POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Matematica

Spectral Clustering Report

Omento Davide
Geard Koci

Academic Year 2024-2025

Contents

List of Tables

List of Figures

1 Introduction

The aim of the project was to build the spectral clustering algorithm, test it on some given data and confront it with other clustering methods.

2 Theory on Spectral Clustering

Spectral Clustering is a clustering method that leverages the spectral properties (eigenvalues and eigenvectors) of a matrix derived from the data to perform grouping. It is based on the analysis of a graph constructed from the data, where nodes represent the points, and the edge weights reflect the similarity between points. This approach is particularly useful for detecting non-linear or arbitrarily shaped clusters.

The Spectral Clustering algorithm can be divided into the following steps:

1. **Construct the Similarity Graph** A similarity graph $G = (V, E)$ is constructed, where:
   - V is the set of nodes representing the data points;
   - E is the set of edges weighted by the similarity between points.

   The similarity matrix $\mathbf{S}$ is defined as:

$$S_{ij} = \begin{cases} \text{similarity between points } x_i \text{ and } x_j, & \text{if } x_i \text{ and } x_j \text{ are neighbors,} \\ 0, & \text{otherwise.} \end{cases}$$

2. **Option: Build a Symmetric k-NN Similarity Graph** To simplify the problem and focus only on the most relevant connections, it is possible to construct a symmetric k-NN similarity graph. Note that the parameter k here (for the k nearest neighbors) is different from m, which represents the number of clusters:
   - For each point x_i, consider its k nearest neighbors based on a distance or similarity metric;
   - Define the weight S_{ij} only if x_j is among the k nearest neighbors of x_i or x_i is among the k nearest neighbors of x_j. This ensures symmetry;
   - Set $S_{ij} = 0$ for all other pairs of points (i, j).

3. **Construct the Graph Laplacian** Using the similarity matrix, the Laplacian matrix $\mathbf{L}$ is defined, which can take several forms:
   - **Unnormalized:** $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ is the diagonal degree matrix with $D_{ii} = \sum_j S_{ij}$;
   - **Normalized symmetric:** $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$.

4. **Compute Eigenvalues and Eigenvectors** The first m eigenvectors of the Laplacian matrix $\mathbf{L}$ are computed. These eigenvectors provide a lower-dimensional representation of the data that preserves the graph's structural properties.

Observing the smallest eigenvalues' magnitudes can provide insight into the correct number of clusters. Specifically, the number of eigenvalues that are zero or very close to zero indicates the number of connected components in the graph, which corresponds to the number of clusters. When the graph is not perfectly disconnected, small non-zero eigenvalues still suggest natural cluster divisions;

5. **Perform Clustering in the Eigenvector Space** The computed eigenvectors are treated as features for the data points. A clustering algorithm (e.g., k-means) is applied to this reduced representation to partition the points into m clusters.

Advantages of Spectral Clustering

- It can identify non-linear or arbitrarily shaped clusters;
- It does not assume any specific cluster shapes (e.g., spherical clusters).

Disadvantages

- It requires the selection of certain hyperparameters, such as the number of clusters m and the method for constructing the similarity graph, including the choice of k for the k-NN graph;
- It can be computationally expensive for very large datasets due to the eigenvector computation.

3 Solutions

3.1 Task 1

After loading the data, we constructed the similarity matrix for both datasets, capturing the similarity between every pair of points. The next step was to implement a function called `knn_sim_graph`, whose purpose is to build the k-nearest neighbors (k-NN) similarity graph through the weight matrix $\mathbf{W}$.

In this function, for each point, we identify the k-closest points based on the similarity function:

$$S(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),$$

where, as required, we set $\sigma = 1$.

An important consideration is that we aim to construct a symmetric k-NN similarity graph. To achieve this, if the value at position (i, j) in the matrix is zero, but the value at position (j, i) is non-zero, we must assign the non-zero value symmetrically to (i, j). This ensures that the graph remains symmetric, reflecting mutual similarity between the points.

Below, we present the plots of the k-NN graph for different values of k:

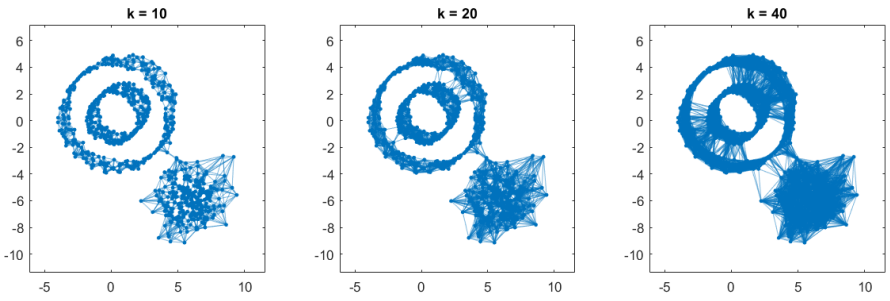

Figure 1: k-NN graphs for Circle dataset for $k = 10, 20, 40$.

