Università della Svizzera italiana

Institute of Computational Science ICS

Numerical Computing **2020**

Student: Stefano Gonçalves Simao Discussed with: Stella Born

Solution for Project 4

Due date: Wednesday, 18 November 2020, 11:55 PM

Numerical Computing 2020 — 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 [60 points]:**

   - (a) Let's consider the set *two spirals*: we can see that the two sets are the two lines that spiral with respect to each other. It is clear that the data, in this case, is not linearly separable. This makes it difficult to some clustering algorithms to find the plane or hyperplane that divides the data. For example k-means cannot find the right clustering as it assumes that the data lies in disjoint convex sets, i.e. the data is linearly separable. With Spectral clustering we don't have this problem because it does not make strong assumptions on the form of the clusters. Other algorithms able to cluster non-linearly separable data are Support Vector Machine with the kernel trick or neural networks with hidden layers.
   - (b) The ϵ factor is the length of the longest edge in a minimal spanning tree of the fully connected graph on the data points. This way the resulting graph is safely connected.
   - (c) We first compute the pairwise distance between the points and we only take the ones equal or less the ϵ factor.
   - (d) In order to compute the adjacency matrix we perform element-wise multiplication between the Gaussian similarity function and the ϵ similarity graph.

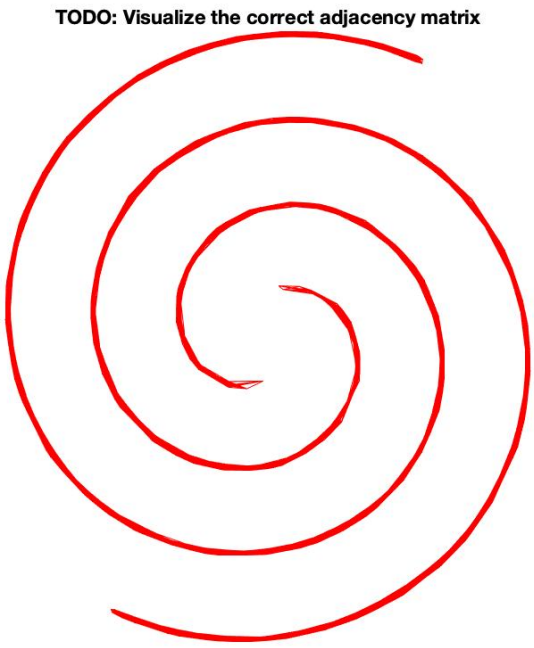

Figure 1: Adjacency matrix for the set *Two spirals*

- (e) Here we compute the eigenvalues, sort them and reorder them and reorder the eigenvectors. Then take only the K smallest and cluster the rows using k-means.
- (f)

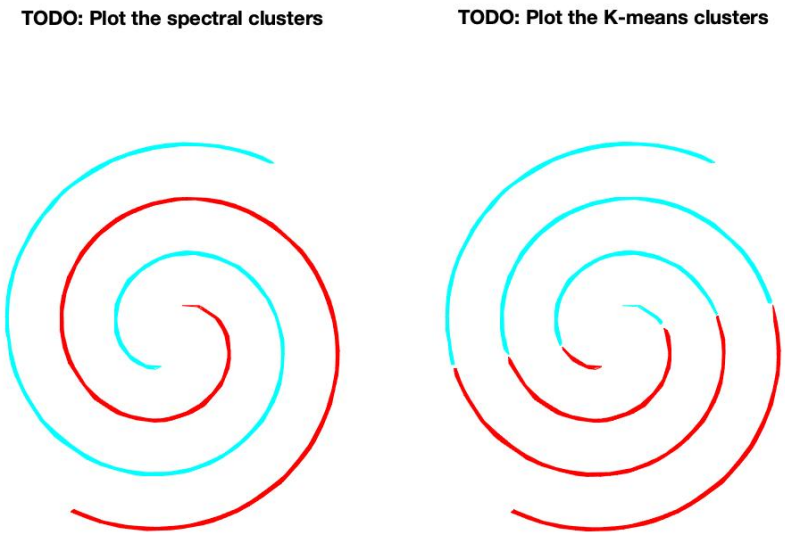

Figure 2: Clustering results for the set *two spirals*

- (g)

