Università della Svizzera italiana | Institute of Computing CI

Numerical Computing **2023**

Student: Leonardo Birindelli **Discussed with:** Morales Jeferson

Solution for Project 1 **Due date:** Wednesday, 11 October 2023, 23:59 AM

1. Theoretical questions [15 points]

(a) **What are an eigenvector, an eigenvalue and an eigenbasis?**
Those three concepts are strictly related to one another. First of all, I would like to start from the eigenvalues and eigenvectors definitions. Indeed, given a linear application $T : V \to V$, where V is a vector space, an eigenvalue $\lambda \in T$ is a scalar value such that there exists an eigenvector v (where $v \neq 0$) that makes the following equation valid: $T(v) = \lambda v$. It is also possible to write the next equation $Ax = \lambda v$ where A is a square matrix $n \times n$.

In order to obtain the eigenvalues of the matrix, and subsequently find the corresponding eigenvectors linked to the scalar values, it is needed to calculate the characteristic polynomial of the matrix A, which corresponds to calculating $\det(A - \lambda I)$.

In conclusion, one or more eigenvectors linked to a specific eigenvalue create an eigenbase, which represents a set of eigenvectors that are able to generate an eigenspace of all eigenvectors linked to the same eigenvalue.

(b) **What assumptions should be made to guarantee convergence of the power method?**

To ensure convergence of the power iteration method, the following three conditions need to be satisfied:

1. The matrix A must be **diagonalizable**

2. The eigenvalue with the largest magnitude, which is denotated as λ_n, must be separated from the other eigenvalues, meaning that $|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$

3.The initial vector $x^{(0)}$ have a non-zero component along the eigenvector $v_1$corresponding to λ_1.Given that A is diagonalizable,it means that there exists a base of eigenvectors and this lets to express $x^{(0)}$ as a linear combination of these eigenvectors :

$$x^{(0)} = \sum_{i=1}^{n} \alpha_i v^{(i)}$$

where $\alpha_1 \neq 0$.

(c) **What is the shift and invert approach?**

The "inverse power method with shift" is a variant of the power iteration method and this technique can be used to compute the eigenvalue of A closest to a given value μ, which is referred to as the "shift." The equation for this method is:

$$(A - \mu I)x = (\lambda - \mu)x$$

In this case, instead of the matrix A, we use $(A - \mu I)$ and the eigenvalue closest to μ is such that $|\lambda - \mu|$ is minimized.

(d) **What is the difference in cost of a single iteration of the power method, compared to the inverse iteration?**

There are several differences in terms of the cost of a single iteration between the power method and the inverse power method, each of which has advantages or disadvantages that directly depend on the specific case under consideration.

In terms of the cost of a single iteration, both the power method and the inverse power method are directly dependent on the size of the matrix they operate on. The complexity of the power method is $O(n^2)$ because it requires a matrix-vector product for each iteration, while the inverse power method has a complexity of $O(n^3)$ because it involves solving a linear system at each iteration.

Although the inverse power method is more computationally expensive in terms of complexity, it has the advantage of reaching convergence faster than the power method. This is because the convergence of the power method can be very slow if the ratio between eigenvalues is close to 1 ($|\frac{\lambda_2}{\lambda_1}| \approx 1$).

It should be noted that the inverse power method is effectively useful only when applied to small-sized problems and not for large-scale problems like web search applications.

The choice between the power method and the inverse power method depends on the specific problem at hand, and the trade-offs between computational complexity and convergence speed must be considered carefully.

(e) **What is a Rayleigh quotient and how can it be used for eigenvalue computations?**

The Rayleigh quotient is a value used to approximate, or even determine, eigenvalue associated to an eigenvector (or to an estimated vector for a matrix). The formula to obtain this value is the following:

$$\lambda(x) = \frac{x^T A x}{x^T x}$$

Where A is a symmetric matrix, and v is a nonzero eigenvector.

Indeed,if v is an eigenvector, $\lambda(v)$ would simply give the associated eigenvalue. If v is not an eigenvector, then the Rayleigh quotient of v gives the closest possible approximation to the eigenvalue.

Thanks to the Rayleigh quotient, once the eigenvector associated with the eigenvalue of maximum modulus is calculated in the power method, it is possible to obtain an approximation of that scalar value.

2. Connectivity matrix and subcliques [5 points]

The connectivity matrix for the ETH500 data set (ETH500.mat) has various small, almost entirely nonzero, submatrices that produce dense patches near the diagonal of the spy chart. You can use the zoom button to find their indices.The first submatrix has, e.g., indices around 80. Mathematically, a graph where all nodes are connected to each otheris known as a clique. Identify the organizations within the ETH community that are responsible for these near cliques.

