Question 2:

Part 1:

First, I used MATLAB to implement a method to find that would find the clusters and the cost and return them. The method will have the following parameters passed to it: data, centers, k, and label. Data is the points that we have, centers is the initial center we initiated through different initialization procedures and k which is the number of clusters and label which will be used for the plot. The variable "c" will specify which cluster each points belong to.

Matlab Code:

```
function [result, c] = findClusters(data, centers, k, label)
initialcenters = centers;
input = data{1};
output = data{2};
% Second chose centers by random points
n = size(input,1);

% Number of iteration used for stopping criteria
N = 10;
%Cluster Assigning
c = zeros(n,1);
for i = 1:N
   %Assign Clusters
   for q = 1:n
      minD = 10000000;
      for m = 1:k
         dist = pdist([input(q),output(q);centers(m,1),centers(m,2)]);
         if (dist < minD)
            c(q) = m;
            minD = dist;
         end
      end
   end
   % Assign new centers

   % Now for each cluster find new center
   for m = 1:k
      % for each cluster
      sum = [0,0];
      count = 0;
     for q = 1:n
        % find points belonging to that cluster
        if c(q)  == m
           sum = sum + [input(q), output(q)];
           count = count +1;
        end
     end
     centers(m,:) = sum/count;
   end

   % Assign clusters after new centers
    for q = 1:n
```

```
        minD = 10000000;
        for m = 1:k
            dist3 = pdist([input(q),output(q);centers(m,1),centers(m,2)]);
            if (dist3 < minD)
                c(q) = m;
                minD = dist3;
            end
        end
    end
end

    % Finding the cost
    cost = 0;
     for (q=1:n)
         clus = c(q);
         cost = cost + pdist([input(q),output(q);centers(clus,:)]);
     end
    result = cost;
    for (p = 1:k)
        clus1 = [0,0];
        c1 = 1;
        for j=1:n
            if (c(j) == p)
                clus1(c1,:) = [input(j), output(j)];
                c1 = c1 + 1;
            end
        end

        % Plot the different clusters
        plot(clus1(:,1), clus1(:,2), 'x');
        hold on
    end
end
```

Now for the initialization of the center, we have three different methods, I will show a quick one by using k = 3:

1- Assigning it by hand:

```
% The centers that we have
centers = zeros(k, 2);
centers(1,:) = [-4,-4];
centers(2,:) = [3,-2];
centers(3,:) = [2,5];
findClusters(s1, centers, k, "By hand Centers selection 1");
```

2- Random Selection of Points

```
%Random Points
index = randi([1,200],1,1);
center1 = inputs(index, :);
index = randi([1,200],1,1);
center2 = inputs(index, :);
```

```
index = randi([1,200],1,1);
center3 = inputs(index, :);
center = [center1;center2;center3];
cost(1) = findClusters(s1, center, 3, "Random Points");
figure();
```

3- Maximum distance initialization

We find the new center that has the maximum sum distance from the previous centers.

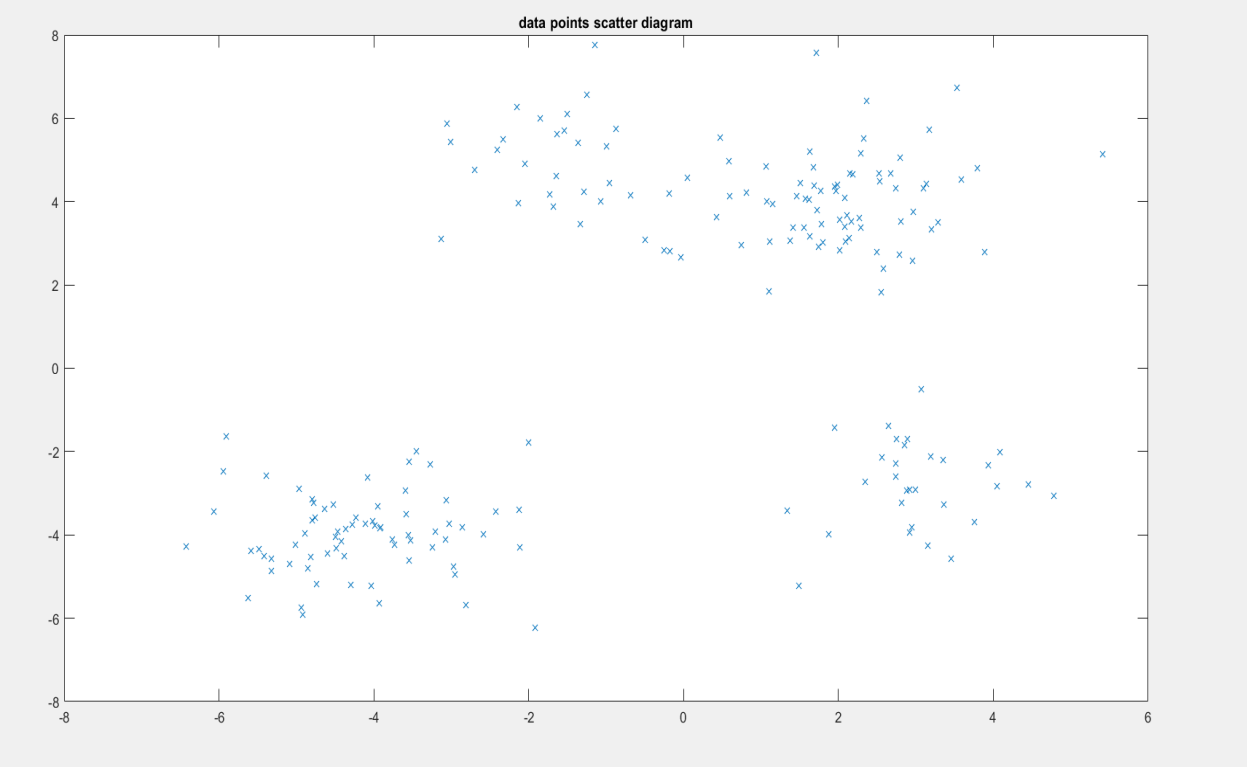
```
for (k = 2:10)
   for m = 2:k
      max = -10000;
      for q=1:n
         dist = 0;
         newPoint = [input(q),output(q)];
         already = 0;
         for (w = 1:m-1)
            dist = dist +  pdist([center(w,:);newPoint]);
         end
         if (dist > max)
            center2 = newPoint;
            max = dist;
         end
      end
      center(m,:) = center2;
   end
   cost(k) = findClusters(s1, center, k, "Maximum distance of Centers");
end
```

