Politecnico di Torino

Master's degree in Data Science and Engineering

Computational Linear Algebra

Page Rank

Lisa Vandi 360247
Riccardo Polazzi 361352

Accademic Year 2025/2026

Contents

1 PageRank Implementation **3**
1.1 Mathematical Formulation . . . 3
1.2 Problems with matrix A . . . 4
1.2.1 Disconnected components . . . 4
1.2.2 Dangling Nodes . . . 5
1.3 The Power Method and Optimization . . . 7
1.3.1 Convergence Analysis . . . 7
1.4 Computational Implementation . . . 9
1.4.1 Matrix construction strategy . . . 9
1.4.2 Implementation of the Compressed Sparse Row (CSR) Format . . . 10
1.4.3 Handling Dangling Nodes: The Algebraic Decomposition 11
1.4.4 Iterative Power Method . . . 12
1.5 Experimental Results . . . 12
1.5.1 Test Case 1: Connected Web . . . 12
1.5.2 Test Case 2: Disconnected Web . . . 13
1.5.3 Test Case 3: Hollins Dataset . . . 14

2 Exercises **16**
2.1 Exercise 1 . . . 16
2.1.1 Mathematical Analysis: Before and After Modification . . 16
2.1.2 Conclusion . . . 17
2.2 Exercise 4 . . . 18
2.2.1 Step 1: matrix formulation . . . 18
2.2.2 Step 2: finding λ . . . 19
2.2.3 Step 3: computing the eigenvector components . . . 20
2.2.4 Normalization and Result . . . 20
2.3 Exercise 5 . . . 21
2.4 Exercise 7 . . . 22
2.4.1 Non-negativity . . . 22
2.4.2 Sum of Columns . . . 22
2.5 Exercise 8 . . . 23
2.5.1 Non-negativity . . . 23
2.5.2 Column Sums . . . 24

2.5.3 Relevance to PageRank 24
2.6 Exercise 9 24
2.6.1 Step 1: apply the no backlinks condition 25
2.6.2 Step 2: simplify the teleportation term 25
2.7 Exercise 11: PageRank Calculation with Matrix M 26
2.7.1 Mathematical Resolution 26
2.7.2 3. Implementation and Results 26
2.7.3 The Random Surfer Model and Convergence 27
2.8 Exercise 12 28
2.8.1 Comparative Analysis 28
2.8.2 Theoretical Significance 29
2.9 Exercise 13 29
2.9.1 Objective and Scenario 29
2.9.2 Analysis of Matrix A: Non-Uniqueness 29
2.9.3 Analysis of Matrix M: uniqueness via positivity 30
2.9.4 Conclusion 31
2.10 Exercise 14 31
2.10.1 Objective of the Exercise 31
2.10.2 Scenario and Error Measurement 31
2.10.3 Analysis: The Second Eigenvalue and the Ergodicity Coefficient 31
2.10.4 Technical Significance: Scalability and the Damping Factor 32
2.10.5 Results 32
2.11 Exercise 15 32
2.11.1 Objective of the Exercise 32
2.11.2 Eigenvector Representation 33
2.11.3 Error decay: Dominance of $|\lambda_2|$ 33
2.11.4 Graphical Analysis 34
2.12 Exercise 17 35
2.12.1 Effect on Computation Time (Convergence Speed) 35
2.12.2 Effect on rankings (Quality and "Democracy") 36
2.12.3 Conclusion: the choice of $m = 0.15$ 36

Chapter 1

PageRank Implementation

1.1 Mathematical Formulation

The PageRank algorithm models the World Wide Web as a directed graph $G = (V, E)$, where V represents the set of n web pages and E represents the set of hyperlinks connecting them. The core intuition, derived from the *democracy of the web* concept, is that a page's importance is determined by the importance of the pages linking to it.
Let x_k denote the quantitative importance score of page k. A basic formulation suggests that a page distributes its importance evenly among its outgoing links. If page j has n_j outgoing links, the score transfer to page k is x_j/n_j. The importance score of any page k is defined recursively as the sum of the importance scores received from all pages linking to it (its *backlinks*). Let $L_k \subset \{1, \ldots, n\}$ denote the set of pages with a link to page k. The score x_k is given by:

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j} \tag{1.1}$$

This equation implies that the importance of a page is derived exclusively from the network structure and the importance of its neighbors.

To solve the ranking system efficiently for the entire web, we translate the recursive relationship between page scores into matrix notation. We define the *link matrix* A of size $n \times n$ as follows:

$$A_{ij} = \begin{cases} 1/n_j & \text{if page } j \text{ links to page } i \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

where n_j represents the *out-degree* (the number of outgoing links) of page j. Using this notation, the system of linear equations that defines the importance of each page can be expressed compactly as:

$$A\mathbf{x} = \mathbf{x} \tag{1.3}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the **importance vector** containing the scores of all pages.

From a linear algebra perspective, the equation $A\mathbf{x} = \mathbf{x}$ is a specific instance of the general **eigenvalue problem**:

$$A\mathbf{x} = \lambda\mathbf{x} \tag{1.4}$$

In our case, the PageRank solution corresponds to the scenario where the eigenvalue λ is exactly 1. Therefore, calculating the ranking of the web pages is equivalent to finding the **eigenvector x** associated with the eigenvalue $\lambda = 1$.

Assuming the web graph contains no *dangling nodes* (pages with no outgoing links, i.e., $n_j > 0$ for all j), every column of A sums to exactly 1. This characteristic defines A as a **column-stochastic matrix**. A fundamental property of such matrices (as stated in **Proposition 1** in the paper) ensures that $\lambda = 1$ is always an eigenvalue and that at least one corresponding eigenvector exists, thereby guaranteeing a solution to the ranking problem.

1.2 Problems with matrix A

The raw link matrix A poses two challenges that necessitate modifications to guarantee a valid ranking.

1.2.1 Disconnected components

The first critical challenge arises from the topological structure of the Web. The raw link matrix A fails to provide a valid ranking if the graph is not strongly connected.
Indeed, the fundamental requirement for a robust ranking algorithm is **uniqueness**.
Mathematically, the ranking vector $\mathbf{x}$ is unique (subject to the normalization $\sum x_i = 1$) if and only if the dimension of the eigenspace corresponding to the eigenvalue $\lambda = 1$ is exactly 1:

$$\dim(V_1(A)) = 1$$

However, the World Wide Web is not a single connected component. If the Web W consists of r disconnected sub-webs $W_1, \ldots, W_r$, then $\dim(V_1(A)) \geq r$.
Hence, there is no unique importance score vector $\mathbf{x} \in V_1(A)$ with $\sum x_i = 1$.
With $\dim(V_1(A)) > 1$, there are infinite valid ranking vectors, making it impossible to determine which one represents the "true" importance.

Solution to uniquess: the Google Matrix M
To force uniqueness, we must modify the system such that the graph becomes effectively connected. We introduce the **Google Matrix** M, defined as:

$$M = (1 - m)A + mS \tag{1.5}$$

where S is an $n \times n$ matrix with all entries equal to $1/n$, and m is the damping factor (typically 0.15). The term mS acts as a perturbation that equally distributes probability mass to every page in the network, regardless of the link structure.

Physically, this implements the "Random Surfer" model:

- With probability $(1 - m)$, the user follows the existing links (matrix A).
- With probability m, the user "teleports" to a random page chosen uniformly from the entire Web (matrix S).

This artificial teleportation creates a link between every pair of pages. Consequently, disconnected sub-webs cease to exist in the mathematical model; the graph becomes a single, strongly connected component where every node is reachable from every other node.

Mathematical proof of Uniqueness
From a linear algebra perspective, the introduction of the term mS (where $m > 0$) ensures that M is a **positive matrix** ($M_{ij} > 0$ for all i, j). The strict positivity of M, combined with the fact that M is column-stochastic (as it is a sum of column-stochastic matrices, as proved in Exercise 2.4), guarantees that $\dim(V_1(M)) = 1$, solving the uniqueness problem.
This conclusion is derived from the interaction of the following properties established in the paper:

1. **Proposition 2**: For a positive, column-stochastic matrix like M, any eigenvector in the subspace $V_1(M)$ must have components of the **same sign** (either all strictly positive or all strictly negative). Vectors with mixed signs are forbidden in this eigenspace.
2. **Proposition 3**: Given two linearly independent vectors, it is always possible to form a linear combination that results in a vector with mixed signs.

Based on these propositions, **Lemma 3.2** demonstrates that $V_1(M)$ cannot contain two linearly independent vectors. If it did, by Proposition 3, we could combine them to form a vector with mixed signs. Since $V_1(M)$ is a vector subspace, this new vector would necessarily belong to $V_1(M)$. However, this is impossible, as **Proposition 2** explicitly forbids vectors with mixed signs in that eigenspace.
Consequently, the dimension of $V_1(M)$ must be exactly 1. This guarantees the existence of a unique eigenvector with positive components summing to 1: the unique PageRank vector.

1.2.2 Dangling Nodes

The second major issue regarding matrix A concerns dangling nodes.
In the context of the PageRank algorithm, **dangling nodes** are defined as web

pages that possess no outgoing hyperlinks. In a directed graph representing the Web, these nodes act as dead ends for navigation.

