AMRITA
VISHWA VIDYAPEETHAM

21MAT212 - MIS-IV

ASSIGNMENT – 4

Group Members:

BATCH-A	TEAM-1
R.SRIVISWA	CB.EN.U4AIE21046
AMAN SIROHI	CB.EN.U4AIE21003
RAKHIL ML	CB.EN.U4AIE21048
VIKHYAT BANSAL	CB.EN.U4AIE21076

Submitted by: Vikhyat Bansal [CB.EN.U4AIE21076]

Creating a graph with a predefined number of nodes and edges.

Code: Function

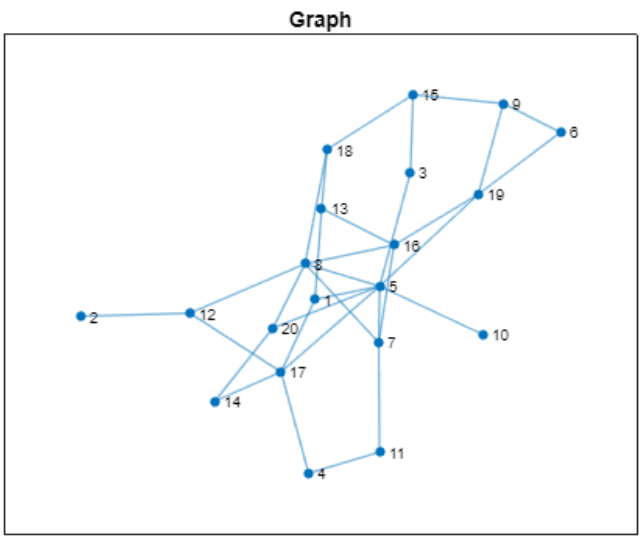
```
function adjMatrix = randomSimpleGraph (n, e)
    adjMatrix = zeros (n);
    while sum (adjMatrix(:)) < 2*e % When the matrix written in row vector
form is less than twice the number of edges
        i = randi (n);
        j = randi (n);
        if i ~= j && adjMatrix(i,j) == 0
            adjMatrix(i,j) = 1;
            adjMatrix(j,i) = 1;
        end
    end
end
```

Input:

```
% for generating a random graph
A = randomSimpleGraph(20,32);
G = graph(A)
plot(G)
title("Graph")
```

Output:

```
G =
  graph with properties:

    Edges: [32×2 table]
    Nodes: [20×0 table]
```

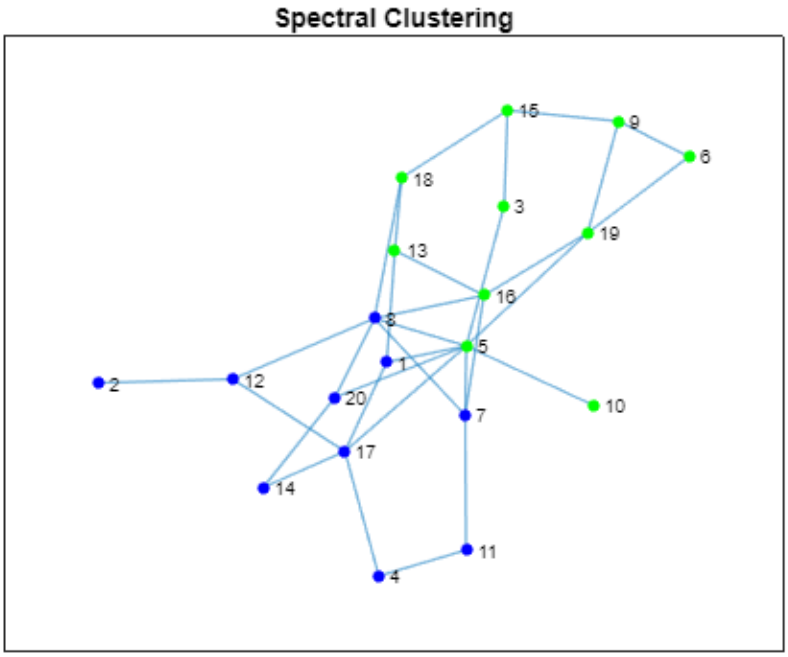


Code for performing Bipartition via spectral method on a graph.

Code:

```
L=laplacian(G); % Getting the laplacian matrix of graph G


[eigvec,eigval]=eigs(L,2,'smallestabs');  %the two smallest absolute
eigenvalues and eigenvectors of L


i=eigvec(:,2); % taking the eigenvector corresponding to eigenvalue 2
c1=[];
c2=[];


for j=1:length(i)
    if i(j) > 0
        c1=[c1 j];
    else
        c2=[c2 j];
    end
end

spec_clus=plot(G);
title("Spectral Clustering")
highlight(spec_clus,c1,'NodeColor','g')
highlight(spec_clus,c2,'NodeColor','b')
```

Output:



Code for performing k-means clustering on a graph.

