
**Numerical Computing** **2023**

**Student:** Harkeerat Singh Sawhney

---

**Due date:** Wednesday, 8 November 2023, 11:59 PM

**Numerical Computing 2023 — Submission Instructions**
**(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)**

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, MATLAB). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  *Project_number_lastname_firstname*
  and the file must be called:
  *project_number_lastname_firstname.zip*
  *project_number_lastname_firstname.pdf*
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

## 1. Spectral clustering of non-convex sets [50 points]

### 1.1.

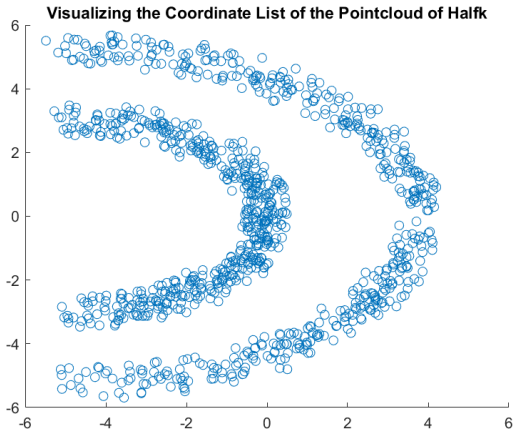



Figure 1: Plot of the data points for HalfK

Figure 1 shows the plot of the data points for HalfK. By observing the figure we can easily see the two clusters in the data. The two clusters are half a circle, in which one is within the other. It is easy for us to look at the graph and distinguish the two clusters, but it is not easy for different algorithms to do the same. There are 2 different approaches which this project focuses on; K-Means and Spectral Clustering.

The K-Means Algorithm takes identifies the two clusters by taking the mean of the two clusters and then assigning the points to the cluster which is the closes to the point. In the other Spectral Clustering is a modified version of K-Means clustering which runs the algorithm on the eigenvectors of the graph's Laplacian Matrix and adjusts the cluster centroids based on its Eigenvectors coordinates. As it can be noticed that calculating the Eigenvectors and Eigenvalues of the Laplacian Matrix is computationally expensive as compare to the K-Means algorithm.

If theoretically both the algorithms run over the data points of HalfK, we should observe different results. K-Means with Spectral method should be able to properly identify the two clusters, but K-Means without Spectral method should not be able to identify the two clusters. This is because the data points are not linearly separable and the original K-Means algorithm would not be able to do the same in this case. The reason is because it uses Euclidean Distance. With the use of Euclidean Distance the data points in each cluster are modeled by the position of their centroids, in which K-Means assumes all clusters should have the same radius. Hence because of this assumption, the K-Means algorithm would not be able to identify proper clusters in Spherical data. In the following questions this theory will be tested.

### 1.2. Computing $\in$ Factor

The points are all in the space and we need a function which lets us define them and compute the similarity of any 2 points. Hence the function which would be used to compute the similarity of any 2 points would be Gaussian Kernel Similarity Function. The function is as follows:

$$s(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{1}$$

The Gaussian Kernel Similarity Function tends to be between 0 and 1. In the function the choice of the $\sigma$ is very important as it controls how big or small the neighborhood of a point is. In our case we chose the given value which is $log(n)$.

Once the Gaussian Kernel Similarity Function is defined, we can then compute the Minimal Spanning Tree so that we can remove any edges which are not required. With this we can at last chose our $\in$ factor. The $\in$ factor is the maximum spanning tree of the Gaussian Similarity. In table 1 we can see the $\in$ factor for each of the data sets.

| **Mesh** | $\in$ **Factor** |
|---|---|
| Two Spirals | 0.6883 |
| Cluster in Cluster | 0.63243 |
| Crescent & Full Moon | 0.55984 |
| Corn | 0.55469 |
| Outliers | 0.29756 |

Table 1: The $\in$ factor for each of the data sets

### 1.3. $\in$ Similarity Graph

We are asked to complete the Matlab Function *epsilonSimGraph()* which takes in the data points and the $\in$ factor and returns the $\in$ Similarity Graph. We can utilize the Pseudo Code given in the lecture notes to complete the function. The function is as follows:

```
n = size(Pts,1);
G = zeros(n,n);

for i = 1:n
    for j = 1:n
        distance = norm(Pts(i,:) - Pts(j,:));
        if distance < epsilon
            G(i, j) = 1;
        end
    end
end
```

In the above code we are iterating over all the points and calculating the distance between them. If the distance is less than the $\in$ factor then we are adding an edge between the two points. The function returns the $\in$ Similarity Graph.