The presence of dangling nodes profoundly alters the structure of the link matrix A: it makes the matrix **column-substochastic**.
Indeed, each dangling node produces a column containing exclusively zeros within the matrix A. This implies that the sum of the elements in certain columns is equal to zero, and therefore less than or equal to one.

The substochastic nature of the raw matrix A introduces several difficulties in the ranking calculation:

1. **Violation of Proposition 1 (loss of eigenvalue 1)**: while for a column-stochastic matrix it is guaranteed that 1 is an eigenvalue, for a substochastic matrix this is no longer guaranteed. Without the eigenvalue 1, it is not possible to identify an eigenvector x such that $Ax = x$ in a standard manner to assign importance scores.

2. **Persistence of substochasticity with M**: even if we attempt to apply the damping factor directly to the raw matrix A using the formula $M = (1 - m)A + mS$, the problem persists. For a dangling node (where the column of A is zero), the corresponding column in M sums to:

$$(1 - m) \cdot 0 + m \cdot 1 = m$$

   Since $m < 1$ (typically 0.15), the resulting matrix is not column-stochastic. Consequently, the standard theorems cannot be fully applied to guarantee the existence of a unique, valid ranking vector $\mathbf{x}$ normalized such that $\sum x_i = 1$.

Handling Dangling Nodes

To ensure the matrix A is perfectly column-stochastic, a mathematical transformation is applied to the columns corresponding to dangling nodes. Specifically, for every node j belonging to the set of dangling nodes, the null column is replaced with a uniform probability distribution:

$$\forall j \in \text{Dangling Set}, \quad A_{ij} = \frac{1}{N} \quad \forall i = 1 \dots N \tag{1.6}$$

This operation ensures that the total probability mass is preserved during iterations, effectively modeling the behavior of a random surfer who, upon reaching a node with no outgoing links, performs a "random jump" to any node in the network with equal probability.

Crucially, the addition of these virtual links to every node in the system **does not alter the ranking hierarchy**. Since the probability mass from a dangling node is distributed perfectly equitably among all N nodes, no individual node receives a competitive advantage or a relative increase in importance compared to others. This approach stabilizes the stochastic matrix and guarantees the existence of a unique stationary distribution while preserving the structural integrity and the relative importance defined by the original set of links.

1.3 The Power Method and Optimization

Given the immense size of the Web, finding the eigenvector with a direct method is computationally infeasible. We employ the **Power Method**, an iterative algorithm that converges to the dominant eigenvector, to solve the eigenvector problem $M\mathbf{x} = \mathbf{x}$.

The iterative step is defined as:

$$\mathbf{x}^{(k)} = M\mathbf{x}^{(k-1)} \tag{1.7}$$

Recursively applying this formula leads to the relation with the initial distribution vector $\mathbf{x}^{(0)}$:

$$\mathbf{x}^{(k)} = M^k\mathbf{x}^{(0)} \tag{1.8}$$

However, constructing the dense matrix M explicitly is computationally prohibitive ($O(n^2)$ memory). To solve this, we exploit the structure of M to compute the product implicitly using the sparsity of A.
The iteration is mathematically rewritten as:

$$\mathbf{x}^{(k)} = (1-m)A\mathbf{x}^{(k-1)} + m\mathbf{s} \tag{1.9}$$

where $\mathbf{s}$ is the vector with all entries $1/n$. This formulation reduces the complexity significantly, as A is sparse.

1.3.1 Convergence Analysis

The convergence of the iterative process $x_k = Mx_{k-1}$ to the steady-state eigenvector q is established through two key propositions.

Proposition 4: Contraction in the Zero-Sum Subspace

Proposition 4 focuses on the subspace $V \subset \mathbb{R}^n$ consisting of vectors v such that $\sum v_j = 0$. It states that for any $v \in V$:

$$||Mv||_1 \leq c||v||_1 \tag{1.10}$$

where $c = \max_{1\leq j\leq n} |1 - 2\min_{1\leq i\leq n} M_{ij}| < 1$. This proves that M acts as a **contraction mapping** on the subspace of vectors whose components sum to zero.

Proposition 5: Uniqueness and Global Convergence

Proposition 5 asserts that every positive column-stochastic matrix M possesses a **unique** vector q with positive components such that $Mq = q$ and $||q||_1 = 1$. The primary significance of this result is that the PageRank vector q can be computed as the limit of the power sequence for any initial probability vector x_0:

$$\lim_{k \to \infty} M^k x_0 = q \tag{1.11}$$

The connection to Proposition 4 is found in how the algorithm handles the **initial error vector** v. Any starting guess x_0 is essentially the sum of the true ranking and this error ($x_0 = q + v$). Since both x_0 and q are normalized to 1, the sum of the components of v must be zero, which identifies it as a member of the **subspace** V defined in Proposition 4.

Because v falls within this specific subspace, the contraction property established in Proposition 4 ensures that the error is diminished with each iteration until it vanishes. This explains why the choice of the initial vector x_0 does not affect the final result: the iterative process naturally "filters out" the error component to reveal the unique steady-state vector q.

Rate of Convergence

Proposition 4 establishes a constant $c = \max_{1 \leq j \leq n} |1 - 2 \min_{1 \leq i \leq n} M_{ij}|$ that represents the **maximum theoretical limit** for error reduction in each iteration. This constant serves as a worst-case bound, guaranteeing that the error norm decreases by at least a factor of c at each step. However, in practice, this bound is often considered "pessimistic".

The actual asymptotic behavior of the algorithm is governed by the spectral properties of the matrix M:

- **The Spectral Gap**: The power method converges asymptotically according to the second largest eigenvalue of M, denoted as λ_2, such that the error at each step follows the relation $||Mx_k - q||_1 \approx |\lambda_2| \cdot ||x_{k-1} - q||_1$.
- **Influence of the Damping Factor**: For Google's modified matrix $M = (1 - m)A + mS$, it can be shown that $|\lambda_2| \leq 1 - m$. In a typical web model, while the theoretical bound c might be as high as 0.94, the actual convergence rate dictated by $1 - m$ (using $m = 0.15$) is a significantly more efficient 0.85.
- **The m Trade-off**: The choice of the damping factor m represents a fundamental trade-off: a smaller m (closer to 0) preserves more of the original link structure of the web but increases $|\lambda_2|$, thereby slowing down the computation time.
- **Computational Efficiency**: With the standard implementation of $m = 0.15$, the error is reduced by approximately 15% in every iteration. This

rapid decay is what enables Google to process the world's largest matrix computation within a reasonable timeframe.

- **Convergence Criterion**: Iterations continue until the difference between successive rank vectors (the residual) falls below a specified tolerance τ:

$$||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}||_1 < \tau \tag{1.12}$$

1.4 Computational Implementation

The implementation of the PageRank algorithm was developed in Python using the `NumPy` and `SciPy` libraries. The code is designed to process the *Hollins* dataset ($N = 6012$ pages) and demonstrates the practical application of the Power Method.
The implementation strategy balances mathematical clarity with computational efficiency, addressing the construction of the link matrix and the resolution of dangling nodes.

1.4.1 Matrix construction strategy

The structure of the link matrix A is strictly dictated by the topology of the analyzed dataset. The *Hollins* dataset consists of $N = 6012$ nodes and 23875 edges. A defining characteristic of this network is the high prevalence of dangling nodes (3189 pages).

In line with the mathematical framework established in the previous sections, the presence of such dangling nodes creates a dual effect on the matrix A:

- **Theoretical effect (sub-stochasticity)**: since dangling nodes have no outgoing links, they generate columns containing exclusively zeros, making the matrix column-substochastic.

- **Computational effect (sparsity)**: the abundance of zero-columns significantly contributes to the sparsity of the matrix, reducing the number of non-zero entries to orders of magnitude less than N^2.

Our implementation strategy prioritizes the preservation of this sparsity to guarantee execution speed and memory efficiency.
While the theoretical remedy for dangling nodes would require physically filling these zero-columns with dense uniform vectors ($1/N$), doing so would catastrophically destroy the sparsity benefit. For this dataset, it would transform a lightweight structure into a dense matrix, making the computation inefficient.

Therefore, we chose not to store the $1/N$ probabilities in memory. Instead, we apply the dangling node patch virtually via algebraic decomposition during the calculation step.
To support this efficient approach, we chose to use the **Compressed Sparse Row (CSR)** format: it allows us to store the matrix in its raw, sub-stochastic

form, occupying memory only for the existing links ($O(E)$), while mathematically simulating a fully stochastic matrix through code logic.

1.4.2 Implementation of the Compressed Sparse Row (CSR) Format

To make PageRank calculations computationally feasible for large-scale graphs, we adopted the **Compressed Sparse Row (CSR)** format. This choice is necessitated by the extreme sparsity of web graphs: the adjacency matrix consists almost entirely of zeros, as each page typically links to only a tiny fraction of the total nodes in the network.

Correspondence Between Theory and Code

According to the formal definition, a sparse matrix $A \in \mathbb{R}^{m \times n}$ is encoded using three primary arrays:

- **`data` (or AA)**: Contains the real values $a_{i,j}$ of the non-zero entries, ordered by row. In our script, this array stores the transition probabilities calculated as $1/\text{out\_degree}(src)$.
- **`indices` (or JA)**: Stores the column indices for each corresponding value in the `data` array. In the context of PageRank, these represent the source nodes of the links.
- **`indptr` (or IA)**: An array of length $m + 1$ that recursively defines the starting and ending points for each row's data. Formally, $IA[i + 1] = IA[i] + \text{number of non-zero entries in row } i$.

Matrix Construction: From COO to CSR

Our implementation utilizes the `scipy.sparse` library. The process begins by gathering link data into the **Coordinate (COO)** format. The COO format is an intuitive storage scheme that represents a matrix as a collection of triplets: *(row_index, column_index, value).*

This format is ideal for the initial data-loading phase because it allows for the incremental building of the matrix as links are read line-by-line from the input file (e.g., *hollins.dat*). However, while COO is efficient for construction, it is not optimized for arithmetic operations. Therefore, once the lists of rows, columns, and values are populated, we convert the structure into a CSR matrix:

```
A_sparse = sparse.csr_matrix((data, (rows, cols)), shape=(N,
N))
```

This conversion transforms the unordered triplets into a compressed, row-aligned structure that is significantly faster for the matrix-vector products required by the Power Method.

Computational and Memory Efficiency

The adoption of the CSR format provides two critical advantages observed during testing on the *Hollins* dataset:

1. **Memory Footprint Reduction**: Storage requirements are reduced from $O(N^2)$ to $O(Nz + m)$, where Nz is the number of non-zero elements. For the *Hollins* dataset ($N = 6012$), a dense matrix would require over 36 million entries. By using CSR, we only store the $Nz = 23,875$ valid links, achieving a memory saving of over 99%.
2. **Matrix-Vector Multiplication Optimization**: The Power Method relies heavily on the repeated calculation of Ax. Since CSR stores non-zero elements in contiguous memory, it allows the algorithm to iterate *only* over existing links while completely ignoring zero entries. This reduces the computational complexity of each iteration from $O(N^2)$ to $O(Nz)$, enabling the algorithm to scale to much larger graphs.

1.4.3 Handling Dangling Nodes: The Algebraic Decomposition

To reconcile the memory benefits of the sparse matrix ($O(E)$ storage) with the theoretical requirement of a column-stochastic matrix, we employ an **algebraic decomposition strategy** during the iterative process.

In a standard dense implementation, stochasticity is enforced by modifying the matrix structure filling empty columns with $1/N$. In our sparse implementation, we leave the matrix A_{sparse} as column-substochastic (where dangling columns sum to zero).
Consequently, a standard multiplication $A_{sparse}\mathbf{x}$ would result in a leakage of probability mass: the importance associated with dangling nodes would simply disappear from the system.

To correct this loss without destroying sparsity, we compute the missing mass dynamically and reinject it into the system. If D is the set of indices corresponding to dangling nodes, the iterative step (Equation 1.9) is decomposed into three distinct components:

$$x_{new} = (1-m)Ax + (1-m)\left[\frac{\text{dangling\_mass}}{N}\right] + \frac{m}{N} \tag{1.13}$$

$$\mathbf{x}_{new} = \underbrace{(1-m)A_{sparse}\mathbf{x}}_{\text{Flow from Links}} + \underbrace{(1-m)\left(\sum_{j\in D} x_j\right)\frac{\mathbf{1}}{\mathbf{N}}}_{\text{Virtual Patch}} + \underbrace{\frac{m}{N}}_{\text{Teleportation}} \tag{1.14}$$

This decomposition mathematically guarantees that the operator remains effectively column-stochastic, despite the underlying matrix being sub-stochastic. The validity of this approach rests on the conservation of probability mass:

- **Loss detection**: the term $A_{sparse}\mathbf{x}$ propagates probability mass only from nodes with outgoing links. The mass residing in dangling nodes, defined as $\Omega = \sum_{j \in D} x_j$, is effectively lost in this step because the corresponding columns in A_{sparse} are zero.

- **Mass recovery (the Virtual Patch)**: the second term in Equation 1.14 captures this exact lost mass Ω and redistributes it according to the theoretical definition of dangling nodes (a uniform jump to any page).
  By adding the scalar value $(1-m)\Omega/N$ to every node, we mathematically simulate the presence of physical links from every dangling node to every other node, without allocating memory for them.

- **Proof of stochasticity**: the column-stochasticity is not violated because the total probability is conserved.
  Let Σ_{valid} be the total probability mass currently held by non-dangling nodes. The sum of the elements in the vector resulting from the first term is $(1-m)\Sigma_{valid}$. The sum of the elements from the virtual patch is $(1-m)\Omega$. Since $\Sigma_{valid} + \Omega = 1$ (the total probability of the previous iteration), the total sum of $\mathbf{x}_{new}$ becomes:

$$\sum_N \mathbf{x}_{new} = (1-m)(\Sigma_{valid} + \Omega) + \sum_N \frac{m}{N} = (1-m)(1) + m = 1$$

This proof confirms that our algebraic implementation strictly adheres to the theoretical requirements of the PageRank algorithm (matrix stochasticity) while maintaining the computational efficiency of sparse matrix operations.

1.4.4 Iterative Power Method

The core calculation is performed by the function `calculate_pagerank_sparse`. The algorithm does not construct the dense Google Matrix M explicitly. Instead, it iterates the decomposition formula described above until convergence.

1.5 Experimental Results

The implemented algorithm was tested on three distinct topologies. For all tests, we used a damping factor $m = 0.15$ and a tolerance $\tau = 10^{-7}$.

1.5.1 Test Case 1: Connected Web

The first test involved the 4-page connected graph described in Figure 1.1.

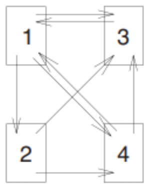


Figure 1.1: Connected graph topology (4 nodes)

- **Topology**: a strongly connected component where probability mass flows freely between all nodes.
- **Convergence:** the algorithm converged in **21** iterations.

Page ID	Score (x_i)	Rank
Page 1	0.3682	1
Page 3	0.2880	2
Page 4	0.2021	3
Page 2	0.1418	4

Table 1.1: PageRank Results for Figure 1.1

Although Page 3 has the highest number of backlinks (receiving votes from all other pages), Page 1 emerges as the most authoritative. This phenomenon is explained by the dynamics of score distribution:

Page 1 receives a decisive vote from Page 3 which, having only a single outgoing link directed specifically toward Page 1, transfers its **entire weight** to it without any dilution.

This confirms that the importance of a page depends not only on the number of votes received, but also on the authority of the voters and how they distribute their score.

1.5.2 Test Case 2: Disconnected Web

The second test used the 5-page graph from Figure 1.2 which consists of two disconnected subwebs ($W_1 = \{1, 2\}$ and $W_2 = \{3, 4, 5\}$).

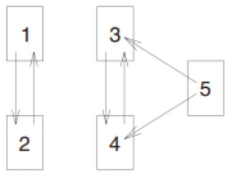


Figure 1.2: Disconnected graph topology (5 nodes)

- **Analysis:** with $m = 0$, the dimension of the eigenspace would be 2. The damping factor $m = 0.15$ successfully connects the components via teleportation.
- **Convergence:** the algorithm converged in **2** iterations.

Page ID	**Score** (x_i)	**Rank**
Page 3	0.2850	1 (Tie)
Page 4	0.2850	1 (Tie)
Page 1	0.2000	3 (Tie)
Page 2	0.2000	3 (Tie)
Page 5	0.0300	5

Table 1.2: PageRank Results for Figure 1.2

The results highlight two key behaviors of the algorithm.
First, the **perfect symmetry** within the sub-webs is reflected in the scores: Page 1 and Page 2 are topologically identical (mutual links), as are Page 3 and Page 4, resulting in tied rankings.
Second, Page 5 receives the lowest score (0.0300). This occurs because Page 5 acts as a pure ”source” node with outgoing links but no incoming links from other pages. Its score is derived exclusively from the **random teleportation component** (m/N), confirming that the algorithm correctly identifies and penalizes nodes that do not receive votes from the rest of the network.

1.5.3 Test Case 3: Hollins Dataset

We applied the algorithm to the `hollins.dat` dataset, representing a real-world subset of the web:

- **Nodes** (n)**:** 6012, of which 3189 are dangling nodes.
- **Edges:** 23875
- **Convergence:** completed in **71** iterations.

Rank	**Page ID**	**Score**
1	2	0.019879
2	37	0.009288
3	38	0.008610
4	61	0.008065
5	52	0.008027
6	43	0.007165
7	425	0.006583
8	27	0.005989
9	28	0.005572
10	4023	0.004452

Table 1.3: Top 10 Pages in Hollins Dataset

The application of the PageRank algorithm to the Hollins dataset reveals meaningful structural properties of the network:

1. **The homepage effect**: the page with ID **2** is identified as the most important node with a score of 0.019879, which is more than double the score of the second-ranked page. In the context of the Hollins University web domain, Page 2 corresponds to the homepage (`http://www.hollins.edu/`). It is expected to be the authority hub, receiving inbound links from navigation bars across most sub-pages in the domain.

2. **Minimum score verification**: we observed a minimum importance score of 5.8×10^{-5} in the computed vector. The theoretical lower bound for PageRank with $m = 0.15$ is the score a page would receive if it had absolutely no incoming links, receiving importance only through teleportation:

$$x_{min}^{theoretical} = \frac{m}{N} = \frac{0.15}{6012} \approx 2.5 \times 10^{-5}$$

   Since our observed minimum (5.8×10^{-5}) is strictly greater than the theoretical floor (2.5×10^{-5}), this confirms that the damping factor is correctly distributing probability mass to all nodes, and no node is left with zero importance.

