

## SPIRAL Dataset k=40

```
clear  
clc
```

Load the file

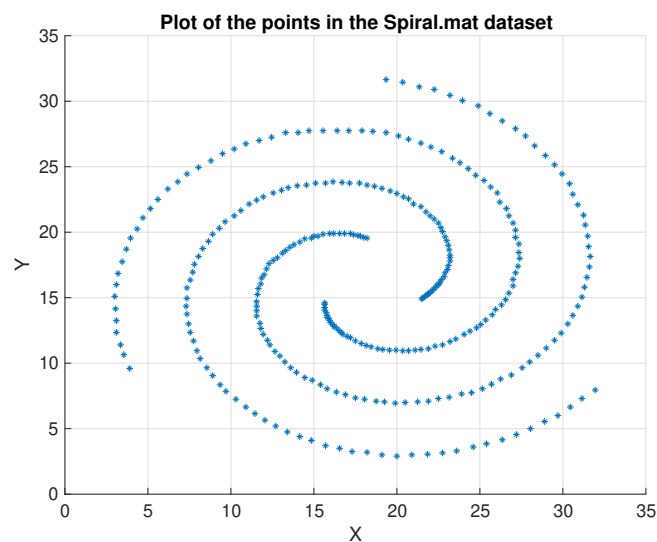
```
spiral_mat = load('Spiral.mat');  
  
% Display the structure of the file  
disp(spiral_mat);
```

```
X: [312x3 double]
```

```
% Extract the matrix of points  
% X_tot contains the right labels of the clusters as third column  
X_tot = spiral_mat.X;  
X = X_tot(:, 1:2);
```

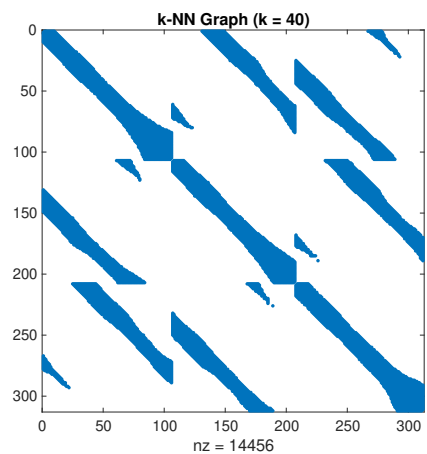
Plot the points

```
figure;  
scatter(X(:,1), X(:,2), 10, Marker='*');  
xlabel('X');  
ylabel('Y');  
title('Plot of the points in the Spiral.mat dataset');  
grid on;
```

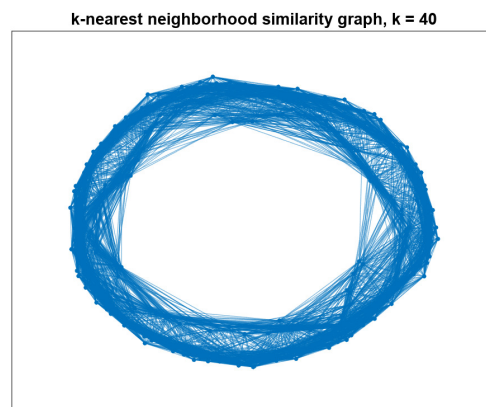


## 1. Similarity matrix and adjacency matrix

```
k_values = [10, 20, 40];  
k = 40;  
  
% Construct the k-nearest neighborhood similarity graph and its adjacency  
% matrix W  
W = knn_graph(X, k);  
  
% Visualize the graph using its similarity matrix  
figure;  
spy(W);  
title(['k-NN Graph (k = ', num2str(k), ')']);
```



```
% Store W as a sparse matrix  
W = sparse(W);  
  
% Visualize the graph G corresponding to the adjacency matrix W  
G = graph(W);  
figure;  
plot(G);  
title(['k-nearest neighborhood similarity graph, k = ', num2str(k)]);
```



