Computational Linear Algebra

Spectral Clustering

Aljosevic Ismail s337769 Amato Daniele s334211

Academic Year 2024–2025

Introduction

This report presents the implementation of Spectral Clustering for identifying cluster structures in data. The method was applied to both provided datasets (Circle and Spiral) and a custom generated 3D dataset. All steps were implemented in Python, including custom implementations of the Inverse Power Method and the Deflation Method for computing the smallest eigenvalues of the Laplacian matrix. The results were compared to classical clustering methods and visually analyzed to evaluate clustering performance.

Datasets

The data used in this project consists of two datasets:

- **Circle** — contains 900 data points represented by two columns corresponding to the x and y coordinates.
- **Spiral** — contains 312 data points represented by three columns. The first two columns correspond to the x and y coordinates, while the third column contains the index of correct cluster.

The plots of both datasets are presented in Figure 1.

Exercise 1

In the first exercise, the task was to construct the k-nearest neighbors similarity graph and its corresponding adjacency matrix W, for $k = 10$, 20, 40, and a given similarity function.

A similarity graph is a way of representing similarity between data points. It is formally defined as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node represents a data point, and an edge between two

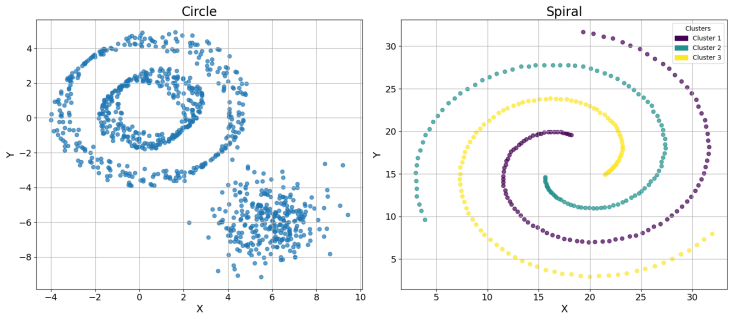

Figure 1: Circle and Spiral datasets

nodes exists if the similarity between the corresponding data points is positive or exceeds a certain threshold. The weight of an edge (i, j) is given by S_{ij}, the similarity score between points i and j.

Before constructing the similarity graph, an appropriate similarity function must be defined. This function should ensure that local neighborhoods reflect the true structure of the data. For data points in the Euclidean space R^n, a commonly used similarity function is the Gaussian kernel, which was also specified in the homework instructions:

$$S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right).$$

According to the exercise requirements, the k-nearest neighbors graph had to be constructed. This graph type is widely recommended as a default choice for building similarity graphs due to its simplicity and effectiveness.

Steps for Constructing the Similarity Graph

1. Define the similarity matrix S, where each entry S_{ij} contains the similarity score between data points i and j.

2. Construct the sparse adjacency matrix W from S by retaining, for each data point, the edges (similarity values) to only its k nearest neighbors.

For this purpose, we used the `create_similarity_matrix` function to compute similarity values between all pairs of data points. The function calculates the squared Euclidean distance between each pair and applies the Gaussian kernel. To prevent self-connections, all diagonal elements of the similarity matrix were set to zero.

To construct the adjacency matrix, we used the `construct_knn_adjacency_matrix` function. For each data point, the function selects the top k most

similar neighbors based on the similarity scores from the corresponding row of matrix S. To preserve symmetry and obtain an undirected graph, we ignored the direction of the edges and connected v_i and v_j if either v_i is among the k-nearest neighbors of v_j, or v_j is among the k-nearest neighbors of v_i. Since the resulting adjacency matrices are sparse by nature, along with Degree and Laplacian matrices used later, we utilized efficient sparse matrix representations from the `scipy.sparse` module. This approach significantly reduces memory usage and improves performance when working with large datasets.

Exercise 2

In the second exercise, the task was to construct the degree matrix D and the Laplacian matrix L, which are fundamental aspects in the spectral clustering pipeline.

The degree matrix D is a diagonal matrix in which each diagonal element D_{ii} represents the degree of node i. The degree is defined as the sum of the weights of all edges connected to node i, and it is computed from the adjacency matrix W as follows:

$$D_{ii} = \sum_j W_{ij}.$$

The graph Laplacian L is a matrix that captures the structure of a graph and is defined based on the degree matrix D and the adjacency matrix W. The most commonly used form is the unnormalized Laplacian, given by:

$$L = D - W.$$

To compute the degree matrix, we implemented the function `construct_degree_matrix`, which calculates the degree of each node by summing the entries in each row of the adjacency matrix. We then developed the `construct_laplacian_matrix` function, which constructs either the unnormalized or the normalized symmetric Laplacian, depending on the value of the parameter `type_`.