3. **Convergence scale**: the algorithm required 71 iterations to converge. While higher than the simple test cases (21 iterations), this is consistent with the spectral gap theory for larger, more complex graphs. It demonstrates that the sparse CSR implementation scales effectively, handling the increased dimensionality without computational bottlenecks.

Chapter 2

Exercises

2.1 Exercise 1

We analyze a scenario of **Ranking Manipulation** where the owners of Page 3 attempt to artificially boost their ranking.
Recall that the importance score x_k is derived from the linear system $Ax = x$. We begin by establishing the baseline scores for the original 4-page topology shown in Figure 1.1.

2.1.1 Mathematical Analysis: Before and After Modification

Case 1: Initial 4-Page Web

In the original configuration, Page 3 links only to Page 1 ($n_3 = 1$), effectively transferring its entire voting weight to that node.

The resulting link matrix A is:

$$A = \begin{bmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix} \tag{2.1}$$

The importance score eigenvector, derived from the Paper, for this matrix is approximately:

$$x \approx [0.387, 0.129, 0.290, 0.194]^T$$

In this state, $x_1 > x_3$ ($0.387 > 0.290$). Even though Page 3 has more backlinks, Page 1 is more important because it receives the full weight of Page 3's vote.

Case 2: Modified 5-Page Web

To boost Page 3's score, owners create Page 5, which links to Page 3, and they modify Page 3 to link back to Page 5. This changes the number of outgoing links for Page 3 to $n_3 = 2$. The link structure changes as follows:

- **Page 3:** previously linked only to Page 1 ($n_3 = 1$). Now links to Page 1 and Page 5 ($n_3 = 2$). Consequently, the probability mass from Page 3 is split: $A_{1,3} = 0.5$ and $A_{5,3} = 0.5$.
- **Page 5:** links only to Page 3 ($n_5 = 1$). Thus, $A_{3,5} = 1.0$.

The modified link matrix $\tilde{A}$ becomes:

$$\tilde{A} = \begin{bmatrix} 0 & 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 & 1 \\ 1/3 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \end{bmatrix} \tag{2.2}$$

The score equations for the modified topology are:

- $x_1 = \frac{x_3}{2} + \frac{x_4}{2}$
- $x_2 = \frac{x_1}{3}$
- $x_3 = \frac{x_1}{3} + \frac{x_2}{2} + \frac{x_4}{2} + x_5 = \frac{x_1}{3} + \frac{x_1}{6} + \frac{x_1}{4} + \frac{x_3}{2}$
- $x_4 = \frac{x_1}{3} + \frac{x_2}{2} = \frac{x_1}{3} + \frac{x_1}{6} = \frac{x_1}{2}$
- $x_5 = \frac{x_3}{2}$

Substituting the values into the equation for x_1:

$$x_1 = \frac{x_3}{2} + \frac{1}{2}\left(\frac{x_1}{2}\right) \implies x_1 = \frac{x_3}{2} + \frac{x_1}{4} \implies \frac{3}{4}x_1 = \frac{x_3}{2} \implies \mathbf{x_3 = 1.5x_1} \tag{2.3}$$

2.1.2 Conclusion

From this exercise, it is clear that the PageRank system can be manipulated through strategic linking. By creating a circle of links between two pages, you create a "trap" that keeps the importance points circulating within a small group rather than letting them flow to the rest of the web.

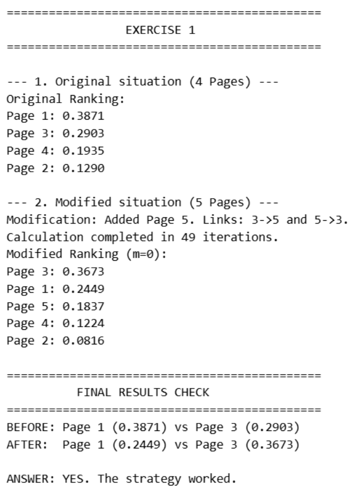
The theoretical derivation establishes a precise relationship: $x_3 = 1.5x_1$. These findings are corroborated by the computational implementation using the Power Method:

$$x_3 \approx 0.3673 \quad \text{and} \quad x_1 \approx 0.2449$$

Calculating the ratio from these numerical values confirms the algebraic solution:

$$\frac{x_3}{x_1} = \frac{0.3673}{0.2449} \approx 1.5$$

This demonstrates that the iterative algorithm converges to the exact theoretical distribution derived from the linear system.



```
=============================================
                 EXERCISE 1
=============================================

--- 1. Original situation (4 Pages) ---
Original Ranking:
Page 1: 0.3871
Page 3: 0.2903
Page 4: 0.1935
Page 2: 0.1290

--- 2. Modified situation (5 Pages) ---
Modification: Added Page 5. Links: 3->5 and 5->3.
Calculation completed in 49 iterations.
Modified Ranking (m=0):
Page 3: 0.3673
Page 1: 0.2449
Page 5: 0.1837
Page 4: 0.1224
Page 2: 0.0816

=============================================
          FINAL RESULTS CHECK
=============================================
BEFORE: Page 1 (0.3871) vs Page 3 (0.2903)
AFTER:  Page 1 (0.2449) vs Page 3 (0.3673)

ANSWER: YES. The strategy worked.
```

Figure 2.1: Computational results showing the modified PageRank scores. The ranking of Page 3 surpasses Page 1.

2.2 Exercise 4

Text: In the web of Figure 1.1, remove the link from page 3 to page 1. In the resulting web page 3 is now a dangling node. Set up the corresponding substochastic matrix and find its largest positive (Perron) eigenvalue. Find a non-negative Perron eigenvector for this eigenvalue, and scale the vector so that components sum to one. Does the resulting ranking seem reasonable?

Solution: To solve this problem, we proceed in three analytical steps.
Firstly, we modify the link matrix A to reflect the topological change.
Secondly, we formulate the eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$. By analyzing the dependencies between variables, we isolate the active subsystem to derive the condition for λ and compute the eigenvector.
Finally, we analyze the resulting probability distribution.

2.2.1 Step 1: matrix formulation

Removing the link $3 \to 1$ creates a dangling node at Page 3 ($n_3 = 0$). The third column of the adjacency matrix becomes entirely zero, making the matrix A

column-substochastic:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

We seek the largest positive eigenvalue λ and the corresponding eigenvector $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$ such that $A\mathbf{x} = \lambda\mathbf{x}$. Writing this out as a system of linear equations, we get:

$$\frac{1}{2}x_4 = \lambda x_1 \tag{2.4}$$

$$\frac{1}{3}x_1 = \lambda x_2 \tag{2.5}$$

$$\frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4 = \lambda x_3 \tag{2.6}$$

$$\frac{1}{3}x_1 + \frac{1}{2}x_2 = \lambda x_4 \tag{2.7}$$

We observe a critical structural property in the system above: the variable x_3 appears **only** in Equation 2.6, and nowhere else. This occurs because Page 3 is a dangling node: it receives probability mass (so x_3 depends on x_1, x_2, x_4) but does not redistribute it (so x_1, x_2, x_4 do not depend on x_3).

Consequently, equations 2.4, 2.5 and 2.7 form a self-contained **subsystem** governing the probability flow between pages 1, 2, and 4. We must solve this independent subsystem first to find the eigenvalue λ. Equation 2.6 will be used subsequently to determine x_3.

2.2.2 Step 2: finding λ

We solve the subsystem using the substitution method to express variables in terms of x_1:

1. From Equation 2.4, isolate x_4: $x_4 = 2\lambda x_1$.
2. From Equation 2.5, isolate x_2: $x_2 = \frac{1}{3\lambda}x_1$.

Substituting these into Equation 2.7:

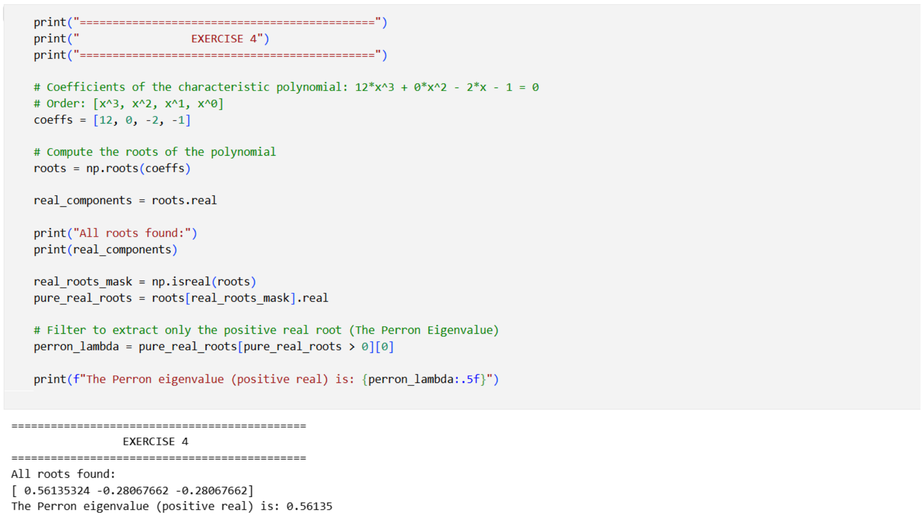
$$\frac{1}{3}x_1 + \frac{1}{2}\left(\frac{1}{3\lambda}x_1\right) = \lambda(2\lambda x_1)$$

