Università della Svizzera italiana

Institute of Computing CI

Numerical Computing **2023**

Student: Harkeerat Singh Sawhney **Discussed with:** FULL NAME

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

Numerical Computing 2023 — 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, MATLAB). 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.

1. Theoretical questions [15 points]

(a) **What are an eigenvector, an eigenvalue and an eigenbasis?**
Eigenvector: In linear transformation an eigenvector is a non-zero vector that can get changed through a constant factor when the linear transformation is applied on it. When the eigenvector is linearly transformed only the scale is changed not the direction.

Eigenvalue: Eigenvalue is often represented with λ which is the multiplying factor which determines how much the eigenvector is linearly transformed along the direction. It is the factor by which the eigenvector is stretched or compressed.

Eigenbasis: Eigenbasis is a diagonal matrix that has eigenvalues along the diagonal and eigenvectors its columns. Eigenbasis are helpful in cases where functions such as matrix powers, exponentials and other similar functions needs to be performed due to the simplification in the computation.

(b) **What assumptions should be made to guarantee convergence of the power method?**
Firstly, we need to assume that the randomly chosen initial vectors must be part of the same direction as the eigenvector. We also need to assume that in the eigenvector which is used to get the Page Rank convergence is the dominant eigenvalue. λ_1 and λ_2 need to be distance to assure a faster convergence. Since the asymptotic error is constant $\frac{\lambda_1}{\lambda_2}$, this means that if the eigenvectors are close to each other then the convergence would be very slow.

(c) **What is the shift and invert approach?** In Shift and Invert technique if the eigenvalue of A are λ_j then the eigenvalues of $A - \alpha I$ are $\lambda_j - \alpha$ and the eigenvalues of $B = (A - \alpha I)^{-1}$ would be as follows:

$$\mu_j = \frac{1}{\lambda_j - \alpha}$$

From this we can see that the close α is to λ_j the larger and more dominant the largest eigenvalue of B will be. This would be mean that if the limit $\alpha- > \lambda_1$ then the first eigenvalue of B to would be going to infinity whereas other eigenvalues would be going to finite values.

Therefoere now if we apply the power method to $B = (A - \alpha I)^{-1}$ rather than towards A, and we also asume that λ_2 is the eigenvalue of A which is the closest to λ_1. With such iteration we would converge linearly, but the rate of convergence would be much faster than the power method.

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

When Power Method is used, in each iteration there would be a matrix-vector multiplication where in the Inverse Iteration we would be solving a linear system. It is computationly expensive to solve a linear system than to perform a matrix-vector multiplication. In order to use the inverse iteration, we must have a fast convergence for the inverse iteration to be faster than the power method. Realisticly speaking this would be very tought to implement in real world scenarios such as the PageRank problem on the web. Hence Inverse Iteration should be used for small problems where the matrix is small and dense.

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

Rayleigh quotient is a form of Inverse Itearation, but the difference is the method in which the eigenvalues are found. Rayleigh quotient guarantees fast convergence, because it change the value of α in each iteration. The value of α is changed in each iteration to the Rayleigh quotient of the current vector. With this approach the convergence speed increases as we

get closer to the eigenvalue. With this the convergence speed is better than linear. The convergence speed in fact in most cases is cubic. Hence it is worth paying the price for having to refactor the matrix in every iteration.

2. Connectivity matrix and subcliques [5 points]

By observing the spy plot, it was observed that there starting from the range 73-100 there were in total 11 cliques. Hence by manually noting down the ranged of the indices for the cliques, the dominant organization was found. Bellow is an example of how the first cliques was found.

```
% Finding the dominant organization
% U = 500 x 1 cell containing the names of the organizations
clique_indices = 73:100;
near_clique_organizations = U(clique_indices);
```

Hence in the Table 1 repersents all the cliques found in the spy plot and the organization that was found to be dominant in those clique.

CLIQUE RANGE	DOMINANT ORGANIZATION
73-100	www.baug.ethz.ch
113-129	www.mat.ethz.ch
164-182	www.mavt.ethz.ch
198-220	www.biol.ethz.ch
221-263	www.chab.ethz.ch
264-315	www.math.uzh.ch
319-348	www.erdw.ethz.ch
358-395	www.usys.ethz.ch
396-435	www.mtec.ethz.ch
436-462	www.gess.ethz.ch
486-499	www.bilanz.ch

