MATH 365 Technical Report 1: Katz Centrality Rankings

Erik Hager Ross Kogel Vichearith Meas

March 13, 2022

Abstract

This report uses R code to implement the Katz method for ranking centrality of nodes, an algorithm which uses adjacency matrix multiplication to measure the number of edges leading to a node in a directed graph from other nodes, added to the respective number of edges leading to those nodes, and the respective number of edges leading to those nodes, and so on. Each new generation of nodes is multiplied by a reduction factor alpha raised to a higher power. Particularly, we use Jacobi iteration to solve the resulting matrix equation from setting up a Katz network structure, in order to find rankings of network members' importance.

1 Katz Centrality

1.1 Overview

The Katz matrix derived from an adjacency matrix A is given by

$$\begin{aligned}
K &= \alpha(A^T) + \alpha^2(A^T)^2 + \alpha^3(A^T)^3 + \cdots \\
&= R + R^2 + R^3 + \cdots && \text{Assuming } R = \alpha(A^T) \\
&= \sum_{i=1}^{\infty} R^i
\end{aligned}$$

In order for $R = \alpha(A^T)$ to converge, we need the eigenvalues of R satisfy $|\lambda| < 1$.

Operating with the assumption that $R = \alpha(A^T)$ converges, the Katz matrix can be written as:

$$\begin{aligned}
K &= (I + R + R^2 + R^3 + \cdots)R && \text{For an identity matrix } I \\
&= \frac{R}{I - R} \\
&= \frac{\alpha A^T}{I - \alpha A^T} \\
&= (I - \alpha A^T)^{-1}\alpha A^T
\end{aligned}$$

To find the Katz status index p, we multiply K by the 1-vector:

$$\begin{aligned}p &= K1 \\ &= (I - \alpha A^T)^{-1}\alpha A^T 1 \\ &= (I - \alpha A^T)^{-1}\alpha d && , \quad d = A^T 1 \\ &= [\alpha(\alpha^{-1}I - A^T)]^{-1}\alpha d \\ &= \alpha^{-1}({}^{-1}I - A^T)^{-1}\alpha d1 \\ &= (\alpha^{-1}I - A^T)^{-1}d \\ &= B^{-1}d && , \quad B = \alpha^{-1}I - A^T\end{aligned}$$

That is, p is also the solution to the matrix equation $Bx = d$.

1.2 Solving Katz Equations via Jacobi Iteration

The Jacobi method is an iterative method for solving $Bx = d$ by decomposing B into a diagonal matrix D and an off-diagonal matrix R:

$$\begin{aligned}Bx &= d \\ (D + R)x &= d \\ Dx &= d - Rx \\ x &= D^{-1}(d - Rx)\end{aligned}$$

Starting with an initial guess x_0, the following iterative form should eventually allow x to converge to the Katz status index (given an appropriate alpha choice):

$$x_i = D^{-1}(d - Rx_{i-1})$$

2 Workplace Influence Network

2.0.1 Adjacency Matrix

One example of a Katz centrality problem is in the network in Figure 1 showing who employees in one workplace turn to for advice. The corresponding adjacency matrix represents the connection the between nodes of this network. Each connection to and from a named node corresponds to an index in a corresponding row and column in the matrix.

2.0.2 The Role of Alpha and In-Degree Rank

The attenuation factor α measures the extent to which indirect connections between nodes are penalized. Lower values of α assign increasingly lesser weight to connections that require a greater number of paths to reach their target.

The final Katz index reached represents a ranking of the members of the network based on the number of in-degrees, or "arrows toward" them, using any number of indirect links. Thus, the member with the greatest value in the Katz index is interpreted to be the most important in terms of seeking advice.

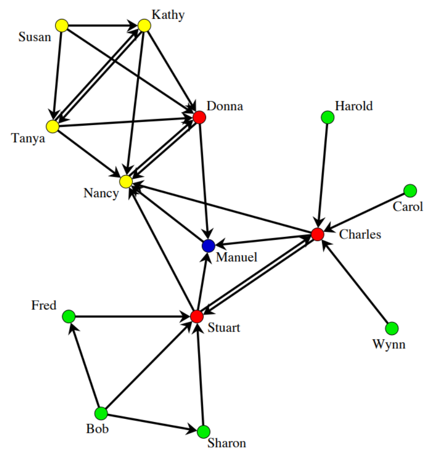


Figure 1: Workplace Influence Graph

$$\begin{pmatrix}
. & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\
1 & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\
1 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
1 & . & 1 & . & 1 & . & . & . & . & . & . & . & . & . \\
1 & . & 1 & 1 & . & . & . & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & . & 1 & . & . & . & . & . & . & . \\
1 & 1 & . & . & . & 1 & . & . & . & . & . & . & . & . \\
. & . & . & 1 & . & . & . & . & . & . & . & . & . & . \\
. & . & . & 1 & . & . & . & . & . & . & . & . & . & . \\
. & . & . & 1 & . & . & . & 1 & 1 & . & . & . & . & . \\
. & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\
. & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\
. & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & 1 & 1 & . & . & . & . & . & . & .
\end{pmatrix}$$

