# CENG 371 - Scientific Computing
# Fall' 2024 - 2025
# Homework 3

Karaçanta, Kaan
e244854@metu.edu.tr

---

## Question 1

3. For the matrix A, we can find the biggest eigenvalue and its corresponding eigenvector by using the power method, with my code in the power_method.py you can see the tests. Likewise, by using inverse power method, we can find the smallest eigenvalue and its corresponding eigenvector, yet we need to add a shift value to the matrix A, which I have chosen as 0.01. The eigenvalues and eigenvectors are as follows:

   The eigenvalues are:

$$\lambda_{max} = 3.7321, \quad \lambda_{min} = 0.2679$$

   The eigenvectors are:

$$v_{max} = \begin{bmatrix} 0.2887 \\ -0.5000 \\ 0.5774 \\ -0.5000 \\ 0.2887 \end{bmatrix}, \quad v_{min} = \begin{bmatrix} 0.2887 \\ 0.5000 \\ 0.5774 \\ 0.5000 \\ 0.2887 \end{bmatrix}$$

4. First, we can find the eigenvalues like the following:

$$\det(B - \lambda I) = 0$$

$$\begin{vmatrix} 0.2-\lambda & 0.3 & -0.5 \\ 0.6 & -0.8-\lambda & 0.2 \\ -1 & 0.1 & 0.9-\lambda \end{vmatrix} = 0$$

$$-\lambda^3 + \frac{3}{10}\lambda^2 + \frac{7}{5}\lambda = 0$$

$$\lambda_1 = 0, \quad \lambda_2 = \frac{3-\sqrt{569}}{20}, \quad \lambda_3 = \frac{3+\sqrt{569}}{20}$$

   Then, we can find the eigenvectors as the following:

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} \frac{8\sqrt{569}+103}{197} \\ \frac{-37\sqrt{569}-895}{394} \\ 1 \end{bmatrix}, \quad v_3 = \begin{bmatrix} \frac{-8\sqrt{569}+103}{197} \\ \frac{37\sqrt{569}-895}{394} \\ 1 \end{bmatrix}$$

   Thus, the largest eigenvalue and its corresponding eigenvector are:

$$\lambda_{max} = \frac{3+\sqrt{569}}{20}, \quad v_{max} = \begin{bmatrix} \frac{-8\sqrt{569}+103}{197} \\ \frac{37\sqrt{569}-895}{394} \\ 1 \end{bmatrix}$$

By using the power method, we are expected to find the same results. Here are the steps of the power method:

$$x_1 = Bx_0 = \begin{bmatrix} 0.2 & 0.3 & -0.5 \\ 0.6 & -0.8 & 0.2 \\ -1 & 0.1 & 0.9 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Since the result is zero, the continuing x values will be zero, and this means it found our eigenvector belonging to the maximum eigenvalue, but as I calculated before, this is not true. The reason for this, as far as I see, it is important to choose a proper starting vector. In this case, the starting vector is the eigenvector of this matrix, and when you multiply it with the matrix, it will give the multiplication of the eigenvector with the eigenvalue, which is not necessarily the biggest eigenvalue.

## Question 2

1. To extend this to find the largest $k$ eigenvalues, the idea involves deflation. After finding the largest eigenvalue and eigenvector is subtracted from the matrix $A$. The deflation formula is:

$$A_{i+1} = A_i - \lambda_i \frac{v_i v_i^T}{v_i^T v_i}$$

This process is repeated $k$ times, with the largest eigenvalue and eigenvector being computed at each step, followed by deflation. The key insight is that the deflation step eliminates the influence of previously computed eigenvalues and eigenvectors, allowing the power method to focus on the next largest eigenvalue. The overall process ensures that the largest $k$ eigenvalues and their corresponding eigenvectors are determined iteratively.

Given matrix $A$ with eigenvalue $\lambda_1$ and corresponding eigenvector $v_1$, we define:

$$P_1 = \frac{v_1 v_1^T}{v_1^T v_1}$$

$$A_2 = A - \lambda_1 P_1$$

This works because:

- $P_1$ is a projection matrix onto span$\{v_1\}$
- For any eigenpair $(\lambda_i, v_i)$ of $A$:

$$A_2 v_i = A v_i - \lambda_1 P_1 v_i = \lambda_i v_i - \lambda_1 (v_i^T v_1) v_1 / \|v_1\|^2$$

- When $i = 1$: $A_2 v_1 = \lambda_1 v_1 - \lambda_1 v_1 = 0$
- When $i \neq 1$: $v_i^T v_1 = 0$ (orthogonality), so $A_2 v_i = \lambda_i v_i$

Therefore, $A_2$ preserves all eigenpairs except $(\lambda_1, v_1)$, which becomes $(0, v_1)$. This allows us to find subsequent eigenvalues iteratively.

4. To compare the performance of power_k method (using deflation) and subspace iteration method on the can229 matrix, I implemented both methods and measured their execution times:

- Power_k method: 6.48 seconds
- Subspace iteration: 2.73 seconds

The subspace iteration method performed significantly better, being approximately 2.37 times faster than the power_k method. This efficiency difference can be attributed to several factors:

(a) The power_k method requires k separate deflations and power iterations, leading to accumulated computational overhead

(b) Subspace iteration works with all k eigenvectors simultaneously, leveraging efficient matrix operations

(c) The QR factorization in subspace iteration provides better numerical stability compared to sequential deflation

For large sparse matrices like can229, subspace iteration's ability to handle multiple eigenvectors simultaneously proves to be more efficient than the sequential approach of power_k method. However, due to the randomization of initial values in the algorithms, the results may vary slightly between runs.