## 2. Construct the degree matrix $D$ and the Laplacian matrix $L$

```
N = size(W, 1);

% Initialize the degree matrix D
D = zeros(N, N);

% The degree of each point is given by the sum of the elements of each row in W
D = diag(sum(W, 2));

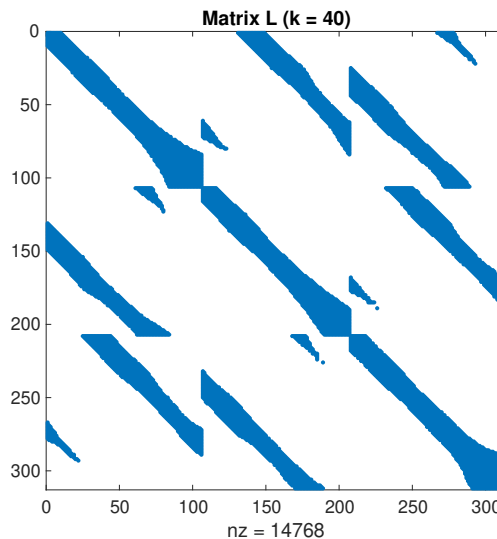
% Save D in a sparse format
D = sparse(D);

% Compute the Laplacian matrix L
L = D - W;

if issparse(L) % issparse(L)==1 means that L is stored in a sparse format
    disp("The matrix L is stored in a sparse format")
end
```

The matrix  $L$  is stored in a sparse format

```
% Plot the Laplacian matrix L
figure;
spy(L);
title(['Matrix L (k = ', num2str(k), ')']);
```



## 3. Compute the number of connected components of the similarity graph

```
% The points with the same number belong to the same connected component
bins = conncomp(G);

% Number of connected components
```

```

num_components = max(bins);

% Display the result
disp(['Number of connected components: ', num2str(num_components)]);

```

Number of connected components: 1

## 4 - 5. Compute eigenvalues and eigenvectors

```

% Set a number M of values to be computed (later it will be changed)
M = 10;

% Initialize the eigenvalues vector and the eigenvectors matrix
eigvalues = zeros(M, 1);
eigvects = zeros(N, M);

% Choose the vector v that will be used for the inverse power method
v = 0.5 * ones(N, 1);
v(1:2:N) = -0.5;

% Max iterations in the power method
maxIter = 1000;
% Relative tolerance
relTol = 1e-10;

% A known fact from theory is that L is semi pos def and has at least one
% eigenvalue = 0 and that the vector of all ones is a corresponding
% eigenvector
% eigvalues(1) = 0;
eigvects(:, 1) = ones(N,1)/ norm(ones(N,1));

% Or use the inverse power method to compute them
[eigvalues(1), eigvects(:, 1)] = invpower_method(L, v(1:end), maxIter, relTol);

% Compute the remaining eigenvectors and eigenvalues
[eigvalues, eigvects, residualnorms] =
    deflation_method(L, v, eigvects, eigvalues, M, maxIter, relTol);

% Check how good the approximation is by comparing with eigs function of
% Matlab
[mat_eigvects, mat_eigs] = eigs(L, M, 'smallestabs');
norm(eigvalues - diag(mat_eigs))

