Università della Svizzera italiana | Faculty of Informatics | Institute of Computing CI

Numerical Computing **2022**

Student: Alessandro Gobbetti Discussed with: Albert Cerfeda

Solution for Project 1

Due date: Wednesday, 12 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 numerical linear algebra algorithms to solve fundamental linear algebra problems that occur in search engines.

[1] This document is originally based on a SIAM book chapter from *Numerical Computing with Matlab* from Clever B. Moler.

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

1. The Reverse Cuthill McKee Ordering [10 points]

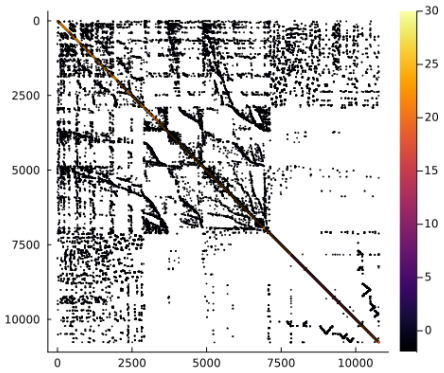

Figure 1: The initial matrix A.

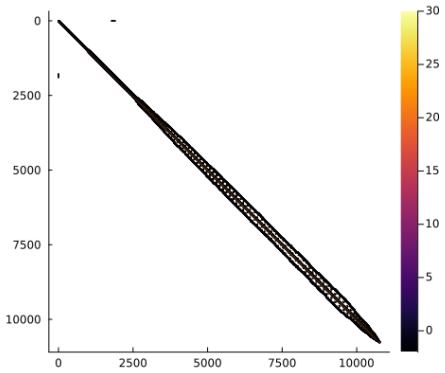

Figure 2: The permuted matrix A.

Given an initial matrix A as in figure 1, we first permute the matrix A using the reverse Cuthill-McKee algorithm:

```
r = symrcm(A[2:end, 2:end]);
prcm = vcat(1, r .+ 1);
A_permuted = A[prcm, prcm];
```

Then we obtain a permuted matrix, figure 2. This martix is a permutation of A where the connections are all ordered to be closest as possible to each other.

We can now compute the Cholesky factors of the two matrices:

```
L = cholesky(A).L;
L_permuted = cholesky(A_permuted).L;
```

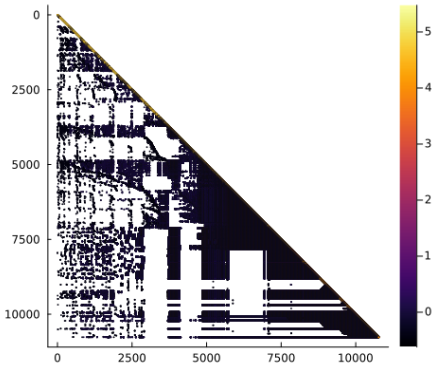

Figure 3: The Cholesky lower triangular matrix of initial matrix A.

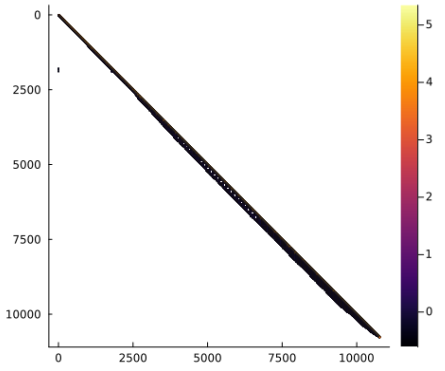

Figure 4: The Cholesky lower triangular matrix of the permuted matrix A.

We get two new matrices, both lower triangular ones.

We can finally compute the number of non-zero elements:

```
nnz(sparse(A))              # 70842
nnz(sparse(A_permuted))     # 70842
nnz(sparse(L))              # 7703122
nnz(sparse(L_permuted))     # 1042133
```

We notice that the first two matrices (figures 1 and 3) have the same number of non-zero elements, they are just reordered, while the lower triangular matrices differ in number of non-zero elements. L has way more non-zero elements than $L_permuted$ since it comes from an ordered matrix.

2. Sparse Matrix Factorization [20 points]

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

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

A)

Given the matrix definition 1 we can construct matrix A for the case $n = 10$:

$$A = \begin{bmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 17 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 19 \end{bmatrix} \tag{2}$$

Matrix A has 44 non-zero elements.

B)

Let's now derive a general formula to compute the number of non-zero entries in matrices defined by 1.

A matrix $A \in \mathbb{R}^{n \times n}$ will have $n \times n$ entries. The non-zero entries are:

- n entries in the first row
- n entries in the last row
- n entries in the first column
- n entries in the last column
- n entries in the diagonal

So $5n$ minus the entries that are counted twice or more:

- A_{11} is counted in the first row, the first column and in the diagonal
- A_{nn} is counted in the last row, the last column and in the diagonal
- A_{1n} is counted in the first row and the last column
- A_{n1} is counted in the last row and the first column

So the total number of non-zero entries is: $5n - 2 - 2 - 1 - 1 = 5n - 6$.

C)

We now define a function `A_construct()` that which takes as input n and returns, as output, the matrix A defined in 1 and its number of non-zero elements nz. See file `ex2c.jl`.

```
function A_construct(n)
    A = sparse(zeros(Int, n,n));
    for i = 1:n
        for j = 1:n
            if (i == 1 || i == n || j ==1 || j == n ) && i != j
                A[i,j] = 1;
            elseif i == j
                A[i,j] = n + i -1;
            end
        end
    end

    nz = nnz(A);

    return A, nz
end
```

By calling it for $n = 10$ we get:

```
n = 10;
A, nz = A_construct(n);
show(stdout, "text/plain", A)
```

```
10×10 SparseMatrixCSC{Int64, Int64} with 44 stored entries:
 10   1   1   1   1   1   1   1   1   1
  1  11   ·   ·   ·   ·   ·   ·   ·   1
  1   ·  12   ·   ·   ·   ·   ·   ·   1
  1   ·   ·  13   ·   ·   ·   ·   ·   1
  1   ·   ·   ·  14   ·   ·   ·   ·   1
  1   ·   ·   ·   ·  15   ·   ·   ·   1
  1   ·   ·   ·   ·   ·  16   ·   ·   1
  1   ·   ·   ·   ·   ·   ·  17   ·   1
  1   ·   ·   ·   ·   ·   ·   ·  18   1
  1   1   1   1   1   1   1   1   1  19
```

The obtained matrix is the same as the matrix 2 computed in exercise 2.A 2.

D)

Here we visualize the e by side the original matrix A and the result of the Cholesky factorization. (`cholesky()` in Julia).

```
L = cholesky(A).L;
plot(spy(A), spy(sparse(L)), layout = (1,2))
```

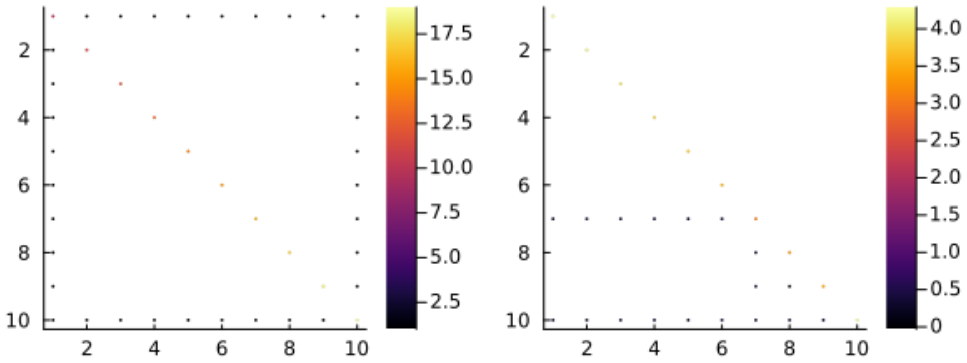

Figure 5: [left] The original matrix A, [right] The result of the Cholesky factorization.

3. Degree Centrality [5 points]

Degree centrality 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.

We can compute the degree centrality of a node just by counting the connections, so summing the 1s in the matrix row.

```
x = vec(sum(A, dims = 1));
i = sortperm(x, rev = true);
```

Degree Centrality **x**	idx **i**	Author **member**	CoAuthors **coautors**
32	1	Golub	Wilkinson, TChan, Varah, Overton, Ernst, VanLoan, ...
16	104	Demmel	Edelman, VanLoan, Bai, Schreiber, Kahan, Kagstrom, ...
14	86	Plemmons	Golub, Nagy, Harrod, Pan, Funderlic, Bojanczyk, ...
13	44	Schreiber	TChan, VanLoan, Moler, Gilbert, Pothen, NTrefethen, ...
13	81	Heath	Golub, TChan, Funderlic, George, Gilbert, Eisenstat, ...

Table 1: Top five author ranked by degree centrality and coauthor's names

4. The Connectivity of the Coauthors [5 points]

Given that the column relative to an author represents all the authors the author collaborated with, to get the common coauthors we can simply compare the two columns and check for common coauthors.

$$\textit{Common coauthors} : A[\textit{author1}] \cap A[\textit{author2}] \tag{3}$$

Then we can just use the formula we derived to compute the common coauthors between two:

- Golum and Moler has as common coauthors: Wilkinson, VanLoan
- Golum and Saunders has as common coauthors: Golub, Saunders, Gill
- TChan and Demmel has as common coauthors: Schreiber, Arioli, Duff, Heath.

5. PageRank of the Coauthor Graph [5 points]

In this section we compute the PageRank value for all authors in `housegraph.mat` by using a modified version of `pagerank.jl` from Project 1.

index **i**	page-rank **x**	in **r**	out **c**	author **authors**
1	0.0630466	31	31	Golub
104	0.0312071	15	15	Demmel
86	0.0269299	13	13	Plemmons
44	0.0248702	12	12	Schreiber
3	0.0235745	10	10	TChan

Table 2: Top five ranked autors.

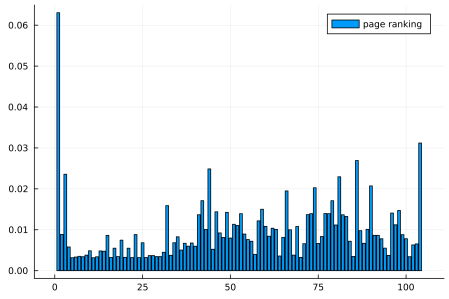

Figure 6: PageRank plot of the Coauthor Graph.

As we can see in Table 5 and figure 6 Golub is by far the most ranked author, since he has contributed with 31 others authors, more than the double of the second ranked author in the list.

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

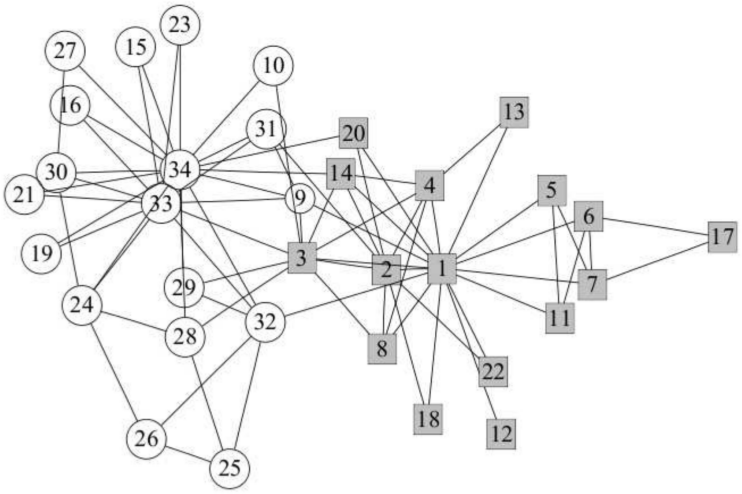

Figure 7: The social network of a karate club at a US university. Image from M. E. J. Newman and M. Girvan, Phys. Rev. E 69,026113(2004)

Figure 7 shows the social network of a karate club at a US university in the 1970s. At the end of the study, a conflict between club members arose. As a consequence, the club formally split into two separate organizations (white circles vs. grey squares).

A)

In this section we rank the nodes in graph 7 by degree centrality. Degree centrality is defined as the number of links incident upon a node (i.e., the number of vertices that a node has).

We can then compute the number of vertices entering and leaving the node as we did in `pagerank.jl`:

```
# c = out-degree, r = in-degree
c = vec(sum(A, dims=1))
r = vec(sum(A, dims=2))
```

Since the matrix A is symmetric we notice that $c = r$. So we can sort them by one of the two.

```
i = sortperm(c, rev = true);
```

In table 6 we can see the top five nodes with the largest degree centrality and their degrees.

member **i**	in-degre/out-degree **c**
34	17
1	16
33	12
3	10
2	9

Table 3: Top five nodes with the largest degree centrality.