Assuming a non-trivial solution ($x_1 \neq 0$), we divide by x_1 and simplify:

$$\frac{1}{3} + \frac{1}{6\lambda} = 2\lambda^2 \implies 12\lambda^3 - 2\lambda - 1 = 0$$

This cubic equation represents the condition for the existence of a non-trivial eigenvector for the subsystem. Solving numerically (see Figure 2.2), the unique positive root is:

$$\lambda \approx 0.56135$$



```python
print("=============================================")
print("                 EXERCISE 4")
print("=============================================")

# Coefficients of the characteristic polynomial: 12*x^3 + 0*x^2 - 2*x - 1 = 0
# Order: [x^3, x^2, x^1, x^0]
coeffs = [12, 0, -2, -1]

# Compute the roots of the polynomial
roots = np.roots(coeffs)

real_components = roots.real

print("All roots found:")
print(real_components)

real_roots_mask = np.isreal(roots)
pure_real_roots = roots[real_roots_mask].real

# Filter to extract only the positive real root (The Perron Eigenvalue)
perron_lambda = pure_real_roots[pure_real_roots > 0][0]

print(f"The Perron eigenvalue (positive real) is: {perron_lambda:.5f}")
```

```
=============================================
                 EXERCISE 4
=============================================
All roots found:
[ 0.56135324 -0.28067662 -0.28067662]
The Perron eigenvalue (positive real) is: 0.56135
```

Figure 2.2: Python calculation for Exercise 4 roots.

2.2.3 Step 3: computing the eigenvector components

The system $A\mathbf{x} = \lambda\mathbf{x}$ defines the *direction* of the eigenvector, but not its magnitude. Because of linearity, if $\mathbf{x}$ is a solution, any scalar multiple $k\mathbf{x}$ is also a solution. The system therefore has one degree of freedom. To fix a specific numerical solution, we arbitrarily set the scale by choosing $\mathbf{x_1 = 1}$.

Using $\lambda \approx 0.56135$ and $x_1 = 1$, we compute the remaining components:

- **Page 2:** $x_2 = \frac{1}{3\lambda} \approx 0.5938$
- **Page 4:** $x_4 = 2\lambda \approx 1.1227$
- **Page 3 (The Sink):** Now we use Equation 2.6:

$$\lambda x_3 = \frac{1}{3}(1) + \frac{1}{2}(0.5938) + \frac{1}{2}(1.1227) \implies x_3 \approx \frac{1.1916}{\lambda} \approx 2.1227$$

The unnormalized eigenvector is $\mathbf{v} \approx [1.0, 0.5938, 2.1227, 1.1227]^T$.

2.2.4 Normalization and Result

To obtain valid probability scores, we normalize the vector such that $\sum x_i = 1$. The sum of components is $S \approx 4.8392$. Dividing each component by S, we obtain the final ranking vector:

$$\mathbf{x} \approx \begin{bmatrix} 0.207 \\ 0.123 \\ 0.439 \\ 0.232 \end{bmatrix}$$

The resulting hierarchy is Page $3 > 4 > 1 > 2$.

- **Mathematically:** the ranking is reasonable.
  Page 3 acts as a "Rank Sink": it receives links from all other nodes but has no outgoing edges. Probability mass flows into it and accumulates before "leaking" out of the system (reflected by $\lambda < 1$).
- **Web Search Context:** the ranking is not reasonable.
  A page that provides no outgoing links acts as a dead end for the user. It should not be considered the most important page on the web solely because it traps users. This highlights the "Rank Sink" problem and justifies the introduction of the rank distribution patch ($1/N$) for dangling nodes used in the Google Matrix construction to fix this anomaly.

2.3 Exercise 5

Text: Prove that in any web the importance score of a page with no backlinks is zero.

Proof: Let A be the $n \times n$ link matrix of the web, where $A_{ij} \neq 0$ if and only if there is a link from page j to page i.
Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ be the eigenvector such that:

$$A\mathbf{x} = \mathbf{x}$$

The i-th component of this equation is given by:

$$x_i = \sum_{j=1}^{n} A_{ij} x_j$$

Assume page k has no backlinks (incoming links). This implies that no page j links to k. In terms of the matrix A, this means the k-th row contains only zeros:

$$A_{kj} = 0 \quad \forall j \in \{1, \ldots, n\}$$

Substituting this into the eigenvector equation for the index k:

$$x_k = \sum_{j=1}^{n} A_{kj} x_j = \sum_{j=1}^{n} 0 \cdot x_j = 0$$

Therefore, the importance score x_k must be zero.

Note: This limitation of the raw link matrix A is one of the reasons for introducing the Google Matrix $M = (1-m)A + mS$. In the modified model, every page receives a minimum probability mass from the teleportation matrix S, ensuring no page has a score of strictly zero.

2.4 Exercise 7

Text: Prove that if A is an $n \times n$ column-stochastic matrix and $0 \leq m \leq 1$, then $M = (1-m)A + mS$ is also a column-stochastic matrix.

Proof: A matrix is defined as *column-stochastic* if all its entries are non-negative and the sum of the entries in each column is equal to 1.

Let M be the matrix defined by the equation:

$$M = (1-m)A + mS$$

where S is the teleportation matrix with entries $S_{ij} = 1/n$ for all i, j.

2.4.1 Non-negativity

Since A is column-stochastic, $A_{ij} \geq 0$. The matrix S contains only positive entries ($1/n > 0$). Given that $0 \leq m \leq 1$, both coefficients $(1-m)$ and m are non-negative. Therefore, every entry $M_{ij} = (1-m)A_{ij} + mS_{ij}$ is a sum of non-negative terms, so $M_{ij} \geq 0$.

2.4.2 Sum of Columns

We need to prove that for any column j, the sum of the elements $\sum_{i=1}^{n} M_{ij}$ is exactly 1.

Let's expand the summation for the generic column j:

$$\sum_{i=1}^{n} M_{ij} = \sum_{i=1}^{n} ((1-m)A_{ij} + mS_{ij})$$

Using the linearity of the summation:

$$\sum_{i=1}^{n} M_{ij} = (1-m)\sum_{i=1}^{n} A_{ij} + m\sum_{i=1}^{n} S_{ij}$$

Now we evaluate the two summations separately:

- Since A is column-stochastic by hypothesis, the sum of its j-th column is 1:
$$\sum_{i=1}^{n} A_{ij} = 1$$
- Since S is the matrix where every entry is $1/n$:
$$\sum_{i=1}^{n} S_{ij} = \sum_{i=1}^{n} \frac{1}{n} = n \cdot \frac{1}{n} = 1$$

Substituting these values back into the equation:

$$\sum_{i=1}^{n} M_{ij} = (1-m)(1) + m(1)$$

$$\sum_{i=1}^{n} M_{ij} = 1 - m + m$$

$$\sum_{i=1}^{n} M_{ij} = 1$$

In conclusion, since M contains non-negative entries and the sum of each column is 1, M is a column-stochastic matrix.

2.5 Exercise 8

Text: Show that the product of two column-stochastic matrices is also column-stochastic.

Proof: Let A and B be two $n \times n$ column-stochastic matrices. By definition, a matrix is column-stochastic if:

1. All its entries are non-negative: $A_{ij} \geq 0$ and $B_{ij} \geq 0$ for all i, j.
2. The entries in each column sum to one: $\sum_{i=1}^{n} A_{ij} = 1$ and $\sum_{i=1}^{n} B_{ij} = 1$ for all j.

Let C be the product matrix $C = AB$. The entry C_{ij} is defined by the standard matrix multiplication formula:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

We must prove that C satisfies the two conditions of a column-stochastic matrix.

2.5.1 Non-negativity

Since every element A_{ik} and B_{kj} is non-negative, their product $A_{ik}B_{kj}$ is non-negative. The sum of non-negative terms is also non-negative. Thus, $C_{ij} \geq 0$ for all i, j.

2.5.2 Column Sums

We calculate the sum of the elements in the j-th column of C:

$$\sum_{i=1}^{n} C_{ij} = \sum_{i=1}^{n} \left(\sum_{k=1}^{n} A_{ik} B_{kj} \right)$$

We can swap the order of summation (finite sums):

$$\sum_{i=1}^{n} C_{ij} = \sum_{k=1}^{n} B_{kj} \left(\sum_{i=1}^{n} A_{ik} \right)$$

Since A is a column-stochastic matrix, the sum of its k-th column is 1 ($\sum_{i=1}^{n} A_{ik} = 1$). We substitute this into the equation:

$$\sum_{i=1}^{n} C_{ij} = \sum_{k=1}^{n} B_{kj}(1) = \sum_{k=1}^{n} B_{kj}$$

Since B is also a column-stochastic matrix, the sum of its j-th column is 1:

$$\sum_{k=1}^{n} B_{kj} = 1$$

Therefore:

$$\sum_{i=1}^{n} C_{ij} = 1$$

In conclusion: since $C = AB$ has non-negative entries and each column sums to 1, C is a column-stochastic matrix.

2.5.3 Relevance to PageRank

This result is crucial for the Power Method used in PageRank. The algorithm computes $x_k = Mx_{k-1} = M^k x_0$. Since M is column-stochastic, this exercise proves that any power M^k is also column-stochastic. This guarantees that the total probability of the system is conserved at every iteration step, ensuring that the surfer is always somewhere on the web and never disappears.

