Numerical Computing **2023**

Student: Jeferson Morales Mariciano

Discussed with: Leonardo Birindelli, Filippo Piloni, Michele Dalle Rive

---

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)

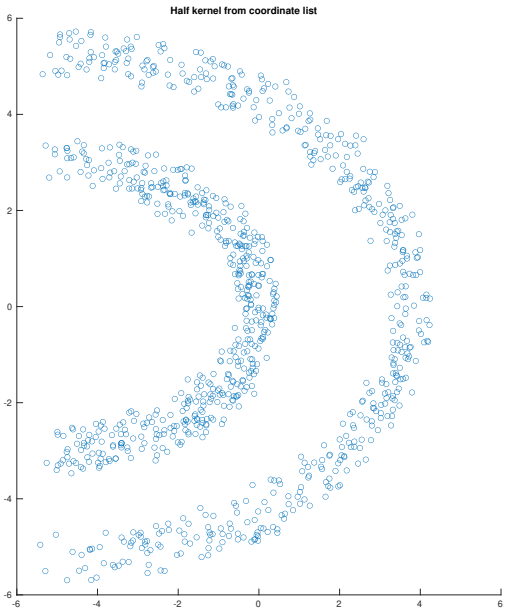
- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, MATLAB). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file 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]

All solutions provided from *ClusterPoints.m*, *epsilonSimGraph.m* source code.
A seed for the random number generator has been set $rng(20020309)$, for the results to be replicable.

1.1. 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.

Figure 1: Halfkernel non-convex set



The two obvious clusters are the two half circles shown in Figure 1.
The difficulty of a clustering algorithm in this scenario, particularly the k-means clustering algorithm, is that centroids would be in the upper and lower half of the graph, and the clustering would assign points to the nearest centroids dividing the graph in half on line $y = 0$.
Dividing the halfkernel with k-means for $K = 2$ and make each semi circle be a cluster is unfeasable.

1.2. Find the minimal spanning tree of the full graph. Use this information to determine the ϵ factor for the ϵ-similarity graph.

```
1 mst = minSpanTree(S);
2 epsilon = max(mst, [], "all");
```

Listing 1: ϵ neighboorhood graph function

Generally, set ϵ to the length of the longest edge in the minimal spanning tree, so the ϵ neighborhood graph is safely connected.
The heuristic followed to get ϵ in this case is: get matrix S from the gaussian similarity function and calculate the maximum largest element. Thus, ϵ is directly influenced by σ.
It provide use with reasonably connected ϵ similarity graph due to the process the gaussian similarity function does to generate S.

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

1.3. Complete the Matlab function $epsilonSimGraph()$. It should generate the similarity matrix of the ϵ-similarity graph.

```
function [G] = epsilonSimGraph(epsilon,Pts)
    n = length(Pts);
    G = zeros(n, n);
    for i = 1:n
        for j = 1:n
            dist = norm(Pts(i,:) - Pts(j,:));
            if dist < epsilon
                G(i,j) = 1;
                G(j,i) = 1;
            end
        end
    end
end
```

Listing 2: ϵ neighboorhood graph function

In code snippet 2, to create the ϵ neighboorhood graph function connect all points whose pairwise distances are smaller than ϵ.

1.4. Create the adjacency matrix for the ϵ similarity graph and visualize the resulting graph using the function *gplotg*().

```
[G] = epsilonSimGraph(epsilon, Pts);
W = S .* G;
gplotg(W, Pts(:,1:2));
title('Visualize adjacency matrix');
```

Listing 3: ϵ neighboorhood graph function

In code snippet 3, notice the element-wise multiplication between the similarity matrix S and the ϵ connectivity graph G.

1.5. 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.

```
[L, ~] = CreateLapl(W);
[U, ~] = eigs(L, K, "smallestabs");
```

Listing 4: ϵ neighboorhood graph function

In code snippet 4, create the Laplacian matrix L and retrieve the first K eigenvectors of L as columns of matrix U in ascending order.

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

```
[~, x_spec] = kmeans_mod(U, K, n);
[D_centroids_kmeans, x_kmeans] = kmeans_mod(Pts, K, n);
figure; gplotmap(W, Pts, x_spec); title('Spectral clusters')
figure; gplotmap(W, Pts, x_kmeans); title('K-means clusters')
```

Listing 5: ϵ neighboorhood graph function

In code snippet 5, cluster rows of eigenvector matrix of L corresponding to K smallest eigenvalues in U and get *x_spec*, *x_kmeans* to plot the mappings using adjacency graph W and poits Pts.

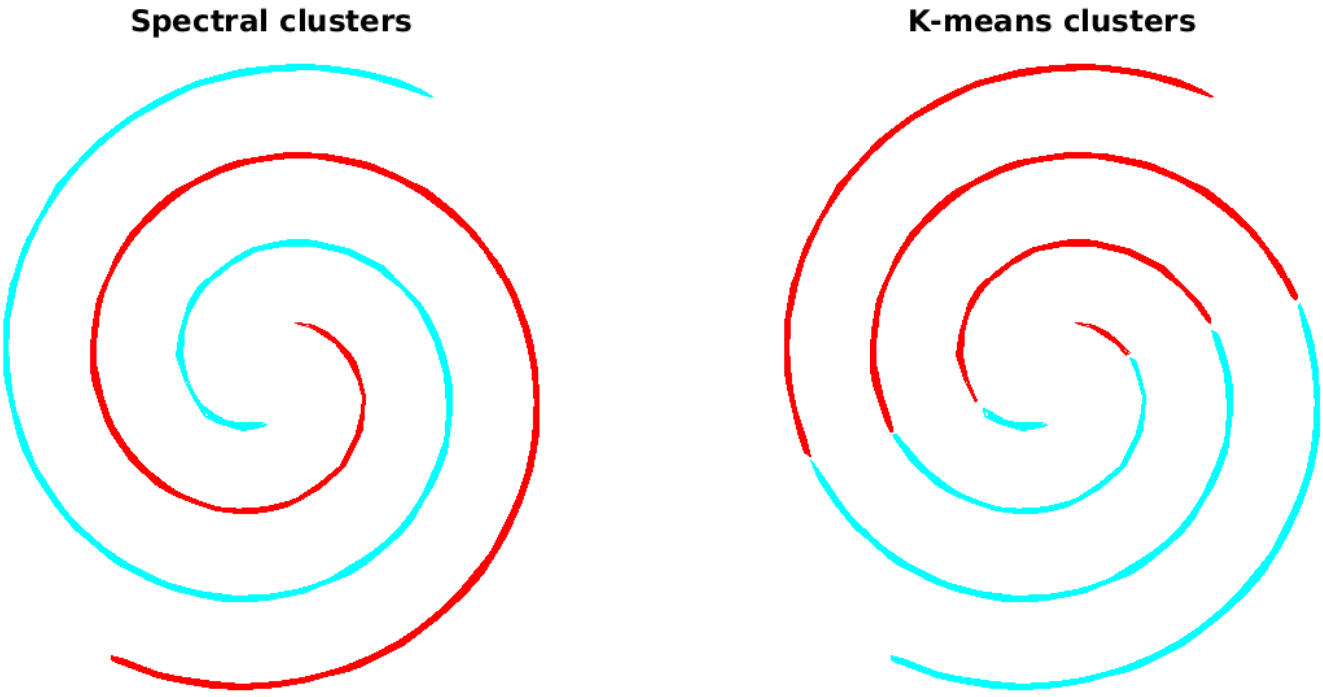
1.7. Cluster the datasets: Two spirals, Cluster in cluster, Crescent & full moon in $K = 2$ clusters, and visualize the results obtained with spectral clustering and k-means directly on the input data. Do the same for Corners, and Outlier for $K = 4$ clusters. Include the values of parameters ϵ, σ

A seed for the random number generator has been set $rng(20020309)$, for the results to be replicable. Table 1 shows the values of ϵ and σ used for each dataset.

