5.2.22

EE25BTECH11019 – Darji Vivek M.

Question

Using elementary transformations, find the inverse of the following matrix.

$$\begin{pmatrix} 2 & 3 \\ 5 & 7 \end{pmatrix}$$

Theoretical Solution

To find the inverse of a matrix using the Gauss–Jordan method:

$$\left(\begin{array}{cc|cc} 2 & 3 & 1 & 0 \\ 5 & 7 & 0 & 1 \end{array}\right) \xrightarrow{R_1 \leftarrow R_1/2,\ R_2 \leftarrow R_2 - 5R_1} \left(\begin{array}{cc|cc} 1 & \frac{3}{2} & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & -\frac{5}{2} & 1 \end{array}\right) \xrightarrow{R_2 \leftarrow -2R_2,\ R_1 \leftarrow R_1 - \frac{3}{2}R_2} \left(\begin{array}{cc|cc} 1 & 0 & -7 & 3 \\ 0 & 1 & 5 & -2 \end{array}\right) \tag{1}$$

Thus, the inverse is:

$$A^{-1} = \begin{pmatrix} -7 & 3 \\ 5 & -2 \end{pmatrix}. \tag{2}$$

C Code: parallel_funcs.c

```
#include <stdio.h>
#define N 2   // matrix size (you can generalize)
void inverse(double A[N][N], double inv[N][N]) {
    // Step 1: Create augmented matrix [A|I]
    double aug[N][2*N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            aug[i][j] = A[i][j];            // copy A
            aug[i][j+N] = (i == j) ? 1 : 0; //
                identity
        }
    }
    // Step 2: Gauss-Jordan elimination
    for (int i = 0; i < N; i++) {
        // Make pivot = 1
        double pivot = aug[i][i];
        for (int j = 0; j < 2*N; j++) {
            aug[i][j] /= pivot;
        }
```

C Code: parallel_funcs.c

```
        // Eliminate other rows
        for (int k = 0; k < N; k++) {
            if (k != i) {
                double factor = aug[k][i];
                for (int j = 0; j < 2*N; j++) {
                    aug[k][j] -= factor * aug[i][j];
                }
            }
        }
    }
    // Step 3: Extract inverse from augmented matrix
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            inv[i][j] = aug[i][j+N];
        }
    }
}
```

Python: Plotting Lines

```
import ctypes
import numpy as np
import sympy as sp

# ====== Load Shared Library (.so) ======
lib = ctypes.CDLL("./libinverse.so")

# Define argument types for C function
lib.inverse.argtypes = [
    ctypes.POINTER((ctypes.c_double * 2) * 2),  # Input matrix A
    ctypes.POINTER((ctypes.c_double * 2) * 2)   # Output inverse
        matrix
]

# ====== Define the Matrix ======
A = np.array([[2, -6],
              [1, -2]], dtype=np.double)

inv = np.zeros((2, 2), dtype=np.double)
```

Python: Plotting Lines

```python
# ====== Call the C Function ======
lib.inverse(
    A.ctypes.data_as(ctypes.POINTER((ctypes.c_double * 2) * 2)),
    inv.ctypes.data_as(ctypes.POINTER((ctypes.c_double * 2) * 2))
)

# ====== Convert and Display Results ======
A_sym = sp.Matrix(A)
A_inv_sym = A_sym.inv()           # Sympy inverse (for verification
    )
A_inv_c = sp.Matrix(inv)          # Inverse from C code

print("Matrix A:")
sp.pprint(A_sym)

print("\nInverse computed in C:")
sp.pprint(A_inv_c)

print("\nInverse computed in Sympy:")
sp.pprint(A_inv_sym)
```