5.4.21

Harsha-EE25BTECH11026

September 5,2025

Question

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

$$\begin{pmatrix} 2 & -6 \\ 1 & -2 \end{pmatrix}$$

Theoretical Solution

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

$$\left(\begin{array}{cc|cc} 2 & -6 & 1 & 0 \\ 1 & -2 & 0 & 1 \end{array}\right) \xleftrightarrow[R_2 \leftarrow R_2 - R_1]{R_1 \leftarrow \frac{R_1}{2}} \left(\begin{array}{cc|cc} 1 & -3 & \frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{2} & 1 \end{array}\right) \xleftrightarrow{R_1 \leftarrow R_1 + 3R_2} \left(\begin{array}{cc|cc} 1 & 0 & -1 & 3 \\ 0 & 1 & -\frac{1}{2} & 1 \end{array}\right) \tag{1}$$

$$\therefore \text{Inverse of the given Matrix: } \begin{pmatrix} -1 & 3 \\ -\frac{1}{2} & 1 \end{pmatrix} \tag{2}$$

C Code -Finding Inverse of a Matrix

```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
        }
    }
```

C Code -Finding Inverse of a Matrix

```
// 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] -= factor * aug[i][j];
              }
          }
      }
  }
```

C Code -Finding Inverse of a Matrix

```
    // 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+C code

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

# Load C library
lib = ctypes.CDLL("./libinverse.so")

# Define function signature
lib.inverse.argtypes = [ctypes.POINTER((ctypes.c_double * 2) * 2)
    ,
                        ctypes.POINTER((ctypes.c_double * 2) * 2)]

# Input matrix
A = np.array([[2, -6],
              [1, -2]], dtype=np.double)

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

Python+C code

```
# Call 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)))

inverse=sp.Matrix(inv)
sp.pprint(inverse)
```

Python code

```
import sympy as sp

A = sp.Matrix([[2, -6], [1, -2]])
A_inv = A.inv()
sp.pprint(A_inv)
```