Università della Svizzera italiana | Institute of Computing CI

Numerical Computing **2024**

Student: POLAD BAKHISHZADE

Solution for Project 2 **Due date:** Wednesday, 23 October 2024, 11:59 PM

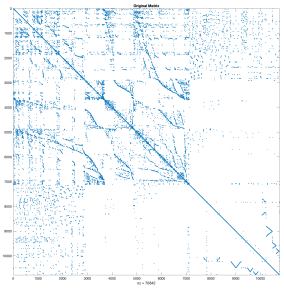
---

1. The reverse Cuthill–McKee ordering [10 points]

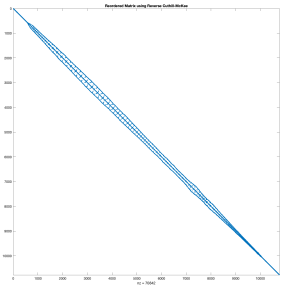
The matrix is loaded from the provided dataset and permuted using the RCM ordering via the MATLAB function `symrcm`.

1.1. Original and Reordered Matrix

Figure 1a shows the sparsity pattern of the original matrix, while Figure 1b shows the reordered matrix using the RCM ordering.



(a) Original Matrix

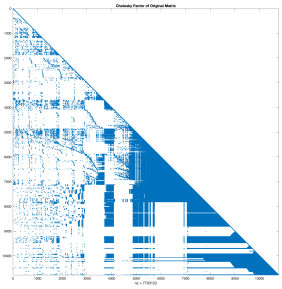


(b) Reordered Matrix Reverse Cuthill-McKee

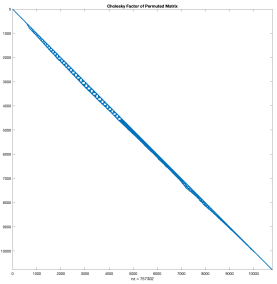
In the original matrix, the sparsity pattern is irregular, while the RCM-permuted matrix reveals a more compact and banded structure. This reordering minimizes the bandwidth and brings the non-zero elements closer to the diagonal.

1.2. Cholesky Factorization

The Cholesky factorizations of both the original and permuted matrices were computed and visualized. Figures 2a and 2b depict the sparsity patterns of the Cholesky factors.



(a) Cholesky Factor of Original Matrix



(b) Cholesky Factor of Permuted Matrix

1.3. Non-zero Comparison

The number of non-zeros in the Cholesky factor of the original matrix was 7,703,122, whereas the number of non-zeros in the Cholesky factor of the permuted matrix was 757,302. This substantial reduction in non-zero elements highlights the efficiency of the RCM ordering in reducing fill-in during Cholesky factorization.

2. Sparse matrix factorization [20 points]

2.1. Constructing Matrix A for $n = 10$

In this task, we construct the matrix $A \in \mathbb{R}^{n \times n}$ for $n = 10$, defined as follows:

- $A_{ij} = 1$ if $i = 1$ or $i = n$, or $j = 1$ or $j = n$, and $i \neq j$.
- $A_{ii} = n + i - 1$ for all diagonal elements.
- $A_{ij} = 0$ otherwise.

The matrix entries for $n = 10$ are:

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

We have successfully constructed the matrix A for $n = 10$, and it contains 44 non-zero elements. The diagonal elements increase progressively, ensuring the matrix is positive definite.

2.2. Formula to Compute the Number of Non-Zero Elements

To derive the number of non-zero entries nz:

- There are n diagonal entries (one for each i).
- The first and last rows contribute $2(n - 1)$ (all entries except diagonal).
- The first and last columns contribute another $2(n - 1)$.
- We subtract 2 for the double counting of A_{11} and A_{nn}.

Combining these counts, we get:

$$nz = n + 2(n-1) + 2(n-1) - 2 = 5n - 6$$

Thus, the total number of non-zero elements in matrix A is:

$$\boxed{5n - 6}$$

2.3. Write a Function `A_construct(n)`

Using the function `A_construct(n)`, we constructed the matrix for $n = 10$ as follows:

```
function [A, nz] = A_construct(n)
    A = zeros(n);
    nz = 0; % non-zero counter

    % Populate matrix A
    for i = 1:n
        for j = 1:n
            if i == j
                A(i, j) = n + i - 1; % Diagonal elements
                nz = nz + 1;
            elseif (i == 1 || i == n || j == 1 || j == n)
                A(i, j) = 1; % Boundary elements
                nz = nz + 1;
            end
        end
    end
end
```

The resulting matrix A is:

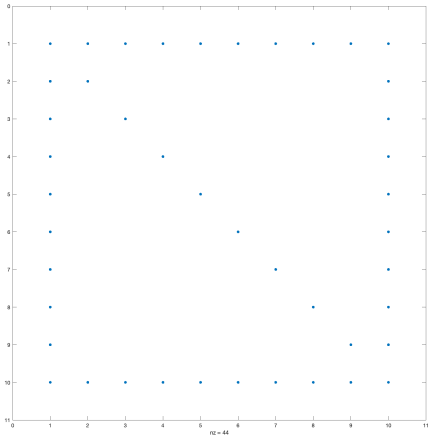
$$A = \begin{bmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 16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

The number of non-zero elements is calculated to be $nz = 44$.

