Università della Svizzera italiana
Faculty of Informatics
Institute of Computing CI

Numerical Computing **2022**

Student: Kelvin Likollari Discussed with: Gerald Prendi

Solution for Project 2

Due date: Wednesday, 26 October 2022, 23:59 AM

Numerical Computing 2022 — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, Julia). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

The purpose of this assignment[1] is to learn the importance of sparse linear algebra algorithms to solve fundamental questions in social network analyses. We will use the coauthor graph from the Householder Meeting and the social network of friendships from Zachary's karate club [1]. These two graphs are one of the first examples where matrix methods were used in computational social network analyses.

Social Networks [Total: 85 points + 15 points for report quality]

1. The Reverse Cuthill McKee Ordering [10 points]

The following image depicts the visualization of the Original Matrix

[1]This document is originally based on a blog from Cleve Moler, who wrote a fantastic blog post about the Lake Arrowhead graph, and John Gilbert, who initially created the coauthor graph from the 1993 Householder Meeting. You can find more information at `http://blogs.mathworks.com/cleve/2013/06/10/lake-arrowhead-coauthor-graph/`. Most of this assignment is derived from this archived work.

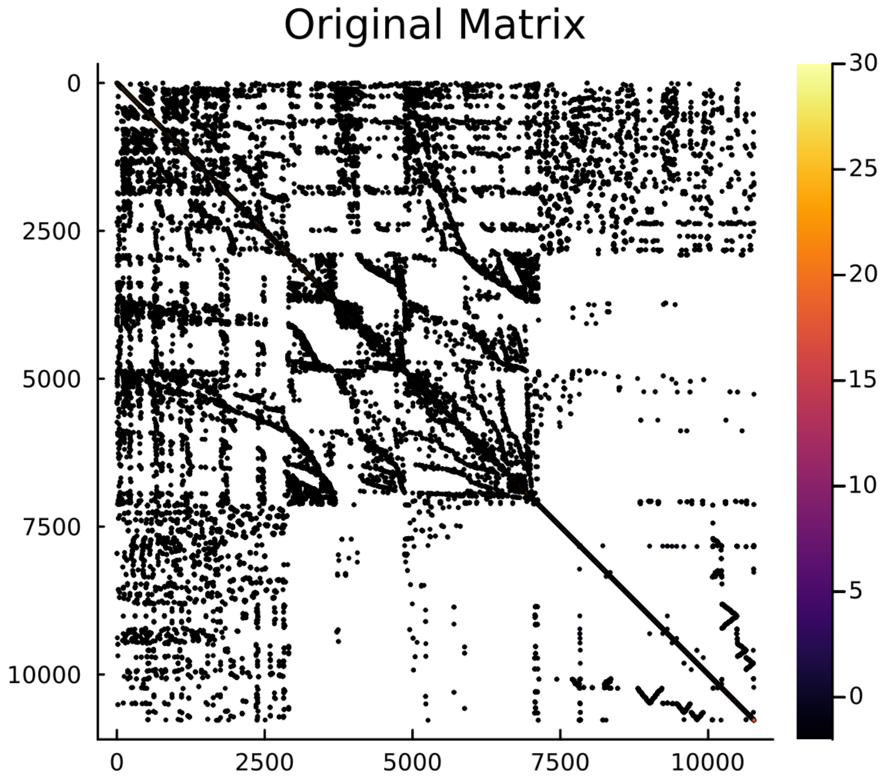


Figure 1: Original Matrix Visualization

In the image depicted above, the intensity of zeros is denser in the higher, upper left, and center-left, and sparser in the bottom left. The sections in the center-right, bottom center, and lower right, on the other hand, are less densely populated with zeros. As the textbook states, *the effectiveness of Gaussian elimination may strongly depend on the sparsity pattern of the matrix.* Effectively, one can claim that the Gaussian Elimination procedure is responsible for the sparsity of the matrix in the aforementioned areas. The textbook mentions various ways to make the Gaussian Elimination procedure for more general sparse as effective as possible by using permutation and reordering strategies, which typically include the **Matrix Graph**, **Two Vertex labeling strategies**, such as **the Reverse Cuthill McKee (RCM)** algorithm.