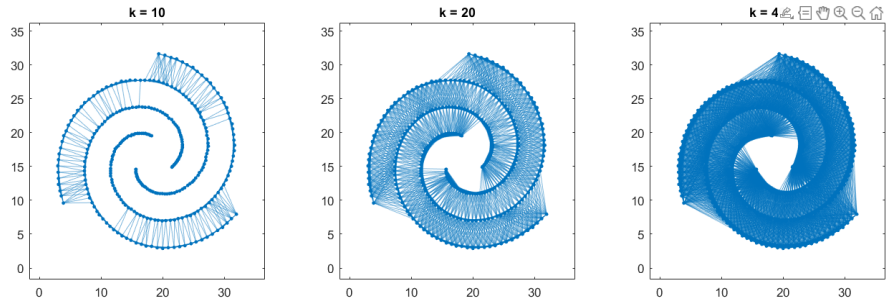

Figure 2: k-NN graphs for Spiral dataset for $k = 10, 20, 40$.

3.2 Task 2

Using the functions `diag()` and `sum()`, we defined the degree matrix $\mathbf{D}$ and the Laplacian matrix $\mathbf{L}$. To optimize memory usage, these matrices, along with the weight matrix $\mathbf{W}$, were stored in sparse format using the `sparse()` function.

3.3 Task 3

To compute the number of connected components of the similarity graph, we have two options:

- The first option is to use the function `conncomp()`, which directly computes the connected components of the graph;
- An alternative approach is to count the number of zero eigenvalues of the Laplacian matrix $\mathbf{L}$. The number of zero eigenvalues is equal to the number of connected components in the graph.

For the computation of the eigenvalues and eigenvectors, we implemented the *Inverse Power Method* combined with the *Deflation Method*. A detailed explanation of these methods will be provided in the following sections(see 3.12).

The number of connected components for the two graphs obtained with different values of k is shown below:

k	**Circle (connected components)**	**Spiral (connected components)**
10	2	1
20	1	1
40	1	1

Table 1: Number of connected components at different values of k

3.4 Task 4

Here we show the eigenvalues(computed with the implemented function `inverse_power_method()`. See 3.12 for details) of the Laplacian matrix respectively for the two datasets and for $k = 10, 20, 40$.

Table 2: 6 biggest eigenvalues for the two datasets at different values of k.

k	**Circle Dataset Eigenvalues**	**Spiral Dataset Eigenvalues**
10	0.0000	0.0000
	0.0000	0.0002
	0.0048	0.0003
	0.0286	0.0041
	0.0425	0.0044
	0.0429	0.0046
20	0.0000	0.0000
	0.0111	0.0018
	0.0652	0.0020
	0.1620	0.0048
	0.1724	0.0054
	0.3220	0.0056
40	0.0000	0.0000
	0.0482	0.0023
	0.7028	0.0025
	0.7797	0.0049
	0.9065	0.0062
	1.3768	0.0067

By analyzing the eigenvalues for $k = 10$, it is clear that the optimal number of clusters is 3 for both the circle and spiral datasets. This choice consistently appears to be the most accurate and reasonable across different values of k. However, for $k = 40$, the eigenvalues indicate that the optimal number of clusters for the circle dataset should be 2. This discrepancy can be attributed to the fact that with a larger k, each node has more neighbors, which leads to distant points being considered as close, thereby affecting the clustering results.

Through the analysis of the eigenvalues of the Laplacian matrix we have identified that:

- $m_{\text{circ}} = 3$ is the optimal number of clusters for the Circle dataset;
- $m_{\text{spiral}} = 3$ is the optimal number of clusters for the Spiral dataset.

3.5 Task 5

After determining the optimal number of clusters, m_{circ} for the Circle dataset and m_{spiral} for the Spiral dataset, we select the first m_{circ} and m_{spiral} eigenvectors corresponding to the smallest m_{circ} and m_{spiral} eigenvalues, respectively. These eigenvectors are then used to construct the matrices $\mathbf{U}_{\mathbf{circ}}$ and $\mathbf{U}_{\mathbf{spiral}}$, where each column represents one of the selected eigenvectors.
The eigenvectors were computed with the implemented function `inverse_power_method()`(See 3.12 for details).

3.6 Task 6

We then normalized the matrices $\mathbf{U}_{\text{circ}}$ and $\mathbf{U}_{\text{spiral}}$, and applied the k-means algorithm to them using the function `kmeans()`. This function returns the indices of the clusters to which each point is assigned. Here, we assumed that the number of clusters for both datasets is three. Therefore, this algorithm will assign each point an index between 1 and 3, representing the cluster to which it belongs.

3.7 Task 7

We used the `cell()` function to create our clusters as cell arrays because they are more flexible than regular arrays. This allowed us to store the points in their correct clusters. Then, using two `for` loops over the length of the spiral and circular datasets, we first copied the index computed earlier in task 6. After that, we accessed the cluster and assigned each point to its appropriate group.

3.8 Task 8

In task 8, we finally plot the results. To do this, we use the `subplot` function to compare the two plots side by side. We create a color map for both plots using the `lines` function, setting the size to match the number of clusters, which is 3. Then, using the `scatter` function, we plot the X and Y coordinates of the points, assigning them the same color as the cluster they belong to. Finally, we add the labels to the first plot and repeat the process for the second plot. The results can be seen in the following figures:

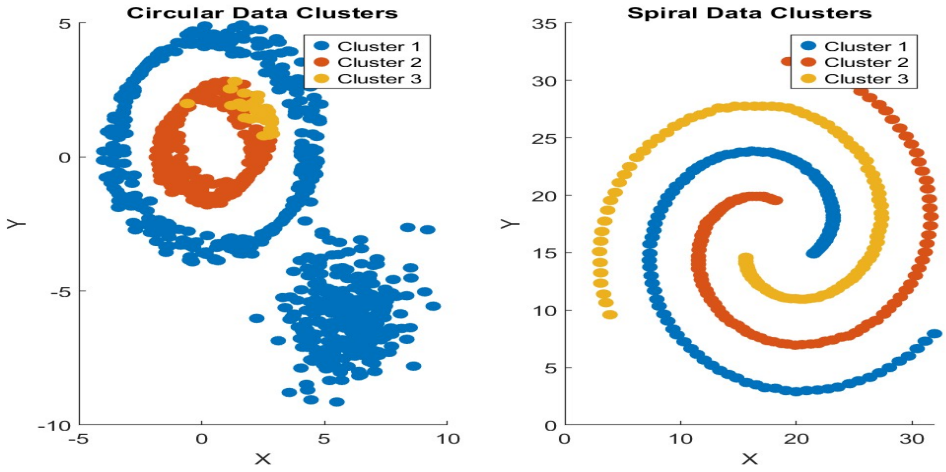

Figure 3: Clustering of points for $k = 10$.

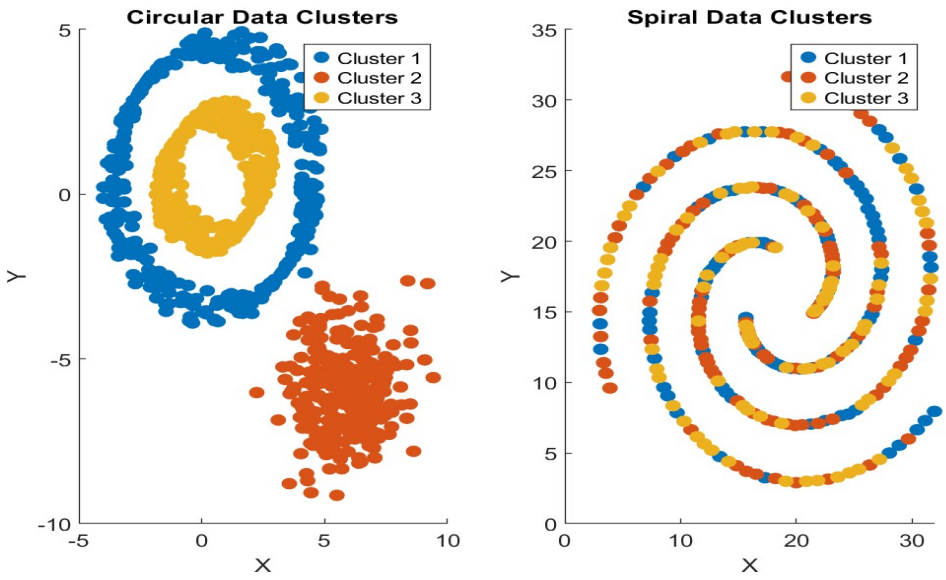

Figure 4: Clustering of points for $k = 20$.