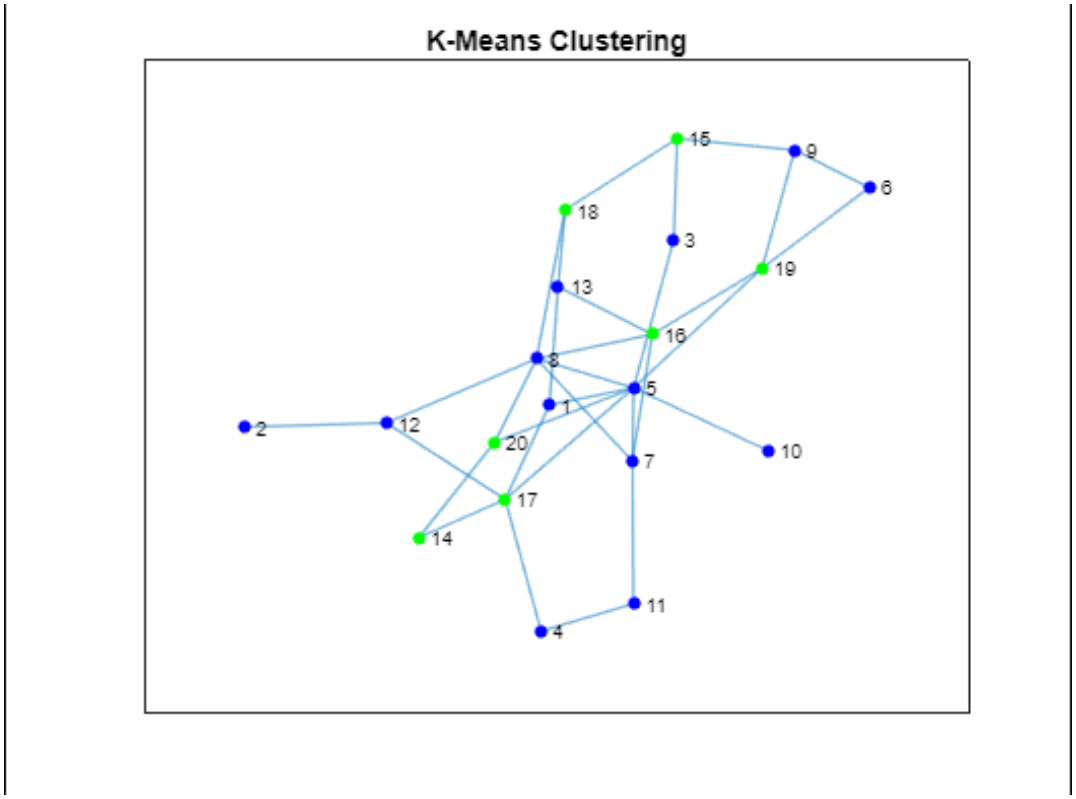
Code: Function

```
function [idx, centroids] = kmeans_using_for_loop(data, k)%function to perform
kmeans
% Randomly initialize the centroids
idx = randi(k, size(data, 1), 1);
centroids = zeros(k, size(data, 2));
% Iterate until convergence
while true
    for i = 1:k
        centroids(i, :) = mean(data(idx == i, :));
    end
% Assign data points to the nearest centroid
    prevIdx = idx;
    for i = 1:size(data, 1)
        distances = sum((data(i, :) - centroids).^2, 2);
        [~, idx(i)] = min(distances);
    end
% Check for convergence
    if isequal(prevIdx, idx)
    break;
    end
end
end
```

Input:

```
evalue=eig(L); % The eigenvalues of graph L
idx=kmeans_using_for_loop(evalue,2); %Performing the kmeans on the graph


kc1=[];
kc2=[];

for j=1:length(idx)
    if idx(j)==1  % Splitting the data points to the clusters based on the
centroid provided by k-means function
        kc1=[kc1 j];
    else
        kc2=[kc2 j];
    end
end

kmeans_clus=plot(G);
title("K-Means Clustering")
highlight(kmeans_clus,kc1,'NodeColor','g')
highlight(kmeans_clus,kc2,'NodeColor','b')
```

Output:



Difference between spectral clustering and k-means clustering

Spectral Clustering and K-Means Clustering are both unsupervised machine learning algorithms used for clustering, but they differ in how they define clusters.

- K-Means Clustering defines clusters as spherical clusters of points, where each point belongs to the cluster with the nearest mean. Spectral Clustering, on the other hand, defines clusters based on the eigenvectors of the similarity matrix of the data.
- In K-Means, the number of clusters is specified in advance and the algorithm tries to find the best clustering based on that number of clusters.In Spectral Clustering, the number of clusters is not specified and the algorithm finds the number of clusters that best fit the data.