Comparing the output from the function with the manually constructed matrix, we find that both methods yield the same matrix A and the same count of non-zero elements:

- Matrix A: Same entries as in point (a).
- Number of non-zero elements: 44.

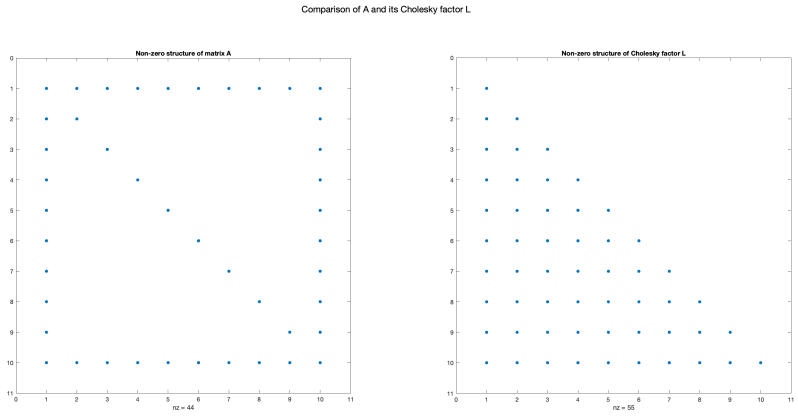
The plot below demonstrates the non-zero structure of the matrix, confirming the placement of the boundary and diagonal elements.



Visualization of the non-zero structure of matrix A using the `spy()` command.

2.4. Visualization of Matrix A and Cholesky Factorization

Using the MATLAB `spy()` command, I visualized side by side the non-zero structure of matrix A and its Cholesky factor L. The following figure illustrates the results:



Comparison of A and its Cholesky factor L

The comparison highlights that matrix A has a dense structure, whereas the Cholesky factor L retains a simpler, lower triangular form. This reduction is beneficial for computational efficiency when solving systems of equations.

2.5. Challenges of Using `chol()` for Large Matrices

Using `chol()` to solve $Ax = b$ for $n = 100,000$ could cause significant challenges such as:

- **Memory Usage:** Storing a dense $100,000 \times 100,000$ matrix requires approximately 80 GB, exceeding typical memory limits.
- **Computational Cost:** The $O(n^3)$ complexity of Cholesky decomposition results in excessive computation time for large matrices.
- **Numerical Instability:** Large matrices can lead to numerical instability, particularly if A is ill-conditioned.

To tackle this issue, consider the following approaches:

- **Use more efficient algorithms:** Instead of relying on Cholesky directly, you might try iterative methods that can handle large matrices without needing to compute everything at once.
- **Simplify the matrix:** If possible, reducing the matrix size or employing techniques like sparse matrix representations can help.

3. Degree centrality [5 points]

Below are the top 5 authors in the Householder coauthor network, along with their coauthors and degree centrality:

1. **Author: Golub**
   Degree Centrality: 32
   Coauthors: 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, Gill
2. **Author: Demmel**
   Degree Centrality: 16
   Coauthors: Edelman, VanLoan, Bai, Schreiber, Kahan, Kagstrom, Barlow, NHigham, Arioli, Duff, Hammarling, Bunch, Heath, Greenbaum, Gragg
3. **Author: Plemmons**
   Degree Centrality: 14
   Coauthors: Golub, Nagy, Harrod, Pan, Funderlic, Bojanczyk, George, Barlow, Heath, Berry, Sameh, Meyer, Nichols
4. **Author: Schreiber**
   Degree Centrality: 13
   Coauthors: TChan, VanLoan, Moler, Gilbert, Pothen, NTrefethen, Bjorstad, NHigham, Eisenstat, Tang, Elden, Demmel
5. **Author: Heath**
   Degree Centrality: 13
   Coauthors: Golub, TChan, Funderlic, George, Gilbert, Eisenstat, Ng, Liu, Laub, Plemmons, Paige, Demmel

The degree centrality of the top authors reveals the key figures in the Householder coauthor network, with Golub having the highest number of coauthor connections. These results provide insight into the central figures in this academic network.