B)

Now we rank five nodes with the largest eigenvector centrality. Since the eigenvector returned by the PageRank algorithm corresponds to the weights associated with each node, we can use `page_rank()` funciton to compute it. Then we just have to sort the nodes:

```
i = sortperm(x, rev = true);
```

In table 6 we see the top five nodes with the largest eigenvector centrality.

member **i**	eigenvector **x**
34	0.100919
1	0.0969973
33	0.0716932
3	0.0570785
2	0.0528769

Table 4: Top five nodes with the largest eigenvector centrality.

C)

Rankings in (a) and (b) are identical for the first five members, but it is just a case: eigenvector centrality differs from in-degree centrality: a node receiving many links does not necessarily have a high eigenvector centrality (it might be that all linkers have low or null eigenvector centrality). Moreover, a node with high eigenvector centrality is not necessarily highly linked (the node might have few but important linkers). However, since the matrix is symmetrical, degree centrality and eigenvector centrality rank the nodes in a similar order. Starting from position 6 we see some differences in the ranks.

D)

Let's now use spectral graph partitioning to find a near-optimal split of the network into two groups of 16 and 18 nodes, respectively.

```
A = triu(A, 1);
A = A + A';
deg = sum(A, dims = 1);
L = diagm(vec(deg)) - A;
D, V = eigen(L);
D = diagm(D);
```

Then if we look at the second smallest eigenvector (figures 8 and 9) we can notice that we can split the members in two groups around positions 16-19, where there is a little gap in the sorted values. We can also notice a group of more isolated members at positions 30-34.

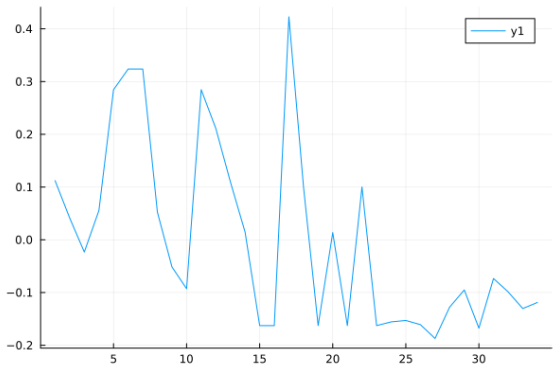

Figure 8: The second smallest eigenvector.

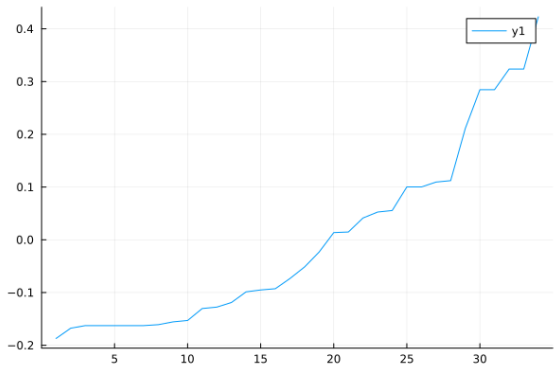

Figure 9: The second smallest sorted eigenvector.

So we can permute the vertices of the graph according to the second smallest ordered eigenvector.

```
p = sortperm(V[:, 2]);
A = A[p, p];
```

And split the two groups:

```
group1 = p[1:16];
group2 = p[17:end];
```

```
Group 1: [10, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34]
Group 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 17, 18, 20, 22, 31]
```

Let's now check if there are more connections within the two groups than between them, we can simply count the edges:

```
Number of edges between the two groups: 23
Number of edges within the first group: 46
Number of edges within the second group: 64
```

We obtained what we wanted, few connection between the groups. It is also interesting to notice that the second group has more internal connections than the first one. Comparing the spectral bisection compare to the real split observed by Zachary in figure 7 we notice that in the Zachary split the square's group corresponds to the Group 2 in spectral bisection plus nodes 9 and 31, so it is pretty similar. The Zachary real split has:

```
Number of edges between the two groups: 25
Number of edges within the first group: 54
Number of edges within the second group: 52
```

So, to conclude, the spectral bisection generates two groups with less connections in between than the Zachary real split.

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.