12.264

EE25BTECH11043 - Nishid Khandagre

October 8, 2025

Question

Consider the system of equations

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 0 \\ x_2 - x_3 &= 0 \\ x_1 + x_2 &= 0 \end{aligned}$$

This system has

- a) a unique solution
- b) no solution
- c) infinite number of solutions
- d) five solutions

Solution

We can represent the system of equations in matrix form as

$$\mathbf{A}\mathbf{x} = \mathbf{0} \tag{1}$$

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & -1 \\ 1 & 1 & 0 \end{pmatrix} \tag{2}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{3}$$

Solution

Augmented matrix

$$\left(\begin{array}{ccc|c} 2 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 1 & 0 & 0 \end{array}\right) \tag{4}$$

Apply the operation $R_1 \to \frac{1}{2}R_1$:

$$\left(\begin{array}{ccc|c} 1 & 0.5 & 0.5 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 1 & 0 & 0 \end{array}\right) \tag{5}$$

Solution

Apply the operation $R_1 \to \frac{1}{2}R_1$:

$$\left(\begin{array}{ccc|c} 1 & 0.5 & 0.5 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 1 & 0 & 0 \end{array}\right) \tag{6}$$

Then $R_3 \to R_3 - R_1$:

$$\left(\begin{array}{ccc|c} 1 & 0.5 & 0.5 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0.5 & -0.5 & 0 \end{array}\right) \tag{7}$$

Solution

Perform the operation $R_3 \to R_3 - 0.5R_2$:

$$\left(\begin{array}{ccc|c} 1 & 0.5 & 0.5 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right) \tag{8}$$

Solution

From the row echelon form, we can write the new system of equations:

$$x_1 + 0.5x_2 + 0.5x_3 = 0 \tag{9}$$
$$x_2 - x_3 = 0 \tag{10}$$

From the second equation, we get:

$$x_2 = x_3 \tag{11}$$

Substitute $x_2 = x_3$ into the first equation:

$$x_1 + 0.5x_2 + 0.5x_2 = 0 \tag{12}$$
$$x_1 + x_2 = 0 \tag{13}$$
$$x_1 = -x_2 \tag{14}$$

Solution

Let $x_2 = t$, where t is any real number. Then, we have:

$$x_3 = t \tag{15}$$

$$x_1 = -t \tag{16}$$

So, the solution vector is:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = t \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \quad t \in \mathbb{R} \tag{17}$$

Since there is one free parameter (t), the system has infinitely many solutions. This is also indicated by the rank of matrix A being less than 3 (rank is 2), and the system is consistent (homogeneous systems are always consistent). Therefore, the answer is option (c).

C Code

```c
#include <stdio.h>

// Function to calculate the determinant of a 3x3 matrix
// The matrix is passed as a 1D array in row-major order:
// [a11, a12, a13, a21, a22, a23, a31, a32, a33]
double calculate_determinant_3x3(double* matrix) {
    double det = 0.0;

    det = matrix[0] * (matrix[4] * matrix[8] - matrix[5] * matrix[7])
        - matrix[1] * (matrix[3] * matrix[8] - matrix[5] * matrix[6])
        + matrix[2] * (matrix[3] * matrix[7] - matrix[4] * matrix[6]);

    return det;
}
```

Python Code using Shared C Library

```
import ctypes
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D

# Load the shared library
lib_solver = ctypes.CDLL(./code21.so)

# Define the argument types and return type for the C function
lib_solver.calculate_determinant_3x3.argtypes = [
    ctypes.POINTER(ctypes.c_double)
]
lib_solver.calculate_determinant_3x3.restype = ctypes.c_double
```