### 1.4. Adjacency Matrix for the $\in$ Similarity Graph

In order to get the Adjacency Matrix we just need to multiply the $\in$-neighborhood Graph with the Gaussian Similarity Matrix.

$$S \odot G = W \tag{2}$$

From this we can graph the resulting Adjacency Matrix. The Adjacency Matrix for the $\in$ Similarity Graph for each of the data sets is shown in figure 2.


Figure 2: Adjacency Matrix for the $\in$ Similarity Graph

### 1.5. Laplacian Matrix for the $\in$ Similarity Graph and Spectral Clustering

In order to obtain the Laplacian Matrix, we utilize the function *CreateLapl*, which takes the Adjacency Matrix as input and returns the Laplacian Matrix. Next, we compute the Eigenvalues and Eigenvectors of the Laplacian Matrix, sort the Eigenvalues in ascending order, and select the K smallest Eigenvalues (which can be either 2 or 4) along with their corresponding Eigenvectors. We then pass these selected Eigenvectors to the *kmeansMod* function, which returns the clusters. Finally, we visualize these clusters by plotting them on the graph.

Bellow is also the code for the Spectral Clustering algorithm.

```
[L, Diag] = CreateLapl(W);
[V, D] = eig(L);

lambda = diag(D);
[lambda, idx] = sort(lambda);
Y = V(:,idx(1:K));

[D_spec,x_spec] = kmeans_mod(Y, K, n);
```

### 1.6. Visualization of Two Clustering Algorithms

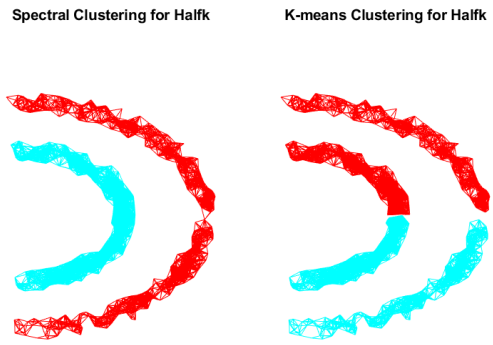


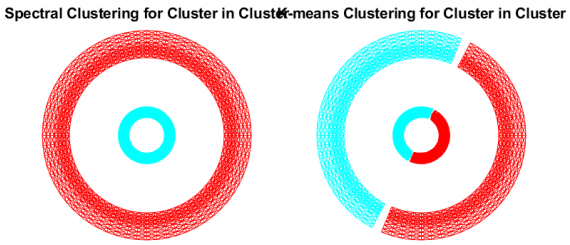

Figure 3: Spectral Clustering and K-Means Clustering on HalfK

In Figure 3, we can see that the Spectral Clustering algorithm is able to identify the two clusters in the data set while the K-Means algorithm is not able to accomplish that. This is mainly due to

the reasons mentioned in the Question 1.1.

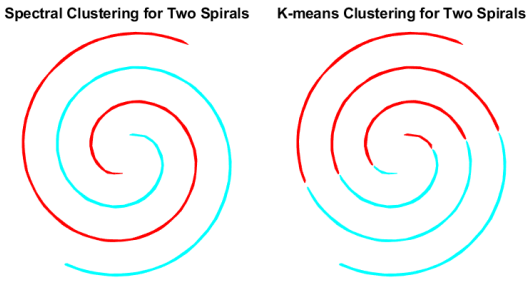
### 1.7. Spectral Clustering and K-Means Clustering on Different Data Sets

This is very similar to the previous question, but instead of using the HalfK data set we are using the other data sets.




(a) Cluster in Cluster

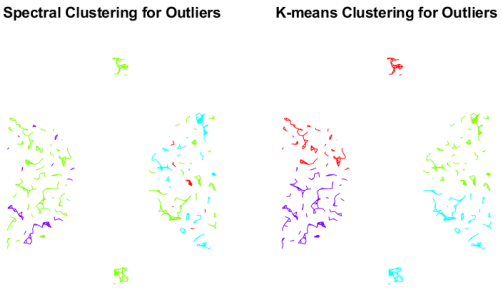

(b) Crescent & Full Moon




(c) Two Spirals

Figure 4: Spectral Clustering and K-Means Clustering on Different Datasets


(a) Corn




(b) Outliers

Figure 5: Spectral Clustering and K-Means Clustering on K = 4

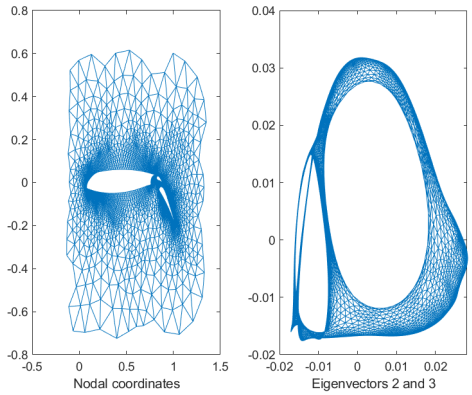
From the Figure 4 and Figure 5 we can see that the Spectral Clustering algorithm is able to identify the clusters in the data set while the K-Means algorithm is not able to accomplish the same that well. We can see for K = 4 that when there outliers it influences the centroid very heavily for K-Means Clustering. We also notice that K-Means Clustering treats all the centroids as the same size as it does not recognize the density of the clusters.

