Simple Mathematical Demonstration (Toy Example)

```
clc; clear; close all;
rng(1);
```

Editable Parameters

```
numClusters = 3;
pointsPerCluster = 40;
k_nn = 6;
k_local = 7;
```

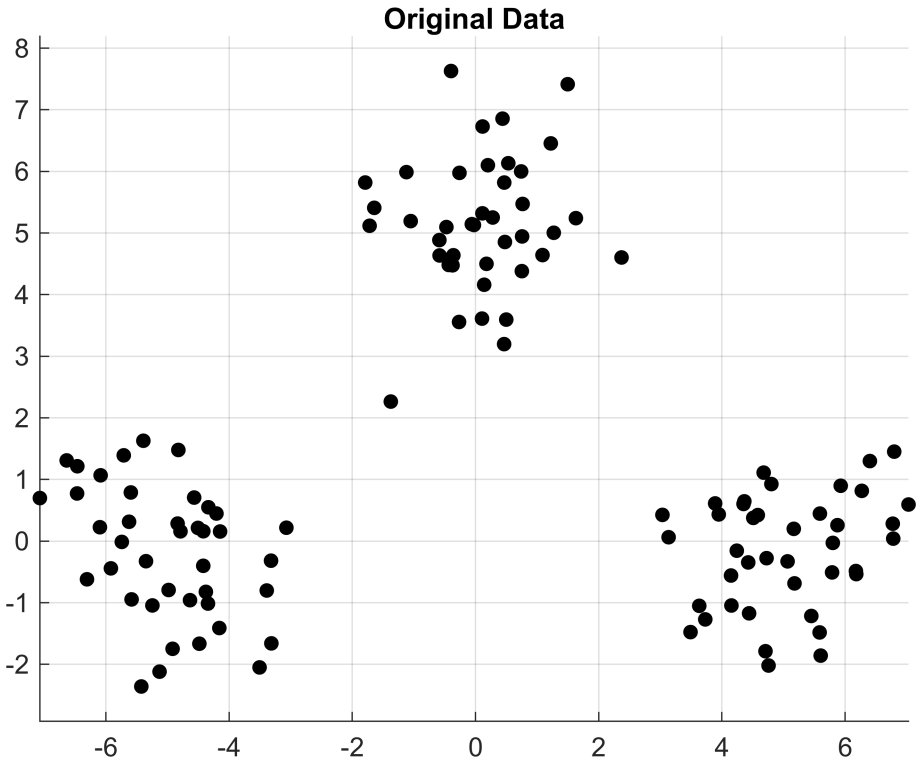
Variable	Use
`numClusters`	Number of clusters
`pointsPerCluster`	Points per cluster
`k_nn`	Nearest neighbors for graph
`k_local`	Local scale for self-tuning

Data Generation

```
C = [ 5  0;
     -5  0;
      0  5 ];

X = [ ...
    randn(pointsPerCluster,2) + C(1,:);
    randn(pointsPerCluster,2) + C(2,:);
    randn(pointsPerCluster,2) + C(3,:)];

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

- `C` contains **centers of clusters**
- `randn` generates Gaussian points
- Each group is shifted to its cluster center
- Final matrix `X` contains all data points

```
%% Plotting Original Data
figure;
scatter(X(:,1), X(:,2), 30, 'k', 'filled');
title('Original Data');
axis equal; grid on;
```




```
%% pairwsie distances between data points
D = squareform(pdist(X));
```

Explanation

- `pdist(X)` → computes Euclidean distances
- `squareform` → converts to NxN matrix
- `D(i,j)` = distance between point `i` and `j`

```
%% kNN CONNECTIVITY
[Ds, Idx] = sort(D,2);
```

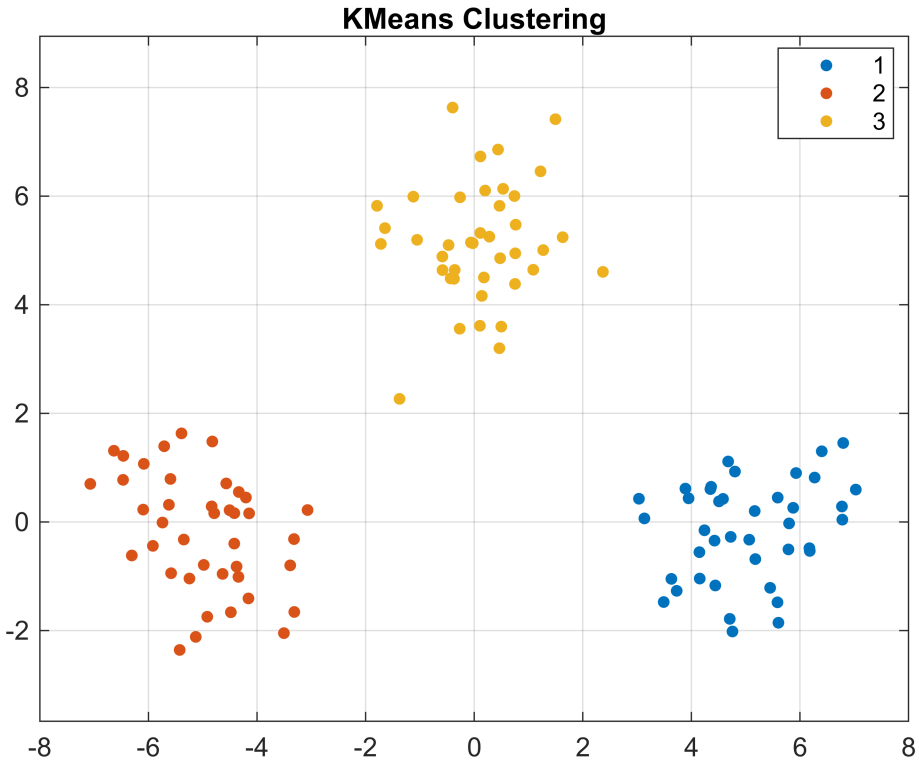
Sorts neighbors for each point