*******FULL CODE FOR THE FIRST FEW QUESTIONS IS IN THE APPENDIX***

Part 2:

Data scatter:

The following is a scatter of all the different points that we have:

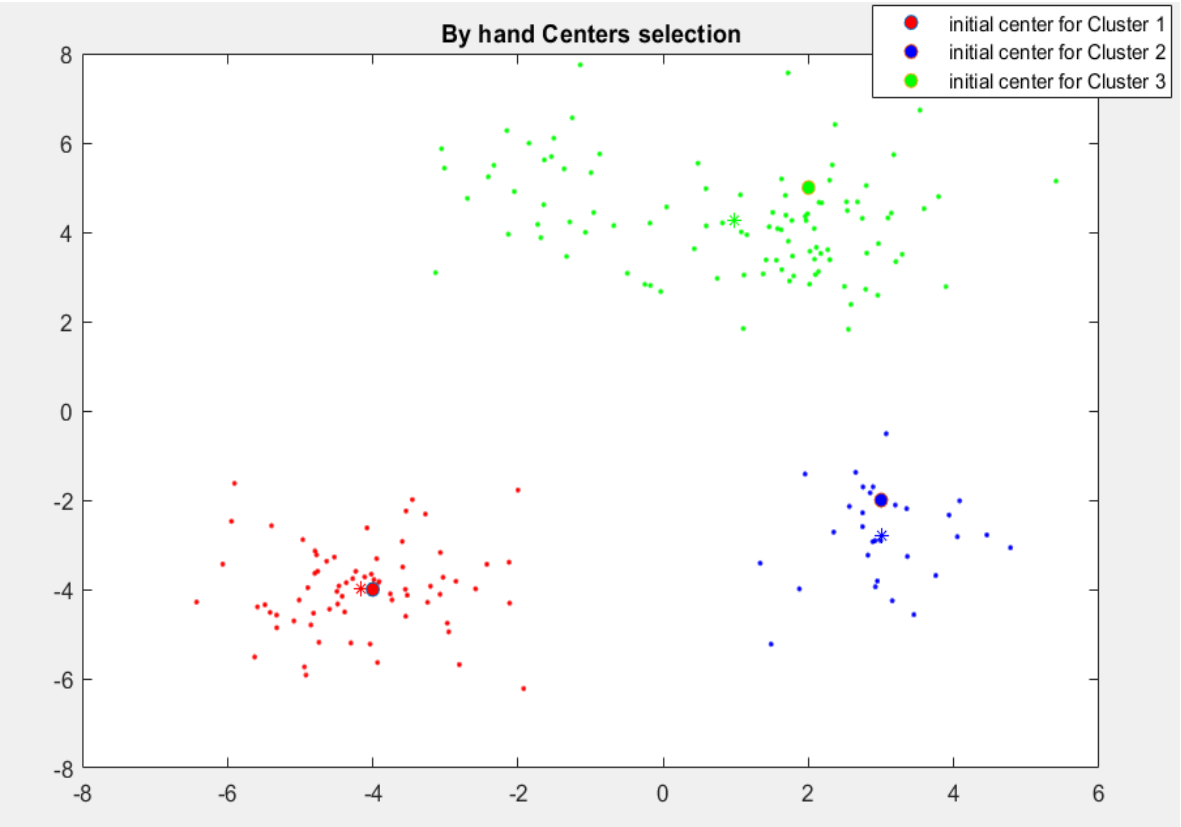


Observation:

From the initial observation, I would say that the number of different clusters is 3.

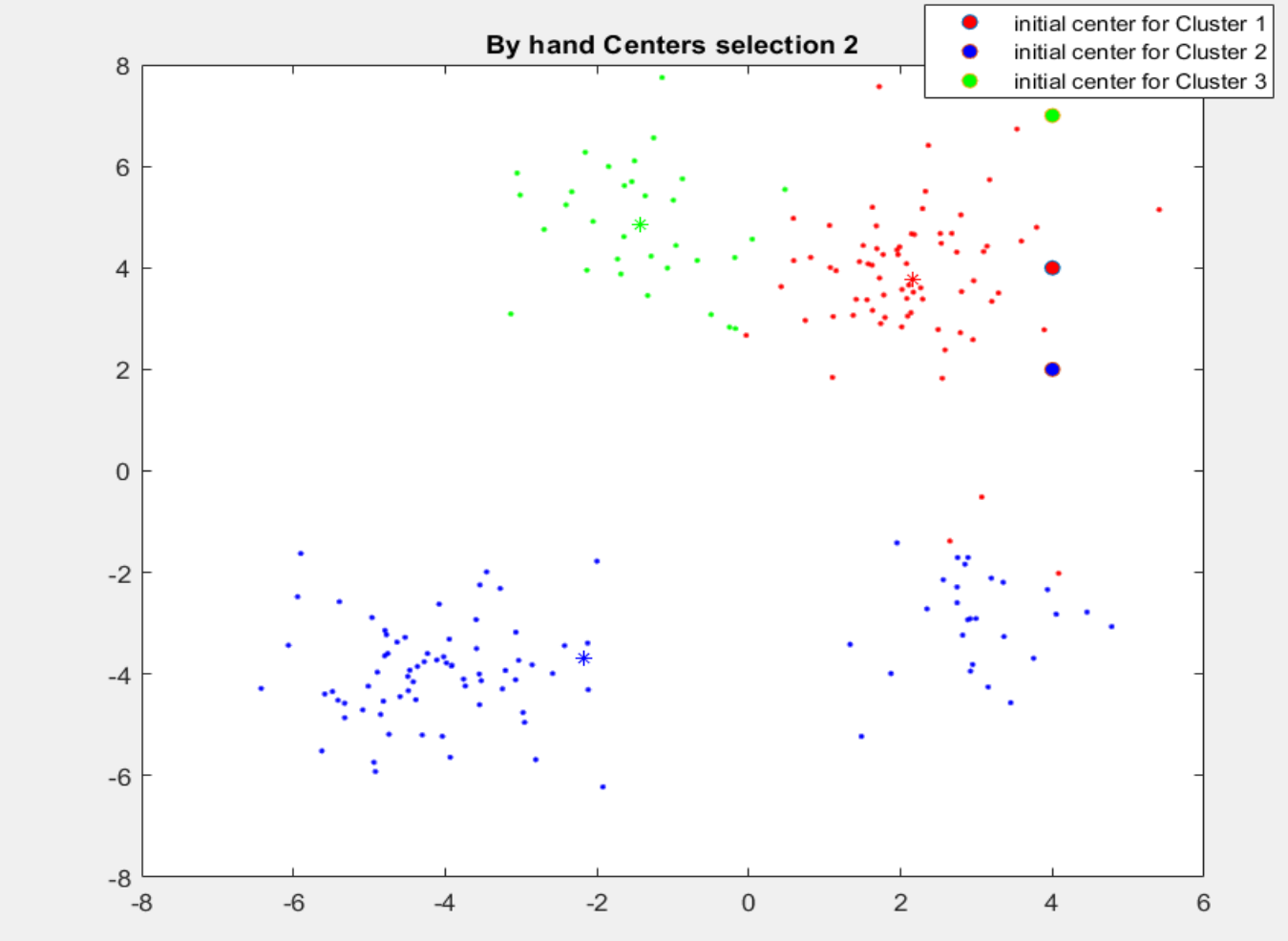
First clustering: By hand selection:

Please check the legend to see the starting center for each cluster. Each different one is represented by a colored filled circle. For example, red is for the first cluster. The points are then the small dots, each color represents a single cluster.



Second clustering: Second by hand selection:

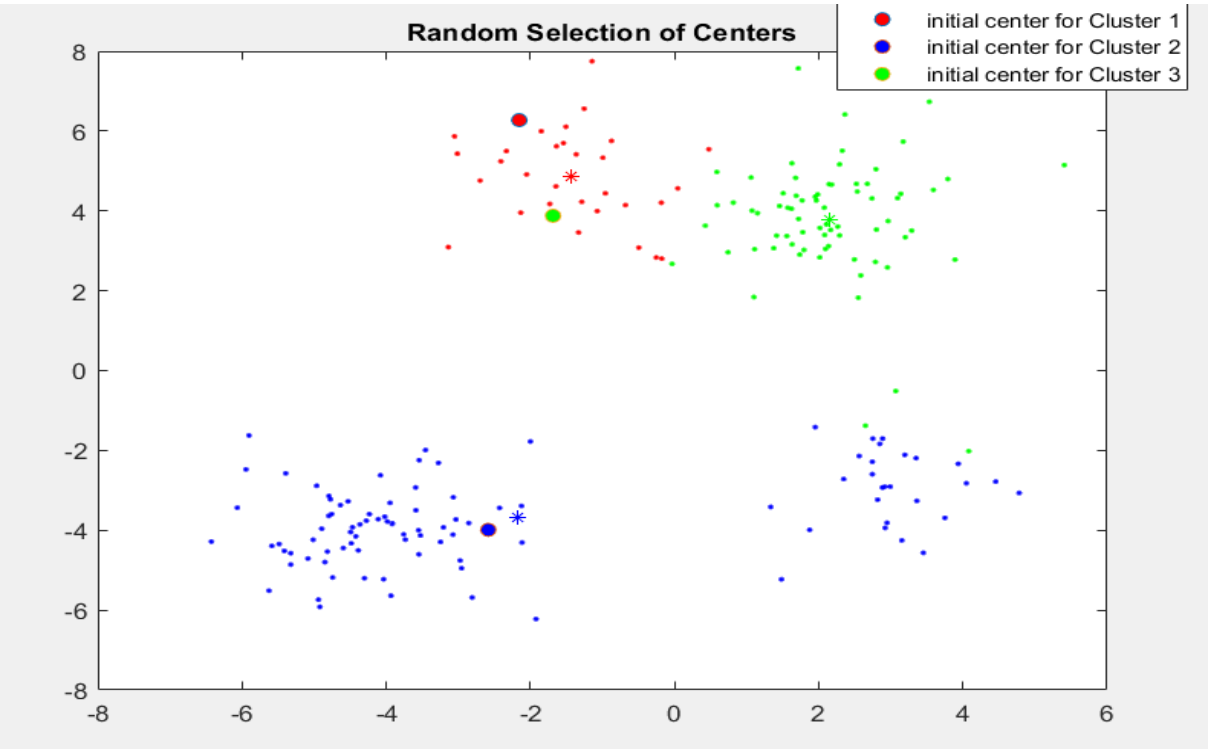
Here we could clearly notice the difference The first three centers were initialized as seen as the three different colored circles.