For the purposes of this exercise, the RCM algorithm will be employed. RCM is simply an efficient technique that reduces bandwidth, and the **Approximate Minimum Degree (AMD)** algorithm serves in minimizing the predicted fill-in, as depicted in the graph above. The results of applying the RCM algorithm can be observed in the image below.

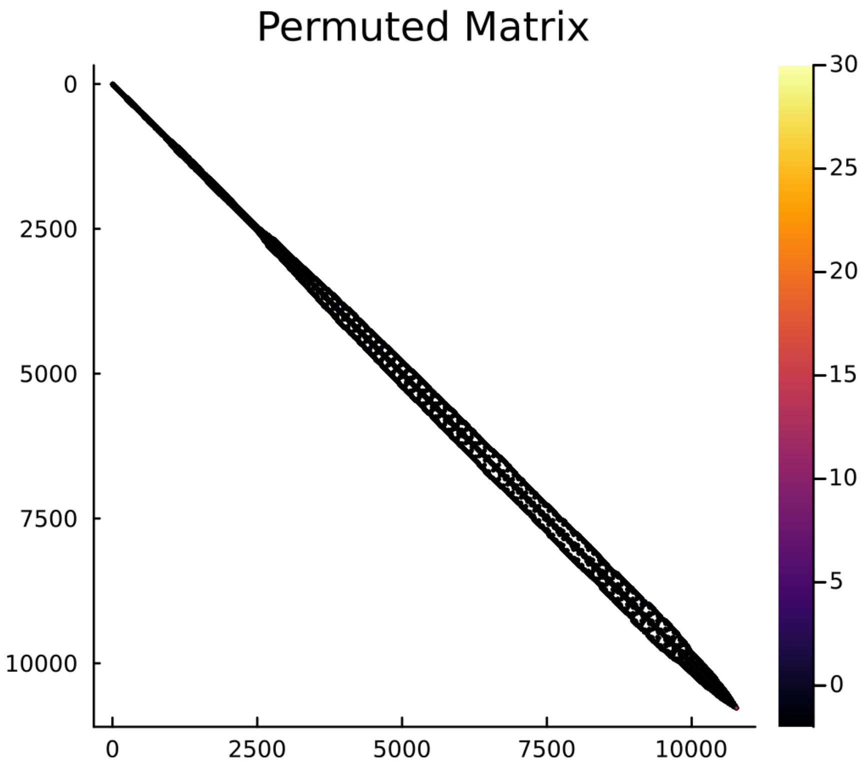


Figure 2: Permuted Matrix Visualization

Having applied the RCM algorithm and viewing the image above, it is evident that the density of the Matrix (i.e. the fill-in) has been greatly reduced. One can now confidently state that the pattern is more appealing and that the bandwidth has been greatly reduced, lowering the costs of matrix operations substantially. The matrix fill-in is effectively projected onto a diagonal line spanning the whole Permuted Matrix, with some alternations in terms of density.

The Cholesky Factors of both the Original Matrix A and the Permuted Matrix are visualized below.

