Spectral Clustering Homework

Romain Emanuele Salvi
Politecnico di Torino
Student ID: s339304
s339304@studenti.polito.it

Mattia Sernia
Politecnico di Torino
Student ID: s339829
s339829@studenti.polito.it

***Abstract*—This paper is a technical report on the spectral clustering homework of the course 'Computational Linear Algebra for Large Scale Problems', A.A. 2024-2025. The report describes the process applied to solve the tasks, the results obtained with different datasets, and the comparison of the results of our implementation of the spectral clustering with the outcomes of other clustering methods, such as DBSCAN and K-means. We also included all of the optional points of the spectral clustering homework in this report.**

I. Definition and creation of the data

To apply the spectral clustering algorithm, we were provided with two datasets: the first, referred to as "Circle", and the second, referred to as "Spiral". The datasets are matrices of dimensions 900×2 and 312×3, respectively. The third column of the Spiral dataset contains the ground truth for the correct clustering of its points, so we separated it from the actual values of the data. So, we stored a vector containing the correct cluster assignments, which would be useful to evaluate our clustering technique. To further test the spectral clustering algorithm we created, we decided to apply it on one more dataset. The "Iris" dataset we used is a modified version of the well-known *Iris* dataset introduced by Fisher in 1936, which contains 150 samples of iris flowers, each described by four features. To create a three-dimensional version of this dataset, we applied Principal Component Analysis (PCA). After performing PCA and reducing the dataset to three principal components, we obtained a 150×3 matrix, where each column represents the values of the corresponding principal component for each data point. We also stored the correct cluster assignments of the data points in an array, following the same process applied for the Spiral dataset.

We visualized these datasets using scatter plots, where the first and second columns of each matrix, labeled 'X1' and 'X2', were plotted along the x- and y-axes. In the case of the Iris dataset, we used a three-dimensional scatter graph to better visualize the distribution of the points. We labeled the third column as 'X3' and plotted its values along the z-axis. Figures 1, 2 and 3 show the resulting scatter plots.

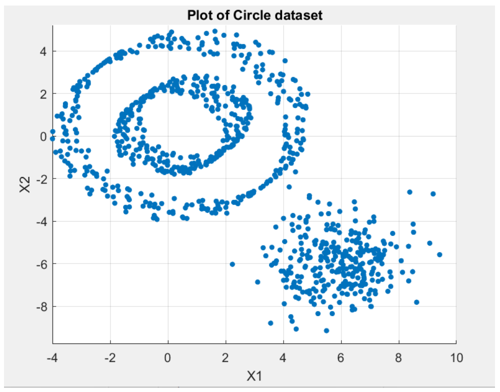


Fig. 1. Scatter plot of the Circle dataset

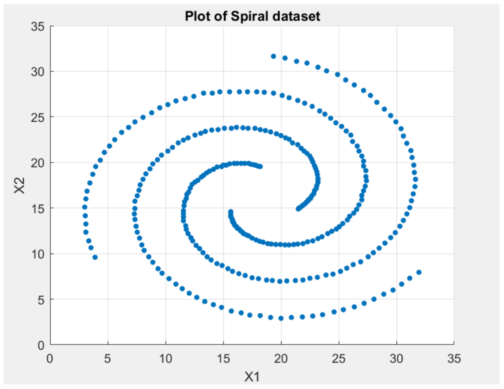


Fig. 2. Scatter plot of the Spiral dataset

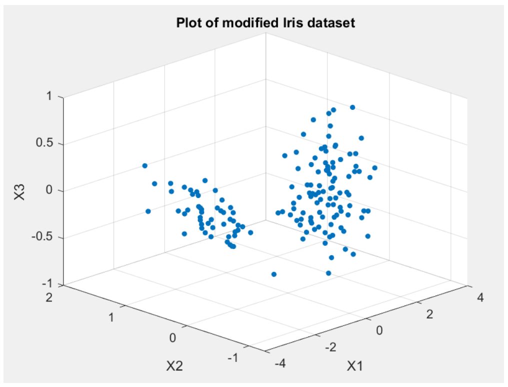


Fig. 3. Scatter plot of the modified Iris dataset

II. Implementation of Spectral Clustering

A. Construction of the k-nearest neighborhood (k-nn) similarity graph and its adjacency matrix

The first step in the process of spectral clustering is the creation of the k-nearest neighborhood similarity graph. In this graph $G = (V, E)$, each vertex $Vi \in V$ represents a data point Xi. An edge between two vertices vi and vj exists if the similarity sij between the corresponding data points, Xi and Xj, is within the top k of vi, this being a k-nearest neighbor graph. We assumed that $sij = sji$ and that the edge connecting vi and vj is weighted by sij. This way, the graph is undirected. Given a dataset X, the similarity function we used for the computation of the weights is the following:

$$s_{i,j} = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right),$$

where X_i and X_j are two points of the dataset X. As mentioned before, for each node we only kept the k closest points. The values of k utilized for this homework are 10, 20, 40. In our case, because the similarity is higher if the points are closer in the two-dimensional space where they are defined, we considered for each node the k edges with the highest similarity (weight). Once we obtained this matrix, we used it to derive the weighted adjacency matrix W which we would then use to create the graph. W is defined as $Wij = sij$, if $i \neq j$ and $Wij = 0$, if $i = j$. One last step was required to finally create the graph: we needed to make W symmetric. Indeed, Xi could be among the k closest neighbors of a node Xj, but it was not guaranteed that Xj would be in the closest neighbors of Xi. To make W symmetric there are two different approaches: the first one is to assume that if the node Xj is among the k closest neighbors of Xi, Xi should be in the k closest neighbors of Xj as well. All elements of the matrix W will follow this rule: $\forall i, j \; Wij = Wji = max(Wij, Wji)$, this way it is possible that a node Xi has more than k neighbors. The other method works in the opposite way, as a node Xj is in the k closest neighbors of Xi only if the node Xi is in the k closest neighbors of Xj. We chose to use the first method because it ensures robust graph connectivity by preserving all significant edges and provides a more accurate representation of asymmetric relationships within the data.