Python Code using Shared C Library

```
# Define the coefficient matrix for the system of equations:
# Coefficient matrix A:
# [2 1 1]
# [0 1 -1]
# [1 1 0]

coefficient_matrix_np = np.array([
    2.0, 1.0, 1.0,
    0.0, 1.0, -1.0,
    1.0, 1.0, 0.0
], dtype=np.float64)
matrix_reshaped = coefficient_matrix_np.reshape(3,3)
print(Coefficient Matrix:)
print(matrix_reshaped)
# Create a ctypes array from the numpy array for the C function
matrix_c = (ctypes.c_double * len(coefficient_matrix_np))(*
    coefficient_matrix_np)
```

Python Code using Shared C Library

```
# Call the C function to calculate the determinant
determinant = lib_solver.calculate_determinant_3x3(matrix_c)

print(f\nCalculated Determinant: {determinant:.4f})

# Determine the nature of the solutions
if abs(determinant) > 1e-9:
    solution_type = a) a unique solution (trivial solution)
    plot_title_suffix = Unique Solution (Intersection at Origin)
else:
    solution_type = c) infinite number of solutions
    plot_title_suffix = Infinite Solutions (Intersection along a
        Line)

print(f\nFor this homogeneous system, the conclusion is:\n{
    solution_type})
```

Python Code using Shared C Library

```
# --- 3D Plotting of Planes ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
# Define a range for x, y, z
r = 3
x = np.linspace(-r, r, 10)
y = np.linspace(-r, r, 10)
X, Y = np.meshgrid(x, y)

# Equation 1: 2x1 + x2 + x3 = 0 => x3 = -2x1 - x2
Z1 = -2*X - Y
ax.plot_surface(X, Y, Z1, alpha=0.5, color='cyan', label='2x1 +
    x2 + x3 = 0')

# Equation 2: x2 - x3 = 0 => x3 = x2
Z2 = Y
ax.plot_surface(X, Y, Z2, alpha=0.5, color='magenta', label='x2 -
     x3 = 0')
```

Python Code using Shared C Library

```
# Equation 3: x1 + x2 = 0 => x2 = -x1
X3_plot, Z3_plot = np.meshgrid(x, np.linspace(-r, r, 10))
Y3_plot = -X3_plot
ax.plot_surface(X3_plot, Y3_plot, Z3_plot, alpha=0.5, color='
    yellow', label='x1 + x2 = 0')

# Find the intersection line: (-t, t, t)
t = np.linspace(-r, r, 100)
intersection_line_x = -t
intersection_line_y = t
intersection_line_z = t
ax.plot(intersection_line_x, intersection_line_y,
    intersection_line_z,
        color='red', linewidth=3, label='Intersection Line (-t, t
            , t)')
# Add a point at the origin (trivial solution)
ax.scatter(0, 0, 0, color='red', s=100, label='Origin (0,0,0)',
    zorder=5)
```

Python Code using Shared C Library

```
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title(fPlanes for System of Equations: {plot_title_suffix}\n(Det = {determinant:.0f}))
custom_lines = [
    Line2D([0], [0], color='cyan', lw=4, alpha=0.5),
    Line2D([0], [0], color='magenta', lw=4, alpha=0.5),
    Line2D([0], [0], color='yellow', lw=4, alpha=0.5),
    Line2D([0], [0], color='red', lw=3)
]
ax.legend(custom_lines,
          ['2x1 + x2 + x3 = 0', 'x2 - x3 = 0', 'x1 + x2 = 0', 'Intersection Line'],
          loc='upper left', bbox_to_anchor=(0.8, 0.95))

plt.tight_layout()
plt.show()
```

Pure Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D

def calculate_determinant_3x3_python(matrix):
    if matrix.shape != (3, 3):
        raise ValueError(Input matrix must be 3x3.)
    a, b, c = matrix[0, 0], matrix[0, 1], matrix[0, 2]
    d, e, f = matrix[1, 0], matrix[1, 1], matrix[1, 2]
    g, h, i = matrix[2, 0], matrix[2, 1], matrix[2, 2]
    det = a * (e * i - f * h) - b * (d * i - f * g) + c * (d * h
         - e * g)
    return det
```

Pure Python Code

