

Power Method

Kovács Renáta, Corduneanu-Huci Maria, Dzhekshenova Aiana

[GitHub Repository](#)

December 2025

Contents

1	Basic Idea	2
2	Finding Eigenvalues and Eigenvectors	2
3	Dominant Eigenvalue	2
4	Power Method for the Dominant Eigenvalue	3
4.1	Definition	3
4.2	Proof	3
4.3	Rayleigh quotient	4
4.4	Overflow, underflow	4
4.5	Normalization	4
4.6	Scaling	5
4.7	Stopping Criteria	5
4.8	Convergence	5
5	MATLAB	5
6	Real-Life Implementations of the Power Method	5
7	Computing PageRank	6
7.1	Method 1: Eigendecomposition	6
7.2	Method 2: Power Method	6
8	Conclusion	7
	References	7

Abstract

The power method is a simple yet highly effective iterative algorithm for approximating the dominant eigenvalue and corresponding to the eigenvalue of largest modulus. Beginning with a random initial vector, the method repeatedly applies the matrix and normalizes the result, producing progressively better approximations. Despite being nearly a century old, the power method remains central in modern numerical algorithms. Its enduring relevance stems from two key strengths: exceptional computational efficiency and remarkable robustness. These properties make the power method a practical and reliable tool for computing the dominant eigenvalue in large-scale problems.

1 Basic Idea

The eigenvalues of an $n \times n$ matrix A are obtained by solving its characteristic equation

$$\lambda^n + c_{n-1}\lambda^{n-1} + c_{n-2}\lambda^{n-2} + \cdots + c_0 = 0.$$

However, for large values of n , solving the characteristic polynomial becomes difficult, and numerical methods for approximating the roots of high-degree polynomials are highly sensitive to rounding errors. Because of these limitations, computing all eigenvalues directly is often impractical.

An alternative approach is to approximate the **dominant eigenvalue** of A - the eigenvalue with the largest absolute value, using iterative methods such as the power method. To understand this idea, we first recall how eigenvalues and eigenvectors are found.

2 Finding Eigenvalues and Eigenvectors

For a square matrix A , an eigenvalue - eigenvector pair (λ, v) satisfies:

$$Av = \lambda v.$$

This equation has nontrivial solutions only when λ satisfies a special condition, which leads to the characteristic polynomial.

Step 1: Form the characteristic polynomial

$$p_A(\lambda) = \det(A - \lambda I).$$

Step 2: Solve the characteristic equation

$$p_A(\lambda) = 0,$$

which gives all eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ of the matrix.

Step 3: Find the eigenvectors

For each eigenvalue λ , find the corresponding set of eigenvectors by solving

$$Av = \lambda v \quad \text{for } v.$$

3 Dominant Eigenvalue

Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the eigenvalues of an $n \times n$ matrix A . We say that λ_1 is the **dominant eigenvalue** of A if

$$|\lambda_1| > |\lambda_i|, \quad i = 2, \dots, n.$$

The eigenvectors corresponding to λ_1 are called *dominant eigenvectors* of A . The dominant eigenvalue governs the long-term behavior of iterative processes and is the one approximated by the power method.

Remark

Not every matrix has a dominant eigenvalue. This occurs when two or more eigenvalues have the same largest absolute value.

4 Power Method for the Dominant Eigenvalue

Finding the eigenvalues of a matrix $A \in \mathbb{R}^{n \times n}$ becomes increasingly difficult for $n \geq 3$, and for $n \geq 5$ there is no general formula for the roots of the characteristic polynomial. To solve this problem, we can use a numerical method and, alternatively, there are algorithms to find the approximations of one or more eigenvalues. The power Method is an iterative technique that helps us find the largest eigenvalue of a matrix. The power method for approximating eigenvalues is iterative (similar to the Jacobi and Gauss-Seidel methods).

The power method is an iterative algorithm that is used to approximate the dominant eigenvalue λ_1 of a matrix A , as well as its associated eigenvector e_1 .

We assume that A is a real matrix $n \times n$ with distinct real eigenvalues:

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$$

Eigenvectors associated with the eigenvalues: e_1, e_2, \dots, e_n .

A has n distinct eigenvalues \Rightarrow the eigenvectors e_1, e_2, \dots, e_n are linearly independent (in total n linearly independent eigenvectors)

4.1 Definition

If the distinct eigenvalues of a matrix A are $\lambda_1, \lambda_2, \dots, \lambda_n$ and $|\lambda_1|$ is greater than $|\lambda_2|, |\lambda_3|, \dots, |\lambda_n|$, then $|\lambda_1|$ is called a dominant eigenvalue of A and its corresponding eigenvectors are called dominant eigenvectors of A .

4.2 Proof

The Power Method is based on the dynamical system x_0, Ax_0, A^2x_0, \dots

Any nonzero column vector $x_0 \neq 0$ can be written as a linear combination of n eigenvectors:

$$x_0 = c_1 e_1 + c_2 e_2 + \dots + c_n e_n$$

where

- c_n is a real constant ($c_n \neq 0$)
- e_n is the eigenvector

$$x_0 = \sum_{i=1}^n c_i e_i$$

Multiply by matrix A :

$$x_1 = Ax_0 = A \sum_{i=1}^n c_i e_i = \sum_{i=1}^n c_i A e_i$$

Since e_i is an eigenvector of A with eigenvalue λ_i :

$$x_1 = \sum_{i=1}^n c_i \lambda_i e_i$$

Multiply by matrix A p times:

$$x_p = A^p x_0 = \sum_{i=1}^n c_i \lambda_i^p e_i$$

We know that λ_1 is the dominant eigenvalue ($|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$) \Rightarrow we can factor out the first term (dominant term):

$$x_p = c_1 \lambda_1^p e_1 + \sum_{i=2}^n c_i \lambda_i^p e_i = \lambda_1^p \left[c_1 e_1 + \sum_{i=2}^n c_i \left(\frac{\lambda_i}{\lambda_1} \right)^p e_i \right].$$

Factoring out λ_1^p we can see that the terms in the sum go to zero relative to $c_1 e_1$.
 $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n| \Rightarrow |\lambda_i/\lambda_1| < 1$ for all $i \geq 2 \Rightarrow$

$$\left(\frac{\lambda_i}{\lambda_1}\right)^p \rightarrow 0 \quad \text{as } p \rightarrow \infty.$$

After sufficiently many iterations, the terms involving the non-dominant eigenvalues become negligible and the vector is well approximated by its dominant term.

$$x_p \approx c_1 \lambda_1^p e_1$$

To estimate the dominant eigenvalue we need x_{p+1} :

$$x_{p+1} \approx \lambda_1 x_p,$$

4.3 Rayleigh quotient

The Rayleigh quotient provides an estimate of λ_1 :

$$\begin{aligned} \lambda_1 &\approx \frac{\mathbf{x}_p^T A \mathbf{x}_p}{\mathbf{x}_p^T \mathbf{x}_p} \\ &\Longleftrightarrow \\ \lambda_1 &\approx \frac{x_p^T x_{p+1}}{x_p^T x_p}. \end{aligned}$$

We can construct this estimate, because we know that x_p is an eigenvector, therefore we can use the definition of the eigenvector ($Av = \lambda v$). We can substitute A with λ and get that the quotient is equal to λ . The Rayleigh quotient can only be used to estimate the largest eigenvalue if the power method converges to an eigenvector.

4.4 Overflow, underflow

After repeated multiplications by A , the vector x_p becomes dominated by the component associated with the eigenvalue of largest magnitude.

$x_{p+1} = Ax_p \approx \lambda_1 x_p \Rightarrow$ the Rayleigh quotient or component ratio converges to λ_1 .

Because x_p becomes proportional to the dominant eigenvector e_1 , we can take x_p itself as an approximation of e_1 (eigenvectors are defined only up to multiplication by a constant \rightarrow in the power method, the vector x_p only approximates e_1 up to a constant factor, and its direction is correct, but its length can vary depending on normalization).

The direct implementation is that we keep multiplying by λ_1 , but:

- if $|\lambda_1| > 1$, $\|x_p\|$ grows and may overflow (a value larger than the largest value that we can represent on the computer)
- if $0 < |\lambda_1| < 1$, $\|x_p\|$ shrinks and may underflow (a value smaller than what we can represent with the computer)

4.5 Normalization

To prevent Overflow or Underflow with each iteration, there are multiple ways to normalize each step. We show two methods. The first one is normalization.

$$x_{p+1} = \frac{Ax_p}{\|Ax_p\|_2},$$

Where $\|\cdot\|_2$ is the Euclidean norm.

Normalization preserves direction, so convergence to e_1 still holds.

4.6 Scaling

The second one is scaling.

$$\mathbf{x}_{p+1} = \frac{A\mathbf{x}_p}{\max_i |(A\mathbf{x}_p)_i|}.$$

$$\|x_{p+1}\|_\infty = \max_i |(Ax_p)_i|$$

$$x_{p+1} = \frac{Ax_p}{\|Ax_p\|_\infty}$$

This technique multiplies the result vector by the reciprocal of the largest absolute value. This ensures that all elements of e_1 are between zero and one. With every iteration the largest absolute values converge to the dominant eigenvalue. The method converges to the actual eigenvector if matrix A is diagonalizable.

4.7 Stopping Criteria

The iteration in the power method is stopped when the approximate eigenvector or eigenvalue has sufficiently converged.

1. **Vector convergence:** stop when the change in the eigenvector between iterations is small:

$$\frac{\|x_{p+1} - x_p\|}{\|x_{p+1}\|} < \varepsilon,$$

where ε is the desired tolerance (e.g.: 10^{-6}).

2. **Eigenvalue convergence:** stop when the estimated dominant eigenvalue stabilizes:

$$\frac{|\lambda^{(p+1)} - \lambda^{(p)}|}{|\lambda^{(p+1)}|} < \varepsilon.$$

3. **Maximum iterations:** to prevent infinite loops, stop if the number of iterations exceeds maximum number of iterations you allow the algorithm to perform.

Iterations are usually stopped when either the vector or eigenvalue convergence condition is met, or when the maximum number of iterations is reached.

4.8 Convergence

The convergence of the power method can be shown using the same equations as before. The rate of convergence depends on λ_2 and λ_1 . The rate of convergence is the quotient of the two values. If the quotient is small the power method converges fast. If it is large the power method converges slowly.

5 MATLAB

In MATLAB, $e = \text{eig}(A)$ returns a column vector containing the eigenvalues of square matrix A $[V, D] = \text{eig}(A)$ returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors

6 Real-Life Implementations of the Power Method

The power method is widely used in systems that need to identify the most influential elements within very large networks. It appears in recommendation engines, social network analysis, and scientific ranking problems because it requires only repeated matrix-vector multiplications, making it efficient even for extremely large and sparse matrices.

One of the most well-known real-world implementations of the power method is **Google's PageRank algorithm**. PageRank models a “random surfer” who moves between webpages by following hyperlinks

with probability 0.85 and randomly jumping to any page with the remaining 0.15. This behavior forms a Markov chain whose long-term stationary distribution assigns an importance score to each page. The web’s massive link structure is represented as a sparse transition matrix, and Google computes PageRank by repeatedly multiplying this matrix by a probability vector until convergence. Because the power method does not require storing the full matrix and naturally exploits sparsity, it scales effectively to billions of webpages.

Beyond search engines, variants of the power method also appear in modern recommendation systems, such as those used by YouTube, TikTok, or Amazon. In these systems, user–item interactions are modeled as large graphs, and algorithms like Personalized PageRank are applied to identify the most relevant or influential items for a specific user. By estimating the dominant eigenvector of an interaction graph, such systems can rank content, suggest new items, and detect influential users or products efficiently at large scale.

7 Computing PageRank

7.1 Method 1: Eigendecomposition

- Compute the dominant eigenvector of the Google matrix.
- **Complexity:** $O(n^3)$
- **Pros/Cons:** Exact solution, but too slow for large web-scale graphs.

7.2 Method 2: Power Method

Constructing the Google Matrix

Let A be the adjacency matrix of a graph, where $A_{ij} = 1$ if node j links to node i , and 0 otherwise.

Column-normalized transition matrix

We first construct a column-stochastic matrix M by normalizing each column of A so that it sums to 1, representing the transition probabilities of a random surfer:

$$M_{ij} = \frac{A_{ij}}{\sum_k A_{kj}} \quad \text{if } \sum_k A_{kj} \neq 0$$

Google matrix with damping

To fix problems such as dangling nodes and rank sinks, Brin and Page introduced damping. The Google matrix \widetilde{M} is defined as:

$$\widetilde{M} = (1 - d)M + d\frac{1}{n}J$$

- $d = 0.15$: probability of “teleporting” to a random page
- J : $n \times n$ all-ones matrix
- \widetilde{M} : ensures the matrix is stochastic, irreducible, and aperiodic, so the PageRank vector is unique and the power method converges

- Start with a uniform vector v_0 and iteratively compute:

$$v_{k+1} = \widetilde{M}v_k$$

until convergence

- **Complexity:** $O(n^2)$ per iteration; for sparse matrices, $O(n)$
- **Pros:** Fast, scalable, and very used in practice
- **Convergence:** Depends on the second-largest eigenvalue $|\lambda_2|$. Smaller $|\lambda_2| \Rightarrow$ faster convergence

8 Conclusion

The power method provides a simple and efficient way to approximate the dominant eigenvalue and eigenvector of a matrix. Its convergence relies on spectral dominance, and with proper normalization it remains stable and effective even for large problems. Because of its scalability, the method continues to play an important role in modern applications such as PageRank and large-scale network analysis.

References

- [1] GeeksforGeeks, *Power Method – Determine Largest Eigenvalue and Eigenvector in Python*, 2025. Available online: <https://www.geeksforgeeks.org/python/power-method-determine-largest-eigenvalue-and-eigenvector-in-python/>
- [2] TU Delft – Linear Algebra Interactive Textbook, 9.3. *The power method*. Available online: <https://interactivetextbooks.tudelft.nl/linear-algebra/Chapter9/PowerMethod.html>
- [3] Indrumar Metode Numerice, *Laborator – Power Method / Metode Numerice*. Available online: <https://ro.scribd.com/document/287208811/Indrumar-Metode-Numerice-Laborator#page=55>
- [4] Hippocampus-Garden, *PageRank*. Available online: <https://hippocampus-garden.com/pagerank/>
- [5] MathWorks, *eig – MATLAB Documentation*. Available online: <https://www.mathworks.com/help/matlab/ref/eig.html>
- [6] GeeksforGeeks, *Page Rank Algorithm Implementation in Python*. Available online: <https://www.geeksforgeeks.org/python/page-rank-algorithm-implementation/>
- [7] CPhys Lecture Notes, *Power Method (Fortran Version)*. Available online: https://ergodic.ugr.es/cphys/lecciones/fortran/power_method.pdf
- [8] Cours 4 Part 2, *Linear Algebra — Lecture Notes*. Available online: <https://math.univ-cotedazur.fr/~frapetti/CorsoF/cours4part2.pdf>
- [9] Wikipedia, *PageRank*, Available online: <https://en.wikipedia.org/wiki/PageRank>
- [10] MathWorks, *PageRank Example (PDF)*, Available online: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/exm/chapters/pagerank.pdf>