- Spectral Clustering is often used when the data is not spherical or when the clusters are not well separated, such as when the data is non-linearly separable. K-Means, on the other hand, is often used when the data is spherical and the clusters are well separated.

In summary, Spectral Clustering is more flexible than K-Means and can be used in more complex scenarios, but it is also more computationally intensive.

Suppose we have a dataset of gene expression levels for different samples, where each sample has thousands of genes. The goal is to cluster the samples based on their gene expression profiles.
In this scenario, the data is likely to be high-dimensional, non-spherical, and the clusters may not be well separated.

Spectral Clustering is well-suited to this type of data because it can capture the underlying structure of the data, even if it is non-linear and high-dimensional. The algorithm can use a similarity matrix to measure the similarity between each pair of data points, and then use the eigenvectors of this matrix to find a low-dimensional representation of the data that preserves the important structure.

On the other hand, K-Means may struggle with this type of data because it assumes that the clusters are spherical and well-separated. It may also be sensitive to the initial choice of centroids and may not be able to capture the complex structure of the data.

Spectral Clustering has been used successfully in bioinformatics for clustering gene expression data, protein-protein interaction networks, and other types of biological data. By using Spectral Clustering, researchers can identify groups of genes or proteins that are co-expressed or functionally related, which can provide insights into important biological processes and pathways.

SPECTRAL CLUSTERING

By R SRIVISWA

CB.EN.U4AIE21046

Spectral clustering is a technique used for clustering data points based on the spectral properties of a similarity matrix or graph. It combines concepts from graph theory, linear algebra, and clustering algorithms to partition the data into distinct groups.

Given below is a brief overview of how Spectral Clustering works:

1. Given a dataset with n data points, construct a similarity matrix or graph representation. Common approaches include using a k-nearest neighbours graph or a Gaussian kernel to measure pairwise similarities between data points.
2. Compute the Laplacian matrix of the similarity matrix/graph. The Laplacian matrix captures the relationships between data points and serves as a measure of connectivity.
3. Compute the eigenvectors corresponding to the k smallest eigenvalues of the Laplacian matrix. These eigenvectors represent a low-dimensional embedding of the data.
4. Perform k-means clustering on the embedded data represented by the eigenvectors. The resulting clusters correspond to the groups of data points.

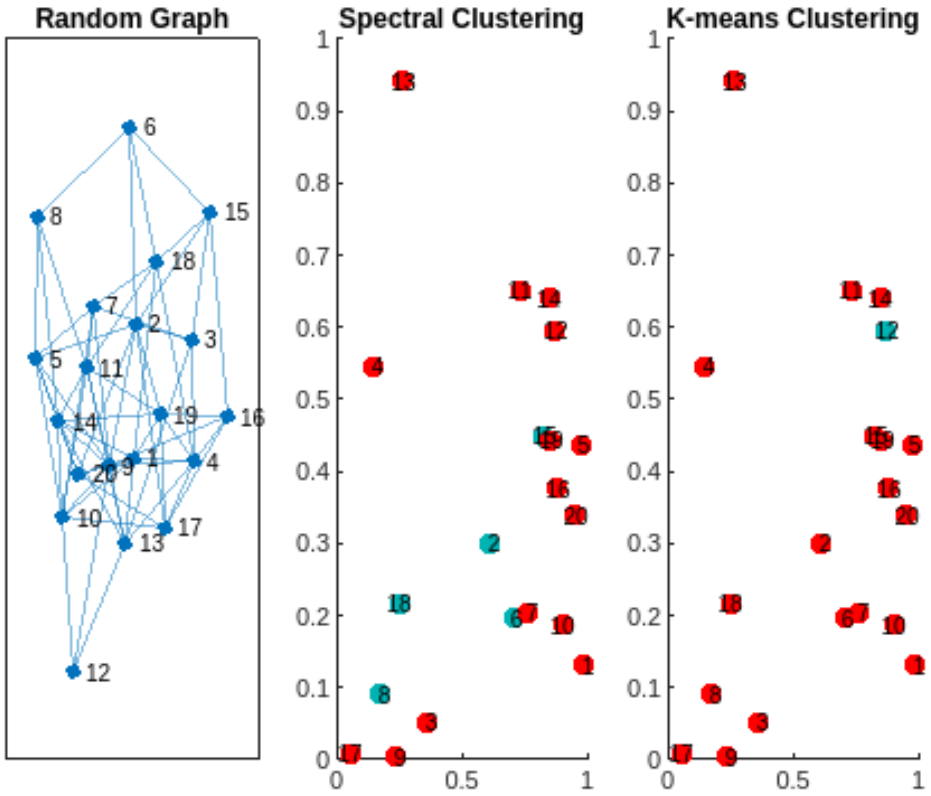
We will be using the same function that we created in the previous assignment for K-Means Clustering.

```
function [idx, centroids] = mykmeans(data, k)
%function to perform kmeans, Randomly initialize the
centroids
idx = randi(k, size(data, 1), 1);
centroids = zeros(k, size(data, 2));
% Iterate until convergence
while true
% Update the centroids
for i = 1:k
centroids(i, :) = mean(data(idx == i, :));
end
% Assign data points to the nearest centroid
prevIdx = idx;
for i = 1:size(data, 1)
distances = sum((data(i, :) - centroids).^2, 2);
[~, idx(i)] = min(distances);
end
% Check for convergence
if isequal(prevIdx, idx)
break;
end
end
end
```

