Fast Linear System Solve via Subspace Iteration

Ibrohim Nosirov
Advisors: Chris Musco, Jonathan Weare
AMSURE 2023

Linear System

A linear system is a problem that can be written down as

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

where $A \in \mathbb{R}^{m \times n}$.

In this talk, we are interested in the case where A is **symmetric positive definite** (eigenvalues are all greater than zero).

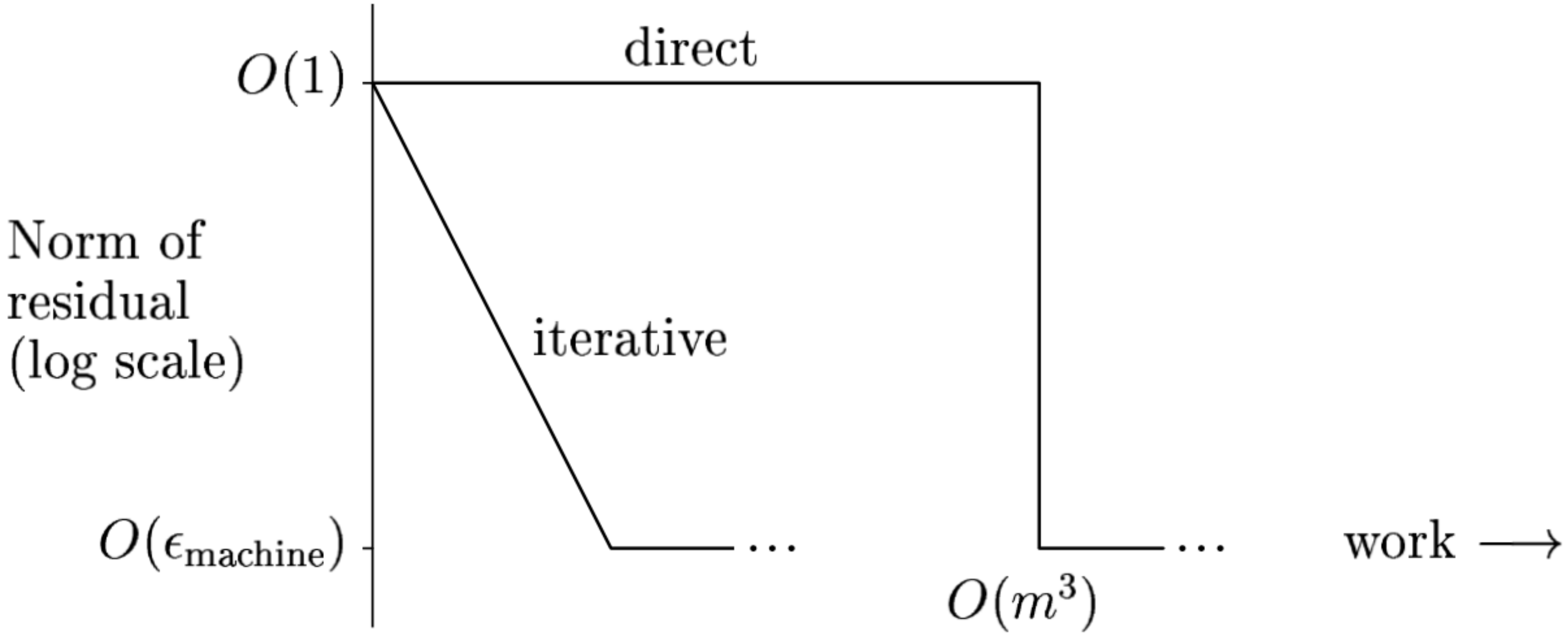
More specifically, we are interested in minimizing the residual,

$$\arg\min_{\mathbf{x}} ||A\mathbf{x} - \mathbf{b}||_2^2.$$

Iterative vs Direct Methods

In general, solving a linear system is an order $\mathcal{O}(n^2)$ time operation, with a worst case complexity of $\mathcal{O}(n^3)$.