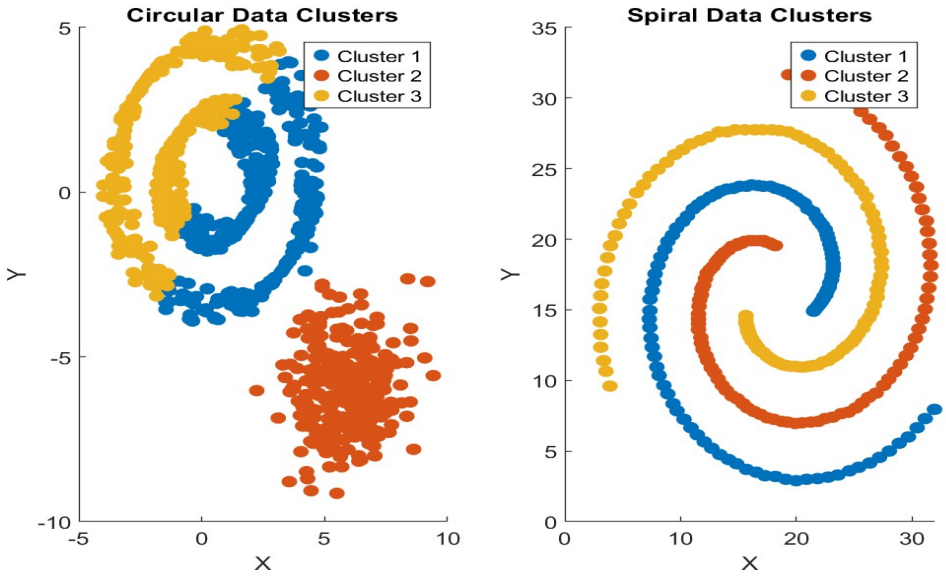

Figure 5: Clustering of points for $k = 40$.

We can see that for some values of k, such as 10 and 40, the Spiral dataset is correctly divided into the appropriate clusters. On the other hand, the Circular dataset performs best for k equal

to 20 and 40. However, with $k = 40$, the high number of neighbors hinders the proper separation of the inner and outer circles.

3.9 Task 9

Now we repeat what we have done from 3.6 until 3.8, but this time on the original dataset, without applying spectral clustering methods.
We directly applied the `kmeans` function to the circular and spiral datasets, created the cell arrays, and assigned the points to the labels following the same indices given by `kmeans`. Finally, we plotted both datasets using the same steps as in 3.8, and these were the results:

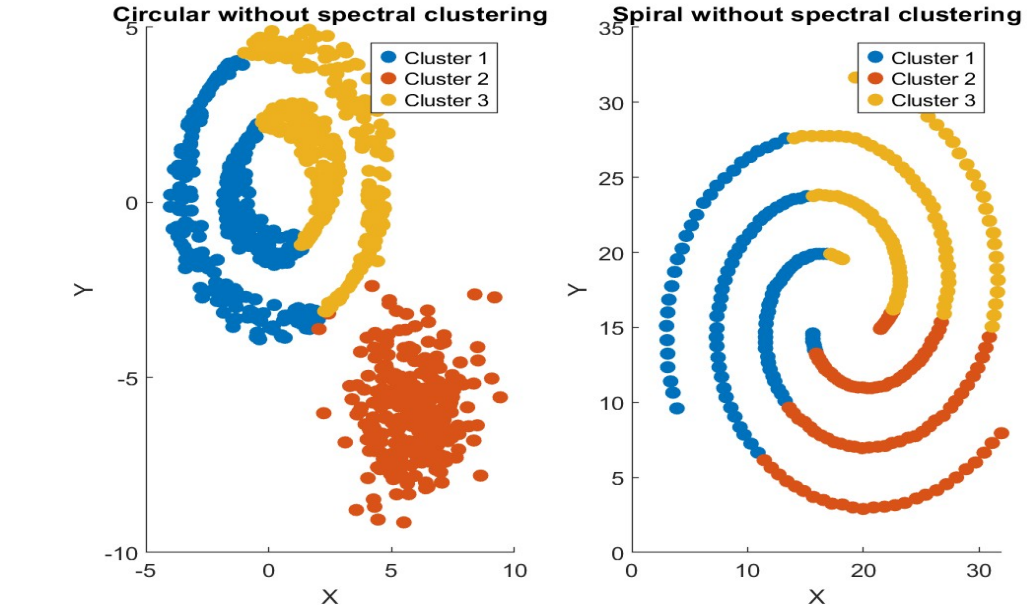

Figure 6: Clustering of points withou tspectral clustering.

It can be noted that the clustering is slightly off because the points are being separated solely based on their spatial coordinates and not by their connections with neighbors. This highlights the importance of spectral clustering methods.