Once derived W, we created and plotted the k-nn similarity graph. Figure 4 shows the result of the plot with $k = 10$ for the Circle dataset. The number of connected components n is also shown as subtitle of the graph. The value of n is always shown in the MATLAB application we designed when spectral clustering is run.

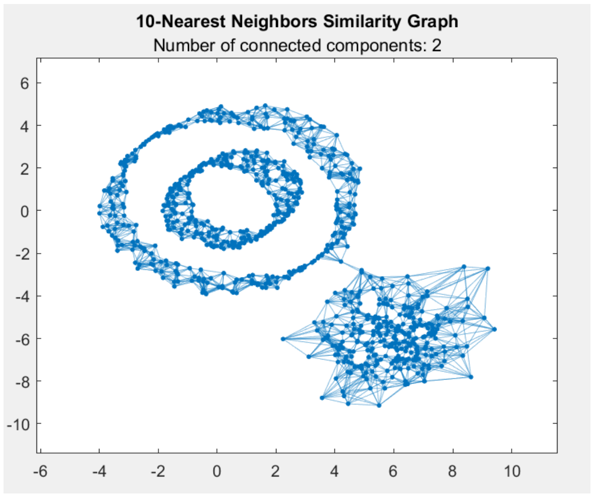


Fig. 4. Visual representation of the k-nearest neighbors graph with $k = 10$ for the Circle dataset

B. Construction of the degree matrix and the Laplacian matrix

In a weighted undirected k-nn graph, the degree matrix D is a diagonal matrix whose values are the sum of the weights of the edges connected to a vertex. So, we summed the weights along each row (which is a vertex on the graph and consequently a data point) and obtained D. This is the formula used:

$$d_i = \sum_{j=1}^{m} w_{i,j}$$

with d_i being the element on the i_{th} row of the i_{th} column of D. Having D, we then computed the graph's unnormalized Laplacian matrix L, which can be written as $L = D - W$.

The Laplacian matrix is a representation of a graph that encodes some of its key structural properties, providing insight into the connectivity and behavior of the graph. The analysis of its eigenvalues provides important information about its connectivity: the number of eigenvalues equal to zero corresponds to the number of connected components in the graph, while all others are positive. This is because L is positive (its values are the sum of positive values to which smaller values are subtracted) and semidefinite. This property will be furtherly explained in Subsection II-C. We are trying to partition the graph into clusters, and the smallest eigenvalues help reveal clusters that are more tightly connected to each other than to the rest of the graph. For example, if there is a large gap between the values of the second-smallest and the third-smallest eigenvalues, this indicates that the division of the graph into two clusters is more adequate than in three. We will use this information to select the adequate number of clusters.

An example of the Laplacian matrix sparsity graph for $k = 10$ is shown for each dataset in figures 5, 6, 7. The plot of the graph for the Spiral dataset with $k = 40$ is shown in figure 8. Analyzing these images, we can visualize the nonzero elements in a matrix to see the sparsity pattern in it. We can also obtain information about the structure and the connections in the k-nn similarity graphs of the datasets. Figure 5 shows the sparsity pattern of the sparse Laplacian matrix derived from the Circle dataset, with a value of $k = 10$. The colored diagonal line that can be seen indicates the presence of groups of nodes that are tightly interconnected between each other, thus they likely follow a definite geometrical shape. The square at the bottom right of the plot, instead, indicates a group of points that is part of a ”blob”, a cloud of points where nodes are both strongly and weakly connected to each other depending on their position in the cloud: central or marginal. This analysis turns out to be correct, as can be seen in Figure 4, where two definite geometrical shapes and a cloud of points can be observed. For the Iris dataset the sparsity plot is composed of three main squares, with some ”noise” on the sides of the second and last square. We can therefore assume that the dataset has three main clusters of data, but while the first is well-defined and separated from the others, the second and the third are closer and more ambiguous. This is mostly true, as we can observe from Figure 3 that the data is composed of two main blocks, and if one were to move the three-dimensional scatter plot around, the second block divides more clearly into two different clusters.

To analyze the key differences in the Laplacian matrices of a dataset varying the number of neighbors, we included the sparsity pattern plot for the Spiral dataset with two different values of k : 10 and 40. In both plots, values around the diagonal of L are well defined, and the fact that they do not have non-zero values in the vicinities mean that the points follow a definite shape (which in this case is spirals). The main difference between the two cases is that, when we set $k = 40$, the sparsity pattern of the Laplacian shows that points might connect not only to its immediate neighbors along the same part of the spiral (preceding or subsequent nodes), but also to points on other spirals. These long-range connections manifest as non-zero entries far from the main diagonal in the Laplacian matrix. This phenomenon also occurs setting $k = 10$ but not as frequently as in the case of $k = 40$, and it is caused by the structure of the dataset: the spirals are concentric within each other, and this causes a point to connect with another even though they are not part of the same spiral.

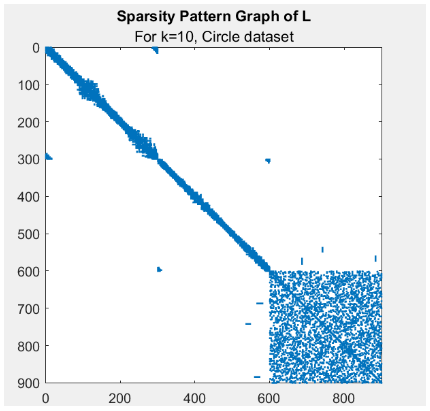


Fig. 5. Sparsity pattern graph of the Laplacian matrix with $k = 10$ for the Circle dataset.