Figure 2: Adjacency Matrix Representation

2.0.3 Jacobi Iteration Unfolding

Given the following vector and matrix definitions,

$$B = \begin{pmatrix} 2 & -1 & \cdots & 0 & 0 \\ -1 & 2 & \cdots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 2 & 0 \\ 0 & 0 & \cdots & 0 & 2 \end{pmatrix} D = \begin{pmatrix} 2 & 0 & \cdots & 0 & 0 \\ 0 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 2 & 0 \\ 0 & 0 & \cdots & 0 & 2 \end{pmatrix} R = \begin{pmatrix} 0 & -1 & \cdots & 0 & 0 \\ -1 & 0 & \cdots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} d = \begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix} x_0 = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$$

A choice of $\alpha = 0.5$ leads to the following series of iterations.

$$x_i = D^{-1}(b - Rx_{i-1})$$

$$x_1 = \begin{pmatrix} 0.5 & 0.0 & \cdots & 0.0 & 0.0 \\ 0.0 & 0.5 & \cdots & 0.0 & 0.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.0 & 0.0 & \cdots & 0.5 & 0.0 \\ 0.0 & 0.0 & \cdots & 0.0 & 0.5 \end{pmatrix} \cdot \left(\begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & -1 & \cdots & 0 & 0 \\ -1 & 0 & \cdots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \right) = \begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

$$x_2 = \begin{pmatrix} 0.5 & 0.0 & \cdots & 0.0 & 0.0 \\ 0.0 & 0.5 & \cdots & 0.0 & 0.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.0 & 0.0 & \cdots & 0.5 & 0.0 \\ 0.0 & 0.0 & \cdots & 0.0 & 0.5 \end{pmatrix} \cdot \left(\begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & -1 & \cdots & 0 & 0 \\ -1 & 0 & \cdots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 12.5 \\ 7.0 \\ \vdots \\ 0.0 \\ 0.0 \end{pmatrix}$$

$$\vdots$$

$$x_{25} = \begin{pmatrix} 0.5 & 0.0 & \cdots & 0.0 & 0.0 \\ 0.0 & 0.5 & \cdots & 0.0 & 0.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.0 & 0.0 & \cdots & 0.5 & 0.0 \\ 0.0 & 0.0 & \cdots & 0.0 & 0.5 \end{pmatrix} \cdot \left(\begin{pmatrix} 6 \\ 4 \\ \vdots \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & -1 & \cdots & 0 & 0 \\ -1 & 0 & \cdots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 24.798578 \\ 16.398927 \\ \vdots \\ 0.000000 \\ 0.000000 \end{pmatrix} \right) = \begin{pmatrix} 24.799058 \\ 16.399289 \\ \vdots \\ 0.000000 \\ 0.000000 \end{pmatrix}$$

2.0.4 Alpha Values and Convergence

In order for the Jacobi iteration to converge, an attenuation factor must be chosen such that it is less than the reciprocal of the maximum eigenvalue of B. The maximum eigenvalue of B in this case is about 1.34, so any α less than $1/1.34 = 0.75$ will let the algorithm converge. Figure 3 shows the algorithm's error descending toward 0 as it iterates, for various convergent choices of α.

For values of α greater than 0.75, the Jacobi algorithm's guesses do not converge, as shown in Figure 4. For each of these non-convergent values, the guesses spiral out from the 1-vector towards infinity.