3.10 Optional Task 1

The first optional task requires the creation or importation of another dataset, possibly in higher dimensions, so we decided to create a synthetic dataset by sampling from a normal multivariate distribution.
First, we set the seed, then using `mvnrnd`, we sample 100 points with unit variance (the identity matrix) and three different means: $[1, 1, 1]$, $[5, 5, 5]$, and $[9, 1, 1]$. Finally, we create the dataset by combining the three samples.

The following lines print the generated dataset, which can be seen in three dimensions in the figure below.

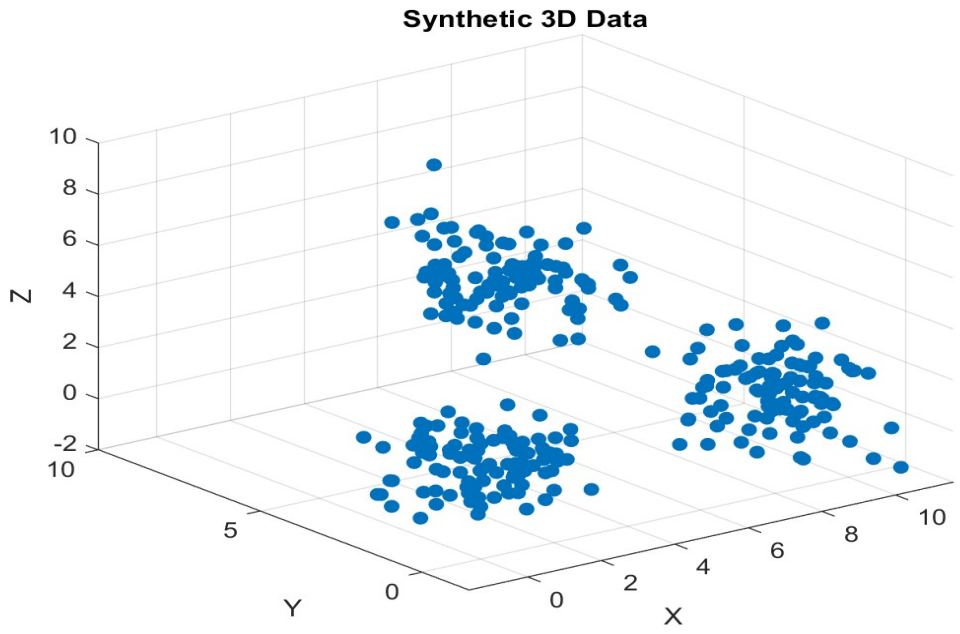

Figure 7: Synthetic dataset.

Now we apply the spectral clustering process to the synthetic dataset.In order, the following steps are performed:

1. Create the similarity graph.
2. Compute the matrices W, D, and L.
3. Apply the inverse power method to compute the eigenvectors corresponding to the three smallest eigenvalues.
4. Use the eigenvectors to construct the matrix U.
5. Normalize the matrix U.
6. Apply the `kmeans` function to perform clustering.

For reference, sections 1 to 7 can be seen.
The last steps involve assigning the points to the correct clusters, which are stored as cell arrays using the indices provided by `kmeans`. Finally, we plot the results, coloring the points according to the cluster they belong to.

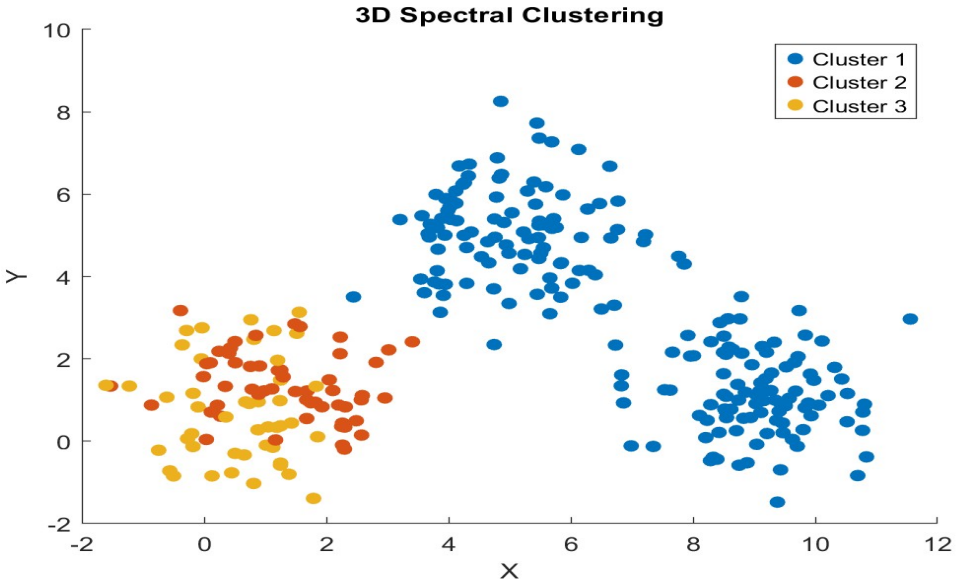

Figure 8: Clustering of points for $k = 10$.

