Department of Mathematics, IIT Madras

MA-5105-Scientific Computing & Advanced Programming Lab

Practice Problems 4

September 6, 2026

Lecturer: Saurav Samantaray *TA: Rohit Kumar Mani*

Loops and Matrix Algorithms

The following problems are designed to strengthen your understanding of `for` loops, nested loops, matrix indexing, and in-place matrix operations.

Instructions:

- Implement all algorithms using Python.
- Do not use NumPy.
- Do not use `max()`, `sum()`, `argmax()`, or list comprehensions unless explicitly permitted.
- Do not hard-code individual matrix elements.
- Your programs should work for matrices of arbitrary appropriate dimensions.

Q. 1 Matrix Multiplication

Given two matrices

$$A \in \mathbb{R}^{m\times n}, \qquad B \in \mathbb{R}^{n\times p},$$

implement matrix multiplication

$$C = AB$$

using three nested `for` loops.

For example, consider

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

Your program should calculate each element using

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj}.$$

(a) Implement the matrix multiplication without NumPy.

(b) Do not use `sum()`.

(c) Use exactly three nested `for` loops for the main computation.

(d) Test your program using matrices of different dimensions.

Hint: Think carefully about the roles of the three indices:

$$i = \text{row of } A, \qquad j = \text{column of } B, \qquad k = \text{summation index}.$$

Q. 2 Largest Element in a Submatrix

Consider the matrix

$$A = \begin{bmatrix} 2 & 8 & 1 & 4 \\ 7 & 3 & 9 & 2 \\ 5 & 6 & 4 & 1 \\ 3 & 2 & 8 & 7 \end{bmatrix}.$$

For each

$$k = 0, 1, 2,$$

search the submatrix

$$A[k : n,\ k : n]$$

and find the element with the largest absolute value.

Your program should return the value and its corresponding row and column indices.

For example, when $k = 1$, the search should be performed only over

$$\begin{bmatrix} 3 & 9 & 2 \\ 6 & 4 & 1 \\ 2 & 8 & 7 \end{bmatrix}.$$

(a) Use nested `for` loops to perform the search.

(b) Do not use `max()` or `argmax()`.

(c) Use absolute values when comparing elements.

(d) Make sure that the starting indices of your loops depend on k.

Challenge: Modify your program so that, for every k, it reports

$$(\text{pivot value}, \text{pivot row}, \text{pivot column}).$$

Q. 3 Gauss–Jordan Row Operations

Consider the system

$$\begin{aligned} 2x + y - z &= 8, \\ -3x - y + 2z &= -11, \\ -2x + y + 2z &= -3. \end{aligned}$$

Represent the system using the augmented matrix

$$[A|b] = \begin{bmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{bmatrix}.$$

Write a program that performs Gauss–Jordan elimination using loops.

At each pivot k:

(a) Normalize the pivot row so that the pivot becomes 1.

(b) For every other row $i \neq k$, calculate the multiplier $m = A_{ik}$.

(c) Update every element of that row according to $A_{ij} \leftarrow A_{ij} - mA_{kj}$.

Your program should eventually transform the augmented matrix into

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \end{bmatrix}.$$

Restrictions:

- Do not perform individual row operations manually.
- Use loops to identify rows and columns.
- Do not use NumPy.

- Print the matrix after each pivot operation.

Q. 4 Gaussian Elimination with Partial Pivoting

Implement Gaussian elimination with partial pivoting for the augmented system

$$[A|b] = \begin{bmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{bmatrix}.$$

Your program should transform the matrix into upper triangular form and then solve the system using back substitution.

At each step k, your algorithm should:

(a) Search for the pivot row among $k, k+1, \ldots, n-1$.

(b) Select the row having the largest absolute value in column k.

(c) Swap the pivot row with row k.

(d) For every row below the pivot, calculate $m_{ik} = \frac{A_{ik}}{A_{kk}}$.

(e) Update all required elements: $A_{ij} \leftarrow A_{ij} - m_{ik}A_{kj}$.

(f) After obtaining an upper triangular system, use back substitution to calculate $x_n, x_{n-1}, \ldots, x_1$.

Restrictions:

- Do not use NumPy.
- Do not use `max()` or `argmax()`.
- Do not hard-code any row operations.
- Use loops for pivot searching, row swapping, elimination, and back substitution.
- Your implementation should work for any square nonsingular matrix.

Final Challenge:

Test your implementation on

$$[A|b] = \begin{bmatrix} 0 & 2 & 1 & 4 \\ 1 & -2 & -3 & -1 \\ 2 & 3 & 1 & 9 \end{bmatrix}.$$

Explain why partial pivoting is essential for this example.