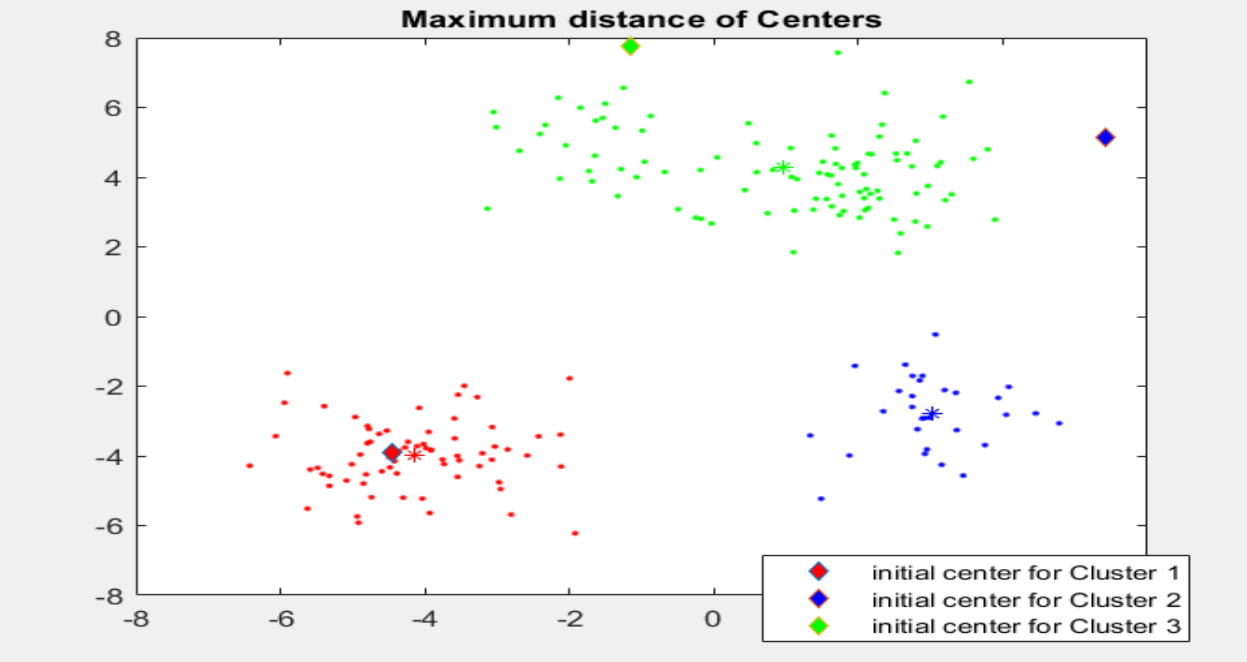
Therefore, as we could see the different initialization points could lead to different clustering. However, in this case, our first selection was right.

Part 3:

Yes of course, this could happen too if the points were chosen randomly. If the points were chosen for example randomly at the top right corner, a different clustering would occur. In the below example, we could see how that random selection appeared to give us a clustering which is not correct. This is why random selection is not a good procedure of initializing here. I ran this algorithm 10 times, I got the correct clustering 7 times and the wrong clustering as shown below three other times.

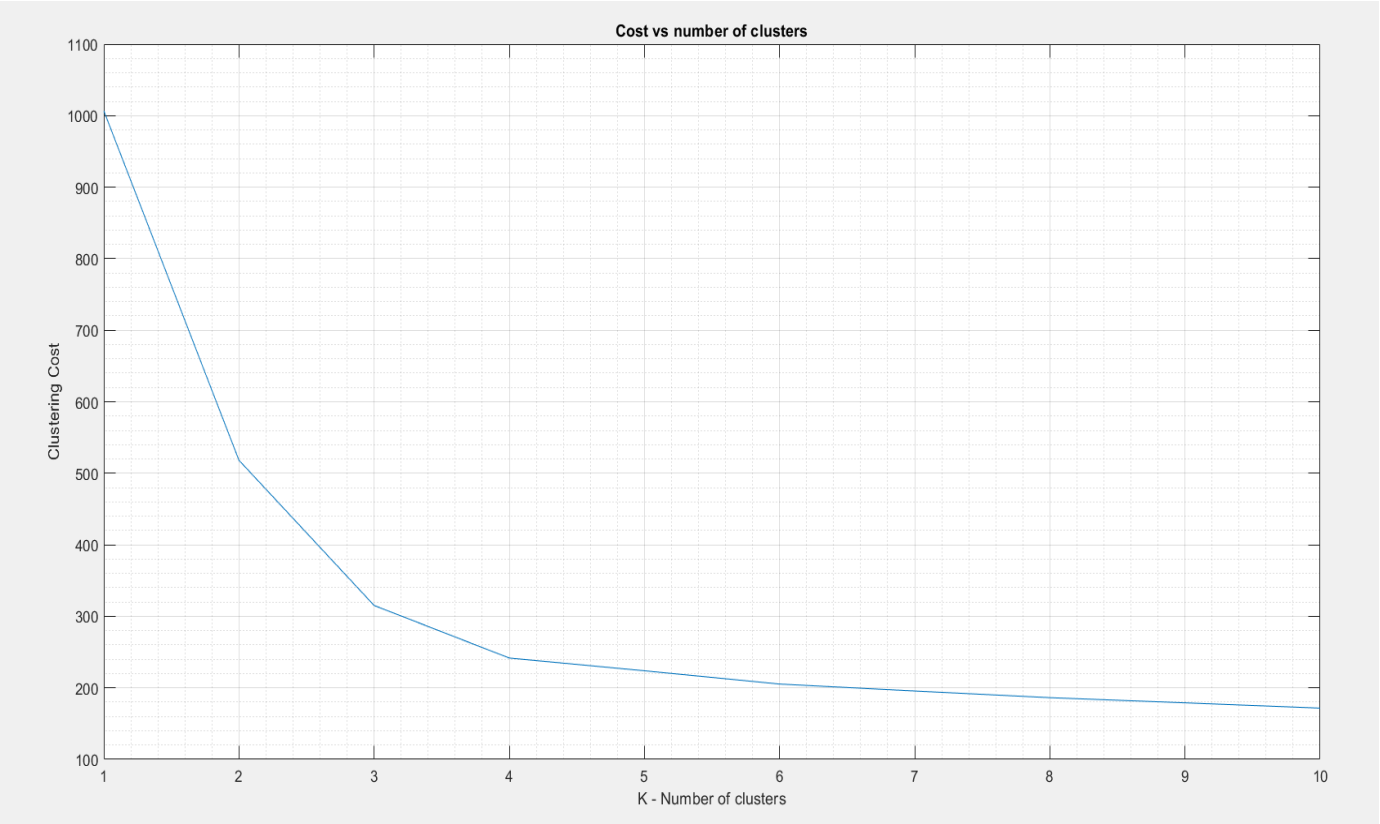


2- Now using the maximum distance initialization method, its very less likely that this would happen, as our selection is more deterministic than random. I tried to run with this procedure several times and they all resulted in correct clustering. However, this does not mean that this way is the right procedure to choose every time as this would not work out in other data sets. I ran this clustering method 10 times, and I got the correct clustering each time.



