5.5.1

Sai Krishna Bakki - EE25BTECH11049

Question

If $\mathbf{A} = \begin{pmatrix} 5 & -1 & 4 \\ 2 & 3 & 5 \\ 5 & -2 & 6 \end{pmatrix}$, find $\mathbf{A}^{-1}$ and use it to solve the following system of equations

$$\begin{aligned} 5x - y + 4z &= 5 \\ 2x + 3y + 5z &= 2 \\ 5x - 2y + 6z &= -1 \end{aligned}$$

Theoretical Solution

$$\left(\begin{array}{ccc|ccc} 5 & -1 & 4 & 1 & 0 & 0 \\ 2 & 3 & 5 & 0 & 1 & 0 \\ 5 & -2 & 6 & 0 & 0 & 1 \end{array}\right) R_3 \leftarrow R_3 - R_1 \left(\begin{array}{ccc|ccc} 5 & -1 & 4 & 1 & 0 & 0 \\ 2 & 3 & 5 & 0 & 1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 1 \end{array}\right) \tag{1}$$

$$[\, R_2 \leftarrow R_2 + 3R_3] \, R_1 \leftarrow R_1 - R_3 \left(\begin{array}{ccc|ccc} 5 & 0 & 2 & 2 & 0 & -1 \\ 2 & 0 & 11 & -3 & 1 & 3 \\ 0 & -1 & 2 & -1 & 0 & 1 \end{array}\right) \tag{2}$$

$$R_3 \leftarrow -R_3 \left(\begin{array}{ccc|ccc} 5 & 0 & 2 & 2 & 0 & -1 \\ 2 & 0 & 11 & -3 & 1 & 3 \\ 0 & 1 & -2 & 1 & 0 & -1 \end{array}\right) \tag{3}$$

$$R_2 \leftrightarrow R_3 \left(\begin{array}{ccc|ccc} 5 & 0 & 2 & 2 & 0 & -1 \\ 0 & 1 & -2 & 1 & 0 & -1 \\ 2 & 0 & 11 & -3 & 1 & 3 \end{array}\right) \tag{4}$$

$$R_1 \leftarrow \frac{1}{5} R_1 \left(\begin{array}{ccc|ccc} 1 & 0 & 2/5 & 2/5 & 0 & -1/5 \\ 0 & 1 & -2 & 1 & 0 & -1 \\ 2 & 0 & 11 & -3 & 1 & 3 \end{array}\right) \tag{5}$$

$$R_3 \leftarrow R_3 - 2R_1 \left(\begin{array}{ccc|ccc} 1 & 0 & 2/5 & 2/5 & 0 & -1/5 \\ 0 & 1 & -2 & 1 & 0 & -1 \\ 0 & 0 & 51/5 & -19/5 & 1 & 17/5 \end{array}\right) \tag{6}$$

Theoretical Solution

$$R_3 \leftarrow \frac{5}{51} R_3 \left(\begin{array}{ccc|ccc} 1 & 0 & 2/5 & 2/5 & 0 & -1/5 \\ 0 & 1 & -2 & 1 & 0 & -1 \\ 0 & 0 & 1 & -19/51 & 5/51 & 17/51 \end{array}\right) \tag{7}$$

$$[\, R_2 \leftarrow R_2 + 2R_3]\, R_1 \leftarrow R_1 - \frac{2}{5} R_3 \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 28/51 & -2/51 & -17/51 \\ 0 & 1 & 0 & 13/51 & 10/51 & -17/51 \\ 0 & 0 & 1 & -19/51 & 5/51 & 17/51 \end{array}\right) \tag{8}$$

$$\therefore \mathbf{A}^{-1} = \begin{pmatrix} 28/51 & -2/51 & -17/51 \\ 13/51 & 10/51 & -17/51 \\ -19/51 & 5/51 & 17/51 \end{pmatrix} \tag{9}$$

Theoretical Solution

Now, Finding system of equations

$$\mathbf{AX} = \mathbf{C} \tag{10}$$

where $\mathbf{C} = \begin{pmatrix} 5 \\ 2 \\ -1 \end{pmatrix}$ and $\mathbf{X} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{C} \tag{11}$$

$$\mathbf{X} = \begin{pmatrix} 28/51 & -2/51 & -17/51 \\ 13/51 & 10/51 & -17/51 \\ -19/51 & 5/51 & 17/51 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \end{pmatrix} \tag{12}$$

$$\therefore \mathbf{X} = \begin{pmatrix} 3 \\ 2 \\ -2 \end{pmatrix} \tag{13}$$

C Code

```c
#include <stdio.h>

// This function will be called from Python.
// It takes three integer values as arguments.
void process_point(int x, int y, int z) {
    // Print a confirmation message to the console.
    printf( C function received point: (%d, %d, %d)\n, x, y, z);
}
```

Python Through Shared Output

```
import ctypes
import subprocess
import os
import platform
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

# --- Part 1: Calculate the Solution using NumPy ---

# Define the coefficient matrix A and the constant vector B
A = np.array([
    [5, -1, 4],
    [2, 3, 5],
    [5, -2, 6]
])
B = np.array([5, 2, -1])

# Use numpy.linalg.solve() to find the intersection point
```

```
try:
    solution = np.linalg.solve(A, B)
    x_int, y_int, z_int = solution
    print(f Python calculated intersection point: ({x_int:.0f}, {
        y_int:.0f}, {z_int:.0f}))
except np.linalg.LinAlgError:
    print(The system of equations has no unique solution.)
    exit()


# --- Part 2: Compile and Call C Function using ctypes ---

c_source_file = points.c
if platform.system() == Windows:
    lib_file = points.dll
else:
    lib_file = points.so
```

