Abstract

WILLS, REBECCA S. When Rank Trumps Precision: Using the Power Method to Compute Google's PageRank. (Under the direction of Ilse C.F. Ipsen.)

The PageRank algorithm, developed by Google founders Larry Page and Sergey Brin, assigns ranking scores to webpages that reflect their relative importance. These scores are based primarily on the link structure of the Web graph and correspond to elements of a dominant left eigenvector, called the PageRank vector, of the stochastic Google matrix. When the starting vector is a probability vector, the iterates of the power method applied to the Google matrix converge to the PageRank vector. Determining when to stop the iterations requires deciding when an iterate vector is good enough. Existing termination criteria rely on various measures of distance between successive iterate vectors. In this dissertation, we investigate how well a power method iterate vector approximates the PageRank vector, we show that the existing termination criteria do not guarantee accurate ranking, and we provide a computationally efficient criterion for determining relative rankings, exact rankings, and ranking intervals of PageRank scores.

WHEN RANK TRUMPS PRECISION: USING THE POWER METHOD TO COMPUTE GOOGLE'S PAGERANK

by

Rebecca S. Wills

a dissertation submitted to the graduate faculty of

north carolina state university

in partial fulfillment of the

requirements for the degree of

doctor of philosophy

mathematics

raleigh, north carolina

2007

approved by:

Biography

Rebecca Smith Wills, daughter of Jane and Danny Smith, was born on April 12, 1977, in Halifax County, North Carolina. She has two older siblings, Danita and Daniel, and one younger sibling, Jonathan. She received a Bachelor of Science degree in mathematics from High Point University on May 8, 1999, and married Jeremiah Wills on May 15, 1999. After working as a Benefits Specialist at Aon Consulting, Inc. for approximately 2.5 years, Rebecca began her graduate work in mathematics at North Carolina State University in the fall of 2002. She received a Master of Science degree in 2004 and completed her doctorate in 2007. While at NC State, she taught several mathematics courses and won the Maltbie Award for excellence in teaching.

Acknowledgements

Anyone who has ever been able to sustain good work has had at least one person – and often many – who have believed in him or her. We just don't get to be competent human beings without a lot of different investments from others.

- Fred Rogers (host of *Mister Rogers' Neighborhood*) [65]

Although this dissertation indicates to others that I have achieved an academic goal, it serves as a reminder to me of those who helped along the way. I am grateful for the financial, academic, and/or personal support I received from many people in my educational pursuits.

Financial Assistance. I thank the NCSU math department for providing support during the past five years through teaching assistantships, summer employment, office space, supplies, and travel funds. In addition, I thank the following organizations for their financial contributions: the NCSU Graduate Student Support Plan, the NCSU Alumni Association (fellowship), the NCSU Preparing the Professoriate program (stipend), the NCSU University Graduate Student Association (travel funds), and the Society for Industrial and Applied Mathematics (SIAM student travel award).

Academic Support. I owe considerable thanks to the members of my advisory committee: Ilse Ipsen, Steve Campbell, Steve Kirkland, Carl Meyer, and Ernie Stitzinger. I often think of how fortunate I have been to get to know and to receive feedback from such a strong group of mathematicians. I also thank David Bird for agreeing to serve as my representative from the graduate school.

I am grateful to my academic advisor and dissertation chair, Ilse Ipsen, for freely investing her time and expertise in my educational development. Through weekly meetings, class discussions, and email correspondence with Ilse Ipsen, I had numerous opportunities to observe her strong work ethic and tireless commitment to excellence in mathematics. Among the many things I learned from her, the two most influential have been the importance of effectively communicating ideas and the benefits of receiving honest critique of those ideas.

The end product of my dissertation research reflects many contributions from all members of my advisory committee. During presentations of my research to the Graduate Student Numerical Analysis Seminar, Steve Campbell always asked me thoughtful questions. In fact, one question ultimately changed the course of my dissertation research. In addition to providing the main idea for the analysis in my dissertation, Steve Kirkland read early drafts of my results and gave helpful feedback. I give credit to Carl Meyer for sparking my interest in the PageRank algorithm and for answering many of my early questions about it. In addition, I consider his books and research papers excellent resources.

I thank Ernie Stitzinger for convincing me to attend NCSU and for always showing that he cares about me as a person. Each conversation I had with him during the past five years brightened my days. I am confident that he will never know the extent of his impact on the lives of so many graduate students.

Many others deserve to be acknowledged for their academic and technical support. I thank the members of the Graduate Student Numerical Analysis Seminar for making me a better speaker. I am grateful to John Griggs for serving as my mentor for the Preparing the Professoriate program and for his interest in my development as an educator. I thank Amy Langville for planning link analysis seminars to discuss the PageRank algorithm, for co-authoring many useful resources on PageRank, and for always pointing me in the right direction when I needed something. I owe thanks to David Gleich for providing data sets for many of my experiments and for promptly answering my questions about the data sets. Without the help of Joel Cason during the month before my defense, I am not sure if I would have been able to complete my dissertation on time. I thank him for quickly fixing numerous computer problems. I appreciate the assistance I received from Miriam Ansley, Seyma Bennett-Shabbir, Di Bucklad, Brenda Currin, Carolyn Gunton, Denise Seabrooks, and Charlene Wallace throughout my tenure at NCSU.

I would be remiss if I failed to mention the tremendous influence of faculty from High Point University. My chemistry professor, Charlie Warde, was the first to tell me that it would be a shame if I did not become "Dr. Smith." I have never forgotten his words nor have I forgotten his delightful personality. For many years, my academic success has been championed by Rob Harger, Manyon Idol, Nelson Page, and Shirley Robertson. Impressed by their commitment to teaching and genuine interest in their students, I continually strive to invest as much in my students as they have invested and continue to invest in me.

Personal Support. I cannot adequately express thanks to my husband, Jay, for his constant support and overwhelming belief in me. Although also working on his Ph.D. in sociology, he always found time to be my counselor, editor, secretary, and biggest fan. From bringing me coffee when I needed it to working long hours with me at NCSU, his actions are those of an unselfish partner. I truly am blessed to share life with him.

Through the years, I have received unwavering support from my parents and my siblings. They always are there to cheer me on. At a very young age, I learned from my mother to be satisfied with nothing less than my best. I respect and admire her strength and character. As a minister, my dad taught me about God's unfailing love and ever present help. Some of my greatest memories from childhood are the debates I had with my parents and siblings. I am so thankful to my parents for encouraging inquisitive children to ask questions and to my siblings, Danita, Daniel, and Jonathan, for enjoying debates as much as I did.

As the years go by, my extended family continues to grow. I thank each member, from nieces and nephews to in-laws, for providing necessary diversions from study during the past five years.

I have enjoyed getting to know and certainly will miss my office-mates and near office-mates: Jordan Bostic, Rebecca Kalhorn, Drew Pasteur, Laurie Zack, April Alston, and Ryan Siskind. Thanks for looking through my presentation slides, helping me figure out Matlab code, and deciding just the right word to include in a sentence. More importantly, thanks for the conversations about nothing in particular. Good luck to each of you in your academic pursuits. Thanks also to Amanda Hall and Rizwana Rehman for your friendship.

Finally, I owe many thanks to my former students. You greatly enhanced my experience at NCSU and served as a daily reminder to me of my primary reason for embarking on this journey.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The most popular modern-day ranking scheme is the PageRank algorithm developed by Google founders Larry Page and Sergey Brin and implemented by Google for many Web search tools. The PageRank algorithm determines the relative importance of webpages by assigning a ranking score to each of more than 25 billion webpages recognized by Google [8]. The PageRank scores are based primarily on the link structure of the Web graph, and they correspond to elements of a dominant left eigenvector, called the PageRank vector, of the stochastic Google matrix.

Google originally computed an approximation to the PageRank vector by means of the power method. When the starting vector is a probability vector, the power method iterate vectors converge to the PageRank vector. Existing termination criteria are based on various measures of distance between successive iterate vectors. In this dissertation, we investigate how well a power method iterate vector approximates the PageRank vector, we show that the existing termination criteria do not guarantee ranking, and we introduce a computationally efficient criterion for determining rankings of PageRank scores.

The dissertation is organized as follows. Chapter 2, Sections 3.1, 3.2, and 4.1 provide background material and literature review. Our contributions begin in Section 3.3 along with summaries of existing results and continue in Sections 4.2 and 4.3.

Chapter 2

Google's PageRank Algorithm

2.1 Background

Google[1] founders Larry Page and Sergey Brin met in 1995 when Page visited the computer science department of Stanford University during a recruitment weekend [2, 12]. Brin, a second-year graduate student at the time, served as a guide for potential recruits, and Page was part of his group. They discussed many topics during their first meeting and disagreed on nearly every issue. Soon after beginning graduate study at Stanford, Page began working on a Web project, initially called BackRub, that exploited the link structure of the Web. Brin found Page's work on BackRub interesting, so the two started working together on a project that would permanently change Web search. Brin and Page realized that they were creating a search engine that adapted to the ever-increasing size of the Web, so they replaced the name BackRub with Google (a common misspelling of *googol*, the number 10^{100}). Unable to convince existing search engine companies to adopt the technology they had developed but certain their technology was superior to any available, Brin and

[1]Most of Chapter 2 and the first two sections of Chapter 3 appeared in [75].

Page decided to start their own company. With the financial assistance of a small group of initial investors, Brin and Page founded the Web search engine company Google, Inc. in September 1998.

Almost immediately, the general public noticed what Brin, Page, and others in the academic Web search community already knew — the Google search engine produced much higher-quality results than those produced by other Web search engines. Other search engines relied entirely on webpage content to determine ranking of results, and Brin and Page realized that webpage developers could easily manipulate the ordering of search results by placing concealed information on webpages.[2] Brin and Page developed a ranking algorithm, named PageRank after Larry Page, that utilizes the link structure of the Web to determine the importance of webpages. During the processing of a query, Google's search algorithm combines precomputed PageRank scores with text-matching scores to obtain an overall ranking score for each webpage.

Although many factors determine Google's overall ranking of search engine results, Google maintains that the heart of its search engine software is PageRank [3]. A few quick searches on the Internet reveal that both the business and academic communities hold PageRank in high regard. The business community is mindful that Google remains the search engine of choice and that PageRank plays a substantial

[2]That is, a developer could add text in the same color as the background of the page, invisible to the user but detected by automated search engines. If the terms of a search query occurred many times in the hidden text, that webpage could appear higher than webpages which were really more informative.

role in the order in which webpages are displayed. Maximizing the PageRank score of a webpage, therefore, has become an important component of company marketing strategies. The academic community recognizes that PageRank has connections to numerous areas of mathematics and computer science such as matrix theory, numerical analysis, information retrieval, and graph theory. As a result, much research continues to be devoted to explaining and improving PageRank.

2.2 Notation

We adopt the following notation throughout the dissertation. All matrices are real $n \times n$ and vectors are real $n \times 1$. The transpose of a vector u is the $1 \times n$ vector u^T. The one-norm of a column vector u with elements u_i is $\|u\|_1 \equiv \sum_i |u_i|$, and the infinity-norm is $\|u\|_\infty \equiv \max_i |u_i|$. The one-norm of a matrix A with elements A_{ij} is the maximal column sum, $\|A\|_1 \equiv \max_j \sum_i |A_{ij}|$, and the infinity-norm is the maximal row sum $\|A\|_\infty \equiv \max_i \sum_j |A_{ij}|$. The vector $\mathbb{1}$ is the column vector of all ones. If u is a vector with all nonnegative elements, we write $u \geq \mathbf{0}$, and $\|u\|_1 = u^T\mathbb{1}$. A vector u with $u \geq \mathbf{0}$ and $u^T\mathbb{1} = 1$ is a *probability vector*. The uniform vector is $\left(\frac{1}{n}\right)\mathbb{1}$. The column vector e_i is the vector with 1 in the ith position and 0 elsewhere. The directed graph associated with matrix A is $\Delta(A)$. If $A_{ij} > 0$, then there is a directed edge from vertex i to vertex j in $\Delta(A)$.

2.3 Mathematics of PageRank

The PageRank algorithm models the behavior of an idealized *random Web surfer* [18, 62]. This Internet user randomly chooses a webpage to view from the listing of available webpages. Then, the surfer randomly selects a link from that webpage to another webpage. The surfer continues the process of selecting links at random from successive webpages until deciding to move to another webpage by some means other than selecting a link. The choice of which webpage to visit next does not depend on the previously visited webpages, and the idealized Web surfer never grows tired of visiting webpages. Thus, the PageRank score of a webpage represents the probability that a random Web surfer chooses to view the webpage.

2.3.1 Directed Web Graph

To model the activity of the random Web surfer, the PageRank algorithm represents the link structure of the Web as a *simple directed graph*[3]. Webpages are vertices (or nodes) of the graph, and links from webpages to other webpages are edges that show direction of movement. Webpages with neither outlinks nor inlinks are isolated vertices, and we assume these vertices have been removed from the directed Web graph. Although the Web graph is very large, the PageRank algorithm can be applied to a simple directed graph of any size. To faciliate our discussion of PageRank, we

[3]A *simple directed graph* is a directed graph with neither multiple edges nor loops.

apply the PageRank algorithm to the simple directed graph with 4 vertices shown in Figure 2.1.

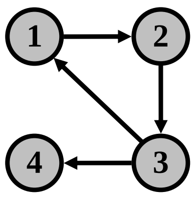


Figure 2.1: Simple directed graph with 4 vertices

2.3.2 Web Hyperlink Matrix

The process for determining PageRank begins by expressing a simple directed graph as the $n \times n$ "hyperlink matrix" H, where n is the number of vertices. If vertex i has *outdegree*[4] $l_i \geq 1$, then $H_{ij} = 1/l_i$ if there is a directed edge from vertex i to vertex j, and $H_{ij} = 0$, otherwise. Thus, H_{ij} represents the likelihood that a random surfer follows a directed edge from vertex i to vertex j. For the directed graph in Figure 2.1,

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

[4]The *outdegree* of vertex i is the number of edges from vertex i to other vertices.

Vertex 4 is called a *dangling node* because the outdegree of vertex 4 is zero (there is no directed edge from vertex 4 to any other vertex). As a result, all entries in row 4 of H are zero. This means the probability is zero that a random surfer moves from vertex 4 to any other vertex. The majority of webpages are dangling nodes in the directed Web graph (e.g., postscript files and image files), so there are many zero rows in in the Web hyperlink matrix [13, Section 1], [25, Section 1]. When a Web surfer lands on dangling node webpages, the surfer can either stop surfing or move to another webpage, perhaps by entering the Uniform Resource Locator (URL) of a different webpage in the address line of a Web browser. Since H does not model the possibility of moving from dangling node webpages to other webpages, the long-term behavior of Web surfers cannot be determined from H alone.

2.3.3 Dangling Node Fix

Several options exist for modeling the behavior of a random Web surfer after landing on a dangling node, and Google does not reveal which option it employs. One option replaces each row of H corresponding to a dangling node by the same probability vector, w^T. The resulting matrix is $S = H + dw^T$, where d is a column vector that identifies dangling nodes, meaning $d_i = 1$ if vertex i is a dangling node and $d_i = 0$, otherwise. The most popular choice in the literature for w is the uniform vector, $w = \left(\frac{1}{n}\right) \mathbb{1}$. This amounts to adding artificial directed edges from dangling nodes to all vertices in the directed graph. With $w = \left(\frac{1}{4}\right) \mathbb{1}$, the directed graph in Figure 2.1 changes (see Figure 2.2).

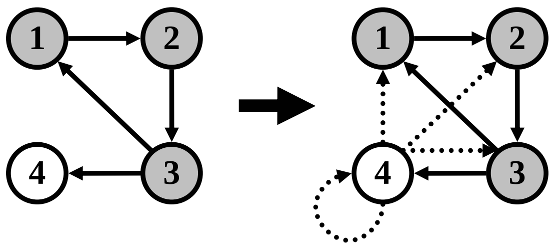


Figure 2.2: Dangling node fix to Figure 1

The matrix $S = H + dw^T$ is,

$$\begin{aligned} S &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}. \end{aligned}$$

Although S depicts the tendency of random surfers to leave dangling nodes, the model is not yet complete. Even when webpages have links to other webpages, a random Web surfer might grow tired of continually selecting links and decide to move to a different webpage some other way. For the graph in Figure 2, there is no directed edge from vertex 2 to vertex 1. On the Web, though, a surfer can move directly from one webpage to another by entering the URL for a webpage in the address line of a Web browser. The matrix S does not account for this possibility.

2.3.4 Google Matrix

The overall behavior of an idealized surfer is modeled by the *Google matrix*

$$G = \alpha S + (1 - \alpha)\mathbb{1}v^T, \tag{2.1}$$

where $0 \leq \alpha < 1$ is the *damping factor* and v is a probability vector called the *personalization vector*. The damping factor, α, allows idealized Web surfers to move to a different webpage with probability $1 - \alpha$ by some means other than selecting a link. Brin and Page performed PageRank experiments with parameter values $\alpha = 0.85$ and $v = \left(\frac{1}{n}\right)\mathbb{1}$ [18, 62, 69]. Choices for α ranging from 0.85 to 0.99 appear in most literature on the PageRank algorithm.

Assigning the uniform vector for v suggests Web surfers randomly jump to new webpages when not selecting links. The uniform vector makes PageRank highly susceptible to *link spamming* [13, Section 4.4], [34, Section 6.4.1]. Link spamming is the practice by some search engine optimization experts of adding more links to their clients' webpages for the sole purpose of increasing the PageRank score of those webpages. This attempt to manipulate PageRank scores is one reason Google does not reveal the current damping factor or personalization vector for the Google matrix. In 2004, however, Gyöngyi, Garcia-Molina, and Pederson developed the TrustRank algorithm to create a personalization vector that decreases the harmful effect of link spamming [34], and Google registered the trademark for TrustRank on March 16, 2005 [6].

2.3.5 PageRank Vector

Because each element G_{ij} of G lies between 0 and 1 ($0 \leq G_{ij} \leq 1$) and the sum of elements in each row of G is 1, the Google matrix is a *stochastic* matrix. It is known that $\lambda = 1$ is not a repeated eigenvalue of G and is greater in magnitude than any other eigenvalue of G [26, 37, 69]. Hence the eigensystem

$$\pi^T G = \pi^T, \qquad \pi \geq \mathbf{0}, \qquad \pi^T \mathbb{1} = 1, \tag{2.2}$$

has the unique solution π, called the PageRank vector[5].

We say that $\lambda = 1$ is the *dominant eigenvalue*[6] of G, and π is a corresponding *dominant left eigenvector*[7] of G. (Since G represents a Markov chain, π is also called the *stationary distribution vector* of G. See [60, § 8.4] for more information on Markov

[5]Though not required, the restriction is often made that the personalization vector v and the dangling node vector w are probability vectors with all positive entries instead of all nonnegative entries. Under this restriction, the PageRank vector also is a probability vector with all positive entries.

[6]If the eigenvalues λ_i of A satisfy $|\lambda_1| > |\lambda_j|$ for all $j \neq 1$, then we say that λ_1 is the *dominant eigenvalue* of A.

[7]If A has dominant eigenvalue λ_1, then an eigenvector corresponding to λ_1 is a *dominant eigenvector* of A.

chains.) The ith element of π is the PageRank score for vertex i. If $\pi_i > \pi_j$ for some $i \neq j$, then the PageRank of vertex i is higher than that of vertex j.

2.3.6 Linear System Formulation of PageRank

Based on the definition of the Google matrix (2.1),

$$\begin{aligned}\pi^T &= \pi^T G \\ &= \pi^T \left(\alpha S + (1-\alpha)\mathbb{1}v^T\right) \\ &= \alpha\pi^T S + (1-\alpha)\left(\pi^T\mathbb{1}\right)v^T \\ &= \alpha\pi^T S + (1-\alpha)v^T, \text{ since } \pi^T\mathbb{1} = 1.\end{aligned}$$

Thus, $\pi^T - \alpha\pi^T S = (1-\alpha)v^T$ implies that $\pi^T(I - \alpha S) = (1-\alpha)v^T$. Therefore, we can equivalently define PageRank as the solution to the linear system,

$$\pi^T(I - \alpha S) = (1-\alpha)v^T. \tag{2.3}$$

Since $\|\alpha S\|_\infty = \alpha < 1$, the matrix $I - \alpha S$ is nonsingular (see [31, Lemma 2.3.3], [54, §7.1], and [60, Example 7.10.7]). Thus,

$$\pi^T = (1-\alpha)\, v^T (I - \alpha S)^{-1}. \tag{2.4}$$

We will see in later chapters that $(I - \alpha S)^{-1}$ plays a major role in analyzing the accuracy of power method iterates as approximations to π. We state several important facts about $I - \alpha S$ and $(I - \alpha S)^{-1}$ in the following property:

Property 2.1. Properties of $I - \alpha S$ and $(I - \alpha S)^{-1}$

1. $I - \alpha S$ is a nonsingular M-matrix.