For large scale problems, iterative algorithms are preferred to direct methods, like LU, due to issues like computer **memory requirements**.



Credit: *Numerical Linear Algebra,* Trefethen & Bau '97

Our Scheme

- We propose an iterative algorithm that **scales well for large matrices**.
- This algorithm uses **randomness** to speed up convergence.
- The convergence rate of this algorithm will depend on the **gaps between eigenvalues** in the spectrum of A.
- Due to time, exciting geometric intuition (e.g. our method's robustness to defective matrices compared to Krylov subspaces) is omitted.

Our Scheme

The algorithm we wish to speed up is called **Gradient Descent**.

GD iteratively minimizes a function, f, where at each step we compute,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \varepsilon \nabla f(\mathbf{x}_k).$$

In our case, this amounts to

$$\mathbf{x}_1 = \mathbf{x}_0 + \varepsilon \left(\mathbf{b} - A\mathbf{x}_0 \right).$$

Our Scheme

We construct a matrix

$$\bar{A} = \begin{pmatrix} 1 & \mathbf{0} \\ \varepsilon\mathbf{b} & I - \varepsilon A \end{pmatrix},$$

where $\varepsilon \leq \dfrac{1}{\lambda_{max}}$.

When applied to some **non-zero (usually random) vector** $\bar{\mathbf{x}}_0 = \begin{pmatrix} 1 \\ \mathbf{x}_0 \end{pmatrix}$, we get

$$\bar{A}\bar{\mathbf{x}}_0 = \begin{pmatrix} 1 \\ \varepsilon\mathbf{b} + \mathbf{x}_0 - \varepsilon A\mathbf{x}_0 \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{x}_0 + \varepsilon\left(\mathbf{b} - A\mathbf{x}_0\right) \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{x}_1 \end{pmatrix},$$

a **gradient descent iteration** in the lower block of the resulting vector.

Power Method

Repeatedly applying a matrix to a vector converges to the dominant eigenpair of that matrix.

This process is called Power Method.

In our case, **we are performing power method on $I - \varepsilon A$**.

The eigenvalues are arranged as $\left\{1 - \lambda_n, 1 - \lambda_{n-1}, \ldots, 1 - \lambda_1\right\}$ where λ_i is the i-th eigenvalue of A.