a) Figure 1

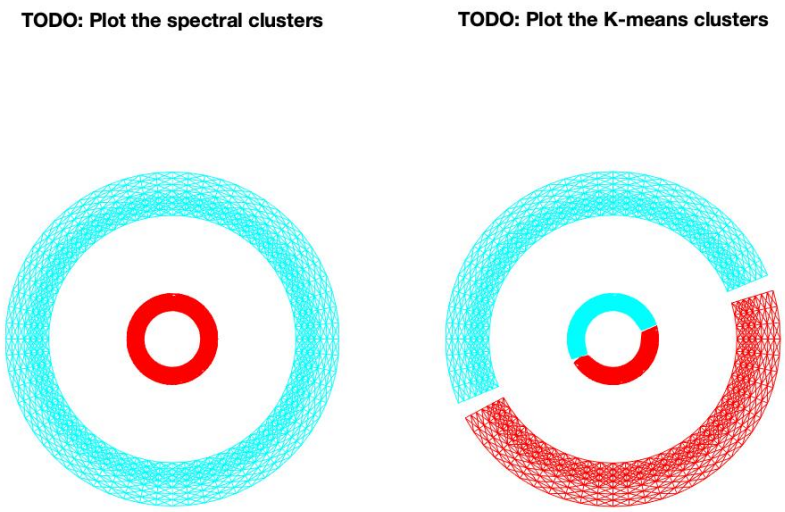

Figure 3: Clustering results for the set *Cluster in cluster*

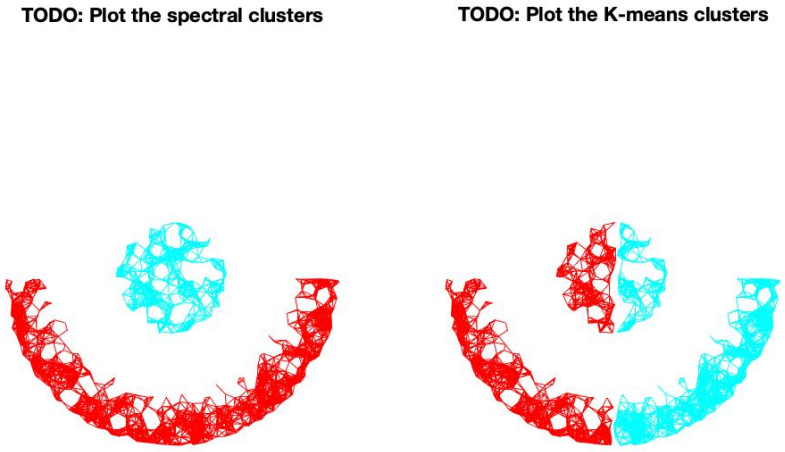

Figure 4: Clustering results for the set *Crescent & full moon*

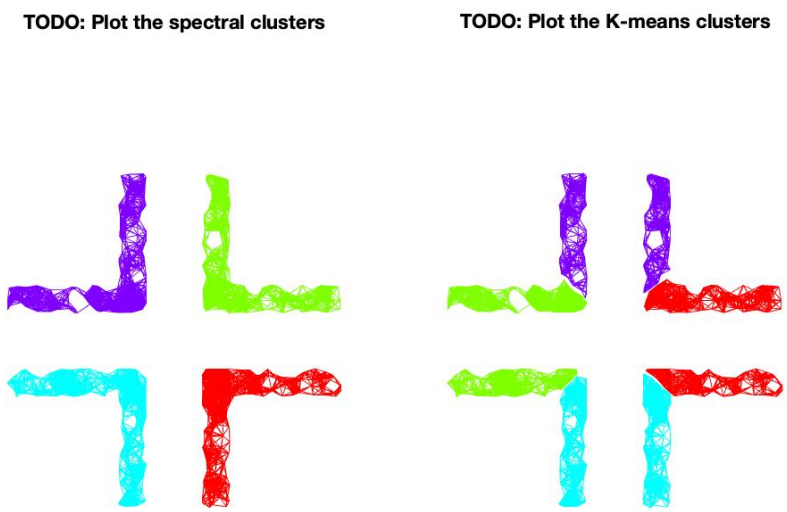

Figure 5: Clustering results for the set *Corners*

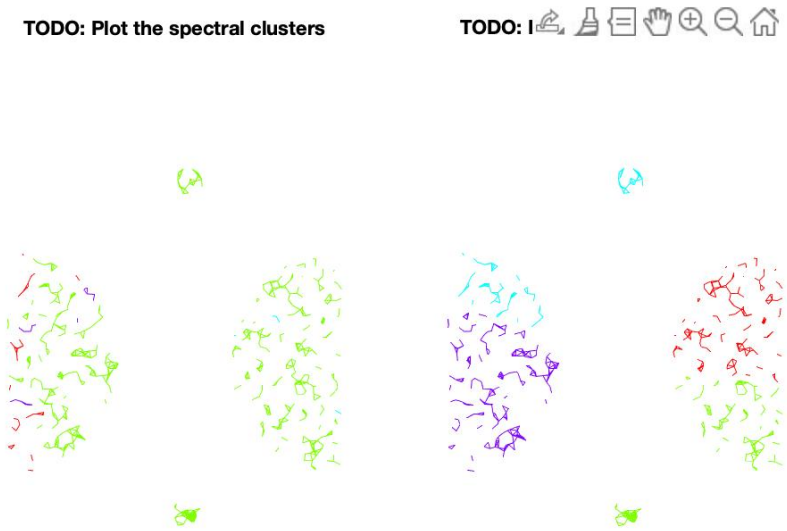

Figure 6: Clustering results for the set *Outlier* with $\sigma = log(n)$

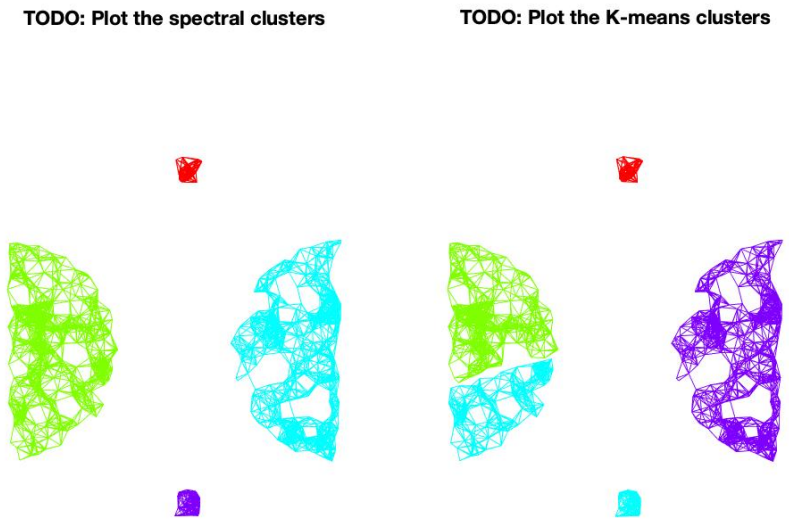

Figure 7: Clustering results for the set *Outlier* with $\sigma = 2 * log(n)$

We can see in the last two figures that changing σ changes drastically the performance of the spectral clustering. By doubling the provided σ successfully used with the other sets, we build a more interconnected graph as the width of the neighborhoods is bigger. This little change gives the algorithm the possibility to find the connections between the real clusters.

2. **Spectral clustering of real-world graphs [40 points]:**
   - (a) For this subtask I just used my code from task 1. I changed the line where the program reordered the eigenvectors in order to keep only the ones associated with the 2nd and 3rd smalest eigenvalues.
   - (b) From Figure 3 of the problem sheet we can see that the K-Means reflects the fact that it produces a Voronoi diagram which consists of linear decision boundaries. Looking at the results shown below, we can clearly see this pattern for every case. The algorithm assigns each point to the nearest mean, producing planes as boundaries and that's clearly reflected in the visualizations below. The spectral clustering however works on the similarity of the points giving more coherent results with the underline truth. For the $grid2$ case the Spectral clusters fail to give a coherent result as, I suppose, the different parts of the graph (different in density) are tightly connected, preventing the algorithm to find the good similarities to use. In the last case, $3elt$, it is also clear that the difference in density and the fact that the graph doesn't show more parts that are connected loosely connected, makes the algorithm perform poorely. One could also say it is good as it provided a big cluster that incorporates almost all graph except for the more dense part.