Python Through Shared Output

```
try:
    # Compile the C code into a shared library
    subprocess.run([gcc, -shared, -o, lib_file, -fPIC,
        c_source_file], check=True)

    # Load the shared library
    c_lib = ctypes.CDLL(os.path.join(os.getcwd(), lib_file))

    # Define the argument types for the C function (three
        integers)
    c_lib.process_point.argtypes = [ctypes.c_int, ctypes.c_int,
        ctypes.c_int]
    c_lib.process_point.restype = None

    # Call the C function, passing the values calculated by NumPy
    # We convert them to standard Python integers first
    c_lib.process_point(int(x_int), int(y_int), int(z_int))
```

Python Through Shared Output

```
except (Exception) as e:
    print(fAn error occurred during C interaction: {e})
    exit()


# --- Part 3: Generate the 3D Plot ---

def plane1(x, y): return (5 - 5*x + y) / 4
def plane2(x, y): return (2 - 2*x - 3*y) / 5
def plane3(x, y): return (-1 - 5*x + 2*y) / 6

x_grid, y_grid = np.meshgrid(np.linspace(-2, 8, 20), np.linspace(-2, 8, 20))
z1 = plane1(x_grid, y_grid)
z2 = plane2(x_grid, y_grid)
z3 = plane3(x_grid, y_grid)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.plot_surface(x_grid, y_grid, z1, alpha=0.6, color='red')
ax.plot_surface(x_grid, y_grid, z2, alpha=0.6, color='green')
ax.plot_surface(x_grid, y_grid, z3, alpha=0.6, color='blue')
ax.scatter(x_int, y_int, z_int, color='black', s=150, ec='white',
     zorder=10)

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Planes Intersecting at a Point Calculated by Python
    ')

legend_elements = [
    Line2D([0], [0], color='red', lw=4, label='5x - y + 4z = 5'),
    Line2D([0], [0], color='green', lw=4, label='2x + 3y + 5z = 2
        '),
    Line2D([0], [0], color='blue', lw=4, label='5x - 2y + 6z = -1
        '),
```

Python Through Shared Output

```
    Line2D([0], [0], marker='o', color='w', markerfacecolor='black', markersize=10, label=f'Intersection: ({x_int:.0f}, {y_int:.0f}, {z_int:.0f})')
]
ax.legend(handles=legend_elements)

plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

# --- Calculations based on the image ---

# Define the matrices A and B from the system of equations AX = B
A = np.array([
    [5, -1, 4],
    [2, 3, 5],
    [5, -2, 6]
])

B = np.array([5, 2, -1])

# Calculate the inverse of A and the solution X
# X = [x, y, z]
try:
    A_inv = np.linalg.inv(A)
```

Python Code

```
    X_solution = A_inv @ B
    x_int, y_int, z_int = X_solution
    print(fThe inverse of A is:\n{A_inv}\n)
    print(fThe solution is x={x_int}, y={y_int}, z={z_int})
except np.linalg.LinAlgError:
    print(Matrix A is singular and does not have an inverse.)
    exit()

# --- Visualization ---

# Define the plane equations by solving for z
# 5x - y + 4z = 5 => z = (5 - 5x + y) / 4
def plane1(x, y):
    return (5 - 5*x + y) / 4

# 2x + 3y + 5z = 2 => z = (2 - 2x - 3y) / 5
def plane2(x, y):
    return (2 - 2*x - 3*y) / 5
```

Python Code

```
# 5x - 2y + 6z = -1 => z = (-1 - 5x + 2y) / 6
def plane3(x, y):
    return (-1 - 5*x + 2*y) / 6

# Create a grid for plotting
x = np.linspace(-2, 8, 50)
y = np.linspace(-2, 8, 50)
X_grid, Y_grid = np.meshgrid(x, y)

Z1 = plane1(X_grid, Y_grid)
Z2 = plane2(X_grid, Y_grid)
Z3 = plane3(X_grid, Y_grid)

# Plotting
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot the three planes
ax.plot_surface(X_grid, Y_grid, Z1, alpha=0.5, color='red')
```

Python Code

```
ax.plot_surface(X_grid, Y_grid, Z2, alpha=0.5, color='green')
ax.plot_surface(X_grid, Y_grid, Z3, alpha=0.5, color='blue')

# Plot the calculated intersection point
ax.scatter(x_int, y_int, z_int, color='black', s=150, label='
    Intersection Point', depthshade=False, zorder=10)
# Labels and title
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Intersection of Three Planes')

# Create a custom legend for the planes and point
legend_elements = [
    Line2D([0], [0], color='red', lw=4, label='5x - y + 4z = 5'),
    Line2D([0], [0], color='green', lw=4, label='2x + 3y + 5z = 2
        '),
```

Python Code

```
    Line2D([0], [0], color='blue', lw=4, label='5x - 2y + 6z = -1'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='black', markersize=10, label=f'Intersection: ({x_int:.0f}, {y_int:.0f}, {z_int:.0f})')
]
ax.legend(handles=legend_elements)

plt.show()
```

Plot By C code and Python Code

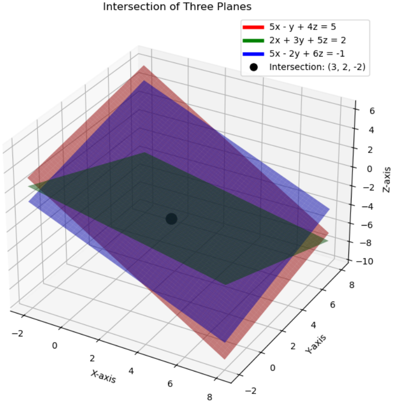

Figure: