Università della Svizzera italiana | Institute of Computing CI

Numerical Computing, Fall Semester 2023
Lecturer: Dr. E. Vecchi
Assistants: L. Gaedke-Merzhäuser, P. Miotti

Project 1 – PageRank Algorithm and Social Networks
Due date: Wednesday, 11 October 2023, 11:59 PM

The purpose of the first part of this project[1] is to learn the importance of numerical algorithms in the solution of fundamental linear algebra problems frequently occurring in search engines. The second part of the mini-project[2] deals—on the other hand—with the application of sparse linear algebra algorithms to the solution of problems pertaining to social network analysis.

1. Linear Systems and the PageRank Algorithm

One of the reasons why Google™ is such an effective search engine is the Page-Rank™ algorithm developed by Google's founders, Larry Page and Sergey Brin, when they were graduate students at Stanford University. PageRank is determined entirely by the link structure of the World Wide Web. It is recomputed about once a month and does not involve the actual content of any Web pages or individual queries. Then, for any particular query, Google finds the pages on the Web that match that query and lists those pages in the order of their PageRank. Imagine surfing the Web, going from page to page by randomly choosing an outgoing link from one page to get to the next. This can lead to dead ends at pages with no outgoing links, or cycles around cliques of interconnected pages. So, a certain fraction of the time, simply choose a random page from the Web. This theoretical random walk is known as a *Markov chain* or *Markov process*. The limiting probability that an infinitely dedicated random surfer visits any particular page is its PageRank. A page has high rank if other pages with high rank link to it.
Let W be the set of Web pages that can be reached by following a chain of hyperlinks starting at some root page and let n be the number of pages in W. For Google, the set W highly fluctuates over time and is not exactly known, but was estimated to be about 30 billion in 2016[3]. Let G be the n-by-n connectivity matrix of a portion of the Web, that is $g_{ij} = 1$ if there is a hyperlink to page i from page j and zero otherwise. The matrix G can be huge, but it is very sparse. Its j-th column shows the links on the j-th page. The number of nonzeros in G is the total number of hyperlinks in W. Let r_i and c_j be the row and column sums of G:

$$r_i = \sum_j g_{ij} \quad \text{and} \quad c_j = \sum_i g_{ij}. \tag{1}$$

The quantities r_j and c_j are the *in-degree* and *out-degree* of the j-th page. Let p be the probability that the random walk follows a link. A typical value is $p = 0.85$. Then $1 - p$ is the probability that some arbitrary page is chosen and $\delta = (1 - p)/n$ is the probability that a particular random page is chosen. Let A be a n-by-n matrix with elements

$$a_{ij} = \begin{cases} p\, g_{ij}/c_j + \delta & c_j \neq 0, \\ 1/n & c_j = 0. \end{cases} \tag{2}$$

Notice that A comes from scaling the connectivity matrix by its column sums. The j-th column is the probability of jumping from the j-th page to the other pages on the Web. If the j-th page is a dead end – i.e., it has no out-links – then we assign a uniform probability of $1/n$ to all the elements in its column. Most of the elements of A are equal

[1]The first part is originally based on a SIAM book chapter from *Numerical Computing with Matlab* by Cleve B. Moler.
[2]The second part is originally based on a blog by Cleve B. 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/`.
[3]see "Estimating search engine index size variability" by van den Bosch et al., Scientometrics, `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4833824/`

to δ, the probability of jumping from one page to another without following a link. If $n = 4 \cdot 10^9$ and $p = 0.85$, then $\delta = 3.75 \cdot 10^{-11}$. Matrix A is the transition probability matrix of the Markov chain. Its elements are all strictly between zero and one and its column sums are all equal to one. An important result from matrix theory, known as the *Perron-Frobenius theorem*, applies to such matrices: a nonzero solution of the equation

$$x = Ax \tag{3}$$

exists and is unique to within a scaling factor. If this scaling factor is chosen so that

$$\sum_i x_i = 1, \tag{4}$$

then x is the *state vector* of the Markov chain and corresponds to *Google's PageRank*. The elements of x are all positive and less than one. Vector x is the solution to the singular, homogeneous linear system

$$(I - A)x = 0 \tag{5}$$

For modest n, an easy way to compute x in Matlab is to start with some approximate solution, such as the PageRanks from the previous month, or

```
x = ones(n, 1) / n
```