The organizations in the ETH Community that are responsible for these near cliques, starting from the beginnig of the diagonal of the spy chart going to the end of it:

1. Dept. of Civil, Environmental and Geomatic Engineering

2. Department of Materials
3. Department of Mechanical and Process Engineering
4. Departement Biologie
5. Department of Chemistry and Applied Biosciences
6. Department of Mathematics
7. Department of Earth Sciences
8. Department of Environmental Systems Science
9. Department of Management, Technology, and Economics
10. Department of Humanities, Social and Political Sciences
11. Bilanz

3. Connectivity matrix and disjoint subgraphs [10 points]

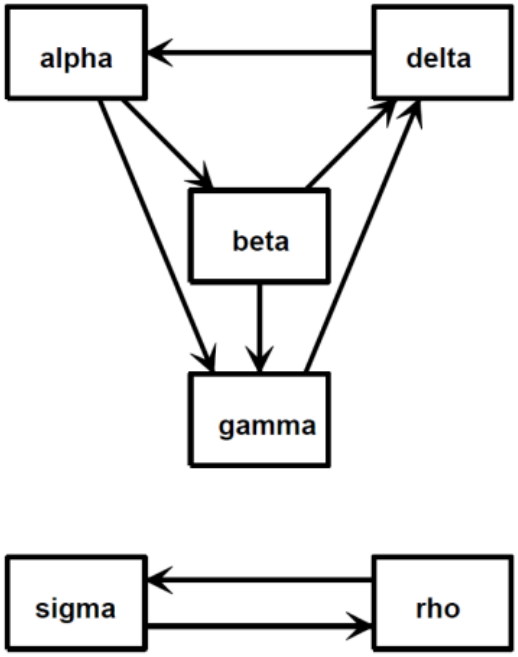

Figure 1: Tiny Web

1. . What is the connectivity matrix G? Which are its entries?

   The connectivity matrix G is a n-by-n matrix that represents a portion of the Web, whose cells that contains value 1 means which there is a hyperlink to page i from page j and zero otherwise, this matrix is it used to understand how the nodes in the graphs are linked. The entries of the connection matrix is the following :

$$G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

2. What are the PageRanks if the hyperlink transition probability p assumes the default value of 0.85?

   When the hyperlink transition probability p assumes the value of 0.85 the PageRanks of the tiny six-node subset of the web is the following :

```
% RageRanks results
ans =
    0.2037
    0.1556
    0.1981
    0.1092
    0.1667
    0.1667
```

3. Describe what happens with this example to both the definition of PageRank and the computation done by pagerank in the limit p $\rightarrow$ 1

   PageRank is an analysis algorithm used to evaluate and rank the importance of interconnected elements through hyperlinks, such as in the case of web pages on the World Wide Web (WWW).

   In the context of web pages, the general idea behind this algorithm is to assign a level of importance to a page within the set based on the links importance towards the other pages. Initially, the algorithm is fed with the elements of the set, where the importance of each element is the same. Subsequently, based on the links between the elements, PageRank provides the ranking of individual components in the system.

   Taking the current example of the subset of web pages into consideration, PageRank, where the probability 'p' is set to 0.85, signifies that when a user is on a page, there is a probability of 'p' to proceed to the next page. Otherwise, the user will be redirected to a random page within the set. This probability 'p' is vital to prevent getting trapped in cyclic paths among web pages. In fact, the closer the probability 'p' gets to 1, the higher chances of getting stuck in these cycles.

4. PageRanks by solving a sparse linear system [25 points]

1. Create pagerank1.m by modifying pagerank.m to use the power method instead of solving the sparse linear system. The key statements are

```
G = p*G*D
z = ((1-p)*(c~=0) + (c==0))/n;
while termination_test
x = G*x + e*(z*x)
end
```

Listing 1: While condition for the power method

   What is an appropriate test for terminating the power iteration?

   An appropriate termination test for the power iteraction method is to check the quality of the solution approximation which is implemented with the following expression
   $\|x_k - x\|_1 \geq \epsilon$ **or** $k < n$, where ϵ represent the acceptable tollerance for the approximation and $k < n$, in which "k" represents the counter of iteractions, establish a maximum number of iteraction if the first condition does not reach an acceptable result.
   In the next page,I show the code of "pagerank1.m" that I implemented.

```
%pagerank1.m while loop and its stop conditions
while norm(x_k - x,1)>=epsilon || k<n
    x=x_k;
    x_k=G*x + e*(z*x);
    x_k=x_k/norm(x_k,2);

    k=k+1;
end
```

Listing 2: While condition for the power method

2. Create pagerank2.m by modifying pagerank.m to use the inverse iteration. The key statements are

```
while termination_test
x = (alpha*I - A)\x
x = x/norm(x,1)
end
```

Listing 3: While condition for the inverse method

Use your functions pagerank1.m and pagerank2.m (set $\alpha = 0.99$) to compute the PageRanks of the six-node example presented in Figure 1. Make sure you get the same result from each of your three functions.
The charts results that I obtain from these three execution are the following:

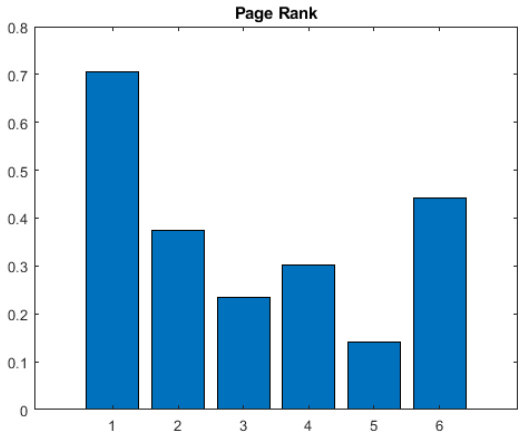

Figure 2: pagerank.m chart

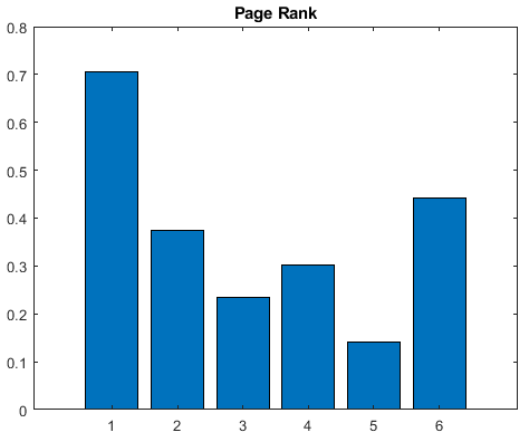

Figure 3: pagerank1.m chart

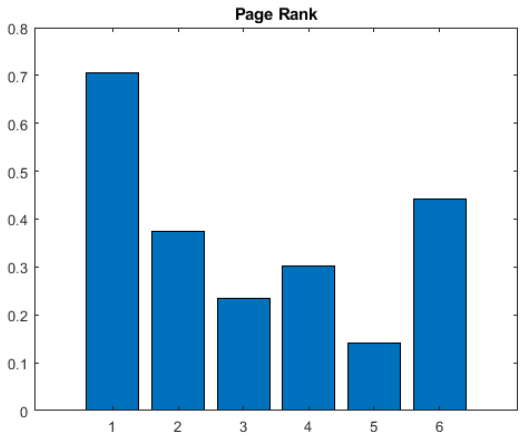

Figure 4: pagerank2.m chart

```
%pagerank2.m while loop and its stop conditions
while norm(eigenVal_k - eigenval,1)>= epsilon
x=x_k;
eigenval=eigenVal_k;
x_k = ( A-alpha*I )\x;
x_k = x_k/norm(x_k,1);
eigenVal_k= transpose(x_k)*A*x_k;


k=k+1;
end
```

Listing 4: While condition for the inverse method

3. We now want to analyse the impact of α on the inverse iteration. Using the ETH500 example, set α equal to 0.8, 0.9, 0.95 and 1. Comment on the different number of iterations the four cases take until convergence. Analyse your results and explain what you observe. Hint: Check your solution x for all 4 cases. Are they always the same?