Table 1: Parameters ϵ, σ, K values

Dataset	σ	ϵ	K
two spirals	7.600	0.687	2
cluster in cluster	6.919	0.632	2
crescent & full moon	7.389	0.595	2
corners	6.907	0.547	4
outlier	54.598	0.984	4

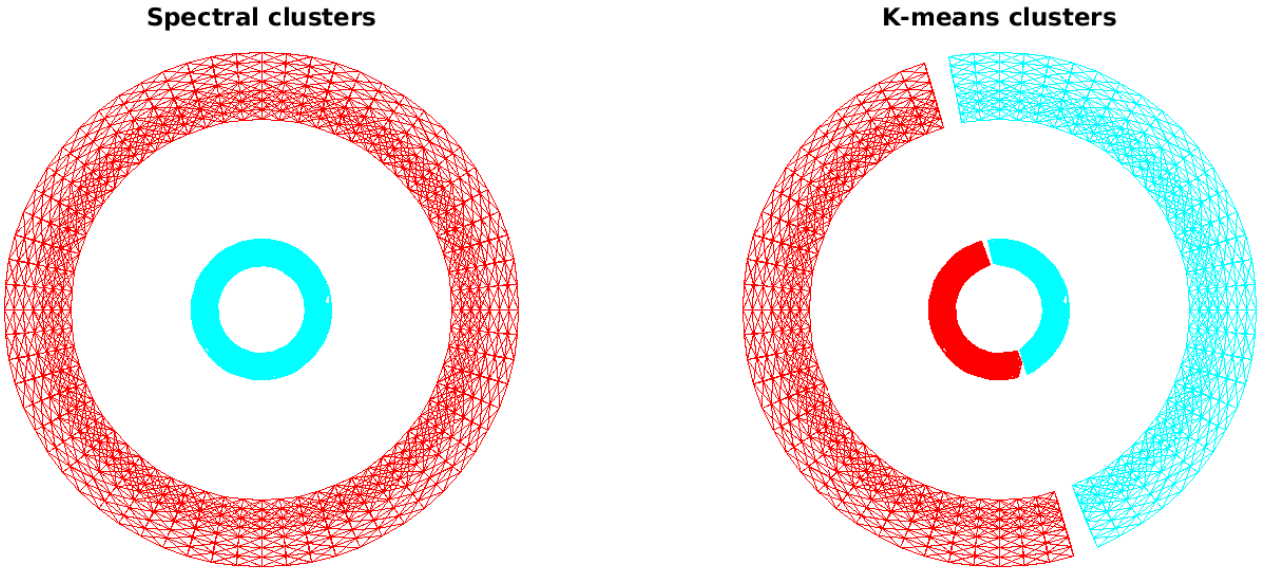
Figure 2: Two Spirals cluster, $K = 2$



Set $\sigma = log(n)$ by default.

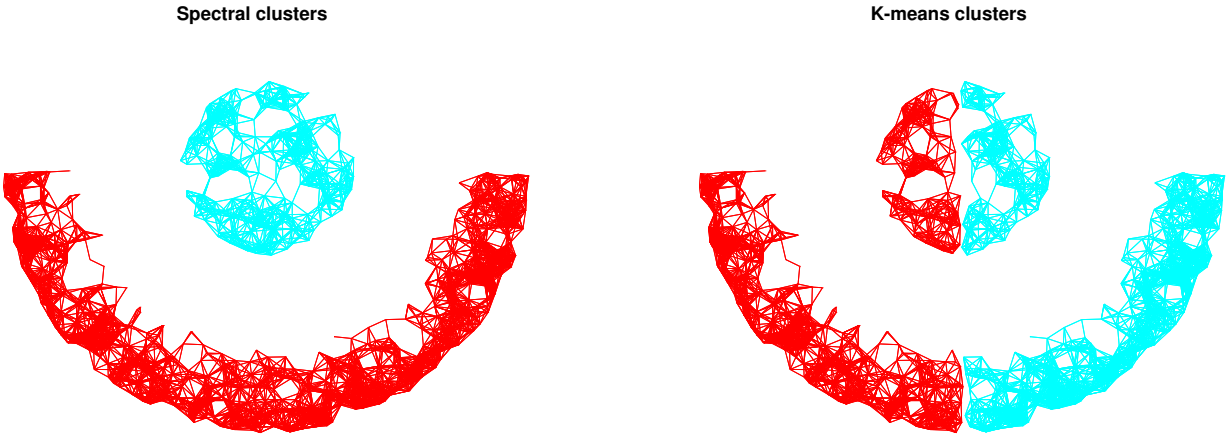
Dataset *twospirals* in Figure 2 has its spectral clusters assigning each cluster to a spiral, hence being visually logical. The k-means clusters are instead divided diagonally with a clear line separating the two spirals in half, not following the visually coordinate structured as spectral clustering. Cetroids are placed in the middle of the cluster, hence one for each half of the graph.