Then simply repeat the assignment statement

```
x = A * x
```

until the difference between successive vectors is within a specified tolerance. This is known as the power method and is about the only possible approach for very large n. In practice, the matrices G and A are never actually formed. One step of the power method would be done by one pass over a database of Web pages, updating weighted reference counts generated by the hyperlinks between pages. The best way to compute PageRank in Matlab is to take advantage of the particular structure of the Markov matrix. Here is an approach that preserves the sparsity of G. The transition matrix can be written

$$A = pGD + ez^\intercal, \tag{6}$$

where D is the diagonal matrix formed from the reciprocals of the out-degrees,

$$d_{jj} = \begin{cases} 1/c_j & c_j \neq 0 \\ 0 & c_j = 0. \end{cases} \tag{7}$$

e is the n-vector of all ones, and z is the vector with components

$$z_j = \begin{cases} \delta & c_j \neq 0 \\ 1/n & c_j = 0. \end{cases} \tag{8}$$

The rank-one matrix $ez^\intercal$ accounts for the random choices of Web pages that do not follow links. The equation

$$x = Ax \tag{9}$$

can be written as

$$(I - pGD)x = \gamma e \tag{10}$$

where

$$\gamma = z^\intercal x. \tag{11}$$

We do not know the value of γ because it depends on the unknown vector x, but we can temporarily take $\gamma = 1$. As long as p is strictly less than one, the coefficient matrix $I - pGD$ is nonsingular and the equation

$$(I - pGD)x = e \tag{12}$$

can be solved for x. Then the resulting x can be rescaled so that

$$\sum_i x_i = 1. \tag{13}$$

Notice that vector z is not involved in this calculation. The following Matlab statements implement this approach.

```
c = sum(G,1);
k = find(c~=0);
D = sparse(k,k,1./c(k),n,n);
e = ones(n,1);
I = speye(n,n);
x = (I - p*G*D)\e;
x = x/norm(x,1);
```

The **power method** [1, 2] can also be implemented in a way that does not actually form the Markov matrix and so preserves sparsity. Compute

```
G = p*G*D;
z = ((1-p)*(c~=0) + (c==0))/n;
```

Start with

```
x = e / n;
```

Then repeat the statement

```
x = G*x + e*(z*x);
```

until x settles down to several decimal places.
It is also possible to use an algorithm known as **inverse iteration** [1, 3] where one defines

```
x = e/n
A = p*G*D + e*z
```

and then the following statement is repeated until convergence given a suitable value α.

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

For details of when and why this method works, please refer to the references cited above. However, we would like to briefly observe what happens when we take $\alpha = 1$ in the expression $(\alpha I - A)$. Since the resulting matrix $(I - A)$ is theoretically singular, with exact computation some diagonal elements of the upper triangular factor of $(I - A)$ would, in principle, be zero and this computation should fail. But with roundoff error, the computed matrix $(I - A)$ is probably not exactly singular. Even if it is singular, roundoff during Gaussian elimination will most likely prevent any exact zero diagonal elements. We know that Gaussian elimination with partial pivoting always produces a solution with a small residual, relative to the computed solution, even if the matrix is badly conditioned. The vector obtained with the backslash operation, $(I - A)\backslash x$, usually has very large components. If it is rescaled by its sum, the residual is scaled by the same factor and becomes very small. Consequently, the two vectors x and Ax equal each other to within roundoff error. In this setting, solving the singular system with Gaussian elimination blows up, but it blows up in exactly the right direction.
Figure 1 is the graph for a tiny example, with $n = 6$ instead of $n = 4 \cdot 10^9$. The pages on the Web are identified by strings known as uniform resource locators, or URLs. Most URLs begin with `https` because they use the hypertext transfer protocol. In Matlab, we can store the URLs as an array of strings in a cell array. The example in Figure 1 can be written as follows by using a 6-by-1 cell array:

```
U = {'https://www.alpha.com'
     'https://www.beta.com'
     'https://www.gamma.com'
     'https://www.delta.com'
     'https://www.rho.com'
     'https://www.sigma.com'}
```

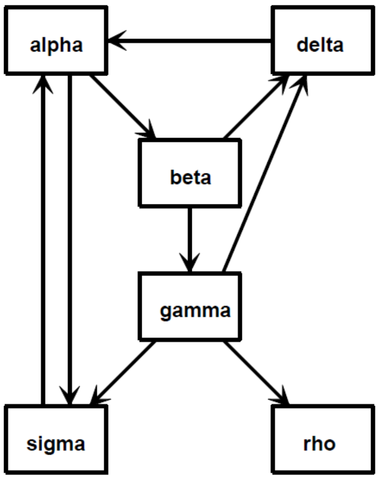

Figure 1: A tiny web graph.

