# Homework: Applied Linear Algebra with NumPy in Scientific Python

## Instructions

For each problem, write Python code using NumPy to perform the necessary computations. Provide explanations and code comments to justify your approach. Ensure that your code runs without errors.

## Problem 1: PageRank Algorithm (25 points)

The **PageRank** algorithm, used by search engines like Google, ranks websites based on the number of links pointing to them. You can represent a simplified web network as a **stochastic matrix** where each entry $M[i, j]$ is the probability that a user on page $j$ will click a link to page $i$.

Given the following matrix representing a small web network:

$$M = \begin{bmatrix} 0 & 0 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 1/2 & 0 \end{bmatrix}$$

1. Use the **power iteration** method to approximate the dominant eigenvector of matrix $M$. This eigenvector represents the PageRank scores of the pages.

2. Start with an initial rank vector $v$ (of ones) and iterate until convergence (i.e., until the difference between consecutive rank vectors is small).

3. Normalize the resulting eigenvector so that the sum of all ranks is 1. What do the entries of the eigenvector represent?

4. Based on the final PageRank scores, which page is ranked the highest, and why?

## Problem 2: Dimensionality Reduction via PCA (25 points)

Principal Component Analysis (PCA) is widely used in data analysis for dimensionality reduction, often to compress data while preserving the most important patterns.

You are given a dataset containing height and weight measurements for 100 people (already standardized):

$$\text{Data} = \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ \vdots & \vdots \\ h_{100} & w_{100} \end{bmatrix}$$

1. Compute the covariance matrix of the data.
2. Perform **eigenvalue decomposition** on the covariance matrix using NumPy's `np.linalg.eig()`.
3. Identify the principal components and explain how they relate to the variance in the dataset.
4. Reduce the dataset to 1D by projecting it onto the principal component that captures the most variance. Plot the original data and the 1D projection.

## Problem 3: Linear Regression via Least Squares (15 points)

A real estate company wants to predict house prices based on features like square footage, number of bedrooms, and age of the house. The company has the following dataset:

$$\text{X} = \begin{bmatrix} \text{Square Footage} & \text{Bedrooms} & \text{Age (years)} \\ 2100 & 3 & 20 \\ 2500 & 4 & 15 \\ 1800 & 2 & 30 \\ 2200 & 3 & 25 \end{bmatrix}$$

The corresponding house prices (in \$1000s) are:

$$\text{y} = \begin{bmatrix} 460 \\ 540 \\ 330 \\ 400 \end{bmatrix}$$

1. Set up the system as a least-squares problem $X\beta = y$, where $\beta$ represents the coefficients (weights) for square footage, bedrooms, and age.

2. Solve for $\beta$ using NumPy's `np.linalg.lstsq()`.

3. Use the resulting model to predict the price of a house with 2400 square feet, 3 bedrooms, and 20 years old.

4. Discuss the role of the least-squares method in this prediction task and compare its performance to an alternative method (such as direct solution via `np.linalg.solve()`).