5.4.23

J.NAVYASRI- EE25BTECH11028

september 2025

Question:

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

$$A = \begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix}$$

Solution:

We want to find the inverse of the matrix

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

Step 1: Assume the inverse matrix

Let

$$A^{-1} = \begin{pmatrix} x & y \\ z & w \end{pmatrix} \tag{1}$$

By definition of inverse, we have

$$AA^{-1} = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2}$$

Solution:

Step 2: Multiply the matrices

$$\begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x & y \\ z & w \end{pmatrix} = \begin{pmatrix} 2x-3z & 2y-3w \\ -x+2z & -y+2w \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} 2x-3z & 2y-3w \\ -x+2z & -y+2w \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

From this multiplication, we get the following system of equations:

$$2x - 3z = 1 \tag{5}$$

$$2y - 3w = 0 \tag{6}$$

Solution:

$$-x + 2z = 0 \tag{7}$$

$$-y + 2w = 1 \tag{8}$$

Step 3: Solve the equations

From equation (7):

$$-x + 2z = 0 \Rightarrow x = 2z \tag{9}$$

Substitute into equation (5):

$$2(2z) - 3z = 1 \Rightarrow 4z - 3z = 1 \Rightarrow z = 1 \tag{10}$$

Then,

$$x = 2z = 2 \tag{11}$$

Solution:

From equation (8):

$$-y + 2w = 1 \Rightarrow y = 2w - 1 \tag{12}$$

Substitute into equation (6):

$$2(2w - 1) - 3w = 0 \Rightarrow 4w - 2 - 3w = 0 \Rightarrow w = 2 \tag{13}$$

Then,

$$y = 2w - 1 = 3 \tag{14}$$

Step 4: Write the inverse matrix

$$A^{-1} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} \tag{15}$$

Python Code

```python
import numpy as np
import matplotlib.pyplot as plt

# Define matrix A and its inverse
A = np.array([[2, -3],
              [-1, 2]])
A_inv = np.array([[2, 3],
                  [1, 2]])

# Define unit square vertices
square = np.array([[0, 0],
                   [1, 0],
                   [1, 1],
                   [0, 1],
                   [0, 0]]) # closed polygon
```

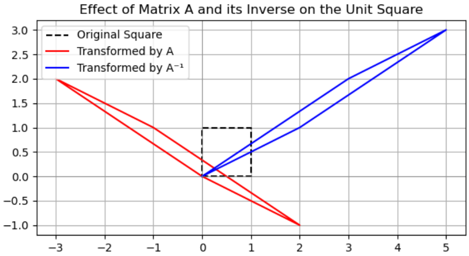
Python Code

```python
# Apply transformations
square_A = square @ A.T
square_Ainv = square @ A_inv.T

# Plotting
plt.figure(figsize=(7, 7))
plt.plot(square[:,0], square[:,1], 'k--', label="Original Square"
    )
plt.plot(square_A[:,0], square_A[:,1], 'r-', label="Transformed
    by A")
plt.plot(square_Ainv[:,0], square_Ainv[:,1], 'b-', label="
    Transformed by A")
```

Python Code

```python
# Style
plt.axhline(0, color='gray', lw=0.5)
plt.axvline(0, color='gray', lw=0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.title("Effect of Matrix A and its Inverse on the Unit Square"
    )
plt.grid(True)
plt.savefig("fig11.png")
plt.show()
```

Plot-Using by Python



C Code

```c
#include <stdio.h>

int main() {
    float a, b, c, d;
    float det;

    // Input matrix
    printf("Enter the elements of 2x2 matrix:\n");
    scanf("%f %f", &a, &b);
    scanf("%f %f", &c, &d);
```

C Code

```c
// Determinant
det = a * d - b * c;

if (det == 0) {
    printf("Inverse does not exist (determinant = 0)\n");
    return 0;
}

printf("Determinant = %.2f\n", det);
```

