Università
della
Svizzera
italiana

Institute of
Computational
Science
ICS

Numerical Computing **2020**

Student: Stefano Gonçalves Simao Discussed with: Stella Born

Solution for Project 1

Due date: Thursday, 8 October 2020, 12:00 AM

Numerical Computing 2020 — 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.

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. Page-Rank Algorithm

1.1. Theory [20 points]

- From page 222 of the book[2]: assuming the dominance of the first eigenvalue, follows that for $j > 1$ we have that

  $$|\frac{\lambda_j}{\lambda_1}|^k \to 0$$

  as

  $$k \to \infty.$$

  Let's consider this sequence $\lambda_n = |\frac{\lambda_j}{\lambda_1}|^n$ that we know converges to $\lambda_\infty = 0$. Now we have

  $$\frac{||\lambda_{n+1} - \lambda_\infty||}{||\lambda_n - \lambda_\infty||} \leq r.$$

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

[2] *A First Course in Numerical Methods* from Uri M. Ascher and Chen Greif

We know that if $r \in (0,1)$ this sequence converges linearly. Using the fact that $\lambda_\infty = 0$ the left part becomes

$$\frac{||\lambda_{n+1}||}{||\lambda_n||} = \frac{|\frac{\lambda_j}{\lambda_1}|^{n+1}}{|\frac{\lambda_j}{\lambda_1}|^n} = |\frac{\lambda_j}{\lambda_1}| \leq r \quad \text{for } j > 1.$$

From the assumption that λ_1 is the dominant eigenvalue we have that

$$|\frac{\lambda_j}{\lambda_1}| < 1$$

which is always in $(0,1)$ and hence it converges linearly. With this we have that the asymptotic error constant is

$$|\frac{\lambda_2}{\lambda_1}|.$$

- In order to guarantee convergence of the power method these assumptions have to be made:
  - we assume that all the vectors in the the matrix A are linearly independent,
  - the eigenvalues are sorted in decreasing order in terms of their magnitude

    $$|\lambda_1| > |\lambda_j|, \quad j = 2, ..., n,$$

  - starting vector $\mathbf{v}_0$ has a nonzero component in the direction of $\mathbf{x}_1$ (the eigenvector associated with the dominant eigenvalue λ_1).
- The inverse iteration uses the shift and invert technique, which gives rise to a substantially faster convergence when we an approximation to a corresponding eigenvalue is already known. The idea is as follows. If the eigenvalues of A are λ_j, the eigenvalues of $A - \alpha I$ are $\lambda_j - \alpha$, and the eigenvalues of $B = (A - \alpha I)^{-1}$ are $1/(\lambda_j - \alpha)$. Then we apply the power method to B. This works in general for computing any eigenvalue, as long as one knows roughly what it is and allows to easily overcome the difficulty of the power method where more than one simple eigenvalue is dominant.
- With the Power Method, the cost of each iteration involves a matrix-vector product $\mathcal{O}(n^2)$, whereas the Inverse Iteration requires solving a linear system $\mathcal{O}(n^3)$. As we can see, convergence of the Inverse Iteration must be very fast for it to be effective.
- The Rayleigh quotient is defined for any given vector by

  $$\mu(\mathbf{v}) = \frac{\mathbf{v}^T A \mathbf{v}}{\mathbf{v}^T \mathbf{v}}$$

  and if we normalize $\mathbf{v_k}$, we have

  $$\mu(\mathbf{v_k}) = \mathbf{v}_k^T A \mathbf{v}_k.$$

  Where if $\mathbf{v}$ is an eigenvector, then $\mu(\mathbf{v})$ would simply give the associated eigenvalue. But if $\mathbf{v}$ is not an eigenvector, then the Rayleigh quotient gives us the best approximation in the least square sense. With this, we may choose the shift α dynamically, i.e., $\alpha = \alpha_k$, setting it to be the Rayleigh quotient and thus the convergence order is better than linear (in most cases it is cubic).

