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

**Numerical Computing**, Fall Semester 2022
**Lecturer:** Prof. O. Schenk
**Assistants:** E. Vecchi, L. Gaedke-Merzhäuser

# Project 2 – Social Networks
# Due date: Wednesday, 26 October 2022, 11:59 PM

The purpose of this mini-project[1] is to learn the importance of sparse linear algebra algorithms to solve fundamental questions in social network analyses. We will use the coauthor graph from the Householder Meeting and the social network of friendships from Zachary's karate club [2]. These two graphs are among the first examples where matrix methods were used in computational social network analysis.

## The Householder XII Meeting

In June, 1993, the UCLA Lake Arrowhead Conference Center in the San Bernardino Mountains, 90 miles east of Los Angeles, was the site of the Householder XII Symposium on Numerical Linear Algebra, organized by Gene Golub[2] and Tony Chan. Nick Trefethen posted a flip chart with Gene Golub's name in the center. He invited everyone present to add their name to the chart and draw lines connecting their name with the names of all their coauthors. The diagram grew denser throughout the week. At the end it was a graph with 104 vertices (or people) and 211 edges. John Gilbert entered the names and coauthor connections into Matlab, creating an adjacency matrix $A$. Using Julia, we can start by retrieving the names and matrix stored in the PARC archive (`housegraph.mat`), which is available on iCorsi.

```
using MAT, SparseArrays, Plots, SymRCM
using Graphs, GraphPlot, Cairo, Fontconfig, Compose
include("drawit.jl")
file = matread("housegraph.mat");
```

The variable `who` shows which variables are stored in the data file.

```
who = collect(keys(file));
```

```
julia> who
117-element Vector{String}:
"Sameh"
"Elden"
"nameperm"
"Borges"
"Greenbaum"
"Arioli"
"Chandrasekaran"
...
```

and `name` shows the participants in the order they were added to the list:

```
name = file["name"];
```

```
julia> name
104-element Vector{String}:
"Golub"
"Wilkinson"
"TChan"
```

[1] This document is originally based on a blog from Cleve Moler, who wrote a fantastic blog post about the Lake Arrowhead graph, and John Gilbert, who initially created the coauthor graph from the 1993 Householder Meeting. You can find more information at `http://blogs.mathworks.com/cleve/2013/06/10/lake-arrowhead-coauthor-graph/`. Most of this assignment is derived from this archived work.

[2] `http://en.wikipedia.org/wiki/Gene_H._Golub`

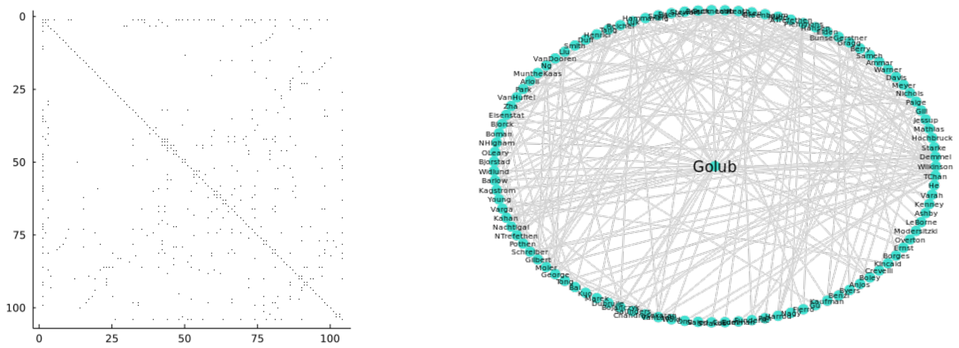

Figure 1: [Left] The coauthor matrix from the meeting (created with `spy(A)`, [Right] The coauthor circle from the meeting.

```
"He"
"Varah"
"Kenney"
"Ashby"
...
```

```
A = file["A"];
```

```
julia> size(A)
(104, 104)
```

Matrix $A$ is 104-by-104 and symmetric. Elements $A_{i,j}$ and $A_{j,i}$ are both equal to one if the people associated with indices $i$ and $j$ are coauthors and equal to zero otherwise. Matrix A is sparse, since most of the attendees are not coauthors and, thus, their corresponding entries have value 0. In order to check the matrix sparsity – i.e., the fraction of nonzero entries over the total number of elements – we can use the following command:

```
sparsity = nnz(A)/prod(size(A));
```

```
julia> sparsity
0.0486
```

As we can observe, only roughly 5% of the entries of matrix A are different from 0 or, in other words, only 5% of all the possible pairs of attendees actually identify coauthors. In the left panel of Figure 1, we show a graphical representation of matrix A by means of a `spy` plot. The latter shows the location of the non-zeros, with Gene Golub in the first row/column, followed by the other authors in the order in which they appeared on Trefethen's flip chart.