Subspace Iteration

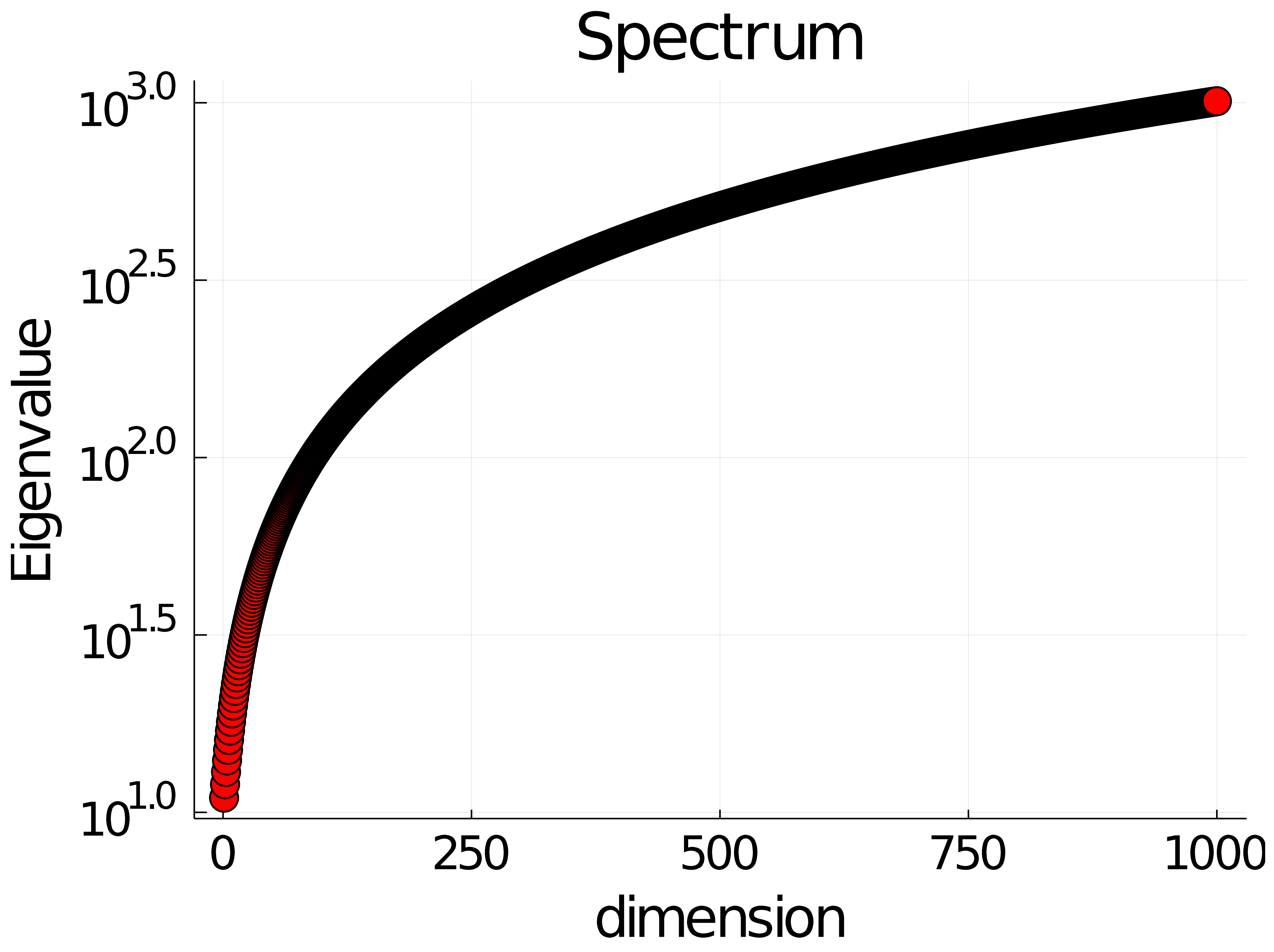
Power Method on $I - \varepsilon A$ has convergence rate that depends on $\dfrac{1-\lambda_{n-1}}{1-\lambda_n}$, where a **small ratio implies fast convergence**.