Figure 3: Cluster in Cluster cluster, $K = 2$



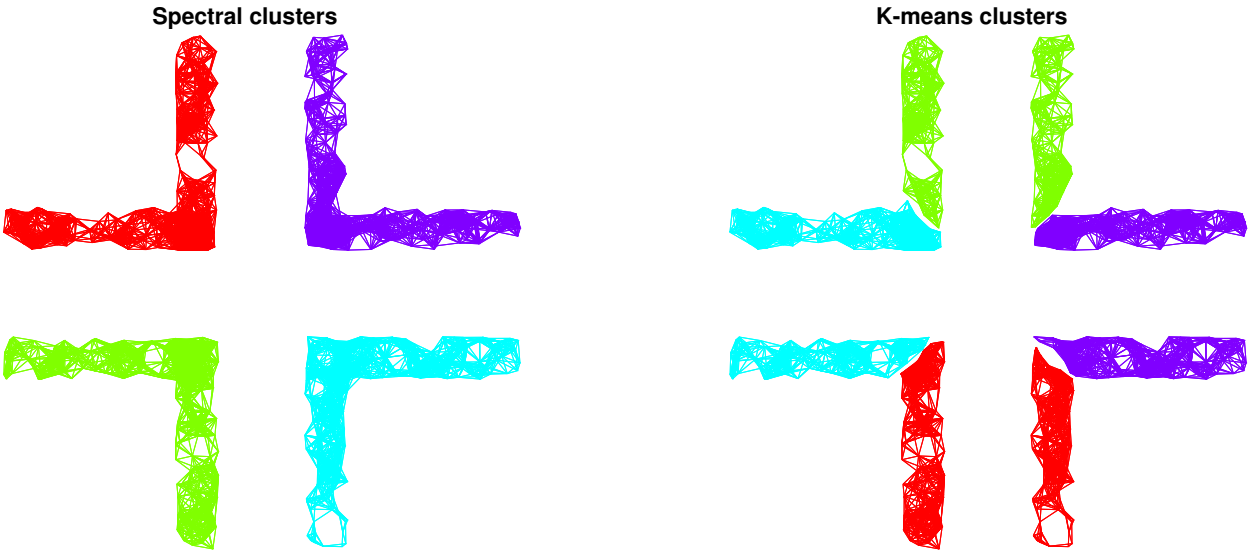
Set $\sigma = log(n)$ by default.
Dataset *clusterincluster* in Figure 3 has its spectral clusters assigning each cluster to its logical visual cluster representation. The k-means clusters are instead dividing again the dataset with a clear line separating diagonally in half the two nested clusters. For both the inner cluster is colored denser because of the higher number of data points in the inner cluster. Again, blame the centroids.

Figure 4: Crescent & Full Moon cluster, $K = 2$



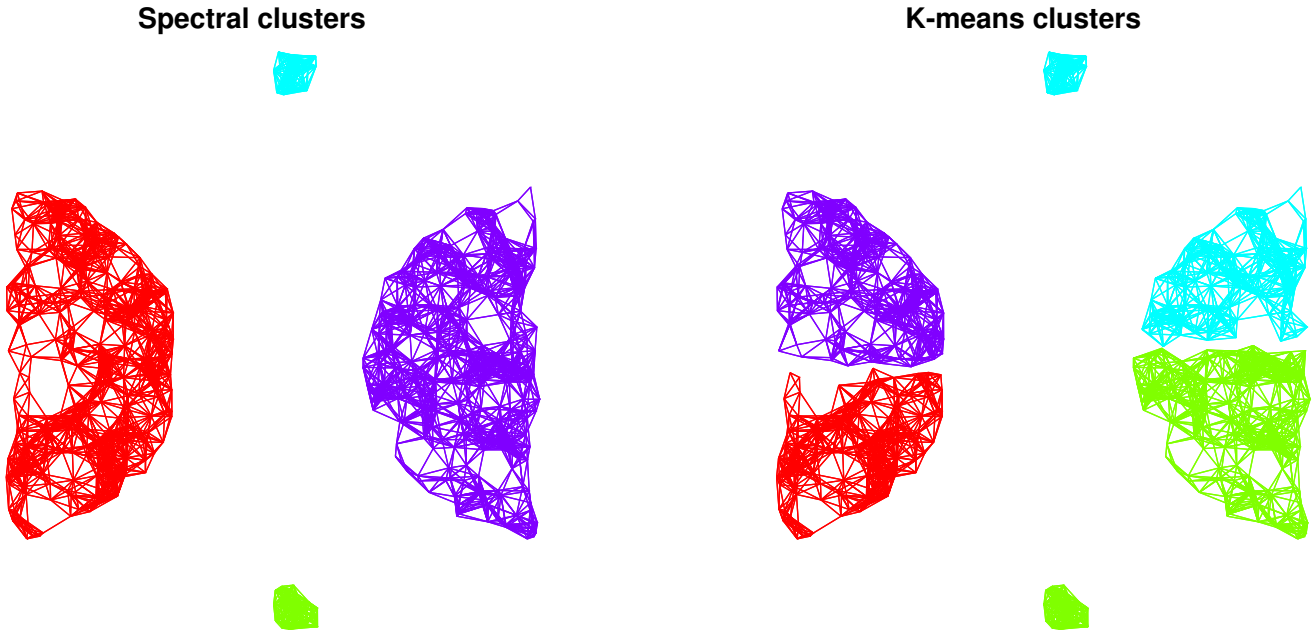
Set $\sigma = exp(K)$ to increase the width of the neighboorhood connectivity and make it dependent to the number of cluster K and solve the ill conditioned problem.
Dataset *crescentfullmoon* in Figure 4 has its spectral clusters assigning each data point to either the full moon of the crescent moon, hence being visually logical. The k-means clusters are instead dividing again the dataset in half. Again, blame the cetroids.

Figure 5: Corners cluster, $K = 4$



Set $\sigma = log(n)$ by default.
Dataset *corners* in Figure 5 has its spectral clusters assigning each data point to an "L" shaped figure cluster, hence being visually logical since the graph is visually interconnected as this. The k-means clusters are instead dividing each "L" shaped figure in half, with the centroids placed between one "L" shaped figure and another, thus connecting near "L" shape halfs.

Figure 6: Outlier cluster, $K = 4$



Set $\sigma = exp(K)$ to increase the width of the neighboorhood connectivity, make it dependent to the number of cluster K and solve the ill conditioned problem.
Dataset *outlier* in Figure 6 has its spectral clusters assigning each data point to an isolated group of the visually diveded 4 graph density agglomerates. The k-means clusters are instead having trouble to find visually meaninful structure.

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

All solutions provided from *ClusterGraphs.m*, *ClusterMetrics.m* source code.
A seed for the random number generator has been set $rng(20020309)$, for the results to be replicable.

2.1. Construct the Laplacian matrix, and draw the graphs using the eigenvectors to supply coordinates. Plot your results for all the three additionally supplied meshes: grid2, barth, 3elt.