## Most Prolific Author

As the adjacency matrix is loaded from the archive, the most prolific coauthor is already in the first column. It was not a surprise to the conference participants that the most prolific coauthor is Gene Golub, one of the organizers.

```
m = findmax(sum(A, dims=1))[2][2];
```

```
julia> m
1
julia> name[m]
"Golub"
```

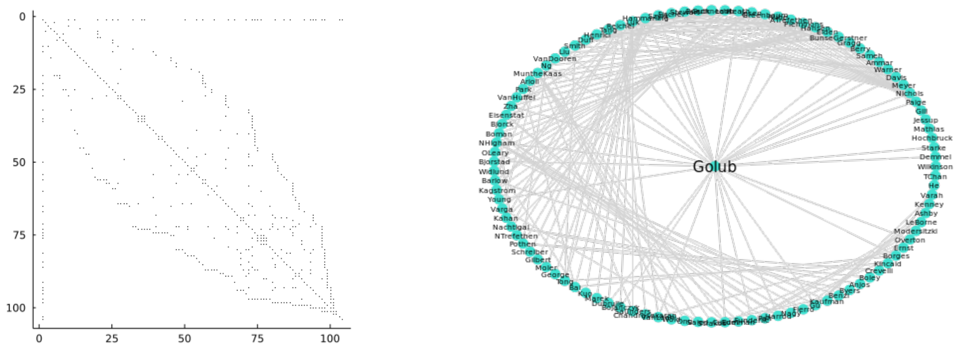

Figure 2: [Left] The reordered coauthor matrix $PAP^T$ from the meeting, [Right] The reordered coauthor circle.

## Circular plot of all the Authors

In the right panel of Figure 1, we created a circular plot with Golub in the center, the other authors around the circumference, and edges connecting the coauthors. If we place the authors around the circumference in the order we retrieve them from the chart, the edges would cross the circle pretty much randomly, as shown in Figure 1.

```
m = findmax(sum(A, dims=1))[2][2];
drawit(A, m, name, "authors.png")
```

## Reorder the Authors

We want to rearrange the authors so that the coauthor connections are as close as possible to the circumference of the circle. This corresponds to a symmetric permutation of matrix $A$, aimed at minimizing, or at least reducing, its bandwidth. In 1969, Elizabeth Cuthill and John McKee described a heuristic for reordering the rows and columns of a matrix to reduce its bandwidth, now known as the *Reverse Cuthill McKee* ordering.

```
r = symrcm(A[2:end, 2:end]);
prcm = vcat(1, r .+ 1);
spy(A[prcm, prcm])
drawit(A[prcm, prcm], m, name, "authors_reorder.png")
```

Figure 2 shows the impact of the reordering $P$ on both matrix $PAP^T$ and on the circular plot.

## Spectral Graph Partitioning and the Laplacian with Julia

In this segment, we will plant an artificial partition in a graph, and then use the eigenvector $v_2$, which is associated to the second smallest eigenvalue $\lambda_2$, to find it. The eigenvector $v_2$ (or *Fiedler vector*) plays an important role in graph partitioning. Its entries indicate a partitioning of the connectivity graph of the matrix it is associated with. In practice, this means that all indices corresponding to vector entries larger than zero belong to one set and all indices corresponding to vector entries smaller than zero belong to the other. The arising partition minimises the number of edges between the two sets. In the rest of this section, we will walk you through the necessary steps and present a practical application. If you would like to read more about this topic, a starting reference can be *Fiedlers theory of spectral graph partitioning*, by Slininger, available on iCorsi. We will start by generating a dataset. In this example, we will be a little more ambitious and use a larger number of vertices.

```
using PrettyTables
using LinearAlgebra
n = 1000;
```

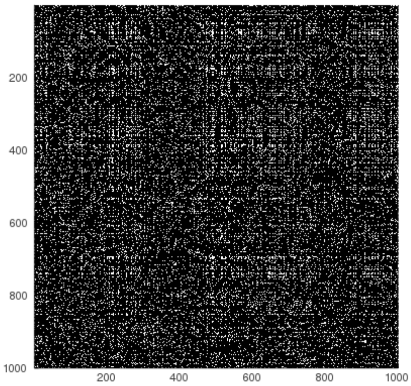

Figure 3: An almost dense artificial dataset.