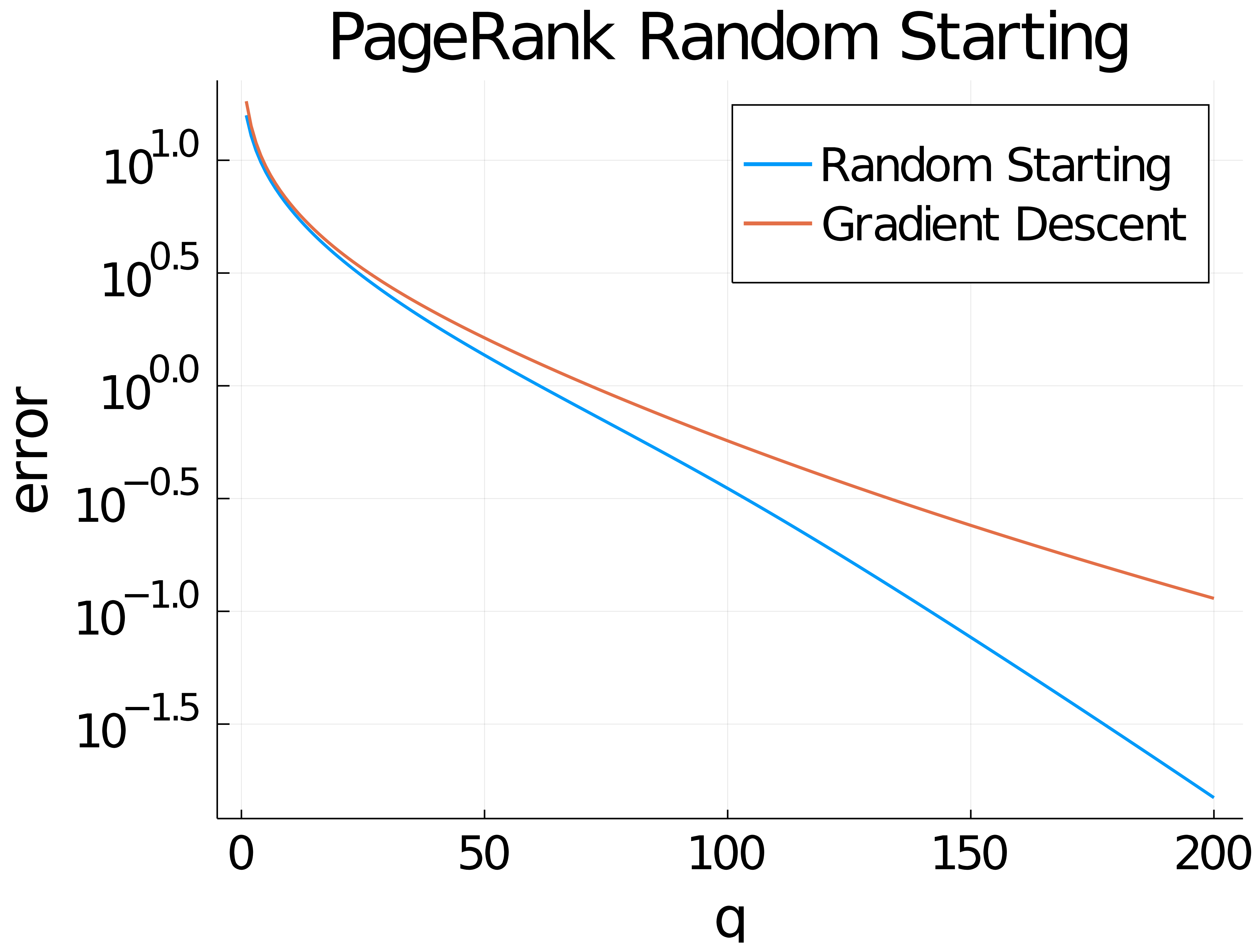
Subspace Iteration is an extension on Power Method:

Replace the starting vector $\begin{pmatrix} 1 \\ \mathbf{x}_0 \end{pmatrix}$ with a starting matrix $\bar{\Pi} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Pi \end{pmatrix}$ where $\Pi \in \mathbb{R}^{n \times k}$ is a random matrix with $n \gg k$.

Subspace iteration convergence depends on $\dfrac{1-\lambda_{n-k}}{1-\lambda_n}$.