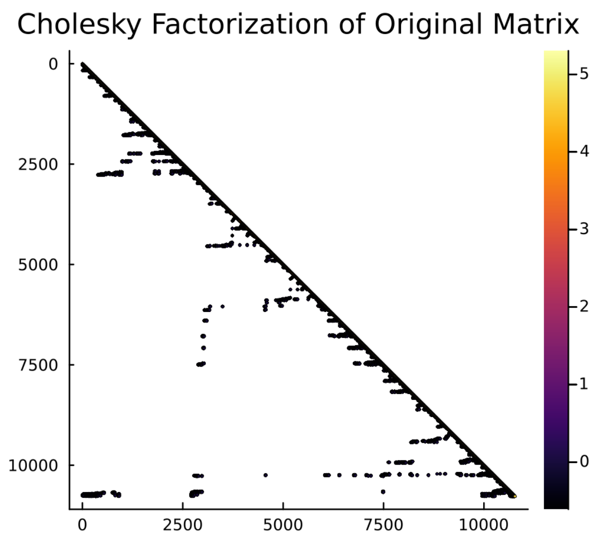


Figure 3: Cholesky Factor Visualization

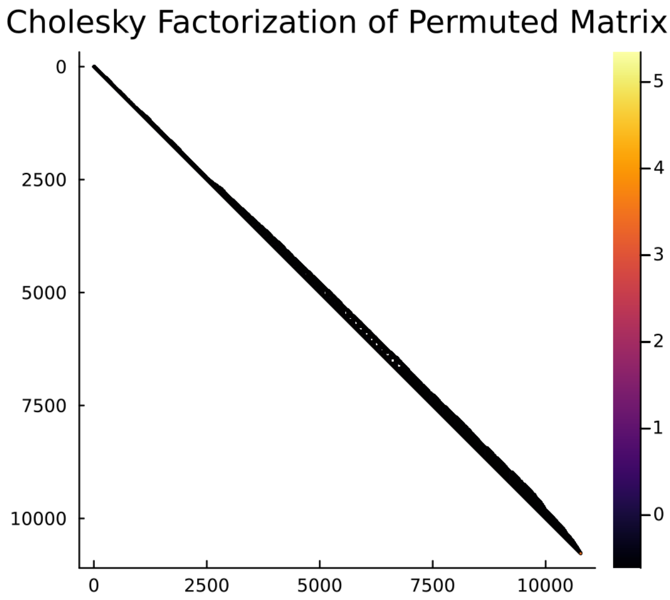


Figure 4: Cholesky Factorization of Permuted Matrix

One can observe that the fill-in has been significantly decreased, in both the original matrix and the permuted one, thereby yielding the fact that computations can be performed faster as the vast majority of the surface of the visualized matrix is not occupied, thereby increasing efficiency.

2. Sparse Matrix Factorization [20 points]

(a)

$$A = \begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 17 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 19 \end{pmatrix}$$

From the above matrix one can observe that along the diagonal, where i = j, the second condition is applied, in which it holds $n + i - 1$.
For example, in (i,j) = (1,1) → 10 + 1 - 1 = 10. Similar procedure is applied on the rest of the diagonal.
The above depicted matrix is a sparse matrix as most of its elements are 0s and at the same time one can observe that another matrix property holds. Specifically, if one takes any arbitrary element in the diagonal, it should be visible that the sum of the rest n - 1 elements of that row, still do not add up to d_{ij} for i = j. According to Wikipedia, this Matrix can also be called a Diagonally Dominant Matrix.

(b) Looking at the Matrix depicted above, a general formula can be inherited to compute the number of non-zero entries. Specifically, a formula used to calculate the number of non-zero elements in a Matrix A is:

(i) Taking the diagonal of A, then the **nnz(diag(A))** $\rightarrow$ **n**

(ii) Taking the position where col = 1, then **nnz(col(j = 1))** $\rightarrow$ **n - 1**. Effectively the -1 is derived because the whole column except the first element was taken, which was already counted in (i) and thus there is no need to count it again.

(iii) Taking the position where col = n, then **nnz(col(j = n))** $\rightarrow$ **n - 1**, as user is counting the whole column except the last element of column n which is also a part of the diagonal, but unlike before, now it is located at the bottom right of Matrix A.

(iv) If user is at row = 1, then **nnz(row(i = 1))** $\rightarrow$ **n - 2**.

(iv) If user is at row = n, then **nnz(row(i = n))** $\rightarrow$ **n - 2**.