1.2. Other webgraphs [10 points]

- www.multiplayer.it:

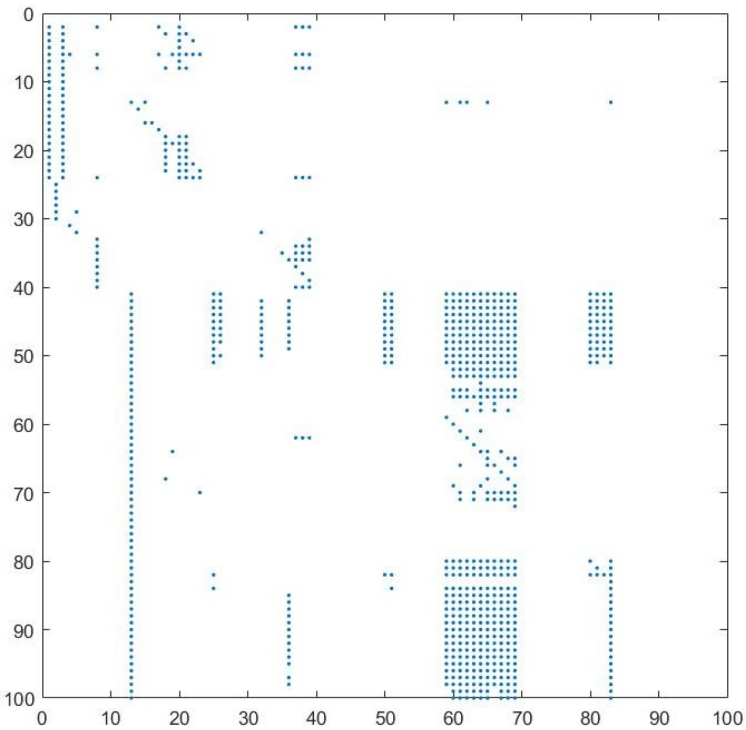


Figure 1: multiplayer.it sparse matrix with depth 100

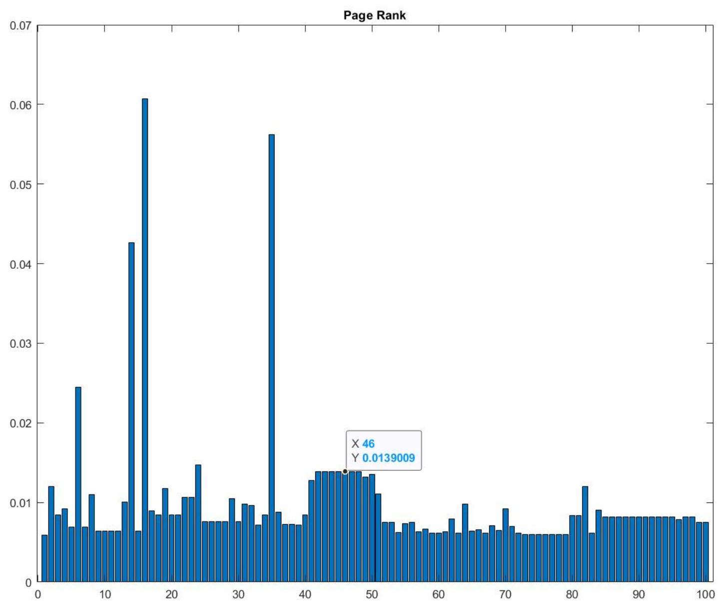


Figure 2: multiplayer.it Page Rank

Page-Rank	In	Out	URL
0.0607	4	1	https://www.instagram.com/multiplayer.it
0.0562	5	1	https://t.me/multiplayershop
0.0426	3	1	https://twitter.com/multiplayerit
0.0245	13	0	https:
0.0147	10	0	https://sb
0.0139	22	0	https:live.adyen.com
0.0139	22	0	https:integration-facebook.payu.in
0.0139	22	0	https:facebook.payulatam.com
0.0139	22	0	https:secure.payu.com
0.0139	22	0	https:facebook.dlocal.com