The normalized symmetric Laplacian is discussed in more detail in the section on optional exercises.

Exercise 3

In the third exercise, the task was to compute the number of connected components in the similarity graph.

To determine the number of connected components in the graph, a key property of the Laplacian matrix is used: the multiplicity of the eigenvalue 0 in the

spectrum of L is equal to the number of connected components.

Practically, connected components represent distinct clusters in the data. Every graph Laplacian matrix has at least one eigenvalue equal to zero, corresponding to the constant eigenvector. This property allows spectral clustering to identify disjoint groups in the data by analyzing the eigenstructure of the Laplacian.

To compute the number of connected components, we implemented the `num_of_connected_components` function, which estimates the number of zero eigenvalues of the Laplacian matrix using the `eigsh` function from `scipy.sparse.linalg` module. The number of connected components corresponds to the number of eigenvalues close to zero, within a predefined numerical tolerance. In our implementation, this tolerance was set to 10^{-9}.

To get a better understanding of the structural properties and sparsity patterns of the matrices, we visualized the Laplacian matrices using spy plots for both datasets.

3.1 Circle Dataset

For the Circle dataset, we obtained the following number of connected components:

- $k = 10$: 2 connected components
- $k = 20$: 1 connected component
- $k = 40$: 1 connected component .

By examining the spy plots shown in Figure 2, we can observe a strong square pattern in the bottom-right corner, indicating that one group of nodes is very well connected and thus represents a separate cluster. As k increases, more off-diagonal elements appear in the matrix. This suggests that nodes begin to connect with more distant neighbors, which can reduce the clarity of the original cluster structure and eventually lead to a fully connected graph, as observed for $k = 20$ and $k = 40$. Nevertheless, we can still recognize a block structure along the diagonal, that reflects the presence of underlying clusters.

3.2 Spiral Dataset

For the Spiral dataset, we obtained the following number of connected components:

- $k = 10$: 1 connected component
- $k = 20$: 1 connected component
- $k = 40$: 1 connected component .

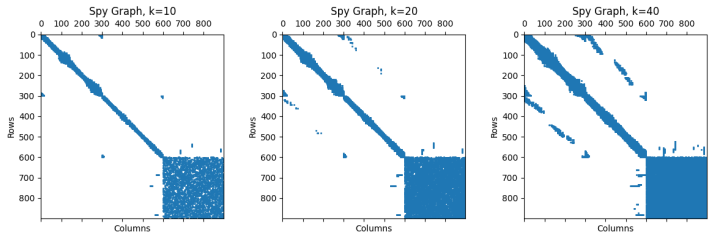

Figure 2: Sparsity pattern of the Laplacian matrix for the Circle dataset.

By examining the spy plots in Figure 3, we observe a block-like structure along the diagonal, which suggests the presence of distinct clusters, even though the results indicate that the graph is fully connected for every value of k. As the parameter k increases, more off-diagonal elements appear, resembling the behavior observed in the Circle dataset.

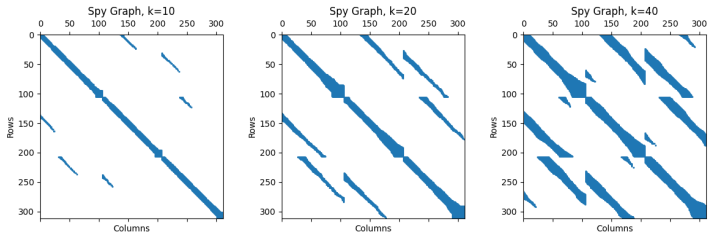

Figure 3: Sparsity pattern of the Laplacian matrix for the Spiral dataset.

Exercise 4

In the fourth exercise, the task was to compute a few of the smallest eigenvalues of the Laplacian matrix and use their values to determine a suitable number of clusters for the given datasets.

In practice, clusters are not entirely isolated. They are typically weakly connected to other groups through sparse or low-weight edges. These weak connections are reflected in small but non-zero eigenvalues of the Laplacian matrix, indicating that the graph is nearly, but not perfectly, separable into disconnected components.

To compute the m smallest eigenpairs (eigenvalues and corresponding eigenvectors), we used our implementations of the Inverse Power Method and the Deflation Method, setting $m = 5$. The implementation details of these methods are discussed in the section on optional exercises.