To plant an artificial partition in our test dataset, we need to determine the vertices in each of the two groups. To do this, we randomly generate a permutation of all the vertices and select one set as group 1 and the rest as group 2. The variable `gs` controls the size of the first group.

```
x = shuffle(collect(1:n));
gs = 450;
group1 = x[1:gs];
group2 = x[gs+1:end];
```

Now, we need to decide on the probabilities of edges within each group and between the two groups. Because we are planting a partition, the probabilities of edges between the two groups should be much lower than the probability of edges within each group. Here, we suppose that group 1 is slightly more connected than group 2.

```
p_group1 = 0.5;
p_group2 = 0.4;
p_between = 0.1;
```

With these probabilities in hand, we can construct our graph. The last few operations symmetrize the matrix.

```
A = sparse(zeros(n, n));
A[group1, group1] = rand(gs, gs) .< p_group1;
A[group2, group2] = rand(n-gs, n-gs) .< p_group2;
A[group1, group2] = rand(gs, n-gs) .< p_between;
```

Next, let us try to visualize the partition with the `spy` command:

```
spy(A);
```

The results are reported in Figure 3. While some might claim to see a partition in this data, we could argue that it is not particularly evident. Now, let us investigate what the eigenvector $v_2$ tells us about this graph. We will use the `eigen` command in Julia to compute all eigenvectors and eigenvalues, and then store them as columns of matrix $V$ and as diagonal entries of matrix $D$, 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);
```

```
julia> D[2,2]
46.469
```

The second smallest eigenvalue $\lambda_2$ is greater than 0 if and only if we are considering a connected graph, and its magnitude gives us an indication on how well-connected the graph is. To visualize our findings, let us plot the associated eigenvector $v_2$, corresponding to the column $V(:,2)$ of the matrix defined above.

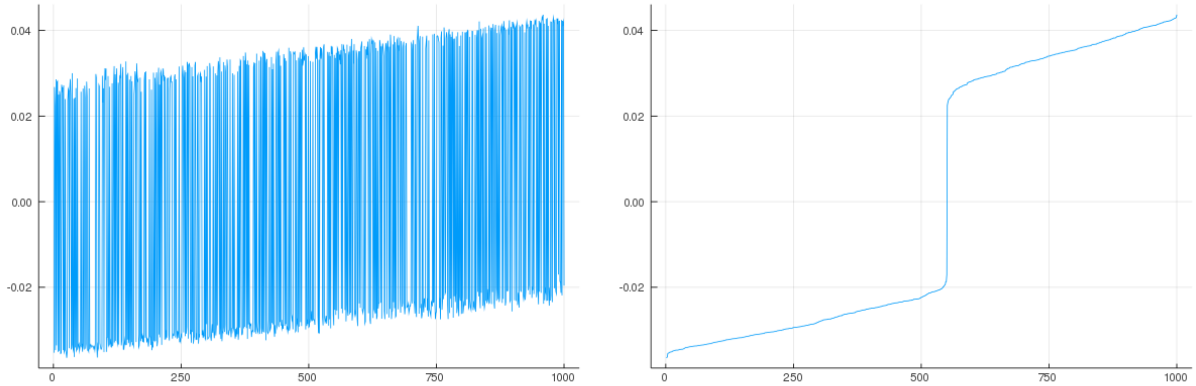

Figure 4: [left] The second smallest eigenvector, [right] The second smallest sorted eigenvector.

```
                    plot(V[:, 2])
```

The result is shown in the left part of Figure 4. The picture is not very useful in helping us understand how the graph is connected. However, we can try to sort the entries of $v_2$ and see if we can extract more information. The result is shown in the right part of Figure 4.

