Community Detection using Spectral Decomposition

Introduction
Community detection is a fundamental task in network analysis that aims to identify groups of nodes within a network that exhibit higher connectivity among themselves than with the rest of the network. This report presents an implementation of community detection using spectral decomposition and the Louvain algorithm on two datasets: Facebook and Bitcoin.

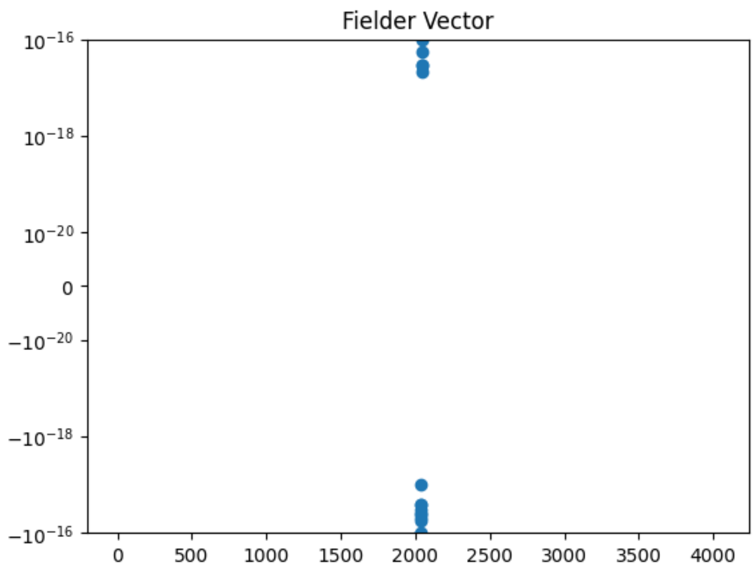
1. Dataset Import:
The code begins by importing the necessary libraries and defining functions to import the Facebook and Bitcoin datasets using the pandas library. These datasets contain information about node connectivity within the respective networks.

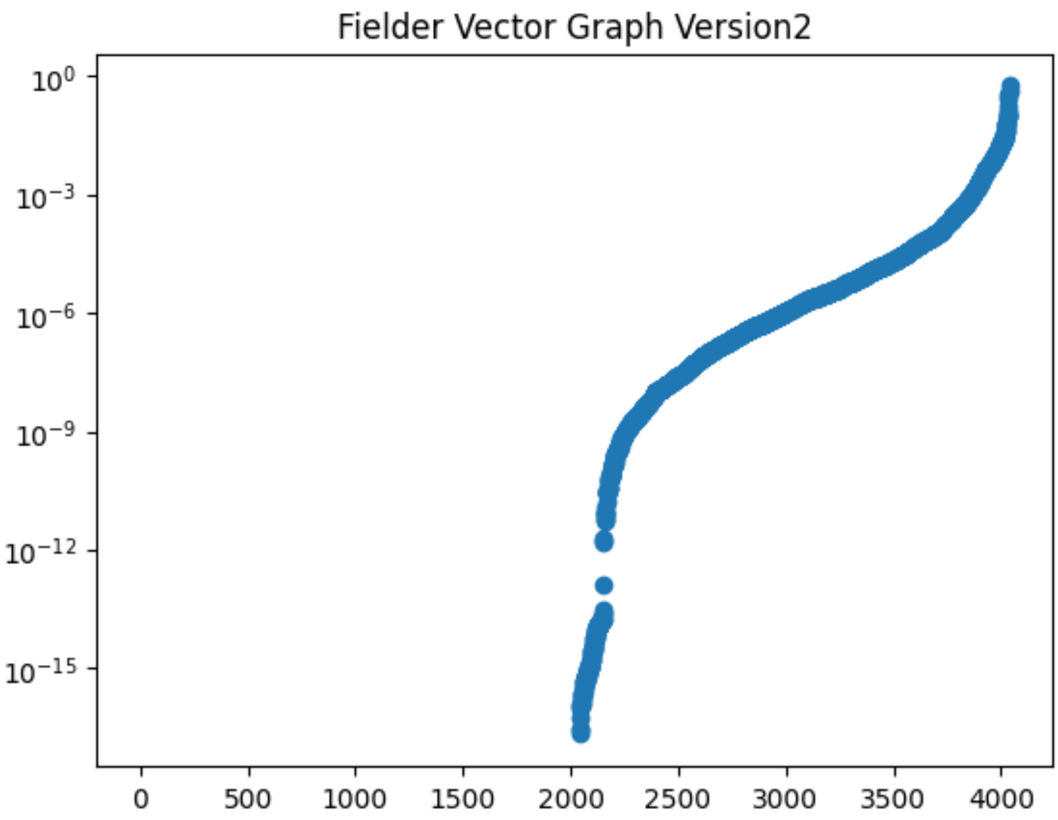
2. Spectral Decomposition:
The `spectralDecomp_OneIter` function is introduced, which performs spectral decomposition on a given dataset. It computes the adjacency matrix, Laplacian matrix, and eigenvalues/eigenvectors using NumPy linear algebra functions. The Fiedler vector, which captures the community structure, is extracted based on the sign of its elements. The function returns the Fiedler vector, adjacency matrix, graph partition, and a flag indicating the success of the community detection process.