SPECTRAL CLUSTERING & KMEANS CLUSTERING

```
%Generate a random graph
n = 20; % Number of nodes
p = 0.3; % Probability of an edge between two nodes
adjacency_matrix = triu(rand(n) < p, 1);
adjacency_matrix = adjacency_matrix + adjacency_matrix.';

%Compute the Laplacian matrix
degree_matrix = diag(sum(adjacency_matrix, 2));
laplacian_matrix = degree_matrix - adjacency_matrix;

%Compute the eigenvectors of the Laplacian matrix
k = 2;
[V, ~] = eigs(laplacian_matrix, k, 'smallestabs');

%Perform spectral clustering
spectral_labels = mykmeans(V, k);

%Perform k-means clustering
kmeans_labels = mykmeans(V(:, 2:k), k);

%Plotting the graph
X = rand(n, 1);
Y = rand(n, 1);
 G = graph(adjacency_matrix);
```

```
figure;

subplot(1, 3, 1);
p1 = plot(G, 'NodeLabel', 1:n);
title('Random Graph');

subplot(1, 3, 2);
light_color_map = hsv(k);
light_color_map(:, 2:3) = light_color_map(:, 2:3) * 0.7;
scatter(X, Y, 50, light_color_map(spectral_labels, :),
'filled');
text(X, Y, num2str((1:n)'), 'FontSize', 8,
'HorizontalAlignment', 'center');
title('Spectral Clustering');

subplot(1, 3, 3);
scatter(X, Y, 50, light_color_map(kmeans_labels, :),
'filled');
text(X, Y, num2str((1:n)'), 'FontSize', 8,
'HorizontalAlignment', 'center');
title('K-means Clustering');
```



Submitted by: Aman Sirohi [CB.EN.U4AIE21003]

CODE:

```
% Generate a random graph
    n = 13; % number of nodes
    p = 0.1; % edge probability
    W = rand(n); % random weights
    W = triu(W,1); % upper triangular part
    A = W + W.'; % make the matrix symmetric
    G = graph(A);

    plot(G);
    title('Random Graph');

    % Performing spectral clustering
    L = laplacian(G);
    [ev,~] = eigs(L,22,'smallestabs');
    v = ev(:,2);
    c1 = find(v > 0);
    c2 = find(v <= 0);
    idx_sc = zeros(1, n);
    idx_sc(c1) = 1;
    idx_sc(c2) = 2;
    % Ploting the graph with coloured nodes
    h = plot(G);
    title('Spectral Clustering')
    highlight(h,c1,'NodeColor','r')
    highlight(h,c2,'NodeColor','b')

    % Performing K-Means clustering
    k = 2; % number of clusters
    [~, C] = kmeans(ev, k);
    [~, idx] = min(pdist2(ev, C), [], 2);
    kc1 = find(idx == 1);
    kc2 = find(idx == 2);
    idx_km = zeros(1, n);
    idx_km(kc1) = 1;
    idx_km(kc2) = 2;
    % Ploting the graph with coloured nodes
```

```
    kh = plot(G);
    title('K-Means Clustering');
    highlight(kh,kc1,'NodeColor','r')
    highlight(kh,kc2,'NodeColor','b')

    % Calculate Jaccard similarity between spectral
clustering and K-Means clustering
    jaccard = sum(idx_sc == idx_km) / n;

    fprintf('Jaccard similarity between spectral
clustering and K-Means clustering: %.2f\n', jaccard);
```

Explanation:

This MATLAB code can be divided into three parts:

1. Graph Generation:

   The first part generates a random graph with 13 nodes and edge probability of 0.1. The adjacency matrix of the graph is generated randomly using the rand function and then converted to a symmetric matrix by adding its transpose. The upper triangular part of the matrix is retained using the triu function to avoid self-loops and duplicate edges. Finally, the graph is plotted using the plot function.

2. Spectral Clustering:

   The second part performs spectral clustering on the generated graph. The Laplacian matrix of the graph is computed using the laplacian function. The second smallest eigenvalue and eigenvector are computed using the eigs function. The nodes are then partitioned into two clusters based on the sign of the second eigenvector. The graph is plotted again with the nodes coloured according to their cluster assignment.

3. K-Means Clustering:

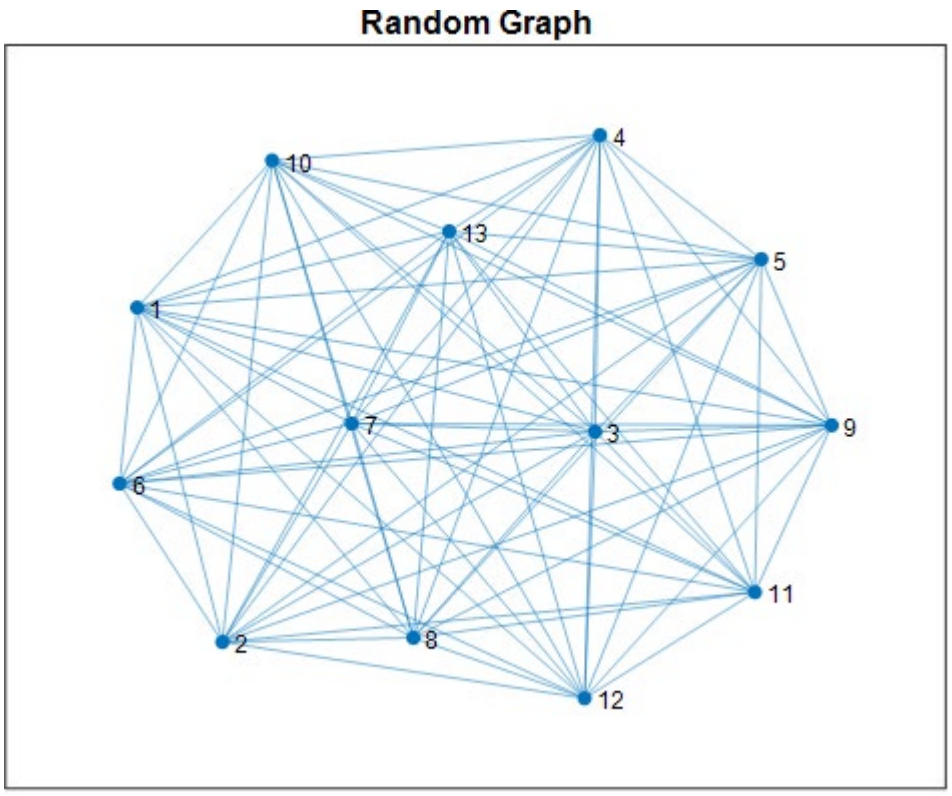
   The third part performs K-means clustering on the same Laplacian matrix eigenvalues used in spectral clustering. The number of clusters is set to 2 and the kmeans function is used to compute the cluster centres. The cluster assignments of the nodes are then determined using the pdist2 function, which calculates the Euclidean distance between each node and the

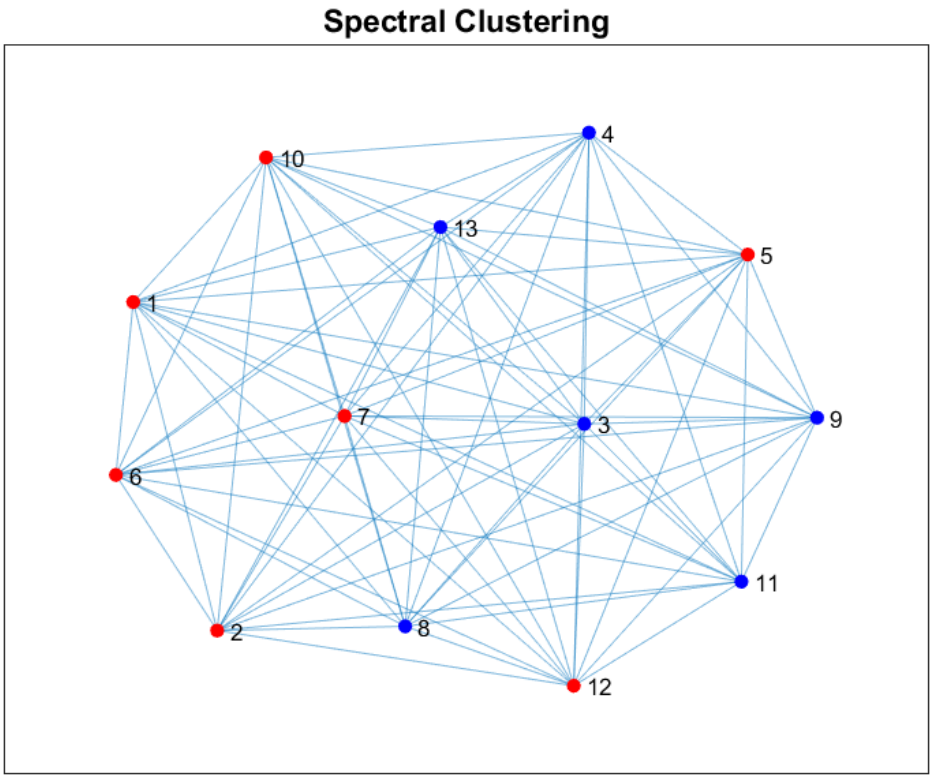
cluster centres. The graph is plotted again with the nodes coloured according to their K-means cluster assignment.

