4.11.24

RAVULA SHASHANK REDDY - EE25BTECH11047

September 29, 2025

Question

Find the equation of the plane through the line of intersection of

$$\mathbf{r}^T(\mathbf{i}+3\mathbf{j})+6=0, \mathbf{r}^T(3\mathbf{i}-\mathbf{j}-4\mathbf{k})=0$$

which is at unit distance from origin.

Equation

Family of Planes

$$\mathbf{n}_1^T\mathbf{x} - c_1 + \lambda(\mathbf{n}_2^T\mathbf{x} - c_2) = 0$$

Solution:

Given:

$$\pi_1 : \mathbf{n}_1 = \begin{pmatrix} 1 \\ 3 \\ 0 \end{pmatrix}, \; c_1 = -6, \tag{1}$$

$$\pi_2 : \mathbf{n}_2 = \begin{pmatrix} 3 \\ -1 \\ -4 \end{pmatrix}, \; c_2 = 0. \tag{2}$$

Family of planes:

$$(\mathbf{n}_1 + \lambda \mathbf{n}_2)^T \mathbf{x} = c_1 + \lambda c_2 \tag{3}$$

$$\mathbf{n}^T \mathbf{x} = c \tag{4}$$

Distance condition:

$$d = \frac{\left|\mathbf{n}^T \mathbf{x} - c\right|}{\left\|\mathbf{n}\right\|} \tag{5}$$

$$\mathbf{x} = \mathbf{0} \tag{6}$$

Solution:

$$\frac{|c_1 + \lambda c_2|}{\|\mathbf{n}_1 + \lambda \mathbf{n}_2\|} = 1 \quad \implies \quad (c_1 + \lambda c_2)^2 = (\mathbf{n}_1 + \lambda \mathbf{n}_2)^T (\mathbf{n}_1 + \lambda \mathbf{n}_2) \tag{7}$$

$$36 = \mathbf{n}_1^T \mathbf{n}_1 + 2\lambda \mathbf{n}_1^T \mathbf{n}_2 + \lambda^2 \mathbf{n}_2^T \mathbf{n}_2 \tag{8}$$

$$\mathbf{n}_1^T \mathbf{n}_1 = 1^2 + 3^2 = 10, \tag{9}$$

$$\mathbf{n}_1^T \mathbf{n}_2 = 1.3 + 3.(-1) + 0.(-4) = 0, \tag{10}$$

$$\mathbf{n}_2^T \mathbf{n}_2 = 3^2 + (-1)^2 + (-4)^2 = 26 \tag{11}$$

Hence:

$$36 = 10 + 26\lambda^2 \quad \implies \quad 26\lambda^2 = 26 \quad \implies \quad \lambda = \pm 1 \tag{12}$$

For $\lambda = 1$:

$$\begin{pmatrix} -\frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \mathbf{x} = 1 \tag{13}$$

For $\lambda = -1$:

$$\begin{pmatrix} \frac{1}{3} & -\frac{2}{3} & -\frac{2}{3} \end{pmatrix} \mathbf{x} = 1 \tag{14}$$

C Code

```
        #include <stdio.h>
#include <math.h>

// Dot product of 3D vectors
double dot(double a[3], double b[3]) {
    return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
}

// Print vector in matrix row form
void print_vec(double v[3]) {
    printf([ %.3f %.3f %.3f ], v[0], v[1], v[2]);
}

int main() {
    // Given plane parameters
    double n1[3] = {1, 3, 0};
    double n2[3] = {3, -1, -4};
    double c1 = -6, c2 = 0;
```

C Code

```c
// Inner products
double n1n1 = dot(n1, n1); // 10
double n1n2 = dot(n1, n2); // 0
double n2n2 = dot(n2, n2); // 26

printf(n1n1 = %.0f, n1n2 = %.0f, n2n2 = %.0f\n, n1n1, n1n2,
    n2n2);

// Quadratic: (c1+c2)^2 = n1n1 + 2 n1n2 + ^2 n2n2
// => (c2^2 - n2n2) ^2 + (2c1c2 - 2n1n2)  + (c1^2 - n1n1) = 0

double A = c2*c2 - n2n2;
double B = 2*c1*c2 - 2*n1n2;
double C = c1*c1 - n1n1;

printf(Quadratic: %.0f ^2 + %.0f  + %.0f = 0\n, A, B, C);
```