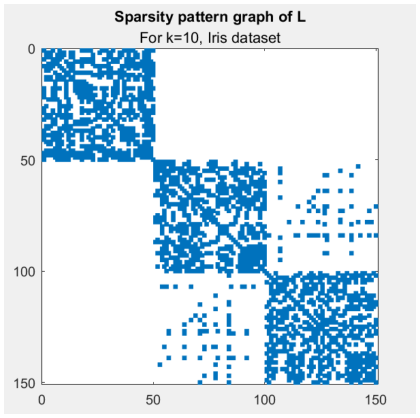


Fig. 6. Sparsity pattern graph of the Laplacian matrix with $k = 10$ for the Iris dataset.

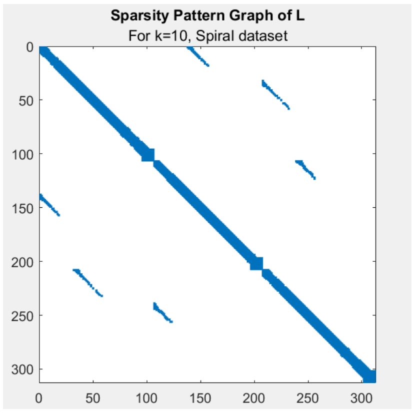


Fig. 7. Sparsity pattern graph of the Laplacian matrix with $k = 10$ for the Spiral dataset.

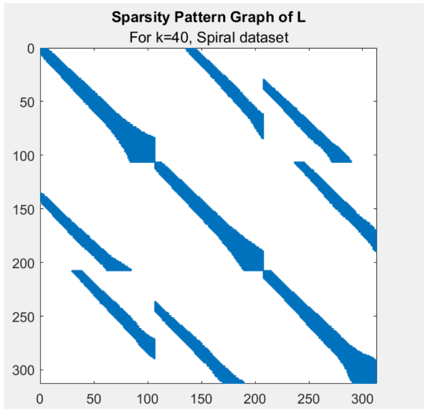


Fig. 8. Sparsity pattern graph of the Laplacian matrix with $k = 40$ for the Spiral dataset

In our case, the matrices W, D, and L are relatively large. Given a dataset of size $|X|$ and $k = 10, 20, 40$:

- W is a matrix of dimensions $|X| \times |X|$ with only about k non-zero elements in each row/column.
- D is a diagonal matrix of dimension $|X| \times |X|$.

- The Laplacian matrix L is the result of the difference between the previous two matrices, so it is $|X| \times |X|$ and has few non-zero elements as well.

For this reason, we decided to store them in the predefined MATLAB sparse format. This method stores only the nonzero elements of the matrix, together with their indices, and reduces computation time by eliminating operations on zero elements.

C. Self implementation of the numerical method for eigenvalue approximation

After computing the Laplacian matrix of the dataset, the next step was to calculate a set of eigenpairs. This was done to determine the optimal number of clusters and the distribution of points among them. The eigenvalues of the Laplacian matrix provide insight into the connectivity of subgraphs within the data. Specifically, $dim(ker(L))$ identifies the number of connected components of the graph, and $dim(ker(L)) \geq 1$ holds because the sum of the elements in each row of the matrix is zero. If an eigenvalue is 0, it implies the presence of a subgraph that is completely disconnected from the rest. On the other hand, if an eigenvalue is small, but not zero, it indicates a subgraph that is weakly connected to the others. A thorough analysis of the eigenvalues is necessary to decide the correct number of clusters. Typically, a threshold is chosen to determine if an eigenvalue is sufficiently small to correspond to a separate cluster. MATLAB offers several useful functions that can compute eigenvalues and eigenvectors, typically relying on the SVD algorithm. However, for our homework, we also opted to implement an algorithm based on the inverse power method and deflation technique as suggested in the third optional point. The power method approximates the largest eigenvalue (λ_1) and its corresponding eigenvector. Let's assume that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$, where $\lambda_2, \lambda_3, ..., \lambda_n$ are all the remaining eigenvalues in descending order. We know that each eigenvector is linearly independent, so we can define $v \in \mathbb{C}^n$ such that $v = \alpha_1 x_1 + \sum_{k=2}^{n} \alpha_k x_k$, where $x_1, x_2, ..., x_n$ are the eigenvectors corresponding to $\lambda_1, \lambda_2, ..., \lambda_n$. Let's assume that we manage to find a v with $\alpha_1 \neq 0$, we can define the sequence $v^{(i)}$ iteratively as: $v^{(0)} = v, \quad v^{(i+1)} = Av^{(i)}$ were A is the matrix containing the eigenvectors. Each term in the sequence can be written as $v^{(m)} = A^m v$, where m is the index of the iteration. Having defined $v^{(m)}$ we can apply the following theorem:

$$\lim_{m \to \inf} \frac{v^{(m)}}{||v||_2^2} = (-1)^s x_1,$$

where $s = sign(\lambda_1) sign(\alpha_1)$. Additionally, the Rayleigh quotient is defined as: $ra(v) = \frac{v^H Av}{||v||_2^2}$ and the following theorem holds:

$$\lim_{m \to \inf} ra(v^{(m)}) = \lambda_1.$$

With this foundation, we are able to implement an algorithm in MATLAB able to approximate the largest value eigenvalue using a maximum number of iterations and a relative tolerance to speed the process up while maintaining a proper level of precision.

