5.2.62

Namaswi-EE25BTECH11060

september 2025

Question

Solve system of linear equations

$$\begin{aligned} 3x - 2y + 3z &= 8 \\ 2x + y - z &= 1 \\ 4x - 3y + 2z &= 4 \end{aligned}$$

Solution

According to question the Equations of line given are

$$\begin{pmatrix} 3 & -2 & 3 \end{pmatrix} X = 8 \tag{1}$$

$$\begin{pmatrix} 2 & 1 & -2 \end{pmatrix} X = 1 \tag{2}$$

$$\begin{pmatrix} 4 & -3 & 2 \end{pmatrix} X = 4 \tag{3}$$

$$\begin{pmatrix} 3 & -2 & 3 \\ 2 & 1 & -1 \\ 4 & -3 & 2 \end{pmatrix} X = \begin{pmatrix} 8 \\ 1 \\ 4 \end{pmatrix} \tag{4}$$

Solution

Forming Argumented Matrix

$$\left(\begin{array}{ccc|c} 3 & -2 & 3 & 8 \\ 2 & 1 & -1 & 1 \\ 4 & -3 & 2 & 4 \end{array}\right). \tag{5}$$

Replace

$$R_1 \to \frac{1}{3}R_3$$

$$\begin{pmatrix} 1 & -\frac{2}{3} & 1 & \frac{8}{3} \\ 2 & 1 & -1 & 1 \\ 4 & -3 & 2 & 4 \end{pmatrix} \tag{6}$$

Solution

Replace

$$R_2 \to R_2 - 2R_1, \quad R_3 \to R_3 - 4R_1$$

$$\left(\begin{array}{ccc|c} 1 & -\frac{2}{3} & 1 & \frac{8}{3} \\ 0 & \frac{7}{3} & -3 & -\frac{13}{3} \\ 0 & -\frac{1}{3} & -2 & -\frac{20}{3} \end{array}\right) \tag{7}$$

Replace

$$R_2 \to \frac{3}{7} R_2$$

$$\left(\begin{array}{ccc|c} 1 & -\frac{2}{3} & 1 & \frac{8}{3} \\ 0 & 1 & -\frac{9}{7} & -\frac{13}{7} \\ 0 & -\frac{1}{3} & -2 & -\frac{20}{3} \end{array}\right) \tag{8}$$

Solution

Replace

$$R_1 \to R_1 + \tfrac{2}{3}R_2, \quad R_3 \to R_3 + \tfrac{1}{3}R_2$$

$$\left(\begin{array}{ccc|c} 1 & 0 & \frac{5}{21} & \frac{22}{21} \\ 0 & 1 & -\frac{9}{7} & -\frac{13}{7} \\ 0 & 0 & -\frac{41}{21} & -\frac{143}{21} \end{array}\right) \tag{9}$$

Replace

$$R_3 \to -\tfrac{21}{41}R_3$$

$$\left(\begin{array}{ccc|c} 1 & 0 & \frac{5}{21} & \frac{22}{21} \\ 0 & 1 & -\frac{9}{7} & -\frac{13}{7} \\ 0 & 0 & 1 & \frac{143}{41} \end{array}\right) \tag{10}$$

Solution

Replace

$$R_1 \to R_1 - \frac{5}{21}R_3, \quad R_2 \to R_2 + \frac{9}{7}R_3$$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & \frac{62}{123} \\ 0 & 1 & 0 & \frac{110}{287} \\ 0 & 0 & 1 & \frac{143}{41} \end{array}\right) \tag{11}$$

Hence,

$$\mathbf{x} = \begin{pmatrix} \frac{62}{123} \\ \frac{110}{287} \\ \frac{143}{41} \end{pmatrix} \tag{12}$$

C Code

```c
   #include <stdio.h>
int main() {
    int n = 3;
    double a[3][4] = {
        {3, -2, 3, 8},
        {2, 1, -1, 1},
        {4, -3, 2, 4}
    };
    // Forward elimination
    for (int i = 0; i < n; i++) {
        // Normalize row
        double pivot = a[i][i];
        for (int j = i; j <= n; j++)
            a[i][j] /= pivot;
```