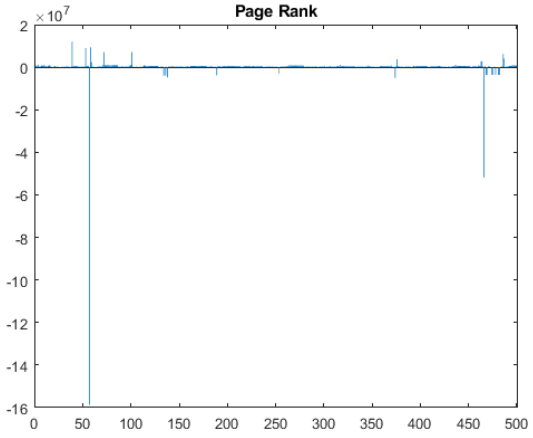

Figure 5: $\alpha = 0.80$ and **5** iteractions for convergence

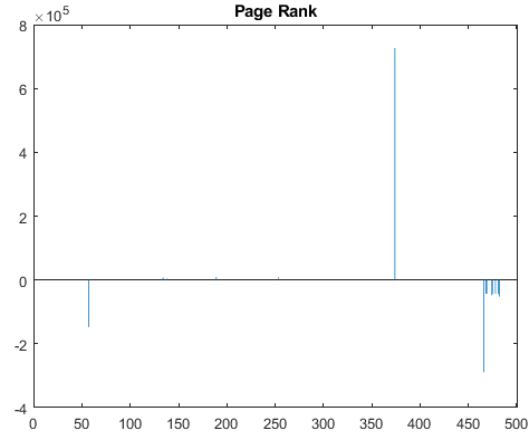

Figure 6: $\alpha = 0.90$ and**70** iteractions for convergence

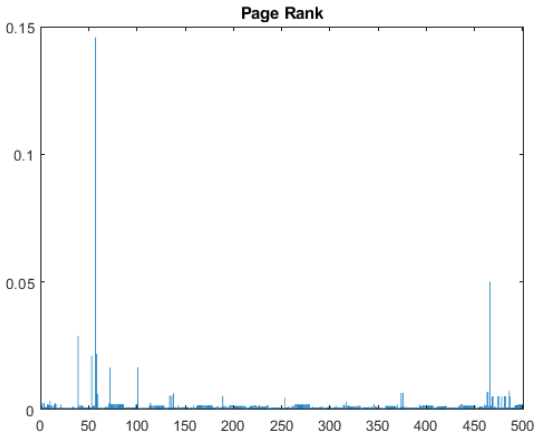

Figure 7: $\alpha = 0.95$ and **12** iteractions for convergence

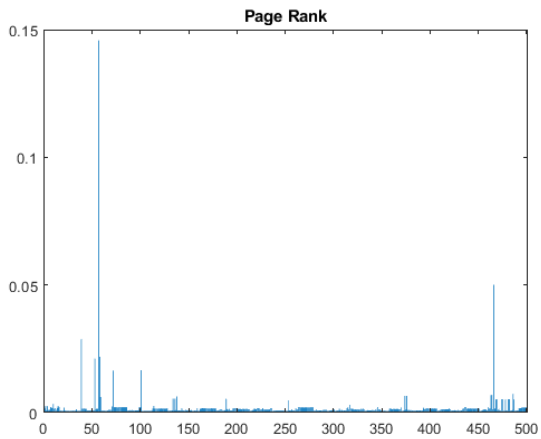

Figure 8: $\alpha = 1$ and **2** iteractions for convergence

What I especially observed during the computations is that in the first two cases where $\alpha = 0.8$ and $\alpha = 0.90$ some PageRank results assume negative values. Another unique case I

noticed is that when $\alpha = 1$, I received a warning: "Matrix is close to singular or badly scaled. Results may be inaccurate."

4. Use your functions pagerank1.m and pagerank2.m (set $\alpha = 0.99$) to compute the PageRanks of three selected graphs (web1.mat, web2.mat and web3.mat). Report on the convergence of the two methods for these subgraphs and summarize the advantages and disadvantages of the power method implemented in pagerank1.m against the inverse iteration in pagerank2.m

