Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: October 26, 2025

Assignment No: Assignment 5 - Question 1
Course Code: DS284
Course Name: Numerical Linear Algebra
Term: AUG 2025

Preparation and Algorithm

The first step in analysing the graph is to obtain the Markov transition graph. The Markov transition matrix for the graph is as follows:

$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We need to run the following power iteration algorithm on the above Markov transition matrix:

Initialise a random vector $\tilde{\mathbf{v}}^{(0)}$ such that $\|\tilde{\mathbf{v}}^{(0)}\|_2 = 1$
for $k = 1, 2, \ldots$ do
    $\tilde{\mathbf{w}} = \mathbf{A}\tilde{\mathbf{v}}^{(k-1)}$
    $\tilde{\mathbf{v}}^{(k)} = \dfrac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|_2}$
    $\lambda^{(k)} = (\tilde{\mathbf{v}}^{(k)})^T \mathbf{A}\tilde{\mathbf{v}}^{(k)}$
end for

The series of steps followed by the Python program is:

1. Build the Markov transition graph **M**, using the nodes in the graph.
2. Run the power iteration for a maximum of 100 iterations. Stop the loop if the iteration converges before this. Convergence is measured by taking the 2-norm of the difference of the eigenvectors of successive iterates, i.e. $||\mathbf{v^{(k+1)}} - \mathbf{v^{(k)}}||_2$. [1]
3. The highest and lowest PageRank and their respective nodes are identified and output.
4. The required plots are saved.

Results

Performing power iterations on this matrix, we obtain the following results:

Number of iterations required: 40
Highest PageRank: Node 2 (PageRank: 0.4685)
Lowest PageRank: Node 3 (PageRank: 0.0000)

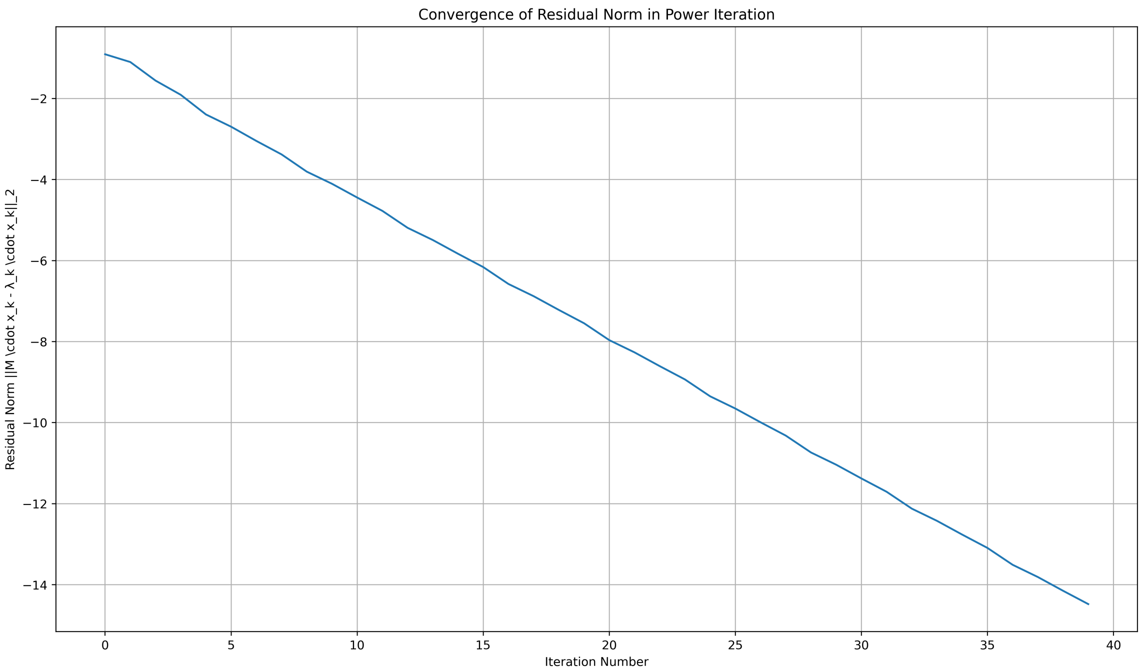
[1] For this question, as no tolerance was specified, I decided to use 10^{-6}.

Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: October 26, 2025

Assignment No: Assignment 5 - Question 1
Course Code: DS284
Course Name: Numerical Linear Algebra
Term: AUG 2025

Plots

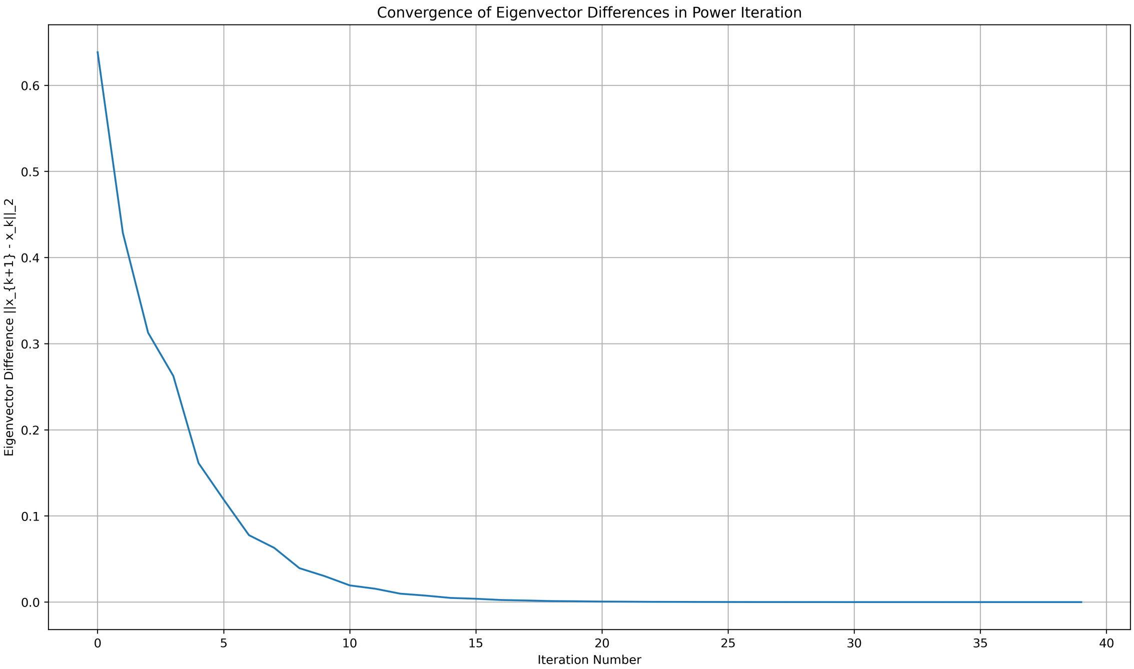
The log plot of the 2-norm of the residual of the eigenvalue problem, i.e. $\log ||\mathbf{M}\mathbf{x}^{(\mathbf{k+1})} - \lambda^{(k+1)}\mathbf{x}^{(\mathbf{k+1})}||_2$, is as follows:



Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: October 26, 2025

Assignment No: Assignment 5 - Question 1
Course Code: DS284
Course Name: Numerical Linear Algebra
Term: AUG 2025

