5.4.5

Finding Inverse

EE25BTECH11010 - Arsh Dhoke

Question

Using elementary transformations, find the inverse of the following matrix:
$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$.

Solution

We know

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{1}$$

where $\mathbf{I}$ is the identity matrix $\mathbf{I}_2$.

Computing inverse

The augmented matrix for the given matrix will be

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

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

Solution (final)

$$\therefore \quad \mathbf{A}^{-1} = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \tag{4}$$

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

We can verify the computed inverse using python code by showing $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

C Code

```c
#include <stdio.h>

int inverse2x2(double A[2][2], double inv[2][2]) {
    double a = A[0][0], b = A[0][1];
    double c = A[1][0], d = A[1][1];
    double det = a*d - b*c;

    if (det == 0) {
        printf("Matrix is singular, inverse doesn't exist.\n");
        return 0;
    }
    inv[0][0] = d / det;
    inv[0][1] = -b / det;
    inv[1][0] = -c / det;
    inv[1][1] = a / det;

    return 1;
}
```

Python Code

```
import numpy as np

#given matrix
A = np.array([[1, 2],[2, 1]])

#computed inverse
A_inverse= np.array([[2/3, -1/3], [-1/3, 2/3]])

#verification
B = A@A_inverse
print(B)
```

Python+ C Code

```
import ctypes
import numpy as np

# Load the shared library
lib = ctypes.CDLL("./code.so")

# Define the function signature
lib.inverse2x2.argtypes = [
    (ctypes.c_double * 2) * 2, # input matrix A
    (ctypes.c_double * 2) * 2 # output matrix inv
]
lib.inverse2x2.restype = ctypes.c_int

# Prepare input matrix
A = ((ctypes.c_double * 2) * 2)()
A[0][0], A[0][1] = 1.0, 2.0
A[1][0], A[1][1] = 2.0, 1.0
```

Python+ C Code

```
# Prepare output matrix
inv = ((ctypes.c_double * 2) * 2)()

# Call the C function
status = lib.inverse2x2(A, inv)

if status:
    # Convert result to numpy array for convenience
    result = np.array([[inv[0][0], inv[0][1]],
                       [inv[1][0], inv[1][1]]])
    print("Inverse matrix:")
    print(result)
else:
    print("Matrix is singular  no inverse.")
```