Università della Svizzera italiana

Institute of Computing CI

**Numerical Computing** **2023**

**Student:** Kyla Kaplan **Discussed with:** FULL NAME

---

## Solution for Project 3 **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)**

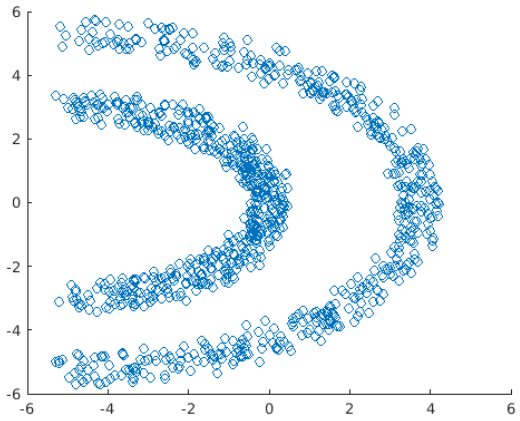
- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, MATLAB). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  *Project_number_lastname_firstname*
  and the file 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. Consider the set named half-kernel: can you identify the two obvious clusters in this dataset? Describe them briefly and explain what difficulties a clustering algorithm could eventually encounter in a scenario of this kind.

   After running *getPoints.m* within the given project files results in a 6-graph output. When running *ClusterPoints.m* for the half-kernel graph, the two clusters become very evident. From direct observation, the unique shape of two 'banana' shaped curve clusters, one placed around the other, have a big spacial gap between them.

   When it comes to clustering algorithms, there are definite difficulties that can arise during the clustering process. While maybe not directly applicable to this graph, if cluster regions of a graph are relatively close to each other, then the algorithm may have a problem when distinguish between the given clusters, as there is difficulty in classifying them. Another challenge (which depends on the clustering algorithm itself), can be because of the graph density. In this case, some parts of the graph are very densely populated, which can create a problem for the algorithm to differentiate when partitioning into multiple clusters. For a k-means clustering approach, such a graph would also prove complicated (as it calculates the distances between the centers), as it is a non-convex graph[1].




(a) Half-kernel graph

2. Use the function *minSpanTree()* to find the minimal spanning tree of the full graph. Use this information to determine the epsilon factor for the epsilon-similarity graph.

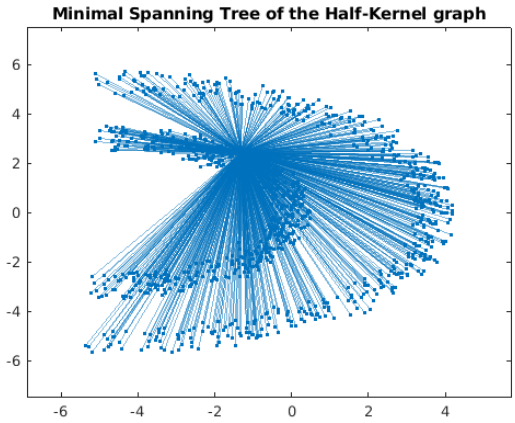
   The necessary code is implemented in *ClusterPoints.m*.

```
G = pdist2(Pts_halfkernel, Pts_halfkernel);
A = double(G < max(G(:)));

min_span_tree = minSpanTree(A);
eps = max(min_span_tree(min_span_tree > 0));

sparse_mst = sparse(min_span_tree);
```

   In order to visualize the graph:

```
graph_mst = graph(sparse_mst);
figure;
plot(graph_mst, 'XData',Pts_halfkernel(:, 1), 'YData', Pts_halfkernel(:, 2));
title('Minimal Spanning Tree of the Half-Kernel graph');
```

[1] I found this source fun and interesting with regards to the project




(b) Minimal Spanning Tree of Half-kernel graph

The minimum spanning tree in this case helps us when counting the minimal subset of edges to connect the vertices in a graph. Then, spectral clustering aids in identifying the relevant clusters, especially in the cases when they are not directly distinguishable.

3. Complete the Matlab function *epsilonSimGraph()*. It should generate the similarity matrix of the epsilon-similarity graph.