As we can see in Fig.1, we have two blocks in 40-55 x 60-70 and 80-100 x 60-70. These are all pages related to facebook.com. Even though the Facebook pages are heavily connected, they are not in the top 5 pages in the Page-rank analysis, that's because they have a low rank themselves. A small group of these Facebook pages appear after the top five having the same page-rank. This website is full of links to Facebook in order to be more visible in the social media. The two pages with best rank are the social media links of the website.

- www.fabiokaeppeli.ch:

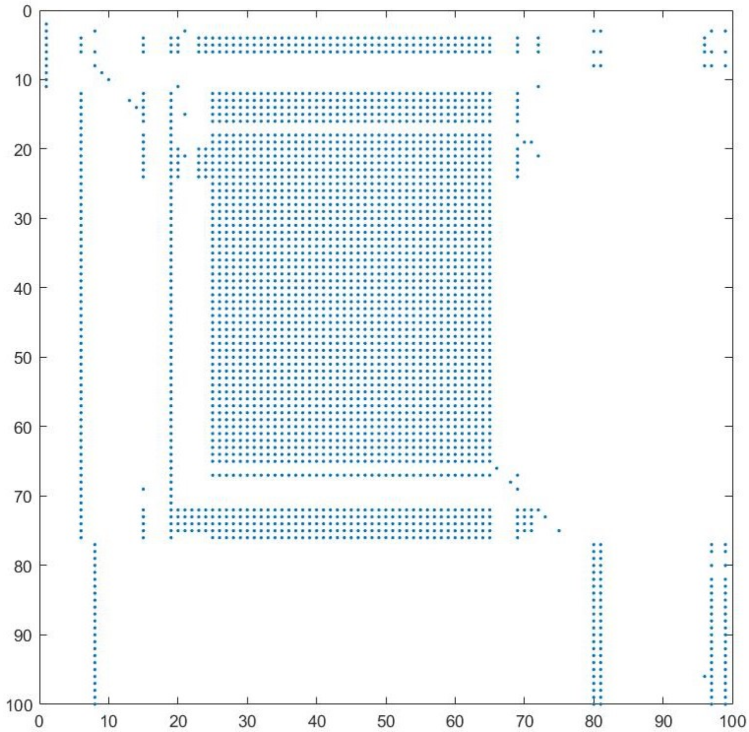


Figure 3: fabiokaeppeli.ch sparse matrix with depth 100

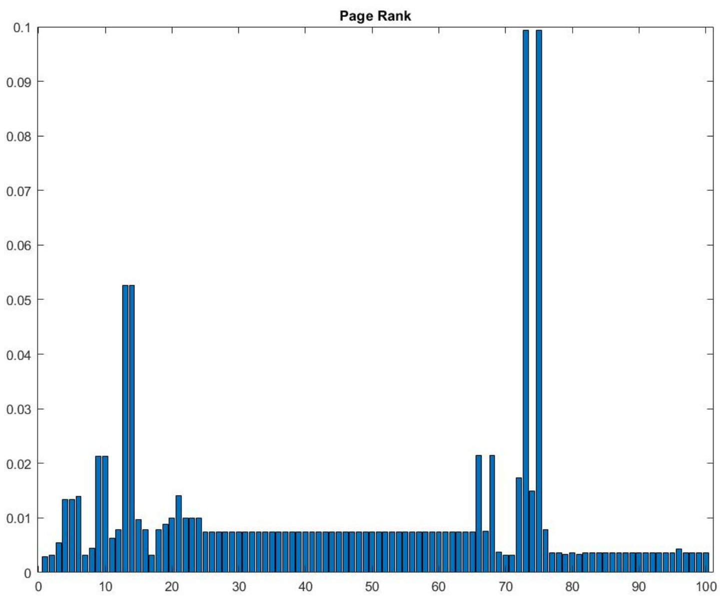


Figure 4: fabiokaeppeli.ch Page Rank

Page-Rank	In	Out	URL
0.0993	53	1	https://publiccode.eu
0.0993	53	1	https://www.facebook.com/WordPress
0.0527	46	1	https://developer.wordpress.org/feed
0.0527	46	1	https://developer.wordpress.org/comments/feed
0.0215	3	1	https://github.com/WP-API/docs/edit/master/index.md
0.0215	3	1	https://en.wikipedia.org/wiki/JSON
0.0212	2	1	https://www.facebook.com/fabio.kaeppeli
0.0212	2	1	https://twitter.com/fabiokep
0.0174	53	6	https://wordpressfoundation.org/donate
0.0174	52	0	https://twitter.com/WordPress

This is a small website and it was made with WordPress, we can clearly see that from the surfer analysis. We can see a big block: links from 20 to 70 link to each other because they are part of developer.wordpress.org. Those links all have the same rank because, as said, they come from the same website, and the rank is quite low. We can see from the Page Rank that publicode.eu and the Facebook page of WordPress are on top with a very high score compared to the others. Also, a lot of pages have a link to themselves, since the diagonal has a lot of nonzero entries.

- www.stackexchange.com:

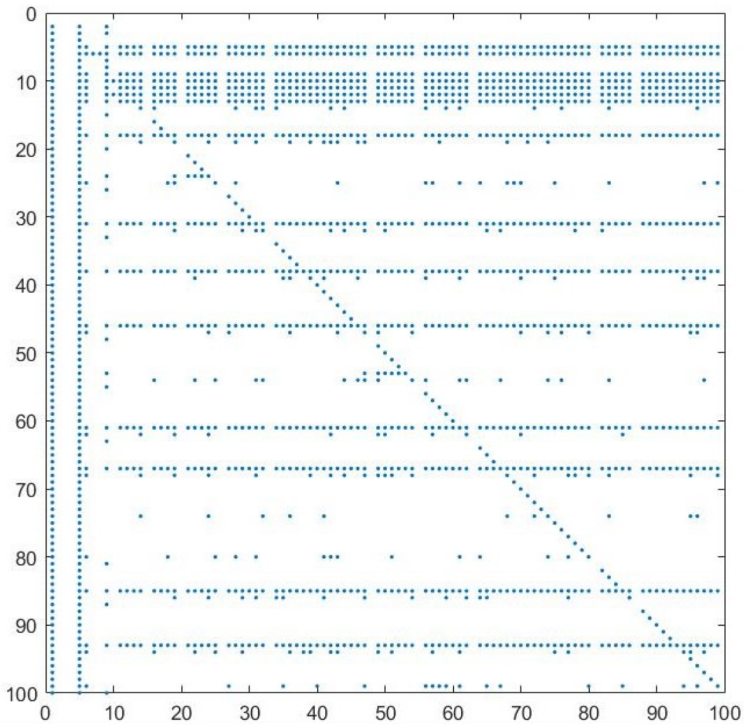


Figure 5: stackexchange.com sparse matrix with depth 100

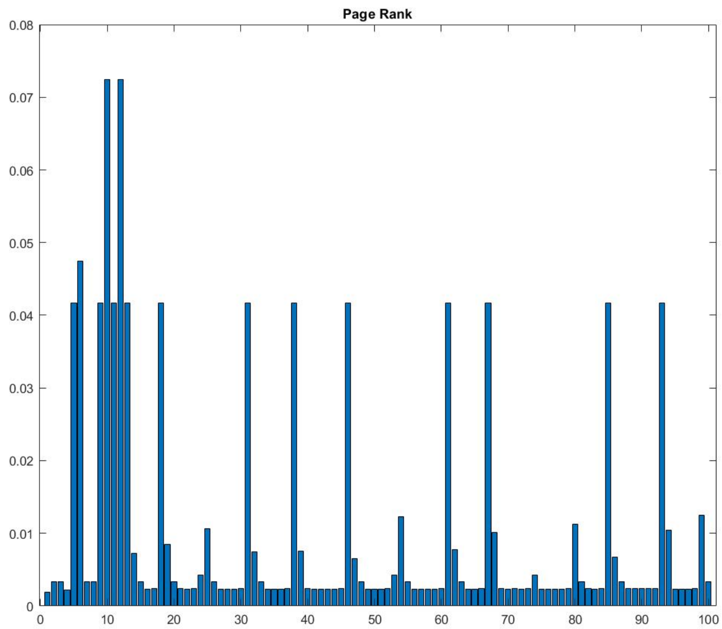


Figure 6: stackexchange.com Page Rank

Page-Rank	In	Out	URL
0.0725	85	2	https://stackoverflow.blog
0.0725	85	15	https://stackoverflow.com/legal/privacy-policy
0.0474	86	22	https://meta.stackexchange.com
0.0417	84	99	https://stackexchange.com
0.0417	84	31	https://stackexchange.com/sites
0.0417	84	15	https://stackoverflow.com/legal/cookie-policy
0.0417	84	15	https://stackoverflow.com/legal/terms-of-service/public
0.0417	84	17	https://money.stackexchange.com
0.0417	84	22	https://academia.stackexchange.com
0.0417	84	16	https://worldbuilding.stackexchange.com

This is a very big website. We can see from the surfer analysis that we have some patterns. The websites from 5 to 15 are linked by almost every page and have a fairly high Page Rank because they come from