



Università
della
Svizzera
italiana

Institute of
Computing
CI

Numerical Computing

2023

Student: Harkeerat Singh Sawhney

Discussed with: Guerrero Toro Cindy

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 $\alpha \rightarrow \lambda_1$ then the first eigenvalue of B 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 computationally 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 tough 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 Iteration, 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.

```
1 % Finding the dominant organization
2 % U = 500 x 1 cell containing the names of the organizations
3 clique_indices = 73:100;
4 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]

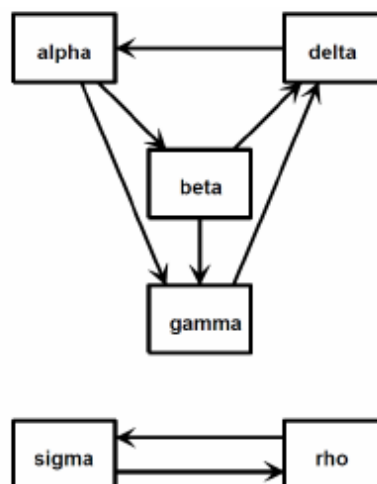


Figure 1: Another Tiny Web

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 *alpha*, *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.

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

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

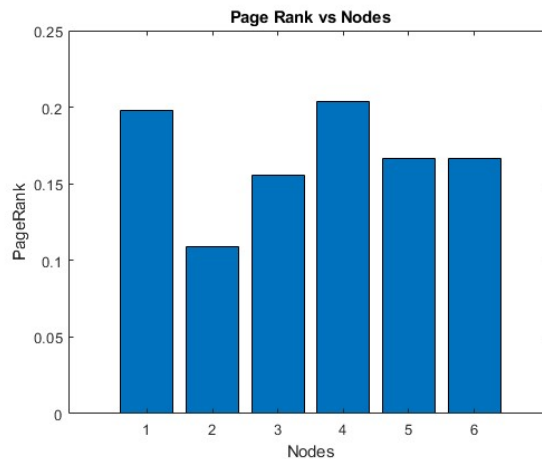


Figure 3: PageRank with $p = 0.85$

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 visualized in the form of a table in Figure 3.

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 $p \rightarrow 1$.

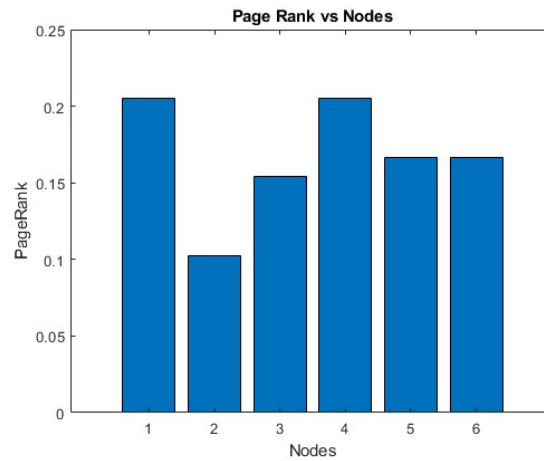


Figure 4: PageRank with $p = 0.9999$

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]

4.1. Create pagerank1.m by modifying pagerank.m to use the power method instead of solving the sparse linear system.

```

1 function x = pagerank1(U,G,p)
2 %.....
3 disp('Using Power Method Implementation\n');
4 G = p * G * D;
5 z = ((1 - p) * (c ~= 0) + (c == 0))/n;
6 prec = 1e-16;
7 x = e/n;
8 x_old = zeros(size(x));
9
10 while((norm(x - x_old)/ norm(x_old)) > prec)
11 x_old = x;
12 x = G * x + e * (z * x);
13 disp(x - x_old)
14 end
15 %.....
16 end

```

The appropriate test for terminating the Power Method is with by getting the norm difference of x and x_{old} and seeing if it is bigger then the precision. The precision is set to $1e - 16$. If it is not bigger, then we assume that there has been a convergence and hence we terminate the Power Method.

index i	page-rank x	in r	out c	url U
4	0.203694	2	1	delta
1	0.19814	1	2	alpha
5	0.166667	1	1	rho
6	0.166667	1	1	sigma
3	0.155623	2	1	gamma
2	0.109209	1	2	beta

Table 4: PageRank1 table

The above example uses the six-node as an example to test the Power Method. The PageRank1 table is shown in Table 4. The pagerank1 function took in total **125** iterations to converge.

4.2. Create pagerank2.m by modifying pagerank.m to use the inverse iteration. 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.