```

ans = 1.0174e-10

Now, find the actual number M of eigenvalues that will be used for the clustering algorithm

```

% Plot the computed eigenvalues
x = 1:M;

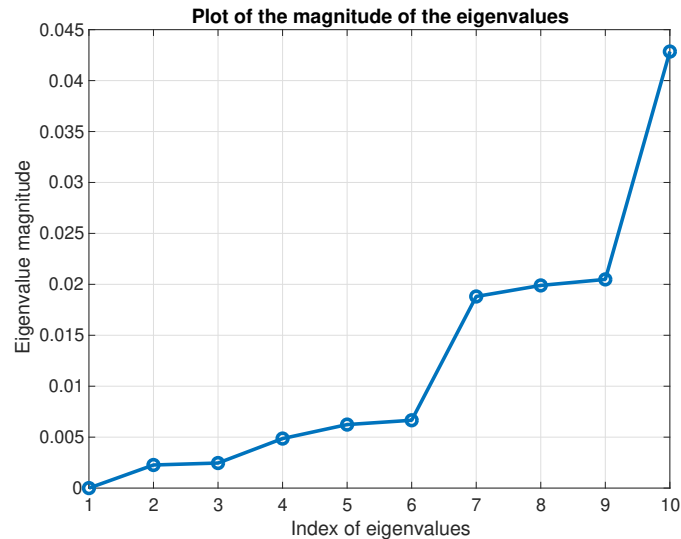
figure

```

```

plot(x, eigvalues, '-o', 'LineWidth', 2);
xlabel('Index of eigenvalues');
ylabel('Eigenvalue magnitude');
title('Plot of the magnitude of the eigenvalues');
grid on;

```



The suitable number of eigenvalues is 3 since eig4 is much larger than eig3 for  $k=40$

```

M = 3;

% Define the matrix U that will be used for the spectral clustering
U = eigvects(:, 1:M);

```

## 6 - 7 - 8. Spectral clustering, k means

```

% Clusterize using k means and obtain the indices (and the centroids)
% inside the clusters of each point
[idx, C] = kmeans(U, M);

% Assing the original data to the corresponding clusters
A = cell(M, 1);

for i = 1:N
    % Find the cluster of y_i
    cluster_idx = idx(i);

    % Assing it to x_i
    A{cluster_idx} = [A{cluster_idx}; X(i, :)];
end

% Plot of the clusterized data in the original space
X_spect_clust = zeros(N, 3);

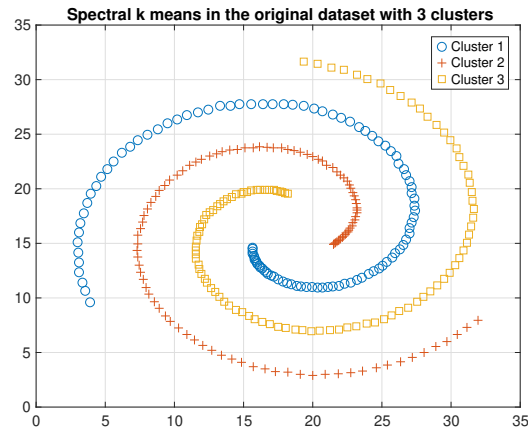
```

```

X_spect_clust(:, 1: 2) = X;
X_spect_clust(:, 3) = idx;

figure;
markers = ['o', '+', 's'];
gscatter(X_spect_clust(:,1), X_spect_clust(:,2), X_spect_clust(:, 3), [], markers, [], 5);
title(['Spectral k means in the original dataset with ', num2str(M), ' clusters']);
legend('Cluster 1', 'Cluster 2', 'Cluster 3');
grid on;

```



## 9.a K MEANS TO THE ORIGINAL DATA

```

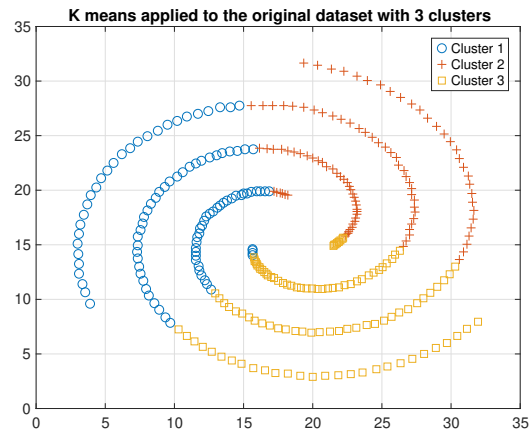
k_value = 3;

% Clusterize the original data
[idx_k, C_k] = kmeans(X, k_value);

% Add the index to X_kmeans
X_kmeans = zeros(N, 3);
X_kmeans(:, 1: 2) = X;
X_kmeans(:, 3) = idx_k;

figure;
markers = ['o', '+', 's'];
gscatter(X_kmeans(:,1), X_kmeans(:,2), X_kmeans(:, 3), [], markers, [], 5);
title(['K means applied to the original dataset with ', num2str(k_value), ' clusters']);
legend('Cluster 1', 'Cluster 2', 'Cluster 3');
grid on;