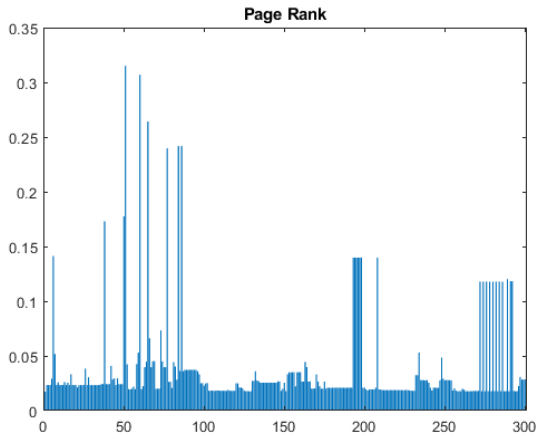

Figure 9: web1.m pagerank1 chart

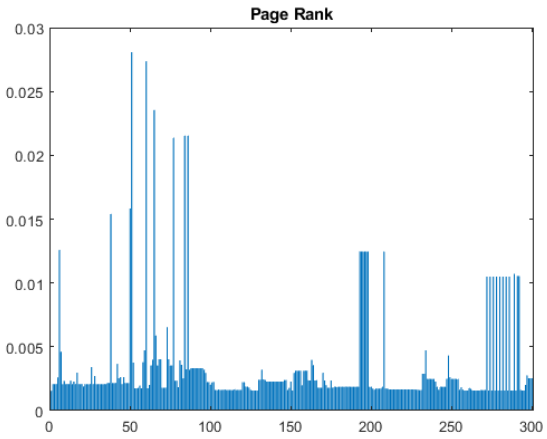

Figure 10: web1.m pagerank2 chart

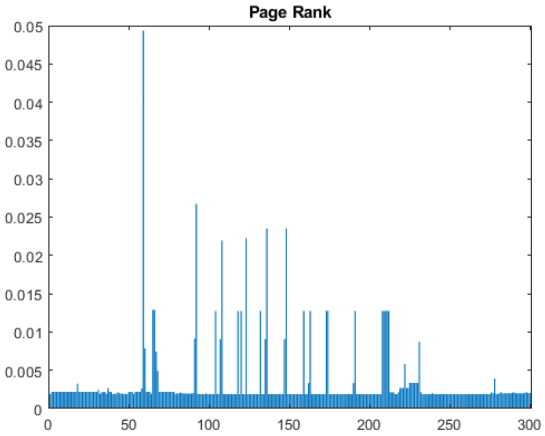

Figure 11: web2.m pagerank1 chart

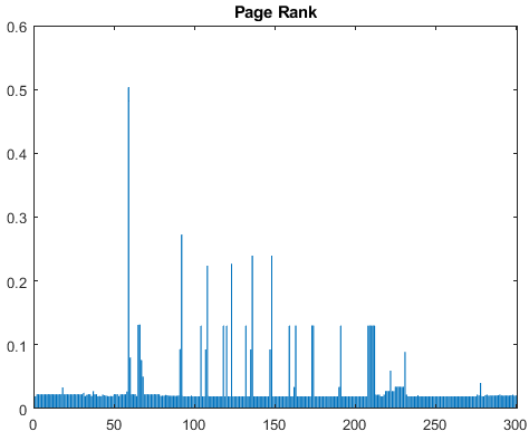

Figure 12: web2.m pagerank2 chart

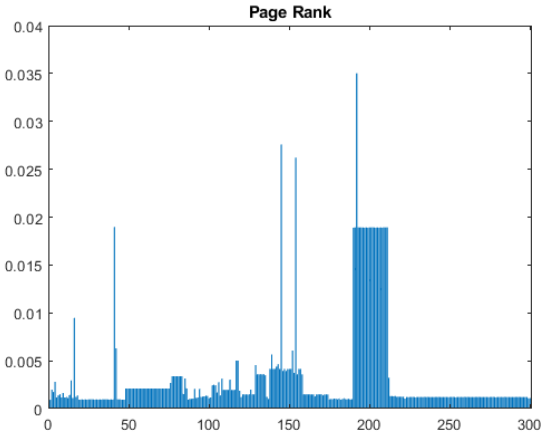

Figure 13: web3.m pagerank1 chart

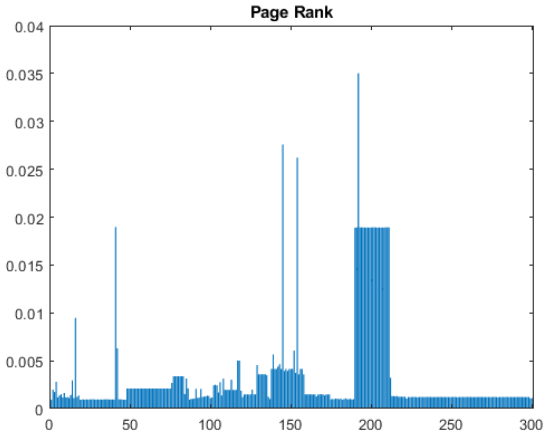

Figure 14: web3.m pagerank2 chart

The advantages and disadvantages of the power method, contained in the file named "pagerank1.m", compared to the inverse power method , implemented in the "pagerank2.m" file, are primarily related to the fundamentally different operations performed in each iteration. Indeed, matrix-vector multiplications are carried out, which results in better algorithmic complexity in the case of the power method.However, it can converge more slowly when the eigenvalues are such that $\lambda_1 > \lambda_2 > \ldots > \lambda_n$ and $|\frac{\lambda_2}{\lambda_1}| \approx 1$. On the other hand, a linear system is solved in each iteration of the while loop, which introduces greater computational complexity compared to the power method in the inverse power method. Nevertheless, the inverse power method overcomes the convergence issues encountered in the power method.

5. The Reverse Cuthill–McKee Ordering [5 points]

Load matrix "A SymPosDef.mat" from the dataset. You can reorder (permute) the matrix using "Reverse Cuthill McKee Ordering" via the Matlab function symrcm(), see Matlab documentation for more information. Visualize both the original and Reverse Cuthill McKee permuted matrix and comment on what you observe. Compute the Cholesky factor of the original matrix and the permuted matrix. Visualize the Cholesky factors and comment on thenumber of nonzeros.

In order to visualize both figures I used the "spy" function in matlab: To show the Figure 3 I executed:

```
>> load('A_SymPosDef.mat')
>> spy(A_SymPosDef)
```

On the other hand I use the following function so as to realize the Figure 4

```
>> i=symrcm(A_SymPosDef);
>> spy(A_SymPosDef(i,i));
```