Effectively, adding the results of the above 5 items, yields:

$$\mathbf{n + (n - 1) + (n - 1) + (n - 2) + (n - 2) = 5n - 6}$$

(c) The following is the nonzero structure of matrix A

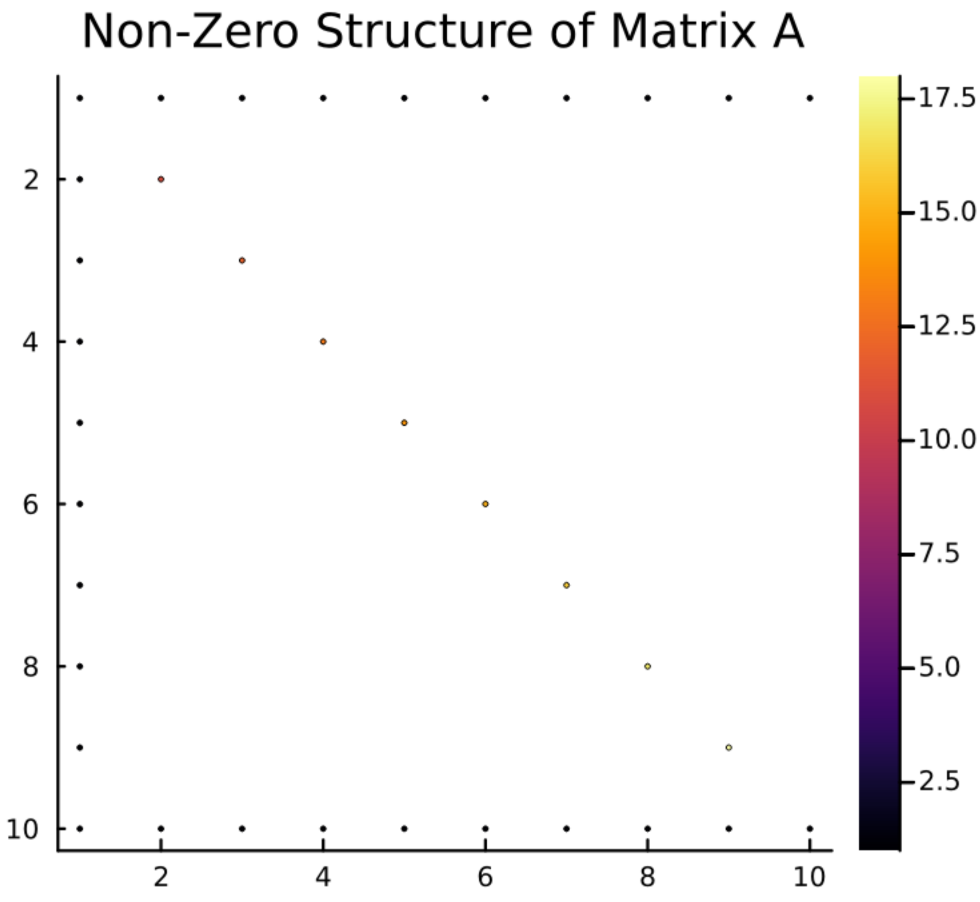
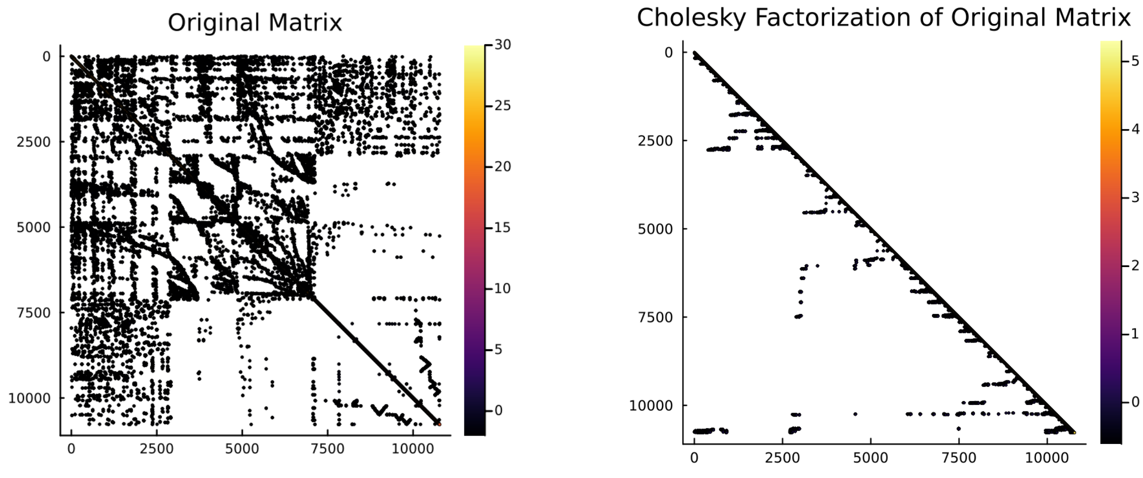