```



## 9.b DBSCAN TO THE ORIGINAL DATA

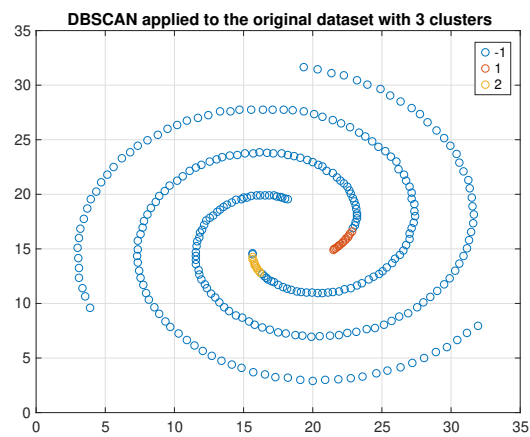
Apply the DBSCAN algorithm

```
% Neighborhood radius
epsilon = 0.5;
% Minimum points for a cluster
minPts = 6;

idx_d = dbscan(X, epsilon, minPts);

X_dbscan = zeros(N, 3);
X_dbscan(:, 1: 2) = X;
X_dbscan(:, 3) = idx_d;

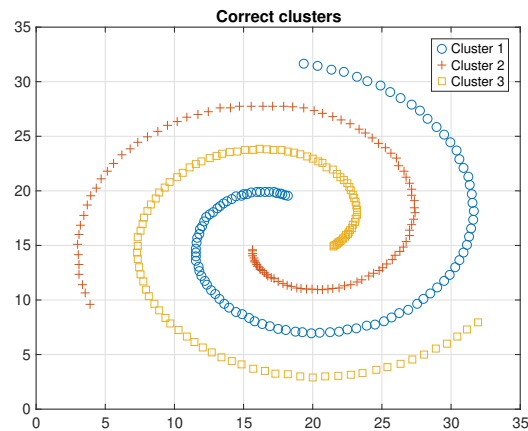
figure;
gscatter(X_dbscan(:,1), X_dbscan(:,2), X_dbscan(:, 3), [], 'o', 5);
title(['DBSCAN applied to the original dataset with ', num2str(M), ' clusters']);
grid on;
```



## Difference with original clusters

Plot the original clusters

```
figure;  
markers = ['o', '+', 's'];  
gscatter(X_tot(:,1), X_tot(:,2), X_tot(:, 3), [], markers, [], 5);  
title('Correct clusters');  
legend('Cluster 1', 'Cluster 2', 'Cluster 3');  
grid on;
```



```
% Compute the difference with the ones given by spectral clustering  
  
% Shift the numbers in X_spect_clust(:,3) to have the same names as in  
% X_tot  
  
first_value = X_spect_clust(1, 3);  
second_value = X_spect_clust(160, 3);  
third_value = X_spect_clust(310, 3);  
  
for i = 1:length(X_spect_clust(:, 3))  
    if X_spect_clust(i, 3) == first_value  
        X_spect_clust(i, 3) = 3;  
    elseif X_spect_clust(i, 3) == second_value  
        X_spect_clust(i, 3) = 1;  
    elseif X_spect_clust(i, 3) == third_value  
        X_spect_clust(i, 3) = 2;  
    end  
end  
  
% Compute the difference  
diff_clust = sum((X_tot(:,3) - X_spect_clust(:,3)))  
  
diff_clust = 0
```