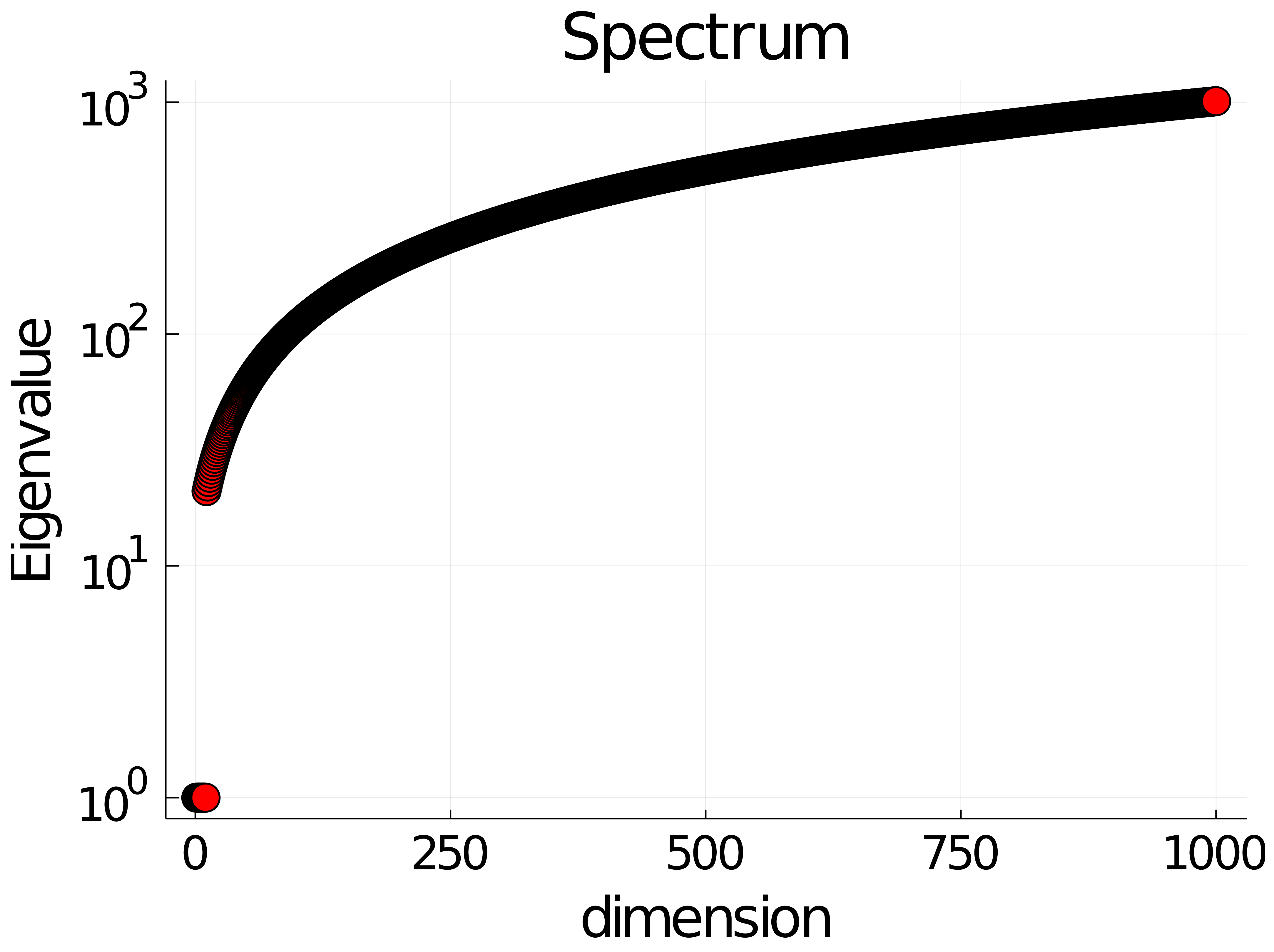
Eigenspectrum v. Convergence

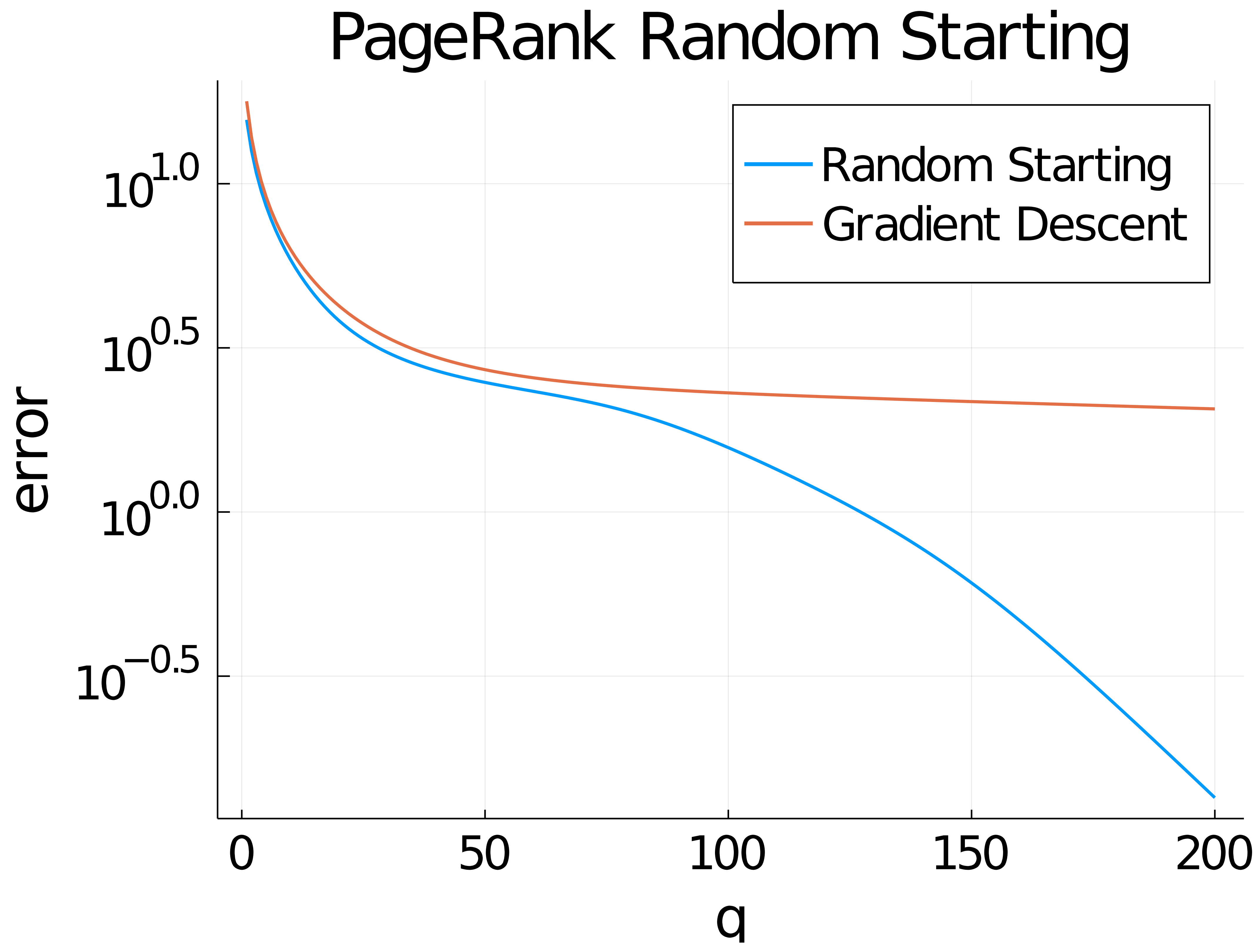




Setting the size of the subspace k=20

Eigenspectrum v. Convergence





Setting the size of the subspace k=20

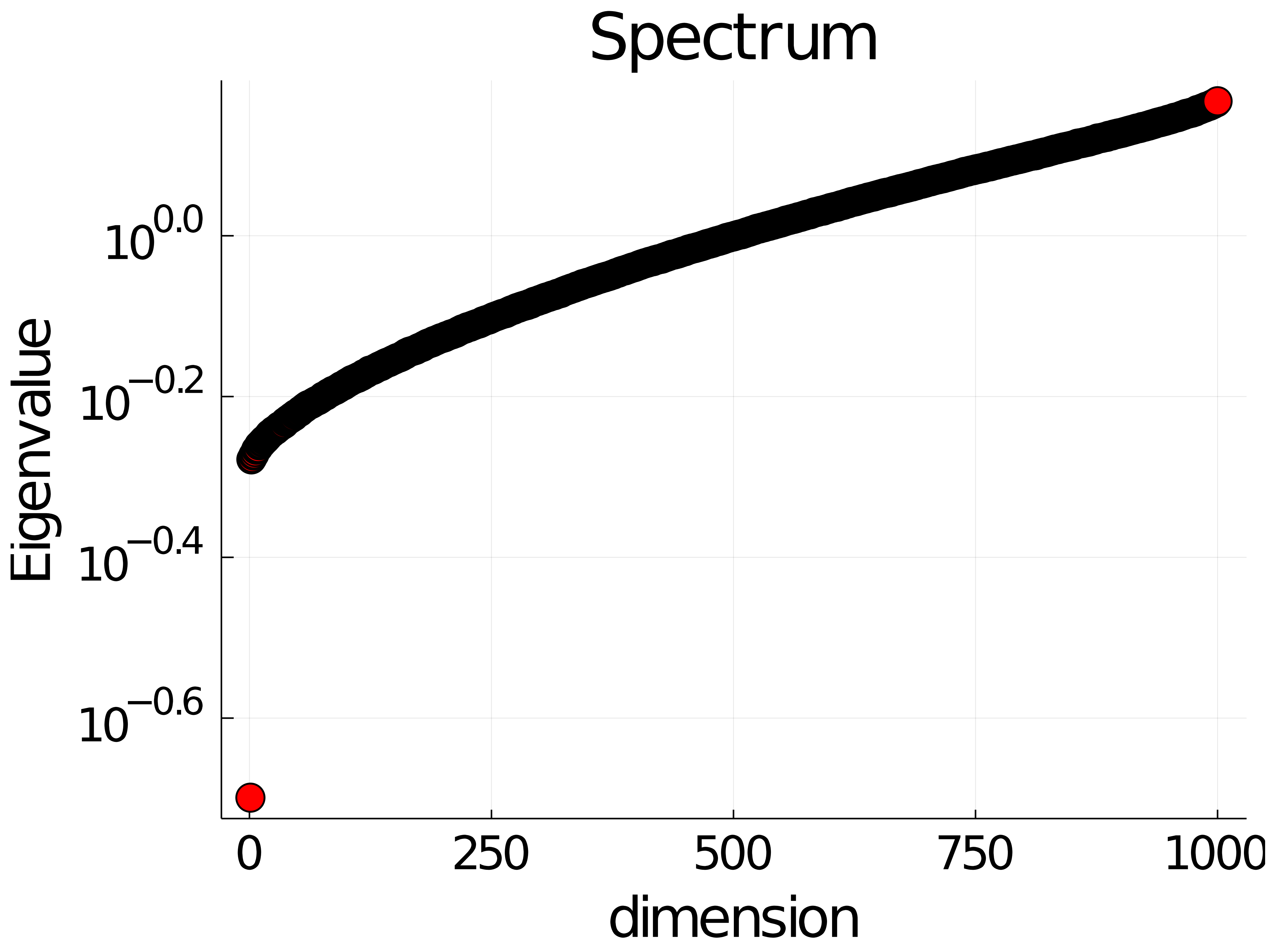