To determine a suitable number of clusters, we applied the eigengap heuristic, which identifies the largest proportional increase between consecutive eigenvalues in the spectrum. This approach takes into account both the absolute magnitude of the eigenvalues and their relative differences. Based on the plots of the sorted eigenvalues, the number of small eigenvalues preceding this largest proportional increase serves as an estimate of the optimal number of clusters.

4.1 Circle Dataset

For the **Circle** dataset, we obtained the following results based on the plots of the smallest eigenvalues from Figure 4:

1. $k = 10$**:** A noticeable eigengap is observed between the third and fourth eigenvalues, suggesting the presence of three clusters.
   Chosen number of clusters: $M = 3$.

2. $k = 20$**:** The first noticeable eigengap appears between the second and third eigenvalues, suggesting the presence of two main clusters. However, the gap between the third and fourth eigenvalues is even larger. Considering both the relative jumps and the magnitudes of the eigenvalues, this implies that selecting three clusters may be a more appropriate choice.
   Chosen number of clusters: $M = 3$.

3. $k = 40$**:** A significant gap is observed between the second and third eigenvalues, suggesting the presence of two clusters.
   Chosen number of clusters: $M = 2$.

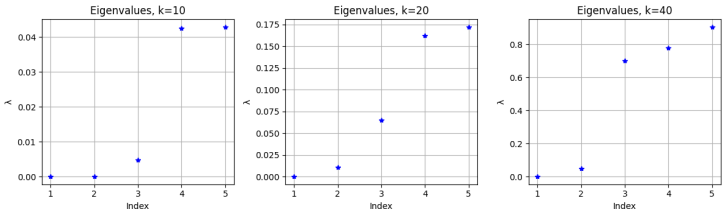

Figure 4: Five smallest eigenvalues for the Circle dataset.

4.2 Spiral Dataset

For the **Spiral** dataset, we obtained the following results based on the plots of the smallest eigenvalues from Figure 5:

1. $k = 10$**:** A clear eigengap appears between the third and fourth eigenvalues, suggesting the presence of three clusters.
   Chosen number of clusters: $M = 3$.

2. $k = 20$**:** The second and third eigenvalues are nearly equal, followed by a noticeable gap after the third, suggesting three clusters.
   Chosen number of clusters: $M = 3$.

3. $k = 40$**:** A similar pattern to the one observed for $k = 20$ is present, with a gap after the third eigenvalue, again suggesting the presence of three clusters.
   Chosen number of clusters: $M = 3$.

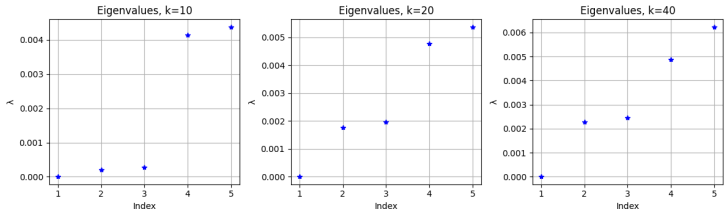

Figure 5: Five smallest eigenvalues for the Spiral dataset.

Exercise 5

In the fifth exercise, the task was to compute the M eigenvectors corresponding to the M smallest eigenvalues of the Laplacian matrix. These eigenvectors are then used to construct the matrix U, where each column corresponds to one of the selected eigenvectors.

The matrix U represents the spectral embedding of the data — a transformed space where data points are more easily separable based on the graph structure. This embedding enhances cluster separability and serves as the input for the traditional clustering algorithms, such as K-means.

In the previous exercise, we computed and stored the first $m = 5$ eigenpairs for each Laplacian matrix. To avoid redundant computation, we simply extracted the required number of eigenvectors from the previously saved results for each dataset and value of k, and used them to construct the matrix U.

Exercise 6

In the sixth exercise, the task was to perform clustering using the K-means algorithm on the spectral embedding obtained in the previous step. The number of clusters was set equal to the number of selected eigenvalues. For this purpose, we used the implementation of the K-means algorithm provided by the `scikit-learn` library with default settings and a fixed random seed.

Exercise 7

In the seventh exercise, the task was to assign each original data point to a cluster based on the clustering results obtained in the spectral space. After applying the K-means algorithm to the spectral embedding, each data point was assigned a cluster label. These labels were then mapped back to the original data space, completing the spectral clustering pipeline.

Exercise 8

In the eighth exercise, the task was to visualize the clustering results by plotting the original data points, colored according to their assigned cluster labels, in order to evaluate the outcome.

8.1 Circle Dataset

For the **Circle** dataset, we obtained the clustering results shown in Figure 6:

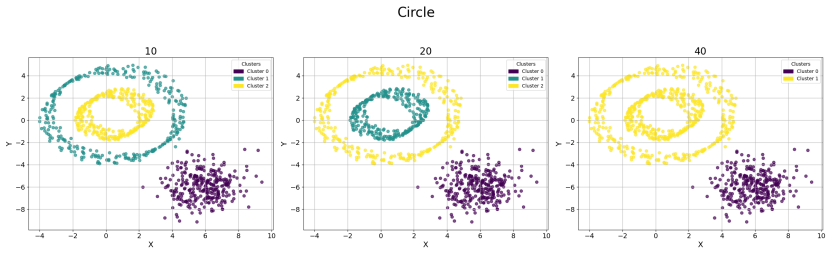

Figure 6: Clusters obtained for k = 10, 20, 40 and M = 3, 3, 2 respectively

The cluster colored in purple corresponds to the square-shaped structure located in the bottom-right corner of the spy plots of the Laplacian matrices, observed earlier.

For $k = 40$, we observe that the circular clusters tend to merge due to increased connectivity between more distant neighbors.

To demonstrate the influence of eigenvalue magnitudes, we also applied K-means on the spectral embedding obtained for $k = 40$ and $M = 3$, as shown in Figure 7. The resulting clustering is clearly not good, primarily because the third eigenvalue is relatively large. This reduces the separation between the embedded points and negatively impacts the quality of the spectral embedding.

Overall, the clustering result for $k = 10$ provides the clearest and most interpretable separation, based on the eigenvalue analysis and the fact that it relies on a smaller number of nearest neighbors.

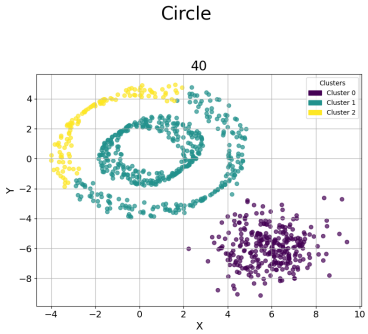

Figure 7: Clusters obtained for k = 40 and M = 3

8.1 Spiral Dataset

For the **Spiral** dataset, we obtained the clustering results shown in Figure 8:

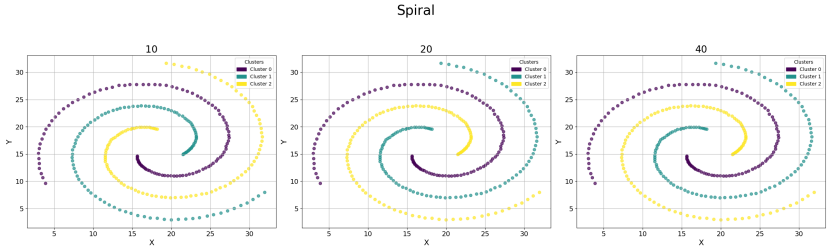

Figure 8: Clusters obtained for k = 10, 20, 40 and M = 3, 3, 3 respectively

This consistency suggests that the spiral structure naturally supports a three-cluster interpretation, regardless of the sparsity level ruled by values of k. This result was expected, given the clear block structure observed earlier in the spy plots of the Spiral Laplacian matrices.

Overall, for the same reasons as in the Circle dataset, the clustering result for $k = 10$ is also the most reliable.

Exercise 9

In the ninth exercise, the task was to compare the results of spectral clustering with those obtained using traditional clustering algorithms. Specifically, we applied the standard K-means algorithm to the original data space using $M = 2, 3$.

9.1 Circle Dataset

For the **Circle** dataset, we obtained the clustering results shown in Figure 9:

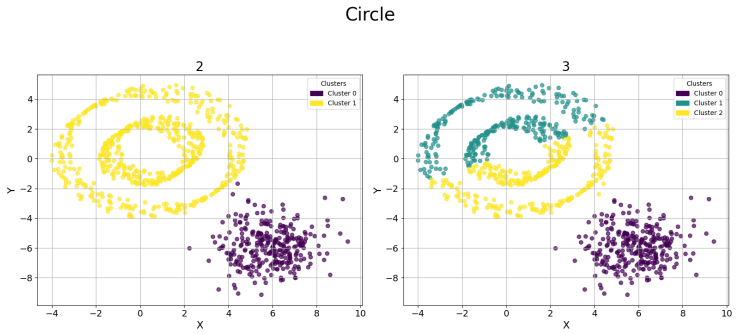

Figure 9: Clusters obtained for M = 2, 3

- For $M = 2$: The K-means clustering produced results that are nearly identical to those obtained from spectral clustering with $k = 40$. The only noticeable difference appears in a few boundary points that were assigned to different clusters.
- For $M = 3$: The K-means clustering results differ significantly from those of spectral clustering. While the purple cluster was correctly identified, the remaining two clusters were separated in a way that does not reflect the true structure of the data.

