The University of Alabama
College of Engineering · Computer Science

Assignment 2

Submission Instructions

- Submit your solutions as a **separate PDF file**.
- Clearly label each problem number and part (e.g., Problem 1a, Problem 2c).
- For Python problems, include screenshots of your code and output.
- Ensure your work is legible and well-organized. Show all steps for full credit.

Problem 1: Change of Basis & Coordinate Systems (15 Points)

Given the standard basis vectors $\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Consider a new basis $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$ defined by the matrix $B = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$.

1. Write expressions for $\mathbf{b}_1$ and $\mathbf{b}_2$ in terms of the standard basis.
2. A vector has coordinates $[\mathbf{v}]_{\mathcal{B}} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$ in the **new basis**. Find its coordinates in the **standard basis**.
3. A vector has coordinates $\mathbf{x} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ in the **standard basis**. Find its coordinates in the **new basis** $\mathcal{B}$. *(Hint: You will need B^{-1}).*

Solution:

■

Problem 2: The Geometry of Determinants **(15 Points)**

1. Let $A = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$. Compute $\det(A)$.
2. Apply A to the unit square vectors $\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Plot or describe where they land.
3. Explain why this transformation makes it impossible to find an inverse (i.e., you cannot "go back" to the original square). Use the concept of "collapsing dimensions" in your answer.

Solution:

■

Problem 3: Gaussian Elimination (Manual) **(20 Points)**

Consider the system of equations:

$$\begin{cases} x + 2y + z = 8 \\ 2x + 5y + 2z = 18 \\ x + 3y + 4z = 19 \end{cases}$$

1. Form the augmented matrix $[A|\mathbf{b}]$.
2. Perform **row operations** step-by-step to reduce the matrix to **Row Echelon Form (Upper Triangular)**. Clearly state each operation (e.g., $R_2 \leftarrow R_2 - 2R_1$).
3. Perform **Back Substitution** to find the values of x, y, z.

Solution:

■

Problem 4: Computing the Inverse **(15 Points)**

Using the $[A|I] \to [I|A^{-1}]$ method (Gauss-Jordan Elimination), compute the inverse of:

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 2 & -1 & 3 \\ 4 & 1 & 8 \end{pmatrix}$$

Show all your steps. Verify your answer by multiplying AA^{-1}.

Solution:

■

Problem 5: The Geometry of Composition **(15 Points)**

Matrix multiplication corresponds to applying geometric transformations in sequence. The order matters!

Let $R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ (Rotation by 90° counter-clockwise).

Let $M = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ (Reflection across the x-axis).

1. Calculate $A = RM$ (Apply Reflection **then** Rotation).
2. Calculate $B = MR$ (Apply Rotation **then** Reflection).
3. Apply both A and B to the vector $\mathbf{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Do they land in the same place?
4. **Think:** Why does the order of operations change the physical outcome in this case?

Solution:

■

Problem 6: Python: Efficiency of Solving Systems (20 Points)

In this problem, you will demonstrate why we rarely compute the inverse explicitly in practice.

Task:

1. Generate a large random $N \times N$ matrix A (where $N = 1000$) and a random vector $\mathbf{b}$.
2. Use 'time' module to measure how long it takes to solve $A\mathbf{x} = \mathbf{b}$ using:
   - Method 1: 'x = np.linalg.inv(A) @ b'
   - Method 2: 'x = np.linalg.solve(A, b)'
3. Report the two times. Which one is faster?
4. Briefly explain why 'np.linalg.solve' is preferred over computing the inverse.

Paste your Python code and output below.

Solution:

```
import numpy as np

# Your Python code here
```

■