stackoverflow.com and stackexchange.com (stackexchange is the network of websites and stackoverflow is one of those). Almost every page has a link to itself, since the diagonal is almost non zero everywhere. We can also see that the pages that are linked a lot (for example pages 31, 38, 46, 61,...) also have a fairly high rank and it is the same.

1.3. Connectivity matrix and subcliques [10 points]

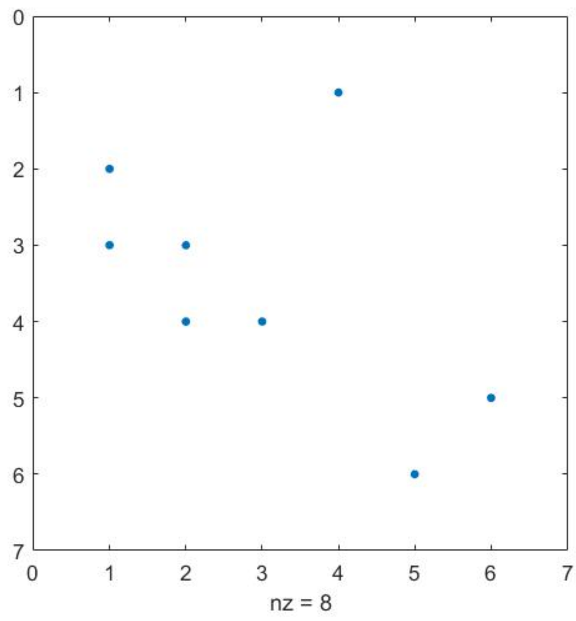
The near cliques that we can see are caused by links that share the same domain.

- Around index 80: www.baug.ethz.ch
- Around index 120: www.mat.ethz.ch
- Around index 170: www.mavt.ethz.ch
- Around index 205: www.biol.ethz.ch
- Around index 230: www.chab.ethz.ch
- Around index 270: www.math.ethz.ch
- Around index 325: www.erdw.ethz.ch
- Around index 365: www.usys.ethz.ch
- Around index 400: www.mtec.ethz.ch
- Around index 445: www.gess.ethz.ch
- Around index 495: www.bilanz.ch