Figure 5: Non-Zero Structure of Matrix A

(d) The following two images depict the visualization of the original matrix A and the Cholesky Factorization. The plots have been generated using the spy() command in Julia.



(a) Original Matrix

(b) Cholesky Factorization of Matrix A

Figure 6: Original Matrix vs Cholesky Factorization of Original Matrix

(e) Taking n = 100.000, and using it to solve Ax = b for a given right hand-side vector b would be problematic. This comes from the fact that increasing the number of n, will inevitably increase the fill-in of the Matrix. This problem can be alleviated with the help of Permutation and Ordering strategies. Effectively, Two vertex labeling algorithms such as the reverse Cuthill McKee (RCM) algorithm would benefit users in a significant factor, as the bandwidth of the Matrix is going to be reduced as well as the expected fill-in in the decomposition stage.

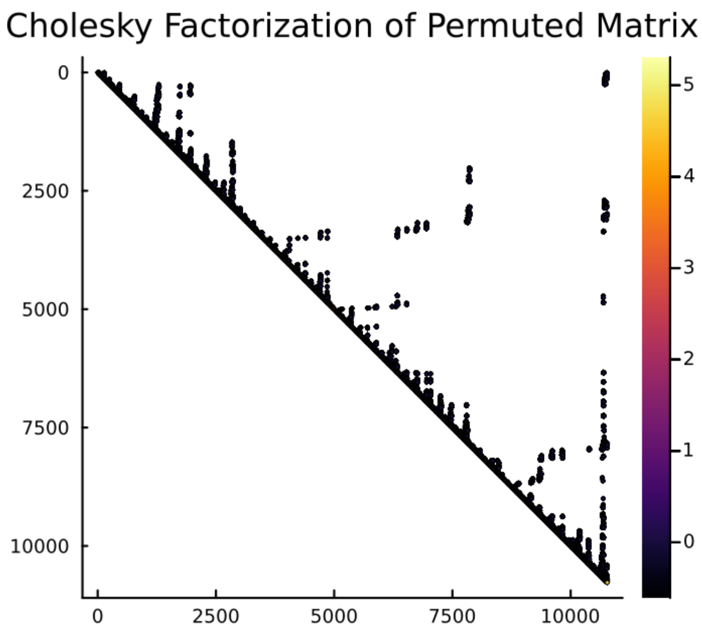


Figure 7: Cholesky Factorization applied on Permuted Matrix A

As it evident from the figure above, the fill-in has been significantly reduced, as well as the bandwidth, thereby resulting the conclusion that Cholesky Factorization on a Permuted Matrix A is optimal, in terms of performance and visualization

3. Degree Centrality [5 points]

The following section contains the degree centrality for the top 5 authors, the coauthors and the degree centrality. The degree, and the number of co-authors decrease as the user progresses down the rank.

