Proseminar Numerische Mathematik 2 am 10 Juni 2025

Exercise 1 (Power iteration algorithm)

Consider the matrix $A \in M_N(\mathbb{R})$ defined by

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

In what follows, we will consider the cases $N = 10$ and $N = 100$.

1. Construct A for the 2 values of N and program the power method to compute the largest eigenvalue of A. Plot the evolution of the computed eigenvalue as a function of the number of iterations. Provide the order and rate of convergence.
2. Program the inverse power method to compute the smallest eigenvalue of A. Plot the evolution of this computed eigenvalue as a function of the number of iterations. Provide the order and rate of convergence.
3. (Optional) It can be shown that the eigenvalues of A lie between 0 and 4. Implement an algorithm to compute the eigenvalue of A closest to 1.

Exercise 2 (Power iteration algorithm)

In this example, we explore the limitations of the power method. Let A be the matrix

$$A = \begin{pmatrix} 0.5172 & 0.5473 & -1.224 & 0.8012 \\ 0.5473 & 1.388 & 1.353 & -1.112 \\ -1.224 & 1.353 & 0.03642 & 2.893 \\ 0.8012 & -1.112 & 2.893 & 0.05827 \end{pmatrix}$$

The eigenvalues of A are $\mu_1 = -4$, $\mu_2 = 3$, $\mu_3 = 2$ and $\mu_4 = 1$. The eigenvectors corresponding to μ_1 and μ_2 are respectively:

$$w_1 = \begin{pmatrix} 0.3225 \\ -0.3225 \\ 0.6451 \\ -0.6129 \end{pmatrix}, \quad w_2 = \begin{pmatrix} -0.1290 \\ 0.1290 \\ 0.7419 \\ -0.6451 \end{pmatrix}$$

1. Implement the power method with the initial vector $x_0 = (1, 0, 0, 0)^T$. Study the evolution of the computed eigenvalue and plot $\|x_k\|_2$ as a function of k.
2. Now start with the initial vector $x_0 = (1, 1, 1)^T$ (almost orthogonal to w_1). Repeat the same analysis. Explain the observed phenomenon.
3. Start with $x_0 = (1, 1, 0, 0)^T$ (orthogonal to both w_1 and w_2). Repeat the same analysis.

Exercise 3 (SVD)

1. Create the following 2D NumPy arrays: A small 3×2 matrix. A 5×5 matrix with repeated rows/columns. A 10×8 matrix with random values
2. For each matrix:
   - Print the original matrix
   - Compute its SVD using `numpy.linalg.svd()`
   - Print matrices U, S, V^T
   - Verify that $U \cdot \text{diag}(S) \cdot V^T$ reconstructs the original matrix
   - Plot the singular values using `matplotlib`
3. Load the image file `dog.png` and convert it to grayscale.
   - Convert the image to a 2D NumPy float array.
   - Compute its full SVD: $X = U \cdot S \cdot V^T$
   - Plot the first 100 singular values and their cumulative normalized sum.
   - Reconstruct and display approximations using the first $r = 5, 20, 100$ singular values.
   - Compute the relative Frobenius norm error:

$$\frac{\|X - X_r\|_F}{\|X\|_F}$$

   - What do you notice about the singular values of small vs. large matrices?
   - How many singular values are needed to retain 90% of the image's energy?
   - How does increasing r affect visual quality?

Hints: You will need the modules:

from PIL import Image

from numpy.linalg import svd, norm

The Frobenius norm is called using: norm(., 'fro')

To compute the relative cumulative sum use np.cumsum(S**2)/np.sum(S**2)

Exercise 4

Implement the QR algorithm with and without shift of the matrix (5.9) given in the script, page 99 and recover expected linear and quadratic convergence of the two approaches respectively as shown in figures 5.4-5.5