These results are consistent with the known limitations of the K-means algorithm. It performs well when clusters are roughly spherical and well-separated, but struggles when one cluster surrounds another. This is evident in the distorted shape of the yellow cluster for $M = 2$, and the less meaningful partitioning observed for $M = 3$.

9.2 Spiral Dataset

For the **Spiral** dataset, we obtained the clustering results shown in Figure 10:

- For $M = 2, 3$: The K-means clustering produced results primarily based on point proximity, which leads to an incorrect clustering that does not capture the true structure of the data.

The same conclusion applies as with the Circle dataset — K-means is ineffective when the clusters are not linearly or spherically separable. Its reliance on Euclidean distance makes it unsuitable for datasets with complex, non-linear structures.

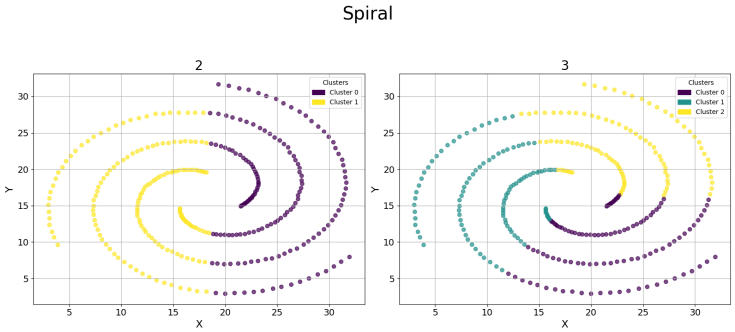

Figure 10: Clusters obtained for M = 2, 3 with K-means

Optional Exercises

Exercise 10

In this exercise, the task was to evaluate the applicability of spectral clustering to higher-dimensional datasets. For this purpose, we generated a custom 3D dataset consisting of three circles in 3D space.

Each circle was generated using the parametric equations of a circle in two dimensions:

$$x(t) = x_0 + r\cos(t), \quad y(t) = y_0 + r\sin(t)$$

To extend the data to three dimensions, one coordinate (e.g., $z = z_0, x = x_0$) was fixed, and each circle was placed in a different plane by varying this coordinate. A small amount of Gaussian noise was added to the generated points to better simulate real-world data. The dataset consists of 450 points and visualization of the dataset is shown in Figure 11

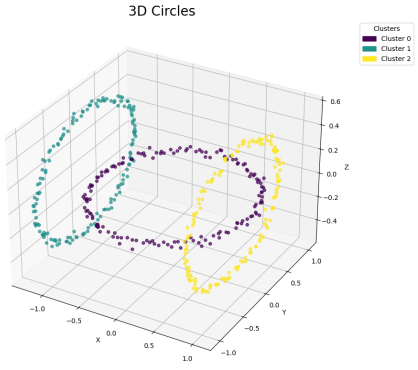

Figure 11: 3D Circles Dataset

The same pipeline used for the Circle and Spiral datasets was applied to the 3D Circle dataset. Specifically, a k-nearest neighbors similarity graph and the corresponding adjacency matrix were constructed for $k = 10, 20, 40$. Furthermore, the Degree matrix and the corresponding Laplacian matrix were computed to determine the number of connected components and computation of m smallest eigenpairs.

We obtained the following number of connected components:

- $k = 10$: 3 connected components
- $k = 20$: 2 connected components
- $k = 40$: 1 connected component .

.
By examining the spy plots in Figure 12, we observe that the matrices exhibit a strong diagonal structure for the 3D Circles dataset as well. Additionally, as k increases, more non-zero values appear away from the diagonal, and the graph becomes fully connected for $k = 40$.

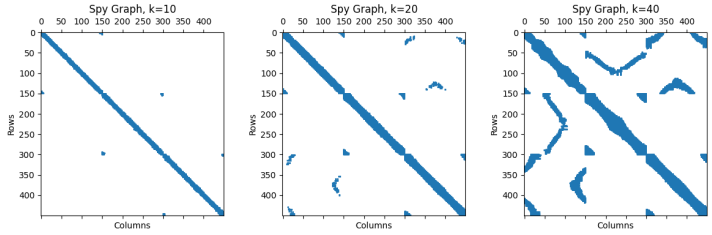

Figure 12: Sparsity pattern of the Laplacian matrices for the 3D Circles dataset

Regarding the optimal number of clusters, we obtained the following results based on the plots of the smallest eigenvalues shown in Figure 13:

1. $k = 10$**:** A noticeable eigengap is observed between the third and fourth eigenvalues, suggesting the presence of three clusters, corresponding to the number of zero eigenvalues.
   Chosen number of clusters: $M = 3$.