2. $(I - \alpha S)^{-1} \geq \mathbf{0}$.

3. The row sums of $I - \alpha S$ are $1 - \alpha$.

4. $\|I - \alpha S\|_\infty = 1 + \alpha$.

5. The row sums of $(I - \alpha S)^{-1}$ are $\frac{1}{1-\alpha}$.

6. $\left\|(I - \alpha S)^{-1}\right\|_\infty = \frac{1}{1-\alpha}$.

7. $(I - \alpha S)^{-1}$ is strictly diagonally dominant of its column entries,

   meaning $(I - \alpha S)^{-1}_{ii} > (I - \alpha S)^{-1}_{li}$, for all $1 \leq i \leq n$, $1 \leq l \leq n$, $l \neq i$.

8. $(I - \alpha S)^{-1} = \sum_{m=0}^{\infty} \alpha^m S^m$.

9. $1 \leq (I - \alpha S)^{-1}_{ii} \leq \frac{1}{1-\alpha}$ for all $1 \leq i \leq n$.

Most of these properties have appeared before, and their proofs are straightforward (see [53, § 5.2] and [54, § 7.1]). Property 2.1.7 follows from [59, Theorem 3.2, Remark 3.3], and Property 2.1.8 follows from [31, Lemma 2.3.3].

2.3.7 Examples

Table 2.1 shows four different Google matrices and their corresponding PageRank vectors (approximated to two decimal places) for the directed graph in Figure 2.2.

The table illustrates the influence of the personalization vector on PageRank scores. For instance, when $\alpha = 0.85$, as is the case for the first and second models, the PageRank scores and the ordering of the scores differ significantly. The first model assigns the uniform vector to v, and vertices 1 and 4 receive the lowest PageRank score. When $v = e_1$, vertex 1 has the highest PageRank score. This personalization vector models idealized surfers moving to vertex 1 once they grow tired of following the edge structure of the graph. For the third and fourth models, $\alpha = 0.95$. The difference in PageRank scores and ordering of scores for these models is less pronounced. Even though $v = e_1$ in the fourth model, the higher damping factor decreases the influence of v.

Table 2.1: Modeling surfer behavior for the directed graph in Figure 2.2

	Damping Factor (α)	Personalization Vector (v^T)	Google Matrix (G)	PageRank Vector ($\approx \pi^T$)
Model 1	0.85	$\begin{pmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}\frac{3}{80} & \frac{71}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80} \\ \frac{37}{80} & \frac{3}{80} & \frac{3}{80} & \frac{37}{80} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}0.21 & 0.26 & 0.31 & 0.21\end{pmatrix}$
Model 2	0.85	$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}\frac{3}{20} & \frac{17}{20} & 0 & 0 \\ \frac{3}{20} & 0 & \frac{17}{20} & 0 \\ \frac{23}{40} & 0 & 0 & \frac{17}{40} \\ \frac{29}{80} & \frac{17}{80} & \frac{17}{80} & \frac{17}{80}\end{pmatrix}$	$\begin{pmatrix}0.30 & 0.28 & 0.27 & 0.15\end{pmatrix}$
Model 3	0.95	$\begin{pmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}\frac{1}{80} & \frac{77}{80} & \frac{1}{80} & \frac{1}{80} \\ \frac{1}{80} & \frac{1}{80} & \frac{77}{80} & \frac{1}{80} \\ \frac{39}{80} & \frac{1}{80} & \frac{1}{80} & \frac{39}{80} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}0.21 & 0.26 & 0.31 & 0.21\end{pmatrix}$
Model 4	0.95	$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}\frac{1}{20} & \frac{19}{20} & 0 & 0 \\ \frac{1}{20} & 0 & \frac{19}{20} & 0 \\ \frac{21}{40} & 0 & 0 & \frac{19}{40} \\ \frac{23}{80} & \frac{19}{80} & \frac{19}{80} & \frac{19}{80}\end{pmatrix}$	$\begin{pmatrix}0.24 & 0.27 & 0.30 & 0.19\end{pmatrix}$

Chapter 3

The Power Method for Computing PageRank

3.1 Properties

For small Google matrices like the ones in Table 2.1, we can easily find exact solutions to the eigensystem, $\pi^T G = \pi^T$. The Google matrix for the entire Web has more than 25 billion rows and columns [8], so computing the exact solution requires extensive time and computing resources. The power method was the first iterative algorithm used to approximate the PageRank vector [62]. Since then, many have suggested other possible algorithms [13, 53, 54]. These include classical iterative methods [9, 28, 14, 55], Krylov subspace methods [28, 29, 30], extrapolation methods [17, 16, 35, 46], aggregation/disaggregation [19, 41, 45, 56], and methods that adapt to the Web graph [42, 44, 57].

Although each suggested algorithm has merit, the power method remains a good option for computing PageRank [28],[54, §4.6]. It is the oldest and easiest technique for approximating a dominant eigenvector of a matrix [71, §3.1]. In addition to being simple to implement and requiring little storage, the power method is robust because its convergence rate depends only on α. For matrices with a dominant eigenvalue, the power method converges to a dominant eigenvector for most starting vectors [22, §9.4]. Recall from Section 2.3.5, $\lambda = 1$ is the dominant eigenvalue of G and π is a dominant left eigenvector. The power method applied to G converges to the PageRank vector when the starting vector is a probability vector, see Section 3.3.

Given a starting vector $x^{(0)}$, with $x^{(0)} \geq \mathbf{0}$ and $x^T \mathbb{1} = 1$, the power method calculates successive iterates

$$\left[x^{(k)}\right]^T = [x^{(k-1)}]^T G, \text{ where } k = 1, 2, ... \tag{3.1}$$

until some convergence criterion is satisfied. Notice that $\left[x^{(k)}\right]^T = [x^{(k-1)}]^T G$ also can be stated $\left[x^{(k)}\right]^T = \left[x^{(0)}\right]^T G^k$. As the number of nonzero elements of the personalization vector increases, the number of nonzero elements of G can increase. Thus, the multiplication of $[x^{(k-1)}]^T$ with G is expensive; however, since $S = H + dw^T$ (for the dangling node fix mentioned in Section 2.3.3) and $G = \alpha S + (1-\alpha)\mathbb{1}v^T$, we can express the multiplication as follows:

$$\begin{aligned} \left[x^{(k)}\right]^T &= [x^{(k-1)}]^T \left(\alpha \left(H + dw^T\right) + (1-\alpha)\mathbb{1}v^T\right) \\ &= \alpha[x^{(k-1)}]^T H + \alpha \left([x^{(k-1)}]^T d\right) w^T + (1-\alpha) \left([x^{(k-1)}]^T \mathbb{1}\right) v^T \\ &= \alpha[x^{(k-1)}]^T H + \alpha \left([x^{(k-1)}]^T d\right) w^T + (1-\alpha) v^T, \end{aligned}$$

because $[x^{(k-1)}]^T \mathbb{1} = 1$ since $[x^{(k-1)}]^T$ is a probability vector. This is a sum of three vectors: a multiple of $[x^{(k-1)}]^T H$, a multiple of w^T, and a multiple of v^T. (Notice that $[x^{(k-1)}]^T d$ is a scalar.) The only matrix-vector multiplication required is with the hyperlink matrix H. A 2004 investigation of Web documents estimates that the average number of outlinks for a webpage is 52 [61]. This means that for a typical row of the hyperlink matrix only 52 of the 25 billion elements are nonzero, so the great majority of elements in H are 0 (H is very sparse). Since all computations involve the sparse matrix H and vectors w^T and v^T, an iteration of the power method is cheap (the operation count is proportional to the matrix dimension n).

3.2 Termination Criteria

The most popular criterion for terminating iterations of the power method is the residual norm. For the PageRank algorithm, the residual is just the difference in successive iterates,

$$\begin{aligned} r_k^T &\equiv \left[x^{(k)}\right]^T (I - G) \\ &= \left[x^{(k)}\right]^T - \left[x^{(k)}\right]^T G \\ &= \left[x^{(k)}\right]^T - \left[x^{(k+1)}\right]^T. \end{aligned} \tag{3.2}$$

The power method stops when $\|r_k\|$ is less than some tolerance (ranging most often from 10^{-8} to 10^{-2}). Although the one norm appears most often in the literature, the infinity norm and the two norm receive some treatment.

The ratio of the two eigenvalues largest in magnitude determines the asymptotic convergence rate of the power method [31]. Haveliwala and Kamvar were the first to prove that the second-largest eigenvalue in magnitude of G is less than or equal to the damping factor α [37]. (See also [26].) This means that the asymptotic convergence rate is at most α for the Google matrix. The non-asymptotic bounds in Section 3.3 are even stronger. They show that the forward error and the residual decrease by a factor of α in each iteration.

3.3 Residual and Forward Error

In this section, we analyze how well a given power method iterate vector approximates the PageRank vector by considering for $k \geq 0$ the residual, $r_k \equiv x^{(k)} - x^{(k+1)}$, and the forward error, $\epsilon_k \equiv x^{(k)} - \pi$. In addition to stating results based on any starting vector $x^{(0)} \geq \mathbf{0}$ with $\left[x^{(0)}\right]^T \mathbb{1} = 1$, we provide results for the particular starting vector $x^{(0)} = v$, the personalization vector. To derive many of the bounds in this section, we apply the following Lemma from [38], also stated in [51, Lemma PS], [60, Exercise 5.1.11] and [21, Lemma 3.1] (and in slightly different form in [68, Theorem 2.10]).

Lemma 3.1. [38, Corollary 2.4(a)] For any vectors c and d such that $c^T \mathbb{1} = 0$,

$$\left|c^T d\right| \leq \tfrac{d_{\max} - d_{\min}}{2} \left\|c\right\|_1 ,$$

where $d_{\max}$ and $d_{\min}$ are the maximum and minimum elements of d.

Proof. For any scalar γ, $c^T(\gamma \mathbb{1}) = \mathbf{0}$. Thus,

$\left|c^T d\right| = \left|c^T(d - \gamma\mathbb{1})\right| \leq \|c\|_1 \|d - \gamma\mathbb{1}\|_\infty$, by the Hölder inequality [31, § 2.2.2].

Let $\|d - \gamma^* \mathbb{1}\|_\infty = \min\limits_{\gamma} \|d - \gamma\mathbb{1}\|_\infty$.

Since $\frac{d_{\max}+d_{\min}}{2}$ is the midpoint of the elements of $d, \gamma^* = \frac{d_{\max}+d_{\min}}{2}$.

Thus, $\min\limits_{\gamma} \|d - \gamma\mathbb{1}\|_\infty = \|d - \left(\frac{d_{\max}+d_{\min}}{2}\right)\mathbb{1}\|_\infty = \frac{d_{\max}-d_{\min}}{2}$.

□

Note that Lemma 3.1 is tighter than the Hölder inequality for $\left|c^T d\right|$ since

$\|d\|_\infty = \max_i |d_i| \geq \frac{d_{\max}-d_{\min}}{2}$.

3.3.1 Residual

In the following theorem, we provide an expression for the residual in terms of the initial residual. The proof of [50, Corollary 3.11] includes the expression, and the derivation for the starting vector $x^{(0)} = v$ appears in the proof of [15, Theorem 5] and in [16, Property 7].

Lemma 3.2 (Power Method Iterates)**.** Let G be the Google Matrix and π its PageRank vector. The power method iterates $\left[x^{(k)}\right]^T \equiv [x^{(k-1)}]^T G$ with $x^{(0)} \geq \mathbf{0}$ and $\|x^{(0)}\|_1 = 1$ can be stated as

$$\left[x^{(k)}\right]^T = \alpha^k \left[x^{(0)}\right]^T S^k + (1-\alpha) v^T \sum_{m=0}^{k-1} \alpha^m S^m, \quad k \geq 1.$$

Further, if $x^{(0)} = v$, then

$$\left[x^{(k)}\right]^T = v^T - v^T (I - S) \sum_{m=0}^{k-1} \alpha^{m+1} S^m.$$

Proof. (by induction on k)

Since $\left[x^{(1)}\right]^T = \left[x^{(0)}\right]^T G = \left[x^{(0)}\right]^T \left(\alpha S + (1-\alpha)\mathbb{1} v^T\right)$

$$= \alpha \left[x^{(0)}\right]^T S + (1-\alpha)\left(\left[x^{(0)}\right]^T \mathbb{1}\right) v^T = \alpha \left[x^{(0)}\right]^T S + (1-\alpha) v^T,$$

the statement is true for $k = 1$. Assume true for k.

That is, assume $\left[x^{(k)}\right]^T = \alpha^k \left[x^{(0)}\right]^T S^k + (1-\alpha) \sum_{m=0}^{k-1} \alpha^m v^T S^m$.

Then, $\left[x^{(k+1)}\right]^T = \left[x^{(k)}\right]^T G = \alpha \left[x^{(k)}\right]^T S + (1-\alpha) v^T$

$$= \alpha \left(\alpha^k \left[x^{(0)}\right]^T S^k + (1-\alpha) \sum_{m=0}^{k-1} \alpha^m v^T S^m \right) S + (1-\alpha) v^T$$

$$= \alpha^{k+1} \left[x^{(0)}\right]^T S^{k+1} + (1-\alpha) \sum_{m=0}^{k-1} \alpha^{m+1} v^T S^{m+1} + (1-\alpha) v^T$$

$$= \alpha^{k+1} \left[x^{(0)}\right]^T S^{k+1} + (1-\alpha) \sum_{m=1}^{k} \alpha^m v^T S^m + (1-\alpha) v^T$$

$$= \alpha^{k+1} \left[x^{(0)}\right]^T S^{k+1} + (1-\alpha) \sum_{m=0}^{k} \alpha^m v^T S^m.$$

Thus, the statement is true for $k+1$.

Hence, by induction, $\left[x^{(k)}\right]^T = \alpha^k\left[x^{(0)}\right]^T S^k + (1-\alpha)\sum_{m=0}^{k-1}\alpha^m v^T S^m, \quad k \geq 1.$

If $x^{(0)} = v$, then

$$\begin{aligned}
\left[x^{(k)}\right]^T &= \alpha^k v^T S^k + (1-\alpha)\sum_{m=0}^{k-1}\alpha^m v^T S^m. \\
&= \sum_{m=0}^{k}\alpha^m v^T S^m - \sum_{m=0}^{k-1}\alpha^{m+1} v^T S^m \\
&= v^T + \sum_{m=1}^{k}\alpha^m v^T S^m - \sum_{m=0}^{k-1}\alpha^{m+1} v^T S^m \\
&= v^T + \sum_{m=0}^{k-1}\alpha^{m+1} v^T \left(S^{m+1} - S^m\right) \\
&= v^T - v^T\left(I - S\right)\sum_{m=0}^{k-1}\alpha^{m+1} S^m.
\end{aligned}$$

□

Theorem 3.3 (Residual)**.** Let G be the Google Matrix and π its PageRank vector. The power method iterates $\left[x^{(k)}\right]^T \equiv [x^{(k-1)}]^T G$ with $x^{(0)} \geq \mathbf{0}$ and $\|x^{(0)}\|_1 = 1$ satisfy

$$r_k^T = \alpha^k\left(\left[x^{(0)}\right]^T - \left[x^{(1)}\right]^T\right)S^k = \alpha^k r_0^T S^k, \quad k \geq 1.$$

Further, if $x^{(0)} = v$, then

$$r_k^T = \alpha^{k+1} v^T \left(I - S\right) S^k.$$

Proof. By Lemma 3.2,

$$\begin{aligned}
r_k^T &= \left[x^{(k)}\right]^T - \left[x^{(k+1)}\right]^T \\
&= \alpha^k \left[x^{(0)}\right]^T S^k + (1-\alpha)\sum_{m=0}^{k-1} \alpha^m v^T S^m - \alpha^{k+1}\left[x^{(0)}\right]^T S^{k+1} - (1-\alpha)\sum_{m=0}^{k} \alpha^m v^T S^m \\
&= \alpha^k \left[x^{(0)}\right]^T S^k - \alpha^{k+1}\left[x^{(0)}\right]^T S^{k+1} - (1-\alpha)\alpha^k v^T S^k \\
&= \alpha^k \left(\left[x^{(0)}\right]^T - \alpha\left[x^{(0)}\right]^T S - (1-\alpha) v^T \right) S^k \\
&= \alpha^k \left(\left[x^{(0)}\right]^T - \left[x^{(1)}\right]^T \right) S^k \\
&= \alpha^k r_0^T S^k.
\end{aligned}$$

If $x^{(0)} = v$, then $r_0^T = v^T - v^T G = v^T - \alpha v^T S - (1-\alpha)v^T = \alpha v^T (I - S)$.

Thus, $r_k^T = \alpha^{k+1} v^T (I-S) S^k$. □

With the help of Theorem 3.3, we derive upper bounds for the one norm and the infinity norm of the residual.

Corollary 3.4 (Residual Bounds)**.** Under the assumptions of Theorem 3.3,

$$\|r_k\|_1 \le \alpha^k \|r_0\|_1 \le 2\alpha^k \quad \text{and} \quad \|r_k\|_\infty \le \tfrac{\alpha^k}{2}\|r_0\|_1 \le \alpha^k.$$

Further, if $x^{(0)} = v$, then

$$\|r_k\|_1 \le 2\alpha^{k+1} \quad \text{and} \quad \|r_k\|_\infty \le \alpha^{k+1}.$$

Proof.

For the one norm,

$\|r_k\|_1 = \|r_k^T\|_\infty = \|\alpha^k r_0^T S^k\|_\infty \le \alpha^k \|r_0^T\|_\infty \|S\|_\infty^k = \alpha^k \|r_0\|_1 \le 2\alpha^k.$

If $x^{(0)} = v$, then $\|r_k\|_1 = \alpha^{k+1}\|v^T(I-S)S^k\|_\infty \leq \alpha^{k+1}\|v^T(I-S)\|_\infty$

$$\leq \alpha^{k+1}\left(\|v\|_1 + \|v\|_1\|S\|_\infty\right) = 2\alpha^{k+1}.$$

For the infinity norm,

$\|r_k\|_\infty = \|r_k^T\|_1 = \|\alpha^k r_0^T S^k\|_1 = \alpha^k\|r_0^T S^k\|_1.$

By Lemma 3.1, since $r_0^T \mathbb{1} = \mathbf{0}$ and S^k is stochastic,

$\|r_k\|_\infty = \alpha^k \max\left|r_0^T S^k e_i\right| \leq \frac{\alpha^k}{2}\|r_0\|_1 \leq \alpha^k.$

If $x^{(0)} = v$, then $\|r_k\|_\infty = \alpha^{k+1}\|v^T(I-S)S^k\|_1 = \alpha^{k+1}\max\left|v^T(I-S)S^k e_i\right|.$

Since $v^T(I-S)\mathbb{1} = \mathbf{0}$,

$\|r_k\|_\infty \leq \frac{\alpha^{k+1}}{2}\|v^T(I-S)\|_\infty \leq \alpha^{k+1}.$

$\square$

3.3.2 Forward Error

In this section, we provide expressions for the forward error $\epsilon_k = x^{(k)} - \pi$ in terms of the initial forward error $\epsilon_0 = x^{(0)} - \pi$, the group generalized inverse of $I - G$, the matrix $(I - \alpha S)^{-1}$, and the difference $x^{(0)} - x^{(k)}$. In addition, we include simple proofs of previously known bounds on the forward error and derive new bounds.

In the following theorem, we give an expression for the forward error in terms of the initial forward error $\epsilon_0 = x^{(0)} - \pi$. Based on the expression for the forward error, we obtain an upper bound for the component-wise error $\left|x_i^{(k)} - \pi_i\right|$.

Theorem 3.5 (Forward Error)**.** Let G be the Google Matrix and π its PageRank vector. The power method iterates $\left[x^{(k)}\right]^T \equiv [x^{(k-1)}]^T G$ with $x^{(0)} \geq \mathbf{0}$ and $\|x^{(0)}\|_1 = 1$ satisfy

$$\epsilon_k^T = \alpha^k \left(\left[x^{(0)}\right]^T - \pi^T \right) S^k = \alpha^k \epsilon_0^T S^k, \quad k \geq 1.$$

Further, if $x^{(0)} = v$, then

$$\epsilon_k^T = \alpha^{k+1} \left(v^T - \pi^T S \right) S^k.$$

Proof. (by induction on k)

Since $\epsilon_1^T = \left[x^{(1)}\right]^T - \pi^T = \left[x^{(0)}\right]^T G - \pi^T G = \left(\left[x^{(0)}\right]^T - \pi^T \right) G$

$$= \left(\left[x^{(0)}\right]^T - \pi^T \right) \left(\alpha S + (1-\alpha) \mathbb{1} v^T \right)$$

$$= \alpha \left(\left[x^{(0)}\right]^T - \pi^T \right) S + (1-\alpha) \left(\left[x^{(0)}\right]^T - \pi^T \right) \mathbb{1} v^T$$

$$= \alpha \left(\left[x^{(0)}\right]^T - \pi^T \right) S = \alpha \epsilon_0^T S,$$

the statement is true for $k = 1$. Assume true for k. That is, assume $\epsilon_k^T = \alpha^k \epsilon_0^T S^k$.

Then, $\epsilon_{k+1}^T = \left[x^{(k+1)}\right]^T - \pi^T = \left(\left[x^{(k)}\right]^T - \pi^T \right) G = \epsilon_k^T G$

$$= \alpha^k \epsilon_0^T S^k \left(\alpha S + (1-\alpha) \mathbb{1} v^T \right)$$

$$= \alpha^{k+1} \epsilon_0^T S^{k+1} + \alpha^k (1-\alpha) \epsilon_0^T S^k \mathbb{1} v^T = \alpha^{k+1} \epsilon_0^T S^{k+1}.$$

Thus, the statement is true for $k+1$. Hence, by induction, $\epsilon_k^T = \alpha^k \epsilon_0^T S^k$.

If $x^{(0)} = v$, then $\epsilon_k^T = \alpha^k \left(v^T - \pi^T \right) S^k = \alpha^k \left(v^T - \pi^T G \right) S^k$

$$= \alpha^k \left(v^T - \alpha \pi^T S - (1-\alpha) v^T \right) S^k = \alpha^{k+1} \left(v^T - \pi^T S \right) S^k.$$

$\square$

Theorem 3.6 (Component-wise Error Bound)**.** Under the assumptions of Theorem 3.5,

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^k}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \|\epsilon_0\|_1.$$

Further, if $x^{(0)} = v$, then

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^{k+1}}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \|v^T - \pi^T S\|_\infty.$$

Proof.

By Lemma 3.1 and Theorem 3.5, $\left|x_i^{(k)} - \pi_i\right| = \left|\left(\left[x^{(0)}\right]^T - \pi^T\right)\left(\alpha^k S^k e_i\right)\right|$.

Since $\left(\left[x^{(0)}\right]^T - \pi^T\right) \mathbb{1} = 0$,

$$\begin{aligned}\left|x_i^{(k)} - \pi_i\right| &\leq \tfrac{\alpha^k}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \|x^{(0)} - \pi\|_1 \\ &= \tfrac{\alpha^k}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \|\epsilon_0\|_1.\end{aligned}$$

If $x^{(0)} = v$, then $\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^{k+1}}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \|v^T - \pi^T S\|_\infty$,

since $\left(v^T - \pi^T S\right) \mathbb{1} = 0$.

$\square$

Since S^k is stochastic, $0 \leq \max_{h,l}\left(S_{hi}^k - S_{li}^k\right) \leq 1$. Also, S_{hi}^k represents the probability of moving within the directed graph from any vertex h to vertex i in exactly k steps. If $\max_{h,l}\left(S_{hi}^k - S_{li}^k\right) = 1$, then there is a walk of length k from vertex h to vertex i only, and some other vertex cannot reach vertex i in k steps. Further, $\max_{h,l}\left(S_{hi}^k - S_{li}^k\right) = 0$ means the probability of moving to vertex i in exactly k steps is the same for every vertex. In this case, the PageRank for vertex i has converged.

We formalize this statement in the theorem below. We also note that columns of S corresponding to unreferenced vertices (vertices with zero *indegree*[1]) have all zero entries (depending on the dangling node fix). Thus, the PageRank scores for these vertices depend only on the personalization vector v ([24, Theorem 3.1] provides a similar statement for unreferenced vertices when the dangling node fix vector and the personalization vector are $w = v = \left(\frac{1}{n}\right)\mathbb{1}$).

Theorem 3.7. Under the assumptions of Theorem 3.5, if $S^k e_i = \beta\mathbb{1}$ for some $\beta \geq 0$ and $k \geq 1$, then $x_i^{(k)} = \pi_i$. Further, if $Se_i = \mathbf{0}$, then $x_i^{(1)} = \pi_i = (1-\alpha)v_i$.

Proof.

If $\left|x_i^{(k)} - \pi_i\right| \leq \frac{\alpha^k}{2} \max\limits_{h,l}\left(S_{hi}^k - S_{li}^k\right)\|\epsilon_0\|_1$ and $\max\limits_{h,l}\left(S_{hi}^k - S_{li}^k\right) = 0$, then $x_i^{(k)} = \pi_i$.

But, $\max\limits_{h,l}\left(S_{hi}^k - S_{li}^k\right) = 0 \Leftrightarrow \max\limits_{h}\left(S_{hi}^k\right) = \min\limits_{h}\left(S_{hi}^k\right) \Leftrightarrow S^k e_i = \beta\mathbb{1}$.

If $Se_i = \mathbf{0}$, then $\left[x^{(0)}\right]^T G e_i = \alpha\left(\left[x^{(0)}\right]^T S e_i\right) + (1-\alpha)\left(\left[x^{(0)}\right]^T \mathbb{1}\right) v^T e_i = (1-\alpha)v_i$.

□

In the following corollary, we provide upper bounds for the one norm and the infinity norm of the forward error. The bound for the one norm also appears without proof in [43, §4] and with a different proof in [14, Theorem 6.1].

[1]The *indegree* of vertex i is the number of edges from other vertices to vertex i.

Corollary 3.8 (Norm-wise Error Bounds)**.** Under the assumptions of Theorem 3.5,

$$\|\epsilon_k\|_1 \leq \alpha^k \|\epsilon_0\|_1 \leq 2\alpha^k \quad \text{and} \quad \|\epsilon_k\|_\infty \leq \tfrac{\alpha^k}{2}\|\epsilon_0\|_1 \leq \alpha^k.$$

Further, if $x^{(0)} = v$, then

$$\|\epsilon_k\|_1 \leq 2\alpha^{k+1} \quad \text{and} \quad \|\epsilon_k\|_\infty \leq \alpha^{k+1}.$$

Proof.

For the one norm,

$\|\epsilon_k\|_1 = \|\epsilon_k^T\|_\infty = \|\alpha^k \epsilon_0^T S^k\|_\infty \leq \alpha^k \|\epsilon_0^T\|_\infty \|S\|_\infty^k = \alpha^k \|\epsilon_0\|_1 \leq 2\alpha^k.$

If $x^{(0)} = v$, then $\|\epsilon_k\|_1 = \alpha^{k+1}\| \left(v^T - \pi^T S\right) S^k\|_\infty \leq \alpha^{k+1}\|v^T - \pi^T S\|_\infty$

$$\leq \alpha^{k+1}\left(\|v\|_1 + \|\pi^T S\|_\infty\right).$$

Since $\pi^T S \geq \mathbf{0}$ and $\left(\pi^T S\right)\mathbb{1} = 1$, $\|\epsilon_k\|_1 \leq 2\alpha^{k+1}$.

The bounds for the infinity norm follow from Theorem 3.6 and the fact that

for every vertex i, $0 \leq \max_{h,l}\left(S_{hi}^k - S_{li}^k\right) \leq 1$.

The upper bounds follow from $\|\epsilon_0\|_1 \leq 2$ and $\|v^T - \pi^T S\|_\infty \leq 2$. $\square$

Group Generalized Inverse of $\mathbf{I} - \mathbf{G}$. In [50, § 2], Kirkland determines how well a power method iterate $x^{(k)}$ approximates π for a general stochastic matrix with 1 as an algebraically simple eigenvalue. Tayloring his general analysis specifically to iterates of the power method applied to the Google matrix, Kirkland recognizes that the group generalized inverse of the matrix $I-G$ is a "key component" in analyzing the the forward error, $\epsilon_k = x^{(k)} - \pi$. Since G is a stochastic matrix, the group generalized inverse $(I-G)^{\#}$ exists [20, Theorem 8.2.1]. It is the unique matrix satisfying the three equations [20, Definition 7.2.4]:

1. $(I-G)(I-G)^{\#}(I-G) = (I-G)$,
2. $(I-G)^{\#}(I-G)(I-G)^{\#} = (I-G)^{\#}$, and
3. $(I-G)(I-G)^{\#} = (I-G)^{\#}(I-G)$.

Further, $(I-G)(I-G)^{\#} = I - \mathbb{1}\pi^T$ [20, Theorem 8.2.3]. Thus,

$$\begin{aligned} r_k^T (I-G)^{\#} &= \left(\left[x^{(k)}\right]^T - \left[x^{(k+1)}\right]^T\right)(I-G)^{\#} \\ &= \left[x^{(k)}\right]^T (I-G)(I-G)^{\#} \\ &= \left[x^{(k)}\right]^T \left(I - \mathbb{1}\pi^T\right) \\ &= \left[x^{(k)}\right]^T - \pi^T. \end{aligned} \tag{3.3}$$

So, $\epsilon_k^T = \left[x^{(k)}\right]^T - \pi^T = r_k^T (I-G)^{\#}$. Therefore, Kirkland concludes that $(I-G)^{\#}$ plays a major role in any discussion of the accuracy of $x^{(k)}$ as an approximation to π.

Kirkland's analysis is based on *directed cycle*[2] lengths in $\Delta(S)$, the directed graph associated with S. Note that $\Delta(H)$, the directed graph associated with the hyperlink matrix, differs from $\Delta(S)$ when there are dangling nodes. Based on his analysis of $(I-G)^{\#}$ and the cycle structure of $\Delta(S)$, Kirkland provides the following forward error bounds for a vertex i.

[2]A *directed cycle* is a directed walk with distinct edges and distinct vertices (except that it begins and ends with the same vertex).

Theorem 3.9. [50, Corollary 3.11] Under the assumptions of Theorem 3.5,

1. If vertex i is on no cycle of length at least 2 in $\Delta(S)$,

   then $\left|x_i^{(k)} - \pi_i\right| \leq \frac{\alpha^k}{2}\left(\frac{1}{1-\alpha S_{ii}}\right)\|r_0\|_1$.

2. If vertex i is on a cycle of length at least 2 and g is the length of the shortest such cycle, then $\left|x_i^{(k)} - \pi_i\right| \leq \frac{\alpha^k}{2}\left(\frac{1}{1-\alpha^g-\alpha S_{ii}(1-\alpha^{g-1})}\right)\|r_0\|_1$.

For the second bound, note that $\frac{1}{1-\alpha^g-\alpha S_{ii}(1-\alpha^{g-1})} \geq 1$ since $0 \leq \alpha < 1$. The bound is tighter for webpages that lie only on long cycles. For both bounds, note that if $S_{ii} \neq 0$, then vertex i is a dangling node and $w_i > 0$. Otherwise, $S_{ii} = 0$ since simple directed graphs do not have loops.

Remark 3.10. If $x^{(0)} = v$, then the bounds in Theorem 3.9 are:

1. If vertex i is on no cycle of length at least 2 in $\Delta(S)$,

   then $\left|x_i^{(k)} - \pi_i\right| \leq \frac{\alpha^{k+1}}{2}\left(\frac{1}{1-\alpha S_{ii}}\right)\left\|v^T(I-S)\right\|_\infty$.

2. If vertex i is on a cycle of length at least 2 and g is the length of the shortest such cycle, then $\left|x_i^{(k)} - \pi_i\right| \leq \frac{\alpha^{k+1}}{2}\left(\frac{1}{1-\alpha^g-\alpha S_{ii}(1-\alpha^{g-1})}\right)\left\|v^T(I-S)\right\|_\infty$.

Linear System Matrix. Recall from Section 2.3.6, that PageRank is the solution to the linear system,

$$\pi^T(I-\alpha S) = (1-\alpha)v^T,$$

where $(I - \alpha S)$ is a nonsingular M-matrix. Using the expanded form of the Google matrix, we obtain

$$\begin{aligned}(I - G)\,(I - \alpha S)^{-1} &= \left((I - \alpha S) - (1 - \alpha)\mathbb{1}v^T\right)(I - \alpha S)^{-1} \\ &= I - (1 - \alpha)\mathbb{1}v^T(I - \alpha S)^{-1} \\ &= I - \mathbb{1}\pi^T.\end{aligned}$$

Thus,

$$\begin{aligned}\epsilon_k^T &= \left[x^{(k)}\right]^T - \pi^T \\ &= \left[x^{(k)}\right]^T (I - G)\,(I - \alpha S)^{-1} \\ &= r_k^T(I - \alpha S)^{-1}, \qquad (3.4)\end{aligned}$$

which appears in [16, Property 11]. Since $I - G$ is a singular matrix, $(I - \alpha S)^{-1}$ cannot be the group generalized inverse of $I - G$; however, $(I - \alpha S)^{-1}$ plays a similar role. In the following theorem, we state an upper bound for the component-wise error $\left|x_i^{(k)} - \pi_i\right|$ based on $\epsilon_k^T = r_k^T(I - \alpha S)^{-1}$.

Theorem 3.11 (Component-wise Error Bound)**.** Under the assumptions of Theorem 3.5,

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{1}{2}\left((I - \alpha S)_{ii}^{-1} - \min_h\,(I - \alpha S)_{hi}^{-1}\right)\|r_k\|_1, \quad k \geq 0.$$

Proof.

$$\left|x_i^{(k)} - \pi_i\right| = \left|r_k^T(I-\alpha S)^{-1}e_i\right|$$

$$\leq \tfrac{1}{2}\|r_k\|_1 \max_{h,l}\left((I-\alpha S)^{-1}_{hi} - (I-\alpha S)^{-1}_{li}\right), \text{ since } r_k^T\mathbb{1} = 0.$$

By Property 2.1.7, $\left|x_i^{(k)} - \pi_i\right| \leq \frac{1}{2}\left((I-\alpha S)^{-1}_{ii} - \min_h (I-\alpha S)^{-1}_{hi}\right)\|r_k\|_1$.

□

Remark 3.12. Applying Corollary 3.4 to Theorem 3.11,

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^k}{2}\left((I-\alpha S)^{-1}_{ii} - \min_h (I-\alpha S)^{-1}_{hi}\right)\|r_0\|_1, \quad k \geq 0.$$

Further, if $x^{(0)} = v$, then

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^{k+1}}{2}\left((I-\alpha S)^{-1}_{ii} - \min_h (I-\alpha S)^{-1}_{hi}\right)\|v^T(I-S)\|_\infty, \quad k \geq 0.$$

Since r_k is easily computable and $\|r_k\|_1 \leq \alpha^k\|r_0\|_1$, these bounds are not better than the bounds in Theorem 3.11.

We know that $1 \leq (I-\alpha S)^{-1}_{ii} \leq \frac{1}{1-\alpha}$. Also, if vertex i is on no cycle of length 1 or greater, then $(I-\alpha S)^{-1}_{ii} = 1$ and $\min_h (I-\alpha S)^{-1}_{hi} = 0$. In this case, the bound in Remark 3.12 is the same as the bound in Theorem 3.9.

Based on $\epsilon_k^T = r_k^T(I-\alpha S)^{-1}$, we have the following upper bounds for the one norm and the infinity norm of the forward error. The bound for the one norm also appears in [16, Property 12] and in [50, Equation 3.1]. The bound for the infinity norm appears in [50, Remark 3.1]. We do not provide a separate statement for $x^{(0)} = v$.

Theorem 3.13 (Norm-wise Error Bounds)**.** Under the assumptions of Theorem 3.5,

$$\|\epsilon_k\|_1 \leq \tfrac{1}{1-\alpha}\|r_k\|_1 \quad \text{and} \quad \|\epsilon_k\|_\infty \leq \tfrac{1}{2(1-\alpha)}\|r_k\|_1.$$

Proof.

For the one norm,

$\|\epsilon_k\|_1 = \|\epsilon_k^T\|_\infty = \|r_k^T(I-\alpha S)^{-1}\|_\infty \leq \|r_k\|_1\|(I-\alpha S)^{-1}\|_\infty = \tfrac{1}{1-\alpha}\|r_k\|_1.$

The result for the infinity norm follows from Theorem 3.11 and the fact that

for any vertex i, $(I-\alpha S)^{-1}_{ii} - \min\limits_h (I-\alpha S)^{-1}_{hi} \leq \tfrac{1}{1-\alpha}$.

□

Initial and Current Iterates. We determine an expression for the PageRank vector π and computable bounds for the forward error in terms of the initial iterate and current iterate. We derive a forward error bound for individual PageRank scores. In addition, we provide upper bounds for the one norm and the infinity norm of the forward error. Since we do not obtain more useful information when $x^{(0)} = v$, we do not include additional statements for $x^{(0)} = v$ in this section.

Theorem 3.14 (PageRank Vector)**.** Under the assumptions of Theorem 3.5,

$$\pi^T = \left(\left[x^{(k)}\right]^T - \alpha^k\left[x^{(0)}\right]^T S^k\right)\left(I - \alpha^k S^k\right)^{-1}, \quad k \geq 1.$$

Proof. By Theorem 3.5,

$$\left[x^{(k)}\right]^T - \pi^T = \alpha^k \left(\left[x^{(0)}\right]^T - \pi^T\right) S^k = \alpha^k \left[x^{(0)}\right]^T S^k - \alpha^k \pi^T S^k.$$

So, $\left[x^{(k)}\right]^T - \alpha^k \left[x^{(0)}\right]^T S^k = \pi^T - \alpha^k \pi^T S^k = \pi^T \left(I - \alpha^k S^k\right).$

Since $\left(I - \alpha^k S^k\right)$ is nonsingular (by [31, Lemma 2.3.3]),

$$\pi^T = \left(\left[x^{(k)}\right]^T - \alpha^k \left[x^{(0)}\right]^T S^k\right) \left(I - \alpha^k S^k\right)^{-1}.$$

□

Theorem 3.15 (Forward Error)**.** Under the assumptions of Theorem 3.5,

$$\epsilon_k^T = \alpha^k \left(\left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k, \quad k \geq 1.$$

Proof. By Theorems 3.5 and 3.14,

$$\begin{aligned}
\epsilon_k^T &= \left[x^{(k)}\right]^T - \pi^T = \alpha^k \left(\left[x^{(0)}\right]^T - \pi^T\right) S^k \\
&= \alpha^k \left(\left[x^{(0)}\right]^T - \left(\left[x^{(k)}\right]^T - \alpha^k \left[x^{(0)}\right]^T S^k\right) \left(I - \alpha^k S^k\right)^{-1}\right) S^k \\
&= \alpha^k \left(\left[x^{(0)}\right]^T \left(I - \alpha^k S^k\right) - \left[x^{(k)}\right]^T + \alpha^k \left[x^{(0)}\right]^T S^k\right) \left(I - \alpha^k S^k\right)^{-1} S^k \\
&= \alpha^k \left(\left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k.
\end{aligned}$$

□

Using the expression for the forward error in Theorem 3.15, we find an upper bound for the component-wise error $\left|x_i^{(k)} - \pi_i\right|$.

Corollary 3.16 (Component-wise Error Bound)**.** Under the assumptions of Theorem 3.5,

$$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^k}{2} \max_{h,l} \left(\left(e_h^T - e_l^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k e_i \right) \|x^{(0)} - x^{(k)}\|_1, \quad k \geq 1.$$

Proof. By Theorem 3.15,
$\left[x^{(k)}\right]^T - \pi^T = \alpha^k \left(\left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T \right) \left(I - \alpha^k S^k\right)^{-1} S^k.$
Since $\left(\left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T \right) \mathbb{1} = \mathbf{0}$, Lemma 3.1 implies
$\left|x_i^{(k)} - \pi_i\right| \leq \tfrac{\alpha^k}{2} \max_{h,l} \left(\left(e_h^T - e_l^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k e_i \right) \|x^{(0)} - x^{(k)}\|_1.$

□

If $S^k e_i = \beta \mathbb{1}$, the bound in Corollary 3.16 is 0. Thus, the bounds in Corollary 3.16 and in Theorem 3.6 equally identify when PageRank scores converge for certain vertices. Based on Corollary 3.16 and the expression for the forward error in Theorem 3.15, we derive upper bounds for the one norm and the infinity norm of the forward error.

Corollary 3.17 (Norm-wise Error Bounds)**.** Under the assumptions of Theorem 3.5,

$$\|\epsilon_k\|_1 \leq \tfrac{\alpha^k}{1-\alpha^k} \|x^{(0)} - x^{(k)}\|_1 \quad \text{and} \quad \|\epsilon_k\|_\infty \leq \tfrac{\alpha^k}{2\left(1-\alpha^k\right)} \|x^{(0)} - x^{(k)}\|_1, \quad k \geq 1.$$

Proof.

For the one norm,

$$\begin{aligned}\|\epsilon_k\|_1 &= \alpha^k \| \left(\left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T \right) \left(I - \alpha^k S^k\right)^{-1} S^k \|_\infty \\ &\le \alpha^k \| \left[x^{(0)}\right]^T - \left[x^{(k)}\right]^T \|_\infty \| \left(I - \alpha^k S^k\right)^{-1} \|_\infty \|S^k\|_\infty \\ &\le \tfrac{\alpha^k}{1-\alpha^k} \|x^{(0)} - x^{(k)}\|_1, \text{ by [31, Lemma 2.3.3].}\end{aligned}$$

The result for the infinity norm follows from Corollary 3.16 and the fact that for any vertex i, $\max\limits_{h,l} \left(\left(e_h^T - e_l^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k e_i \right) \le \frac{1}{1-\alpha^k}$.

□

The bounds in Corollary 3.17 are tighter than the upper bounds in Corollary 3.8 if $\frac{1}{1-\alpha^k}\|x^{(0)} - x^{(k)}\|_1 < 2$ and $\frac{1}{2\left(1-\alpha^k\right)}\|x^{(0)} - x^{(k)}\|_1 < 1$, respectively. They are tighter than the bounds in Theorem 3.13 when $\frac{\alpha^k}{1-\alpha^k}\|x^{(0)} - x^{(k)}\|_1 < \frac{1}{1-\alpha}\|r_k\|_1$. Note that as k increases, $\frac{1}{1-\alpha^k}$ approaches 1, and $x^{(k)}$ approaches π. Also, if $x^{(0)} = v$, then storage of an additional vector is not required to compute the bounds in Corollary 3.17.

3.3.3 Summary of Bounds

Several bounds appear in Sections 3.3.1 and 3.3.2, so we provide a summary in the following table:

Table 3.1: Summary of Residual and Forward Error Bounds

Type of Bound	**Location**	**Bound**
Residual	Cor. 3.4	$\\|r_k\\|_1 \leq \alpha^k \\|r_0\\|_1 \leq 2\alpha^k$ and $\\|r_k\\|_\infty \leq \frac{\alpha^k}{2}\\|r_0\\|_1 \leq \alpha^k$
Forward Error	Thm. 3.6	$\left\lvert x_i^{(k)} - \pi_i \right\rvert \leq \frac{\alpha^k}{2} \max_{h,l} \left(S_{hi}^k - S_{li}^k\right) \\|\epsilon_0\\|_1$
	Cor. 3.8	$\\|\epsilon_k\\|_1 \leq \alpha^k \\|\epsilon_0\\|_1 \leq 2\alpha^k$ and $\\|\epsilon_k\\|_\infty \leq \frac{\alpha^k}{2}\\|\epsilon_0\\|_1 \leq \alpha^k$
	Thm. 3.9	$\left\lvert x_i^{(k)} - \pi_i \right\rvert \leq \frac{\alpha^k}{2} \left(\frac{1}{1-\alpha S_{ii}}\right) \\|r_0\\|_1$ (when vertex i has longest cycle length less than 2) $\left\lvert x_i^{(k)} - \pi_i \right\rvert \leq \frac{\alpha^k}{2} \left(\frac{1}{1-\alpha^g - \alpha S_{ii}(1-\alpha^{g-1})}\right) \\|r_0\\|_1$ (when the shortest cycle length greater than 1 for vertex i is g)
	Thm. 3.11	$\left\lvert x_i^{(k)} - \pi_i \right\rvert \leq \frac{1}{2} \left((I-\alpha S)_{ii}^{-1} - \min_h (I-\alpha S)_{hi}^{-1}\right) \\|r_k\\|_1$
	Thm. 3.13	$\\|\epsilon_k\\|_1 \leq \frac{1}{1-\alpha}\\|r_k\\|_1$ and $\\|\epsilon_k\\|_\infty \leq \frac{1}{2(1-\alpha)}\\|r_k\\|_1$
	Cor. 3.16	$\left\lvert x_i^{(k)} - \pi_i \right\rvert \leq \frac{\alpha^k}{2} \max_{h,l} \left(\left(e_h^T - e_l^T\right) \left(I - \alpha^k S^k\right)^{-1} S^k e_i\right) \\|x^{(0)} - x^{(k)}\\|_1$
	Cor. 3.17	$\\|\epsilon_k\\|_1 \leq \frac{\alpha^k}{1-\alpha^k}\\|x^{(0)} - x^{(k)}\\|_1$ and $\\|\epsilon_k\\|_\infty \leq \frac{\alpha^k}{2\left(1-\alpha^k\right)}\\|x^{(0)} - x^{(k)}\\|_1$

Since the residual is computable and the forward error is not, we devoted much more time to deriving bounds for the forward error. Also, the residual, $x^{(k)} - x^{(k+1)}$, indicates the closeness of successive power method iterate vectors while the forward error, $x^{(k)} - \pi$, indicates how close a power method iterate vector is to the PageRank vector, which is what we really want to know. Thus, finding tight, computable upper bounds for the forward error is important.

Although each forward error bound we presented contributes to the analysis of forward error, some of the bounds are not computable unless π is known yielding them useless in practice. The component-wise error bounds in Theorem 3.9, Theorem 3.11, and Corollary 3.16 are computable without access to π; however, the bounds are less practical for large matrices since the bounds require knowledge of cycle lengths, access to elements of $(I - \alpha S)^{-1}$, or access to elements of $\left(I - \alpha^k S^k\right)^{-1} S^k$. The upper bounds in Corollary 3.8 and the bounds in Theorem 3.13 and Corollary 3.17 are computable. Further, the bounds in Theorem 3.13 perform best for large values of k in our experiments (see empirical results in Section 3.3.4).

3.3.4 Experiments

All experiments were performed in Matlab without consideration for finite precision errors.

Small Directed Graph

For our first set of experiments, we return to the directed graph with four nodes from Chapter 2.

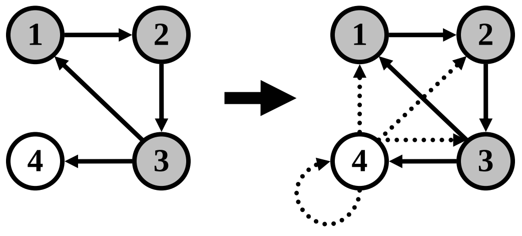


Figure 3.1: Chapter 2 directed graph with dangling node fix $w = \left(\frac{1}{n}\right) \mathbb{1}$

For this graph, we presented four different models of surfer behavior. The Google matrices and PageRank vectors for the models appear in Table 2.1. We report experimental results for the first two models, so we list them again in Table 3.2.

Since we found the PageRank vectors for these models, we are able to compare the computable norm-wise forward error bounds to $\|\epsilon_k\|$. The starting vector influences each bound, so we check the bounds for two different starting vectors. (One reason

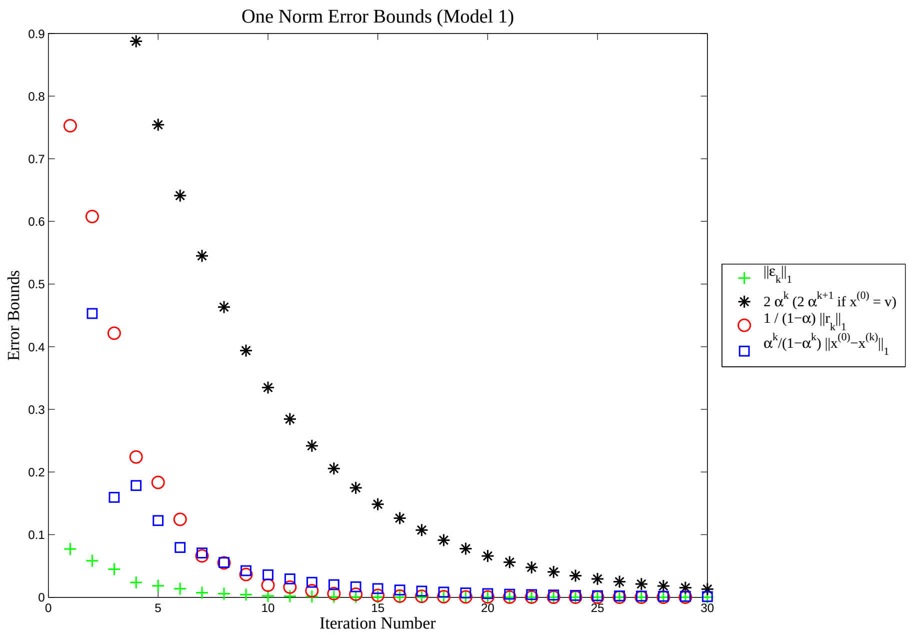
Table 3.2: Chapter 2 Example

	Damping Factor (α)	Personalization Vector (v^T)	Google Matrix (G)	PageRank Vector ($\approx \pi^T$)
Model 1	0.85	$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{80} & \frac{71}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80} \\ \frac{37}{80} & \frac{3}{80} & \frac{3}{80} & \frac{37}{80} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$	$\begin{pmatrix} 0.21 & 0.26 & 0.31 & 0.21 \end{pmatrix}$
Model 2	0.85	$\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \frac{3}{20} & \frac{17}{20} & 0 & 0 \\ \frac{3}{20} & 0 & \frac{17}{20} & 0 \\ \frac{23}{40} & 0 & 0 & \frac{17}{40} \\ \frac{29}{80} & \frac{17}{80} & \frac{17}{80} & \frac{17}{80} \end{pmatrix}$	$\begin{pmatrix} 0.30 & 0.28 & 0.27 & 0.15 \end{pmatrix}$

we include this example is that the reader can easily verify the results for the small directed graph.)

The plots in Figure 3.2 and Figure 3.3 show that ϵ_k is much smaller and the bounds are significantly tighter for Model 1 when $x^{(0)} = v = \left(\frac{1}{4}\right) \mathbb{1}$ than when $x^{(0)} = e_1$. The plots in Figure 3.4 and Figure 3.5 for Model 2 reveal that $x^{(0)} = \left(\frac{1}{4}\right) \mathbb{1}$ is again a better choice for the starting vector even though $v = e_1$.

Although $\alpha^k \|\epsilon_0\|_{1,\infty}$ are not computable for large matrices, all plots indicate that the computable bounds from Corollary 3.17 perform better than the upper bounds in Corollary 3.8 after a few iterations, and the bounds in Theorem 3.13 eventually outperform the bounds from Corollary 3.17.



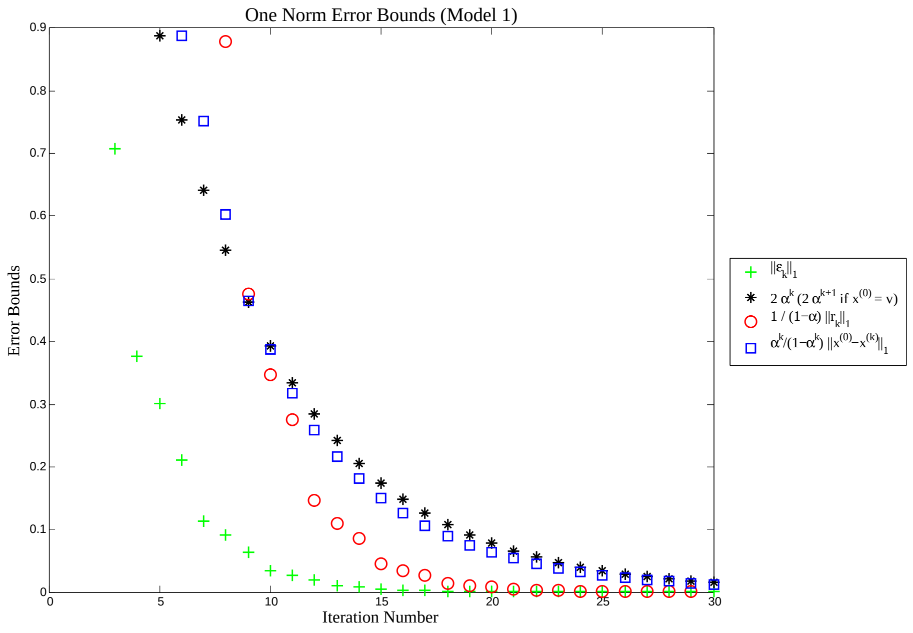
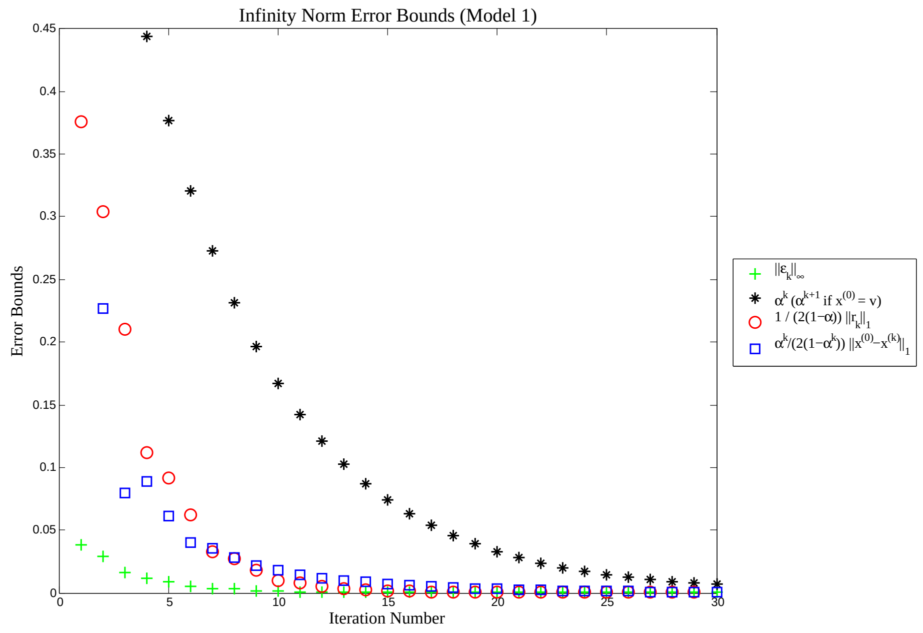


Figure 3.2: One norm – Model 1 (Row 1: $x^{(0)} = v = \left(\frac{1}{4}\right)\mathbb{1}$; Row 2: $x^{(0)} = e_1$)



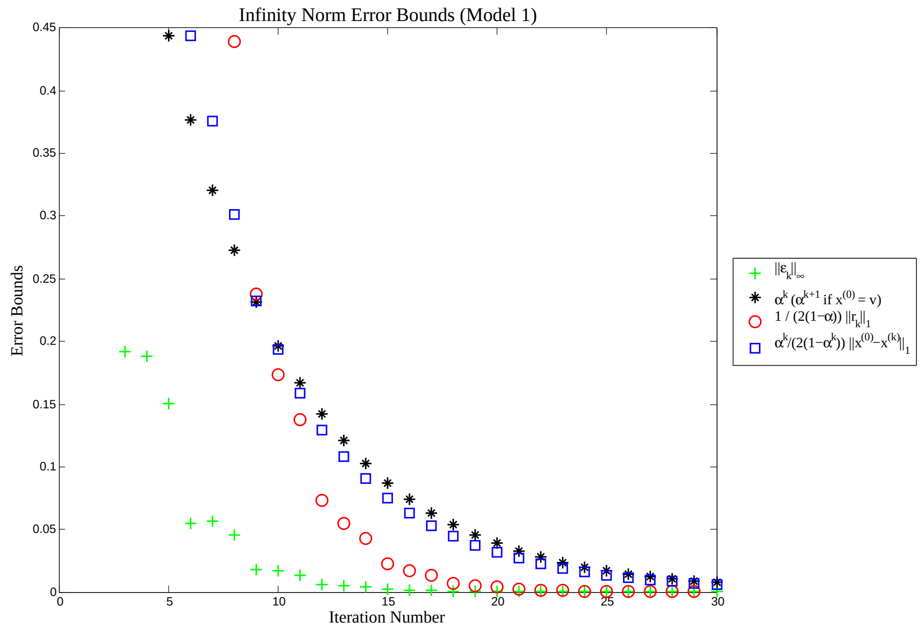
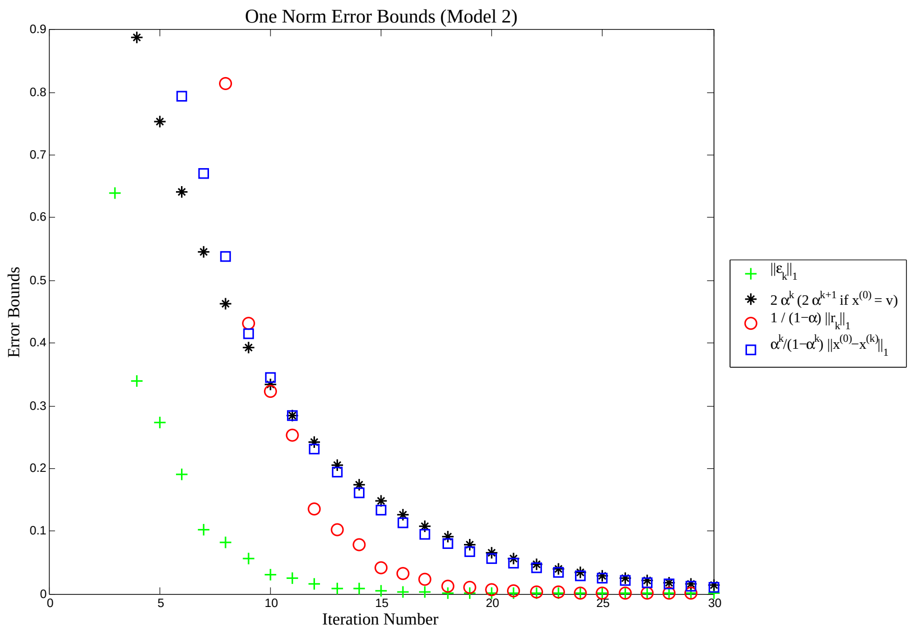


Figure 3.3: Infinity norm – Model 1 (Row 1: $x^{(0)} = v = \left(\frac{1}{4}\right) \mathbb{1}$; Row 2: $x^{(0)} = e_1$)



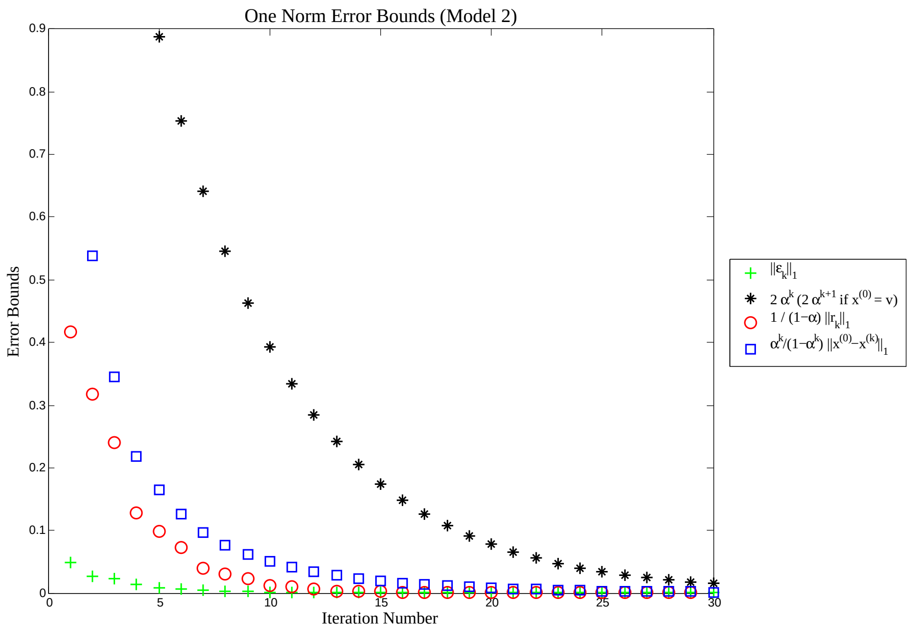
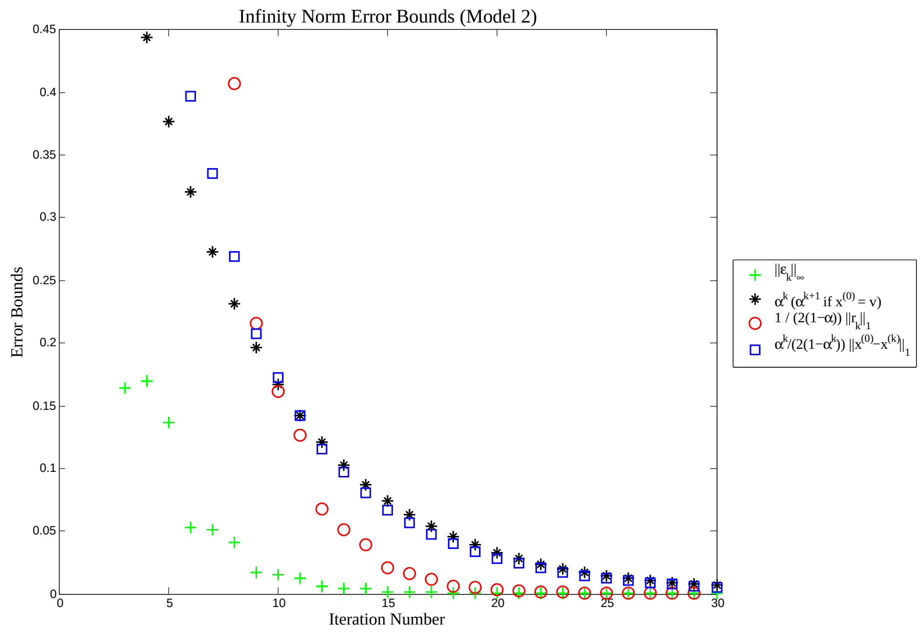


Figure 3.4: One norm – Model 2 (Row 1: $x^{(0)} = v = e_1$; Row 2: $x^{(0)} = \left(\frac{1}{4}\right) \mathbb{1}$)



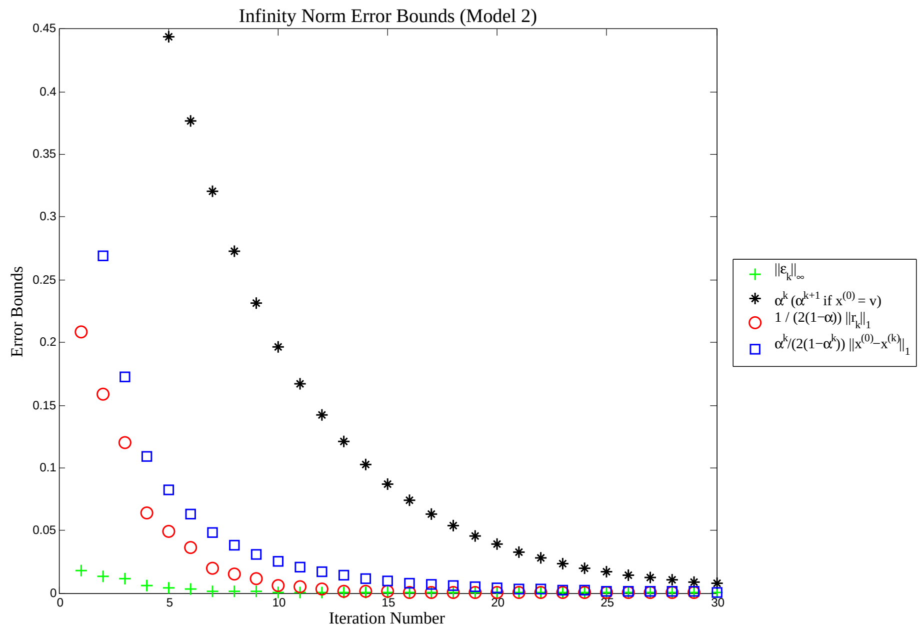


Figure 3.5: Infinity norm – Model 2 (Row 1: $x^{(0)} = v = e_1$; Row 2: $x^{(0)} = \left(\frac{1}{4}\right) \mathbb{1}$)

Larger Directed Graphs

We include empirical results for the six data sets[3] summarized in Table 3.3 and Table 3.4. The Stanford and Stanford-Berkeley data sets are available for download from Sep Kamvar's Stanford homepage [7]. These data matrices appear often in the literature[4] (see [45, § 2] for a description of the data sets).

The *wb-cs.stanford* and Wikipedia data sets are available for download from David Gleich's "datasets graphs and more" website [1]. The *wb-cs.stanford* data set is based on a subgraph of the *webbase.graph* file formed from a 2001 data crawl. The Wikipedia data sets capture the internal link structure between articles in the english Wikipedia site. Gleich formed the Wikipedia data sets by using the XML file from Wikipedia data dumps, and he removed non-article pages.

[3]The percentage of dangling nodes in these data sets is much smaller than the percentage of dangling nodes in the Web graph, see [13, Section 1], [25, Section 1].

[4]For information on the possible drawbacks of using the Stanford matrices for testing purposes, see Sebastiona Vigna's note [72].

Table 3.3: Characteristics of data sets used in experiments

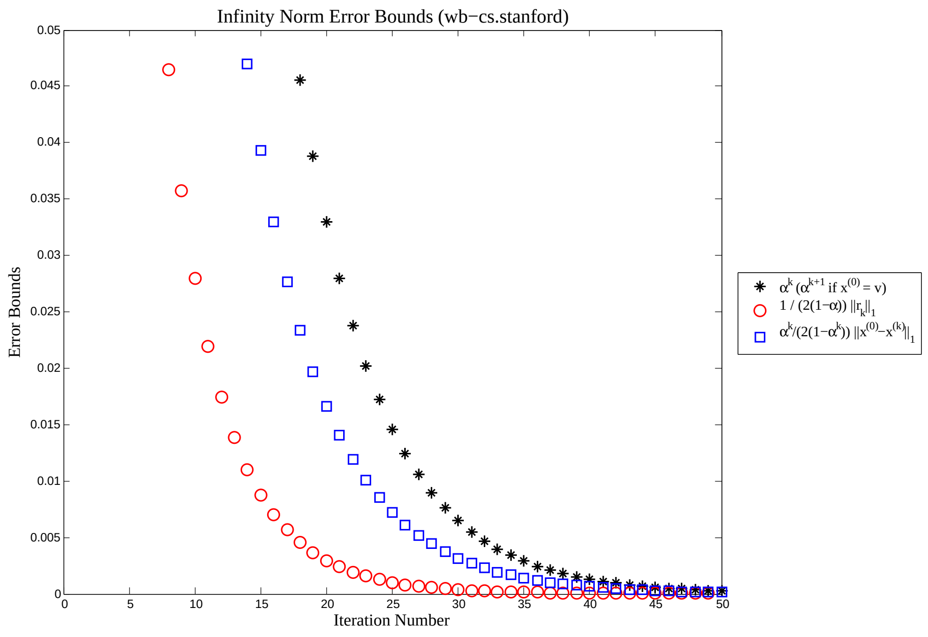
Date	**Name**	**Source**	**Vertices**	**Directed Edges**	**Unreferenced Vertices**	**Dangling Nodes**
2001	wb-cs.stanford	Stanford WebBase Project	9,914	36,854	699 (7.05%)	2,861 (28.86%)
09/02	Stanford	Stanford WebBase Project	281,903	2,312,497	20,315 (7.21%)	172 (0.06%)
12/02	Stanford-Berkeley	Stanford WebBase Project	683,446	7,583,376	68,062 (9.96%)	4,735 (0.69%)
11/05/05	wikipedia-20051105	Wikipedia	1,634,989	19,753,078	464,135 (28.38%)	72,556 (4.44%)
09/25/06	enwiki-20060925 -pages-articles	Wikipedia	2,983,494	37,269,096	873,634 (29.28%)	88,970 (2.98%)
11/04/06	enwiki-20061104 -pages-articles	Wikipedia	3,148,440	39,383,235	932,906 (29.63%)	91,462 (2.91%)

Table 3.4: Degree information of data sets in Table 3.3

Name	**Vertices**	**Largest Indgree**	**Largest Outdegree**	**Indegree < 10**	**Outdegree < 10**
wb-cs.stanford	9,914	340	277	9,488 (95.70%)	9,381 (94.62%)
Stanford	281,903	38,606	255	252,527 (89.58%)	213,135 (75.61%)
Stanford-Berkeley	683,446	83,448	249	595,324 (87.11%)	456,817 (66.84%)
wikipedia-20051105	1,634,989	75,547	4,970	1,353,923 (82.81%)	1,103,768 (67.51%)
enwiki-20060925-pages-articles	2,983,494	159,378	5,852	2,449,537 (82.10%)	2,013,096 (67.50%)
enwiki-20061104-pages-articles	3,148,440	168,685	6,576	2,583,669 (82.06%)	2,124,759 (67.52%)

The Matlab code for our experiments is available in the appendix. For all experiments, we ran 50 iterations of the power method and compared the infinity norm error bounds in Theorem 3.11 $\left(\frac{1}{2(1-\alpha)}\|r_k\|_1\right)$ and Corollary 3.17 $\left(\frac{\alpha^k}{2(1-\alpha^k)}\|x^{(0)} - x^{(k)}\|_1\right)$ to the upper bound in Corollary 3.8 (α^k if $x^{(0)} \neq v$ and α^{k+1} if $x^{(0)} = v$), and we plotted the results. Since the one norm forward error bounds are two times the infinity norm error bounds, we chose not to include plots showing a comparison of the one norm bounds.

For our first set of experiments, we assigned the uniform vector $\left(\frac{1}{n}\right)\mathbb{1}$ to the personalization vector v, the dangling node fix vector w, and the starting power method iterate vector $x^{(0)}$. We chose the damping factor $\alpha = 0.85$. See Figures 3.6 - 3.8 for a plot of each comparison. The figures show that for all six data sets, the infinity norm forward error bound in Corollary 3.17 is slightly better than α^{k+1}, and the bound in Theorem 3.13 is substantially better.



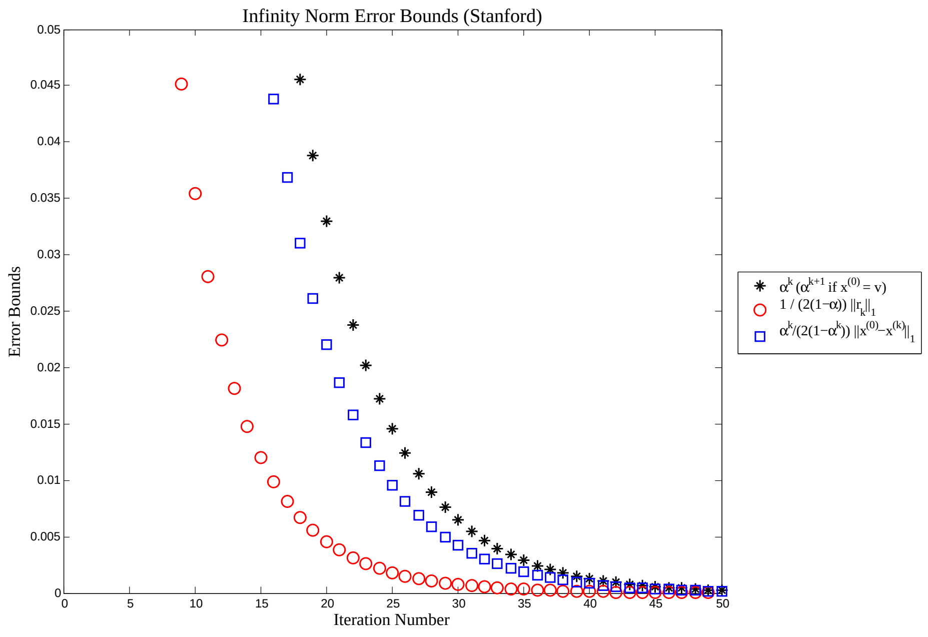
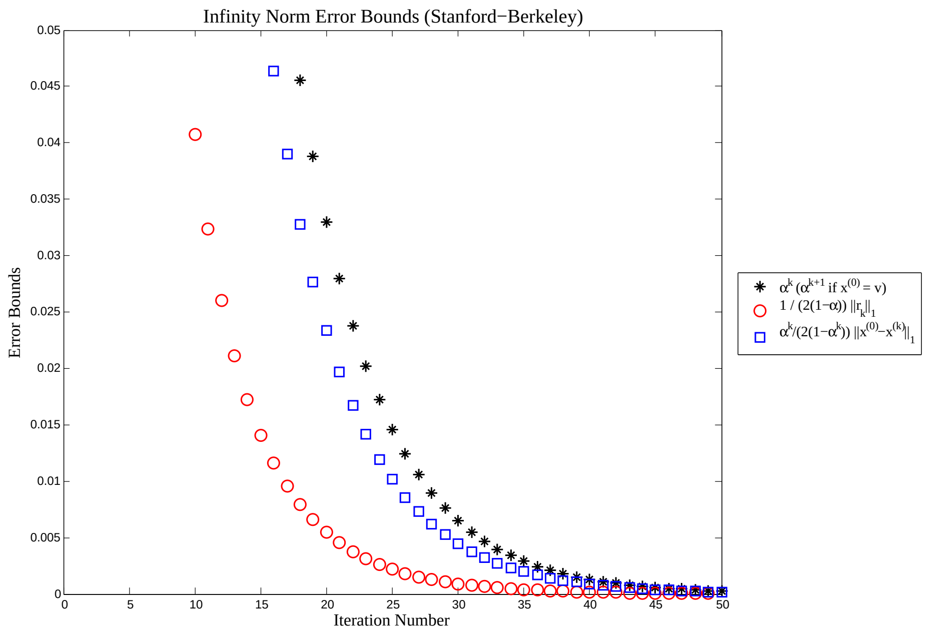


Figure 3.6: *wb-cs.stanford* and *Stanford* $(v = w = x^{(0)} = \left(\frac{1}{n}\right) \mathbb{1};\ \alpha = 0.85)$



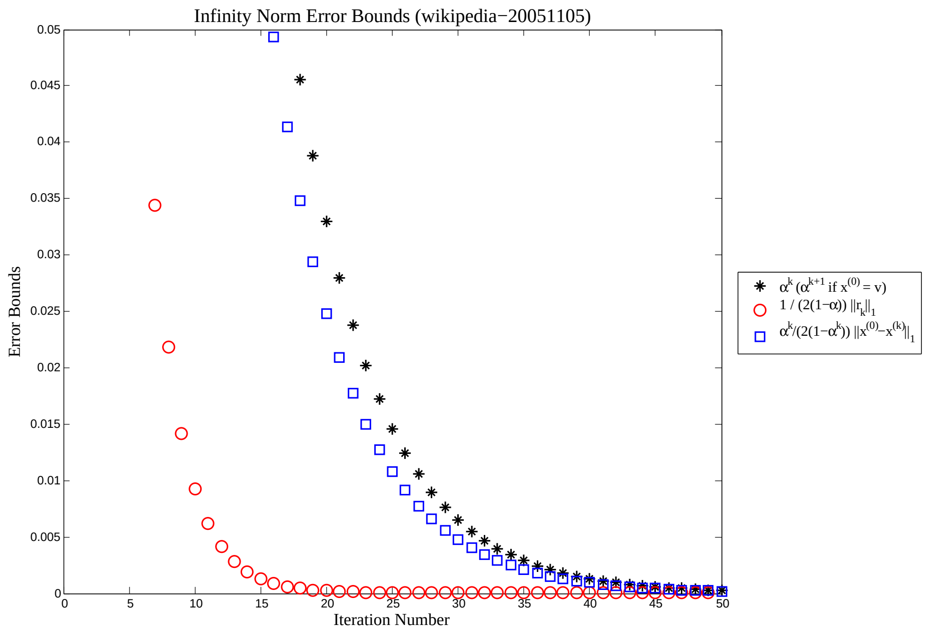
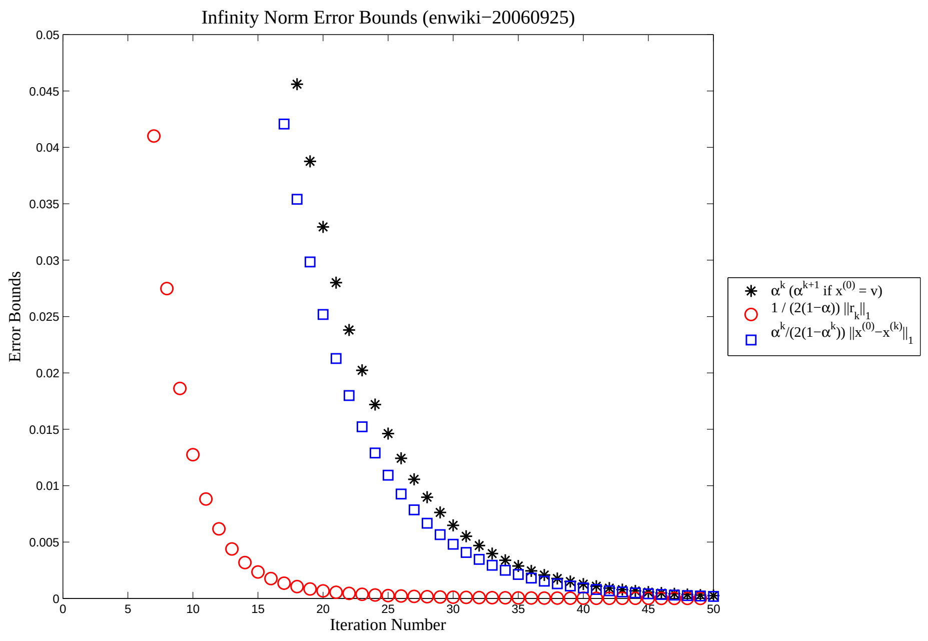


Figure 3.7: *Stanford-Berkeley* and *wikipedia-20051105* ($v = w = x^{(0)} = \left(\frac{1}{n}\right)\mathbb{1}$; $\alpha = 0.85$)



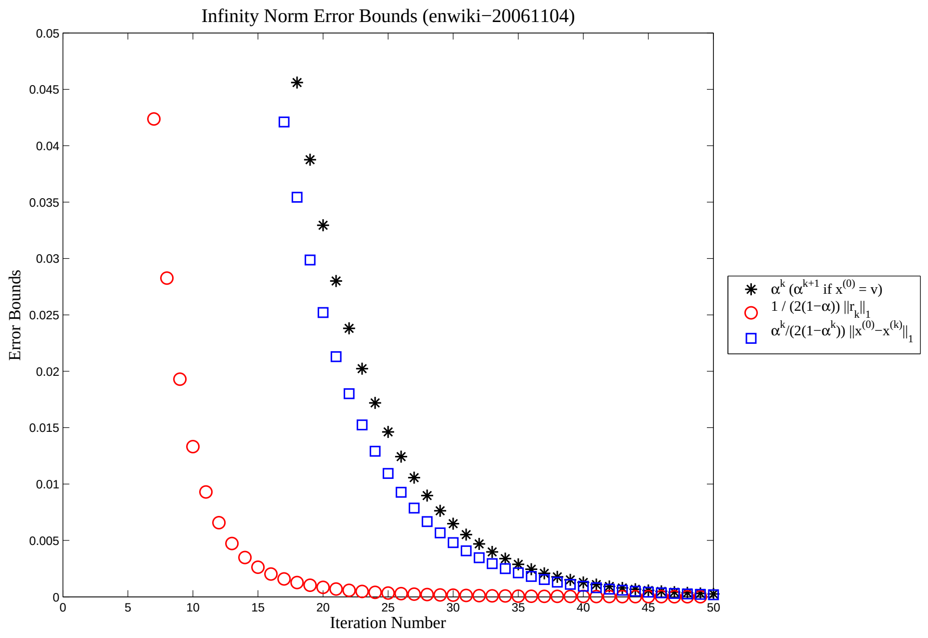


Figure 3.8: *enwiki-20060925* and *enwiki-20061104* ($v = w = x^{(0)} = \left(\frac{1}{n}\right) \mathbb{1}$; $\alpha = 0.85$)

The results are nearly identical for the Stanford and Stanford-Berkeley data sets and are similar for the Wikipedia data sets. Thus, we limit reports from additional experiments to the data sets *wb-cs.stanford*, *Stanford*, and *enwiki-20061104*.