2.6 Exercise 9

Text: Show that a page with no backlinks is given importance score m/n by formula 1.9.

Proof: Let us recall the PageRank equation derived from the Google Matrix $M = (1 - m)A + mS$:

$$\mathbf{x} = (1 - m)A\mathbf{x} + mS\mathbf{x}$$

The component-wise form for the importance score of a specific page k, denoted as x_k, is:

$$x_k = (1-m)\sum_{j=1}^{n} A_{kj}x_j + m\sum_{j=1}^{n} S_{kj}x_j$$

where:

- n is the total number of pages in the web.
- A_{kj} is the entry of the link matrix (probability of moving from j to k).
- $S_{kj} = 1/n$ is the entry of the teleportation matrix.
- **x** is the probability vector, so $\sum_{j=1}^{n} x_j = 1$.

2.6.1 Step 1: apply the no backlinks condition

If page k has no backlinks, it means no page j links to k. In terms of the link matrix A, this implies that the k-th **row** consists entirely of zeros:

$$A_{kj} = 0 \quad \forall j \in \{1, \ldots, n\}$$

Substituting this into the first summation term:

$$\sum_{j=1}^{n} A_{kj}x_j = \sum_{j=1}^{n} 0 \cdot x_j = 0$$

2.6.2 Step 2: simplify the teleportation term

Now we analyze the second term involving matrix S. Since $S_{kj} = 1/n$ for all j:

$$m\sum_{j=1}^{n} S_{kj}x_j = m\sum_{j=1}^{n} \frac{1}{n}x_j$$

We can factor out the constant $1/n$:

$$= \frac{m}{n}\sum_{j=1}^{n} x_j$$

Since **x** is a normalized PageRank vector, the sum of all importance scores must be 1 ($\sum x_j = 1$). Therefore:

$$= \frac{m}{n} \cdot 1 = \frac{m}{n}$$

In conclusion, combining the two parts:

$$x_k = (1-m)(0) + \frac{m}{n}$$

$$x_k = \frac{m}{n}$$

Thus, a page with no backlinks receives exactly the minimum score $\frac{m}{n}$.

2.7 Exercise 11: PageRank Calculation with Matrix M

In this exercise, we recalculate the page ranking to the same modified graph from Exercise 1, which includes Page 5 and its reciprocal link cycle with Page 3 creating a loop, but this time using the Google Matrix M:

$$M = (1 - m)A + mS \tag{2.8}$$

2.7.1 Mathematical Resolution

1. Construction of the Modified Link Matrix A

The column-stochastic link matrix A is:

$$A = \begin{bmatrix} 0 & 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 & 1 \\ 1/3 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \end{bmatrix} \tag{2.9}$$

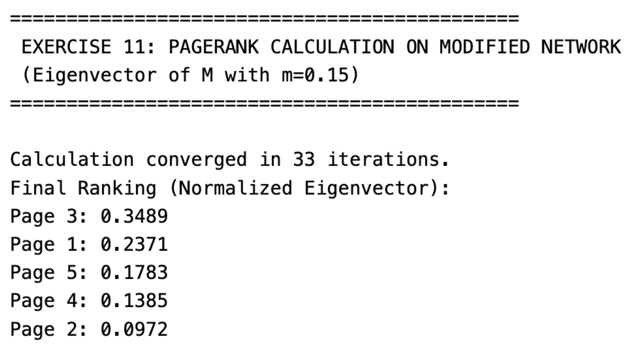
2. Calculation of the Google Matrix M

Using $n = 5$ and $m = 0.15$, the matrix S consists of entries $1/5 = 0.2$. The final matrix M used for the ranking is:

$$M = 0.85 \begin{bmatrix} 0 & 0 & 0.5 & 0.5 & 0 \\ 0.33 & 0 & 0 & 0 & 0 \\ 0.33 & 0.5 & 0 & 0.5 & 1 \\ 0.33 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \end{bmatrix} + 0.15 \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \tag{2.10}$$

2.7.2 3. Implementation and Results

The ranking is computed as the unique steady-state vector q such that $Mq = q$. This corresponds to the Random Surfer Model: the surfer follows a link with 85% probability or "teleports" to a random page with 15% probability[. Mathematically, q is efficiently found using the Power Method ($x_k = Mx_{k-1}$), which is guaranteed to converge regardless of the initial starting vector x_0.



```
============================================
 EXERCISE 11: PAGERANK CALCULATION ON MODIFIED NETWORK
 (Eigenvector of M with m=0.15)
============================================

Calculation converged in 33 iterations.
Final Ranking (Normalized Eigenvector):
Page 3: 0.3489
Page 1: 0.2371
Page 5: 0.1783
Page 4: 0.1385
Page 2: 0.0972
```

Figure 2.3: Results of Exercise 11

2.7.3 The Random Surfer Model and Convergence

By introducing the jump probability m, the algorithm ensures that the random surfer can eventually "break out" of any artificial trap. This makes the ranking more robust against strategic manipulation and ensures mathematical stability across the global web.

The comparison reveals the stabilizing effect of the Google Matrix:

1. **Mitigation of Manipulation:** In Exercise 1 ($m = 0$), the strategic link loop between Page 3 and Page 5 allowed Page 3 to reach a massive score of 0.3673. In Exercise 11 ($m = 0.15$), the introduction of teleportation introduces a probability "leak" in the loop. The random surfer has a 15% chance at each step to escape the trap and jump to other pages. Consequently, the score of Page 3 drops significantly (from 0.3673 to 0.3241).

2. **Global Redistribution and Mitigation of Strategic Inflation:** Although Page 3 remains the top-ranked node due to its structural prominence, the artificial boost provided by Page 5 is significantly dampened. This is particularly evident in Page 5's score, which sees a substantial drop; since its only source of importance is the reciprocal link from Page 3, it is heavily affected by the 15% "leakage" introduced by the damping factor. Instead of the authority being entirely trapped in the $3 \leftrightarrow 5$ cycle, a portion of it is now constantly redistributed to the rest of the network via the teleportation term mS, preventing artificial link structures from monopolizing the ranking.

3. **Conclusion:** In conclusion, Exercise 11 demonstrates that the Google Matrix M effectively mitigates strategic link manipulation. While the reciprocal structure between Page 3 and Page 5 still influences the ranking, the 15% damping factor prevents artificial score inflation by redistributing authority globally. This ensures a more balanced, robust, and mathematically stable ranking compared to the "pure democracy" model used in Exercise 1.

2.8 Exercise 12

The primary objective of this exercise is to demonstrate how the **Google Matrix** M resolves the issue of nodes with no backlinks. In the standard eigenvector formulation based on the raw link matrix A, such pages are assigned a null importance score, effectively making them invisible to the ranking system. This exercise illustrates how the damping factor introduces a strictly positive lower bound for importance scores.

The network topology from the previous exercise is expanded by introducing a sixth node (Page 6). This new page is characterized by the following connectivity properties:

- **Outgoing Links:** It links to every other page in the network.
- **Incoming Links:** It receives no links from any other page (out-degree > 0, in-degree $= 0$).

We compute and compare the rankings derived from the raw link matrix A ($m = 0$) and the Google Matrix M ($m = 0.15$).

2.8.1 Comparative Analysis

The analysis reveals a fundamental divergence between the purely topological model and the random surfer model:

- **Model with Link Matrix A ($m = 0$):**
  In the raw matrix formulation, the entry A_{ij} represents the probability of moving from j to i. Since no page j links to Page 6, the 6th row of the matrix A consists entirely of zeros:

  $$A_{6j} = 0 \quad \forall j \in \{1, \dots, n\}$$

  Consequently, the eigenvector equation component for this page becomes $x_6 = \sum A_{6j} x_j = 0$. The page is assigned an importance score of zero, implying it has no relevance within the network.

- **Model with Google Matrix M ($m = 0.15$):**
  Even though the contribution from the link structure $(1 - m)A$ remains zero for Page 6, the teleportation component mS ensures a non-zero probability flow.
  For any page i, the contribution from S is $\sum_j \frac{m}{N} x_j = \frac{m}{N}$. Therefore, for a 6-page web with $m = 0.15$, Page 6 receives exactly the minimum baseline score:

  $$x_6 = \frac{0.15}{6} = 0.025$$

2.8.2 Theoretical Significance

This result provides a practical demonstration of the spectral properties discussed in the theoretical section. Because the teleportation matrix S is strictly positive, the Google Matrix M satisfies the condition $M_{ij} > 0$ for all i, j. According to **Proposition 2** (Perron-Frobenius theorem for positive matrices), the dominant eigenvector of a positive matrix must have strictly positive components.

This guarantees two critical properties for the search algorithm:

1. **Global visibility:** no page, regardless of its isolation or lack of backlinks, is ever assigned a score of zero.

2. **Minimum importance floor:** the term m/N acts as a universal lower bound for PageRank scores, ensuring that every indexed page retains a minimal probability of being visited by the random surfer.

2.9 Exercise 13

2.9.1 Objective and Scenario

The objective of this exercise is to prove that the **Google Matrix** M enables a unique and globally comparable ranking even when the web structure consists of disconnected components. The analysis relies strictly on the spectral theory of positive matrices to show how the teleportation factor resolves the ambiguity inherent in disconnected graphs.