## 2. Spectral clustering of real-world graphs [35 points]

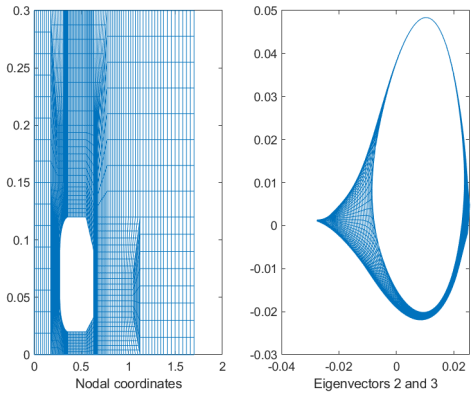
### 2.1. Construction of Laplacian Matrix and Visualization of the Graph

In this question we do everything the same as the Question 1.5, except that once we get the eigenvectors we take the 2nd and 3rd smallest eigenvectors. We then combine them into the matrix which is passed on to the *kmeansMod* function. The code for this is as follows:

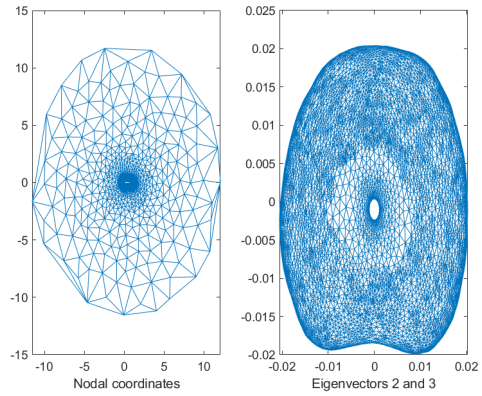
```
lambda = diag(D);
[lambda, idx] = sort(lambda);
Y = V(:, idx(1:K));

v2 = Y(:,2);
v3 = Y(:,3);

new_coords = [v2, v3];
```



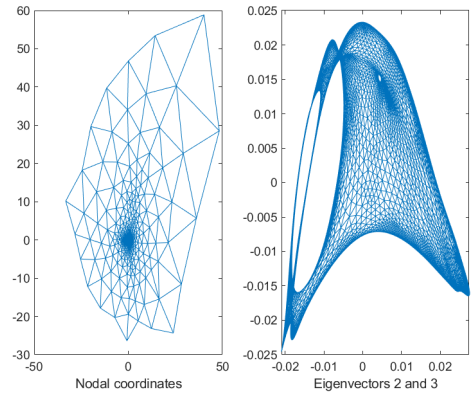

(a) Airfoil1




(b) Grid2




(c) Barth



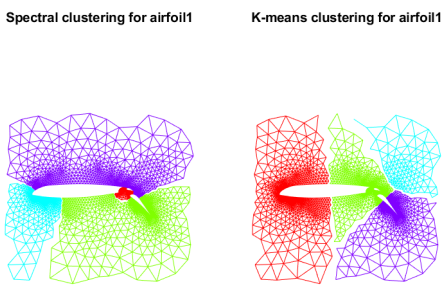

(d) 3elt

Figure 6: Visualization of Different Graphs for Airfoil

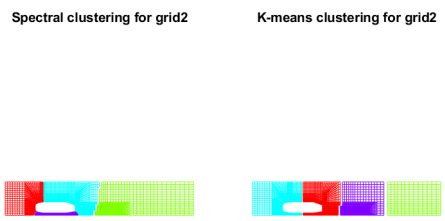
In the Figure 7 we can see the visualization of the graph. We can see that it is visualized in two ways, one being with Nodal Coordinates and other being with the 2nd and 3rd smallest eigenvectors.

## 2.2. Spectral Clustering and K-Means Clustering on Different Meshes

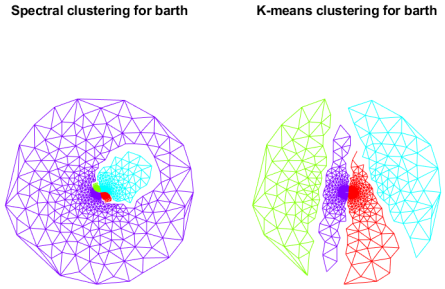
In this question we are asked to run the Spectral Clustering and K-Means Clustering on the different meshes.