4. Graph Visualization for Spectral decomposition:

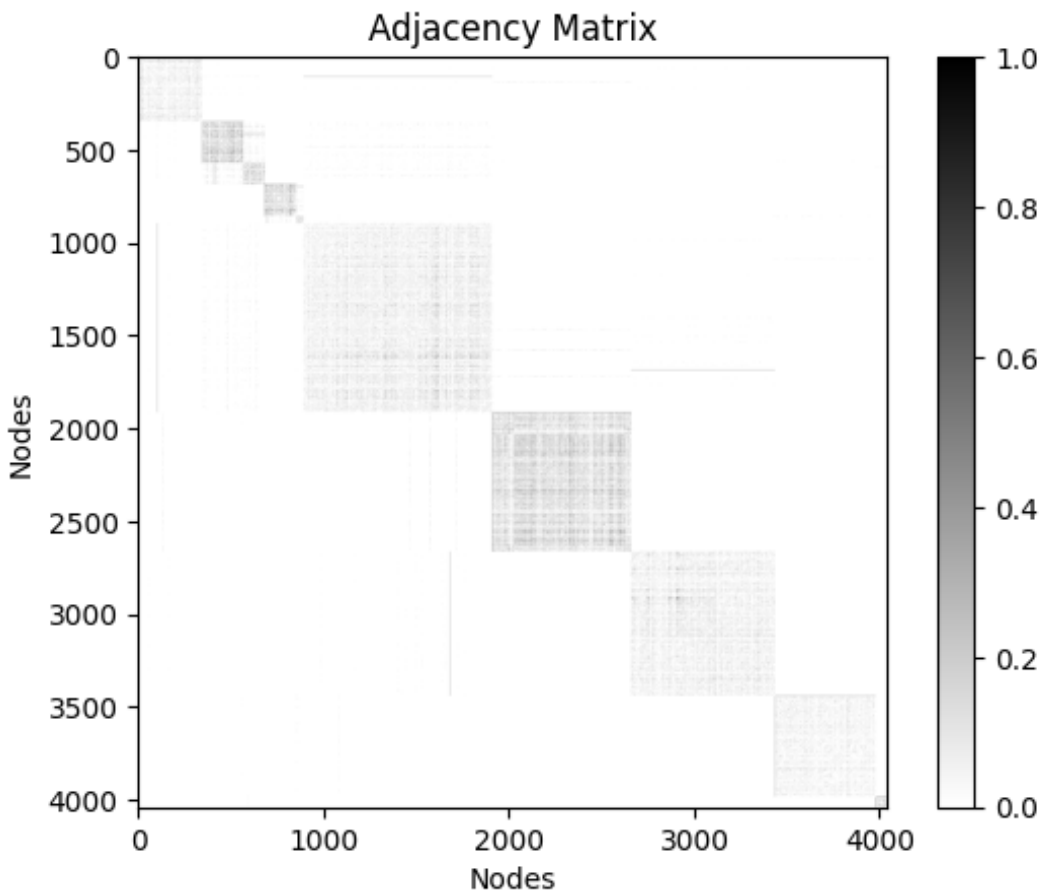
a. **Facebook dataset**

Plots for Q1