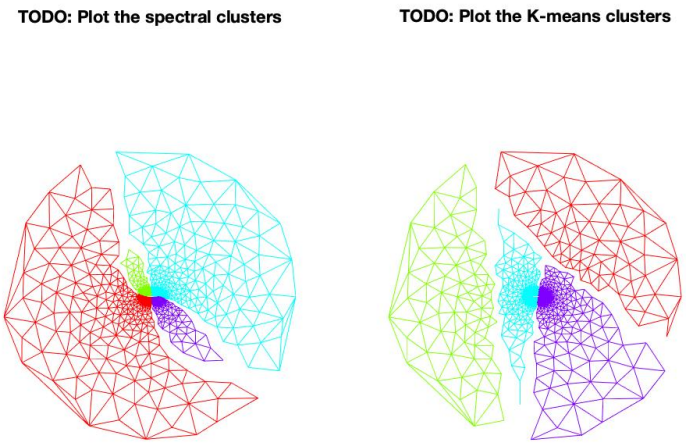

Figure 8: Clustering results for the set *barth*

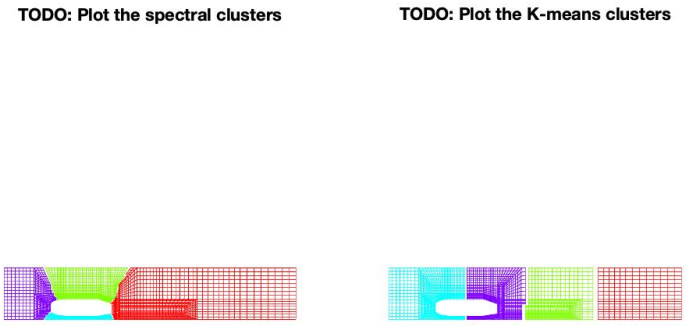

Figure 9: Clustering results for the set *grid2*

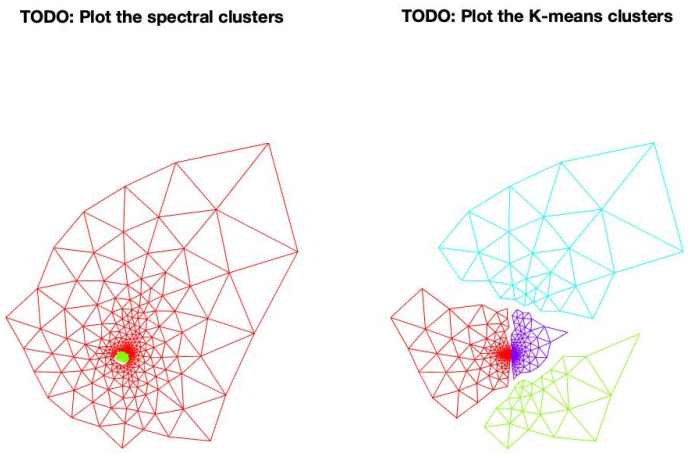

Figure 10: Clustering results for the set *3elt*

- (c) Clustering results, K = 4

Case	Spectral	K-Means
grid2	986, 370, 579, 1361	604, 1183, 238, 1271
barth	1537, 1708, 1564, 1882	71, 3800, 2752, 68
3elt	1269, 935, 1773, 743	30, 28, 3146, 1516