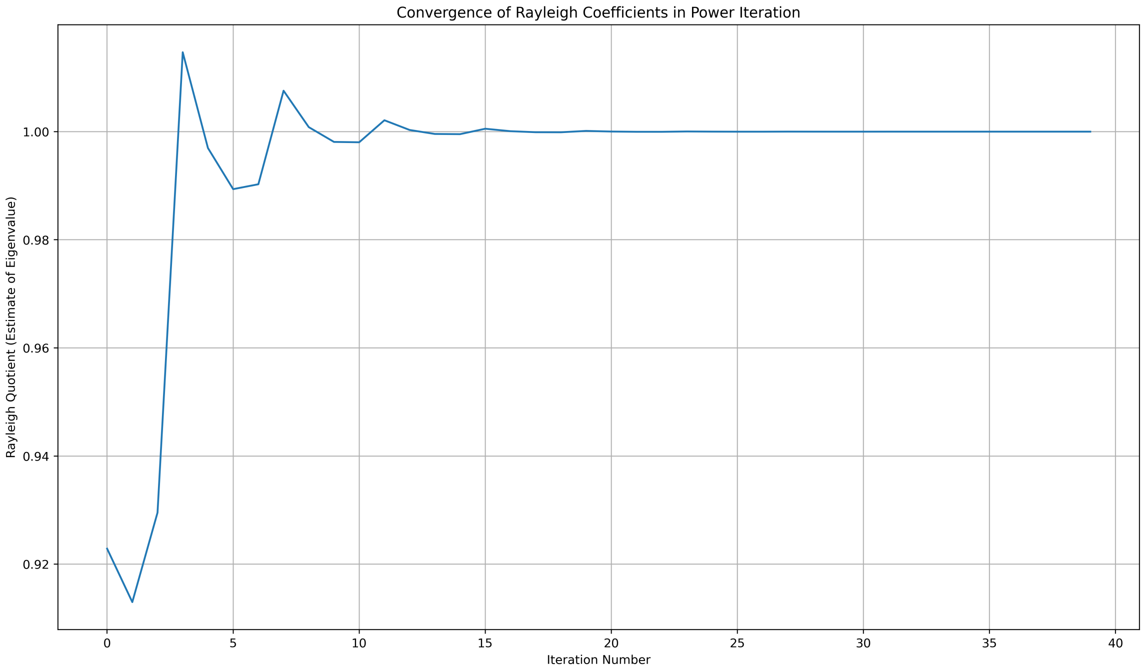
The 2-norm plot of the difference between the vectors of successive iterates, i.e. $||\mathbf{x}^{(\mathbf{k+1})} - \mathbf{x}^{(\mathbf{k})}||_2$, is as follows:



Name: Suhas Kamath
SR No: 06-18-01-10-51-25-1-25945
Email ID: suhaskamath@iisc.ac.in
Date: October 26, 2025

Assignment No: Assignment 5 - Question 1
Course Code: DS284
Course Name: Numerical Linear Algebra
Term: AUG 2025

The Rayleigh quotient calculated at each stage of the iteration is as follows:



Name:	Suhas Kamath	**Assignment No:**	Assignment 5 - Question 1
SR No:	06-18-01-10-51-25-1-25945	**Course Code:**	DS284
Email ID:	suhaskamath@iisc.ac.in	**Course Name:**	Numerical Linear Algebra
Date:	October 26, 2025	**Term:**	AUG 2025