Part 4:

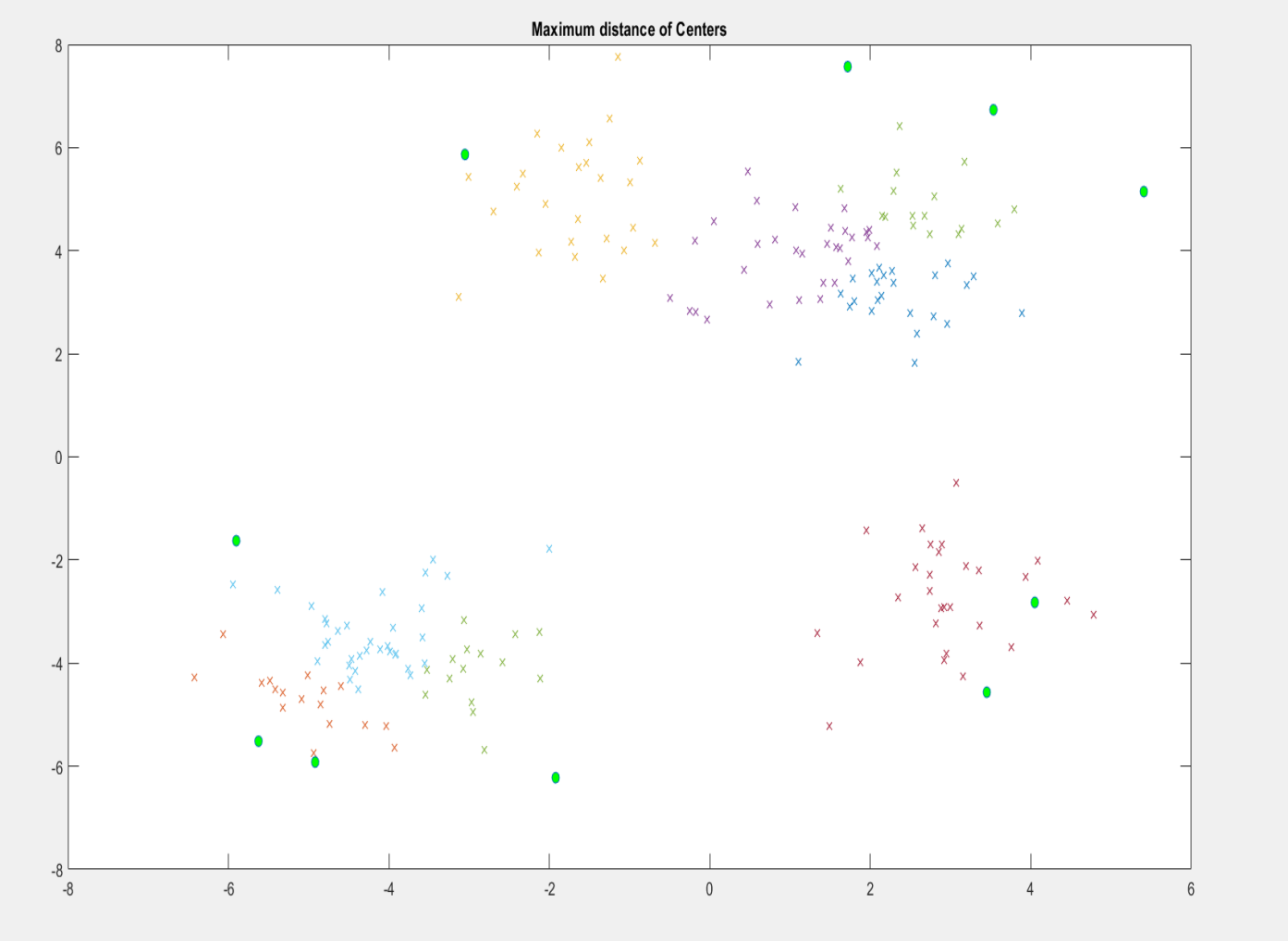
The following is the cost vs number of cluster method. The following is the code I used to find the different costs:



Observation:

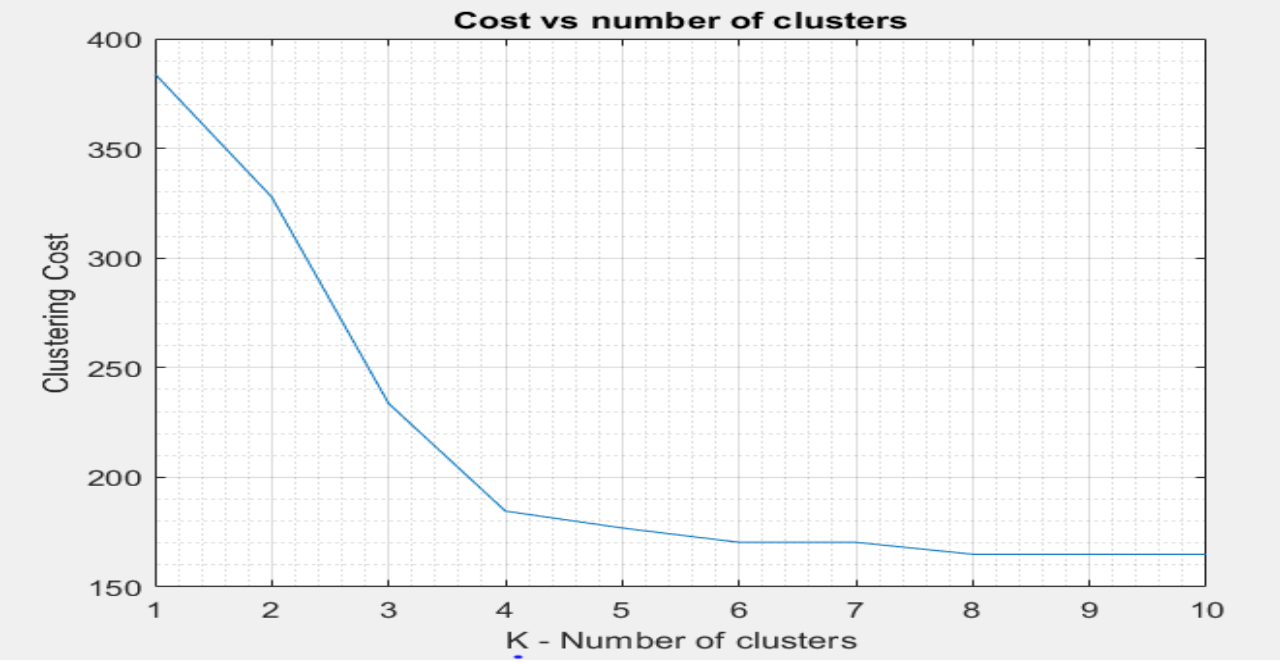
Here we could notice that the cost always decreases or stays the same, but will never increase. Here we also notice that we could determine the number of clusters by looking at the point where the slope change is no longer steep. This point in our case is 3, after that point the slope becomes flatter. Therefore, we could use this fact to help predict the suitable number of clusters. Where we would choose the number where slope is no longer steep, and start stabilizing.

