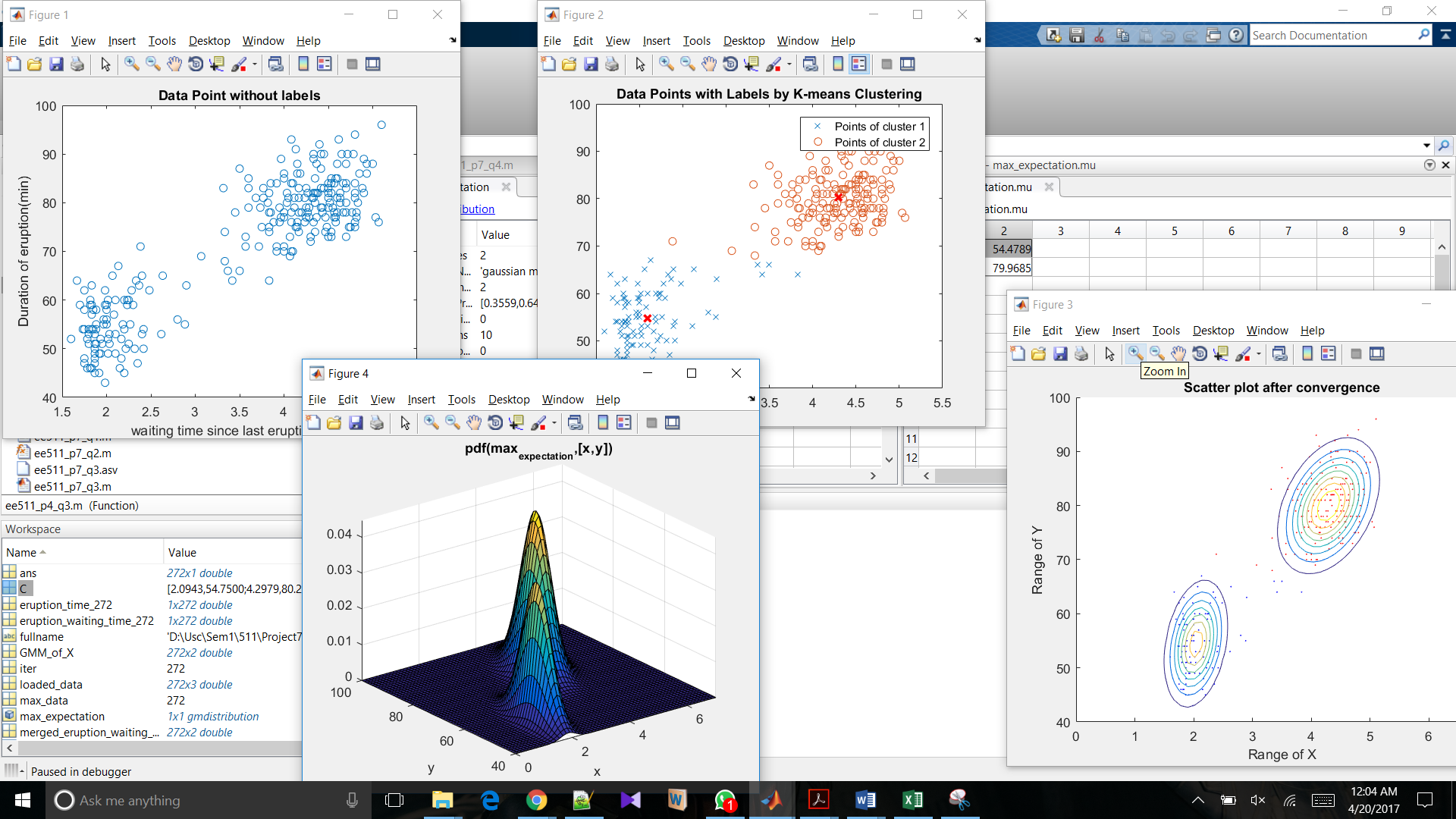
figure(4);

pdf(max\_expectation,GMM\_of\_X);

ezsurf(@(x,y)pdf(max\_expectation,[x y]),[0,7],[40 100]);

**Results and Plots:**





**Observations:**

* The plot via Expectation Maximization approach overlays the scattered plot via K-means cluster algorithm.
* Mean calculated by the kmeans () function is almost close to the estimated mean of the EM approach for GMM distribution.