Figure 7: Mesh airfoil1

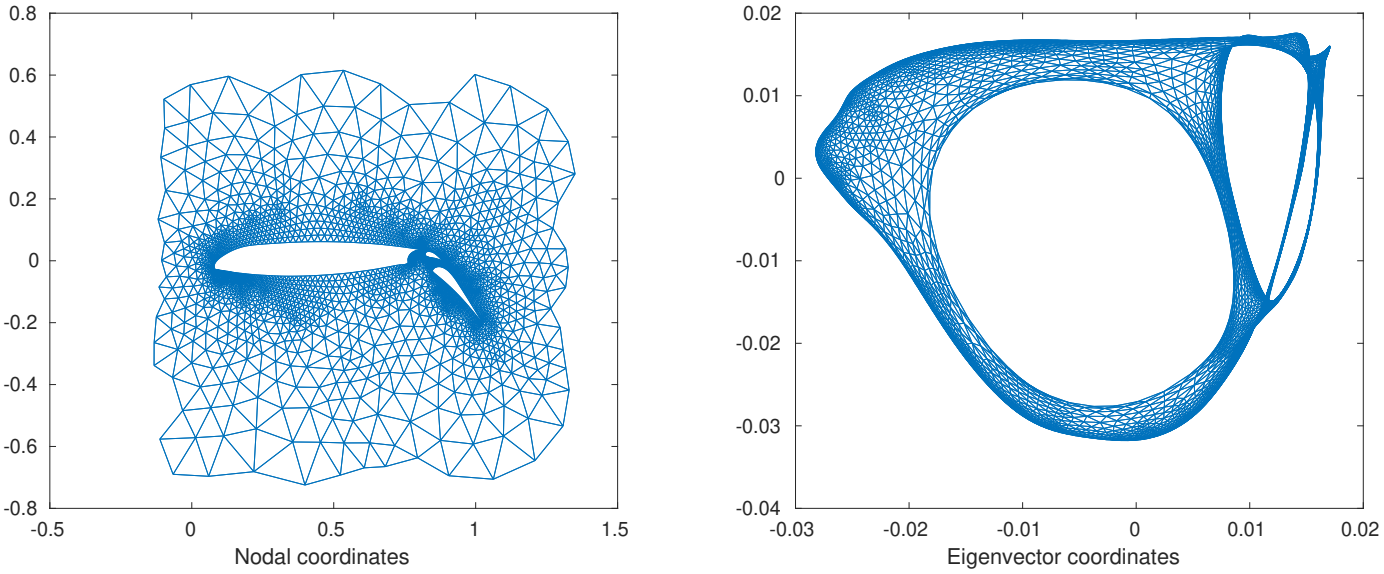


Figure 8: Mesh barth

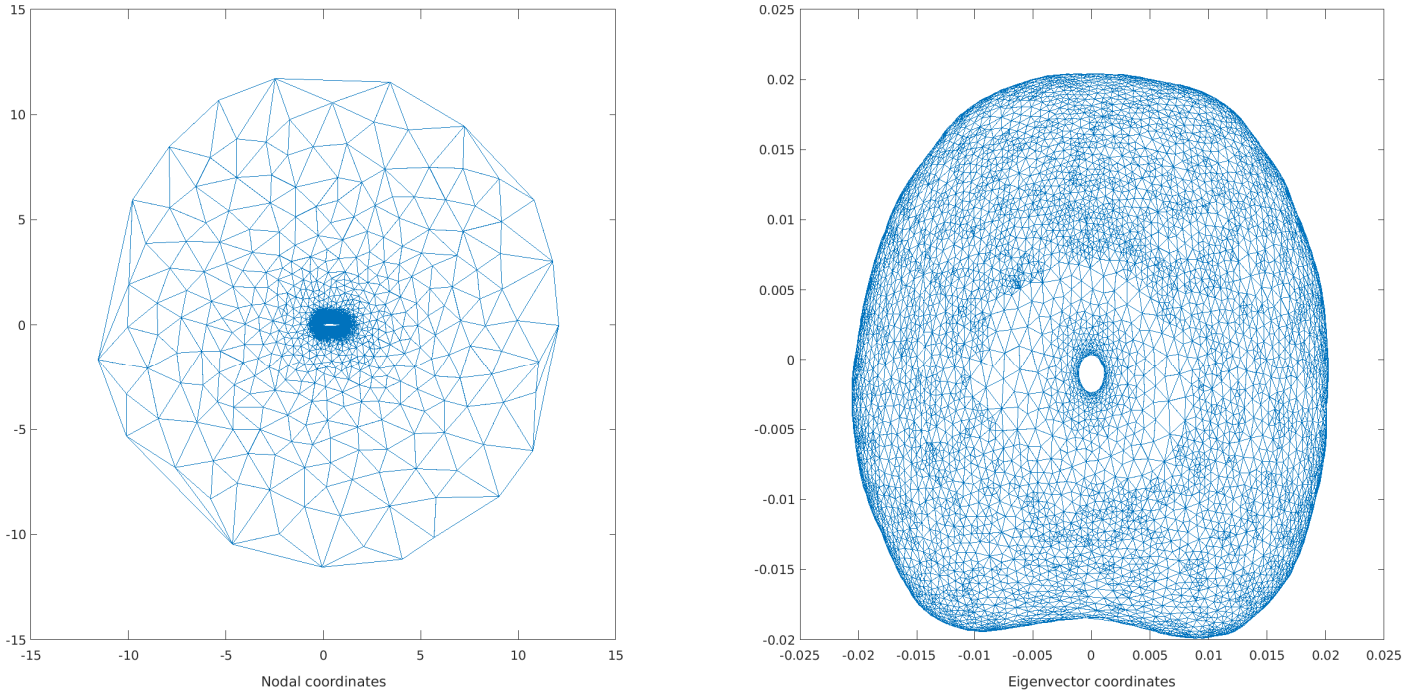


Figure 9: Mesh grid2

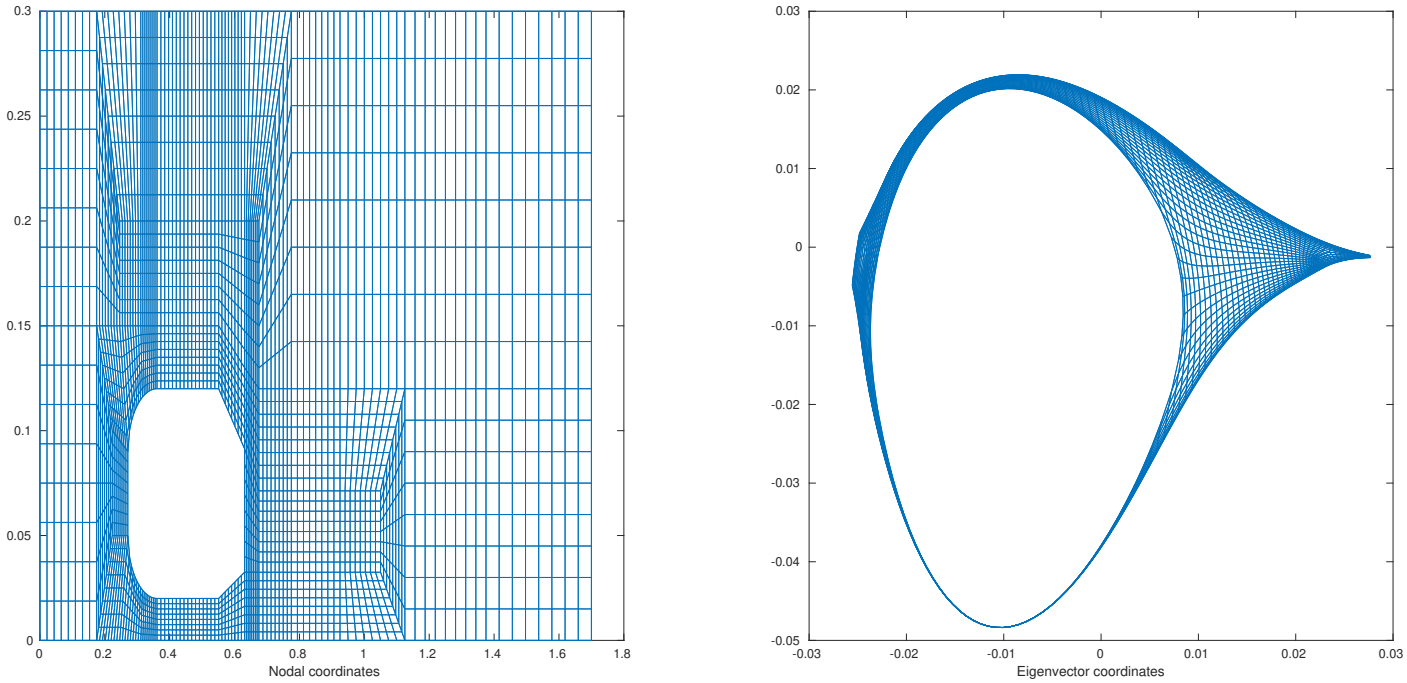
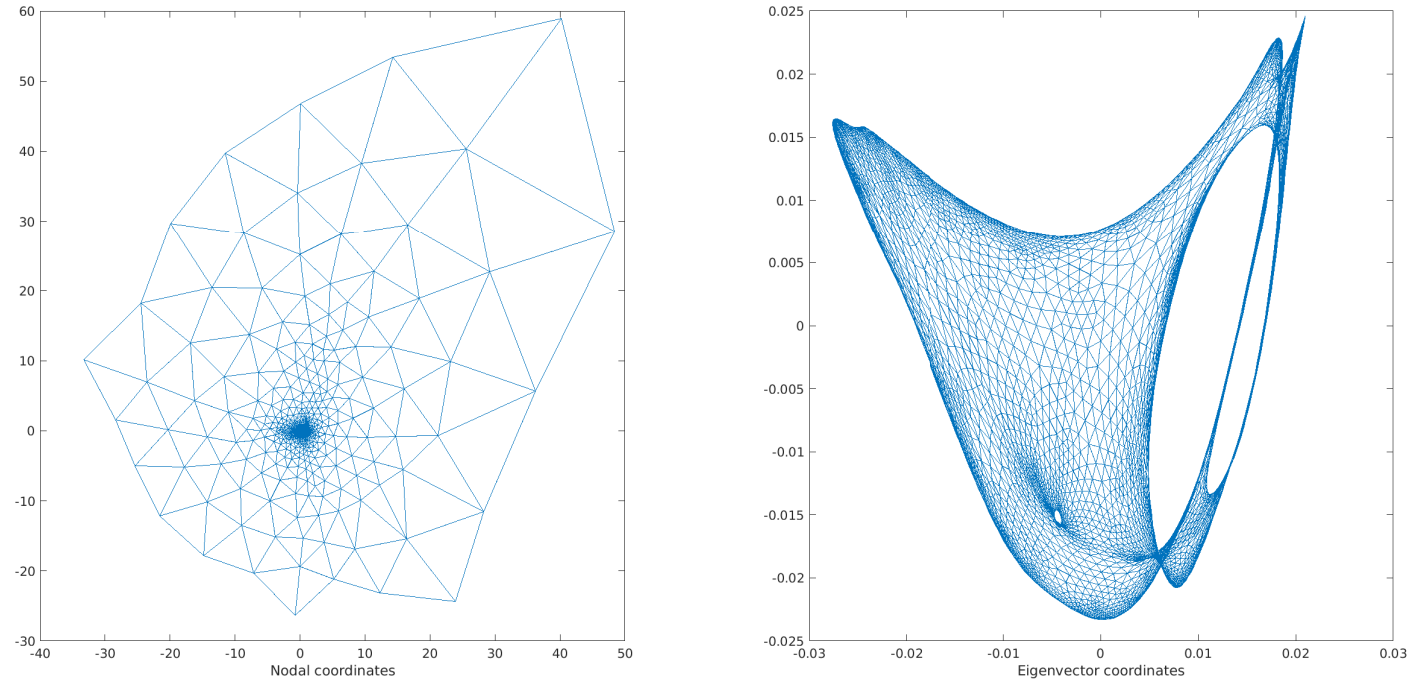


Figure 10: Mesh 3elt