Consider a web W partitioned into two disjoint subwebs, W_1 and W_2, such that there are no links from W_1 to W_2 and vice versa. In graph theory terms, the graph is not **strongly connected**. This structure reflects the reality of the global web, where distinct clusters of websites may exist without direct reciprocal links.
By indexing the pages of W_1 first and those of W_2 second, the link matrix A assumes a **block-diagonal form**:

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \tag{2.11}$$

where A_1 and A_2 are the column-stochastic matrices representing the internal links of the two subwebs. The off-diagonal zero blocks mathematically represent the total isolation between the two communities.

2.9.2 Analysis of Matrix A: Non-Uniqueness

Since A_1 and A_2 are column-stochastic, they each possess an eigenvector corresponding to the eigenvalue $\lambda = 1$. Let $\mathbf{v}_1$ be the normalized eigenvector for W_1 and $\mathbf{v}_2$ for W_2.

We can construct two linearly independent eigenvectors for the global matrix A by padding with zeros:

$$\mathbf{x}_A = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{x}_B = \begin{bmatrix} \mathbf{0} \\ \mathbf{v}_2 \end{bmatrix}$$

It is easily verified that $A\mathbf{x}_A = \mathbf{x}_A$ and $A\mathbf{x}_B = \mathbf{x}_B$. Since the eigenspace $V_1(A)$ is a vector space, any linear combination of these basis vectors is a valid solution:

$$\mathbf{x} = \alpha\mathbf{x}_A + \beta\mathbf{x}_B \tag{2.12}$$

For $\mathbf{x}$ to be a valid PageRank vector, it must represent a probability distribution, meaning the sum of its components must equal 1. Let's compute the sum of the components of the combined vector:

$$\sum_i x_i = \alpha \sum_i (\mathbf{x}_A)_i + \beta \sum_i (\mathbf{x}_B)_i$$

Since $\mathbf{v}_1$ and $\mathbf{v}_2$ are normalized, the sums of $\mathbf{x}_A$ and $\mathbf{x}_B$ are both 1. Therefore:

$$\sum_i x_i = \alpha(1) + \beta(1) = \alpha + \beta$$

To satisfy the probability requirement $\sum x_i = 1$, we must enforce the constraint $\alpha + \beta = 1$.

However, even with this constraint, the ranking depends entirely on the arbitrary choice of coefficients α and β. It is mathematically impossible to determine objectively whether a page in W_1 is more important than a page in W_2 based only on the links.

2.9.3 Analysis of Matrix M: uniqueness via positivity

To resolve this ambiguity, the algorithm introduces the Google Matrix $M = (1 - m)A + mS$.
Physically, the teleportation matrix S (with entries $1/n$) acts as a perturbation that fills the zero blocks of A, modeling a random surfer who can jump between isolated subwebs.
Since the teleportation matrix S has strictly positive entries ($1/n$) and $m > 0$, the matrix M becomes a **strictly positive matrix** ($M_{ij} > 0$ for all i, j). According to **Proposition 2** and **Lemma 3.2**, a positive column-stochastic matrix possesses a unique eigenvector with eigenvalue $\lambda = 1$ and strictly positive components.
Thus, the introduction of the teleportation factor mS forces the existence of a unique ranking vector $\mathbf{x}$. This allows the algorithm to objectively compare the importance of pages in W_1 against pages in W_2, merging them into a single global scale.

2.9.4 Conclusion

This exercise proves that the damping factor m is not merely a numerical trick for convergence, but a structural necessity for defining a global ranking.
Without teleportation, the relative importance between disconnected communities (W_1 versus W_2) is mathematically undefined, leaving the ranking subject to arbitrary initialization parameters (α, β).
By ensuring the matrix is strictly positive, the Google formulation eliminates these degrees of freedom. It establishes an objective exchange rate of probability between isolated subwebs, transforming a set of independent local rankings into a single, cohesive global hierarchy.

2.10 Exercise 14

2.10.1 Objective of the Exercise

The primary objective of this exercise is to analyze the computational efficiency of the PageRank algorithm by monitoring the convergence of the **Power Method**. It demonstrates how the error between the iterative vector x_k and the steady-state vector q decays over time, and how this decay rate is strictly governed by the spectral properties of the Google Matrix M.

2.10.2 Scenario and Error Measurement

Starting with an initial probability vector x_0, the Power Method generates a sequence of approximations through the iteration $x_k = Mx_{k-1}$. The convergence is quantified using the L_1 **norm** of the error:

$$e_k = \|x_k - q\|_1 \tag{2.13}$$

By computing the ratio of successive errors, $\frac{e_k}{e_{k-1}}$, for $k = 1, 5, 10, 50$, the exercise shows that the sequence stabilizes. This ratio provides the asymptotic rate of convergence, indicating how much "error" is eliminated in each step of the algorithm.

2.10.3 Analysis: The Second Eigenvalue and the Ergodicity Coefficient

The exercise verifies two critical theoretical bounds discussed in the paper:

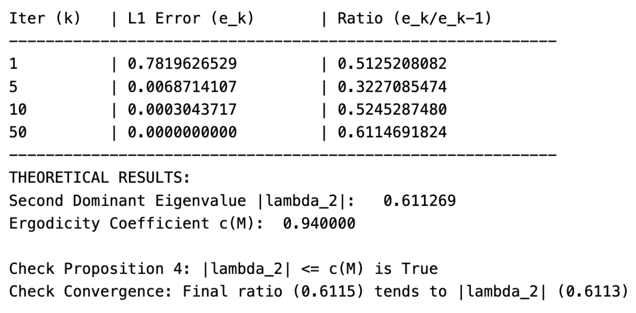
- **The Spectral Gap:** The asymptotic convergence rate is determined by the absolute value of the second largest eigenvalue, $|\lambda_2|$. For the matrix M under examination, the ratio $\frac{e_k}{e_{k-1}}$ stabilizes around a specific value (e.g., ≈ 0.61 or ≈ 0.85 depending on the specific link structure), confirming that $|\lambda_2|$ dictates the rhythm of the algorithm.

- **The Ergodicity Coefficient** $c(M)$**:** The exercise requires calculating $c(M) = \max_j |1 - 2\min_i M_{ij}|$. This coefficient provides a guaranteed upper bound for the convergence rate, ensuring that $|\lambda_2| \leq c(M)$. By showing that the actual convergence ratio is less than or equal to $c(M)$, the exercise experimentally validates the matrix M as a **contraction mapping** in the space of probability vectors.

2.10.4 Technical Significance: Scalability and the Damping Factor

This analysis provides the mathematical justification for why Google can process billions of pages efficiently. The convergence speed is intrinsically linked to the **damping factor** m. Because $|\lambda_2| \leq 1 - m$, setting $m = 0.15$ guarantees that $|\lambda_2| \leq 0.85$, ensuring that the error decreases by at least 15% at every iteration. This explains why the system reaches a stable state in relatively few iterations, making PageRank computationally tractable for the global scale of the World Wide Web.

2.10.5 Results



```
Iter (k)   | L1 Error (e_k)       | Ratio (e_k/e_k-1)
------------------------------------------------------------
1          | 0.7819626529         | 0.5125208082
5          | 0.0068714107         | 0.3227085474
10         | 0.0003043717         | 0.5245287480
50         | 0.0000000000         | 0.6114691824
------------------------------------------------------------
THEORETICAL RESULTS:
Second Dominant Eigenvalue |lambda_2|:   0.611269
Ergodicity Coefficient c(M):  0.940000

Check Proposition 4: |lambda_2| <= c(M) is True
Check Convergence: Final ratio (0.6115) tends to |lambda_2| (0.6113)
```

Figure 2.4: Caption

2.11 Exercise 15

2.11.1 Objective of the Exercise

The primary objective of this exercise is to provide a rigorous mathematical foundation for the convergence behavior observed in the Power Method. By applying **spectral decomposition**, we analyze how any initial starting vector eventually converges to the unique PageRank vector q, and demonstrate why the speed of this convergence is governed by the second largest eigenvalue, $|\lambda_2|$.

2.11.2 Eigenvector Representation

To analyze the convergence of the Power Method, we consider a positive column-stochastic matrix M that is **diagonalizable**. This assumption is crucial because it ensures that the matrix possesses a complete set of n linearly independent eigenvectors, which together form a **basis** for $\mathbb{R}^n$[cite: 319].

By having a basis of eigenvectors $\{q, v_1, v_2, \ldots, v_{n-1}\}$, any initial probability vector x_0 can be uniquely expressed as a linear combination:

$$x_0 = aq + b_1 v_1 + b_2 v_2 + \cdots + b_{n-1} v_{n-1} \tag{2.14}$$

In this decomposition, each term has a specific physical and mathematical meaning:

- **The Solution (q):** The vector q is the steady-state vector (PageRank) corresponding to the dominant eigenvalue $\lambda_1 = 1$. Since both x_0 and q are normalized probability vectors, the coefficient a must be equal to 1.
- **The Spectral Errors (v_k):** The other eigenvectors $\{v_1, \ldots, v_{n-1}\}$ represent the "noise" or the initial error of our guess x_0[cite: 319]. The sum of the components of each v_k must equal 0, meaning they represent shifts in probability mass that do not affect the total sum.