Table 1: Dominant organization in each clique

3. Connectivity matrix and disjoint subgraphs [10 points]

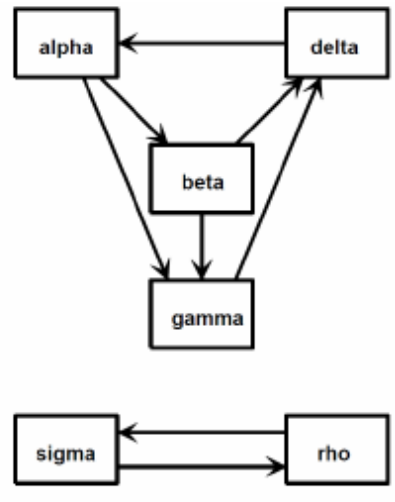

Figure 1: Another Tiny Web

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

The connectivity matrix G is a mathematical representation which is used to describe the graph in the 1. The matrix describes the connections between the nodes in a directed graph. For a directed graph with n nodes, the connectivity matrix G is an $n \times n$ matrix, where the g_{ij} is defined as follows:

- If there is a directed edge from node i to j then $g_{ij} = 1$
- If there is no directed edge from node i to j then $g_{ij} = 0$

Bellow is the connectivity matrix G for the graph 2. In the graph there are 6 nodes and hence the connectivity matrix is a 6×6 matrix. From the first row to the last row, and from first column to the last column, the nodes are ordered as *alpa, beta, gamma, delta, rho, sigma.*

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

Figure 2: Connectivity matrix G for the graph in 1

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

In order to find the PageRanks for the graph in 1, we first need to compute the variable G, U to use the function *pagerank*. We know that $p = 0.85$. The code for the same is shown bellow.

```
% Finding the PageRanks for the graph in \ref{fig:another-tiny-web}
i = [1, 2, 3, 3, 4, 4, 5, 6] % row indices
j = [4, 1, 1, 2, 2, 3, 6, 5] % column indices
G = sparse(i, j, 1, 6, 6); % sparse matrix
U = {"alpha", "beta", "gamma", "delta", "rho", "sigma"} % cell array
pagerank(U, G, 0.85) % PageRanks
```

Hence we have obtained the PageRanks for the graph in 1 as shown in Figure 3.

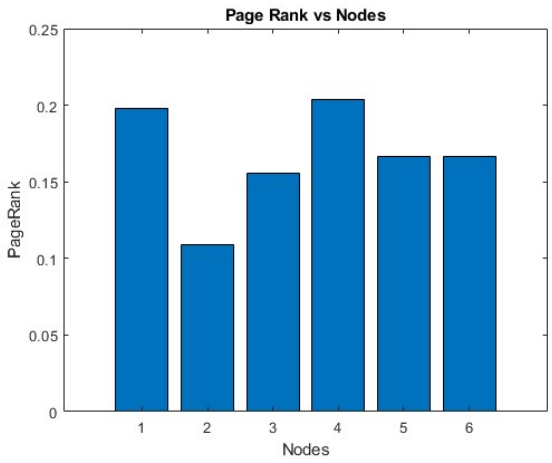

Figure 3: PageRank with p = 0.85

As it can be seen from the Figure 3, the PageRank for the node *delta* is the highest ranking amongst other nodes. The above data can also be visualzed in the form of a table in Figure 4.

Index	Node	PageRank	in	out
4	delta	0.2037	2	1
1	alpha	0.1981	1	2
5	rho	0.1667	1	1
6	sigma	0.1667	1	1
3	gamma	0.1556	2	1
2	beta	0.1092	1	2

Table 2: PageRank Table with p = 0.85

3.3. Describe what happens with this example to both the definition of PageRank and the computation done by pagerank in the limit p → 1.

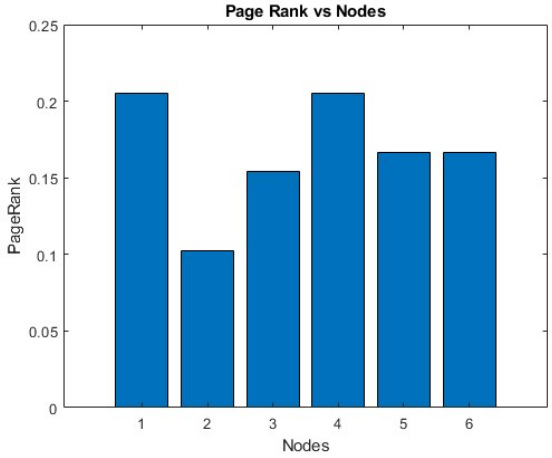

Figure 4: PageRank with p = 0.9999

Index	Node	PageRank	in	out
4	delta	0.2051	2	1
1	alpha	0.2051	1	2
5	rho	0.1667	1	1
6	sigma	0.1667	1	1
3	gamma	0.1556	2	1
2	beta	0.1092	1	2

Table 3: PageRank Table with p = 0.9999

When the value of p is increased to close to 1, the PageRank of the nodes in the graph will more evenly distributed. This is because the probability of the random surfer to jump to another node is very low. This can be noticed in the Figure as for the nodes with the same number of in and out links have the same PageRank. When p was set to 0.85 the weightage of the incoming links are much higher. However when it is set to 0.99999 it makes the PageRank of the nodes more evenly distributed.