2.2. Cluster each graph in $K = 4$ clusters with the spectral and the k-means method. Plot all of your results and describe the differences that you can see in the output of the 2 different algorithms and where they might come from.

Figure 11: Plot - Mesh airfoil1

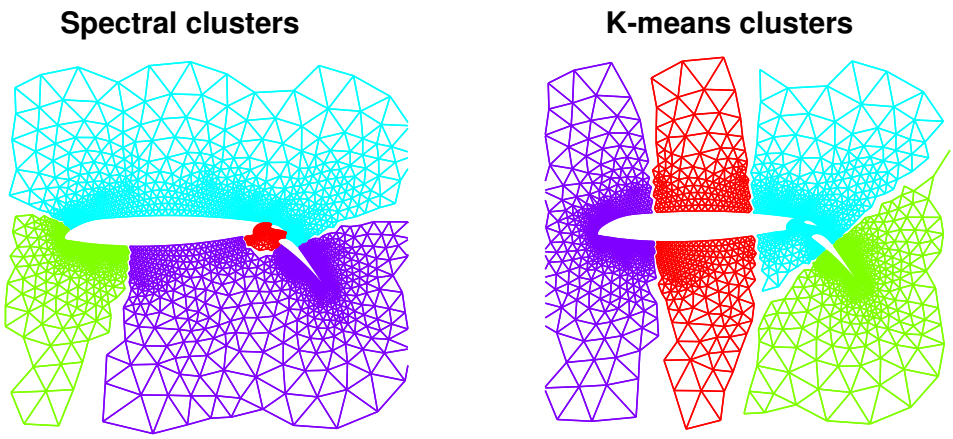
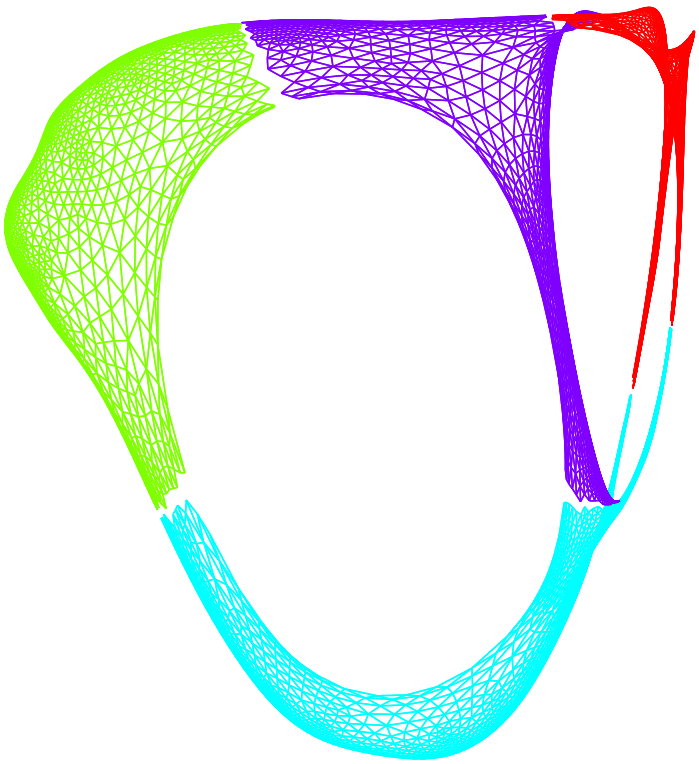