However, our goal is to find the smallest possible eigenvalue, so we applied the inverse power method. The inverse power method works by executing the power method to the inverse matrix A^{-1}. This is based on the relationship that if $Ax_i = \lambda_i xi$, then $A^{-1}x_i = \frac{1}{\lambda_i} xi$, meaning that the eigenvalues of A^{-1} are reciprocals of the eigenvalues of A. Therefore, by finding the largest eigenvalue $\lambda_1^{'}$ of A^{-1} we can derive the smallest eigenvalue λ_n of the matrix A: $\lambda_n = \frac{1}{\lambda_1^{'}}$. But the inverse power method requires A to be invertible. Since the Laplacian matrix L has $\lambda_n = 0$ (because the matrix is symmetrical), the matrix L is not invertible. The eigenvector corresponding to λ_n is $x_n = 1_n$, being the sum of the elements of each row equal to 0, so the eigenvector corresponding to the smallest eigenvalue is known.

To find the second smallest eigenvalue of the matrix L we need to find a matrix that has the same eigenvalues of L, with the exception of $\lambda_n = 0$, and apply the inverse power method to it. To do this we use a technique called shifting, which is able to modify the value of only one eigenvalue knowing the corresponding eigenvector, if and only if the matrix is symmetric. Indeed, in this case this property holds: $X^T AX = D$, where X is the matrix that has as columns the eigenvector of the matrix, and D is a diagonal matrix with all the possible eigenvalues on the diagonal. The equation written before can also be displayed as follows: $A = XDX^T$. So the matrix A can be written as such:

$$A = \sum_{j=1}^{n} \lambda_j x_j x_j^T = \lambda_n x_n x_n^T + \sum_{j=1}^{n-1} \lambda_j x_j x_j^T.$$

So we can compute the matrix $A_1 = A + \alpha x_n x_n^T$, where α is a random number, and the eigenvalues of the matrix A_1 will be $\lambda_1, \lambda_2, ..., \lambda_n + \alpha$. If $\alpha > \lambda_1$ we assure that the largest eigenvalue of A_1 is $\lambda_n + \alpha$, and the second smallest eigenvalue of A called λ_{n-1} becomes the smallest eigenvalue of A_1, and it can be found by applying the inverse power method on A_1.

To implement this algorithm in our code, we first applied the power method on the matrix L, which is symmetrical, to find the largest eigenvalue λ_1. After that, since we know both $\lambda_n = 0$ and its eigenvector $x_n = 1_n$, we used the shifting technique to the matrix L:

$$L_1 = L + \frac{3}{2} \lambda_1 x_n x_n^T$$

to shift the eigenvalues λ_n. After that we applied the inverse power method to the matrix L_1 to find λ_{n-1} and we repeat this process ten times to get the ten smallest eigenvalues of L and their corresponding eigenvectors. It is worth noting that we initially considered using 1_n as the starting vector for the inverse power method. However, since 1_n is the eigenvector corresponding to $\lambda = 0$ the algorithm would converge to this eigenvector, which we aimed to avoid. To address this issue, we chose an alternative initialization vector IV, defined as 1_{n-1} with a zero appended as the last element.

D. Clusters construction using M and U

We chose a significant number of eigenvalues and their corresponding eigenvectors (10), so that we could select the number of clusters, M, in which our dataset would be divided. This number is not chosen randomly, but should correspond to the number of weakly connected components of the graph.

As mentioned before the eigenvalues of the Laplacian matrix are indicators of the connectivity of the components, so the smaller the eigenvalue, the more weakly connected is the corresponding subgraph to the others. So, we needed to decide which eigenvalues are small enough to identify a cluster. Therefore we set a relative tolerance, $relTol$, and measured the difference between an eigenvalue λ_i and its immediate predecessor λ_{i-1}. If $\lambda_i - \lambda_{i-1} \leq (relTol)\lambda_{i-1}$ then λ_i identifies a cluster, otherwise the last eigenvalue identifying a cluster is λ_{i-1}. This method of selection works only if $\lambda_{i-1} \neq 0$, so when $\lambda_{i-1} = 0$ we chose, analyzing all the plot of all the eigenvalues, a threshold to decide whether λ_i identifies a cluster or not based on its magnitude. M will then be defined as the number of eigenvalues λ identifying a cluster.

Having obtained the M eigenpairs, we obtained $U_1, ..., U_M$ eigenvectors of size n, being n the number of rows in the Laplacian matrix L, which is equal to the number of data points in a dataset. So we construct the matrix U, whose columns are the previously mentioned eigenvectors. For these reasons, $U \in \mathbb{R}^{n \times m}$. Each row of U defines a new set of coordinates with respect to the basis of the eigenvectors: they are a new reference system. Its axes are mutually orthogonal, having derived the eigenvectors from a symmetric positive matrix, in this case L. Thereby we applied a clustering algorithm to assign each data point in the new reference system to one of the M clusters. We ran the k-means algorithm to the rows of U, assigning each row to one of the M clusters. We then mapped this clustering, assigning each point in the original dataset to the same cluster as its corresponding row of U. We can finally visualize the dataset by coloring all points within the same cluster using a consistent color. The results of the spectral clustering are displayed as the output of the Graphical User Interface (GUI) of the MATLAB application we have designed. This also holds for the other clustering methods.

E. Computation of the symmetric normalized version of the Laplacian matrix

The second optional of the homework was to rerun all the codes using the matrix

$$Lsym = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

instead of L. This matrix is the symmetric normalization of the Laplacian matrix L and is used to make the contribution of the nodes independent of their grade. The matrix $Lsym$ does not share the same properties as L: specifically, the sum of the elements in each row of $Lsym$ is no longer zero. As a result, we cannot assume that the eigenvector, x_n, corresponding to the smallest eigenvalue is equal to 1_n. However, since $Lsym$ is symmetric, all eigenvalues are real, and the smallest eigenvalue remains $\lambda_n = 0$. To apply the inverse power method to this matrix, we can no longer shift the smallest eigenvalue, $\lambda_n = 0$, because the associated eigenvector is unknown. Therefore, we must apply the inverse power method directly to the original matrix, despite it being singular. Fortunately, MATLAB can handle the inversion of symmetric matrices by introducing a small shift, which approximates the result as accurately as possible.

