4.11.18

Naman Kumar-EE25BTECH11041

14 September,2025

Question

Find the equation of the plane which contains the line of intersection of the planes $\mathbf{r} \cdot (\imath - 2\jmath + 3\hat{k}) - 4 = 0$ and $\mathbf{r} \cdot (-2\imath + \jmath + \hat{k}) + 5 = 0$ and whose intercept on X axis is equal to that of on Y axis.

Solution

Given Planes,

$$n_1^T x = c_1, n_2^T x = c_2 \tag{1}$$

Where

$$n_1 = \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}, n_2 = \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix}, c_1 = 4, c_2 = -5 \tag{2}$$

Let Required equation of plane

$$n_3^T x = c_3 \tag{3}$$

Solution

Since we can write,

$$P_3 = P_1 - \lambda P_2 \text{ (Where } P_1, P_2, P_3 \text{ are equation of planes)} \tag{4}$$

Because All three planes intersect at same line,Therefore

$$(n_1 - \lambda n_2)^T x = c_1 - \lambda c_2 \tag{5}$$

$$\tag{6}$$

Given,

$$X - intercept = Y - intercept \tag{7}$$

$$\tag{8}$$

Solution

for X-intercept

$$(n_1 - \lambda n_2)^T \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix} = c_1 - \lambda c_2 \tag{9}$$

$$(n_1 - \lambda n_2)^T x e_1 = c_1 - \lambda c_2 \tag{10}$$

Therefore,

$$X - intercept = \frac{c_1 - \lambda c_2}{(n_1 - \lambda n_2)^T e_1} \tag{11}$$

Solution

Similarly

$$Y - intercept = \frac{c_1 - \lambda c_2}{(n_1 - \lambda n_2)^T e_2} \tag{12}$$

Comparing equations (11) and (12)

$$\frac{c_1 - \lambda c_2}{(n_1 - \lambda n_2)^T e_1} = \frac{c_1 - \lambda c_2}{(n_1 - \lambda n_2)^T e_2} \tag{13}$$

$$(n_1 - \lambda n_2)^T e_1 = (n_1 - \lambda n_2)^T e_2 \tag{14}$$

$$\begin{pmatrix} 1 + 2\lambda \\ -2 - 1\lambda \\ 3 - 1\lambda \end{pmatrix}^T e_1 = \begin{pmatrix} 1 + 2\lambda \\ -2 - 1\lambda \\ 3 - 1\lambda \end{pmatrix}^T e_2 \tag{15}$$

$$1 + 2\lambda = -2 - 1\lambda \tag{16}$$

$$\lambda = -1 \tag{17}$$

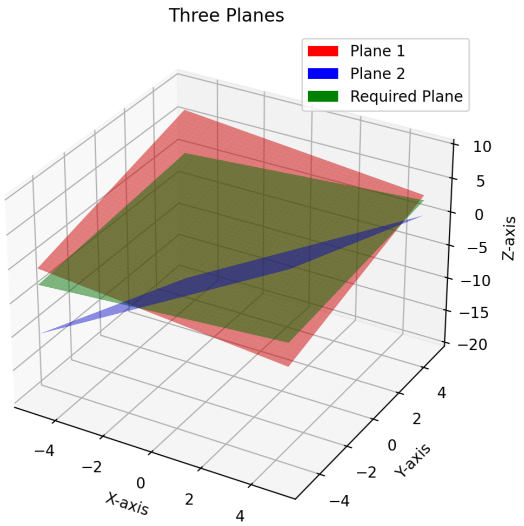
Solution

Therefore equation of required plane is

$$\begin{pmatrix} 1+2(-1) \\ -2-1(-1) \\ 3-1(-1) \end{pmatrix}^T x = 4+5(-1) \tag{18}$$

$$\begin{pmatrix} -1 \\ -1 \\ 4 \end{pmatrix}^T x = -1 \tag{19}$$

Figure


Three Planes
Plane 1
Plane 2
Required Plane
10
5
0
−5
−10
−15
−20
Z-axis
4
2
0
−2
−4
Y-axis
−4
−2
0
2
4
X-axis