1.4. Connectivity matrix and disjoint subgraphs [10 points]

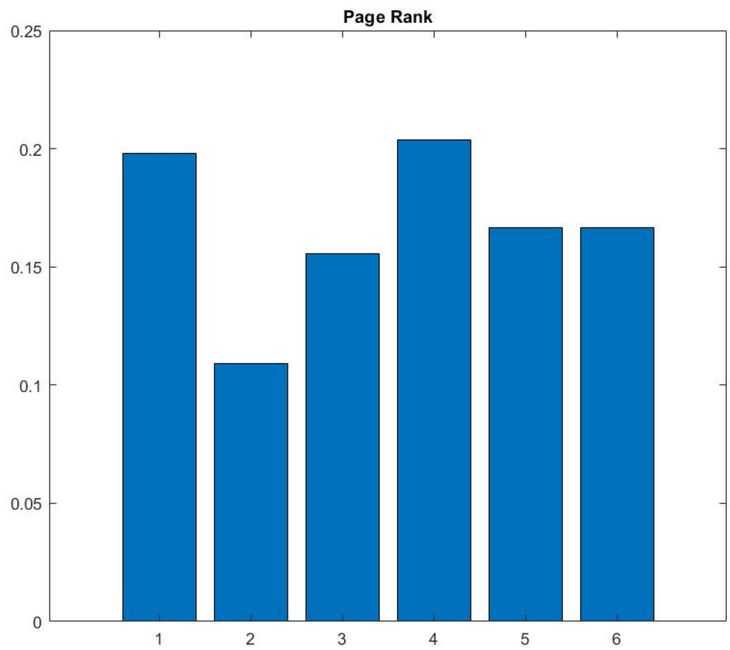
1. The connectivity matrix G:

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



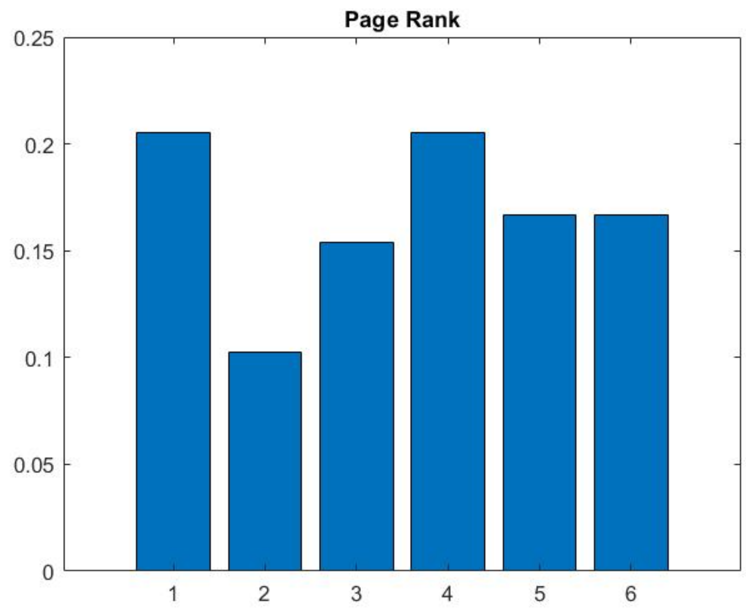
2. Page ranks with default p = 0.85:

Page-Rank	In	Out	URL
0.2037	2	1	http://www.delta.com
0.1981	1	2	http://www.alpha.com
0.1667	1	1	http://www.rho.com
0.1667	1	1	http://www.sigma.com
0.1556	2	1	http://www.gamma.com
0.1092	1	2	http://www.beta.com



3. Let's try using pagerank with p = 0.99999:

Page-Rank	In	Out	URL
0.2051	2	1	http://www.delta.com
0.2051	1	2	http://www.alpha.com
0.1667	1	1	http://www.rho.com
0.1667	1	1	http://www.sigma.com
0.1538	2	1	http://www.gamma.com
0.1026	1	2	http://www.beta.com



As we can see, alpha has the same rank as delta. That's because with probability $p \to 1$, the pagerank algorithm doesn't let you choose another page randomly:

$$\delta = \frac{1-p}{n}$$
$$\delta = \lim_{p \to 1} \frac{1-p}{n} = 0,$$

with δ the probability that a particular random page is chosen.