When we apply the matrix M iteratively for k steps using the Power Method, we obtain:

$$\begin{aligned}
&\text{iteration 1:} && Mv = \lambda v \\
&\text{iteration 2:} && M(Mv) = M(\lambda v) = \lambda(Mv) = \lambda(\lambda v) = \lambda^2 v \\
& && \vdots \\
&\text{iteration k:} && M^k v = \lambda^k v
\end{aligned}$$

$$x_k = M^k x_0 = q + c_2 \lambda_2^k v_2 + c_3 \lambda_3^k v_3 + \cdots + c_n \lambda_n^k v_n \tag{2.15}$$

The error at step k, defined as the deviation from the steady state $e_k = x_k - q$, becomes:

$$e_k = c_2 \lambda_2^k v_2 + c_3 \lambda_3^k v_3 + \cdots + c_n \lambda_n^k v_n \tag{2.16}$$

2.11.3 Error decay: Dominance of $|\lambda_2|$

Since the eigenvalues of the Google matrix are ordered by magnitude ($1 > |\lambda_2| \geq |\lambda_3| \geq \ldots$), the error components associated with smaller eigenvalues ($\lambda_3, \lambda_4, \ldots$) decay exponentially faster than the term associated with λ_2.

This disparity in decay rates creates two distinct phases in the convergence process:

- **Elimination of Spectral Noise:** Components associated with smaller eigenvalues (like $\lambda_3, \lambda_4, \ldots$) represent "noise" that vanishes almost immediately. Using our calculated values where $|\lambda_2| \approx 0.61$ and assuming a smaller eigenvalue $|\lambda_3| \approx 0.30$, we can compare their magnitudes after 10 iterations:

- The λ_3 component shrinks to: $0.30^{10} \approx 0.000005$
- The λ_2 component remains at: $0.61^{10} \approx 0.007$

The "noise" from λ_3 is already nearly 1,500 times smaller than the error from λ_2.

- **Asymptotic Dominance:** After very few iterations, all components except the one associated with the second largest eigenvalue have effectively reached zero. Consequently, the total error e_k begins to behave exactly like the term

$$e_k \approx c_2 \lambda_2^k v_2 \tag{2.17}$$

This explains why the ratio of successive errors monitored in Exercise 14 converges to $|\lambda_2|$:

$$\frac{\|e_k\|}{\|e_{k-1}\|} \approx \frac{\|c_2 \lambda_2^k v_2\|}{\|c_2 \lambda_2^{k-1} v_2\|} = |\lambda_2| \tag{2.18}$$

2.11.4 Graphical Analysis

The figure below (Exercise 15 Plot) illustrates the convergence on a semi-logarithmic scale.

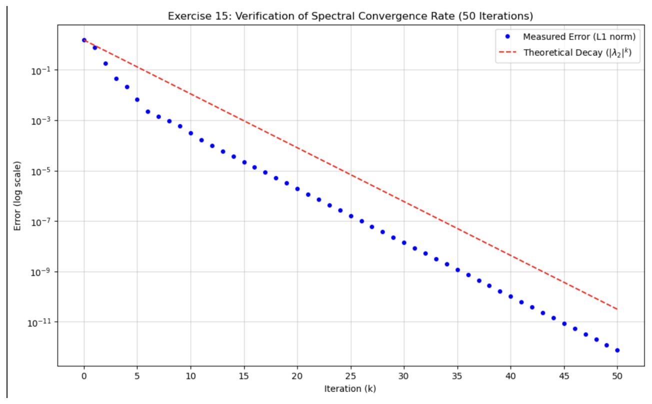


Figure 2.5: Caption

As illustrated in the Plot, the Measured Error (blue dots) initially decays faster than the theoretical $|\lambda_2|^k$ line (red dashed line). This behavior is mathematically consistent with the spectral decomposition theory:

- **Initial Transient Phase:** In the first 5–10 iterations, the blue dots drop sharply. This is due to the contribution of the smaller eigenvalues $(\lambda_3, \lambda_4, \ldots)$, which represent the "fast-decaying" components of the error.

Since $|\lambda_i| \ll |\lambda_2|$, these components vanish almost immediately, causing a more rapid initial decrease in the total error than predicted by $|\lambda_2|$ alone.

- **Asymptotic Parallelism:** After this transient phase, the blue dots settle into a linear trend that is perfectly **parallel** to the red dashed line. In a semi-logarithmic plot, parallelism indicates that the two series share the same constant ratio. This proves that the asymptotic convergence ratio:

$$\frac{\|e_k\|}{\|e_{k-1}\|} \approx |\lambda_2| \tag{2.19}$$

  is indeed governed by the second largest eigenvalue, confirming the **spectral gap theory**.

The fact that the measured error sits below the theoretical line is a positive result: it demonstrates that the real-world convergence of the Power Method is even more efficient than the "worst-case" theoretical limit dictated by $|\lambda_2|$, although its final speed (the slope) is strictly determined by it.

2.12 Exercise 17

The choice of m represents a fundamental trade-off in the design of the PageRank algorithm. The Google Matrix is defined as a weighted average:

$$M = (1-m)A + mS \tag{2.20}$$

The parameter m balances two opposing forces: the "democratic" nature of the web structure (represented by A) against the mathematical stability and convergence speed provided by the teleportation matrix (S).

2.12.1 Effect on Computation Time (Convergence Speed)

As established in the analysis of the Power Method, the convergence rate is strictly governed by the magnitude of the **second largest eigenvalue** $|\lambda_2|$. The error at step k decays according to the ratio $|\lambda_2|^k$.

For the Google Matrix M, the paper implies a direct relationship between the damping factor and the spectral gap:

$$|\lambda_2| \approx 1 - m \tag{2.21}$$

This relationship dictates the computational cost:

- **Small m (e.g., $m = 0.01$):** the second eigenvalue is large ($|\lambda_2| \approx 0.99$). The error decays very slowly (only 1% per iteration). This would require thousands of iterations to reach convergence, which is computationally prohibitive for a web-scale graph.
- **Large m (e.g., $m = 0.50$):** the second eigenvalue is small ($|\lambda_2| \approx 0.50$). The error is halved at every step, leading to extremely fast convergence.

2.12.2 Effect on rankings (Quality and "Democracy")

If speed were the only factor, we would choose a large m. However, increasing m degrades the quality of the information. The term $(1-m)$ represents the **weight of the true web topology**: it reflects the principle of the web where hyperlinks act as votes conferred by human authors.

- **High fidelity ($m \to 0$):** the ranking is determined almost exclusively by actual links. However, without the damping factor, the system suffers from critical topological issues: **Rank Sinks** trap probability mass, **disconnected components** lead to non-unique rankings, and pages with **no backlinks** receive a score of zero (invisibility).
- **High noise ($m \to 1$):** as m increases, the matrix is increasingly dominated by S, which distributes probability uniformly $(1/n)$.
  If we chose a high value like $m = 0.50$, the algorithm would derive 50% of a page's score from random teleportation rather than its actual backlinks. This would flatten the ranking vector, reducing the distinction between authoritative pages and obscure ones, effectively destroying the relevance of the search results.

2.12.3 Conclusion: the choice of $m = 0.15$

The standard value of $m = 0.15$ is the industry-standard compromise that optimally resolves this trade-off. It is often cited as a "threshold" value because it maintains a specific physical property of navigation, derived from the statistical behavior of the random surfer.

1. **The Path Length threshold:** the expected number of steps (clicks) a random surfer performs before teleporting is governed by the probability of continuing the chain, which is $(1-m)$.
   Mathematically, the number of steps k until the first teleportation follows a geometric distribution with success probability $p = m$. The expected value (mean) of this distribution is calculated as the sum of the probabilities of the surfer surviving at least k steps. This forms a geometric series:
   $$E[\text{steps}] = \sum_{k=0}^{\infty} P(\text{surviving } k \text{ steps}) = \sum_{k=0}^{\infty} (1-m)^k$$
   This summation corresponds to the famous **geometric series formula** $\sum_{k=0}^{\infty} r^k$. For $|r| < 1$, it converges as:
   $$\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$$
   So, since $0 < 1-m < 1$, substituting our ratio $r = 1-m$ into the formula, we obtain the exact relationship:
   $$E[\text{steps}] = \frac{1}{1-(1-m)} = \frac{1}{m}$$

This derivation proves why the choice of m is critical for the depth of exploration:

- With $m = 0.15$, the average path is $1/0.15 \approx \mathbf{6.66}$ clicks. This depth allows the algorithm to propagate voting authority through roughly 6 layers of the network, capturing the deep structure of the web.
- If we increased m to just 0.30, the average path would drop to $1/0.30 \approx \mathbf{3.33}$ clicks. This represents a "short-sighted" exploration: the ranking would become too local, failing to distinguish global authorities that are reachable only via longer paths.

2. **Structural Integrity:** by setting $1 - m = 0.85$, the algorithm preserves the vast majority of the true link structure. The "noise" introduced (15%) is sufficient to ensure strong connectivity (resolving disconnected components) and guarantee a minimum score for every page (fixing the zero-score problem for pages with no backlinks), without overpowering the human-generated votes.

3. **Acceptable Speed ($|\lambda_2| \approx 0.85$):** while not instant, a decay factor of 0.85 ensures that the error is reduced by roughly 15% at each iteration. This allows the algorithm to converge to a high degree of precision in a reasonable number of steps, making the computation feasible even for billions of pages.