Code

The Python code is as follows:

```python
import numpy as np
import matplotlib.pyplot as plt
import os

# --- Build the Markov transition matrix M ---
M = np.zeros((10, 10))

# Outgoing links dictionary: for each node j, list nodes i that j links to.
# Interpretation: j -> i for each i in links[j].
links = {
    0: [1, 2], 1: [2, 5], 2: [0, 1], 3: [4], 4: [5, 7],
    5: [6, 9], 6: [2, 7], 7: [4, 5], 8: [1, 7], 9: [8]
}

# Fill M so each nonzero column j distributes mass uniformly over the
# outgoing links of j. For example, if node j points to k_j targets,
# each target i receives 1/k_j probability from column j.
for j, i_list in links.items():
    k_j = len(i_list)
    for i in i_list:
        M[i, j] = 1.0 / k_j

# --- Power iteration parameters and initialization ---
max_iterations = 100        # hard cap to avoid infinite loops
tolerance = 1e-6            # stopping criterion on eigenvector change

# Random initial vector (nonzero), normalized to unit 2-norm.
x_k = np.random.rand(M.shape[0])
x_k /= np.linalg.norm(x_k)

# Containers to store convergence diagnostics for plotting/analysis
residuals = []                  # ||M x_k - lambda_k x_k||_2
eigenvector_differences = []    # ||x_{k+1} - x_k||_2
eigenvalues = []                # Rayleigh quotient estimates

print("Starting power iteration...")

# --- Power iteration loop ---
for k in range(max_iterations):
    # Apply the matrix and renormalize to keep vector magnitude bounded
    x_k_plus_1 = M @ x_k
    x_k_plus_1 /= np.linalg.norm(x_k_plus_1)

    # Rayleigh quotient provides an estimate of the eigenvalue for x_{k+1}
    eigenval = x_k_plus_1.T @ M @ x_k_plus_1
    eigenvalues.append(eigenval)

    # Residual norm: measures how well (lambda, x) satisfies the eigen eqn
    residuals.append(np.log(np.linalg.norm(M @ x_k_plus_1 - eigenval * x_k_plus_1)))

    # Change in eigenvector between iterations (useful stopping metric)
    eigenvector_differences.append(np.linalg.norm(x_k_plus_1 - x_k))

```

```python
    x_k = x_k_plus_1

    # Check convergence
    if eigenvector_differences[-1] < tolerance:
        print(f"\nConverged after {k+1} iterations.")
        break

# PageRank interpretation: normalize so components sum to 1 (probability)
pagerank_vector = x_k / np.linalg.norm(x_k)

print(pagerank_vector)

print(f"Highest PageRank: Node {np.argmax(pagerank_vector)} (Rank: {np.max(pagerank_vector):.4f})")
print(f"Lowest PageRank: Node {np.argmin(pagerank_vector)} (Rank: {np.min(pagerank_vector):.4f})")

save_dir = r"H:\My Drive\Numerical Linear Algebra\Assignments\Assignment 5"
print(f"\nAttempting to save plots to: {save_dir}")

#Plot 1
plt.figure(figsize=(16,9))
plt.plot(residuals)
plt.title('Convergence of Residual Norm in Power Iteration')
plt.xlabel('Iteration Number')
plt.ylabel('Residual Norm ||M x_k - lambda_k x_k||_2')
plt.grid(True)
plt.savefig(os.path.join(save_dir, 'residual_norm.png'), dpi=400)


#Plot 2
plt.figure(figsize=(16,9))
plt.plot(eigenvector_differences)
plt.title('Convergence of Eigenvector Differences in Power Iteration')
plt.xlabel('Iteration Number')
plt.ylabel('Eigenvector Difference ||x_{k+1} - x_k||_2')
plt.grid(True)
plt.savefig(os.path.join(save_dir, 'eigenvector_differences.png'), dpi=400)


#Plot 3
plt.figure(figsize=(16,9))
plt.plot(eigenvalues)
plt.title('Convergence of Rayleigh Coefficients in Power Iteration')
plt.xlabel('Iteration Number')
plt.ylabel('Rayleigh Quotient (Estimate of Eigenvalue)')
plt.grid(True)
plt.savefig(os.path.join(save_dir, 'rayleigh_coefficients.png'), dpi=400)

print("\nAll plots saved.")
```

Output

The output of the code is as follows:

```
Starting power iteration...

Converged after 40 iterations.
[0.23426054 0.4685212  0.46852117 0.         0.1561738  0.46852112
 0.23426067 0.31234782 0.23426101 0.23426067]
Highest PageRank: Node 1 (Rank: 0.4685)
Lowest PageRank: Node 3 (Rank: 0.0000)

Attempting to save plots to: H:\My Drive\Numerical Linear
 ↪ Algebra\Assignments\Assignment 5

All plots saved.
```