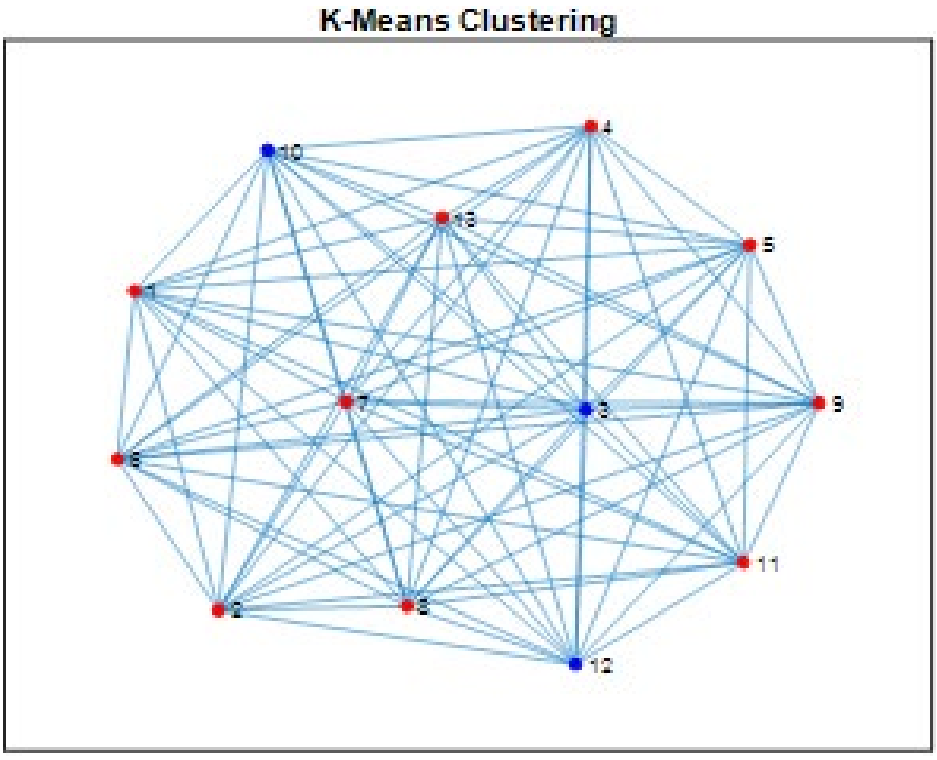
Finally, the Jaccard similarity coefficient (the Jaccard similarity coefficient is a measure of similarity between two sets of data) between the spectral clustering and K-means clustering results is computed and printed using the fprintf function, which displays the similarity coefficient with two decimal places. Basically, Jaccard is the ratio of the number of nodes that have the same cluster assignment under both methods (i.e., idx_sc(i) == idx_km(i)), to the number of nodes that have different cluster assignments under the two methods (i.e., idx_sc(i) ~= idx_km(i)).

In summary, this code generates a random graph, performs spectral clustering and K-means clustering on the graph, and computes the Jaccard similarity between the two clustering methods.

Output:







Jaccard similarity between spectral clustering and K-Means clustering: 0.46

Submitted by: Rakhil ML [CB.EN.U4AIE21048]

Clustering of graph using spectral clustering and compare it with k-means clustering