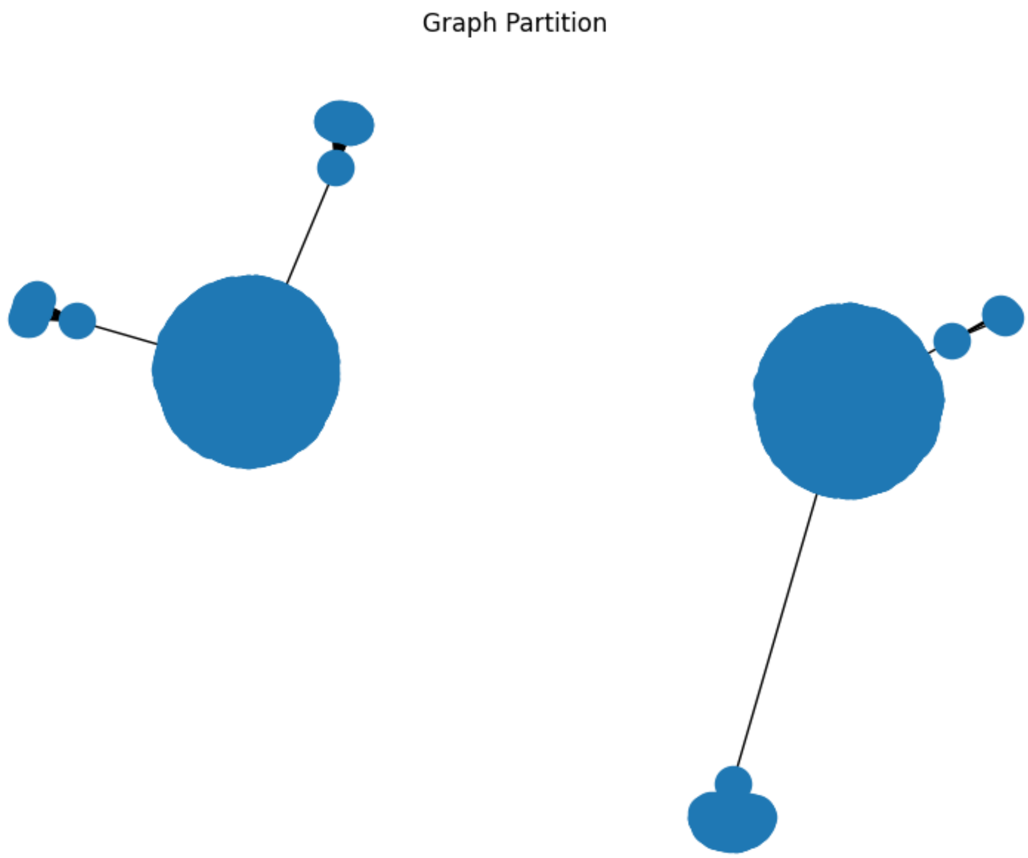
Fielder Vector Graph Version2
10^{0}
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
0
500
1000
1500
2000
2500
3000
3500
4000