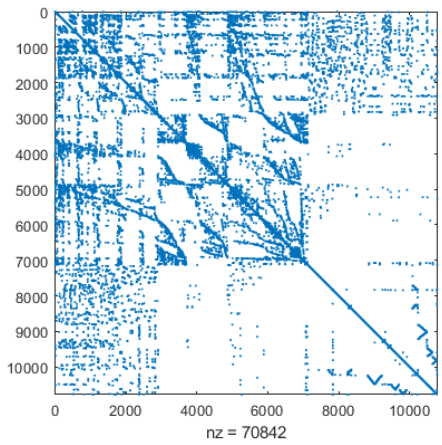

Figure 15: The original matrix

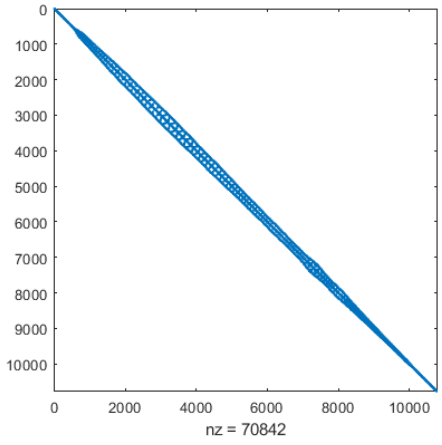

Figure 16: Reverse Cuthill McKeen permuted matrix

In order to compute and then visualize the Cholesky factor of both original matrix and the permuted matrix obtained by the application of the Reverse Cuthill Mckeen matlab function , I executed the following functions :

```
% Load the matrix "A_SymPosDef.mat"
load('A_SymPosDef.mat');
% Save the Cholesky factor in the variable "chol_matrix"
chol_matrix = chol(A_SymPosDef);
% Visualize the Cholesky factor
spy(chol_matrix);
```

Listing 5: Cholesky factor of original matrix

```
% Load the matrix "A_SymPosDef.mat"
load('A_SymPosDef.mat');
% Save the permutation returned by the "symrcm" function
i=symrcm(A_SymPosDef);
% Save the Cholesky factor in the variable "chol_Pmatrix"
chol_Pmatrix = chol(A_SymPosDef);
% Visualize the Cholesky factor
spy(chol_Pmatrix);
```

Listing 6: Cholesky factor of permuted matrix

The charts I get from the application of those code lines are the following:

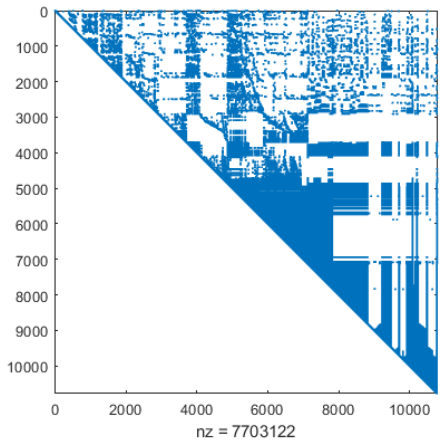

Figure 17: "chol" chart of original matrix

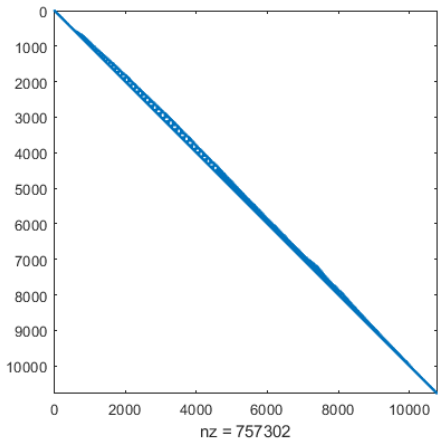

Figure 18: "chol" chart of permuted matrix

Looking at the two charts, Cholesky factor behaves in different ways directly depends on the matri. Indeed in the chart on the left side , the factor is directly applied on a Hermitian matrix and it return the corrispective upper triangle matrix . On the other hand, the application of "chol" function on the permuted matrix,obtained by the application of the Reverse Cuthill Mckeen algoritm, gives back a chart where all nonzero elements are close to the diagonal of the matrix and the number of nonzero cells is lower than the first matrix

6. Sparse Matrix Factorization [10 points]

Let $A \in R^{n \times n}$ be symmetric and positive definite, with entries A_{ij} defined as follows:

$$A_{ij} = \begin{cases} 1, & \text{if } i = 1 \text{ or } i = n \text{ or } j = 1 \text{ or } j = n \text{ and } i \neq j \\ n + i - 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

Please note that the increasingly larger values on the diagonal are necessary to ensure the positive definiteness of matrix A.

1. Construct matrix A for the case n = 10 and explicitly write down its entries. How many non-zero elements does it have?

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

The previous square matrix A with dimension n equals to 10 has 44 non-zero elements.

2. We now want to derive a general formula to compute the number of non-zero entries.Show that, for a given matrix $A \in R^{n \times n}$ with this structure, the number of non-zero is $5n - 6$.

   The number of non-zero elements is 5n-6 for a given $A \in R^{n \times n}$; in order to dimonstrate this affermation it is possible to say that in each matrix with these properties there always will be the first and the last rows containg $2 * n$ elements .The others $n - 2$ rows of the structure are going to contain 3 non-zero elements, the one in the first cell, in the last and one in the diagonal of the matrix, in conclusion the number of non-zero numbers is given by the expression $3 * (n - 2)$. Now I have enough information to demonstrate that the general formula is true:

   $2 * n + 3 * (n - 2) = 2 * n + 3 * n - 6 = 5n - 6$

   I demonstrate that the relation $5n - 6$ is true for every matrix $A \in R^{n \times n}$