For our second set of experiments, we defined the personalization vector v by assigning 0 to each element of v corresponding to a vertex with indegree or outdegree less than 10. We assigned equal weight to the remaining elements of v. (See Matlab code in Appendix B.) For the *wb-cs.stanford* data set, if $v_i \neq 0$, then $v_i = \frac{1}{165}$. For the Stanford data set, if $v_i \neq 0$, then $v_i = \frac{1}{23{,}088}$. For the *enwiki-20061104* data set, if $v_i \neq 0$, then $v_i = \frac{1}{388{,}201}$. This means that 1.66%, 8.19%, and 12.33% of all vertices in each respective data set have both indegree and outdegree greater than or equal to 10. We set $w = x^{(0)} = v$ and again chose $\alpha = 0.85$. See Figure 3.9 for a plot of each comparison. Notice that the bound in Theorem 3.13 once again outperforms the bound in Corollary 3.17.

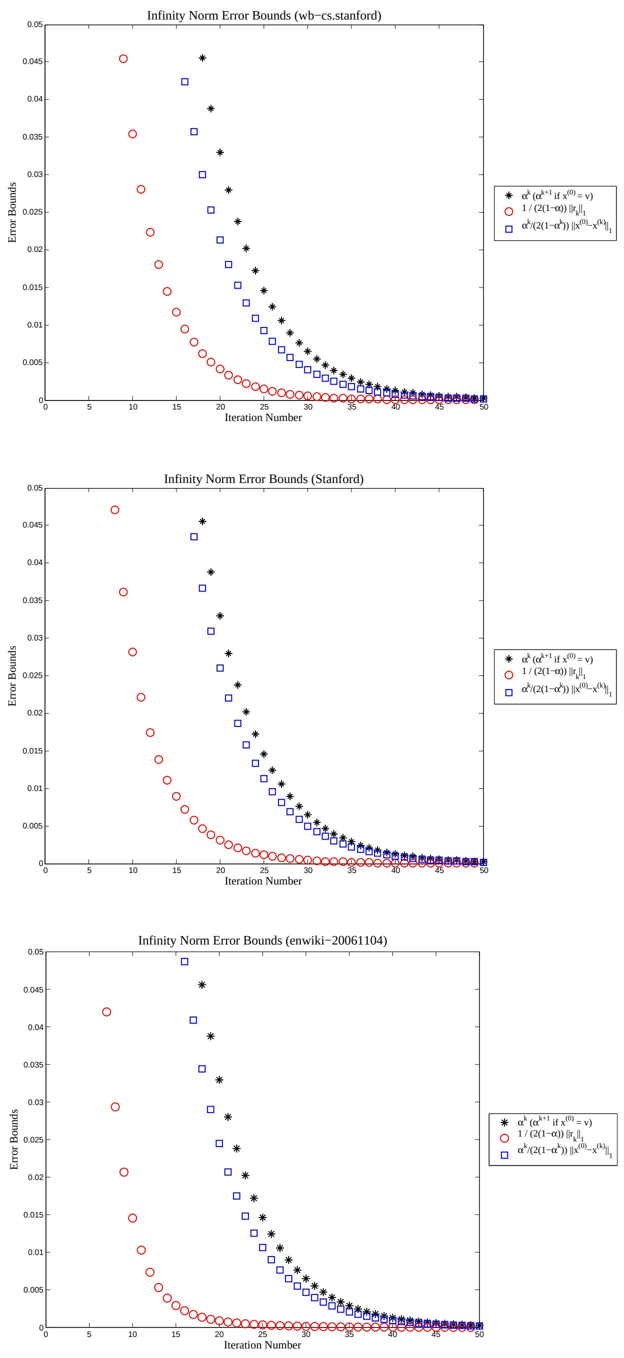


Figure 3.9: Infinity Norm Bounds ($\alpha = 0.85$; $v = w = x^{(0)}$, where v is defined by the Matlab code in Appendix B)

For all experiments so far, we defined the starting vector to be the personalization vector. We include one final experiment in this section for the enwiki-20061104 data set for which we chose a different starting vector. For this experiment, we defined $v = w = e_{1,715,731}$, the canonical vector with 1 in the $1,715,731$ position and zeros elsewhere (this corresponds to the vertex with largest indegree $168,685$). We chose starting vector $x^{(0)} = \left(\frac{1}{n}\right) \mathbb{1}$ and damping factor $\alpha = 0.85$.

Figure 3.10 shows that the bound in Corollary 3.17 is only slightly better than the upper bound α^k; however, the bound in Theorem 3.13 once again is much better than the upper bound α^k. This example suggests that the bound in Corollary 3.17 is more sensitive to the starting vector than the bound in Theorem 3.13, which is plausible since the bound is stated in terms of $x^{(0)}$.

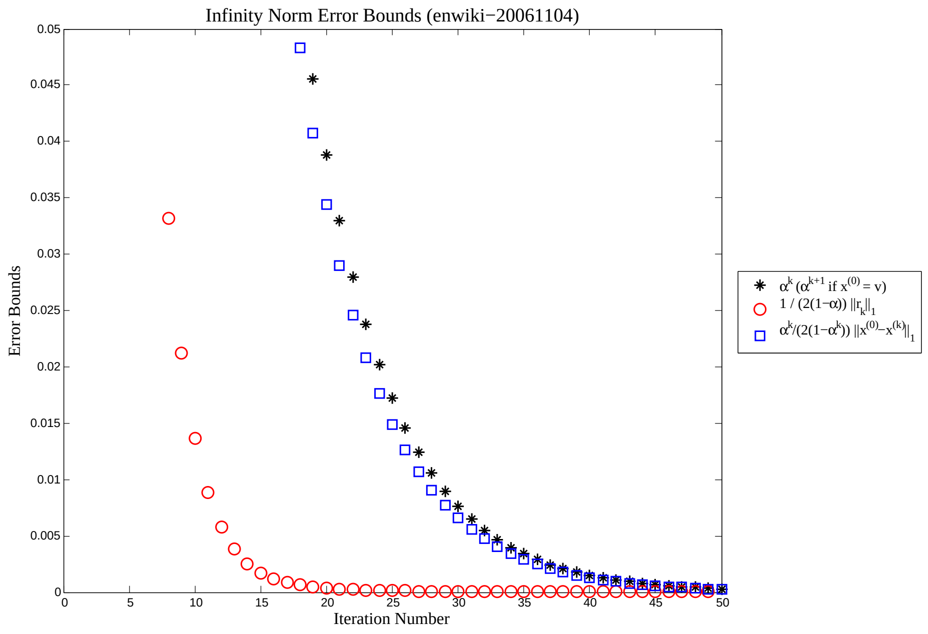


Figure 3.10: *enwiki_20061104* ($v = w = e_{1,715,731}$; $x^{(0)} = \left(\frac{1}{n}\right) \mathbb{1}$; $\alpha = 0.85$)

Chapter 4

Ranking Convergence of the Power Method

4.1 Ranking Distance

Brin and Page developed the PageRank algorithm for the purpose of ranking vertices of a simple directed graph (specifically, the Web graph). Thus, we should analyze the ranking of elements of power method iterate vectors when determining the accuracy of an approximation to π. As mentioned in Section 3.2, the residual norm is the most popular criterion for terminating iterations of the power method applied to G; however, determining convergence based on the residual norm fails to take into account the ranking of elements of the iterate vectors. In 1999 Haveliwala noted [36],

> The residual... is one possible measure of the convergence. A more useful approach to analyzing convergence involves looking at the *ordering* of pages induced by the *Rank* vector. If the PageRank values will be used strictly for determining the *relative* importance of pages, the convergence should be measured based on how the ordering changes as the number of iterations increases.

Haveliwala ran the PageRank algorithm on a Google matrix representing approximately 20 million webpages formed from a Stanford WebBase archive. He commented that "as few as 10 iterations can provide a useful ranking assignment," a much lower number than the iteration count necessary to reduce the residual norm to a specified value.

Comparing the ranking of elements of successive power method iterates is easy. Simply count the number of pairwise disagreements between the ranking of corresponding elements of each iterate. If the number is zero, the rankings agree. If the number is "small", the rankings are similar. The *Kendall's τ coefficient* is a popular measure of the correlation between two rankings of a listing of n items. Sir Kendall describes the coefficient τ as "a simple function of the minimum number of interchanges between neighbours required to transform one ranking into another" [48, §1.13]. Specifically, Kendall's τ is a number between -1 and 1, defined as

$$\tau \equiv 1 - \frac{2s}{\frac{1}{2}n(n-1)},$$

where s, the *Kendall's τ distance*, is the number of integer pairs appearing in the opposite order in two rankings of n items. Thus, given two *rank vectors*[1] of size n, σ and ϕ, the Kendall's τ distance between their rankings is

$$s \equiv \left|\{(i,j): \quad 1 \le i,j \le n, \quad \sigma(i) < \sigma(j) \quad \text{and} \quad \phi(i) > \phi(j)\}\right|.$$

[1]Here, we assume the elements of an n-vector are distinct real numbers, so the corresponding elements of the *rank vector* represent the ranks of the elements, where "1" is highest and "n" is lowest.

Note that $\sigma(i)$ is the position of i in σ in contrast to σ_i, the ith element of σ.

For example, let $\sigma^T = (1 \quad 6 \quad 2 \quad 3 \quad 4 \quad 5)$ and $\phi^T = (2 \quad 3 \quad 1 \quad 6 \quad 4 \quad 5)$. Then, $\sigma(2) = 3$ since 2 is the third element of σ. The Kendall's τ distance, s, between σ and ϕ is four since $s = |\{(1,2),(1,3),(6,2),(6,3)\}| = 4$. Kendall's τ is 7/15. We count $(1,2)$ as a discrepancy since $\sigma(1) < \sigma(2)$ and $\phi(1) > \phi(2)$ (note: $\sigma(1) = 1$, $\sigma(2) = 3$, $\phi(1) = 3$ and $\phi(2) = 1$). Kendall's τ distance, s, indicates that we can transform σ into ϕ in no fewer than the following four transpositions:

$$\sigma = \begin{pmatrix} 1 \\ 6 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \xrightarrow{(6,2)} \begin{pmatrix} 1 \\ 2 \\ 6 \\ 3 \\ 4 \\ 5 \end{pmatrix} \xrightarrow{(6,3)} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 6 \\ 4 \\ 5 \end{pmatrix} \xrightarrow{(1,2)} \begin{pmatrix} 2 \\ 1 \\ 3 \\ 6 \\ 4 \\ 5 \end{pmatrix} \xrightarrow{(1,3)} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 6 \\ 4 \\ 5 \end{pmatrix} = \phi.$$

Figure 4.1: Kendall's τ Distance Example

Recently, Kendall's-τ based residuals have become popular measures of convergence of iterative algorithms applied to the Google matrix [10, 23, 49, 63, 67, 73]. To compute PageRank scores for subsets of the entire Web graph, Kamvar, Haveliwala, Manning, and Golub suggest using the Kendall's τ distance between successive iterates to determine convergence [45]. They state, "When the Kendall's-τ residual is 0, that means the ordering is correct, and the local PageRank computation can be stopped."

Since the Web graph is large, computing Kendall's τ for all elements of successive iterates is expensive. Kamvar et al. define a measure called KDist that compares the distances between the top k rankings of successive iterates [45, 46]. KDist is identical to Kendall's τ when $k = n$. In addition, Kamvar et al. consider another top k measure based on Kendall's τ called $\mathrm{K}_{\mathrm{min}}$ defined by Fagin, Kumar, and Sikakumar [27]. Kamvar et al. provide empirical evidence to show that for the PageRank algorithm the one norm of the residual closely matches the KDist and $\mathrm{K}_{\mathrm{min}}$ distances between the top 100 rankings of successive iterates [46].

4.2 Incorrectly Ranked Approximations to PageRank

In this section, we investigate the effectiveness of the residual norm and of Kendall's τ based residuals in identifying and producing accurately ranked approximations to the PageRank vector. We provide simple examples to show the following: (1) correct ranking can be achieved for some power method iterate vector and destroyed for the next, (2) a small residual norm does not guarantee correct ranking, (3) zero ranking distance between power method iterate vectors does not guarantee correct ranking, and (4) correct ranking can occur for successive iterates before the norm-wise distance between the iterates is sufficiently small.

4.2.1 Examples to Support Claims

Directed Ring Graph

For the directed ring graph with n vertices in Figure 4.2, the Google matrix is $G = \alpha C + (1-\alpha)\mathbb{1}v^T$, where C is the $n \times n$ *basic circulant permutation matrix.*

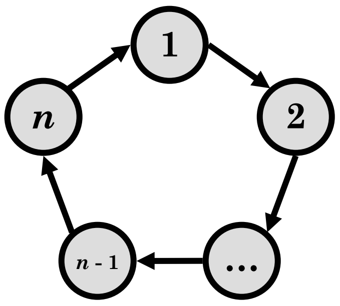


Figure 4.2: Directed Ring Graph with n Vertices

That is,

$$C \equiv \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Using $C^n = I$ [40, § 0.9.6] in $\left(\sum_{m=0}^{n-1} \alpha^m C^m\right)(I - \alpha C) = I - \alpha^n C^n$, gives

$$(I - \alpha C)^{-1} = \left(\tfrac{1}{1-\alpha^n}\right) \sum_{m=0}^{n-1} \alpha^m C^m.$$

Thus, the PageRank vector for the directed ring graph is

$$\pi^T = \left(\tfrac{1-\alpha}{1-\alpha^n}\right) \sum_{m=0}^{n-1} \alpha^m v^T C^m.$$

In addition, if $x^{(0)} \geq 0$ with $\|x^{(0)}\|_1 = 1$ is the initial power method iterate vector, then

$$\left[x^{(k)}\right]^T = \alpha^k \left[x^{(0)}\right]^T C^k + (1-\alpha) v^T \sum_{m=0}^{k-1} \alpha^m C^m, \qquad k \geq 1,$$

by Lemma 3.2.

Claim 4.1. Correct ranking can be achieved in some iteration and destroyed in the next.

For the directed ring graph with five vertices and personalization vector $v = e_1$, $\pi^T \approx (1 \quad \alpha \quad \alpha^2 \quad \alpha^3 \quad \alpha^4)$, where "$\approx$" means "a multiple of". Thus, if $\alpha \in (0,1)$, $rank(\pi^T) = (1 \quad 2 \quad 3 \quad 4 \quad 5)$. Table 4.1 shows that correct ranking is achieved after 5 iterations of the power method. (Note that correct ranking can occur after 3 or 4 iterations for small values of α, but for $\alpha > 0.725$, 5 iterations are necessary.) The sixth iterate is

$$\left[x^{(6)}\right]^T \approx \left(1+\alpha^5 \quad \alpha\left(1+\tfrac{\alpha^5}{1-\alpha}\right) \quad \alpha^2 \quad \alpha^3 \quad \alpha^4\right).$$

When $1+\alpha^5 < \alpha\left(1+\frac{\alpha^5}{1-\alpha}\right)$, the ranking of $x^{(6)}$ is $(2 \quad 1 \quad 3 \quad 4 \quad 5)$. This occurs when $\alpha > 0.716$. Furthermore, if $\alpha = 0.85$, the Kendall's τ distance, s, between successive iterates is not zero until $k = 20$; however, iterates corresponding to $k = 5, 10,$ and 15 are ranked correctly.

Table 4.1: Correct Ranking After 5 Iterations

Iteration k	**Approximate Iterate Vector**
0	$\begin{pmatrix}1 & 0 & 0 & 0 & 0\end{pmatrix}$
1	$\begin{pmatrix}1 & \frac{\alpha}{1-\alpha} & 0 & 0 & 0\end{pmatrix}$
2	$\begin{pmatrix}1 & \alpha & \frac{\alpha^2}{1-\alpha} & 0 & 0\end{pmatrix}$
3	$\begin{pmatrix}1 & \alpha & \alpha^2 & \frac{\alpha^3}{1-\alpha} & 0\end{pmatrix}$
4	$\begin{pmatrix}1 & \alpha & \alpha^2 & \alpha^3 & \frac{\alpha^4}{1-\alpha}\end{pmatrix}$
5	$\begin{pmatrix}1+\frac{\alpha^5}{1-\alpha} & \alpha & \alpha^2 & \alpha^3 & \alpha^4\end{pmatrix}$

Claim 4.2. A small residual norm does not guarantee correct ranking.

For the directed ring graph with 1,000 vertices and personalization vector $v = e_1$, $\pi^T \approx (1 \quad \alpha \quad \alpha^2 \quad ... \quad \alpha^{999})$. If $\alpha \in (0,1)$, $rank(\pi^T) = (1 \quad 2 \quad 3 \quad ... \quad 1000)$. If $x^{(0)} = v$, then 33 and 118 iterations again are necessary to reduce the one norm of the residual to 10^{-2} and 10^{-8}, respectively. This is due to the fact that

$$\|r_k\|_\infty = \|x^{(k)} - x^{(k+1)}\|_{1,\infty} \leq \alpha^k \|x^{(0)} - x^{(1)}\|_{1,\infty},$$

as stated in Corollary 3.4. The iterates corresponding to $k = 33$ and $k = 118$ are:

$$\left[x^{(33)}\right]^T \approx \begin{pmatrix}1 & \alpha & \alpha^2 & \cdots & \alpha^{32} & \frac{\alpha^{33}}{1-\alpha} & 0 & \cdots & 0\end{pmatrix},$$
$$\left[x^{(118)}\right]^T \approx \begin{pmatrix}1 & \alpha & \alpha^2 & \cdots & \alpha^{117} & \frac{\alpha^{118}}{1-\alpha} & 0 & \cdots & 0\end{pmatrix}.$$

Thus, many elements of $x^{(33)}$ and of $x^{(118)}$ receive the same ranking. In addition, if $\alpha = 0.85$, elements 107 through 118 of $x^{(118)}$ are incorrectly ranked since $\frac{\alpha^{118}}{1-\alpha} > \alpha^{107}$. The first correct ranking occurs for this example when $k = 1,000$.

Claim 4.3. Zero ranking distance between iterates does not guarantee correct ranking.

We return to the directed ring graph with five vertices and define the personalization vector $v^T = \frac{1}{15}(1 \quad 2 \quad 3 \quad 4 \quad 5)$. When $\alpha = 0.95$, $rank(\pi^T) = (3 \quad 5 \quad 4 \quad 2 \quad 1)$. If the initial power method iterate vector is the uniform vector, $x^{(0)} = \left(\frac{1}{5}\right)\mathbb{1}$, the Kendall's τ distance, s, between iterates 23 and 24 is zero, but $rank([x^{(23)}]^T) = (2 \quad 4 \quad 5 \quad 3 \quad 1)$, which does not match $rank(\pi^T)$. After 68 iterations, it appears that the correct ranking stabilizes.

Claim 4.4. Correct ranking can occur for successive iterates before the norm-wise distance between the iterates is sufficiently small.

For the example supporting Claim 4.1, let $\alpha = 0.85$ and $x^{(0)} = v$. Then, 33 and 118 iterations are necessary to reduce the one norm of the residual to 10^{-2} and 10^{-8}, respectively; however, correct ranking appears to stabilize in 20 iterations.

The example supporting Claim 4.2 also illustrates that correct ranking of iterate vector elements can take n iterations, where n is the order of the matrix. In addition, the examples show that ranking depends on α, n, v and the initial power method iterate vector.

4.2.2 Experiments

In this section, we provide a summary of the number of iterations required to satisfy the popular termination criteria for the small directed graph introduced in Chapter 2 and the *wb-cs.stanford* data set.

Small Directed Graph

Table 4.2 shows the number of iterations necessary to reduce the residual norm to $10^{-2}, 10^{-8}$, and 10^{-10} for Model 1 and Model 2 with two different initial iterate vectors. The table also includes the number of iterations required for element ranking agreement of successive iterate vectors. For these models and starting vectors, correct ranking occurs and appears to stabilize before $\|r_k\|_1 < 10^{-2}$. Thus, reducing the one norm of the residual to 10^{-2} requires more iterations than necessary to compute a correctly ranked approximation to the PageRank vector.