```

1 function x = pagerank1(U,G,p)
2 %.....
3 disp('Using Inverse Iteration Implementation\n');
4 G = p * G * D;
5
6 z = ((1 - p) * (c ~= 0) + (c == 0))/n;
7 A = p * G * D + e * z;
8
9 prec = 1e-16;
10 x = e/n;
11 x_old = zeros(size(x));
12 alpha = 0.99;
13 iter = 0;
14 while((norm(x - x_old)/ norm(x_old)) > prec)

```

```

15     x_old = x;
16     x = (alpha * I - A) \ x;
17     x = x / norm(x,1);
18     iter = iter + 1;
19 end
20 %.....
21 end

```

index	page-rank	in	out	url
i	x	r	c	U
4	0.203694	2	1	delta
1	0.19814	1	2	alpha
5	0.166667	1	1	rho
6	0.166667	1	1	sigma
3	0.155623	2	1	gamma
2	0.109209	1	2	beta

Figure 5: PageRank1 table

index	page-rank	in	out	url
i	x	r	c	U
4	0.203694	2	1	delta
1	0.19814	1	2	alpha
5	0.166667	1	1	rho
6	0.166667	1	1	sigma
3	0.155623	2	1	gamma
2	0.109209	1	2	beta

Figure 6: PageRank2 table

While pagerank1 took **125** iterations to converge, pagerank2 took **52** iterations to converge.

4.3. We now want to analyse the impact of alpha on the inverse iteration. Using the ETH500 example, set alpha 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.

Degree Centrality	Author	Index
31	Golub	1
15	Demmel	104
13	Plemmons	86
12	Schreiber	44
12	Heath	81

Table 5: Top 5 authors with highest degree centrality

As it can be seen from the table above, as the value of α increases the number of iterations required to converge also increases the iteration as well increases with it. This is because as the value of α increases the value of $\frac{1}{\lambda_1 - \alpha}$ decreases. Hence the convergence is slower.

4.4. 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.

Website	PageRank 1 Iteration	PageRank 2 Iteration
Web 1	173	98
Web 2	145	74
Web 3	212	93

Figure 7: PageRank1 vs PageRank2

From the above data which was computed it can be seen that PageRank2 performed better in terms of iteration as compare to PageRank1. However PageRank1 which uses the Power Method is computationally cheaper than PageRank2 which uses the Inverse Iteration. Due to this the time it takes to compute PageRank1 is much less as compare to PageRank2. Hence PageRank1 is usefull in majority of the cases, but takes much more iterations to converge. In the other PageRank1 congerges in less iterations, but is very sensitive to its alpha value.

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

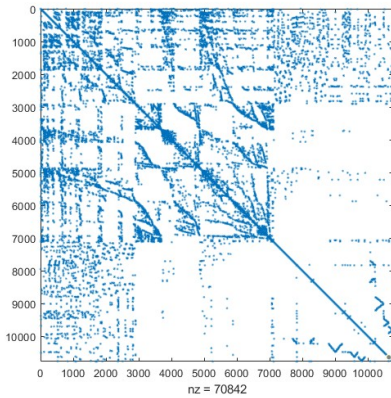


Figure 8: Original Matrix

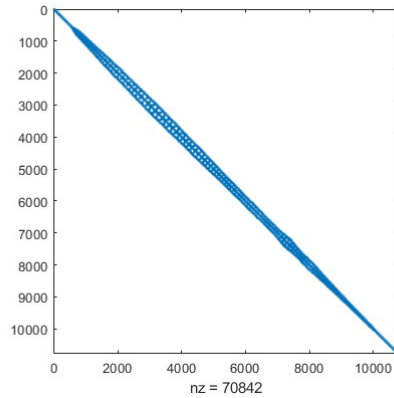


Figure 9: Reverse Cuthill McKee Matrix

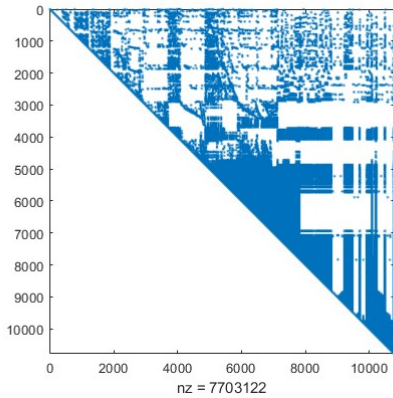


Figure 10: Cholesky factorization of the original matrix

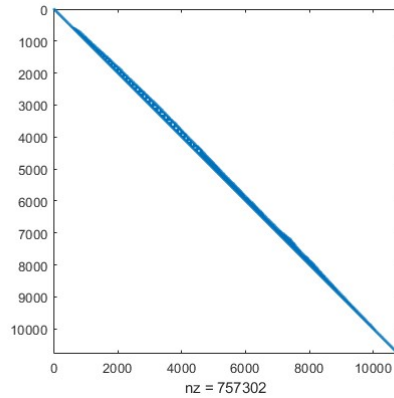


Figure 11: Cholesky factorization of the reverse Cuthill McKee matrix

The non-zero elemnts in the Cholesky Factorization of the original matrix is **7703122**, while for the Cholesky Factorization of the reverse Cuthill McKee matrix is **757302**.

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 derive 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 observe that the matrix boundaries and its diagonal are all non-zero. This is a really important property as we 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 diagonal matrix. We can see the size of there is n , in this case 10. Hence we can derive that the number of non-zero elements are $(2+2+1)n = 5n$. However as it can be noticed we are counting 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()`.