3. Write a function `A_construct()`, which takes as input n and returns, as output, the matrix A defined in Eq. 14 and its number of non-zero elements nz. Test your function in a script ex2c.m for n = 10 and compare your results with those you obtained in point (a). Furthermore, within the same script, visualise the non-zero structure of matrix A by using the command spy().

```
%A_constructfunction
function [matrix_A,nz] = A_construct(n)

% Initialize the matrix A with zeros
matrix_A=zeros(n);

% Fill in the entries according to the given properties
for i = 1:n
    for j = 1:n
        if i==j
            matrix_A(i, j) = n + i - 1;
        elseif i == 1 || i == n || j == 1 || j == n
            matrix_A(i, j) = 1;
        end
    end
end
% counts Non-zeros values
nz=nnz(matrix_A);
end
```

Listing 7: A_costruct function

Using the following script file named "ex2c.m" for $n = 10$ I obtain the following result :

and the matrix has exactly

```
matrix =
    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

nz =
    44
```

Listing 8: Results of the A_construct function

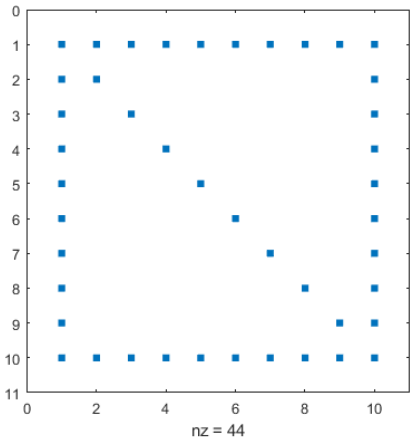

Figure 19: Output of spy(A)

4. Using again the spy() command, visualize side by side the original matrix A and the result of the Cholesky factorization (chol() in Matlab).

   Here I visualize with "spy" function both original matrix and the corrisponding Cholensky factor application

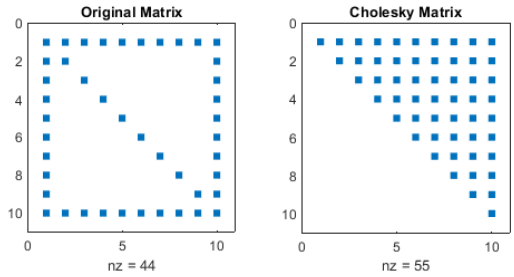

Figure 20: Output of "spy(A)" on the left and "spy(chol(A))" on the other side

5. Explain why, for n = 100, 000, using chol() to solve Ax = b for a given right-hand-side vector b would be problematic. Are there ways to mitigate this issue?

   It is problematic because `chol()`would require a lot of memory to store the matrix, and the calculations would be very complex and inefficient. To solve this issue, there are highly efficient techniques for matrix inversion in the form of $Ax = b$.

7. Degree Centrality [5 points]

In graph theory and network analysis, centrality refers to indicators which identify the most important vertices within a graph. Applications include identifying the most influential person(s) in a social network, key infrastructure nodes in the Internet or urban networks, and super spreaders of disease. Here we are interested in the Degree centrality, which is conceptually simple. It is defined as the number of links incident upon a node (i.e., the number of vertices that a node has). The degree centrality of a vertex v, for a given graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges, is defined as the numbers of edges of vertex v. Compute the degree centrality for the top 5 authors. Include them in an ordered list, and show the authors, the their coauthors and the degree centrality.

```
Author name : Golub
Coauthors of the 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
- Gill
Degree centrality of the author: 32
```

Listing 9: 1° author

```
Author name : Demmel
Coauthors of the author : Demmel
- Edelman
- VanLoan
- Bai
- Schreiber
- Kahan
- Kagstrom
- Barlow
- NHigham
- Arioli
- Duff
- Hammarling
- Bunch
- Heath
- Greenbaum
- Gragg
Degree centrality of the author: 16
```

Listing 10: 2° author

```
Author name : Plemmons
Coauthors of the author : Plemmons
- Golub
- Nagy
- Harrod
- Pan
- Funderlic
- Bojanczyk
- George
- Barlow
- Heath
- Berry
- Sameh
- Meyer
- Nichols
Degree centrality of the author: 14
```

Listing 11: 3° author

```
Author name : Schreiber
Coauthors of the author : Schreiber
- TChan
- VanLoan
- Moler
- Gilbert
- Pothen
- NTrefethen
- Bjorstad
- NHigham
- Eisenstat
- Tang
- Elden
- Demmel
Degree centrality of the author: 13
```

Listing 12: 4° author

```
Author name : Heath
Coauthors of the author : Heath
- Golub
- TChan
- Funderlic
- George
- Gilbert
- Eisenstat
- Ng
- Liu
- Laub
- Plemmons
- Paige
- Demmel
Degree centrality of the author: 13
```

Listing 13: 5° author

8. The Connectivity of the Coauthors [5 points]

How many coauthors have the authors in common? Think about a general procedure that allows you to compute the list of common coauthors of two authors and express it in matrix notation. Use the formula you derived to compute the common coauthors of the pairs (Golub, Moler), (Golub, Saunders), and (TChan, Demmel). Who are these common coauthors? Report their names.