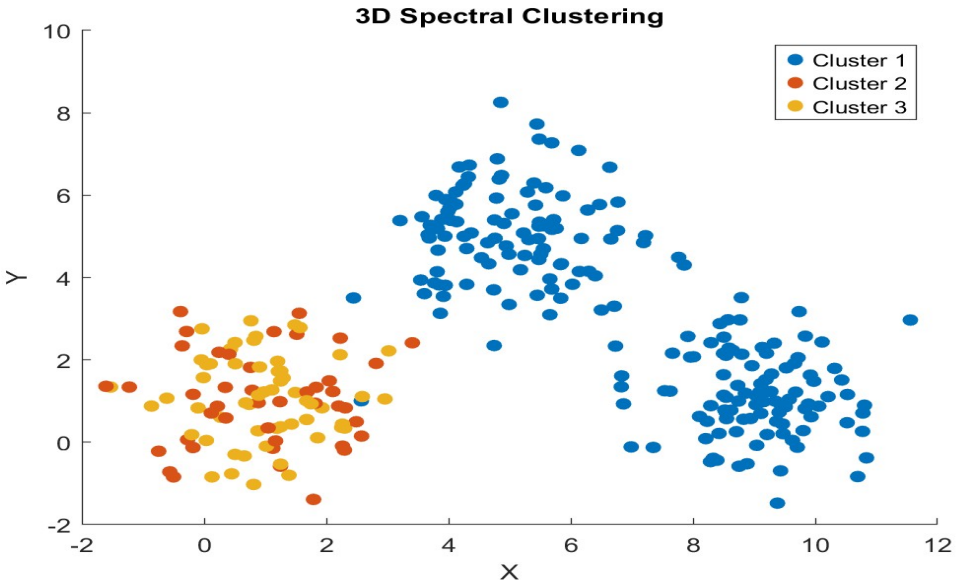

Figure 9: Clustering of points for $k = 20$.

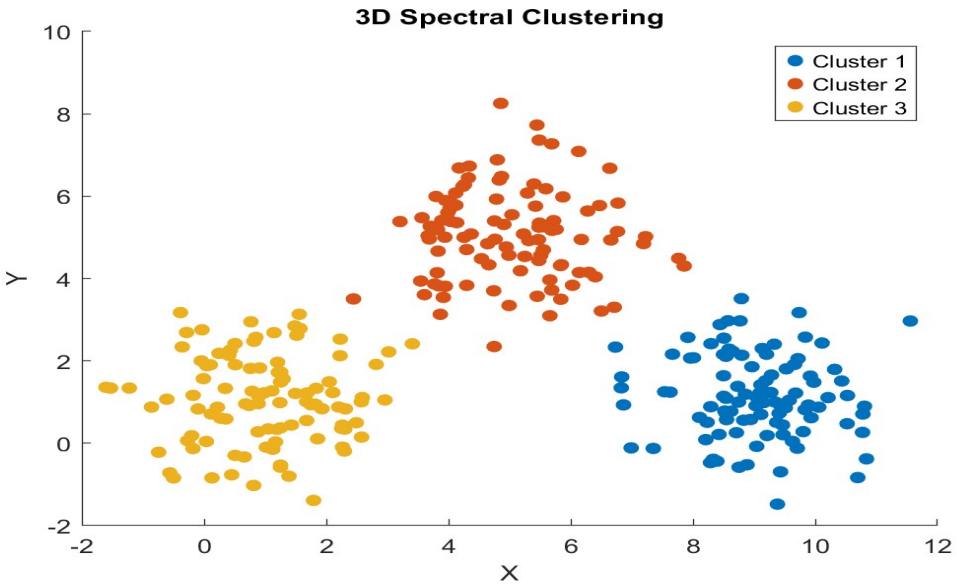

Figure 10: Clustering of points for $k = 40$.

It is easily understandable that the correct assignment of points happens with a higher k, such as $k = 40$ in this case. On the other hand, for values of k such as 10 and 20, the results are pretty much identical.