1.5. PageRanks by solving a sparse linear system [50 points]

1. ✓

2. We want x with $x = Ax$. Knowing that the eigenvalue is $\lambda = 1$, we use as a stop criterion

$$||Ax^k - x^k|| \leq \epsilon$$

with ϵ the machine precision. With $A^k x_0 = x^k$ we have $Ax^k = AA^k x_0 = A^{k+1} x_0$ and:

$$||x - old\_x|| \leq \epsilon.$$

```
% ---------------------------------- EX5.2 ------------------------------
G = p*G*D;
z = ((1-p) * (c~=0) + (c==0)) /n;
x = e/n;
x_old = 0;
while norm(x - x_old) > eps
    x_old = x;
    x = G*x + e*(z*x);
end
% ---------------------------------------------------------------------------
```

3. Having in mind that our dominant λ is equal to one, we have that setting $\alpha = 1$ gives us the fastest convergence, likely much faster than the Power method where more than one

eigenvalue is dominant. As we go further away from 1 ($\alpha = 0.9$ and $\alpha = 0.8$) the convergence slows down.
In order to prevent a division by zero we can use Gaussian elimination with partial pivoting, this always produces a solution with a small residual that prevents any exact zero divisions. Another way can be to approximate $\alpha = 1$ with something like $\alpha = 0.9999$, preventing some diagonal element from being exactly 0.

4. All three algorithms give the same result:

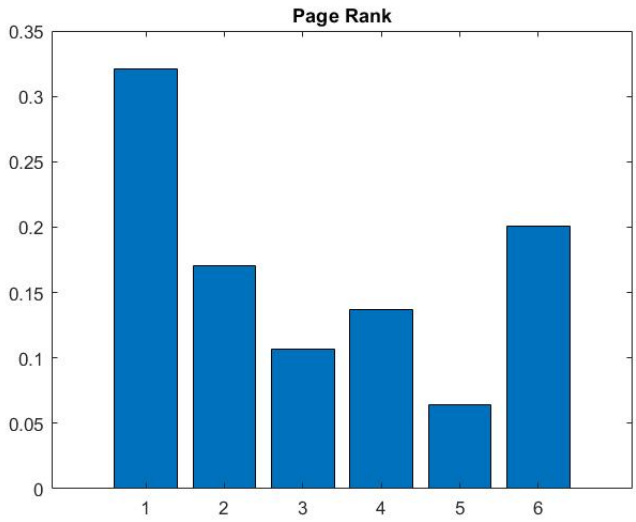


Figure 7: pagerank.m

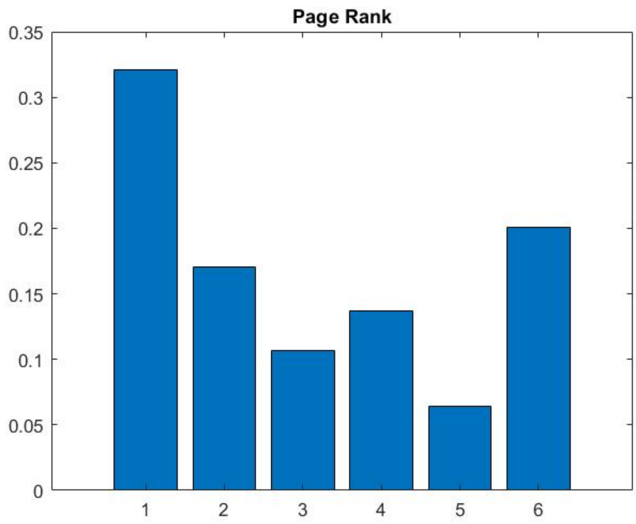


Figure 8: pagerank1.m

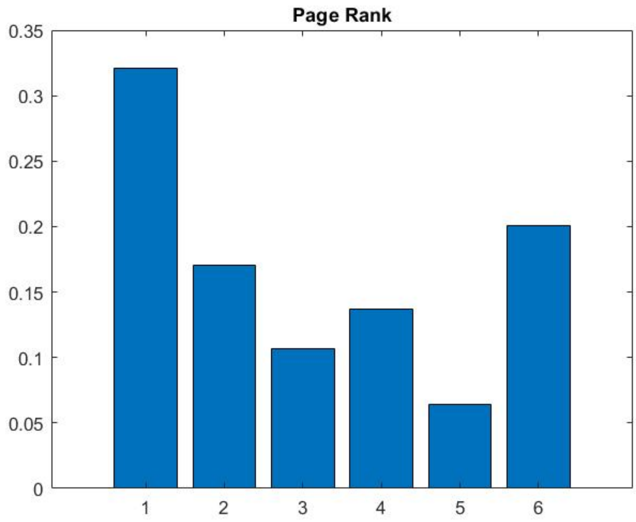


Figure 9: pagerank2.m

With the following PageRanks:

Page-Rank	In	Out	URL
0.3210	2	2	http://www.alpha.com
0.2007	2	1	http://www.sigma.com
0.1705	1	2	http://www.beta.com
0.1368	2	1	http://www.delta.com
0.1066	1	3	http://www.gamma.com
0.0643	1	0	http://www.rho.com

5. Also for figure 5 the three algorithms produce the same result:

Page-Rank	In	Out	URL
0.2037	2	1	http://www.delta.com
0.1981	1	2	http://www.alpha.com
0.1667	1	1	http://www.rho.com
0.1667	1	1	http://www.sigma.com
0.1556	2	1	http://www.gamma.com
0.1092	1	2	http://www.beta.com

There is some difference between task 5.5 and 5.4:

Page-Rank - Fig5	Page-Rank - Fig1	URL
0.1981	0.3210	http://www.alpha.com
0.1092	0.1705	http://www.beta.com
0.1556	0.1066	http://www.gamma.com
0.2037	0.1368	http://www.delta.com
0.1667	0.0643	http://www.rho.com
0.1667	0.2007	http://www.sigma.com

We can clearly see that two different graphs produce very different Pageranks. For example, alpha loses one incoming link and goes from first to second place. We can conclude that even slightly changing the links between the pages can have a huge impact in the Pageranks.

6. For the website multiplayer.it the three algorithms produced the same result as in exercise 2. pagerank1.m required 168 iterations. pagerank2.m required 2 iterations.

   For the website fabiokaeppeli.ch the three algorithms produced the same result as in exercise 2. pagerank1.m required 126 iterations. pagerank2.m required 2 iterations.

   For the website stackexchange.com the three algorithms produced the same result as in exercise 2. pagerank1.m required 38 iterations. pagerank2.m required 2 iterations.

   The advantage of the power method is that we exploit the sparsity matrix without actually forming a Markov matrix. This way we don't have to solve a linear system ($\mathcal{O}(n^3)$ and for real network applications this is simply unfeasible) but just the addition of two multiplication of sparse matrices and a vector ($\mathcal{O}(n^2)$) which is repeated until the stop criterion is reached. The ratio of convergence is given by $|\frac{\lambda_2}{\lambda_1}|$, if λ_2 is close in magnitude to λ_1 the convergence may be extremely low. In pagerank2 we implemented the inverse method that gives much faster convergence (as we can see from the previous results) at the cost of having to solve a linear system at each iteration. Even if pagerank2 needs to compute a linear system as in the original pagerank, using the shift and invert technique and with α very close to λ_1, convergence of pagerank2 is expected to be very fast and much faster than pagerank and pagerank1 (but only for small problems).