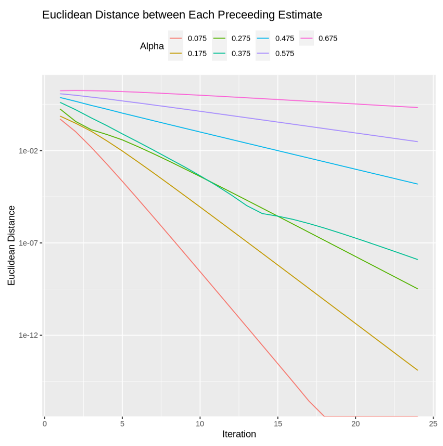


Figure 3: For $\alpha < 0.75$

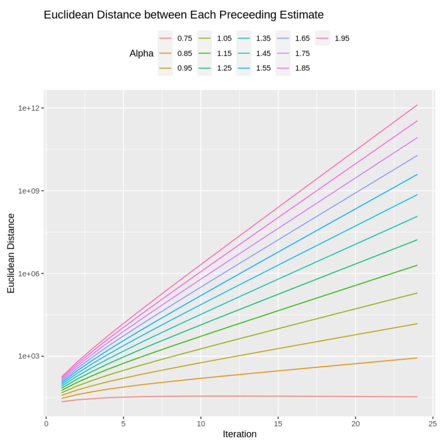


Figure 4: For $\alpha > 0.75$

3 Wikipedia Page Network

3.1 Data

As another implementation of our previous analysis, we use a dataset of the 990 most visited pages on Wikipedia to construct a 990*990 adjacency matrix based on which of these articles link to which other articles. We then proceed to conduct the same Katz index and Jacobi iteration analysis as with the workplace network, in order to find a vector that ranks the articles by number of connections toward them. The code below was used to create the matrix (comments explain what happens at each step).

```r

# HELPER:  The following functions convert between the site labels and the matrix index
SiteToIndex <- function(s) {match(s,sites)}
IndexToSite <- function(s) {sites[s]}

## List: hold all row index (to)
ii <- rep(0,length(to))
## extract the index of site in matrix using helper function
for (i in 1:length(from))
  ii[i] <- SiteToIndex(from[i])

## List: hold all column index (from)
jj <- rep(0,length(from))
## extract the index of site in matrix using helper function
for (i in 1:length(to))
  jj[i] <- SiteToIndex(to[i])

## all-ones vector to mark connection between rows and columns
xx <- rep(1,length(jj))
```

3.2 Jacobi Implementation

Rank	Alpha = 0.0022		Alpha = 0.00055		Alpha = 0.000275	
	Name	Rating	Name	Rating	Name	Rating
1	2007	13.09403	2007	1.1831424	United_States	0.2031472
2	United_States	12.76101	United_States	1.140917	2006	0.1949758
3	2006	12.69015	2006	1.0930386	2008	0.1830918
4	March_4	12.51434	2005	0.9931338	2005	0.1712652
5	January_1	12.50435	2008	0.9825344	United_Kingdom	0.1605785

The table above is the output of the Jacobi algorithm used on the Wikipedia dataset to rank the centrality of the various different webpages. Different alpha values changed the attenuation factor of secondary and tertiary connections, which in turn altered how high up the webpages scored in the Katz index. For example, a low alpha value resulted in United States being the most visited webpage, implying that United States can be considered to be higher when more consideration is given to secondary and tertiary connections, but 2007 scored the highest with a higher alpha value, implying that there are a lot of pages that link directly to the 2007 index, and not necessarily relatively a lot of secondary and tertiary links.

4 Conclusion

The R code used here is one example of a simple way to use linear algebra to compute centrality in a network. The Jacobi method and its application in the R language are a good tool for analysis of directed graphs and adjacency matrices, and we hope this report helps further understanding of graphs and the algorithms that analyze them.

5 Appendix

5.1 Full Matrices of Workplace Influence Network Example

$$B = \begin{pmatrix}
2 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & -1 & 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & -1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{pmatrix}$$

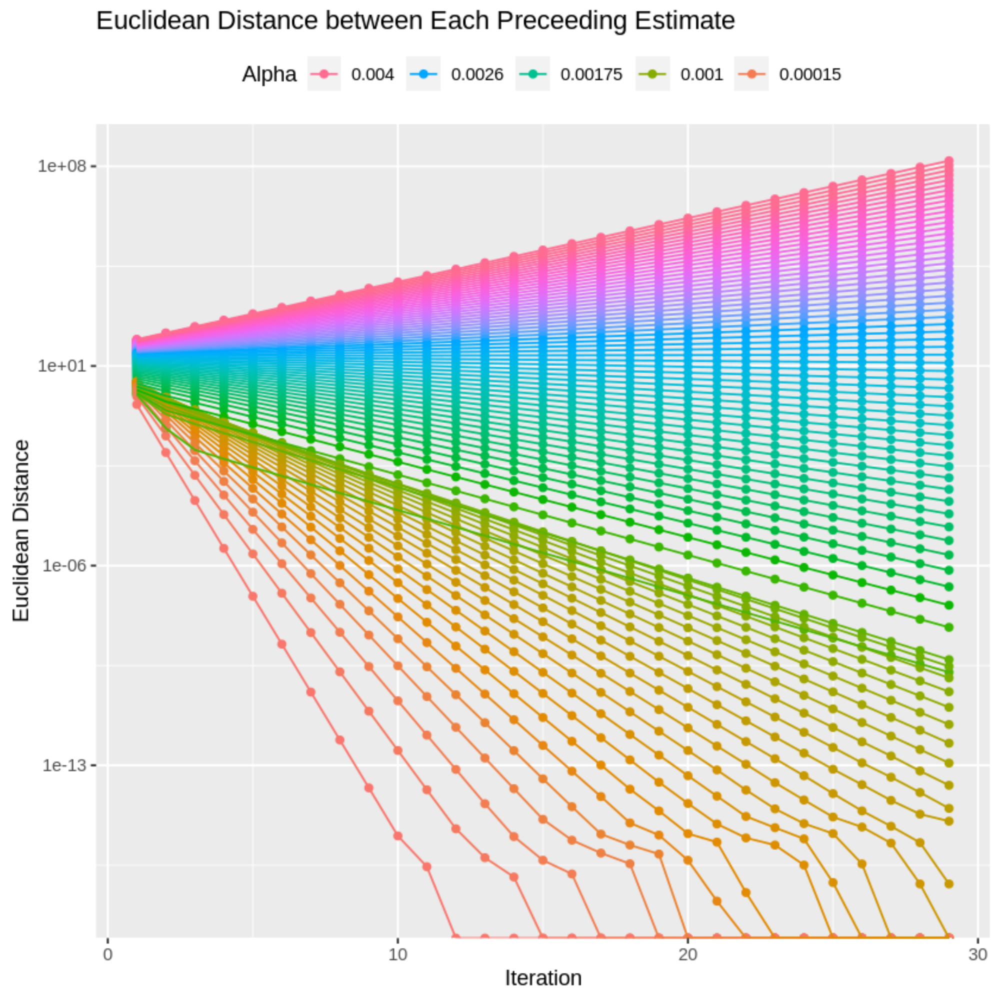
$$1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} d = \begin{pmatrix} 6 \\ 4 \\ 3 \\ 4 \\ 4 \\ 2 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$D = \begin{pmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{pmatrix}$$