Figure 12: Plot eigvec coords - Mesh airfoil1 spectral clusters



Clustering for *airfoil1* mesh in Figure 11 Figure 12 show such reasoning.
The k-means clustering instead, gives a more pleasing visual result as the random nature of the centroids position leverage such peculiarity.

Figure 13: Plot - Mesh barth

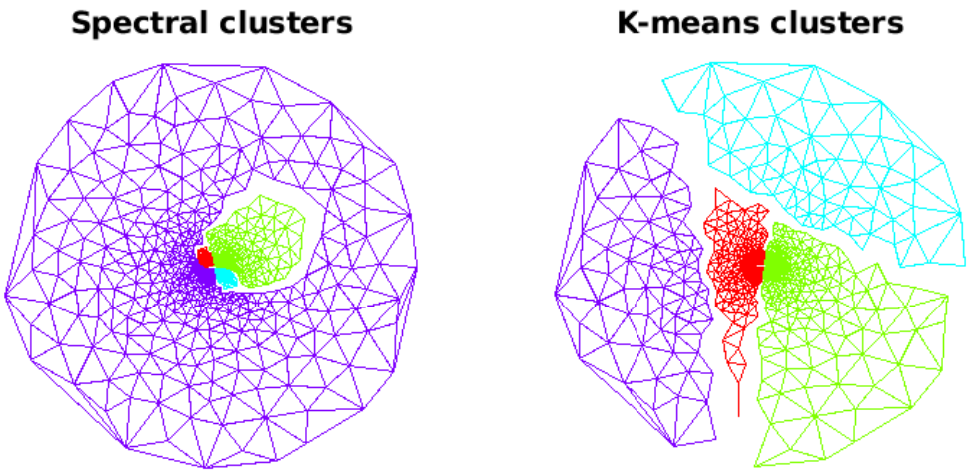
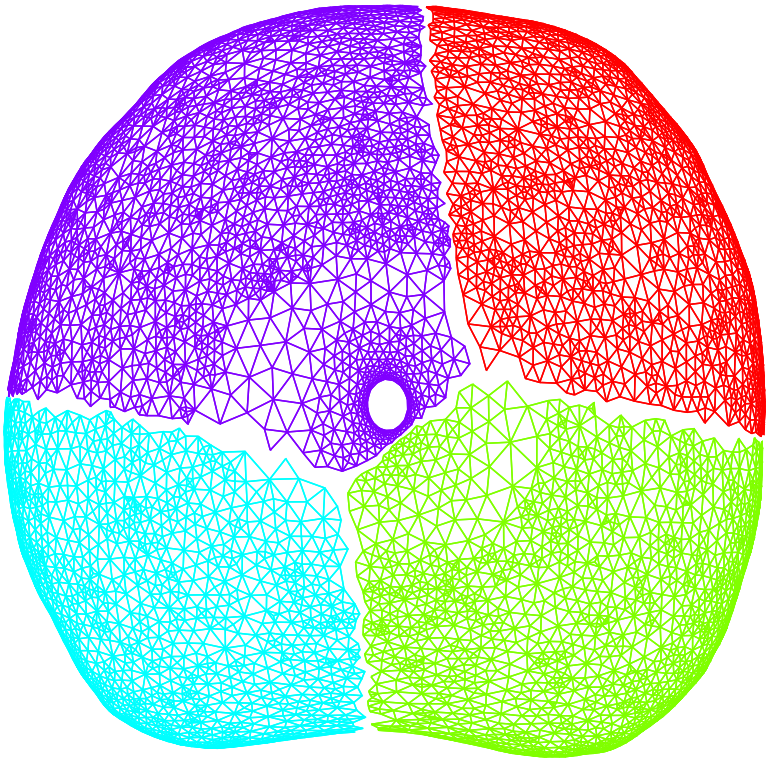


Figure 14: Plot eigvec coords - Mesh barth spectral clusters



Clustering for *barth* mesh in Figure 13 presents some particurality: the spectral clustering considers the graph density as in Figure 8, which is really high at the center of the mesh, hence, though we hardly see the 4 clusters, they are balanced divided in density in comparison with one another. Figure 14 show such reasoning.
The k-means clustering instead, gives a more pleasing visual result as the random nature of the centroids position leverage such peculiarity.

Figure 15: Plot - Mesh grid2

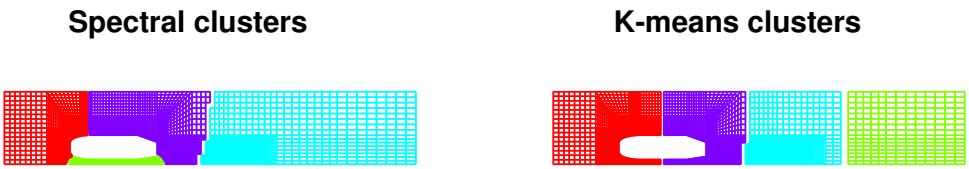
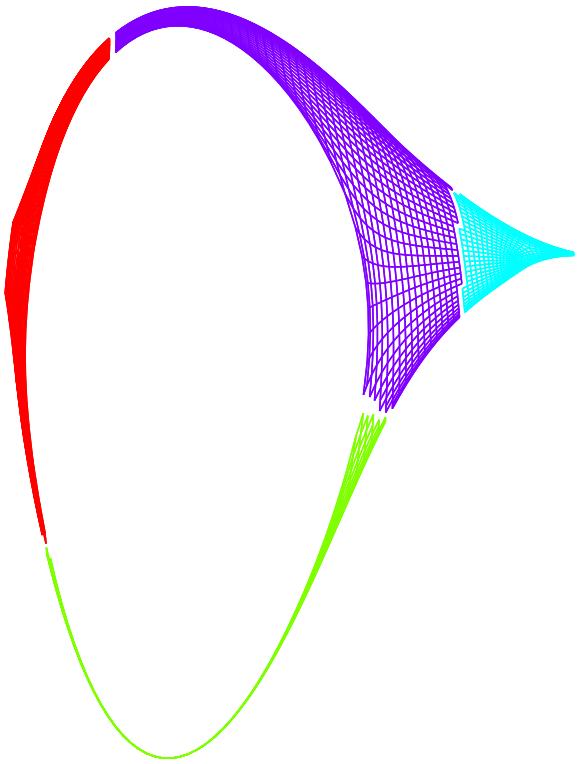


Figure 16: Plot eigvec coords - Mesh grid2 spectral clusters



Clustering for the *grid2* mesh in Figure 15 is the most visully balanced of all the meshes provided: that is visually inferrable from its nodal coordinates in Figure 9 that k-means clustering take in consideration, hence its geometrical distance.
The spectral clustering have the peculiarity of marking the right large area of graph as a single cluster and the base under the "bridge" as another one, which is again derived and visible from the structure of the plotted graph using its eigenvector coordinates in Figure 9.

