5.4.41

RAVULA SHASHANK REDDY - EE25BTECH11047

October 1, 2025

Question

Using elementary transformations, find the inverse of the matrix

$$A = \begin{pmatrix} 2 & 1 & 3 \\ 4 & -1 & 0 \\ -7 & 2 & 1 \end{pmatrix}.$$

Solution

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

$$[A \mid I] = \begin{pmatrix} 2 & 1 & 3 & 1 & 0 & 0 \\ 4 & -1 & 0 & 0 & 1 & 0 \\ -7 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$\xrightarrow{R_1 \to \frac{1}{2} R_1} \begin{pmatrix} 1 & \frac{1}{2} & \frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 4 & -1 & 0 & 0 & 1 & 0 \\ -7 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$\xrightarrow{R_2 \to R_2 - 4R_1,\ R_3 \to R_3 + 7R_1} \begin{pmatrix} 1 & \frac{1}{2} & \frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 0 & -3 & -6 & -2 & 1 & 0 \\ 0 & \frac{11}{2} & \frac{23}{2} & \frac{7}{2} & 0 & 1 \end{pmatrix} \tag{4}$$

Solution

$$\xrightarrow{R_2 \to -\frac{1}{3} R_2} \begin{pmatrix} 1 & \frac{1}{2} & \frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 2 & \frac{2}{3} & -\frac{1}{3} & 0 \\ 0 & \frac{11}{2} & \frac{23}{2} & \frac{7}{2} & 0 & 1 \end{pmatrix} \tag{5}$$

$$\xrightarrow{R_1 \to R_1 - \frac{1}{2} R_2,\ R_3 \to R_3 - \frac{11}{2} R_2} \begin{pmatrix} 1 & 0 & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 1 & 2 & \frac{2}{3} & -\frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{6} & \frac{11}{6} & 1 \end{pmatrix} \tag{6}$$

$$\xrightarrow{R_3 \to 2R_3} \begin{pmatrix} 1 & 0 & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 1 & 2 & \frac{2}{3} & -\frac{1}{3} & 0 \\ 0 & 0 & 1 & -\frac{1}{3} & \frac{11}{3} & 2 \end{pmatrix} \tag{7}$$

Solution

$$\xrightarrow{R_1 \to R_1 - \frac{1}{2}R_3,\ R_2 \to R_2 - 2R_3} \begin{pmatrix} 1 & 0 & 0 & \frac{1}{3} & -\frac{5}{3} & -1 \\ 0 & 1 & 0 & \frac{4}{3} & -\frac{23}{3} & -4 \\ 0 & 0 & 1 & -\frac{1}{3} & \frac{11}{3} & 2 \end{pmatrix} \tag{8}$$

$$A^{-1} = \begin{pmatrix} \frac{1}{3} & -\frac{5}{3} & -1 \\ \frac{4}{3} & -\frac{23}{3} & -4 \\ -\frac{1}{3} & \frac{11}{3} & 2 \end{pmatrix} \tag{9}$$

C Code

```c
    #include <stdio.h>

int main() {
    int i, j, k;
    double A[3][3] = {
        {2, 1, 3},
        {4, -1, 0},
        {-7, 2, 1}
    };

    double I[3][3] = {
        {1, 0, 0},
        {0, 1, 0},
        {0, 0, 1}
    };

    // Perform Gauss-Jordan elimination
    for (i = 0; i < 3; i++) {
```

C Code

```c
        // Make the diagonal element 1
        double diag = A[i][i];
        for (j = 0; j < 3; j++) {
            A[i][j] /= diag;
            I[i][j] /= diag;
        }

        // Make other elements in the column 0
        for (k = 0; k < 3; k++) {
            if (k != i) {
                double factor = A[k][i];
                for (j = 0; j < 3; j++) {
                    A[k][j] -= factor * A[i][j];
                    I[k][j] -= factor * I[i][j];
                }
            }
        }
    }
```

```
    // Print the inverse
    printf(Inverse matrix is:\n);
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            printf(%8.3f , I[i][j]);
        }
        printf(\n);
    }

    return 0;
}
```

```
    import numpy as np
import libs.line.funcs as line
import libs.triangle.funcs as triangle

# Given matrix
A = np.array([[2, 1, 3],
              [4, -1, 0],
              [-7, 2, 1]], dtype=float)

# Compute inverse using numpy
A_inv = np.linalg.inv(A)
```

Python Direct

```
# Display results
print(Matrix A:)
print(A)
print(\nInverse of A:)
print(A_inv)

# Verification
I_check = A @ A_inv
print(\nVerification A * A_inv = )
print(I_check)
```

Python Shared

```
    import ctypes
import numpy as np
import libs.line.funcs as line
import libs.triangle.funcs as triangle

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

# Define function signature
lib.inverse.argtypes = [ctypes.POINTER(ctypes.c_double),
                        ctypes.POINTER(ctypes.c_double)]
lib.inverse.restype = None

# Input matrix
A = np.array([[2, 1, 3],
              [4, -1, 0],
              [-7, 2, 1]], dtype=np.double)
```

Python Shared

```
A_inv = np.zeros((3,3), dtype=np.double)

# Call C function
lib.inverse(A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
            A_inv.ctypes.data_as(ctypes.POINTER(ctypes.c_double)))

print(Matrix A:)
print(A)

print(\nInverse from C (via ctypes):)
print(A_inv)

# Verify
print(\nVerification A * A_inv =)
print(A @ A_inv)
```