C code

```c
#include <stdio.h>

// Function to compute coefficients of required plane
// Plane form: Ax + By + Cz + D = 0
// Returns coefficients via array coeff[4]
void find_plane(double coeff[4]) {
    // Plane 1: x - 2y + 3z - 4 = 0
    double A1 = 1, B1 = -2, C1 = 3, D1 = -4;

    // Plane 2: -2x + y + z + 5 = 0
    double A2 = -2, B2 = 1, C2 = 1, D2 = 5;

    // Required plane = pi1 + *pi2
    // (A1 + A2)x + (B1 + B2)y + (C1 + C2)z + (D1 + D2) = 0
```

C code

```c
    // Condition: intercept on X = intercept on Y  A = B
    // So: A1 + A2 = B1 + B2
    double lambda = (B1 - A1) / (A2 - B2);

    double A = A1 + lambda * A2;
    double B = B1 + lambda * B2;
    double C = C1 + lambda * C2;
    double D = D1 + lambda * D2;

    coeff[0] = A;
    coeff[1] = B;
    coeff[2] = C;
    coeff[3] = D;
}
```

Python code with SO file

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

# Load shared C library
lib = ctypes.CDLL('./libplane.so')

# Define argument/return types
lib.find_plane.argtypes = [np.ctypeslib.ndpointer(dtype=np.
    float64, ndim=1, flags="C")]

# Prepare coeff array
coeff = np.zeros(4, dtype=np.float64)
lib.find_plane(coeff)
```

Python code with SO file

```python
A, B, C, D = coeff
print("Required plane equation: {:.2f}x + {:.2f}y + {:.2f}z +
    {:.2f} = 0".format(A, B, C, D))

# Define planes
def plane1(x, y):
    return (4 - x + 2*y) / 3

def plane2(x, y):
    return (-5 + 2*x - y)

def plane3(x, y):
    return (-D - A*x - B*y) / C
```

Python code with SO file

```
# Grid
x = np.linspace(-5, 5, 20)
y = np.linspace(-5, 5, 20)
X, Y = np.meshgrid(x, y)

Z1 = plane1(X, Y)
Z2 = plane2(X, Y)
Z3 = plane3(X, Y)
```

Python code with SO file

```python
# Plot
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

ax.plot_surface(X, Y, Z1, alpha=0.5, color='red', label="Plane 1"
    )
ax.plot_surface(X, Y, Z2, alpha=0.5, color='blue', label="Plane 2
    ")
ax.plot_surface(X, Y, Z3, alpha=0.7, color='green', label="
    Required Plane")
```

Python code with SO file

```
# Legend hack
plane_proxy = [plt.Rectangle((0,0),1,1,fc=c) for c in ["red","
    blue","green"]]
ax.legend(plane_proxy, ["Plane 1","Plane 2","Required Plane"])

ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("Intersection of Three Planes")
plt.savefig("figure.png", dpi=200)
plt.show()
```

Direct python code

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

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')

# Plane 1
x1 = np.linspace(-5,5,100)
y1 = np.linspace(-5,5,100)
X1, Y1 = np.meshgrid(x1,y1)
Z1 = (-X1 + 2*Y1 + 4) / 3
```

Direct python code

```
# Plane 2
x2 = np.linspace(-5,5,100)
y2 = np.linspace(-5,5,100)
X2, Y2 = np.meshgrid(x2,y2)
Z2 = -Y2 + 2*X2 - 5

# Plane 3 (required plane)
x3 = np.linspace(-5,5,100)
y3 = np.linspace(-5,5,100)
X3, Y3 = np.meshgrid(x3,y3)
Z3 = (X3 + Y3 - 1) / 4
```

Direct python code

```
# Plot surfaces
ax.plot_surface(X1, Y1, Z1, alpha=0.5, color='red')
ax.plot_surface(X2, Y2, Z2, alpha=0.5, color='blue')
ax.plot_surface(X3, Y3, Z3, alpha=0.5, color='green')

# Legend proxies (like in main.py)
plane_proxy = [plt.Rectangle((0,0),1,1,fc=c) for c in ["red","
    blue","green"]]
ax.legend(plane_proxy, ["Plane 1", "Plane 2", "Required Plane"],
    loc="best")
```

Direct python code

```
# Labels and title
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("Three Planes")

plt.savefig("Figure.png", dpi=200)
plt.show()
```