$$R = \begin{pmatrix}
0 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1- & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$D^{-1} = \begin{pmatrix} 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 \end{pmatrix}$$

5.2 R Code



```
jacobi_method = function(a, A, I, x, maxIteration = 100){
    # store all estimation in each iteration
    history <- matrix(NA,nrow=length(x),ncol= maxIteration)
    B = (1/a)*I - t(A)                          # find the B matrix
    n = ncol(B)                                 # number of row/col of B
    oneVector = c(rep(1, ncol(A)))              # one-vector in p=K1 equation
    d = t(A)*oneVector                          # d in p=K1 equation
    D = diag(diag(B))                              # diagonal matrix of Jacobi method
    R = B - D                                      # remaining non-diagonal matrix (Jacobi)
    D_1 = solve(D)
    for(j in 1:maxIteration){                          # make appr for iter_max iterations
        x = D_1*(d - (R*x))                        # update new x
        history[,j] = x[,1]                            # store approximation
    }
    return(list(x = x, history = history))         # return last estimation and history
}
```

```
library(tibble)

calculateNorm = function(M, p){
    norm = as_tibble(data.frame(
                  iteration=integer(),
                  norm=double()))
    for(c in 1:ncol(M)){
        norm = norm %>% add_row(iteration = c, norm = vnorm(M[,c],p = 2))
    }
    return(norm)
}

run_test = function(alphaList, A , x0, max_iteration){
    euclidean_distances = as_tibble(data.frame(
                  iteration=integer(),
                  error=double(),
                  alpha=double()))
    n = ncol(A)
    I = spMatrix(n,n,1:n, 1:n, x=rep(1,n))
    for(a in alphaList) {
        ## find answer
        answer = jacobi_method(a, A , I, x0, max_iteration)

        # Calculate error
        error = answer$history[,2:max_iteration] - answer$history[,1:(max_iteration-1)]
        norm = calculateNorm(error, 2)$norm
        euclidean_distances = euclidean_distances %>%
                                    add_row(
                                        iteration = 1:(length(norm)), alpha=rep(a, length(norm))
                                            ,
                                        error = norm
                                    )
    }
    return(euclidean_distances)
}
```

```
library(Matrix)
# The following functions convert between the site labels and the matrix index
SiteToIndex <- function(s) {match(s,sites)}
IndexToSite <- function(s) {sites[s]}
ii <- rep(0,length(to))
jj <- rep(0,length(from))
for (i in 1:length(from))
  ii[i] <- SiteToIndex(from[i])
for (i in 1:length(to))
  jj[i] <- SiteToIndex(to[i])
xx <- rep(1,length(jj))
A <- spMatrix(nrow=n,ncol=n,i=ii,j=jj,x=xx)
```

```
# Read data
W <- read.csv("http://www.macalester.edu/~dshuman1/data/365/links.1000.csv")
# Extract source
from = W$from
# Extract destination
to = W$to
# Extract all unique websites and sort in ascending order
```

```
sites = sort(union(unique(to), unique(from)))
# Count of all sites
n = length(sites)

# Read in Site name file
site_names <- read.csv("http://www.macalester.edu/~dshuman1/data/365/titles-sorted-pruned.
    csv")
site_names <- site_names %>% clean_names()
```