Figure 17: Plot - Mesh 3elt

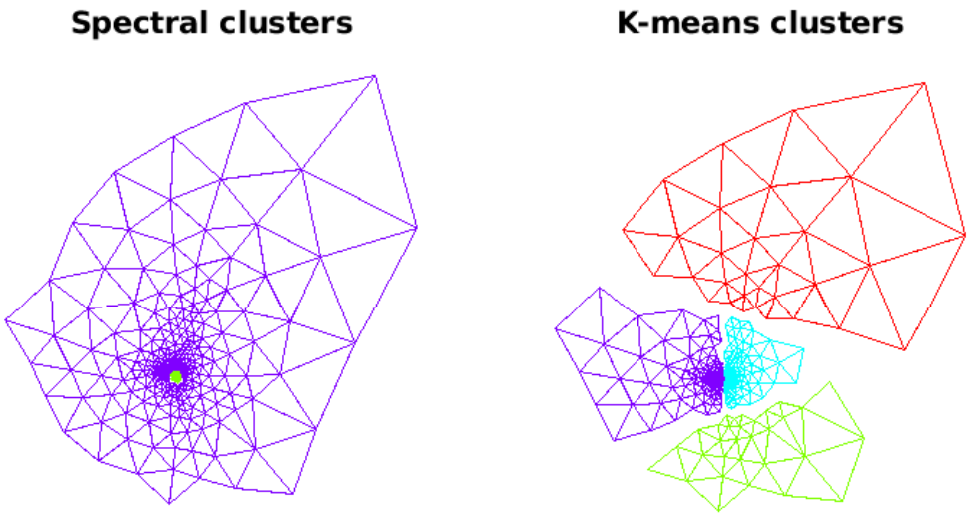
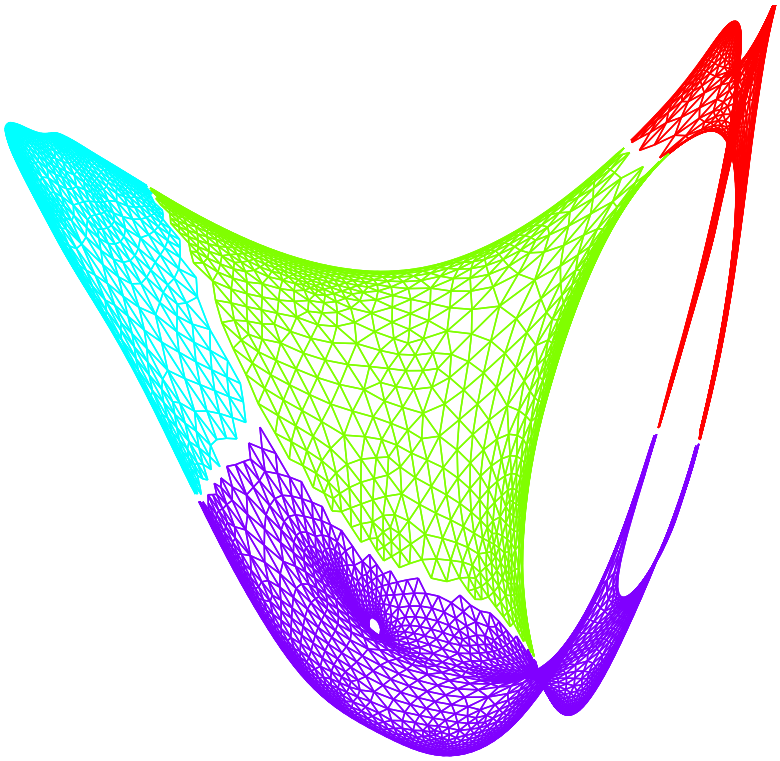


Figure 18: Plot eigvec coords - Mesh 3elt spectral clusters



Clustering for the *3elt* mesh in Figure 17 has the spectral clustering presenting the most visually non intuitive clustering: there are 4 clusters but they are almost inexistent visually, with one outer cluster predominating and another in the center of the graph with the other 2 not visible from the plot. This phenomenon is explainable again by looking at the eigenvector coordinates plot in Figure 10 as the mesh is highly dense in the center while the outer part is almost visually sparse.

2.3. Report in Table 2 the number of nodes per cluster and plot a histogram of the reported values. What can you observe?

A seed for the random number generator has been set $rng(20020309)$, for the results to be replicable.

Table 2: Clustering results, $K = 4$

Case	Spectral	K-Means
airfoil1	1150, 1050, 971, 1082	1846, 1292, 412, 703
grid2	785, 1305, 827, 379	604, 1183, 1271, 238
barth	1407, 2206, 1490, 1588	65, 71, 3884, 2671
3elt	874, 1794, 965, 1087	3098, 1560, 31, 31

