5.4.36

Nipun Dasari - EE25BTECH11042

January 9, 2025

Question

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

$$\begin{pmatrix} 2 & -1 & -2 \\ 0 & 2 & -1 \\ 3 & -5 & 0 \end{pmatrix}$$

Theoretical Solution

Let us solve the given question theoretically and then verify the solution computationally.

To solve for the inverse of a matrix, we can employ the Gauss-Jordan approach.

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

$$R_3 \leftarrow R_3 - 3R_1 \left(\begin{array}{ccc|ccc} 1 & -1/2 & -1 & 1/2 & 0 & 0 \\ 0 & 2 & -1 & 0 & 1 & 0 \\ 0 & -7/2 & 3 & -3/2 & 0 & 1 \end{array}\right) \tag{2}$$

$$\frac{1}{\quad} \left(\begin{array}{ccc|ccc} 1 & -1/2 & -1 & 1/2 & 0 & 0 \\ & & & & & \end{array}\right)$$

Theoretical Solution

$$\therefore \text{Inverse of the given Matrix: } \begin{pmatrix} -1 & 2 & 1 \\ -3/5 & 6/5 & 2/5 \\ -6/5 & 7/5 & 4/5 \end{pmatrix} \tag{7}$$

C Code

```c
#include <stdio.h>
#define N 3 // 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
        }
    }
```

C Code

```
// Step 2: GaussJordan 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;
        }
        // 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]
```

C Code

```
                                    // 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 Code using shared output

```
import ctypes
import numpy as np
import sympy as sp
# Load C library
lib = ctypes.CDLL(./5.4.36.so)
# Define function signature
lib.inverse.argtypes = [ctypes.POINTER((
    ctypes.c_double * 3) * 3),
ctypes.POINTER((ctypes.c_double * 3) * 3)]
```

Python Code using shared output

```
# Input matrix
A = np.array([[2, -1, -2],
[0, 2, -1],
[3, -5, 0]], dtype=np.double)

inv = np.zeros((3,3), dtype=np.
    double)
# Call C function
lib.inverse(A.ctypes.data_as(ctypes.
    POINTER((ctypes.c_double * 3) *
    3)),
inv.ctypes.data_as(ctypes.POINTER((
    ctypes.c_double * 3) * 3)))
inverse=sp.Matrix(inv)
sp.pprint(inverse)
```

Python Code

```
import sympy as sp
A = sp.Matrix([[2, -1, -2], [0, 2,
    -1],[3, -5, 0]])
A_inv = A.inv()
sp.pprint(A_inv)
```