C Code

```
    // Eliminate column
        for (int k = 0; k < n; k++) {
            if (k != i) {
                double factor = a[k][i];
                for (int j = i; j <= n; j++)
                    a[k][j] -= factor * a[i][j];
            }
        }
    }

    printf("Solution:\n");
    for (int i = 0; i < n; i++) {
        printf("x%d = %lf\n", i+1, a[i][n]);
    }

    return 0;
}
```

Python Code

```
    import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # registers 3D projection
from matplotlib.patches import Patch

# Plane coefficients: a*x + b*y + c*z = d
planes = [
    (3, -2, 3, 8, "3x - 2y + 3z = 8"),
    (2, 1, -1, 1, "2x + y - z = 1"),
    (4, -3, 2, 4, "4x - 3y + 2z = 4")
]
colors = ["tab:blue", "tab:orange", "tab:green"]

# Solve linear system to find intersection point (if unique)
A = np.array([[p[0], p[1], p[2]] for p in planes], dtype=float)
b = np.array([p[3] for p in planes], dtype=float)
```

Python Code

```
    intersection = None
try:
    intersection = np.linalg.solve(A, b) # [x0, y0, z0]
    has_unique = True
except np.linalg.LinAlgError:
    has_unique = False

# Build a grid in x-y around the intersection (or default range)
if has_unique:
    x0, y0, z0 = intersection
    rng = 3.0
    x_min, x_max = x0 - rng, x0 + rng
    y_min, y_max = y0 - rng, y0 + rng
else:
    x_min, x_max = -5, 5
    y_min, y_max = -5, 5
```

Python Code

```
        xx, yy = np.meshgrid(np.linspace(x_min, x_max, 40),
                             np.linspace(y_min, y_max, 40))

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

# Plot each plane
patches = []
for (a, b_coef, c, d, label), col in zip(planes, colors):
    # compute z = (d - a*x - b*y)/c (works since c != 0 for these
        planes)
    zz = (d - a * xx - b_coef * yy) / c
    surf = ax.plot_surface(xx, yy, zz, alpha=0.5, rstride=1,
        cstride=1, linewidth=0, antialiased=True)
    patches.append(Patch(facecolor=col, label=label))
    # Color the surface by setting the facecolors (plot_surface
        doesn't accept label directly)
    surf.set_facecolor(col)
    surf.set_edgecolor((0,0,0,0)) # hide edges for smooth look
```

Python Code

```
        # Plot intersection point if exists
if has_unique:
    ax.scatter([x0], [y0], [z0], color="red", s=60, label="
        Intersection point")
    ax.text(x0, y0, z0, f" ({x0:.2f}, {y0:.2f}, {z0:.2f})", color
        ="red")

# Labels, limits and legend
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_xlim(x_min, x_max)
ax.set_ylim(y_min, y_max)
```

Python Code

```
    # Create legend: include plane labels and intersection point
        if present
legend_handles = [Patch(facecolor=colors[i], label=planes[i][4])
    for i in range(len(planes))]
if has_unique:
    legend_handles.append(plt.Line2D([0], [0], marker='o', color=
        'w',
                                     markerfacecolor='red',
                                        markersize=8, label='
                                        Intersection point'))
ax.legend(handles=legend_handles, loc='upper left',
    bbox_to_anchor=(1.05, 1))

ax.set_title("3 Planes: 3x-2y+3z=8, 2x+y-z=1, 4x-3y+2z=4")
plt.tight_layout()
plt.show()
```

C and Python Code

```
      import ctypes
import numpy as np
# Load the shared library
gauss = ctypes.CDLL("./gauss.so")
# Define argument types
gauss.gauss.argtypes = [((ctypes.c_double * 4) * 3), ctypes.
    POINTER(ctypes.c_double)]
# Input augmented matrix
A = np.array([
    [3, -2, 3, 8],
    [2, 1, -1, 1],
    [4, -3, 2, 4]
], dtype=np.float64)
```