Here is the plot of 10 different clusters:

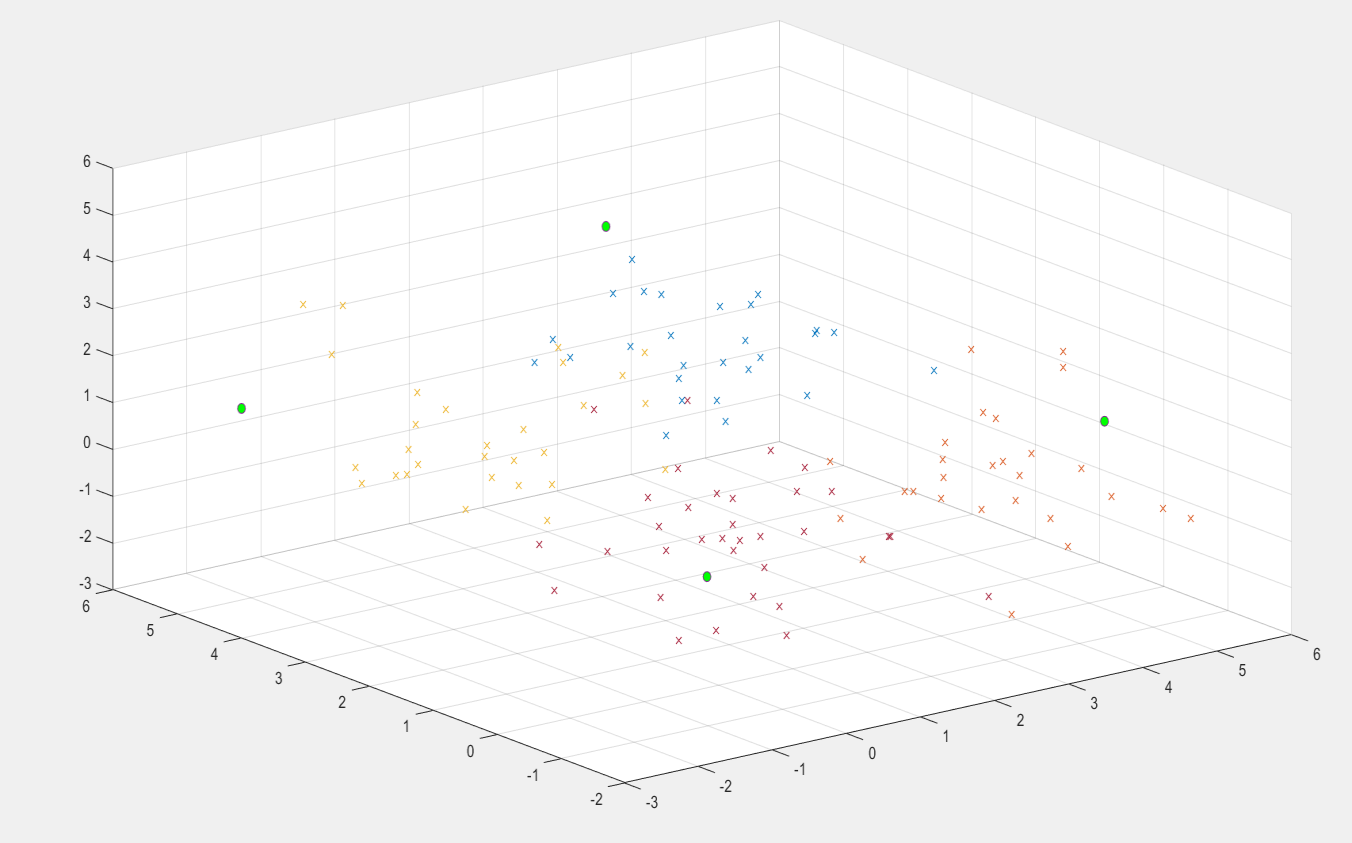


Part 5:

Now we will apply the same thing to the threedpoint data set. After observing this, I would predict that the suitable number of clusters is 4. Here is plot of the cost vs number of clusters.

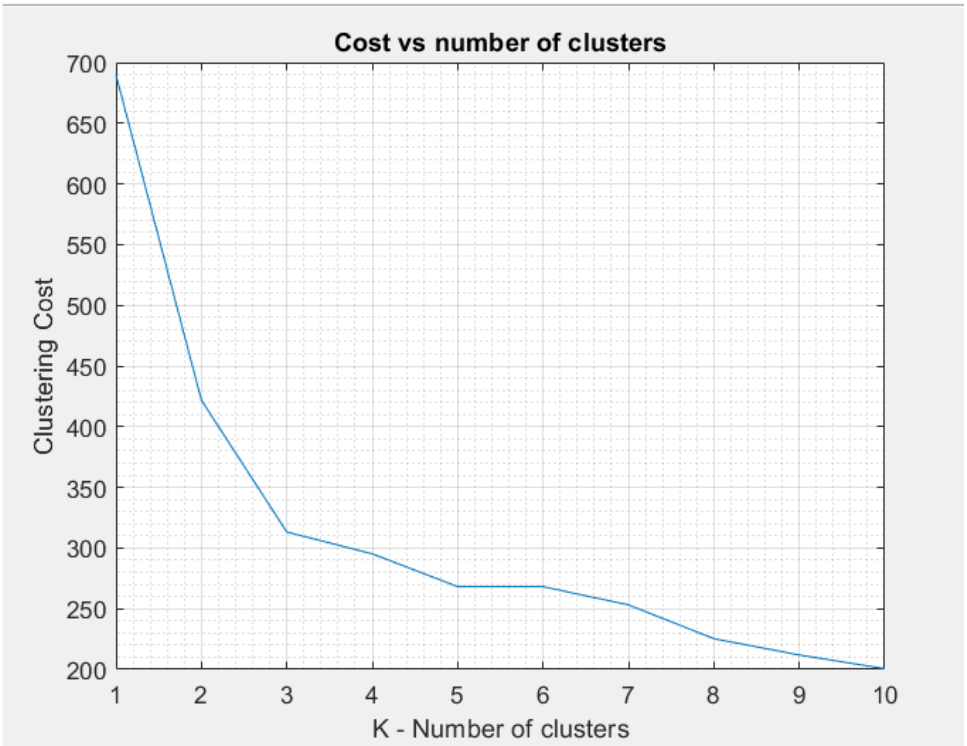


Confirming through a visualization:



Part 6:

After loading the seeds data set, we used the maximum distance we could the following plot. Where we could notice that the suitable number of clusters would be equal to 3.