```
coefficient_matrix_np = np.array([
    [2.0, 1.0, 1.0],
    [0.0, 1.0, -1.0],
    [1.0, 1.0, 0.0]
], dtype=np.float64)
print(Coefficient Matrix:)
print(coefficient_matrix_np)
determinant = calculate_determinant_3x3_python(
    coefficient_matrix_np)
if abs(determinant) > 1e-9:
    solution_type = a) a unique solution (trivial solution)
    plot_title_suffix = Unique Solution (Intersection at Origin)
    has_intersection_line = False
else:
    solution_type = c) infinite number of solutions
    plot_title_suffix = Infinite Solutions (Intersection along a
        Line)
    has_intersection_line = True
```

Pure Python Code

```
print(f\nFor this homogeneous system, the conclusion is:\n{solution_type})

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

r_range = 3
x_vals = np.linspace(-r_range, r_range, 10)
y_vals = np.linspace(-r_range, r_range, 10)
X, Y = np.meshgrid(x_vals, y_vals)

Z1 = -2*X - Y
ax.plot_surface(X, Y, Z1, alpha=0.5, color='cyan', label='2x1 + x2 + x3 = 0')

Z2 = Y
ax.plot_surface(X, Y, Z2, alpha=0.5, color='magenta', label='x2 - x3 = 0')
```

Pure Python Code

```
X3_plot, Z3_plot = np.meshgrid(x_vals, np.linspace(-r_range,
    r_range, 10))
Y3_plot = -X3_plot
ax.plot_surface(X3_plot, Y3_plot, Z3_plot, alpha=0.5, color='
    yellow', label='x1 + x2 = 0')

if has_intersection_line:
    t = np.linspace(-r_range, r_range, 100)
    intersection_line_x = -t
    intersection_line_y = t
    intersection_line_z = t
    ax.plot(intersection_line_x, intersection_line_y,
        intersection_line_z,
            color='red', linewidth=3, label='Intersection Line (-t
                , t, t)')

ax.scatter(0, 0, 0, color='red', s=100, label='Origin (0,0,0)',
    zorder=5)
```

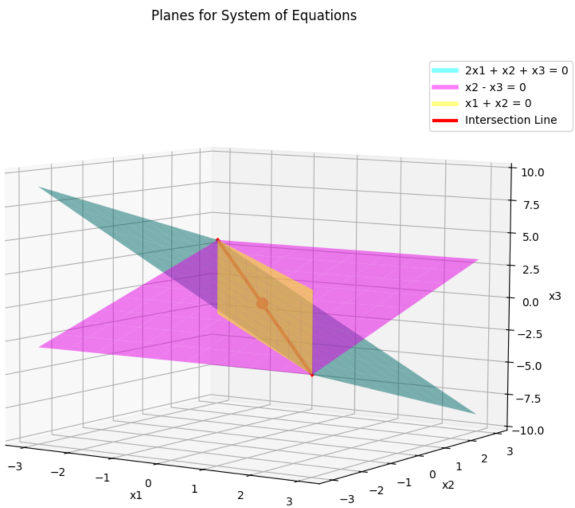
Pure Python Code

```
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title(fPlanes for System of Equations)

custom_lines = [
    Line2D([0], [0], color='cyan', lw=4, alpha=0.5),
    Line2D([0], [0], color='magenta', lw=4, alpha=0.5),
    Line2D([0], [0], color='yellow', lw=4, alpha=0.5),
    Line2D([0], [0], color='red', lw=3)
]
ax.legend(custom_lines,
          ['2x1 + x2 + x3 = 0', 'x2 - x3 = 0', 'x1 + x2 = 0', '
              Intersection Line'],
          loc='upper left', bbox_to_anchor=(0.8, 0.95))

plt.tight_layout()
plt.show()
```

Plot by Python using shared output from C



Plot by Python only