Figure 19: airfoil1 histogram - nodes per cluster

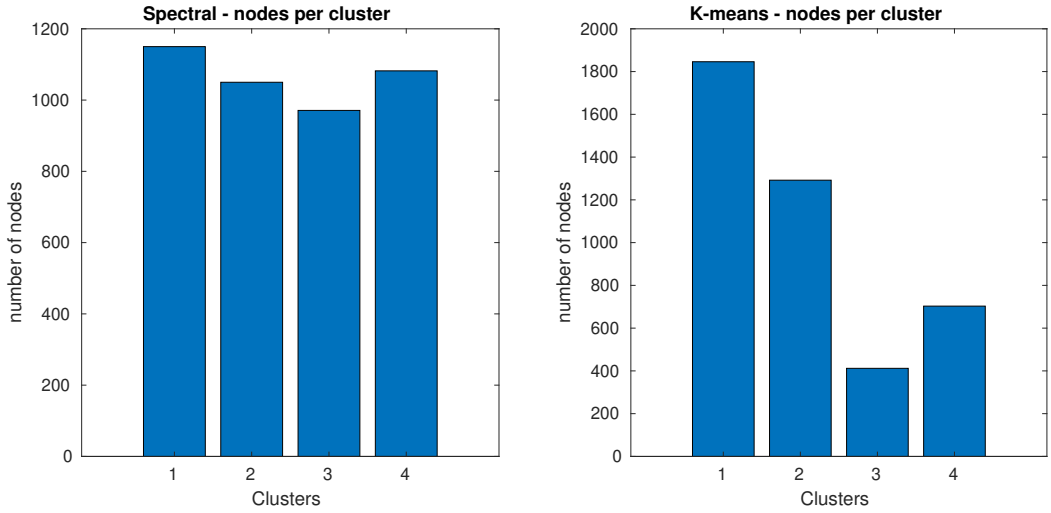
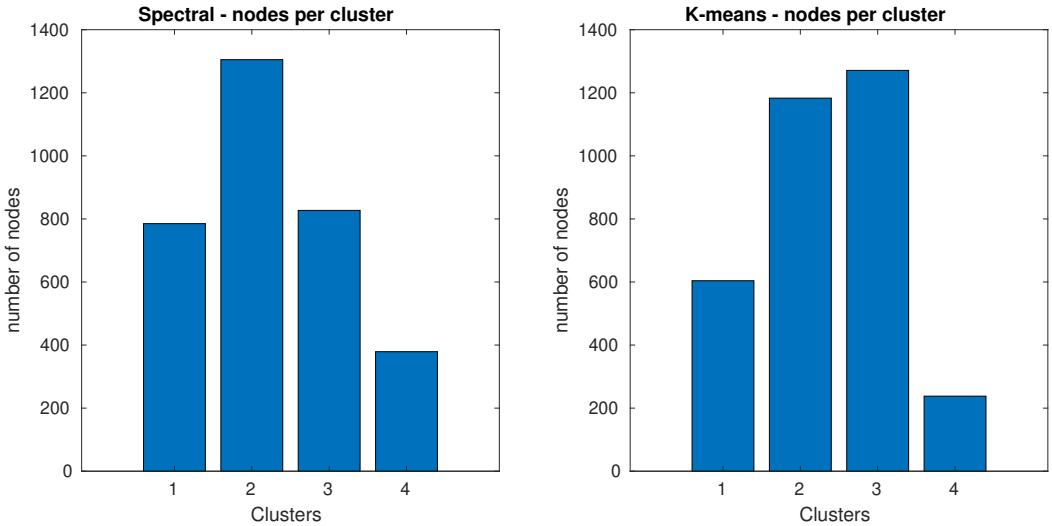
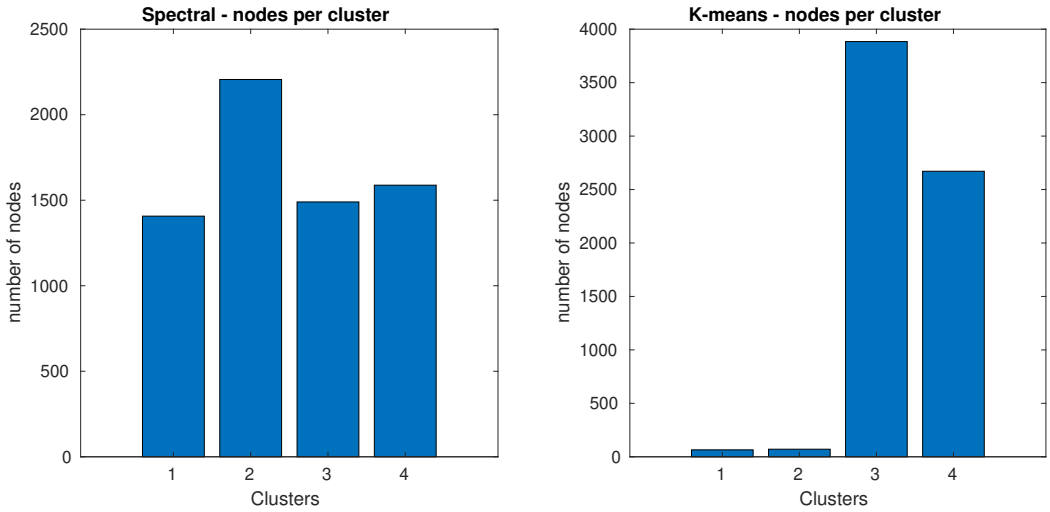


Figure 20: grid2 histogram - nodes per cluster



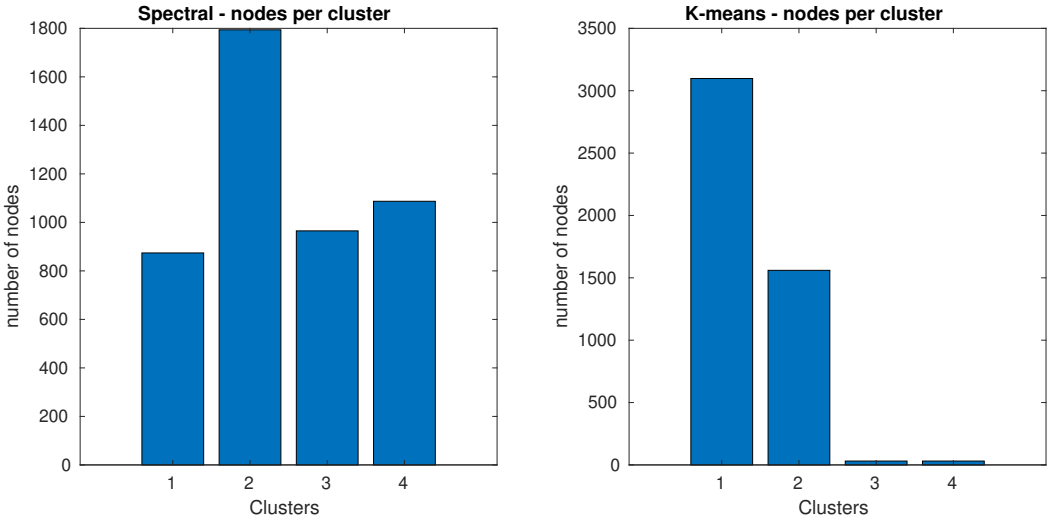
In Figure 20, both the clustering algorithms are presenting unbalanced clusters, with the k-means clustering presenting the maximum difference in number of nodes between the clusters:
k-means: $max\#nodes - min\#nodes = 1271 - 238 = 1033$
spectral: $max\#nodes - min\#nodes = 1305 - 379 = 926$
Such result from k-means is what we expected, though the difference in number of nodes is generally expected to be lower on the spectral clustering algorithm since is known for its stable and diffuse cluster results.

Figure 21: barth histogram - nodes per cluster



In Figure 21, the stability expected from the spectral clustering algorithm is met and the clusters for this example are the most balanced of all the meshes provided. Instead, the k-means clustering algorithm is clearly presenting a discrepancy in balance of number of nodes per cluster, with the first 2 having very few almost neglectible number of nodes and the last 2 comprending most of them.

Figure 22: 3elt histogram - nodes per cluster



In Figure 22, the stability and balance expected from the spectral clustering algorithm holds with only cluster #2 being an bit of an exception. Instead, the k-means clustering algorithm is again presenting an evident discrepancy in balance of number of nodes per cluster.