Table 4.2: Termination Criteria for Chapter 2 Example

	Damping Factor (α)	Personalization Vector (v^T)	Google Matrix (G)	PageRank Vector ($\approx \pi^T$)	Ranking of Vertices (1=Highest)
Model 1	0.85	$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{80} & \frac{71}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80} \\ \frac{37}{80} & \frac{3}{80} & \frac{3}{80} & \frac{37}{80} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$	$\begin{pmatrix} 0.21 & 0.26 & 0.31 & 0.21 \end{pmatrix}$	$\begin{pmatrix} 3 & 2 & 1 & 3 \end{pmatrix}$
Model 2	0.85	$\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \frac{3}{20} & \frac{17}{20} & 0 & 0 \\ \frac{3}{20} & 0 & \frac{17}{20} & 0 \\ \frac{23}{40} & 0 & 0 & \frac{17}{40} \\ \frac{29}{80} & \frac{17}{80} & \frac{17}{80} & \frac{17}{80} \end{pmatrix}$	$\begin{pmatrix} 0.30 & 0.28 & 0.27 & 0.15 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$

Model 1 $\|r_k\|_1 < \delta$			
$x^{(0)} = v$		$x^{(0)} = e_1$	
δ	Iterations	δ	Iterations
10^{-2}	7	10^{-2}	15
10^{-8}	42	10^{-8}	50
10^{-10}	54	10^{-10}	61

Model 2 $\|r_k\|_1 < \delta$			
$x^{(0)} = v$		$x^{(0)} = \frac{1}{4}\mathbb{1}$	
δ	Iterations	δ	Iterations
10^{-2}	15	10^{-2}	7
10^{-8}	50	10^{-8}	41
10^{-10}	61	10^{-10}	52

Model 1 Ranking Distance is Zero	
$x^{(0)} = v$	$x^{(0)} = e_1$
3	9

Model 2 Ranking Distance is Zero	
$x^{(0)} = v$	$x^{(0)} = e_1$
13	5

wb-cs.stanford **Data Set**

For the second experiment performed in Section 3.3.4 on the *wb-cs.stanford* data set, we defined the personalization vector v by assigning 0 to each element of v corresponding to a vertex with indegree or outdegree less than 10. We assigned equal weight to the remaining elements of v. For this definition, if $v_i \neq 0$, then $v_i = \frac{1}{165}$. We chose the damping factor $\alpha = 0.85$, and we set $w = x^{(0)} = v$. We return to this example to check the effectiveness and/or practicality of the residual norm and the Kendall's τ based residuals as termination criteria.

The number of iterations required to reduce the one norm of the residual to 10^{-2}, 10^{-8}, and 10^{-10} are $12, 83$, and 109, respectively. The Kendall's τ distance between successive iterate vectors is first 0 for iterations 220 and 221. This example supports Claim 4.2 (a small residual norm does not guarantee correct ranking). It appears that ranking stabilizes after iteration 220; however, this is a large number of iterations[2].

In addition to checking ranking agreement between all elements of successive iterate vectors, we checked for ranking agreement between the top 100 ranked elements of successive iterate vectors. The top 100 ranked elements first agree for iterations 20 and 21; however, they do not agree for iterations 23 and 24 and iterations 27 and 28. We observed no disagreements in the top 100 rankings after iteration 28. Based on these findings, the termination criterion $\|r_k\|_1 < 10^{-2}$ is satisfied before ranking

[2]The one norm of the residual is much less than 10^{-16} at this point.

has converged for even the top 100 PageRank scores. Also, these results show that ranking agreement between the top 100 elements of successive iterate vectors does not imply that ranking has converged for those elements.

4.3 Forward Error Bounds and Ranking

The examples in Section 4.2 show that the existing criteria for terminating iterations do not guarantee accurately ranked approximations to the PageRank vector. In fact, they do not guarantee an approximation to the PageRank vector that has similar rankings. In this section, we address this problem. Specifically, based on Theorem 3.13, we derive a condition that can determine relative rankings of elements of the PageRank vector[3]. This derivation closely parallels work from [50], and the first line of the proof appears as Equation 3.2 in [50]. Following the derivation, we show how to apply the condition to identify upper and lower bounds on the element rankings of the PageRank vector.

[3]We could make similar statements based on other results from Section 3.3.2. We chose to focus on the residual bound from Theorem 3.13 since it is easily computed and exhibited the best long-term performance in our experiments.

Theorem 4.5 (Relative Ranking Agreement)**.** Under the assumptions of Theorem 3.5,

$$\text{if } x_i^{(k)} > x_j^{(k)} + \tfrac{1}{1-\alpha}\|r_k\|_1, \text{ then } \pi_i > \pi_j.$$

Proof.

By Theorem 3.13, for any vertex i, $\left|x_i^{(k)} - \pi_i\right| \leq \|\epsilon_k\|_\infty \leq \frac{1}{2(1-\alpha)}\|r_k\|_1$.

Thus, for vertices i and j, $\left|x_i^{(k)} - \pi_i\right| + \left|x_j^{(k)} - \pi_j\right| \leq 2\left(\frac{1}{2(1-\alpha)}\|r_k\|_1\right) = \frac{1}{1-\alpha}\|r_k\|_1$.

Since $x_i^{(k)} - \pi_i + \pi_j - x_j^{(k)} \leq \left|x_i^{(k)} - \pi_i\right| + \left|x_j^{(k)} - \pi_j\right|$,

$x_i^{(k)} - \pi_i + \pi_j - x_j^{(k)} \leq \frac{1}{1-\alpha}\|r_k\|_1$.

If $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, then $0 < x_i^{(k)} - \left(x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1\right) \leq \pi_i - \pi_j$.

Therefore, $\pi_i > \pi_j$.

□

Remark 4.6. Theorem 4.5 also follows from

$$\left|x_i^{(k)} - \pi_i\right| + \left|x_j^{(k)} - \pi_j\right| \leq \|\epsilon_k\|_1 \leq \tfrac{1}{1-\alpha}\|r_k\|_1.$$

We adopt the ranking convention that assigns identical elements of a vector the same ranking. We also assume that if x elements of π (or $x^{(k)}$) receive the same ranking b, then the element(s) of π (or $x^{(k)}$) ranked just below these x elements receive ranking $b + x$. For example, if $\pi^T = (0.3 \quad 0.2 \quad 0.2 \quad 0.1 \quad 0.07 \quad 0.07 \quad 0.06)$, then $rank(\pi^T) = (1 \quad 2 \quad 2 \quad 4 \quad 5 \quad 5 \quad 7)$. With this ranking convention[4], we establish

[4]According to http://en.wikipedia.org/wiki/Ranking, this ranking method is called *competition ranking*.

upper and lower bounds on the ranking of elements of the PageRank vector, and we state conditions that identify actual element rankings of the PageRank vector. Note that if $\pi_i > \pi_j$, then $rank(\pi_i) < rank(\pi_j)$ since 1 is the highest ranking.

Theorem 4.7 (Element Rankings of the PageRank Vector)**.** Let $1 \leq b < a < n$. Under the assumptions of Theorem 3.5,

1. If $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, $rank(x_i^{(k)}) = b$, and $rank(x_j^{(k)}) = b+1$,

   then $rank(\pi_i) \leq b$.

2. If $x_l^{(k)} > x_i^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, $rank(x_l^{(k)}) = b$, $rank(x_i^{(k)}) = b+1$, and $rank(x_j^{(k)}) = b+2$, then $rank(\pi_i) = b+1$.

3. If $x_l^{(k)} > x_i^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$, $rank(x_l^{(k)}) = b$ and $rank(x_j^{(k)}) = a$, then $b < rank(\pi_i) < a$.

Proof.

For part 1, $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$ implies $\pi_i > \pi_j$.

Suppose $rank(x_i^{(k)}) = b = rank(x_j^{(k)}) - 1$.

Then, for all h such that $rank(x_h^{(k)}) \geq b+1, \pi_i > \pi_h$. Thus, $rank(\pi_i) \leq b$.

For part 2, $rank(\pi_i) \leq b+1$ (by part 1).

For every h such that $rank(x_h^{(k)}) \leq b, \pi_h > \pi_i$. Thus, $rank(\pi_i) = b+1$.

For part 3, $\pi_l > \pi_i$ and $\pi_i > \pi_j$, so $rank(\pi_l) < rank(\pi_i) < rank(\pi_j)$.

Thus, by parts 1 and 2, $a < rank(\pi_i) < b$. □

Since Theorem 4.5 requires $x_i^{(k)}$ to exceed $x_j^{(k)}$ by at least $\frac{1}{1-\alpha}\|r_k\|_1$ to conclude that π_i is greater than π_j, the inequality cannot identify the rankings of vertices with the same PageRank scores; however, Theorem 4.7 provides a means to say something about the rankings of these vertices by providing lower and upper bounds.

4.3.1 Experiments

We apply Theorem 4.7 to the small directed graph and *wb-cs.stanford* data set examples.

Small Directed Graph

For Model 1 with $x^{(0)} = \frac{1}{4}\mathbb{1}$, the ninth iterate vector is the first for which the inequality in Theorem 4.5 applies. It is $[x^{(9)}]^T \approx (0.2148 \quad 0.2638 \quad 0.3066 \quad 0.2148)$ and $rank([x^{(9)}]^T) = (3 \quad 2 \quad 1 \quad 3)$. The residual bound is $\frac{1}{1-0.85}\|x^{(9)} - x^{(10)}\|_1 \approx 0.0363$. Since $x_3^{(9)} > x_2^{(9)} + \frac{1}{1-0.85}\|x^{(9)} - x^{(10)}\|_1$ and $rank(x_2^{(9)}) = 2$, Theorem 4.7 guarantees that $rank(\pi_3) = 1$. Also, $x_2^{(9)} > x_1^{(9)} + \frac{1}{1-0.85}\|x^{(9)} - x^{(10)}\|_1$, and $rank(x_1^{(9)}) = 3$, so $rank(\pi_2) = 2$. Although the ranking distance between $x^{(2)}$ and $x^{(3)}$ is zero, the inequality in Theorem 4.7 is not satisfied until iteration 9. Also, notice that we actually need to compute the tenth iterate vector in order to check the bound $\frac{1}{1-\alpha}\|r_9\|_1$. Even so, after iteration 10, we can determine that vertex 3 receives the highest ranking

PageRank score and vertex 2 receives the second highest ranking. In addition, we know that vertices 1 and 4 cannot receive a ranking less than 3.

Table 4.3: Forward Error Bounds to Identify Ranking for Chapter 2 Example

	Damping Factor (α)	Personalization Vector (v^T)	Google Matrix (G)	PageRank Vector ($\approx \pi^T$)	Ranking of Vertices (1=Highest)
Model 1	0.85	$\begin{pmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}\frac{3}{80} & \frac{71}{80} & \frac{3}{80} & \frac{3}{80}\\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80}\\ \frac{37}{80} & \frac{3}{80} & \frac{3}{80} & \frac{37}{80}\\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\end{pmatrix}$	$\begin{pmatrix}0.21 & 0.26 & 0.31 & 0.21\end{pmatrix}$	$\begin{pmatrix}3 & 2 & 1 & 3\end{pmatrix}$
Model 2	0.85	$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}\frac{3}{20} & \frac{17}{20} & 0 & 0\\ \frac{3}{20} & 0 & \frac{17}{20} & 0\\ \frac{23}{40} & 0 & 0 & \frac{17}{40}\\ \frac{29}{80} & \frac{17}{80} & \frac{17}{80} & \frac{17}{80}\end{pmatrix}$	$\begin{pmatrix}0.30 & 0.28 & 0.27 & 0.15\end{pmatrix}$	$\begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix}$

For Model 2 with $x^{(0)} = e_1$, the inequality first identifies the ranking of PageRank scores for iterate vector $x^{(19)}$. Thus, only 20 iterations are required to know all element rankings of the PageRank vector.

wb-cs.stanford **Data Set**

It is not surprising that Theorem 4.5 quickly identified rankings of PageRank scores for the small directed graph examples. After all, the distance between each distinct PageRank score is fairly significant. For larger graphs, the distance between successive PageRank scores can be much smaller, so the number of iterations required to establish ranking bounds for a large percentage of vertices can increase. To see what information can be gained by application of Theorem 4.5 and Theorem 4.7 to larger graphs, we return a final time to the second experiment performed in Section 3.3.4 on the *wb-cs.stanford* data set. In Table 4.4, we again list some of the characteristics of this data set.

Table 4.4: Characteristics of *wb-cs.stanford* data set

Vertices	**Directed Edges**	**Unreferenced Vertices**	**Dangling Nodes**	**Indegree < 10**	**Outdegree < 10**
9,914	36,854	699 (7.05%)	2,861 (28.86%)	9,488 (95.70%)	9,381 (94.62%)

For the second experiment performed on the *wb-cs.stanford* data set, recall that we defined the personalization vector v by assigning 0 to each element of v corresponding to a vertex with indegree or outdegree less than 10. We assigned equal weight to the remaining elements of v. For this definition, if $v_i \neq 0$, then $v_i = \frac{1}{165}$. We chose the damping factor $\alpha = 0.85$, and we set $w = x^{(0)} = v$.

Based on our definition of v, Theorem 3.7 guarantees that the PageRank scores for the unreferenced vertices are 0, so the unreferenced vertices receive the lowest ranking[5]. Thus, we know the PageRank scores and ranking of those scores for 7.05%

[5]Depending on how the personalization vector is defined, unreferenced vertices can have different PageRank scores. In addition, the ranking of the scores might not converge for many iterations.

of vertices before starting the power method. In addition, Theorem 3.7 implies that PageRank scores for vertices only on short paths converge quickly; however, the theorem does not imply that their rankings converge quickly.

We tested elements of each iterate vector according to part 1 of Theorem 4.7. That is, in each iteration, we only checked to see if $x_i^{(k)} > x_j^{(k)} + \frac{1}{1-\alpha}\|r_k\|_1$ for $rank(x_i^{(k)}) = rank(x_j^{(k)}) - 1$. We limited our testing to this because we cannot obtain a tight lower bound on the ranking of the PageRank score for vertex i unless this is satisfied[6]. Of course, this also produces upper bounds for rankings of PageRank scores.

In Section 4.2.2, we stated that the number of iterations required to reduce the one norm of the residual to 10^{-2}, 10^{-8}, and 10^{-10} are $12, 83$, and 109, respectively. In Table 4.5, we summarize the ranking information we obtain for these iteration counts.

Table 4.5: Ranking Information for *wb-cs.stanford* Example

Residual Norm	Iteration	Vertices Satisfying Condition	Lowest Ranking Satisfying Condition
10^{-2}	12	0	N/A
10^{-8}	83	1,288	5,389
10^{-10}	109	3,205	8,118

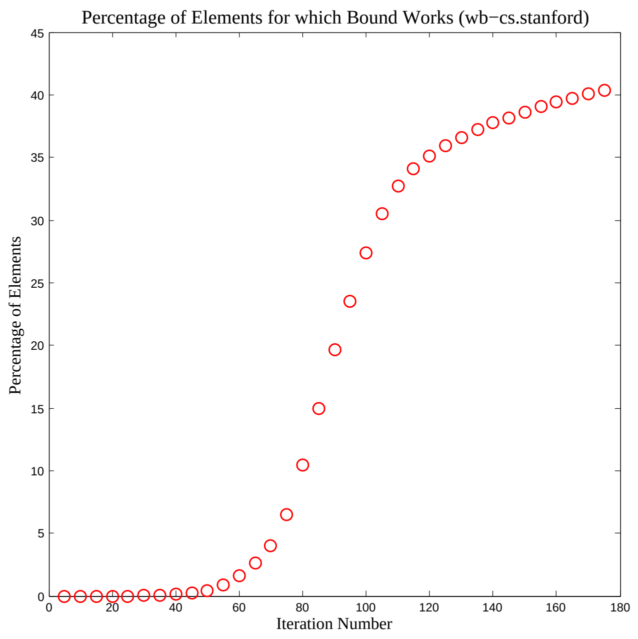
[6]See appendix for Matlab code.

The condition is not satisfied before iteration 29. At that point, the highest ranked vertex can be identified. The second highest ranked vertex is known after iteration 31. By iteration 48, we are able to identify the top 100 rankings (although we are not able to establish the exact rankings for them at this point). Figure 4.3 shows the percentage of vertices satisfying the condition for each iteration (in multiples of 5 up to 180 iterations). Also provided is the percentage of indistinct elements. Although exact rankings cannot be inferred for vertices with indistinct values, the inequality does identify upper and lower bounds on their rankings.

4.3.2 Discussion

Our experiments show that the inequality in Theorem 4.5 does not apply to all components of the PageRank vector. For instance, it cannot identify rankings of vertices with indistinct PageRank scores; however, it can provide upper and lower bounds for their rankings. Also, even when the inequality cannot identify the top r rankings, it might be able to identify the vertices that receive a ranking no greater than r.

For example, Google does not reveal actual PageRank scores; however, Google's toolbar includes a PageRank display feature that provides "an indication of the PageRank" for a webpage being visited [5]. The PageRank scores on the toolbar are integer values from 0 (lowest) to 10 (highest). Although some search engine optimization experts discount the accuracy and the importance of toolbar scores [32, 33,



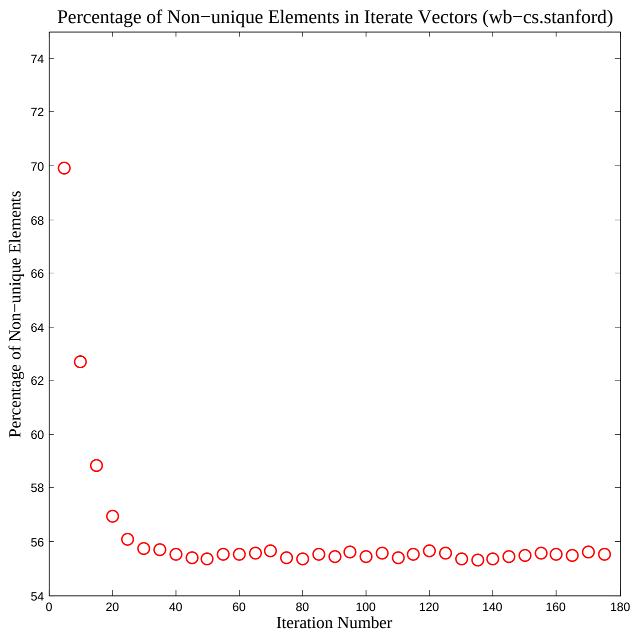


Figure 4.3: *wb-cs.stanford* ($\alpha = 0.85$; $v = w = x^{(0)}$, where $v_i = 0$ if the indegree or outdegree of vertex i is less than 10 and $v_i = \frac{1}{165}$, otherwise)

64], a Google webpage on toolbar features [4] states:

> PageRank Display: Wondering whether a new website is worth your time? Use the Toolbar's PageRank$^{\text{TM}}$ display to tell you how Google's algorithms assess the importance of the page you're viewing.

Google does not explain how toolbar PageRank scores are determined. If ranking is ignored when computing PageRank scores, it is possible that webpages are assigned toolbar scores that are too high or too low. Theorem 4.7 can be used to determine which webpages receive the same toolbar PageRank score. For example, if Theorem 4.7 identifies the top 100 ranked vertices, these vertices could be assigned toolbar PageRank score 10. For iteration 109 in the *wb-cs.stanford* example above, Theorem 4.7 applied for the vertex ranked 8,118 but did not apply for any vertices with lower ranking. This could be used to justify assigning toolbar PageRank score 0 to all vertices ranked lower than $8,118$ in the 108th iterate vector.

Chapter 5

Conclusion

5.1 Summary of Contributions

The following statements appearing in [36, 45, 46] inspired this research:

> The residual... is one possible measure of the convergence. A more useful approach to analyzing convergence involves looking at the *ordering* of pages induced by the *Rank* vector. If the PageRank values will be used strictly for determining the *relative* importance of pages, the convergence should be measured based on how the ordering changes as the number of iterations increases. [36]

> This suggests a stopping criteria [sic] for local PageRank computations. At each stage in a local PageRank computation, one could compute the Kendall's-τ residual ... When the Kendall's-τ residual is 0, that means the ordering is correct, and the local PageRank computation can be stopped. [45]

> A rigorous explanation for the close match between the L_1 residual and the Kendall's τ based residuals is an interesting avenue of future investigation. [46]

Finding no analysis of these issues in the literature, we began ours. Our earliest results appear in Section 4.2.1. We found very simple examples to support the following: (1) correct ranking can be achieved for some power method iterate vector and destroyed for the next, (2) a small residual norm does not guarantee correct ranking, (3) zero ranking distance between power method iterate vectors does not guarantee correct ranking, and (4) correct ranking can occur for successive iterates before the norm-wise distance between the iterates is sufficiently small.