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

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

6. Sparse Matrix Factorization [10 points]

6.1. 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 total number of non-zero elements are **44**.

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

In order to dervive a general formula to compute the number of non-zero entries in the matrix we should first observe the matrix and see some of its properties. We can observie that the matrix boundries and its diagonal are all non-zero. This is a really important property as wel can use this to derive the general formula.

We can see that there is 2 rows (top-most and bottom-most) which are non-zero, there are 2 columns (left-most and right-most) and at last there is a diagnol matrix. We can see the size of there is n, in this case 10. Hence we can dervive that the number of non-zero elements are $(2+2+1)n = 5n$. However as it can be noticed we are couting some of the non-zero elements more than once. Hence we need to subtract the number of non-zero elements which are counted more than once.

Finding all the elements being counted more than once:

- A_{11} is being counted 2 extra times.
- A_{nn} is being counted 2 extra times.
- A_{1n} is being counted 1 extra time.
- A_{n1} is being counted 1 extra time.

Hence in total there are 6 elements which are being counted more than once. Hence the total number of non-zero elements are:

$$5n - 6$$

.

6.3. 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 6.1. Furthermore, within the same script, visualise the non-zero structure of matrix A by using the command spy().

```
function [A,nz] = A_construct(n)
    A = zeros(n);        % initialize matrix
    for i = 1:n          % iterate over rows
        for j = 1:n      % iterate over columns
            if i == j
                A(i, j) = n + i - 1;
            elseif i == 1 || i == n || j == 1 || j == n
                A(i, j) = 1;
            else
                A(i, j) = 0;
            end
        end
    end
nz = nnz(A);
end
```

By running the above function, which replicates the Eq. 14, we obtained the same matrix which we had obtained in 6.1. The number of non-zero elements are also the same. The spy plot for the matrix A is shown in Figure 5.

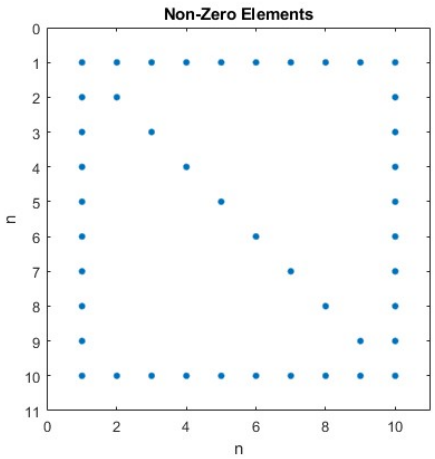

Figure 5: Spy plot for the matrix A

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

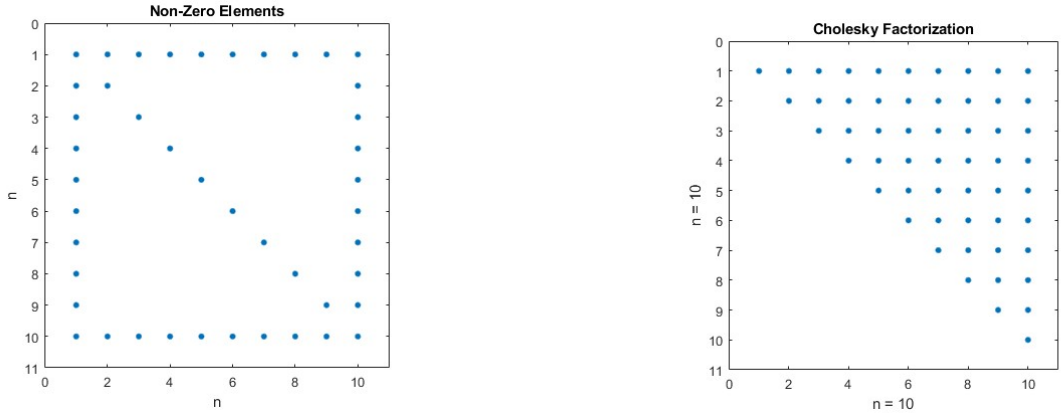

Figure 6: Spy plot for the matrix A vs Cholesky factorization for the matrix A

6.5. 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?

7. Degree Centrality [5 points]

Degree Centrality	Author	Index
32	Golub	1
16	Demmel	104
14	Plemmons	86
13	Schreiber	44
13	Heath	81

Table 4: Top 5 authors with highest degree centrality

8. The Connectivity of the Coauthors [5 points]

9. PageRank of the Coauthor Graph [5 points]

10. Quality of the Report [15 points]