In order to access the string contained in a cell we can simply write $U(k)$, with $k = 1, \ldots, 6$ in this small web graph. Thus, $U(1)$ would, e.g., correspond to the string "https://www.alpha.com".
We can generate the connectivity matrix by specifying the pairs of indices (i, j) of the nonzero elements. Because there is a link to beta.com from alpha.com, the $(2, 1)$ element of G is nonzero. The nine connections are described by

```
i = [ 2 6 3 4 4 5 6 1 1]
j = [ 1 1 2 2 3 3 3 4 6]
```

A sparse matrix is stored in a data structure that requires memory only for the nonzero elements and their indices. This is hardly necessary for a 6-by-6 matrix with only 27 zero entries, but it becomes crucially important for larger problems. The statements

```
n = 6
G = sparse(i,j,1,n,n);
full(G)
```

generate the sparse representation of an n-by-n matrix with ones in the positions specified by the vectors i and j and display its full representation.

```
     0     0     0     1     0     1
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     1     1     0     0     0
     0     0     1     0     0     0
     1     0     1     0     0     0
```

The statement

```
c = full(sum(G))
```

computes the column sums

```
 2     2     3     1     0     1
```

Notice that $c(5) = 0$ because the 5th page, labeled rho, has no out-links. The statements

```
x = (I - p*G*D)\e
x = x/norm(x,1)
```

solve the sparse linear system to produce

```
x =
0.3210
0.1705
0.1066
0.1368
0.0643
0.2007
```

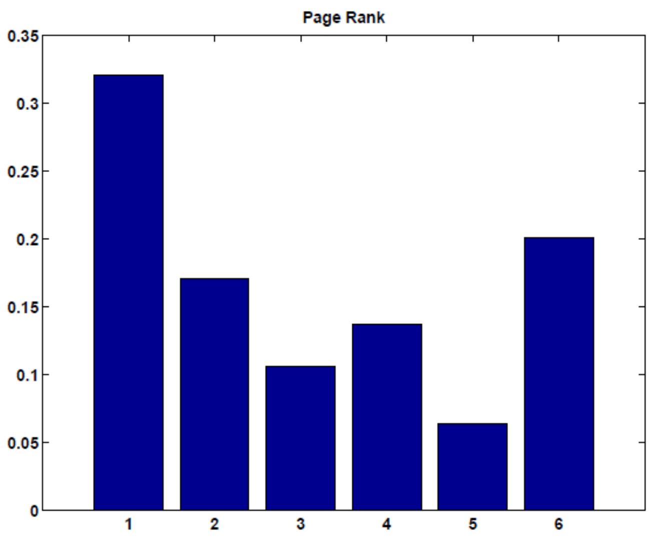

Figure 2: Page Rank for the tiny web graph

The bar graph of x is shown in Figure 2. If the URLs are sorted in PageRank order and listed along with their in- and out-degrees, the result is

```
   page-rank    in      out     url
1  0.3210        2       2      https://www.alpha.com
6  0.2007        2       1      https://www.sigma.com
2  0.1705        1       2      https://www.beta.com
4  0.1368        2       1      https://www.delta.com
3  0.1066        1       3      https://www.gamma.com
5  0.0643        1       0      https://www.rho.com
```

We see that `alpha` has a higher PageRank than `delta` or `sigma`, even though they all have the same number of in-links. A random surfer will visit `alpha` over 32% of the time and `rho` only about 6% of the time.
For this tiny example with $p = .85$, the smallest element of the Markov transition matrix is $\delta = .15/6 = .0250$.

```
A =
  0.0250 0.0250  0.0250  0.8750  0.1667  0.8750
  0.4500 0.0250  0.0250  0.0250  0.1667  0.0250
  0.0250 0.4500  0.0250  0.0250  0.1667  0.0250
  0.0250 0.4500  0.3083  0.0250  0.1667  0.0250
  0.0250 0.0250  0.3083  0.0250  0.1667  0.0250
  0.4500 0.0250  0.3083  0.0250  0.1667  0.0250
```