2. $k = 20$**:** A clear eigengap appears between the second and third eigenvalues, suggesting two main clusters, corresponding to the number of zero eigenvalues.
   Chosen number of clusters: $M = 2$.

3. $k = 40$**:** A noticeable gap is present between the first and second eigenvalues, with a significant difference in magnitude. This suggests that $k = 40$ may not be a suitable choice for clustering this dataset. Nevertheless, for

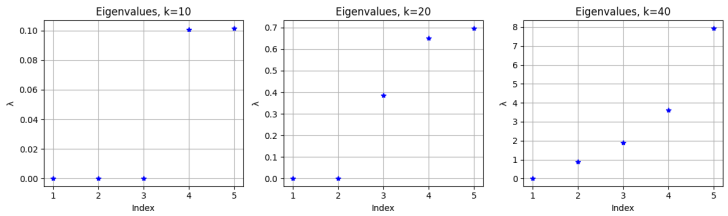

Figure 13: Five smallest eigenvalues for the 3D Circles dataset.

testing purposes, the first two eigenvalues were used.
Chosen number of clusters: $M = 2$.

In 3D Circles dataset, the analysis is more straightforward, as the eigengaps and eigenvalue magnitudes are clearly distinguishable.

For the 3D Circles dataset, we obtained the clustering results shown in Figure 14.

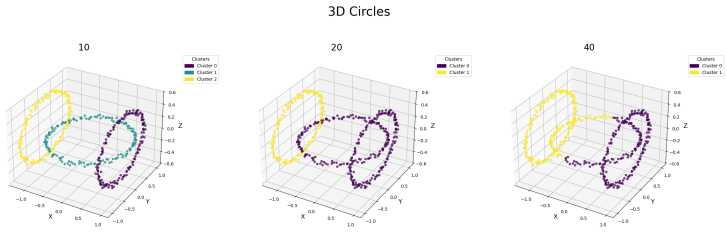

Figure 14: Clusters obtained for k = 10, 20, 40 and M = 3, 2, 2 respectively

- For $k = 10$, all three circles are correctly identified and clearly separated.
- For $k = 20$ and $k = 40$, two clusters are detected in a clearly suboptimal way.

These outcomes were expected based on the previously observed eigenvalue distributions. Therefore, selecting an appropriate value of k is crucial for achieving meaningful clustering results on this dataset.

The limitations of the standard K-means algorithm are evident in this case as well. As shown in Figure 15, the algorithm fails to correctly separate the clusters, due to its known limitation in handling non-linearly separable data.

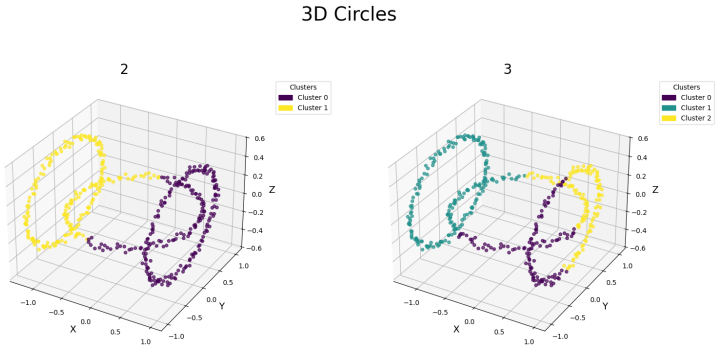

Figure 15: Clusters obtained for M = 2, 3 with K-means

Exercise 11

In this exercise, the task was to apply the same spectral clustering pipeline to the normalized symmetric Laplacian matrix $L_{\text{sym}} \in R^{N \times N}$.

The symmetric Laplacian is defined as:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

and it has several useful properties:

- Ensures that the eigenvectors are orthonormal
- Prevents high-degree nodes from dominating the results.

The method `construct_laplacian_matrix` includes a `type_` parameter. Setting `'symmetric'` enables the computation of the symmetric normalized Laplacian, where the standard Laplacian is first computed and then multiplied on both sides by $D^{-1/2}$.