After making these observations, we sought a way to efficiently identify when computed PageRank scores are accurately ranked. Steve Kirkland's paper [50] provided the main idea as well as many useful tools for our analysis. In fact, our experiments illustrate that a computationally efficient criterion for determining relative ranking of PageRank scores follows directly from [50, Equation 3.2]. Many theoretical contributions in Section 3.3 employ Lemma 3.2, and the proof of [50, Theorem 3.1] brought this result to our attention. Theorem 4.5 makes it possible to determine ranking bounds for PageRank scores, which accomplishes what we set out to do.

Additional contributions include a new statement for π (Theorem 3.14) and a new statement for the forward error (Theorem 3.15) both in terms of $x^{(0)}$ and $x^{(k)}$ for

$k \geq 1$. Also, our component-wise error bound in Theorem 3.6 led to Theorem 3.7, which implies that PageRank scores converge in k iterations for vertices with longest path lengths $k-1$, where $k < n$. Since unreferenced vertices have indegree 0, the PageRank scores for these vertices depend only on the personalization vector v and dangling node fix w.

5.2 Future Research

5.2.1 Short-term

- Incorporate adjustments to Theorem 4.5 to account for finite precision errors to ensure accurate element rankings of computed iterates.
- Formalize procedures to implement Theorem 4.7 to find exact, top k, and/or interval rankings of PageRank scores. For instance, determine at what iteration count to begin checking ranking conditions.
- Find good choices for personalization, dangling node fix, and initial iterate vectors.
- Determine which iterate elements converge fastest with respect to ranking.
- Identify if restrictions on components of the Google matrix exist to ensure

that ranking agreement between successive iterate vectors implies correct ranking.

- Apply the PageRank algorithm to specific classes of directed graphs.

5.2.2 Long-term

- Extend analysis to other ranking algorithms that make use of Perron-Frobenius theory to approximate the dominant eigenvector of non-negative matrices.

  Two other popular Web ranking algorithms are HITS [52] and SALSA [58]. Other algorithms rank football teams, students by examination scores, best methods on which to base admissions tests, social alternatives, etc. [11, 39, 47, 66, 70, 74]. Each of these algorithms uses the power method to approximate a dominant eigenvector (right or left) of a nonnegative data matrix; however, none of these algorithms takes into account ranking to determine accuracy of an iterate.

- Develop algorithms similar to PageRank for other ranking schemes.

List of References

[1] *http://www.stanford.edu/˜dgleich/data/*. datasets graphs and more, organized by David Gleich.

[2] *http://www.google.com/corporate/history.html*. Google Corporate Information: Google Milestones.

[3] *http://www.google.com/technology/index.html*. Our Search: Google Technology.

[4] *http://www.google.com/support/toolbar/*. Google Toolbar (Select Features).

[5] *http://toolbar.google.com/options_help.html*. Google Toolbar: About Toolbar Option.

[6] *http://www.uspto.gov/main/patents.htm*. United States Patent and Trademark Office official website.

[7] *http://www.stanford.edu/˜sdkamvar/research.html*. Stanford University Homepage for Sepandar D. Kamvar.

[8] *http://www.webrankinfo.com/english/seo-news/topic-16388.htm*, January 2006. Increased Google Index Size?

[9] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin, *PageRank computation and the structure of the Web: Experiments and algorithms*, in The Eleventh International WWW Conference, ACM Press, New York, May 2002.

[10] R. Baeza-Yates, P. Boldi, and C. Castillo, *Generalizing PageRank: damping functions for link-based ranking algorithms*, in Proceeding of ACM SIGIR, Seattle, Washington, USA, August 2006, ACM Press, pp. 308–315.

[11] E. Barbeau, *Perron's result and a decision on admissions tests*, Mathematics Magazine, 59 (1986), pp. 12 – 22.

[12] J. Battelle, *The Search: How Google and its Rivals Rewrote the Rules of Business and Transformed our Culture*, Penguin Group, 2005.

[13] P. Berkhin, *A survey on PageRank computing*, Internet Mathematics, 2 (2005), pp. 73–120.

[14] M. Bianchini, M. Gori, and F. Scarselli, *Inside PageRank*, ACM Trans. on Inter. Tech., 5 (2005), pp. 92–128.

[15] P. Boldi, M. Santini, and S. Vigna, *Pagerank as a function of the damping factor*, in WWW '05: Proceedings of the 14th international conference on World Wide Web, New York, NY, USA, 2005, ACM Press, pp. 557–566.

[16] C. Brezinski and M. Redivo-Zaglia, *The PageRank vector: Properties, computation, approximation, and acceleration*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 551–575.

[17] C. Brezinski, M. Redivo-Zaglia, and S. Serra-Capizzano, *Extrapolation methods for PageRank computations*, Comptes Rendus de l'Académie des Sciences de Paris, Series I, 340 (2005), pp. 393–397.

[18] S. Brin and L. Page, *The anatomy of a large-scale hypertextual Web search engine*, Comput. Networks and ISDN Systems, 30 (1998), pp. 107–117.

[19] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen, *Efficient PageRank approximation via graph aggregation*, Information Retrieval, 9 (2006), pp. 123–138.

[20] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, Dover, New York, 1991.

[21] G. E. Cho and C. D. Meyer, *Comparison of perturbation bounds for the stationary distribution of a Markov chain*, Linear Algebra and its Applications, 335 (2001), pp. 137–150.

[22] G. Dahlquist and Åke Björck, *Numerical Methods in Scientific Computing*, vol. II, SIAM, Philadelphia, to be published. http://www.math.liu.se/~akbjo/dqbjch9.pdf.

[23] J. V. Davis and I. S. Dhillon, *Estimating the global PageRank of Web communities*, in KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2006, ACM Press, pp. 116–125.

[24] C. Ding, X. He, P. Husbands, H. Zha, and H. D. Simon, *Pagerank, hits and a unified framework for link analysis*, in SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, 2002, ACM Press, pp. 353–354.

[25] N. Eiron, K. S. McCurley, and J. A. Tomlin, *Ranking the Web frontier*, 2004.

[26] L. Eldén, *A note on the eigenvalues of the Google matrix*, tech. report, LiTH-MAT-R-04-01, Department of Mathematics, Linköping University, 2004.

[27] R. Fagin, R. Kumar, and D. Sivakumar, *Comparing top* k *lists*, SIAM J. Discrete Math., 17 (2003), pp. 134–160.

[28] D. Gleich, L. Zhukov, and P. Berkhin, *Fast parallel PageRank: A linear system approach*, tech. report, Yahoo!, 2004.

[29] G. H. Golub and C. Greif, *Arnoldi-type algorithms for computing stationary distribution vectors, with application to PageRank*, Tech. Report SCCM-04-15, Stanford University, 2004.

[30] ——, *An Arnoldi-type algorithm for computing PageRank*, BIT, 46 (2006), pp. 759–771.

[31] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 3rd ed., 1996.

[32] M. Grehan, *What price PageRank?* ClickZ Network, July 2005. www.clickz.com.

[33] ——, *What price PageRank? Part 2.* ClickZ Network, July 2005. www.clickz.com.

[34] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, *Combating Web spam with TrustRank*, in Proceedings of the 30th International Conference on Very Large Databases, Morgan Kaufmann, 2004, pp. 576–587.

[35] T. Haveliwala, S. Kamvar, D. Klein, C. Manning, and G. Golub, *Computing PageRank using power extrapolation*, tech. report, Stanford University, 2003.

[36] T. H. Haveliwala, *Efficient computation of PageRank*, Tech. Report 1999-31, Stanford University, 1999.

[37] T. H. Haveliwala and S. D. Kamvar, *The second eigenvalue of the Google matrix*, Tech. Report 2003-20, Stanford University, 2003.

[38] M. Haviv and L. V. D. Heyden, *Perturbation bounds for the stationary probabilities of a finite markov chain*, Adv. Appl. Prob., 16 (1984), pp. 804–818.

[39] I. Hofuku and K. Oshima, *A total ranking based on examination scores for a small number of subjects.* Tokyo University of Science, School of Management, Discussion Paper, November 2002.

[40] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[41] I. C. Ipsen and S. Kirkland, *Convergence analysis of a PageRank updating algorithm by Langville and Meyer*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 952–967.

[42] I. C. Ipsen and T. M. Selee, *PageRank computation, with special attention to dangling nodes.* Accepted for publication in SIAM J. Matrix Anal. Appl., 2007.

[43] I. C. Ipsen and R. S. Wills, *Mathematical properties and analysis of Google's PageRank*, Bol. Soc. Esp. Mat. Apl., 34 (2006), pp. 191–196.

[44] S. Kamvar, T. Haveliwala, and G. Golub, *Adaptive methods for the computation of PageRank*, Linear Algebra Appl., 386 (2004), pp. 51–65.

[45] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, *Exploiting the block structure of the Web for computing PageRank*, tech. report, Technical Report 2003-17, Stanford University, 2003.

[46] ——, *Extrapolation methods for accelerating PageRank computations*, in The Twelfth International World Wide Web Conference, ACM Press, New York, 2003, pp. 261–270.

[47] J. P. Keener, *The Perron-Frobenius theorem and the ranking of football teams*, SIAM Review, 35 (1993), pp. 80 – 93.

[48] S. M. G. Kendall, *Rank Correlation Methods*, Charles Griffin & Company Limited, 1975.

[49] A. Khalil and Y. Liu, *Experiments with PageRank computation*, tech. report, Technical Report 603, Computer Science Department at Indiana University, 2004.

[50] S. J. Kirkland, *Conditioning of the entries in the stationary vector of a Google-type matrix*, Linear Algebra and its Applications, 418 (2006), pp. 665–681.

[51] S. J. Kirkland, M. Neumann, and B. L. Shader, *Applications of Paz's inequality to perturbation bounds for Markov chains*, Linear Algebra and its Applications, 268 (1998), pp. 183–196.

[52] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM, 46 (1999).

[53] A. N. Langville and C. D. Meyer, *Deeper inside PageRank*, Internet Mathematics, 1 (2005), pp. 335–380.

[54] ——, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, New Jersey, 2006.

[55] ——, *A reordering for the PageRank problem*, SIAM J. Sci, Comput., 27 (2006), pp. 2112–2120.

[56] ——, *Updating Markov chains with an eye on Google's PageRank*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 968–987.

[57] C. P. Lee, G. H. Golub, and S. A. Zenios, *A fast two-stage algorithm for computing PageRank and its extensions*, tech. report, Stanford University, 2003.

[58] R. Lempel and S. Moran, *The stochastic approach for link-structure analysis (SALSA) and the TKC effect*, in The Ninth International World Wide Web Conference, ACM Press, New York, 2000.

[59] J. McDonald, M. Newmann, H. Schneider, and M. Tsatsomeros, *Inverse* M-*matrix inequalities and generalized ultrametric matrices*, Linear Algebra and Its Applications, 220 (1995), pp. 321–341.

[60] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, PA, 2000.

[61] A. Nanavati, A. Chakraborty, D. Deangelis, H. Godil, and T. D'Silva, *An investigation of documents on the World Wide Web.* http://www.iit.edu/˜dsiltho/Investigation.pdf, December 2004.

[62] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation ranking: Bringing order to the Web*, tech. report, Stanford University, 1998.

[63] F. Qiu and J. Cho, *Automatic identification of user interest for personalized search*, in WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, 2006, ACM Press, pp. 727–736.

[64] C. Ridings and M. Shishigin, *PageRank uncovered.* Technical Paper for the Search Engine Optimization Online Community.

[65] F. Rogers, *Life's Journeys According to Mister Rogers: Things to Remember Along the Way*, Family Communications, Inc., New York, NY, 2005.

[66] T. L. Saaty, *Rank according to Perron: A new insight*, Mathematics Magazine, 60 (1987), pp. 211 – 213.

[67] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz, *To randomize or not to randomize: space optimal summaries for hyperlink analysis*, in WWW '06: Proceedings of the 15th International Conference on World Wide Web, New York, NY, USA, 2006, ACM Press, pp. 297–306.

[68] E. Seneta, *Non-negative Matrices and Markov Chains*, Springer-Verlag, 2nd ed., 1981.

[69] S. Serra-Capizzano, *Jordan canonical form of the Google matrix: A potential contribution to the PageRank computation*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 305–312.

[70] G. Slutzki and O. Volij, *Scoring of web pages and tournaments - axiomatizations*, Social Choice and Welfare, 26 (2006), pp. 75–92.

[71] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[72] S. Vigna, *Stanford matrix considered harmful.* http://vigna.dsi.unimi.it/papers.php, 2007.

[73] Y. Wang and D. J. DeWitt, *Computing PageRank in a distributed Internet search system*, in Proceedings of the 30th VLDB Conference, 2004.

[74] S. White and P. Smyth, *Algorithms for estimating relative importance in networks*, in KDD ́03: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2003, ACM Press, pp. 266–275.

[75] R. S. Wills, *Google's PageRank: The math behind the search engine*, Math. Intelligencer, 28 (2006), pp. 6–11.

Appendices

Appendix A

Matlab File: adjacency_to_hyperlink.m

```
function
[A,p,total_in,D_in,q,total_out,D_out,D_out_keep,n]=adjacency_to_hyperlink(A);
%
% A is an adjacency matrix meaning A_ij = 1 if (i,j) is an element of the
% directed graph, and A_ij = 0 otherwise.
%
% A = hyperlink matrix
% p = largest indegree plus one
% total_in = vector with number of vertices for each indegree 0 to p-1
% D_in = vector with the indegree of each vertex
% q = largest outdegree plus one
% total_out = vector with number of vertices for each outdegree 0 to q-1
% D_out = vector used to create the hyperlink matrix
% D_out_keep = vector with the outdegree of each vertex
% n = size of A
%
%

[m,n]=size(A);

D_in=zeros(n,1);

for i=1:n
    D_in(i,1)=nnz(A(:,i));
end

'Finished forming D_in'

D_out = zeros(n,1);
B=sparse(A'); % Matlab is performs nnz(A(i,:)) too slowly.
for i=1:n
    D_out(i,1)=nnz(B(:,i));
end

'Finished forming D_out'


D_out_keep = D_out;

% Redefining D_out to create the hyperlink matrix.

for i=1:n
    if D_out(i,1)~=0
       D_out(i,1)=1/D_out(i,1);
    end
end

'Finished redefining D_out to be used to create hyperlink matrix'

for i = 1:n
    B(:,i)=D_out(i,1)*B(:,i);
end
A = B';
clear B; % Clearing B since we no longer need it
```

```
'Finished creating hyperlink matrix'

p=max(D_in)+1;
total_in=zeros(p,1);

for j=1:n
    for i=1:min(50,p)
        if D_in(j,1)==i-1
           total_in(i,1)=total_in(i,1)+1;
        end
    end
end

'Finished determining indegree numbers'

q=max(D_out_keep)+1;
total_out=zeros(q,1);

for j=1:n
    for i=1:min(50,q)
        if D_out_keep(j,1)==i-1
           total_out(i,1)=total_out(i,1)+1;
        end
    end
end

'Finished determining outdegree numbers'

'Finished'
```

Appendix B

Matlab File: pers_vector.m

Matlab File: pers_vector.m

```
function [v]=pers_vector(D_in, D_out_keep, n);

v=ones(1,n);
for i=1:n
    if D_in(i,1)<10
       v(1,i)=0;
    end
    if D_out_keep(i,1)<10
       v(1,i)=0;
    end
end


norm(v,1)
v=v/norm(v,1);
```

Appendix C

Matlab File: new_pagerank_experiments.m

```
function [x,up_err_bd,res_k_bd,k_err_bd,k_max,ind_k_max,res_k_one,res_k_inf,
bound_works,k_tau,k_tau_top_k,which_ones,rank_xk,applies]
=new_pagerank_experiments(H,alpha,x0,max_it,v,w,top_k);

%
% To convert adjacency matrices to hyperlink matrices, first run:
%
[A,p,total_in,D_in,q,total_out,D_out,D_out_keep,n]=adjacency_to_hyperlink(A);
%
% The code for finding the dangling node vector, d, appears in Langville
% and Meyer's book, "Google's PageRank and Beyond."
%
% The function new_pagerank_experiments uses the power method.
%
% The code was not written with speed in mind.  It was written to test
% results only.
%
%
% INPUT:
% H = hyperlink matrix (n x n)
% alpha = damping factor (between 0 and 1)
% x0 = starting vector (1 x n)
% max_it = number of iterations to run
% v = personalization vector (1 x n)
% w = dangling node fix vector (1 x n)
% top_k = top_k values to return for any iterate vector
%
% OUTPUT:
% x = last iterate vector (corresponding to max_it)
% up_err_bd (max_it by 1) for Cor. 3.8
% res_k_bd (max_it by 1) for Thm. 3.13
% k_err_bd (max_it by 1) for Cor. 3.17
% k_max (top_k x max_it) returns the top_k elements for each iteration
% ind_k_max (top_k x max_it) returns the index of the top_k elements for
%       each iteration
% res_k_one (max_it x 1) contains ||r_k||_1 for each iteration
% res_k_inf (max_it x 1) contains ||r_k||_inf for each iteration
% bound_works (top_k-1 x max_it) checks to see when
%       min(k_error_bound_inf,res_k_inf_bound) begins identifying ranking
% k_tau (max_it x 1) checks to see when successive iterates have same
%       ranking
% k_tau_top_k (max_it x 1) checks to see when the top_k iterates have same
%       ranking
% which_ones (top_k-1 x max_it)
% rank_xk (top_k-1 x max_it)
% applies (2*top_k-1 x 2*max_it)
%
%

[m,n]=size(H);

% Calculating the dangling node vector d:
% We need to know which rows have all zero entries.
```

```
row_sum_H=ones(1,n)*H';
non_zero_rows_H=find(row_sum_H);
zero_rows_H=setdiff(1:n,non_zero_rows_H);
z = length(zero_rows_H);
% Currently, the dangling node vector is the following:
d=sparse(zero_rows_H,ones(z,1),ones(z,1),n,1);
beta=1-alpha;
gamma=1/beta;
% Calculating the PageRank vector:
iterations = 0;
x = x0;
res_k_one=zeros(max_it,1);
res_k_inf=zeros(max_it,1);
res_k_bd=zeros(max_it,1);
up_err_bd=zeros(max_it,1);
alpha_k=zeros(max_it,1);
alpha_k_1=zeros(max_it,1);
one_norm_x0_xk=zeros(max_it,1);
k_err_bd=zeros(max_it,1);
k_max=zeros(top_k,max_it);
ind_k_max=zeros(top_k,max_it);
k_tau=zeros(max_it,1);
k_tau_top_k=zeros(max_it,1);
x_sort_old = zeros(n,1);
index_x_sort_old=zeros(n,1);
for i = 1:max_it
    x_old = x;
    x = alpha*x*H+alpha*(x*d)*w+beta*v;
    one_norm_x0_xk(i,1)=norm(x0-x,1);
    k_err_bd(i,1)=alpha^i/(1-alpha^i)*one_norm_x0_xk(i,1);
    alpha_k(i,1)=alpha^i;
    alpha_k_1(i,1)=alpha^(i+1);
    res_k_one(i,1)=norm(x-x_old,1);
    res_k_inf(i,1)=norm(x-x_old,inf);
    res_k_bd(i,1)=gamma*res_k_one(i,1); %This will produce the bound for the
prior iterate.
    [x_sort,index_x_sort]=sort(x','descend');
    k_max(:,i)=x_sort(1:top_k,1);
    ind_k_max(:,i)=index_x_sort(1:top_k,1);
    if index_x_sort_old == index_x_sort
        k_tau(i,1)=i;
    end
    if index_x_sort_old(1:top_k,1)==index_x_sort(1:top_k,1)
        k_tau_top_k(i,1)=i;
    end
    x_sort_old = x_sort;
    index_x_sort_old = index_x_sort;

end
bound_works=zeros(top_k-1,max_it);
which_ones=zeros(top_k-1,max_it);
rank_xk=zeros(top_k-1,max_it);
applies=zeros(2*top_k-1,2*max_it);
```

```
for i = 1:max_it-1
    for j = 1:top_k-1
        if k_max(j,i) > k_max(j+1,i) + res_k_bd(i+1,1)
        %if k_max(j,i) > k_max(j+1,i) + min(k_err_bd(i,1),res_k_bd(i,1))
          bound_works(j,i)=i;
          which_ones(j,i)=ind_k_max(j,i);
          rank_xk(j,i)=j;
       else if k_max(j,i)==k_max(j+1,i)
          bound_works(j,i)=1000;
           end
       end
    end
       p=nnz(which_ones(:,i));
       applies(1:p,2*i)=nonzeros(which_ones(:,i));
       applies(1:p,2*i-1)=nonzeros(rank_xk(:,i));
end



if x0==v
   up_err_bd=alpha_k_1;
else up_err_bd=alpha_k;
end
```