(1) Author: Golub → Wilkinson, TChan, Varah, Overton, Ernst, VanLoan, Saunders, Bojanczyk, Dubrulle, George, Nachtigal, Kahan, Varga, Kagstrom, Widlund, OLeary, Bjorck, Eisenstat, Zha, VanDooren, Tang, Reichel, Luk, Fischer, Gutknecht, Heath, Plemmons, Berry, Sameh, Meyer → Degree: 16.

(2) Author: Demmel → Edelman, VanLoan, Bai, Schreiber, Kahan, Kagstrom, Barlow, NHigham, Arioli, Duff, Hammarling, Bunch, Heath, Greenbaum, Gragg → Degree: 15

(3) Author: Plemmons → Golub, Nagy, Harrod, Pan, Funderlic, Bojanczyk, George, Barlow, Heath, Berry, Sameh, Meyer, Nichols → Degree: 14

(4) Author: Schreiber → TChan, VanLoan, Moler, Gilbert, Pothen, NTrefethen, Bjorstad, NHigham, Eisenstat, Tang, Elden, Demmel → Degree: 13

(5) Author: Heath → Golub, TChan, Funderlic, George, Gilbert, Eisenstat, Ng, Liu, Laub, Plemmons, Paige, Demmel → Degree: 12

4. The Connectivity of the Coauthors [5 points]

To find the coauthor that the authors have in common, effectively means to try and locate the point in which the edges of 2 authors cross. If they cross, this effectively yields the scenario that the authors have a common coauthor. In matrix notation, one has to try to locate a point in the matrix M, say M(i, j) and check if M[i, j] == 1 and M[j, i] == 1. If this holds, then a shared co-author is detected.

Golub, Moler → Wilkinson, VanLoan
Golub, Saunders → Gil
TChan, Demmel → Schreiber, Arioli, Duff, Heath

Table 1: Common Co-Authors Table

5. PageRank of the Coauthor Graph [5 points]

The PageRank visualization for all the authors is visualized below.

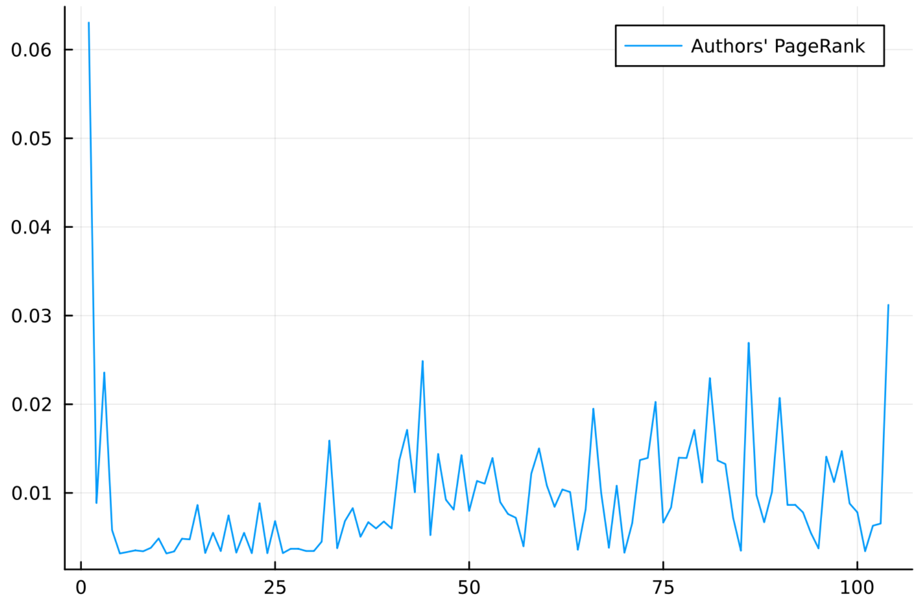


Figure 8: Authors' PageRank

After having sorted the authors in descending order, the new PageRank visualization has significantly changed, as seen by the image below.