We can notice that all the columns of matrix A sum to one, thus agreeing with the definition of left stochastic matrix. This mini-project includes the Matlab file `surfer.m`. A statement like

```
[U,G] = surfer('https://www.xxx.zzz',n)
```

starts at a specified URL and tries to surf the Web until it has visited n pages. If successful, it returns an n-by-1 cell array of URLs and an n-by-n sparse connectivity matrix. Surfing the Web automatically is a dangerous undertaking and this function must be used with care. Some URLs contain typographical errors and illegal characters. There is a list of URLs to avoid that includes .gif files and Web sites known to cause difficulties. Most importantly, surfer can get completely bogged down trying to read a page from a site that appears to be responding, but that never delivers the complete page. When this happens, it may be necessary to have the computer's operating system ruthlessly terminate Matlab. With these precautions in mind, you can use surfer to generate your own PageRank examples. The statement

```
[U, G] = surfer('https://inf.ethz.ch/',500);
```

accesses the home page of the Department of Computer Science at the Swiss Federal Institute of Technology in Zurich (ETH) and generates a 500-by-500 test case. The file `ETH500.mat` included in the code folder was generated by using this function in September 2014. In the following, it is indicated as the *ETH500 data set*. The statement

```
spy(G)
```

produces a spy plot (Figure 3) that shows the nonzero structure of the connectivity matrix. The statement

```
pagerank(U, G)
```

computes the page ranks, produces a bar graph (Figure 4) of the ranks, and prints the most highly ranked URLs in PageRank order. The highly ranked pages are

```
>> pagerank(U,G)
        page-rank   in  out  url
57       0.145755    292  1   http://www.zope.org
466      0.049996    19   1   http://purl.org/dc/elements/1.1
39       0.028766    311  0   http://www.ethz.ch
58       0.021863    291  0   http://www.infrae.com
53       0.021202    288  0   http://www.cd.ethz.ch/services/web
101      0.016601    234  6   http://www.hk.ethz.ch/index_EN
72       0.016473    235  0   http://www.ethz.ch/index_EN
486      0.007374    14   2   http://www.handelszeitung.ch
463      0.006896    17   0   http://ns.adobe.com/xap/1.0
```

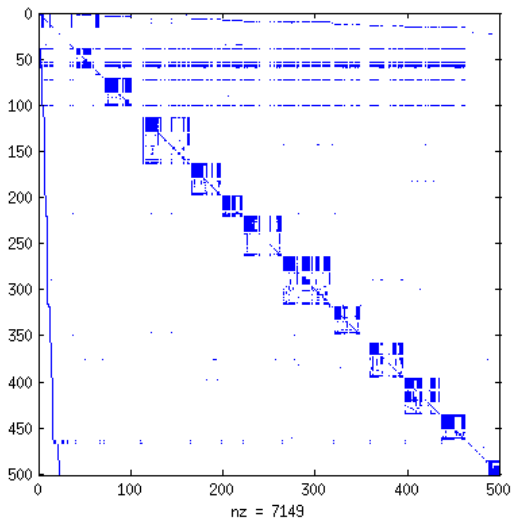

Figure 3: Spy plot of the Department of Computer Science (ETH Zurich) web graph.

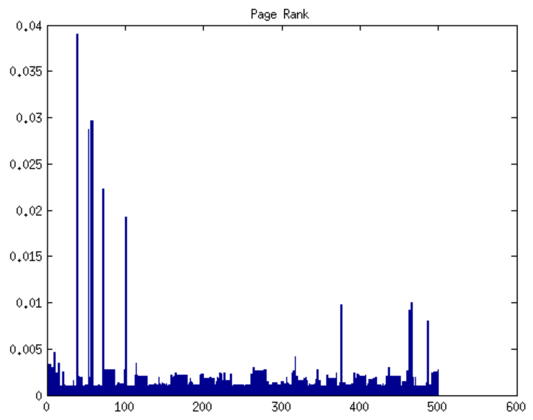

Figure 4: PageRank of the Department of Computer Science (ETH Zurich) web graph.