Using the normalized version of the matrix results in faster and more efficient computations. This is because normalization adjusts the scale of the matrix, making the eigenvalues and eigenvectors more stable and easier to compute numerically.

III. Application of other clustering methods, result evaluation and comparison

A. DBSCAN and K-means

In order to compare the results of our spectral clustering algorithm, we needed to clusterize the points with other techniques. The choice of DBSCAN was derived from our observation of the datasets: DBSCAN is a density-based algorithm resistant to noise that can most importantly handle clusters of different shapes and sizes, following arbitrarily-shaped clusters. It can find a cluster completely surrounded by a different cluster, and these properties seemed to be suited for our datasets: the points in the Spiral dataset follow the shapes of three parallel concentric spirals, and some of the points in the Circle dataset form a circle within another circle. So we are expecting it to work perfectly on the Spiral dataset and for it to have a good result on the Circle as well. Briefly, DBSCAN assigns a label between Noise, Border, and Core to each point, based on the number of points withing a certain range to it. The number of points is $minPoints$ and the range is ϵ. The choice of these numbers is crucial for the outcome of the clustering, so we chose them selecting the values that would give us the lowest Adjusted Rand Index for the Spiral, the highest Average Silhouette Score for the Iris dataset, and arbitrarily for the Circle. The choice of the evaluation method for each dataset is explained in the following corresponding subsections. The values we selected are shown in Table I:

Dataset	ϵ	$minPoints$
Circle	0.8	3
Spiral	1.2	3
Iris	0.8	3

TABLE I
DBSCAN Parameters for each Dataset

We also chose K-means so that we could have a completely different algorithm than DBSCAN: K-means works well when the data is divided in globular shapes and the clusters are about the same size. It is adequate for a dataset made of "blobs" like the Iris dataset or a part of the Circle dataset, but its limits are evident when it is implemented on datasets such as the Spiral one, which has its data following geometrical shapes. It has a partitional clustering approach, as it uses the randomly initialized centroids as centers of their corresponding Voronoi regions. All points are then assigned to the region of the closest centroid, and the coordinates of the new centroids are recomputed as the mean of the coordinates of the points assigned to its Voronoi region. This process is repeated until the centroids don't change. Finally, the points in each Voronoi region form a cluster. We iterated over possible values of K (the number of centroids) to find, for each dataset, the optimal K, based on the metric used for evaluation mentioned before. Table II shows the chosen values:

Dataset	K
Circle	2
Spiral	3
Iris	2

TABLE II
OPTIMAL NUMBER OF CENTROIDS IN K-MEANS

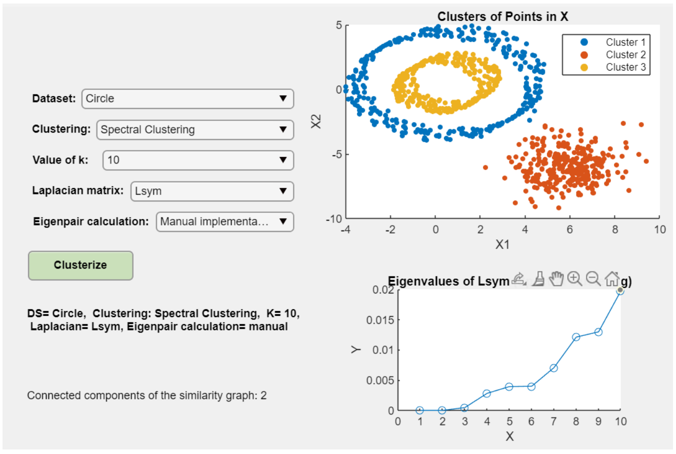
It must be noted that the optimal value of K in terms of Adjusted Rand Index (ARI) for the Spiral dataset is 2, having ARI=0, but showing the results for $K = 3$ is more significant when comparing the Spectral Clustering algorithm to K-means. For $K = 3$, ARI = -0.01, so there is not much difference from the two ARI values.

All results and values shown in the following Subsections III-B, III-C, can be replicated using the MATLAB application we created.

B. Circle dataset

The Circle dataset is shown in Figure 1 as a plot on the two dimensions it is composed by. Its points represent two concentric "rings" and a separated "blob" shape. Due to the fact that we do not have the ground truths on the cluster assignments of its points, we could not use the ARI to evaluate our clustering. The average silhouette score would not have been effective: if the two rings were classified into separate clusters, because one is nested within the other, any point in either cluster would exhibit a low silhouette score. The silhouette score favors points within clusters with low intra-cluster distances and high inter-cluster distances. In our situation, computing the silhouette score for a point in the outer ring would highlight that it is closer to points in the inner ring than it is to points in the distant parts of the outer ring. So what we would evaluate as a correct clustering would produce a low silhouette score. Ultimately, because of the characteristics of the datasets, we decided to do an external evaluation of the results.