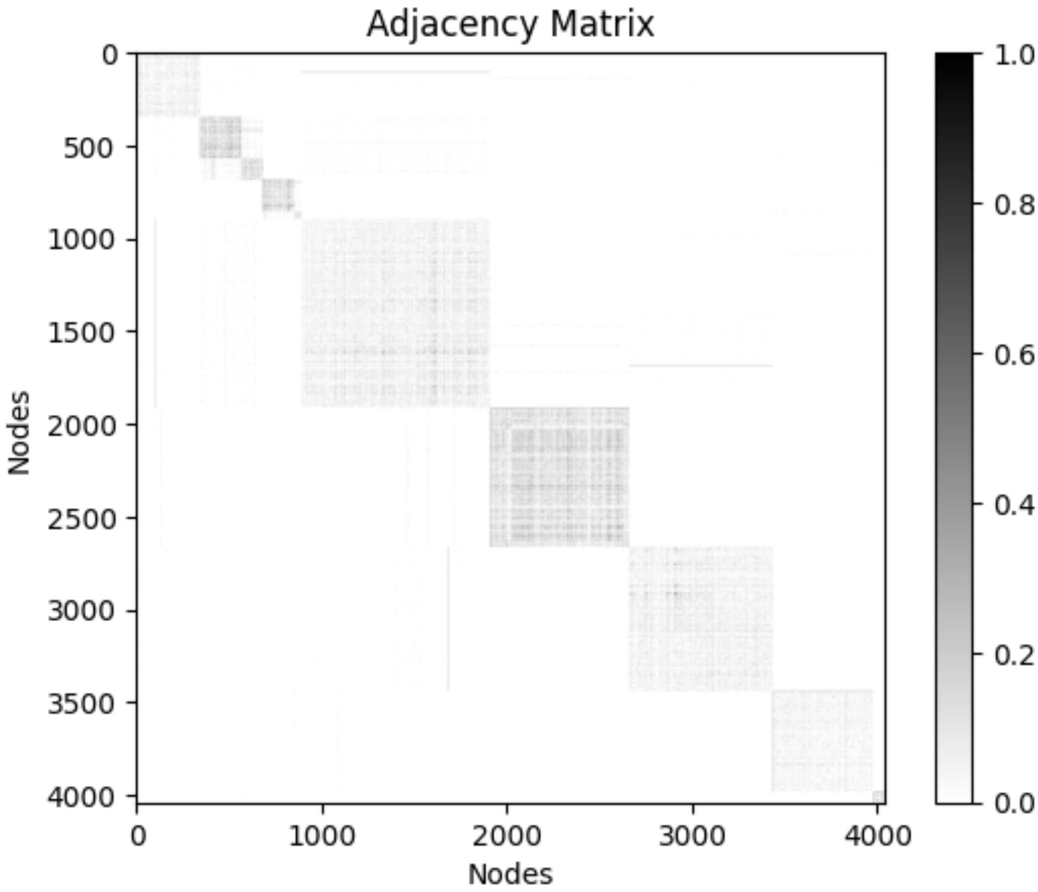


Adjacency Matrix
Nodes
Nodes
0
500
1000
1500
2000
2500
3000
3500
4000
0
1000
2000
3000
4000
1.0
0.8
0.6
0.4
0.2
0.0