In order to create a procedure that allows to compute the list of common coauthors between two authors I thought to create two functions contained in the files "exerciseN8.m" and "findIndex.m". The first function aims to find the common coauthors between the specified two authors names passed as paramethers.Once the function is executed, it will return a sparse matrix where the common coauthors identifiers are the column positions of the non-zeros elements and also print the name of those coauthors in the terminal. The second file ,instead, is a support function used by the "exerciseN8.m" document to returns the index of the author name given the matrix in which it has to search.

```
function comn_IdCoauthors = exerciseN8(authorName1,authorName2)
load('housegraph.mat', 'A')
load('housegraph.mat', 'name')

%% memorize the matrix index of the corresponding author name
coauthor1_index=findIndex(name,authorName1);
coauthor2_index=findIndex(name,authorName2);

%% return the row vector containing all coauthors of the specified authors
coauthors_1=A(coauthor1_index,:);
coauthors_2=A(coauthor2_index,:);

%% it find the coautors in common, only the indexes
comn_IdCoauthors=coauthors_1 & coauthors_2;

%Remove the authors has coauthors between each others
comn_IdCoauthors(1,coauthor1_index)=0;
comn_IdCoauthors(1,coauthor2_index)=0;

%print the name of common coauthors
fprintf("Common coauthors between \'%s\' and \'%s\' :\n",authorName1,
    authorName2);
for i=1:numel(comn_IdCoauthors)
     if(comn_IdCoauthors(1,i)==1)
     fprintf("%s\n",name(i,:));
     end

end

end
```

Listing 14: exerciseN8.m function

```
%This function returns the index of the author name (formatted as a char
%array) given the matrix in which it has to search
%if there is a match, the function will return the index of the
%name, otherwise it will return 0

function index = findIndex(matrix,nameCharArr)
%initialize 'index' equals to 0
index=0;

for i=1:size(matrix,1) %scroll the matrix rows
   %stampe di debug- da levare
    %matrix(i,:);
    %strncmp(matrix(i,:),nameCharArr,numel(nameCharArr));

    if strncmp(matrix(i,:),nameCharArr,numel(nameCharArr))
        index=i;
        break;
    end
end

end
```

Listing 15: findIndex.m function

```
% Common coauthors between 'Golub' and 'Moler' :
% Wilkinson
% VanLoan
exerciseN8('Golub','Moler');

% Common coauthors between 'Golub' and 'Saunders' :
% Gill
exerciseN8('Golub','Saunders');


% Common coauthors between 'TChan' and 'Demmel' :
% Schreiber
% Arioli
% Duff
% Heath
exerciseN8('TChan','Demmel');
```

Listing 16: exerciseN8.m function calls

9. PageRank of the Coauthor Graph [5 points]

Compute the PageRank value (e.g., by using a modified version of pagerank.m from Project 1) for all authors and provide a graph of all authors in descending order according to the PageRank. Include your script in the submission.

```
%exercise number 9
load('housegraph.mat', 'A')
load('housegraph.mat', 'name')

%pagerank1_mod is an edited version of pagerank1.m file
%in which it does not need the 'U' parameter
ranking= pagerank1_mod(A,0.85);

% Sort authors by PageRank in descending order
[sorted_pageRank_val,sorted_indices] = sort(ranking, 'descend');

sorted_names = name(sorted_indices, :);

%generate a graph of all authors in descending order according to the
%Pagerank valeus
figure;
ylabel('PageRank values');
xlabel('Authors names');
title('Authors by PageRank (Descending Order)');
bar(sorted_pageRank_val);
set(gca, 'XTick', 1:numel(sorted_names), 'XTickLabel', sorted_names);
```

Listing 17: "exerciseN8.m" function

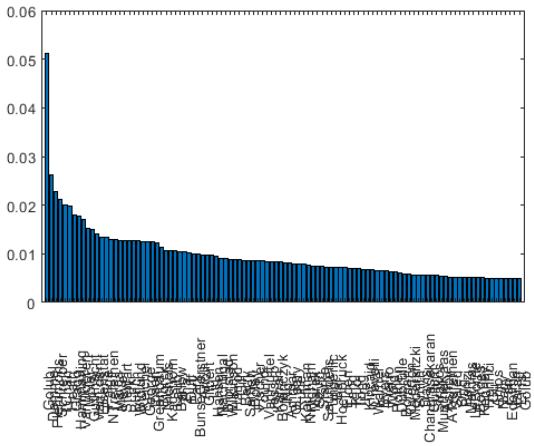

Figure 21: Graph of exercise 9

10. References

1 The power method and variants, Chapter 8: Eigenvalues and Singular Values, SIAM Book "A First Course on Numerical Methods", C. Greif, U. Ascher

2 Pagerank Slides of the lesson of 26th September 2023 avaiable on iCorsi platform.pdf

3 Power iteration method, http://en.wikipedia.org/wiki/Power_iteration

4 Inverse iteration, http://en.wikipedia.org/wiki/Inverse_iteration