Classifier:

A simple way classifier could be easily set up using the three different clusters. Each cluster would be set up that it would represent one class. Every point in that cluster would be classified under the class that the cluster represents.

Finding the empirical loss:

The empirical binary loss is as follows, out of the 210 points, only 23 were misclassified.

```
count =

    23


ans =

    0.1095
```

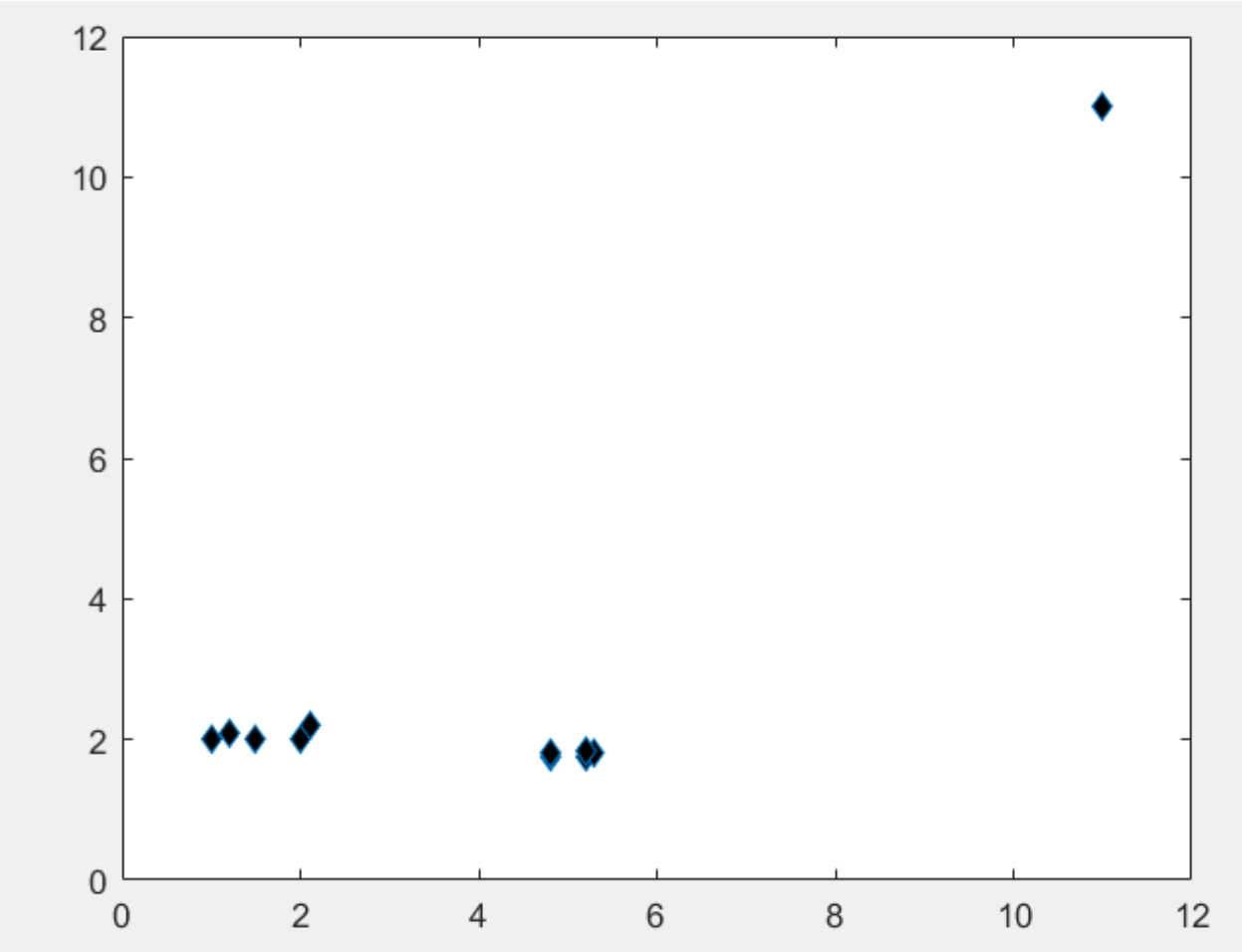
Loop to check the binary loss is:

```
for (i=1:n)
   if (classes(i) ~= c(i))
      count = count + 1;
   end
end
```

Part 7:

To prove that this wont work. We will create a dataset with one ***outlier*** value that is really far. That that one outlier will always be chosen as a center, which will mess up the k-means algorithm.