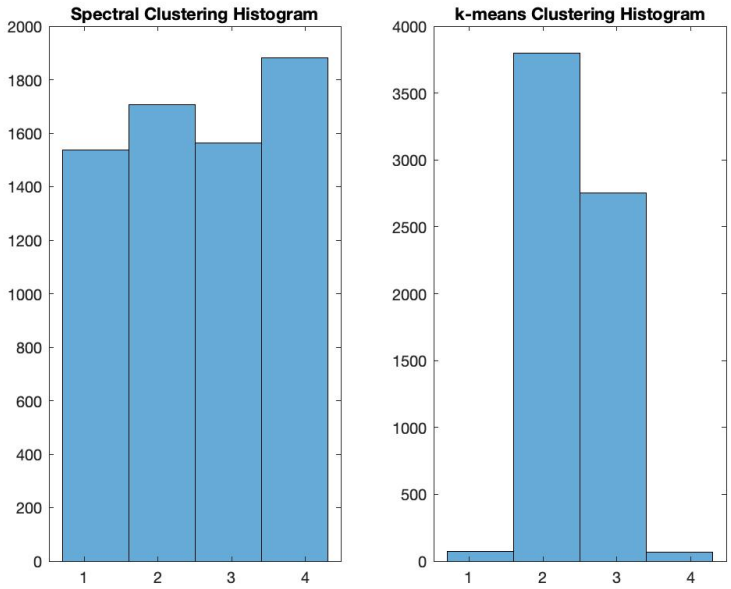

Figure 11: Histogram for the set *barth*

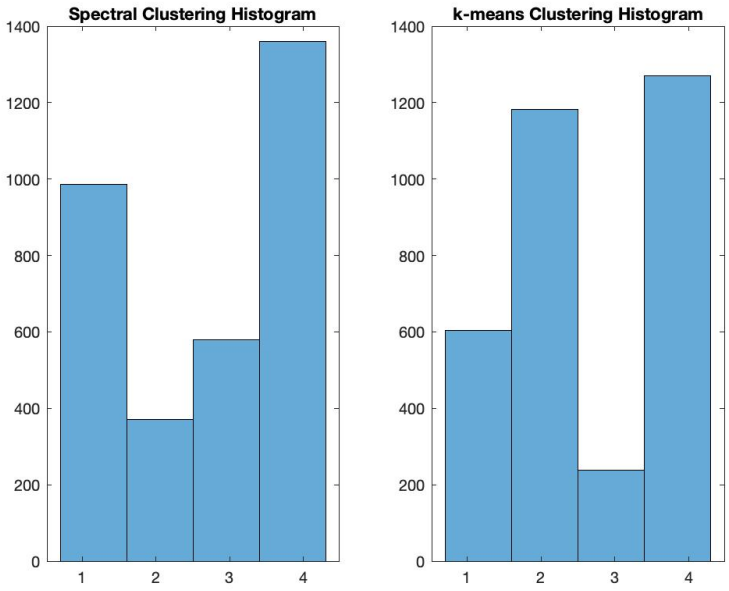

Figure 12: Histogram for the set *grid2*

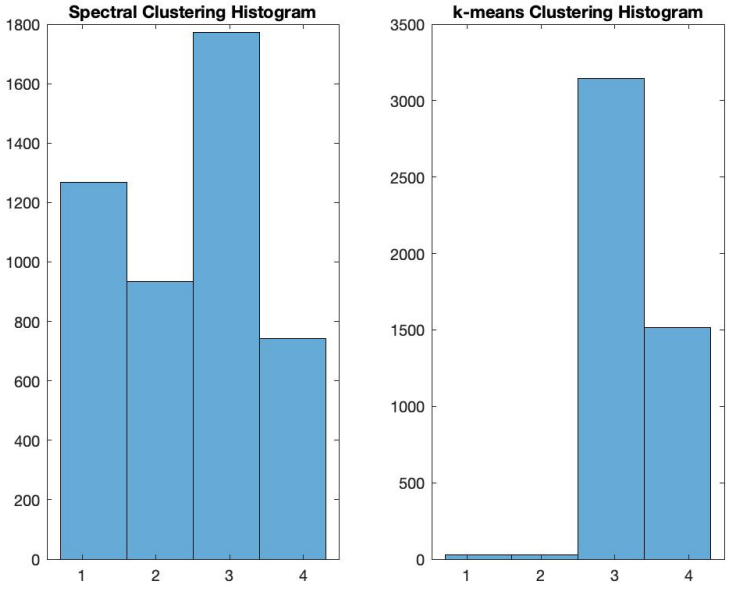

Figure 13: Histogram for the set *3elt*

For *barth*, where there are 4 cluster, the spectral clustering produced clusters similar in size. When the partition is not straightforward, both spectral and K-Means produce more unbalanced results. For K-Means that is obvious as it searches for the means and doesn't look at the size of the cluster. Spectral reflects better the structure if the points are similarly connected, but when density changes it produces less balances results.