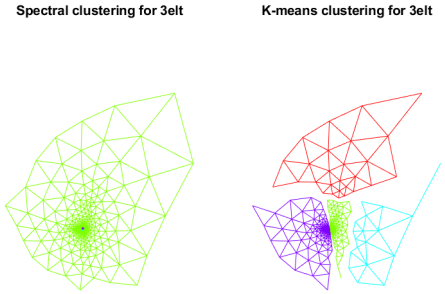

(a) Airfoil1




(b) Grid2




(c) Barth




(d) 3elt

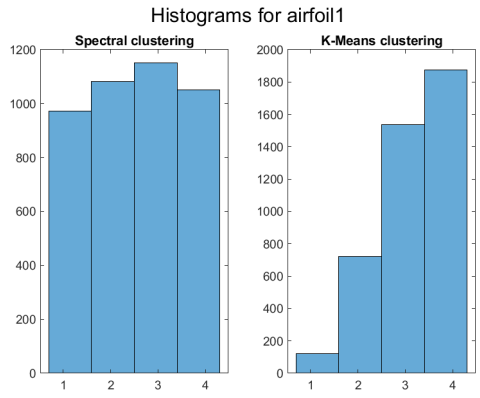
Figure 7: Spectral Clustering and K-Means Clustering on Different Meshes

### 2.3. Visualization of Tables and Histograms

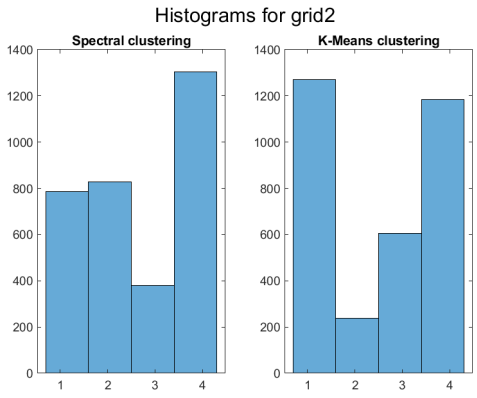
What we can observe from all the data collected is that K-Means clustering fails to identify the Cluster Density and uses it partitioning based on the distance between points on their spatial distance. This makes it weak in meany cases in which Spectral Clustering is able to perform better. Bellow are the tables and histograms for the different meshes.

| | **Spectral Nodes** | | | | **K-Mean Nodes** | | | |
|---|---|---|---|---|---|---|---|---|
| **Grid2** | 785 | 827 | 379 | 1305 | 1271 | 238 | 604 | 1183 |
| **Barth** | 1490 | 2195 | 1601 | 1405 | 75 | 2894 | 69 | 3653 |
| **3elt** | 1098 | 1497 | 1199 | 926 | 3139 | 1536 | 18 | 27 |

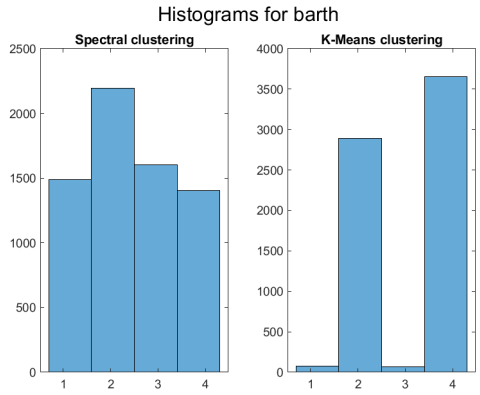
Table 2: Number of nodes per cluster for different methods



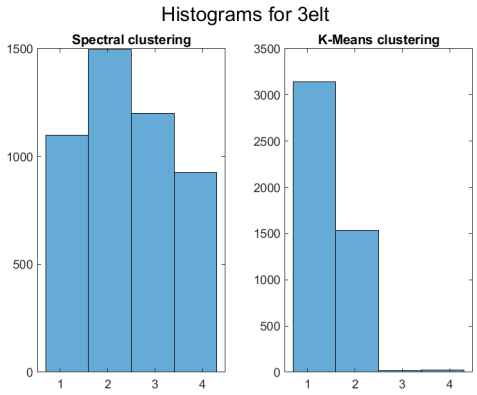

(a) Airfoil1




(b) Grid2




(c) Barth




(d) 3elt

Figure 8: Histograms for Different Meshes