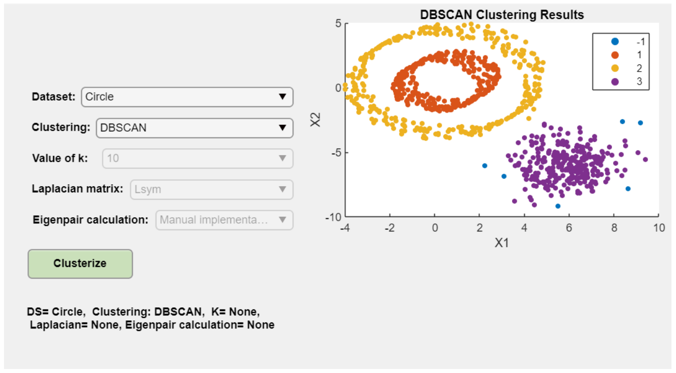
Any kind of Spectral clustering with the value of k of the k-neighbourhood similarity graph set to 10 results in a division of the dataset in three separate clusters. This is shown in Figure 9. The relatively low value of K ensures that the points of the concentric rings are almost only neighbors of other points in the same ring. For this reason, the graph has a higher number of connected components (in this case two), and the third smallest eigenvalue is small enough to consider a third cluster. The three clusters are the two concentric rings and the separated cloud of points visible on the bottom right of the two-dimensional plane. A similar division of the dataset is obtained when using the DBSCAN clustering algorithm, with the difference that this method identifies some marginal points of the cloud as noise points. In order to solve this, we tried to lower the number of $minPoints$ to 2, or to lower the value of ϵ but that turned out to be useless, as the resulting clusters were inadequate. We expected both DBSCAN and spectral clustering to perform well on this dataset, and this expectation was confirmed. The results of the clustering with the correct DBSCAN settings ($minPoints = 3$, $\epsilon = 0.8$) are displayed in Figure 10.


Dataset: Circle
Clustering: Spectral Clustering
Value of k: 10
Laplacian matrix: Lsym
Eigenpair calculation: Manual implementa...

DS= Circle, Clustering: Spectral Clustering, K= 10, Laplacian= Lsym, Eigenpair calculation= manual

Connected components of the similarity graph: 2

Fig. 9. Results of Spectral Clustering for the Circle dataset with $k = 10$


Dataset: Circle
Clustering: DBSCAN
Value of k: 10
Laplacian matrix: Lsym
Eigenpair calculation: Manual implementa...

DS= Circle, Clustering: DBSCAN, K= None, Laplacian= None, Eigenpair calculation= None

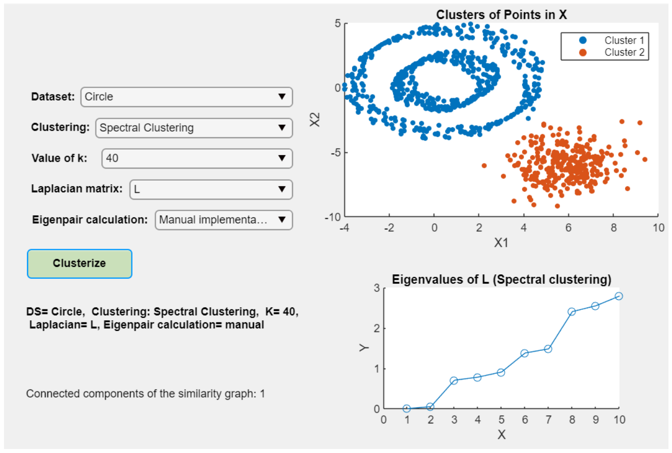
Fig. 10. Results of DBSCAN for the Circle dataset

If the value of k is set to 20 or 40, instead, only two clusters are identified by our spectral clustering algorithm. This occurs for any configuration of Laplacian matrix or eigenpair calculation. The two concentric rings are assigned to the same cluster, while the portion of points on the bottom right is identified as a different cluster. The analysis of the eigenvalues confirms this division. Let's take as example the following configuration:

- $k = 40$,
- Laplacian $= L$
- Eigenpair calculation $=$ Manual implementation

With this setting, the values of the first two eigenvalues are 0 and 0.0482. These are sufficiently small to consider the existence of two clusters. The value of the third, instead, is

0.703. Its magnitude and its difference from the other two indicate that the right choice for the number of clusters is two. The huge discrepancy between the second and the third eigenvalue can also be observed from the graph on the bottom right of Figure 11.



Dataset: Circle

Clustering: Spectral Clustering

Value of k: 40

Laplacian matrix: L

Eigenpair calculation: Manual implementa...

DS= Circle, Clustering: Spectral Clustering, K= 40,
Laplacian= L, Eigenpair calculation= manual

Connected components of the similarity graph: 1

Fig. 11. Results of Spectral clustering for the Circle dataset with $k = 40$

Applying the K-means algorithm to this dataset with $K = 2$ yields a result that looks very much the same, differing only in two points that get placed in the other cluster. The usage of $K = 2$ was imposed by the limitations of the K-means algorithm. If $K = 3$ were used, the "blob" in the bottom-right corner would be correctly identified but the two rings would be incorrectly split.

C. Spiral dataset

The Spiral dataset is two-dimensional, and its visual representation is shown in Figure 2. As can be seen, it consists of three concentric parallel spirals, the data points follow definite geometrical figures. For this reason we are expecting Spectral Clustering to be very effective: the clusters are well-separated in the feature space, and their shape is complex and non-linear. This expectation held for DBSCAN as well.

For this dataset, we were also provided with the ground truth assignments to the correct clusters. The vector containing the right clustering was part of the dataset and we extracted it using array slicing methods. Having the ground truths, we decided to use the Adjusted Rand Index (ARI) [1] to evaluate the quality of the clustering methods that we applied. The Rand Index (RI) is an evaluation metric that assesses clustering performance by comparing the ground truth vector with our clustering vector. Here's how it's calculated: the RI evaluates the quality of a clustering by checking all pairs of data points, determining if each pair is correctly identified as belonging to the same or different clusters, based on the ground truth. It is the ratio of correct agreements (same or different) to the total number of pairs. Its adjusted form, the Adjusted Rand Index (ARI), corrects the RI for chance, considering that random clustering may still result correct. It does this by normalizing the RI against the expected similarity for random labels. Table III displays the value of the ARI for each kind of clustering method.