   Implementation can be found in *ClusterPoints.m*.

```
G = squareform(pdist(Pts)) < epsilon;
```

The epsilon similarity graph helps generating the similarity matrix, which is necessary for the epsilon-similarity graph. Thus, the epsilon-similarity graph is produced via connecting the points which have a distance less than epsilon, enabling to see the structure of the local clusters.

4. Create the adjacency matrix for the epsilon similarity graph and visualize the resulting graph using the function gplotg().

   Respective implementation can be found in *ClusterPoints.m*.

```
W = S .* G;
% To plot:
figure;
gplotg(W, Pts_halfkernel(:,1:2))
title('Adjacency matrix for the Half-Kernel graph')
```

The output is visualized here:

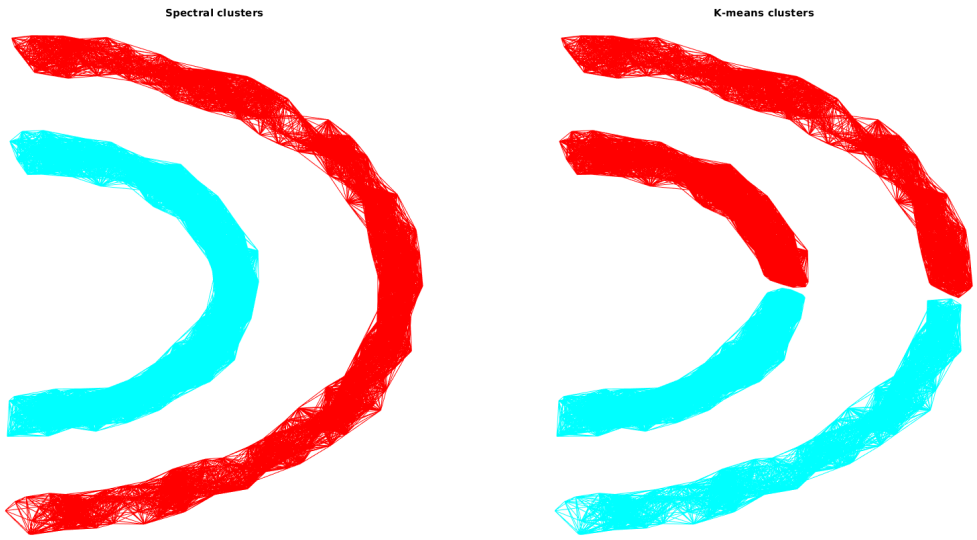

(c) Adjacency matrix for the Half-kernel graph

5. Create the Laplacian matrix and implement spectral clustering. Your goal is to find the eigenvectors of the Laplacian corresponding to the K = 2 smallest eigenvalues. Afterwards, use the function *kmeans_mod()* to cluster the rows of these eigenvectors.

   Respective implementation can be found in *ClusterPoints.m*.

```
[L, Diag] = CreateLapl(W);
% Extract
[eigs, eigvals] = eig(L);
% K smallest eigenvalues of the resulting Laplacian matrix
[~, id_eig] = sort(diag(G));
% K-smallest eigval's
V = eigs(:,id_eig(1:K));
% of the corresponding eigenvectors
% V normalized
V_norm = diag(1./sqrt(sum(V.^2, 2))) * V;
```

6. Use the *kmeans_mod()* function to perform k-means clustering on the input points. Visualize the two clustering results using the function *gplotmap()*.