Graph Partition

Plots for Q2 & Q3

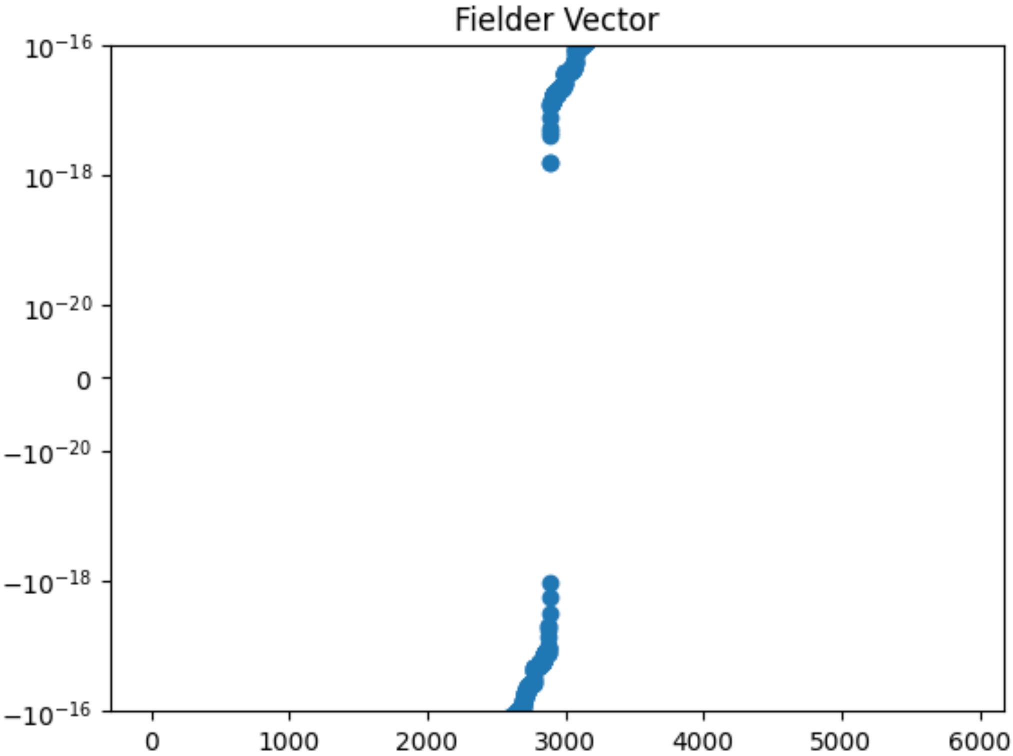


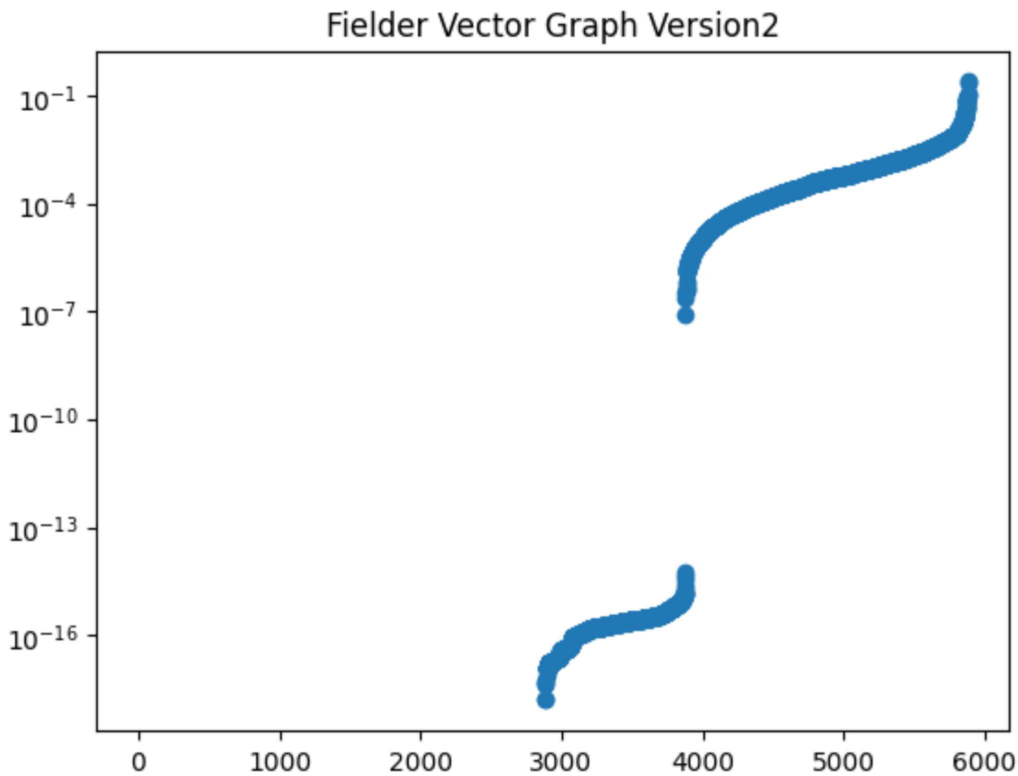


Adjacency Matrix
Nodes
Nodes
0
500
1000
1500
2000
2500
3000
3500
4000
0
1000
2000
3000
4000
1.0
0.8
0.6
0.4
0.2
0.0