Clustering	k	Laplacian	Eigenpair calculation	ARI
Spectral	Any	Any	Any	1
DBSCAN	–	–	–	1
K-means	–	–	–	-0.01

TABLE III
VALUES OF ADJUSTED RAND INDEX FOR EACH CLUSTERING CONFIGURATION

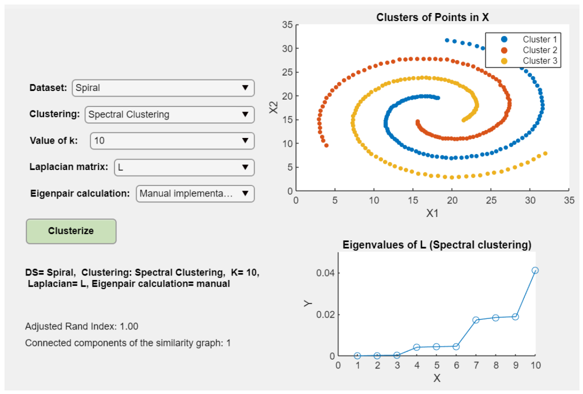
The "Any" that appears in the table means that for every kind of configuration of the spectral clustering method, the ARI does not change. For clarity, here are all possible values:

- $k = 10, 20, 40,$
- Laplacian $= L, Lsym$
- Eigenpair calculation $=$ MATLAB default, Manual implementation

For this dataset, we decided to show two GUI outputs: one displaying a correct clustering with ARI $= 1$, and one with ARI $= -0.01$. Figure 12 shows the result of the implementation of a spectral clustering. As the ARI value suggests, the clustering works correctly, as all of the points are assigned to the right cluster. All possible configurations of the spectral clustering obtain the same result, even though the eigenvalues vary depending on the chosen k, the Laplacian matrix, and the method of calculation of the eigenpairs. As expected, this technique works well on this dataset because of its structure. The results shown are obtained with this setup:

- $k = 10$
- Laplacian $= L$
- Eigenpair calculation $=$ Manual implementation

Manual implementation means that the inverse power method and the deflation are both in use for the calculation of the eigenvalues.



Dataset: Spiral

Clustering: Spectral Clustering

Value of k: 10

Laplacian matrix: L

Eigenpair calculation: Manual implementa...

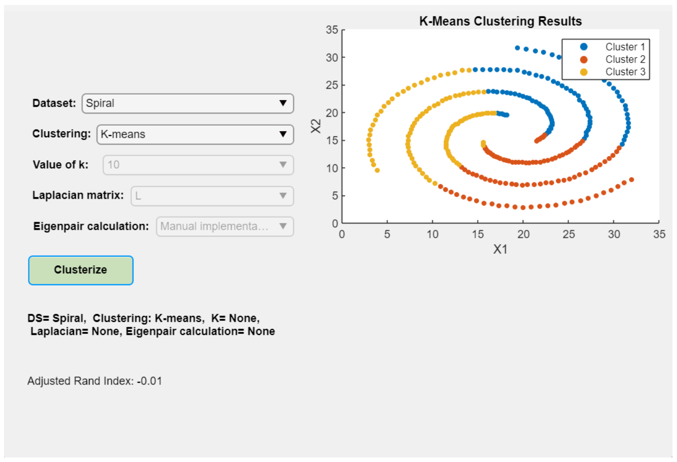
DS= Spiral, Clustering: Spectral Clustering, K= 10,
Laplacian= L, Eigenpair calculation= manual

Adjusted Rand Index: 1.00

Connected components of the similarity graph: 1

Fig. 12. Results of Spectral clustering for the Spiral dataset

Figure 13, instead, displays the output of the K-means clustering on the Spiral dataset, and its results in terms of ARI are low. A negative ARI means that the clustering obtains worse results than a random assignment of the points to the clusters would. For the K-means algorithm shown in the picture, the value of K is three, so the number of clusters created is three. As expected, the algorithm divides the plane in three sections, and the spiral structure of the dataset is completely neglected. For this reason, the ARI value is equal to -0.01.


Dataset: Spiral
Clustering: K-means
Value of k: 10
Laplacian matrix: L
Eigenpair calculation: Manual implementa...
Clusterize

DS= Spiral, Clustering: K-means, K= None,
Laplacian= None, Eigenpair calculation= None

Adjusted Rand Index: -0.01

Fig. 13. Results of K-means clustering for the Spiral dataset. K=3

D. Iris dataset

The Iris dataset is three-dimensional, and its visual representation is shown in Figure 3. This dataset is derived by the application of Principal Components Analysis on the toy $150x4$ dataset with the same name. The initial dataset also provides the label of each flower, and we used this as ground truth for the calculation of the ARI. As can be seen in Figure 3, the data points have an interesting distribution pattern. They form two distinct clusters, one clearly separated and located far from the others. All clusters exhibited a linear-like shape, which suggested that the underlying structure of the data was well suited for clustering algorithms to perform effectively.

This initial observation led us to anticipate that the clustering algorithms we implemented would provide good results. Our hypothesis was confirmed, as demonstrated in Table IV.