```
rows = repmat((1:N)', 1, k_nn);
cols = Idx(:, 2:k_nn+1);

Adj = false(N,N);
Adj(sub2ind([N N], rows(:), cols(:))) = true;
Adj = Adj | Adj';
```

- **Adjacency matrix**

- Each point connects ONLY to its **k nearest neighbors**
- Graph becomes:
- Sparse
- Local
- Robust to noise

```
%% 1) Kmeans clustering
idx1 = kmeans(X, numClusters);

figure;
gscatter(X(:,1), X(:,2), idx1);
title('KMeans Clustering');
axis equal; grid on;
```



- Clustering directly on data
- Uses Euclidean distance

```
%% 2) SPECTRAL (GLOBAL SIGMA)
sigma = median(D(:));

Wg = exp(-(D.^2)/(2*sigma^2)) .* Adj;
```

- Builds **weighted graph**
- Nearby points → strong edges
- Far points → weak/no edges

```
Lg = diag(sum(Wg,2)) - Wg;
```

- `sigma` = global scale
- Gaussian similarity function
- Similarity only where graph exists
- Laplacian captures graph structure

```
[Ug,~] = eigs(Lg, numClusters, 'smallestabs');
```

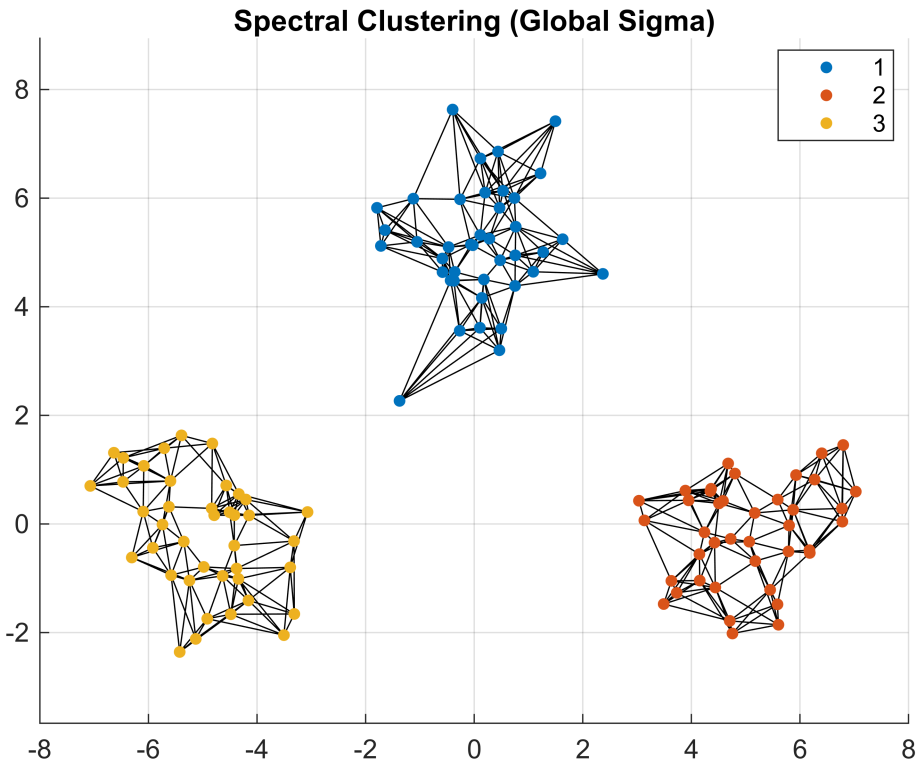
```
Warning: First input matrix is close to singular or badly scaled (RCOND =  1.602423e-20) and results may be
inaccurate. Consider specifying a small nonzero numeric sigma value instead of 'smallestabs' to improve the
condition of the matrix.
```

Finds **low-frequency eigenvectors**

```
idx2 = kmeans(Ug, numClusters);
```

Points in same cluster → similar eigenvectors

```
figure; hold on;
gplot(Wg, X, '-k');
gscatter(X(:,1), X(:,2), idx2);
title('Spectral Clustering (Global Sigma)');
axis equal; grid on;
```




```
%% 3) SPECTRAL (SELF-TUNING)
sigma_i = Ds(:, k_local+1);
```

- Each point gets its **own scale**
- Dense regions → small sigma
- Sparse regions → large sigma

```
Ws = exp(-(D.^2) ./ (sigma_i * sigma_i')) .* Adj;
```

- Similarity becomes **relative**, not absolute
- Adapts to:
- Density changes
- Irregular spacing

```
Ls = diag(sum(Ws,2)) - Ws;
```

New Laplacian based on adaptive graph

```
[Us,~] = eigs(Ls, numClusters, 'smallestabs');
```

```
Warning: First input matrix is close to singular or badly scaled (RCOND =  1.905671e-18) and results may be
inaccurate. Consider specifying a small nonzero numeric sigma value instead of 'smallestabs' to improve the
condition of the matrix.
```

```
idx3 = kmeans(Us, numClusters);

figure; hold on;
gplot(Ws, X, '-k');
gscatter(X(:,1), X(:,2), idx3);
title('Spectral Clustering (Self-Tuning)');
axis equal; grid on;
```