11.1 Circle Dataset

Spy plots and eigenvalue distributions for the symmetric normalized Laplacian matrices for the Circle dataset are shown in Figure 16 and Figure 17, respectively. The spy plots for both the symmetric normalized and unnormalized Laplacians are identical, indicating that node degrees are relatively uniform. Additionally, the results for the smallest eigenpairs are also consistent. The main difference lies in the scale of the eigenvalues—those from the normalized Laplacian are smaller due to the normalization process.

In this case, the choice of Laplacian does not significantly affect the clustering outcome. Therefore, the same number of clusters was selected as in the unnormalized case: $M = 3, 3, 2$ for $k = 10, 20, 40$, respectively. The corresponding clustering results are shown in Figure 18.

The clustering results obtained by applying K-means to the spectral embeddings computed using L_{sym} are identical to those obtained with the unnormalized Laplacian, which was expected. The only noticeable difference occurs for $k = 40$, where one point from the purple cluster is misclassified and assigned to the yellow cluster.

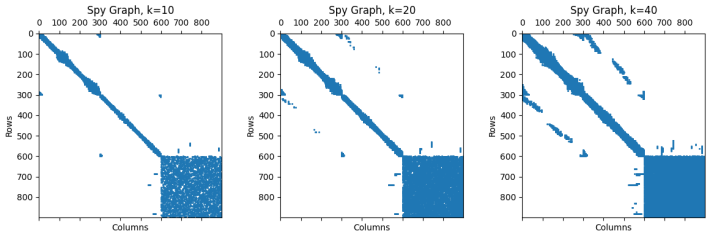

Figure 16: Sparsity pattern of the symmetric normalized Laplacians for the Circle dataset

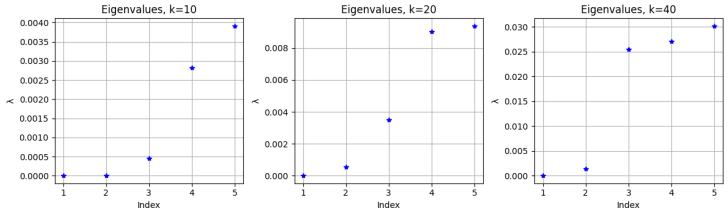

Figure 17: Five smallest symmetric Laplacian eigenvalues for the Circle dataset

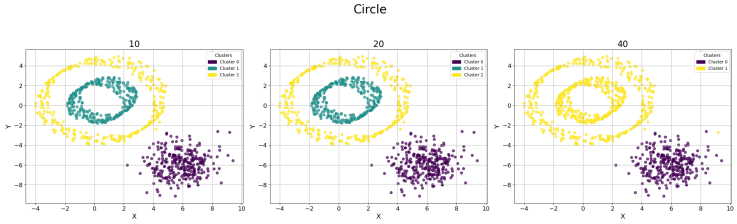

Figure 18: Clusters obtained for k = 10, 20, 40 and M = 3, 3, 2 respectively

11.2 Spiral Dataset

Spy plots and eigenvalue distributions for the symmetric normalized Laplacian matrices for the Spiral dataset are shown in Figures 19 and 20, respectively. The results are identical to those obtained using the unnormalized Laplacian. The only difference, as in the Circle dataset, lies in the scale of the eigenvalues due to the normalization process. Therefore, the same number of clusters was selected as in the unnormalized case: $M = 3, 3, 3$ for $k = 10, 20, 40$, respectively. The clustering outcomes are shown in Figure 21 and are completely identical to those obtained with the unnormalized Laplacians, as expected.

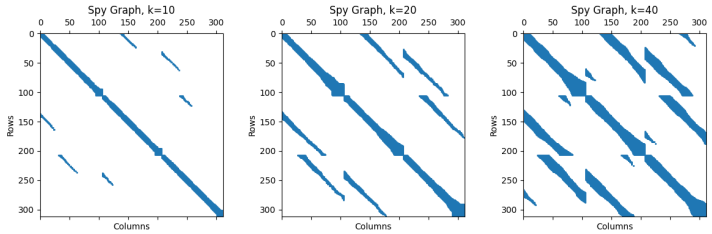

Figure 19: Sparsity pattern of the symmetric normalized Laplacians for the Spiral dataset

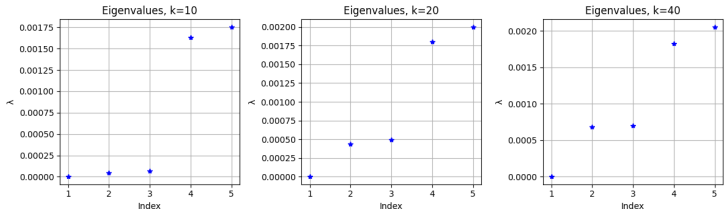

Figure 20: Five smallest symmetric Laplacian eigenvalues for the Spiral dataset

Exercise 12

In this exercise, the task was to implement the Inverse Power and Deflation method to compute the smallest eigenvalues of the Laplacian matrix.