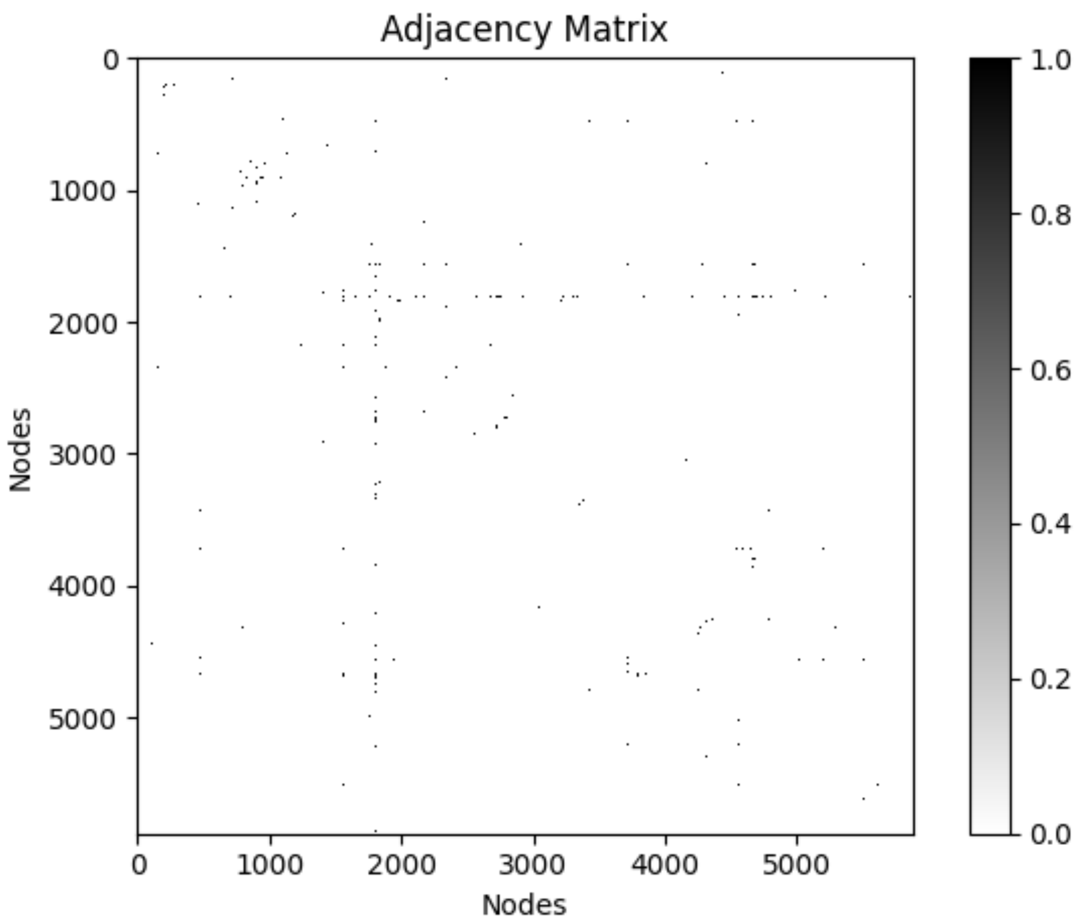
b. Bitcoin dataset

Plots for Q1



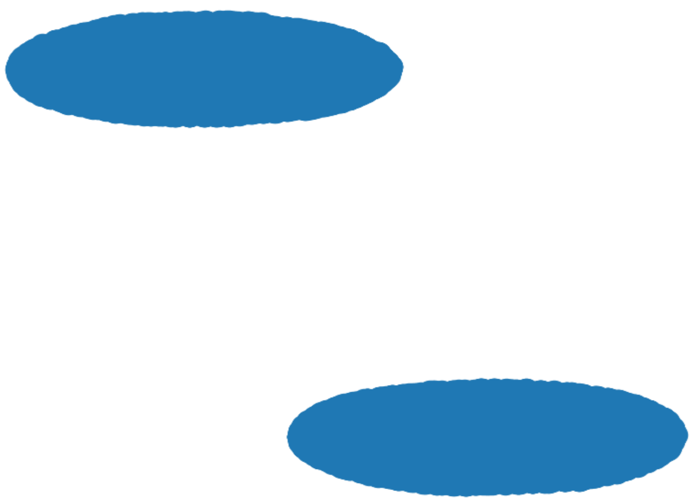


Fielder Vector Graph Version2
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-16}
0
1000
2000
3000
4000
5000
6000