C Code

```c
    // Inverse formula: 1/det * [ d -b ]

    // [ -c a ]
    float inv_a = d / det;
    float inv_b = -b / det;
    float inv_c = -c / det;
    float inv_d = a / det;

    printf("Inverse matrix is:\n");
    printf("%.2f %.2f\n", inv_a, inv_b);
    printf("%.2f %.2f\n", inv_c, inv_d);

    return 0;
}
```

Python and C Code

```python

import ctypes
import numpy as np
import matplotlib.pyplot as plt

# Load compiled C library
# For Linux/Mac
lib = ctypes.CDLL('./libmatrix_inverse.so')
# For Windows:
# lib = ctypes.CDLL('matrix_inverse.dll')

# Define argument types
lib.inverse2x2.argtypes = [ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)]
```

Python and C Code

```python

# Input matrix A = [[2, -3], [-1, 2]]
A = np.array([2.0, -3.0, -1.0, 2.0], dtype=np.float64)
A_inv = np.zeros(4, dtype=np.float64)

# Call C function to get inverse
lib.inverse2x2(A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
               A_inv.ctypes.data_as(ctypes.POINTER(ctypes.c_double
                   )))

# Reshape to 2x2 matrices
A_matrix = A.reshape(2, 2)
A_inv_matrix = A_inv.reshape(2, 2)
```

Python and C Code

```
print("Original matrix A:")
print(A_matrix)
print("Inverse matrix A (from C):")
print(A_inv_matrix)

# Define the unit square
square = np.array([
    [0, 0],
    [1, 0],
    [1, 1],
    [0, 1],
    [0, 0]
])
```

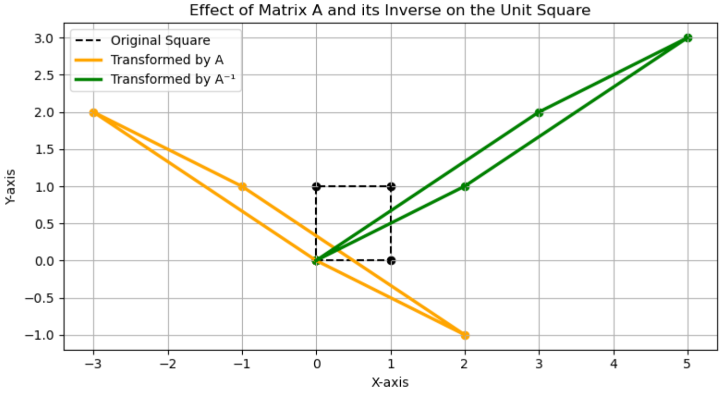
Python and C Code

```python

# Apply transformations
square_A = square @ A_matrix.T
square_Ainv = square @ A_inv_matrix.T

# Plot
plt.figure(figsize=(8, 6))
plt.plot(square[:, 0], square[:, 1], 'k--', linewidth=1.5, label=
    'Original Square') # Black dashed
plt.plot(square_A[:, 0], square_A[:, 1], color='orange',
    linewidth=2.5, label='Transformed by A') # Orange
plt.plot(square_Ainv[:, 0], square_Ainv[:, 1], color='green',
    linewidth=2.5, label='Transformed by A') # Green

# Optional: add markers to show square corners
plt.scatter(square[:, 0], square[:, 1], color='black')
plt.scatter(square_A[:, 0], square_A[:, 1], color='orange')
plt.scatter(square_Ainv[:, 0], square_Ainv[:, 1], color='green')
```

Python and C Code

```
plt.grid(True)
plt.gca().set_aspect('equal', adjustable='box')
plt.title('Effect of Matrix A and its Inverse on the Unit Square'
    )
plt.legend()
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.tight_layout()
plt.show()
```

Plot-Using by C and Python


Effect of Matrix A and its Inverse on the Unit Square
Original Square
Transformed by A
Transformed by A⁻¹
X-axis
Y-axis