```

1 function [A,nz] = A_construct(n)
2 A = zeros(n);           % initialize matrix
3 for i = 1:n             % iterate over rows
4     for j = 1:n         % iterate over columns
5         if i == j
6             A(i, j) = n + i - 1;
7         elseif i == 1 || i == n || j == 1 || j == n
8             A(i, j) = 1;
9         else
10            A(i, j) = 0;
11        end
12    end
13 end
14 nz = nnz(A);
15 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 12.

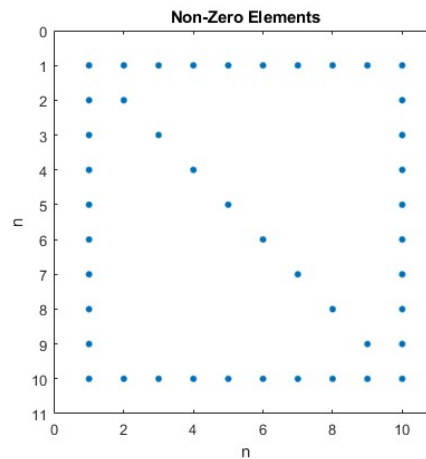


Figure 12: Spy plot for the matrix A

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).

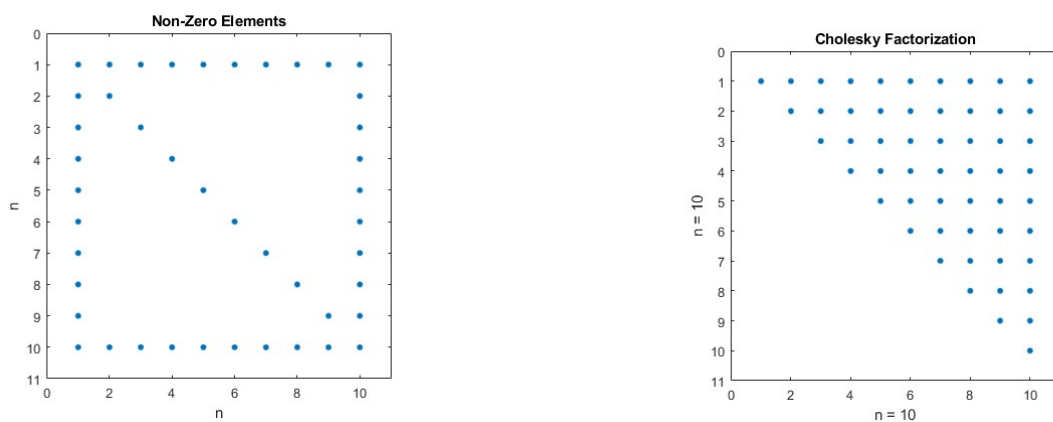


Figure 13: Spy plot for the matrix A vs Cholesky factorization for the matrix A

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?

When $n = 100,000$ is run using `chol()` Matlab gives an error saying there is not enough memory for the operation to take place. This is because the matrix A is a dense matrix and hence it takes a lot of memory to perform all the computing with `chol()`. The complexity for this operation is $O(n^3)$. Hence if this operations is needed to perform either a powerfull machine is required or the matrix A needs to formed as a lower triangular matrix which would make the computation much faster.

7. Degree Centrality [5 points]

Degree Centrality	Author	Index
31	Golub	1
15	Demmel	104
13	Plemmons	86
12	Schreiber	44
12	Heath	81

Table 6: Top 5 authors with highest degree centrality

The above table repersents the Top 5 authors with the highest degree centrality. The degree centrality is the number of edges that are connected to the node. In this case the node is the author. Hence the degree centrality is the number of coauthors that the author has.

8. The Connectivity of the Coauthors [5 points]

Authors	Common Authors
Golub, Moler	Wilkinson, VanLoan
Golub, Saunders	Saunders
TChan, Demmel	Schreiber, Arioli, Duff, Heath

Figure 14: Table of Common Authors

9. PageRank of the Coauthor Graph [5 points]

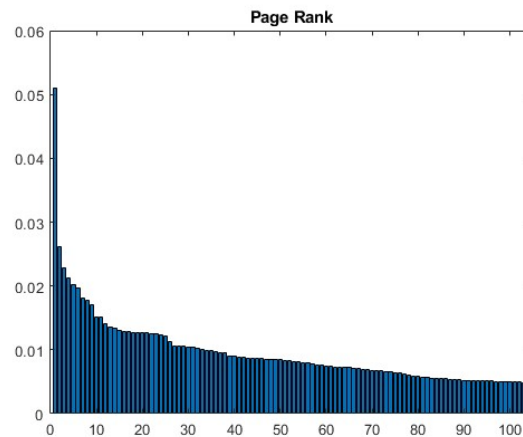


Figure 15: PageRank of all the Authors

In the Figure 15 the PageRank of all the authors is shown. The PageRank1 was modified which used the power method, in which it took the authors as the inputs to compute the pagerank for them. This script can be found as named as *exc9.m*.