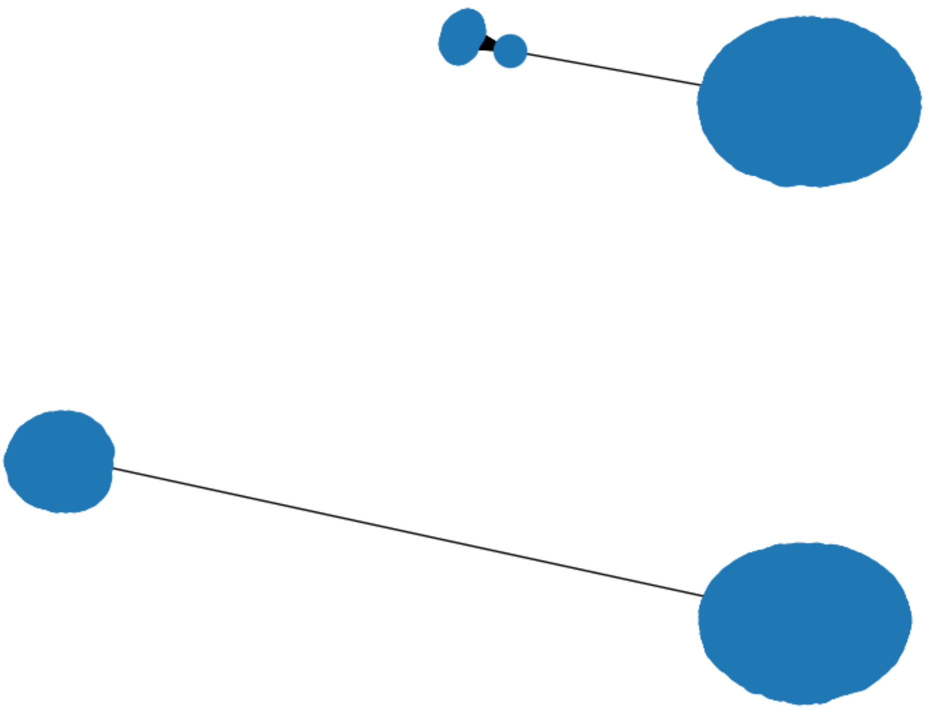
Adjacency Matrix
Nodes
Nodes
0
1000
2000
3000
4000
5000
1.0
0.8
0.6
0.4
0.2
0.0