C and Python Code

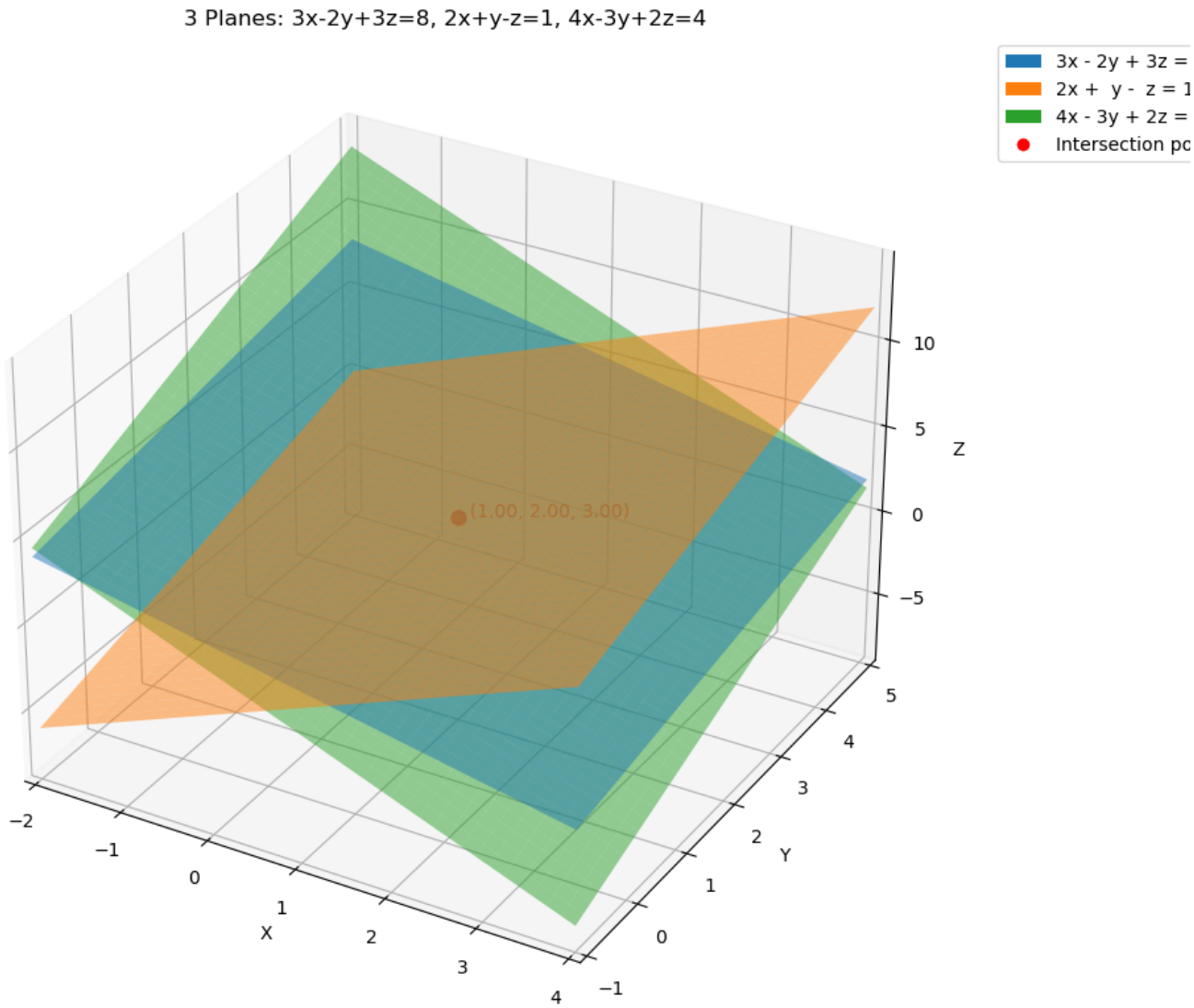
```
        # Convert numpy to C array
a_c = ((ctypes.c_double * 4) * 3)(*([tuple(row) for row in A]))

# Allocate solution vector
sol = (ctypes.c_double * 3)()

# Call C function
gauss.gauss(a_c, sol)

# Print solution
print("Solution from C function:")
print("x =", sol[0])
print("y =", sol[1])
print("z =", sol[2])
```

plot


3 Planes: 3x-2y+3z=8, 2x+y-z=1, 4x-3y+2z=4
3x - 2y + 3z =
2x + y - z = 1
4x - 3y + 2z =
Intersection pc
(1.00, 2.00, 3.00)
X
Y
Z