4.5.14

BEERAM MADHURI - EE25BTECH11012

September 2025

Question

Using elementary transformations, find the inverse of the following matrix

$$\begin{pmatrix} 2 & 2 \\ 4 & 3 \end{pmatrix}$$

We know that

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

where $\mathbf{I}$ is the 2×2 identity matrix
Now we get the augmented matrix

$$\left(\begin{array}{cc|cc} 2 & 2 & 1 & 0 \\ 4 & 3 & 0 & 1 \end{array}\right) \xrightarrow{R_2 \to R_2 - 2R_1} \left(\begin{array}{cc|cc} 2 & 2 & 1 & 0 \\ 0 & -1 & -2 & 1 \end{array}\right) \tag{2}$$

(3)

$$\xrightarrow[R_2 \to -R_2]{R_1 \to \frac{R_1}{2}} \begin{pmatrix} 1 & 1 & | & \frac{1}{2} & 0 \\ 0 & 1 & | & 2 & -1 \end{pmatrix} \xrightarrow{R_1 \to R_1 - R_2} \begin{pmatrix} 1 & 0 & | & -\frac{3}{2} & 1 \\ 0 & 1 & | & 2 & -1 \end{pmatrix} \tag{4}$$

Therefore

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

This can be verified by $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

Python Code

```
import numpy as np


a = np.array([[2, 2], [4, 3]])
inverse_a = np.array([[],[]])

b = a@inverse_a
print(b)
```

```c
#include <stdio.h>
#include <stdlib.h> // For exit()

// A function to print a 2x2 matrix
void printMatrix(double mat[2][2]) {
    for (int i = 0; i < 2; i++) {
        printf(" |");
        for (int j = 0; j < 2; j++) {
            // %.2f prints the float with 2 decimal places
            printf("%8.2f", mat[i][j]);
        }
        printf(" |\n");
    }
}
```

C Code

```c
int main() {
    // The original matrix from the question
    double matrix[2][2] = {
        {2.0, 2.0},
        {4.0, 3.0}
    };

    printf("Original Matrix A:\n");
    printMatrix(matrix);
```

C Code

```c
// --- Step 1: Check if the inverse exists (determinant != 0) ---
// Determinant = ad - bc
double det = matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
if (det == 0) {
    printf("\nInverse does not exist because the determinant is zero.\n");
    return 1; // Exit with an error
}
```

C Code

```c
// --- Step 2: Create an augmented matrix [A|I] ---
// 'I' is the 2x2 identity matrix
double augmented[2][4];
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        augmented[i][j] = matrix[i][j]; // Copy matrix A
    }
}
// Append the identity matrix
augmented[0][2] = 1.0;
augmented[0][3] = 0.0;
augmented[1][2] = 0.0;
augmented[1][3] = 1.0;
```

C Code

```c
// --- Step 3: Apply Gauss-Jordan elimination ---
// Goal: Transform the left side of the augmented matrix into the identity matrix.

// Make the first element of the first row (pivot) equal to 1
// R1 -> R1 / 2
double pivot1 = augmented[0][0];
for (int j = 0; j < 4; j++) {
    augmented[0][j] /= pivot1;
}
```

C Code

```c
// Make the first element of the second row equal to 0
// R2 -> R2 - 4 * R1
double factor1 = augmented[1][0];
for (int j = 0; j < 4; j++) {
    augmented[1][j] -= factor1 * augmented[0][j];
}
// Make the second element of the second row (pivot) equal to 1
// R2 -> R2 / -1
double pivot2 = augmented[1][1];
for (int j = 0; j < 4; j++) {
    augmented[1][j] /= pivot2;
}
```

C Code

```
    // Make the second element of the first row equal to 0
    // R1 -> R1 - 1 * R2
    double factor2 = augmented[0][1];
    for (int j = 0; j < 4; j++) {
        augmented[0][j] -= factor2 * augmented[1][j];
    }

    // --- Step 4: Extract the inverse matrix ---
    // The inverse is now on the right side of the augmented
        matrix
```

C Code

```
    double inverse[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            inverse[i][j] = augmented[i][j + 2];
        }
    }
    printf("\nFound Inverse Matrix A:\n");
    printMatrix(inverse);
    return 0;
}
```

Python and C Code

```
import ctypes

# Define the C double type
DoubleArray2x2 = ctypes.c_double * 2
Matrix2x2 = DoubleArray2x2 * 2 # 2x2 matrix

def print_matrix(mat):
    for i in range(2):
        print(" |", end='')
        for j in range(2):
            print(f"{mat[i][j]:8.2f}", end='')
        print(" |")
```

Python and C Code

```python
# Initialize matrix A
matrix = Matrix2x2(
    DoubleArray2x2(2.0, 2.0),
    DoubleArray2x2(4.0, 3.0)
)

print("Original Matrix A:")
print_matrix(matrix)
```

Python and C Code

```
# Step 1: Calculate determinant
det = matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
if det == 0:
    print("\nInverse does not exist because the determinant is
        zero.")
    exit(1)
# Step 2: Create augmented matrix [A | I] using ctypes
AugmentedRow = ctypes.c_double * 4
AugmentedMatrix = AugmentedRow * 2
augmented = AugmentedMatrix(
    AugmentedRow(matrix[0][0], matrix[0][1], 1.0, 0.0),
    AugmentedRow(matrix[1][0], matrix[1][1], 0.0, 1.0)
)
```

Python and C Code

```
# Step 3: Gauss-Jordan Elimination
# Row 1 normalization
pivot1 = augmented[0][0]
for j in range(4):
    augmented[0][j] /= pivot1
# Row 2 elimination
factor1 = augmented[1][0]
for j in range(4):
    augmented[1][j] -= factor1 * augmented[0][j]
```

Python and C Code

```
# Row 2 normalization
pivot2 = augmented[1][1]
for j in range(4):
    augmented[1][j] /= pivot2

# Row 1 elimination
factor2 = augmented[0][1]
for j in range(4):
    augmented[0][j] -= factor2 * augmented[1][j]
```

Python and C Code

```
# Step 4: Extract inverse
inverse = Matrix2x2(
    DoubleArray2x2(augmented[0][2], augmented[0][3]),
    DoubleArray2x2(augmented[1][2], augmented[1][3])
)

print("\nFound Inverse Matrix A:")
print_matrix(inverse)
```