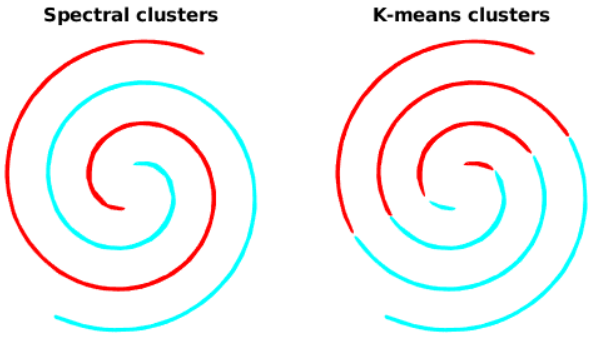



(d) Spectral and K-means clustering for Half-kernel graph

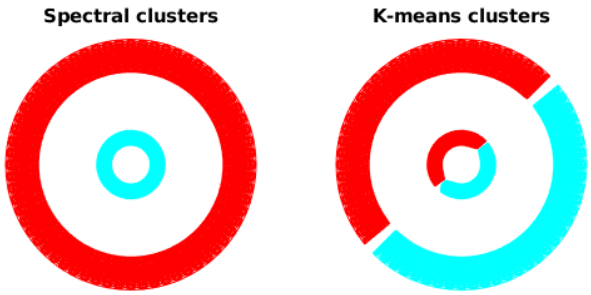
I attempted to play with the $\sigma$ value of the Gaussian similarity function in order to result in a 'sparser-looking' matrix as in Figure 1, however, I wasn't really able to reach such a similar result, so in the submission I left it blank.

7. Cluster the datasets i) Two spirals, ii) Cluster in cluster, and iii) Crescent & full moon in $K = 2$ clusters, and visualize the results obtained with spectral clustering and k-means directly on the input data. Do the same for i) Corners, and ii) Outlier for $K = 4$ clusters.

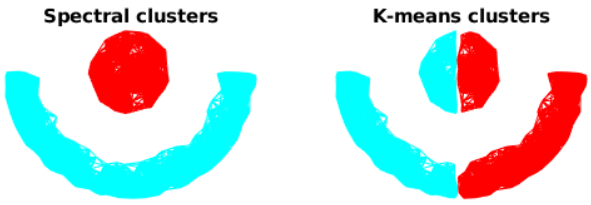
   **For $K = 2$:**




(e) Spectral and K-means clustering for Two Spiral graph

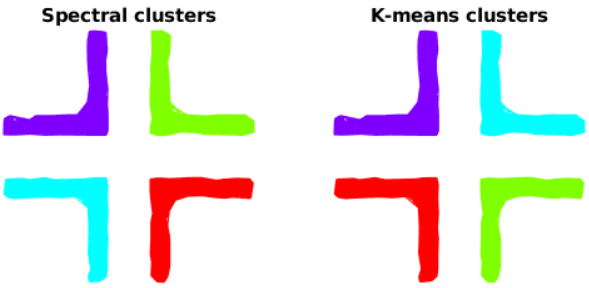



(f) Spectral and K-means clustering for Cluster-in-cluster graph

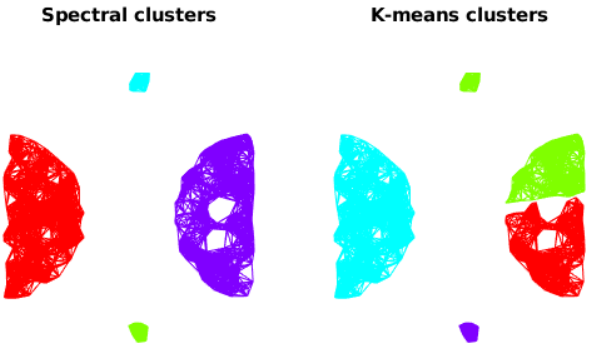



(g) Spectral and K-means clustering for Crescent & full moon graph

**For $K = 4$:**




(h) Spectral and K-means clustering for Corners graph




(i) Spectral and K-means clustering for Outlier graph

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

1. Construct the Laplacian matrix, and draw the graphs using the eigenvectors to supply coordinates. Locate vertex $i$ at position:

$$(x_i, y_i) = (v_2(i), v_3(i))$$

where $v_2$, $v_3$ are the eigenvectors associated with the 2nd and 3rd smallest eigenvalues. Figure 2 illustrates these 2 ways of visualizing the airfoil1 graph. Plot your results for all the three additionally supplied meshes, *grid2*, *barth*, *3elt*.

Unfortunately, unsure to exactly why, but I was unable to properly render the spectral clustering for the various graphs, therefore I have left the second part of the project unfinished, and included only all relevant files for the first part.