2. Social Networks and 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[4] 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 Matlab, we can start by retrieving the names and matrix stored in the PARC archive (`housegraph.mat`), which is available on iCorsi.

```
load housegraph
```

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

```
who
```

```
Your variables are:
A          Bunch          Funderlic  Ipsen       Moler      Reichel
ATrefethen BunseGerstner  George     Jessup      MuntheKaas Ruhe
Ammar      Byers          Gilbert    Kagstrom    NHigham    Saied
Anjos      Chandrasekaran Gill       Kahan       NTrefethen Sameh
Arioli     Crevelli       Golub      Kaufman     Nachtigal  Saunders
Ashby      Cullum         Gragg      Kenney      Nagy       Schreiber
Bai        Davis          Greenbaum  Kincaid     Ng         Smith
Barlow     Demmel         Gu         Kuo         Nichols    Starke
Benzi      Dubrulle       Gutknecht  Laub        OLeary     Stewart
Berry      Duff           Hammarling LeBorne     Ong        Strakos
Bjorck     Edelman        Hansen     Liu         Overton    Szyld
Bjorstad   Eisenstat      Harrod     Luk         Paige      TChan
Bojanczyk  Elden          He         Marek       Pan        Tang
Boley      Ernst          Heath      Mathias     Park       Tong
Boman      Fierro         Henrici    Meyer       Plemmons   VanDooren
Borges     Fischer        Hochbruck  Modersitzki Pothen     VanHuffel
VanLoan    Varah          Varga      Warner      Widlund    Wilkinson
Wold       Young          Zha        name        prcm       xy
```

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

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

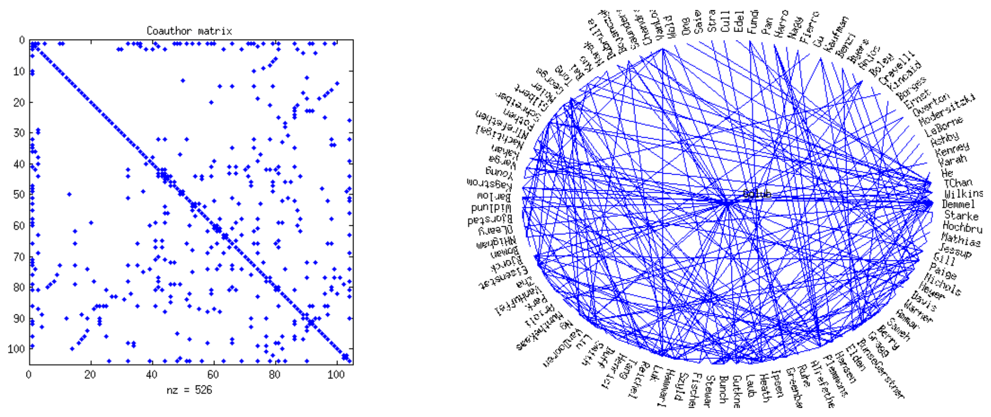

Figure 5: [Left] The coauthor matrix from the meeting (created with `spy(A)`, [Right] The coauthor circle from the meeting.

```
name
```

```
name =
Golub
Wilkinson
TChan
He
Varah
Kenney
Ashby
..
..
```

```
size(A)
```

```
ans =
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:

```
format short
sparsity = nnz(A)/prod(size(A))
```

```
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 5, 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.

2.1. 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.

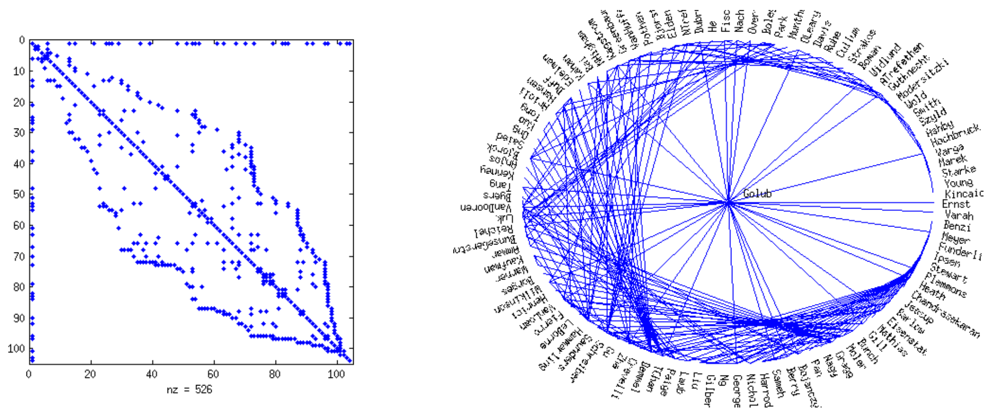

Figure 6: [Left] The reordered coauthor matrix PAP^T from the meeting, [Right] The reordered coauthor circle.

```
m = find(sum(A) == max(sum(A)))
name(m,:)
```

```
m =
1
ans =
Golub
```