3.11 Optional Task 2

In this task, we again perform the spectral clustering process, as done in the previous tasks and the first optional one. However, this time we don't need to create a new dataset, but rather just define the matrix L_{sym} to use instead of L, which is computed as:

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

Then, the process follows the exact same steps as described in the list in 3.10.

The main difference can be found in the plots, as we decided to plot all four different plots, respectively with L and L_{sym}, to more easily denote the differences.

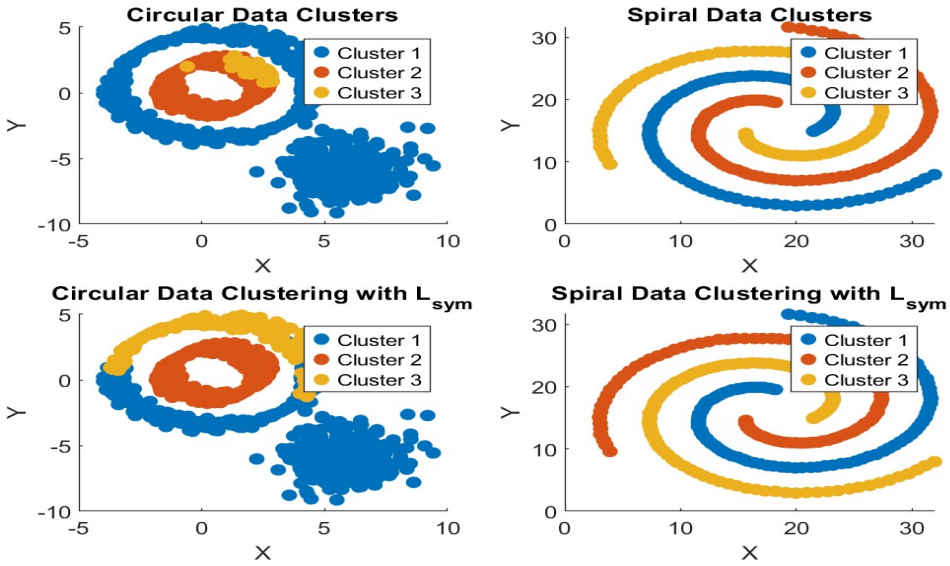

Figure 11: Clustering of points for $k = 10$.

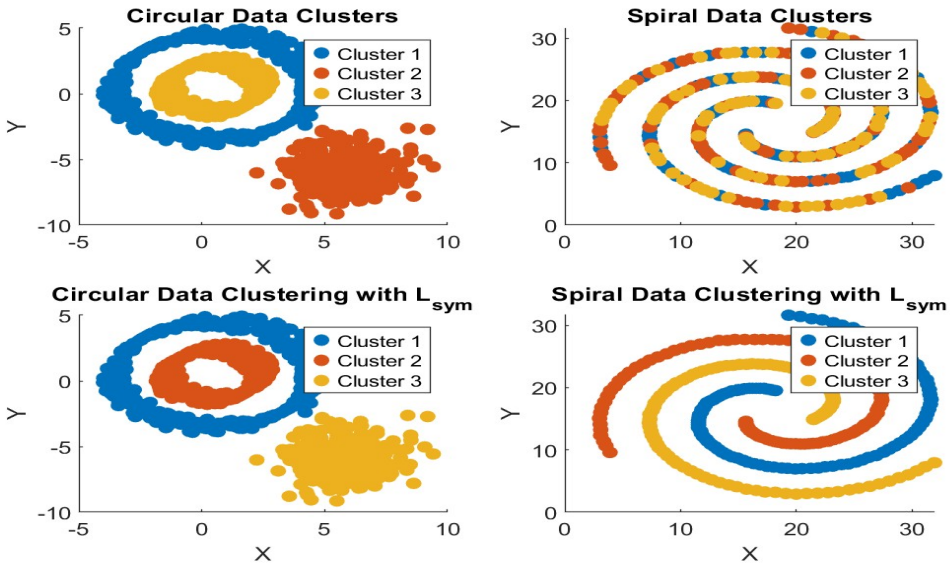

Figure 12: Clustering of points for $k = 20$.