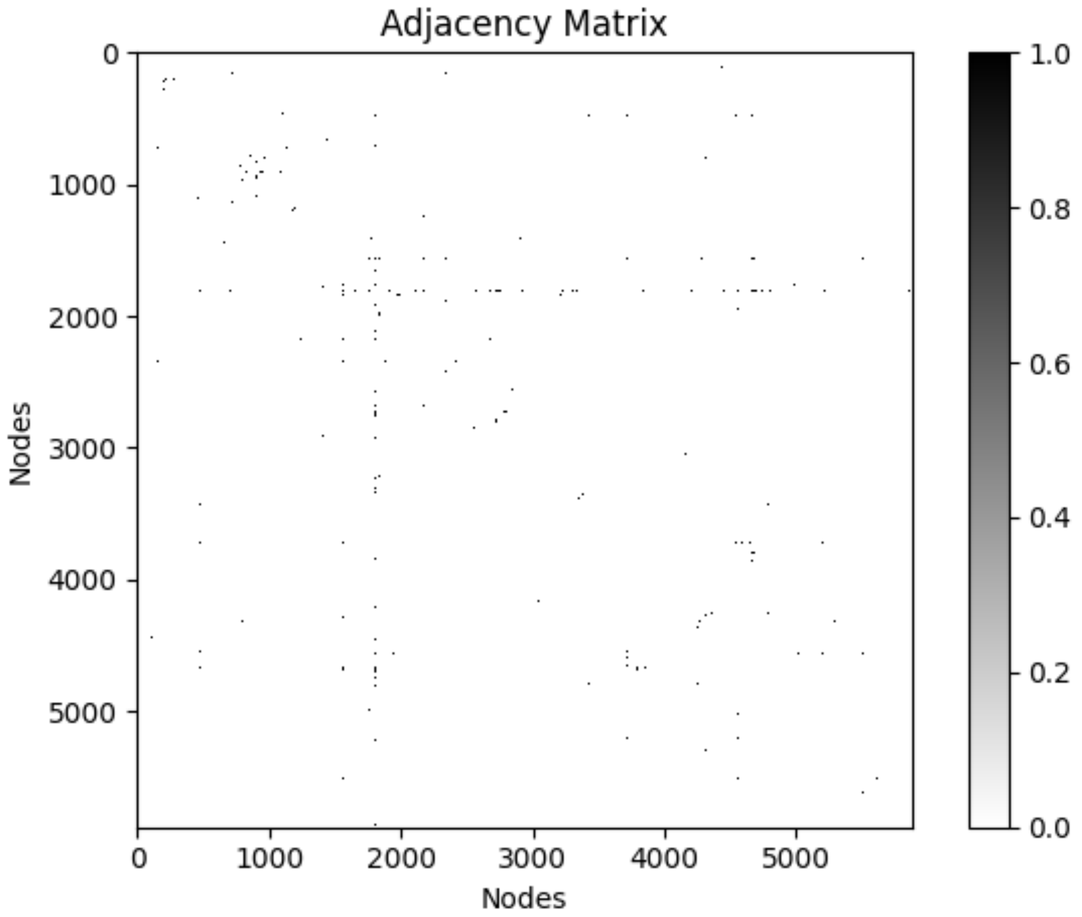
Graph Partition



Plots for Q2 & Q3

Graph Partition





5.

Iterative Community Detection:
Towards the end of the code, there is a while loop designed to iteratively perform community detection until no more communities can be formed. The loop starts with an empty `graph_tracker` list and appends the initial graph partition (with all nodes in a single community). Within the loop, spectral decomposition is performed for each community, and the graph partition is updated accordingly.

Louvain Algorithm for Community Detection in Network Analysis

1. Dataset Import:
The code begins by importing the required libraries, including NetworkX and NumPy. It then loads the Facebook network data from the "facebook_combined.txt" file using the `numpy.loadtxt` function. The data represents node connectivity within the network.

2. Graph Construction:
Next, the code creates a graph object using NetworkX and populates it with edges from the loaded data. Each edge represents a connection between two nodes in the Facebook network.

3. Louvain Algorithm:

The implementation proceeds to run one iteration of the Louvain algorithm, a two-phase community detection approach. In the first phase, each node is assigned to its own community. Then, in the second phase, the algorithm iteratively improves the modularity of the graph by moving nodes between communities.

4. Iterative Improvement:
The code employs a `for` loop to iterate a fixed number of times (in this case, once) to improve the modularity of the graph. Within the loop, the algorithm identifies the best community for each node based on modularity gain. It considers the number of within-community edges, between-community edges, and total edges to calculate the modularity. Nodes are assigned to the community that maximizes the modularity gain.

5. Graph Partition:
The resulting community assignments are stored in the `communities` dictionary, where each node is associated with its corresponding community. The code then extracts the graph partition by converting the `communities` dictionary into a NumPy array called `graph_partition`, which represents the node-to-community mapping.

Conclusion

Running time:
The running time of the Spectral decomposition algorithm and the Louvain algorithm depends on the size and complexity of the datasets. The Spectral decomposition algorithm has a higher computational cost due to the eigenvalue decomposition step. The Louvain algorithm, on the other hand, is more efficiency and can handle large-scale networks more efficiently.

Better communities:
Spectral decomposition algorithm gives better communities.