2.2. Circular plot of all the Authors

In the right panel of Figure 5, 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 5.

```
drawit;
snapnow
```

2.3. 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 = [1 r+1];
spy(A(prcm,prcm))
title('Coauthor matrix with reduced bandwidth');
drawitrcm;
snapnow
```

Figure 6 shows the impact of the reordering P on both matrix PAP^T and on the circular plot.

Solve the following problems [85 points]:

Preliminary: Read "A First Course on Numerical Methods"

Read chapter 8 and section 5.7 on *Permutations and ordering strategies, pp. 122-127* from the textbook [1], in order to gain a better understanding of the topic summarized above.

Università della Svizzera italiana

Institute of Computing CI

Numerical Computing, Fall Semester 2023
Lecturer: Dr. E. Vecchi
Assistants: L. Gaedke-Merzhäuser, P. Miotti

1. Theoretical questions [15 points]

(a) What are an eigenvector, an eigenvalue and an eigenbasis? ✓

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

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

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

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

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 plot. 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 other is known as a clique. Identify the organizations within the ETH community that are responsible for these near cliques.

3. Connectivity matrix and disjoint subgraphs [10 points] ✓

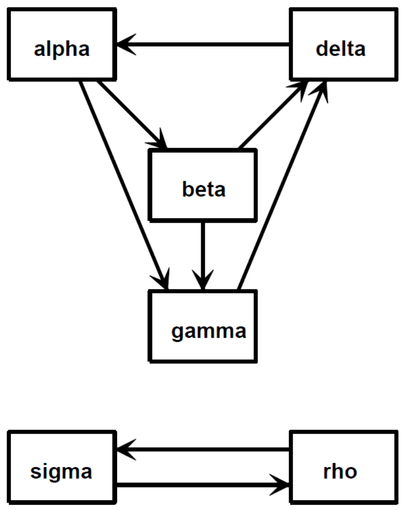

Figure 7: Another tiny Web.

Figure 7 reports the graph of a tiny six-node subset of the Web. Unlike the example reported in Figure 1, in this case there are two disjoint subgraphs.

1. What is the connectivity matrix G? Which are its entries? ✓
2. What are the PageRanks if the hyperlink transition probability p assumes the default value of 0.85? ✓
3. Describe what happens with this example to both the definition of PageRank and the computation done by pagerank in the limit $p \to 1$. ✓

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

The function `pagerank(U,G)` computes PageRanks by solving a sparse linear system. It then plots a bar graph and prints the dominant URLs.

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
   ```

   What is an appropriate test for terminating the power iteration?

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
   ```

   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.

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?

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`.

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 the number of nonzeros.

6. Sparse Matrix Factorization [10 points]

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

$$A_{ij} = \begin{cases} 1, & \text{if } i = 1 \text{ or } i = n \text{ or } j = 1 \text{ or } j = n \text{ and } i \neq j \\ n+i-1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

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

1. Construct matrix A for the case $n = 10$ and explicitly write down its entries. How many non-zero elements does it have? ✓
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 $5n - 6$. ✓
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()`. ✓
4. Using again the `spy()` command, visualize side by side the original matrix A and the result of the Cholesky factorization (`chol()` in Matlab). ✓
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]

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.

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. ✓

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. ✓

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 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=16926`). Submit your gzipped archive file (tar, zip, etc.) – named `project_number_lastname_firstname.zip/tgz` – **on iCorsi** (see *[NC 2023] Project 1 – Submission PageRank Algorithm*) **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 15:30-17:00 or on Wednesdays 13:30-15:00 in room C1.03.

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, pp. 219-229.

[2] Power iteration, `http://en.wikipedia.org/wiki/Power_iteration`

[3] Inverse iteration, `http://en.wikipedia.org/wiki/Inverse_iteration`