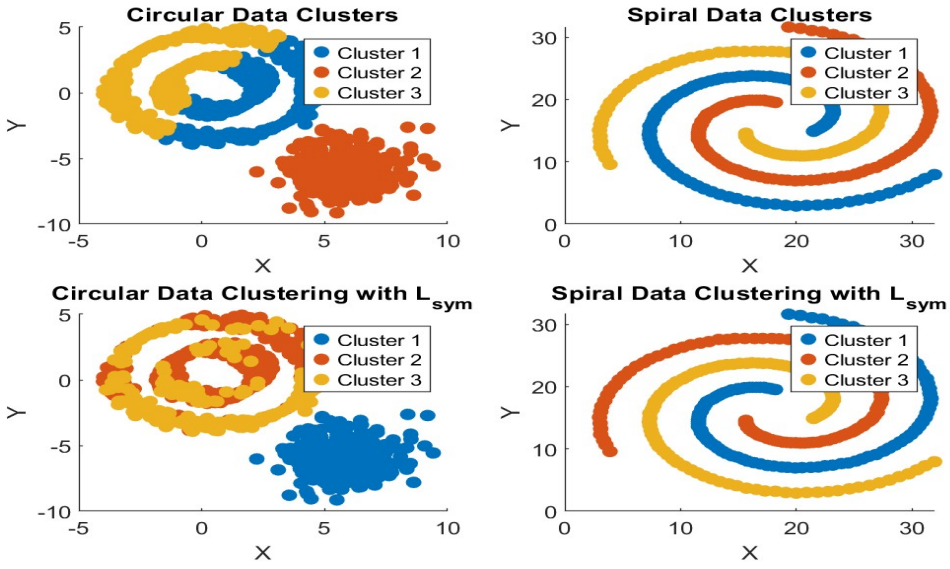

Figure 13: Clustering of points for $k = 40$.

We can analyze that for each value of k, symmetrizing the L matrix (i.e., using the matrix L_{sym}) allows for better assignments of the points in the spiral dataset. The symmetrization process helps to better capture the structure of the data, making it easier to assign points to their correct clusters.
However, the results are more challenging for the circular dataset. Specifically, for $k = 10$ or $k = 20$, the performance seems to improve compared to the original plot, as the points are more accurately grouped. But for higher values of k, the clustering becomes less precise, indicating that symmetrizing the matrix may not always lead to the best outcome for every case.
Therefore, while using L_{sym} has certain advantages, it also has drawbacks. The decision to use L_{sym} or not depends on the specific dataset and the value of k, and should be evaluated on a case-by-case basis.

3.12 Optional Task 3

For this task we implemented the inverse power method for computing the smallest eigenvalues and their corresponding eigenvectors of a given matrix B. The following provides a breakdown of the process:

1. **Input Parameters:** The function takes in four arguments:
    - B: The matrix for which the eigenvalues and eigenvectors are being computed.
    - M: The number of smallest eigenvalues and eigenvectors to compute.
    - tol: The tolerance for convergence, determining when to stop the iterations.
    - max_iter: The maximum number of iterations to perform before stopping, even if convergence hasn't been achieved.

2. **Initialization:** The function begins by determining the size of matrix B, specifically the number of rows (or columns, as the matrix is assumed to be square). This value is stored in n. The function initializes two arrays:

   - eigval, which will hold the computed eigenvalues.
   - eigvec, which will hold the corresponding eigenvectors.

3. **Eigenvalue and Eigenvector Calculation:** The function uses a loop that runs M times to compute M smallest eigenvalues and eigenvectors. For each iteration:

   - A random vector x is generated, which serves as an initial guess for the eigenvector. It is then normalized to have unit length.
   - The inverse power method is used to approximate the smallest eigenvalue and eigenvector. In each iteration, the matrix equation $B \cdot x = v$ is solved, where v is the updated vector.
   - The eigenvalue is computed using the Rayleigh quotient:

     $$\lambda = \frac{x' \cdot v}{x' \cdot x},$$

     which approximates the eigenvalue associated with the eigenvector.
   - The vector v is normalized to ensure that the next iteration starts with a unit vector, and the process continues until the change in the eigenvalue is smaller than the specified tolerance, indicating convergence.

4. **Deflation:** After computing each eigenvalue and eigenvector, the matrix B is updated (deflated) to remove the contribution of the recently computed eigenvalue/eigenvector pair. This is done by subtracting a rank-1 update:

   $$B = B - \lambda \cdot (x \cdot x'),$$

   from B. This deflation ensures that the next eigenvalue/eigenvector pair corresponds to the next smallest eigenvalue of the remaining matrix. The matrix B is adjusted to maintain numerical consistency after this update.

5. **Output:** After all the eigenvalues and eigenvectors are computed, the function returns two outputs:

   - eigval, a vector containing the computed smallest eigenvalues.
   - eigvec, a matrix where each column corresponds to an eigenvector corresponding to each eigenvalue.

The inverse power method is primarily used to find the smallest eigenvalues and their corresponding eigenvectors. This method is useful in spectral clustering, where eigenvectors associated with the smallest eigenvalues of the Laplacian matrix are used to identify clusters in data.