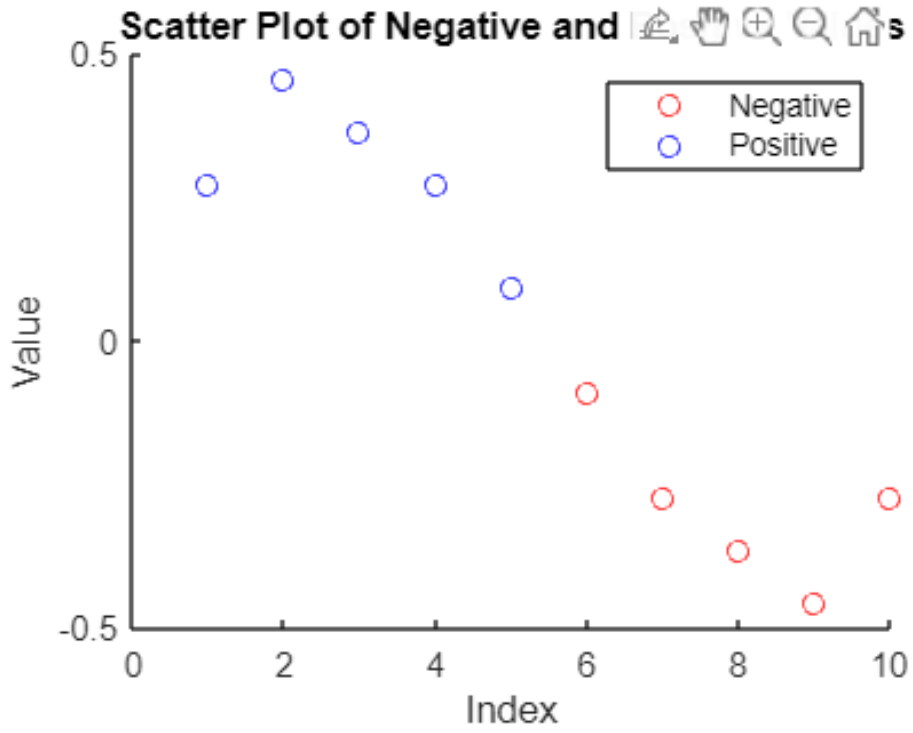
CODE :

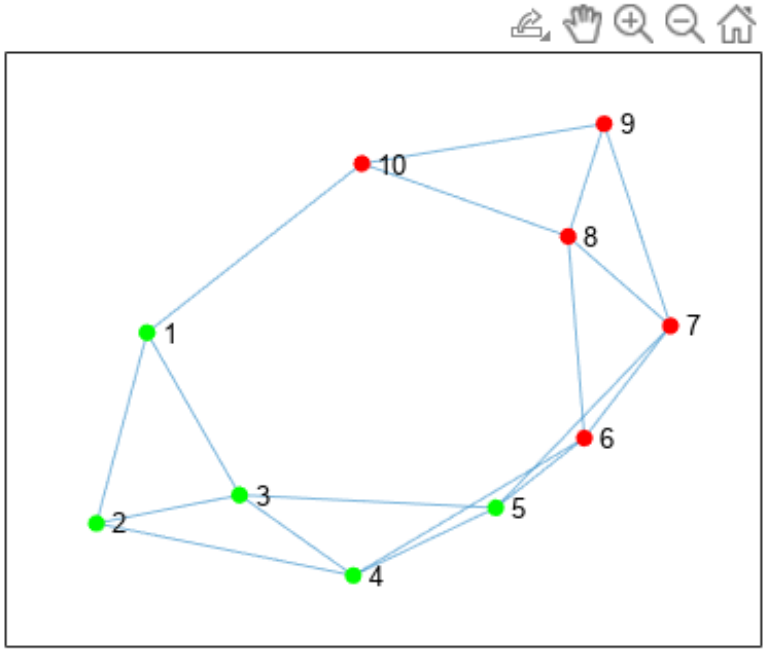
```
% Adjacency matrix
adjacency = [0, 1, 1, 0, 0, 0, 0, 0, 0, 1;
             1, 0, 1, 1, 0, 0, 0, 0, 0, 0;
             1, 1, 0, 1, 1, 0, 0, 0, 0, 0;
             0, 1, 1, 0, 1, 1, 0, 0, 0, 0;
             0, 0, 1, 1, 0, 1, 1, 0, 0, 0;
             0, 0, 0, 1, 1, 0, 1, 1, 0, 0;
             0, 0, 0, 0, 1, 1, 0, 1, 1, 0;
             0, 0, 0, 0, 0, 1, 1, 0, 1, 1;
             0, 0, 0, 0, 0, 0, 1, 1, 0, 1;
             1, 0, 0, 0, 0, 0, 0, 1, 1, 0];


% Degree matrix
degree = diag(sum(adjacency, 2));


% Laplacian matrix
Laplacian = degree - adjacency;


% Compute eigenvalues and eigenvectors
[V, E] = eig(Laplacian);


% Create a graph
g = graph(adjacency);
plot(g);


% Select the second smallest eigenvector
s = V(:, 2);


% Find indices of negative and positive values in the eigenvector
negative = find(s < 0);
positive = find(s >= 0);


% Plot scatter plot of negative and positive values
figure;
```

```
scatter(negative, s(negative), 'r');
hold on;
scatter(positive, s(positive), 'b');
xlabel('Index');
ylabel('Value');
title('Scatter Plot of Negative and Positive Values');
legend('Negative', 'Positive');


% Prepare node colors for plotting based on negative and positive values
node = zeros(10, 3);
node(negative, 1) = 1;
node(positive, 2) = 1;


% Plot the graph with colored nodes
figure;
plot(g, 'NodeColor', node);


% Perform K-means clustering on the second smallest eigenvector
k = 2;
clusters = kmeans(V(:, 2), k);


% Plot the K-means clustering
figure;
gscatter(V(:, 1), V(:, 2), clusters);
xlabel('X');
ylabel('Y');
title('K-means Clustering');


% Prepare node colors for plotting after K-means clustering
node_2 = zeros(10, 3);
node_2(clusters == 1, 1) = 1;
node_2(clusters == 2, 2) = 1;


% Plot the graph with colored nodes after K-means clustering
figure;
plot(g, 'NodeColor', node_2);
```