Application: PageRank

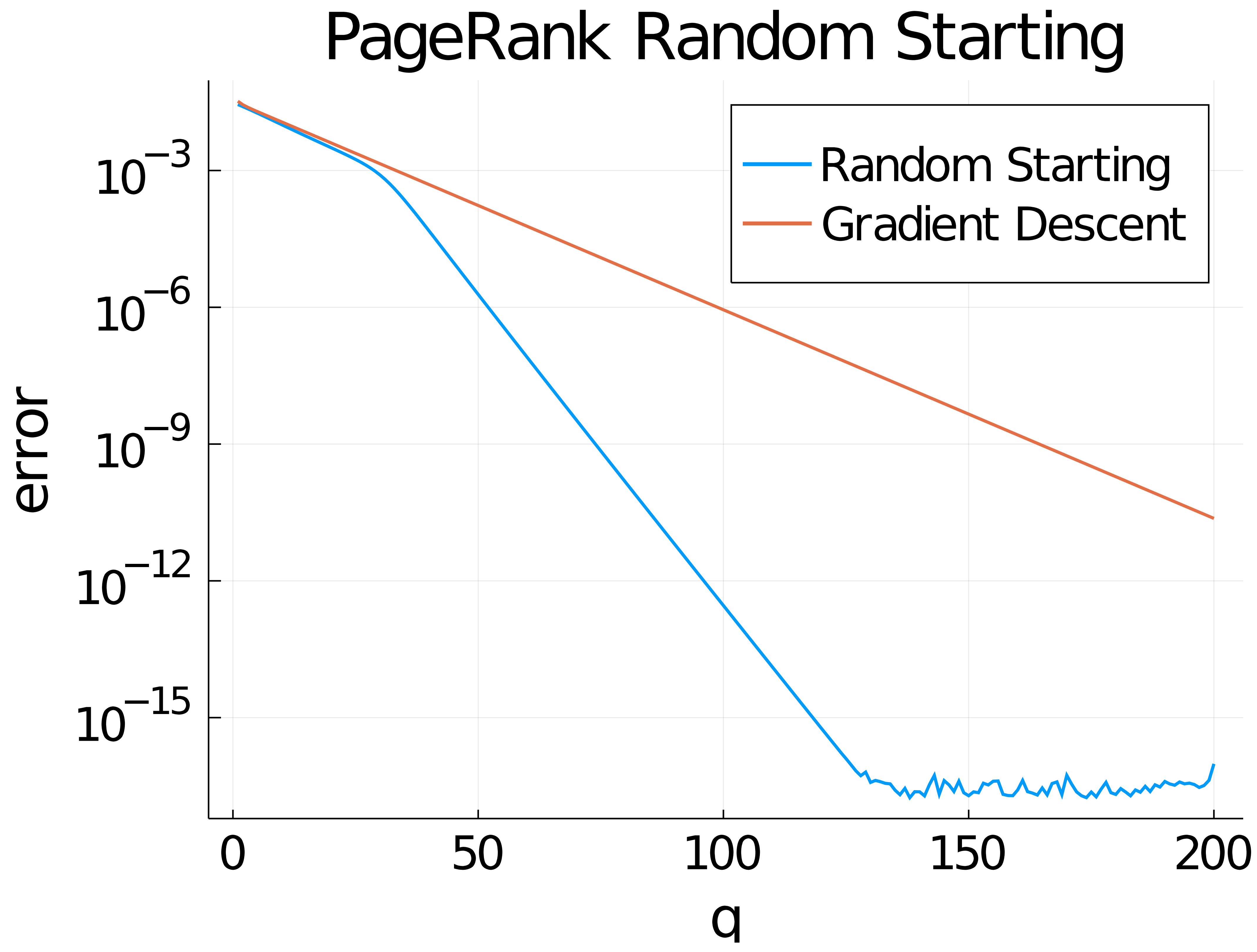
The PageRank problem shows up in many contexts.

The solution to this problem outputs a ranking of nodes (hyperlinks) in a graph based on an internet surfer's likelihood of going to each link.

For our purposes, it suffices to say that $A = I - \omega P$ where P **is a stochastic matrix** (columns are probability vectors) and ω **is a constant.**

PageRank Random Starting Result





Setting subspace size k=3

Conclusion

We have an algorithm that iteratively converges to a solution to a linear system on the order of $\dfrac{(1-\lambda_{n-k})}{(1-\lambda_n)}$.

This scheme works particularly well when there is a **cluster of small eigenvalues and a cluster of large eigenvalues.**

This is often true for stochastic matrices that show up in problems like PageRank.