Clustering	k	Laplacian	Eigs. calculation	Avg. S.S.
Spectral	10	Lsym	Any	0.74146
Spectral	20	Lsym	Any	0.72985
Spectral	Any	Any	Any	0.84911
DBSCAN	–	–	–	0.84911
K-means	–	–	–	0.85296

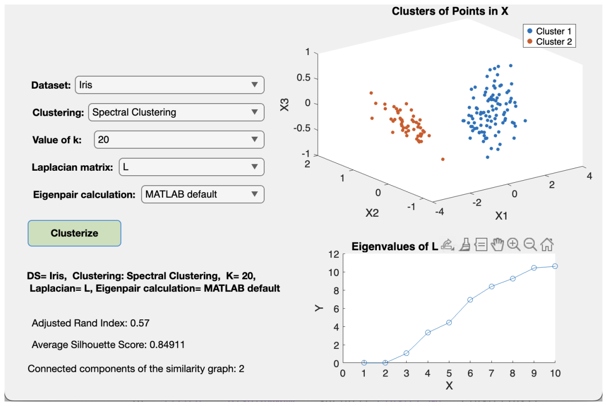
TABLE IV
VALUES OF AVERAGE SILHOUETTE SCORE

The average silhouette score derived from the implementation of all the algorithms was notably high and consistent across the board, with the exception of using $Lsym$ with $k = 10, 20$, which gave a slightly lower score. The high and uniform silhouette scores suggest that there is no single algorithm that significantly outperformed the others. Instead, all algorithms demonstrated similar performance, making them equally suitable for this specific dataset and clustering task.

As mentioned before, we had the ground truth cluster assignments. This allowed us to use these labels to evaluate the performance of our clustering algorithms, similarly to what we did with the Spiral dataset. The first observation is that if we use L as the basis for spectral clustering, all the methods cluster the data into only two clusters. There is one cluster that is significantly isolated, much farther away from the other clusters. Its distance from the nearest cluster is substantially greater than the typical distances between any of the other clusters: even if more were to be formed, none would be as distant from the main group as this isolated cluster.

Let's see this configuration as an example:

- $K = 20$,
- Laplacian $= L$
- Eigenpair calculation $=$ MATLAB default


Dataset: Iris
Clustering: Spectral Clustering
Value of k: 20
Laplacian matrix: L
Eigenpair calculation: MATLAB default
Clusterize

DS= Iris, Clustering: Spectral Clustering, K= 20,
Laplacian= L, Eigenpair calculation= MATLAB default

Adjusted Rand Index: 0.57

Average Silhouette Score: 0.84911

Connected components of the similarity graph: 2

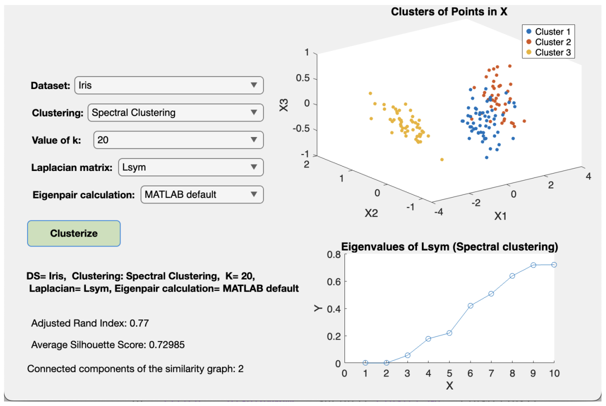
Fig. 14. Results of Spectral clustering for the Iris dataset using matrix L and k = 20

In the context of spectral clustering, we notice that the first two eigenvalues are exactly 0, meaning that they define two distinct connected components inside the graph. On the other hand, the third smallest eigenvalue λ_{n-2} is greater than a threshold we set $\lambda_{n-2} > 0.085$, which indicates a weaker connection among the remaining clusters and prevents further subdivision when using L.

This behavior changes when we use $Lsym$: in this case, the algorithm divides the dataset into three clusters, for two different values of k, such as $k = 10$ and $k = 20$.

The result shown below follows this configuration:

- $k = 20$,
- Laplacian $= Lsym$
- Eigenpair calculation $=$ MATLAB default


Dataset: Iris

Clustering: Spectral Clustering

Value of k: 20

Laplacian matrix: Lsym

Eigenpair calculation: MATLAB default

DS= Iris, Clustering: Spectral Clustering, K= 20, Laplacian= Lsym, Eigenpair calculation= MATLAB default

Adjusted Rand Index: 0.77

Average Silhouette Score: 0.72985

Connected components of the similarity graph: 2

Fig. 15. Results of Spectral clustering for the Iris dataset using matrix Lsym and k = 20

For these configurations, the average Adjusted Rand Index is approximately 0.755, while for all other possible combinations of algorithms and k the average ARI was approximately 0.57. As shown in Table IV, the worst ARI is obtained using the K-Means algorithm, with a value of $ARI = 0.54$. This result is unexpected, especially considering that K-Means achieved the highest average Silhouette Score. This discrepancy can be attributed to the incorrect choice of the number of clusters, K, for the K-means algorithm. Although the true number of clusters in the dataset is 3, we opted to use $K = 2$ for K-means to optimize the Silhouette Score of the algorithm. If we chose $K = 3$ for the K-means algorithm, its Silhouette Score would have decreased to 0.742, the lowest among all methods. However, the ARI would increase up to 0.73. This happens because with $K = 3$, the intra-cluster distances decrease due to the proximity of clusters 1 and 2 compared to the distance between clusters 1 and 3. Despite this improvement, the K-means algorithm couldn't reach the performance of spectral clustering using $Lsym$. The table below summarizes the ARI values for different clustering methods. For K-means, the calculation is based on using only 2 clusters:

Clustering	K	Laplacian	Eigs. calculation	API
Spectral	10	Lsym	Any	0.74
Spectral	20	Lsym	Any	0.77
K-means	–	–	–	0.54
Any	Any	Any	Any	0.57

TABLE V
VALUES OF ADJUSTED RAND INDEX FOR EACH CLUSTERING CONFIGURATION

The results suggest that leveraging $Lsym$ with spectral clustering is particularly advantageous for datasets like Iris, where the natural clusters have varying degrees of separation.

REFERENCES

[1] Chris McComb. (2025). Adjusted Rand Index. Retrieved from https://github.com/cmccomb/rand_index.