The Inverse Power Method is an iterative algorithm for approximating the eigenvalue of smallest absolute magnitude of a diagonalizable matrix. Starting from

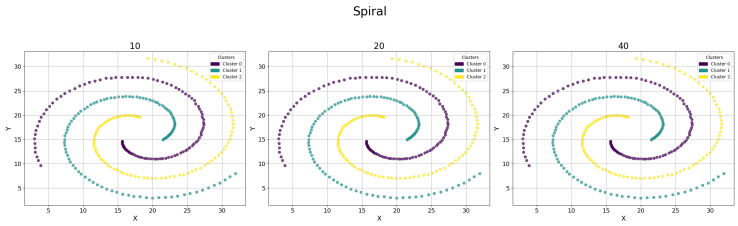

Figure 21: Clusters obtained for k = 10, 20, 40 and M = 3, 3, 3 respectively

a nonzero initial vector, the method repeatedly applies the inverse of the matrix and normalizes the result.

Given $A \in R^{n \times n}$ and $\mathbf{y}_0 \neq \mathbf{0}$, the iterations are:

$$\mathbf{z}_k = A^{-1}\mathbf{y}_{k-1}, \quad \mathbf{y}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|}$$

The process continues until convergence, and the corresponding eigenvalue is approximated by the Rayleigh quotient:

$$\lambda_k = \frac{\mathbf{y}_k^\top A \mathbf{y}_k}{\mathbf{y}_k^\top \mathbf{y}_k}$$

Since the Laplacian matrix is symmetric and therefore orthogonally diagonalizable, the method is applicable in our case. However, since the Laplacian is singular (it has at least one zero eigenvalue), we apply the Shifted Inverse Power Method to avoid inversion issues and target nonzero eigenvalues. This method involves shifting the matrix by a scalar p and applying the inverse power iteration to $(A - pI)^{-1}$. By choosing p close to the desired eigenvalue, the method ensures convergence to solution.

For this purpose, we implemented the `inverse_power_method` function, which takes as input the Laplacian matrix L, a shift value $p = 10^{-4}$, a convergence threshold of 10^{-12}, and a maximum number of iterations set to 1000. At each iteration, instead of computing the matrix inverse explicitly, we solve the linear system $(L - pI)\mathbf{z}_k = \mathbf{y}_{k-1}$ to obtain the next approximation vector. The algorithm stops either when the convergence threshold is met or when the maximum number of iterations is reached.

To compute multiple smallest eigenvalues and perform deflation, we introduce Householder transformation, which serves to remove the influence of the previously computed eigenvector and its corresponding eigenvalue. Given an eigenvector $\mathbf{x}$, the matrix is updated as

$$L' = HLH,$$

where H is the Householder matrix constructed to reflect $\mathbf{x}$. To eliminate the contribution of the provided eigenvector, it is sufficient to exclude the first row

and column of L'. For this purpose we implemented the `householder_deflation` function. It constructs a Householder matrix:

$$H = I - 2\frac{\mathbf{v}\mathbf{v}^\top}{\|\mathbf{v}\|^2},$$

where $\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$ and $\mathbf{u} = \mathbf{x} + \text{sign}(x_1)\|\mathbf{x}\|\mathbf{e}_1$, with $\mathbf{e}_1$ being the first standard basis vector. This transformation reflects the eigenvector $\mathbf{x}$ onto the first axis. The matrix is updated as $L' = HLH$, and the first row and column of L' are removed to obtain the reduced matrix L_2. The function returns both L_2 and H.

To combine the Inverse Power Method and deflation steps, we implemented the `find_M_smallest_eigenvalues` function. This method iteratively applies the Inverse Power Method to the input matrix L to compute the smallest eigenvalue and its corresponding eigenvector. After each step, Householder deflation is applied to eliminate the influence of the obtained eigenpair, reducing the matrix dimension. The shift parameter p is updated based on the most recently computed eigenvalue, to ensure that the next smallest eigenvalue is targeted in the next iteration.

Since deflation reduces the size of the matrix, the eigenvectors obtained in smaller subspaces are reconstructed in the original space by padding with zero and applying the stored Householder matrices in reverse order. Throughout the process, the function stores all computed eigenvalues and their corresponding eigenvectors, which are sorted before being returned.