4. The connectivity of the coauthors [10 points]

The following procedure was used to compute the common coauthors. The coauthor relationships are stored in an adjacency matrix A, where $A(i, j) = 1$ if authors i and j are coauthors. Common coauthors between authors i and j are found by performing a logical AND between the i-th and j-th rows of A:

$$C_{i,j} = A(i,:) \wedge A(j,:)$$

Non-zero entries in $C_{i,j}$ indicate the common coauthors.

The following function was used to compute the common coauthors:

```
function common_coauthors = find_common_coauthors(A, author1, author2)
    common_coauthors_mask = A(author1, :) & A(author2, :);
    common_coauthors = find(common_coauthors_mask);
end
```

To find the indices of the authors in the matrix, I used:

```
golub = find(strcmp(strtrim(full_name), 'Golub'));
moler = find(strcmp(strtrim(full_name), 'Moler'));
saunders = find(strcmp(strtrim(full_name), 'Saunders'));
tchan = find(strcmp(strtrim(full_name), 'TChan'));
demmel = find(strcmp(strtrim(full_name), 'Demmel'));
```

The following common coauthors were computed for the given pairs:

- **Golub and Moler**: 2 common coauthors
  (Wilkinson, VanLoan)
- **Golub and Saunders**: 3 common coauthors
  (Golub, Saunders, Gill)
- **TChan and Demmel**: 4 common coauthors
  (Schreiber, Arioli, Duff, Heath)

The adjacency matrix allowed us to efficiently compute common coauthors between the specified author pairs. A logical AND operation on the matrix rows was applied to find common coauthors, as demonstrated by the provided code snippets.

5. PageRank of the coauthor graph [5 points]

In this task, we compute the PageRank values for all authors in the Householder coauthor network using the modified PageRank algorithm. The PageRank values are then sorted in descending order, and a bar graph is plotted to show the ranking of the authors.

The modified 'pagerank1.m' function from Project 1 was used to compute the PageRank values for all authors. The damping factor $p = 0.85$ was used for the calculation.

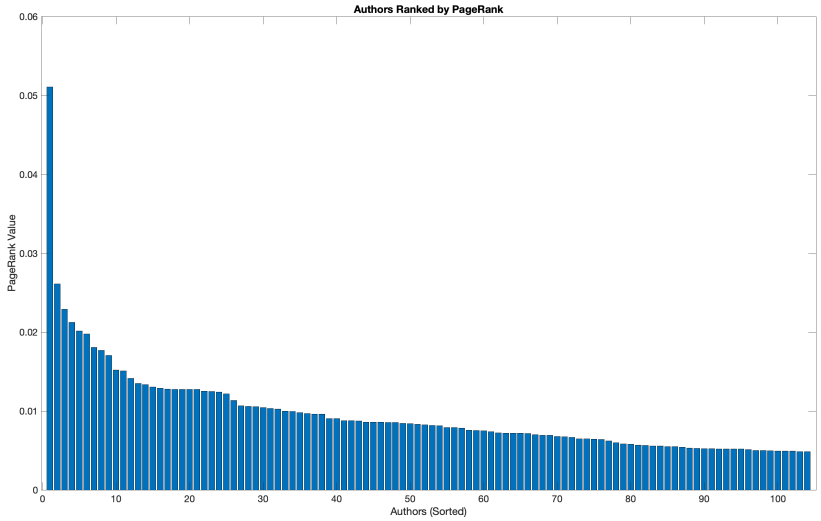
Below are the critical parts of the 'pagerank1_authors.m' script that compute the PageRank of authors using the Power Method:

```
% Initialize variables for power method
A_scaled = p * A * D;
z = ((1 - p) * (c ~= 0) + (c == 0)) / n;
x = e / n;
prevx = zeros(n, 1);

% Power iteration
limit = 1e-5; % Convergence limit
iterations = 0;

while norm(x - prevx) > limit
    prevx = x;
    x = A_scaled * x + e * (z * x);
    iterations = iterations + 1;
end
```

The following script was used to compute and plot the PageRank values:

```
load('housegraph.mat');

[x, iterations] = pagerank1_authors(A, 0.85);

[sorted_pagerank, sorted_indices] = sort(x, 'descend');

disp('Authors ranked by PageRank:');
disp(name(sorted_indices, :));

figure;
bar(sorted_pagerank);
title('Authors Ranked by PageRank');
xlabel('Authors (Sorted)');
ylabel('PageRank Value');
```

The PageRank values were computed and sorted for all authors. Below is a bar graph showing authors ranked by their PageRank values. The following table lists the top 5 authors by PageRank.



Authors Ranked by PageRank in the Householder Network

Rank	Author
1	Golub
2	Demmel
3	Plemmons
4	Schreiber
5	TChan

Top 5 Authors by PageRank

The PageRank algorithm was successfully applied to the coauthor network, providing an indication of the most influential authors based on their connections.