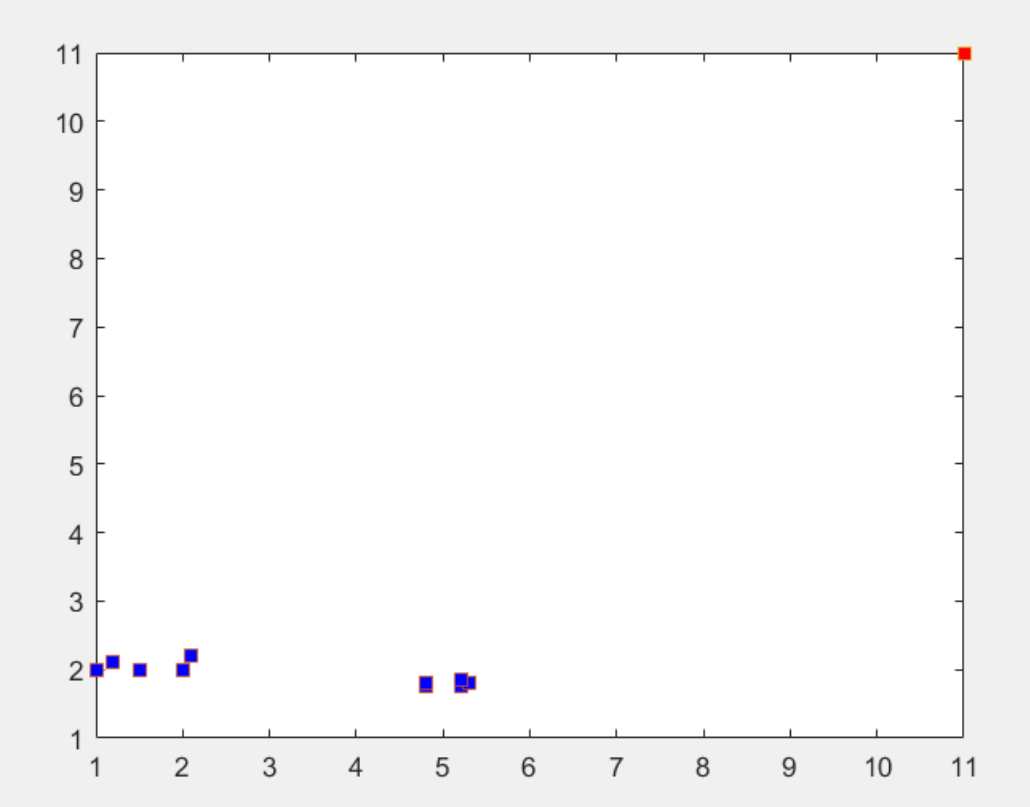
In the following data set:



From the points:

	1	2
1	2.1000	2.2100
2	2	2
3	1	2
4	1.5000	2
5	1	2
6	1.2000	2.1000
7	5.2000	1.7500
8	5.3000	1.8000
9	5.2000	1.8500
10	4.8100	1.7500
11	4.8000	1.8000
12	11	11
13		

The classification is always as follows:



Therefore, we always need a new way to be not deterministic in finding the furthest away point. But actually, the point should find a probabilistic way to find how often we randomly chose such points and use those probabilities to select the point. As choosing the furthest away would mess up if any outlier exists.

Appendix:

```
clc;
close all;
clear all;

% Scanning input from the files
fid1=fopen("C:\\Users\\Mohammed\\Desktop\\York Final Semester\\EECS
4404\\twodpoints.txt");
s1=textscan(fid1,'%f,%f');
fclose(fid1);
input = s1{1};
output = s1{2};
inputs = [input output];



%Random Points
index = randi([1,200],1,1);
center1 = inputs(index, :);
index = randi([1,200],1,1);
center2 = inputs(index, :);
index = randi([1,200],1,1);
center3 = inputs(index, :);
center = [center1;center2;center3];
cost(1) = findClusters(s1, center, 3, "Random Points");
figure();


% Second chose centers by random points
n = size(input,1);
index = randi([1,200],1,1);
center1 = [input(index,1), output(index,1)];
center(1,:) = center1;
cost(1) = findClusters(s1, center, 1, "Maximum distance of Centers");

% Second chose centers by random points
n = 120;
index = randi([1,120],1,1);
center1 = inputs(index, :);
center(1,:) = center1;
cost(1) = findClusters(s1, center, 1, "Maximum distance of Centers")
for (k = 2:10)
    for m = 2:k
        max = -10000;
        for q=1:n
            dist = 0;
            newPoint = inputs(q,:);
            already = 0;
            for (w = 1:m-1)
                if (newPoint == center(w,:))
                    already = 1;
                end
            end
            if (already == 0)
```

```
                for (w = 1:m-1)
                    dist = dist +  pdist([center(w,:);newPoint]);
                end
                if (dist > max)
                    center2 = newPoint;
                    max = dist;
                end
            end
        end
        center(m,:) = center2;
    end
    cost(k) = findClusters(s1, center, k, "Maximum distance of Centers");
end
plot(center(:,1), center(:,2), 'o', 'MarkerFaceColor', 'g')
hold on
hold off
figure();
plot([1:10], cost);
xlabel('K - Number of clusters');
ylabel('Clustering Cost');
title('Cost vs number of clusters');
grid on
grid minor

%plot(input, output, '.');

function [result] = findClusters(data, centers, k, label)
initialcenters = centers;
input = data{1};
output = data{2};
% Second chose centers by random points
n = size(input,1);

% Number of iteration used for stopping criteria
N = 10;
%Cluster Assigning
c = zeros(n,1);
for i = 1:N
    %Assign Clusters
    for q = 1:n
        minD = 10000000;
        for m = 1:k
            dist = pdist([input(q),output(q);centers(m,1),centers(m,2)]);
            if (dist < minD)
                c(q) = m;
                minD = dist;
            end
        end
    end
    % Assign new centers

    % Now for each cluster find new center
    for m = 1:k
        % for each cluster
        sum = [0,0];
        count = 0;
       for q = 1:n
           % find points belonging to that cluster
           if c(q)  == m
               sum = sum + [input(q), output(q)];
               count = count +1;
           end
       end
       centers(m,:) = sum/count;
```

```
    end

    % Assign clusters after new centers
     for q = 1:n
        minD = 10000000;
        for m = 1:k
            dist3 = pdist([input(q),output(q);centers(m,1),centers(m,2)]);
            if (dist3 < minD)
                c(q) = m;
                minD = dist3;
            end
        end
     end
end

    % Finding the cost
    cost = 0;
     for (q=1:n)
         clus = c(q);
         cost = cost + pdist([input(q),output(q);centers(clus,:)]);
     end
    result = cost;
    for (p = 1:k)
        clus1 = [0,0];
        c1 = 1;
        for j=1:n
            if (c(j) == p)
                clus1(c1,:) = [input(j), output(j)];
                c1 = c1 + 1;
            end
        end

        % Plot the different clusters
        plot(clus1(:,1), clus1(:,2), 'x');
        hold on
    end
    title(label);
end
```