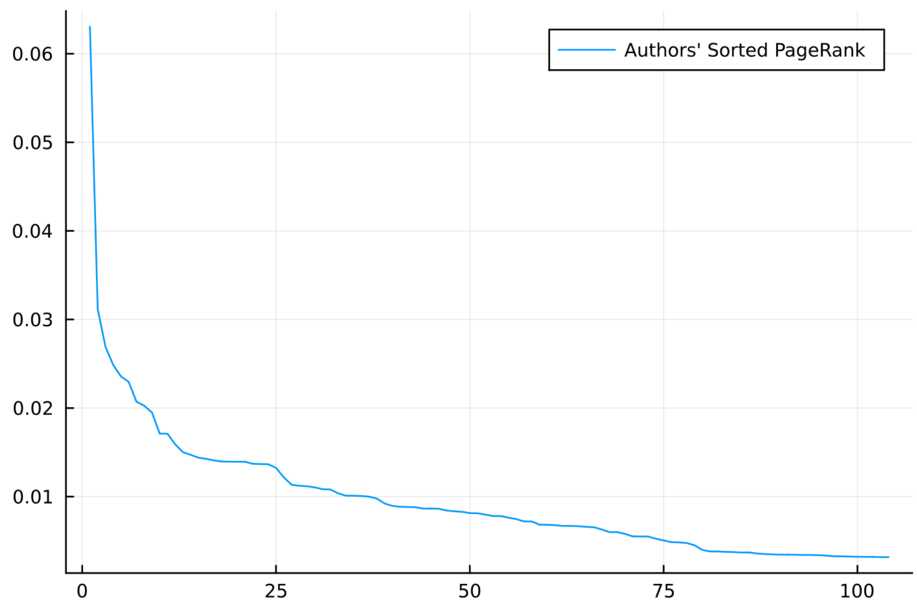


Figure 9: Authors' PageRank

6. Zachary's karate club: social network of friendships between 34 members [40 points]

(a) The following snippet of code ranks the five nodes with the largest degree centrality.

```
A = zeros(1,34);
for i in 1:34
```

```
        for j in 1:34
            if lines[i,j] == 1
                A[i] += 1
            end
        end
    end

    deg = sum(A, dims=1)
    i = sortperm(vec(deg), rev=true)
    top5 = i[1:5]
    println("Top 5 nodes and their degree: ", top5, " ", deg[top5])
```

The above snippet of code produces the following tabular view of Nodes and their equivalent Degrees:

Nodes	Degrees
34	17.0
1	16.0
33	12.0
3	10.0
2	9.0

Table 2: Top-5 Nodes with highest degree cardinality

(b) To display the 5 nodes with the largest eigenvector centrality, the PageRank algorithm should be employed. Using the PageRank algorithm, the results for the largest eigenvector centrality and the properly normalized centralities will look like:

34	0.10091
1	0.09699
33	0.07169
3	0.05707
2	0.05287

Table 3: Sorted Pagerank results

34	0.09699
1	0.052876
33	0.05707
3	0.03585
2	0.02197

Table 4: Properly Normalized Eigenvector Centralities

(c) The rankings in (a) and (b) seem to be identical. This happens because ranking a set of nodes with respect to the largest degree centrality and the largest eigenvector centrality is effectively the same thing. No changes to the ranking results should be expected.

(d) Having used spectral graph partitioning to find a near-optimal split such as to divide the network into two groups of 16 and 18 nodes each, one can notice the following results:

Greater than treshold	Lower than treshold
3	1
5	2
6	4
7	8
11	9
12	10
13	14
15	17
16	25
18	26
19	27
20	28
21	29
22	30
23	31
24	32
-	33
-	34

Table 5: Spectral Graph Partitioning: Network Split

The treshold has been decided to be the median of the numbers in the set [1, 34].

References

[1] The social network of a karate club at a US university, M. E. J. Newman and M. Girvan, Phys. Rev. E 69,026113 (2004) pp. 219-229.