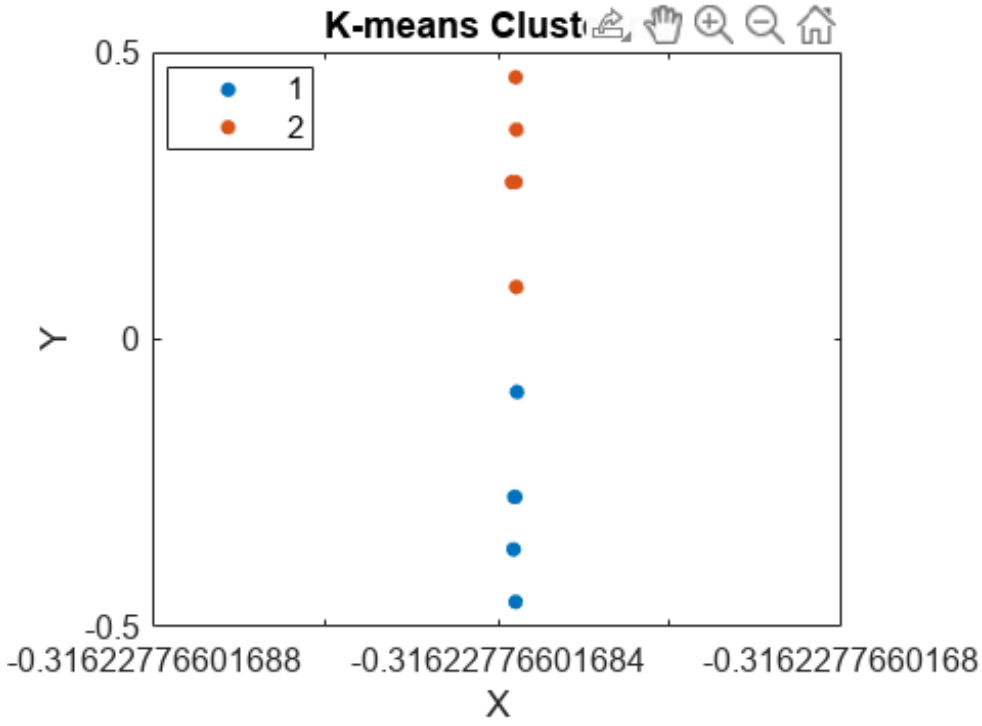
OUTPUT :

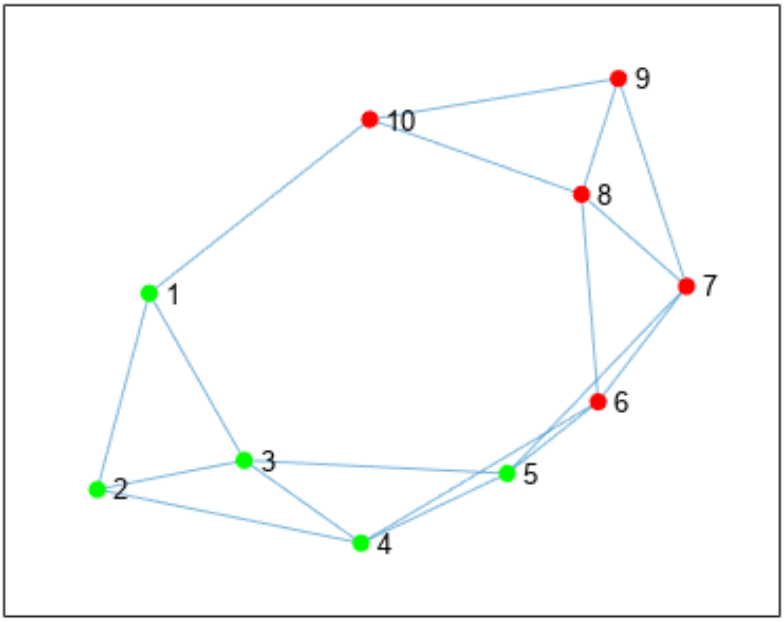
SPECTRUM CLUSTERING :





K-MEANS CLUSTERING :





THANK YOU !!