6. Zachary's karate club: an introduction to spectral graph partitioning [35 points]

6.1. Degree Centrality

Degree centrality measures the number of direct connections a node has.

The degree centrality for each node was computed by summing its connections in the adjacency matrix and then sorted:

```
degree_centrality = sum(karate, 2);
[sorted_degree, sorted_indices] = sort(degree_centrality, 'descend');
```

The following are the top 5 nodes ranked by degree centrality and their respective degrees:

- Node 34: Degree = 17
- Node 1: Degree = 16
- Node 33: Degree = 12
- Node 3: Degree = 10
- Node 2: Degree = 9

Nodes 34 and 1 have the highest number of direct connections in the network.

6.2. Eigenvector Centrality

Eigenvector centrality is a global measure of influence that considers both a node's direct connections and the importance of its neighbors.

The eigenvector centrality was computed by extracting the dominant eigenvector from the adjacency matrix and then properly normalized:

```
[eigenvectors, eigenvalues] = eig(karate);
[max_eigenvalue, max_index] = max(diag(eigenvalues));
eigenvector_centrality = eigenvectors(:, max_index);

eigenvector_centrality = abs(eigenvector_centrality / norm(eigenvector_centrality));
```

The following are the top 5 nodes ranked by eigenvector centrality, with normalized values:

- Node 34: Eigenvector Centrality = 0.3734
- Node 1: Eigenvector Centrality = 0.3555
- Node 3: Eigenvector Centrality = 0.3172
- Node 33: Eigenvector Centrality = 0.3086
- Node 2: Eigenvector Centrality = 0.2660

Node 34 ranks highest by eigenvector centrality, indicating that it is not only highly connected but also connected to influential nodes.

6.3. Comparison and Analysis of Degree and Eigenvector Centrality Ranking Results

Similarities

Nodes 34, 1, 33, 3, and 2 appear in both rankings, indicating that these nodes are highly central by both degree and eigenvector measures. This shows that the most connected nodes are also globally influential in the network.

Differences

While the top 5 nodes are the same in both rankings, their **ordering** differs. For example, Node 33 ranks 3rd by degree but 4th by eigenvector centrality, while Node 3 ranks 4th by degree but 3rd by eigenvector centrality. This difference suggests that although Node 33 has slightly more direct connections than Node 3, the neighbors of Node 3 are more influential in the global structure of the network.

Comments

Nodes with many direct connections (degree centrality) do not always have the highest global influence (eigenvector centrality). For example, Node 3 ranks higher than Node 33 in eigenvector centrality because it is connected to more influential nodes, even though Node 33 has more direct connections. This highlights how eigenvector centrality provides a broader measure of influence that accounts for a node's position in the overall network structure.

6.4. Spectral Graph Partitioning

Spectral graph partitioning was applied to divide the Karate Club network into two groups using the Fiedler vector, which is the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix. Nodes with positive values in the Fiedler vector were assigned to **Group 1**, and nodes with negative or zero values were assigned to **Group 2**:

```
fiedler_vector = eigenvectors_L(:, 2);
group1 = find(fiedler_vector > 0); % Positive values -> Group 1
group2 = find(fiedler_vector <= 0); % Negative/Zero values -> Group 2
```

The partition resulted in the following groups:

- **Group 1 (Positive values)**: 3, 9, 10, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34
- **Group 2 (Negative/Zero values)**: 1, 2, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20, 22

Zachary's real split (shown in Figure 6) divided the club into two groups based on a conflict that arose among the members. The split was as follows:

- **Group 1**: Circles (white nodes)
- **Group 2**: Squares (grey nodes)

Comparison with Zachary's Split

The spectral partitioning largely aligns with Zachary's real split but with some differences, such as Node 3 being placed in Group 1 by the spectral method, even though it belonged to Group 2 in the real split. However, some nodes like Node 1, are correctly classified in Group 2 (negative values), as they were in grey nodes in the real split.

Comments on Results

The spectral partitioning produced two groups: Group 1 (19 nodes) and Group 2 (15 nodes). This does not exactly match the required 16-18 split but is close.

Key observations:

- **Node 1** and **Node 34**, the most influential members, were correctly placed in different groups, aligning with their leadership roles in the real-world split.

- **Node 3**, however, was assigned to Group 1 by the spectral method, but in Zachary's real split, it belongs to Group 2. This suggests that the structural role of Node 3 may not align with the actual social dynamics.
- Most other nodes were accurately placed in their respective groups, showing that spectral partitioning largely aligns with the real split based on the network's structure.

Overall, spectral partitioning identifies the key players in the network but shows minor deviations, particularly for nodes with more subtle social roles, like Node 3.