```
                    plot(sort(V[:, 2]))
```

Now, this picture is much more informative than the previous one! We can notice a large gap in the middle of the sorted values. Interestingly enough, the number of points to the right of the gap is the same as `gs`, the size of our planted group. Let us see what happens when we permute the vertices of the graph according to this ordering.

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

In Figure 5 we can finally clearly observe our partitioning. In the spectral analysis of a graph, the second smallest eigenvalue $\lambda_2$ is **very helpful** in finding a partitioning that splits the graph into **two subgraphs**, while simultaneously **minimizing the edges** between these two subgraphs.

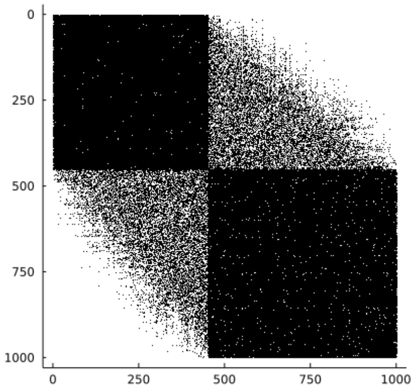

Figure 5: The results after the partitioning of our artificial dataset.

## 1. Solve the following Social Networks problems:

(i) **Reading [0 points]:** Read section 5.7 on *Permutations and ordering strategies, pp. 122-127* from the book *A First Course in Numerical Methods.*

1. **The Reverse Cuthill McKee Ordering [10 points]:** Load matrix ”$A$_SymPosDef.mat” from the dataset. You can reorder (permute) the matrix using the “Reverse Cuthill McKee Ordering” via the Julia SymRCM function `symrcm()`, see SymRCM Julia package documentation for more information. Visualize both the original and Reverse Cuthill McKee permuted matrix and comment on what you observe. Compute the Cholesky factors of both the original matrix and of the permuted matrix. Visualize the Cholesky factors and comment on the number of nonzeros.

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:

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

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

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

b) 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 $5n - 6$.

c) Write a function `A_construct()`, which takes as input $n$ and returns, as output, the matrix $A$ defined in Eq. 1 and its number of non-zero elements *nz*. Test your function in a script `ex3c.jl` 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()`.

d) Using again the `spy()` command, visualize side by side the original matrix $A$ and the result of the Cholesky factorization (`cholesky()` in Julia).

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

(ii) **Reading [0 points]:** Read the introductory section 1 (pp. $1 - 9$), and section 3.3 (pp. $28 - 33$) on *Spectral Bisection* from the report ”Graph Partitioning: A survey” by Ulrich Elsner. This reference will be useful also for the next assignment, where we will consider various graph partitioning techniques in more detail.

3. **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.

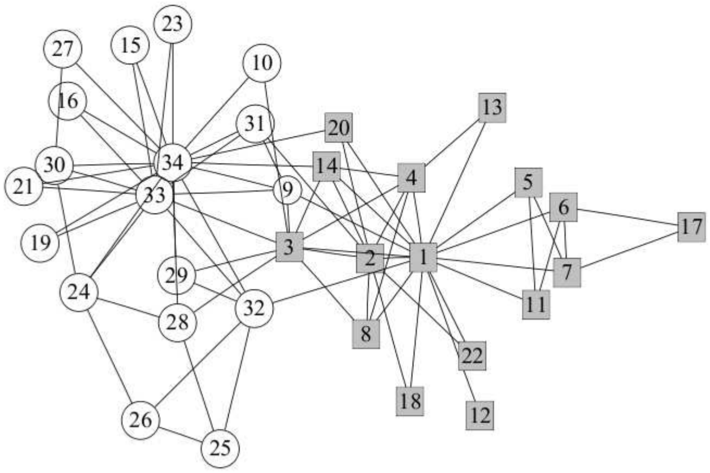

Figure 6: The social network of a karate club at a US university. Image from M. E. J. Newman and M. Girvan, Phys. Rev. E 69,026113(2004)

4. **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.

5. **PageRank of the Coauthor Graph [5 points]:** Compute the PageRank value (e.g., by using a modified version of `pagerank.jl` 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.

6. **Zachary's karate club: social network of friendships between 34 members [40 points]:** Figure 6 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). Please use the adjacency matrix `karate.txt` and answer the following numerical questions.

   a) Write a Julia code that ranks the five nodes with the largest degree centrality. What are their degrees?

   b) Rank the five nodes with the largest **eigenvector centrality**.[3] What are their (properly normalized) eigenvector centralities?

   c) Are the rankings in (a) and (b) identical? Briefly explain the similarities and differences.

   d) Use spectral graph partitioning to find a near-optimal split of the network into two groups of 16 and 18 nodes, respectively. List the nodes belonging to the two groups. How does spectral bisection compare to the real split observed by Zachary (see Fig. 6)?

## Quality of the code and of the report [15 Points]

The highest possible score of each project is 85 points and up to 15 additional points can be awarded based on the quality of your report and code (maximum possible grade: 100 points). Your report should be a coherent document, structured according to the template provided on iCorsi. If there are theoretical questions, provide a complete and detailed answer. All figures must have a caption and must be of sufficient quality (include them either as .eps or .pdf). If you made a particular choice in your implementation that might be out of the ordinary, clearly explain it in the

[3] **Eigenvector centrality** is defined by the principle introduced in Project 1. The eigenvector returned by the PageRank algorithm corresponds to the weights associated with each node.

report. The code you submit must be self-contained and executable, and must include the set-up for all the results that you obtained and listed in your report. It has to be readable and, if particularly complicated, well-commented.

## Additional notes and submission details

Summarize your results and experiments for all exercises by writing an extended LaTeX report, by using the template provided on iCorsi (`https://www.icorsi.ch/course/view.php?id=14666`). Submit your gzipped archive file (tar, zip, etc.) – named `project_number_lastname_firstname.zip/tgz` – **on iCorsi** (see *[NC 2022] Project 2 - Submission Social Networks*) **before the deadline**. Submission by email or through any other channel will not be considered. Late submissions will not be graded and will result in a score of 0 points for that project. You are allowed to discuss all questions with anyone you like, but: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently. Please remember that plagiarism will result in a harsh penalization for all involved parties.

## In-class assistance

If you experience difficulties in solving the problems above, please join us in class either on Tuesdays (16:30-18:15) or on Wednesdays (14:30-16:15) in room C1.03.

## References

[1] "A First Course on Numerical Methods", C. Greif, U. Ascher, SIAM books, pp. 122-127.

[2] 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.