C Code

```
    double disc = B*B - 4*A*C;
    if (disc < 0) {
        printf(No real solutions for .\n);
        return 0;
    }

    double lambda1 = (-B + sqrt(disc)) / (2*A);
    double lambda2 = (-B - sqrt(disc)) / (2*A);

    printf(Solutions: 1 = %.3f, 2 = %.3f\n, lambda1, lambda2);

    // Compute normals for each
    double n_case1[3], n_case2[3];
    for(int i=0; i<3; i++) {
        n_case1[i] = (n1[i] + lambda1*n2[i]) / (c1 + lambda1*c2);
        n_case2[i] = (n1[i] + lambda2*n2[i]) / (c1 + lambda2*c2);
    }
```

C Code

```
    printf(\nEquation of required planes:\n);

    printf(1) );
    print_vec(n_case1);
    printf( * x = 1\n);

    printf(2) );
    print_vec(n_case2);
    printf( * x = 1\n);

    return 0;
}
```

Python Direct

```
    import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# local imports
from libs.line.funcs import *
from libs.triangle.funcs import *

# Given normals and constants
n1 = np.array([1, 3, 0])
n2 = np.array([3, -1, -4])
c1, c2 = -6, 0

# Inner products
n1n1 = np.dot(n1, n1)
n1n2 = np.dot(n1, n2)
n2n2 = np.dot(n2, n2)
```

Python Direct

```
# Quadratic coefficients
A = c2**2 - n2n2
B = 2*c1*c2 - 2*n1n2
C = c1**2 - n1n1

disc = B**2 - 4*A*C
if disc < 0:
    raise ValueError(No real solutions for )

lambda1 = (-B + np.sqrt(disc)) / (2*A)
lambda2 = (-B - np.sqrt(disc)) / (2*A)

# Compute normals in n^T x = 1 form
n_case1 = (n1 + lambda1*n2) / (c1 + lambda1*c2)
n_case2 = (n1 + lambda2*n2) / (c1 + lambda2*c2)

print(Plane 1: , n_case1,  x = 1)
print(Plane 2: , n_case2,  x = 1)
```

Python Direct

```python
# -------- Plotting --------
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Meshgrid
x = np.linspace(-4, 4, 20)
y = np.linspace(-4, 4, 20)
X, Y = np.meshgrid(x, y)

def plane_eq(normal, X, Y):
    a, b, c = normal
    if abs(c) < 1e-6:
        return np.full_like(X, np.nan)
    return (1 - a*X - b*Y) / c

Z1 = plane_eq(n_case1, X, Y)
Z2 = plane_eq(n_case2, X, Y)
```

```
# Plot both planes
ax.plot_surface(X, Y, Z1, alpha=0.6, color='lightblue', label=
    Plane 1)
ax.plot_surface(X, Y, Z2, alpha=0.6, color='lightgreen', label=
    Plane 2)

# Origin
ax.scatter(0, 0, 0, color='red', s=50)
ax.text(0, 0, 0, (0,0,0), color='red')

# Intersection line: cross product of n1 and n2 gives direction
d = np.cross(n1, n2)
# Solve for one point on the line (z=0 case)
A_mat = np.vstack([n1, n2])
b_vec = np.array([c1, c2])
# Take only x,y by ignoring z in this solve
P = np.linalg.lstsq(A_mat[:, :2], b_vec, rcond=None)[0]
point = np.array([P[0], P[1], 0])
```

Python Direct

```
t = np.linspace(-3, 3, 2)
line_points = (point.reshape(3,1) + np.outer(d, t))
ax.plot(line_points[0,:], line_points[1,:], line_points[2,:], 'k
    --', label=Intersection line)

# Labels
ax.text(line_points[0,0], line_points[1,0], line_points[2,0],
    Line of Intersection, color='black')

# Axes
ax.set_xlabel(X)
ax.set_ylabel(Y)
ax.set_zlabel(Z)
ax.set_title(Planes through intersection at unit distance from
    origin)
ax.view_init(20, 30)

plt.show()
```

Python Shared

```
    import numpy as np
import ctypes
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

# local imports (as per your requirement)
from libs.line import funcs as line_funcs
from libs.triangle import funcs as tri_funcs

# Given plane parameters
n1 = np.array([1, 3, 0], dtype=float)
n2 = np.array([3, -1, -4], dtype=float)
c1, c2 = -6, 0

# Inner products
n1n1 = np.dot(n1, n1)
n1n2 = np.dot(n1, n2)
n2n2 = np.dot(n2, n2)
```

Python Shared

```
# Quadratic coefficients
A = c2**2 - n2n2
B = 2*c1*c2 - 2*n1n2
C = c1**2 - n1n1

disc = B**2 - 4*A*C
if disc < 0:
    raise ValueError(No real solutions for )

lambda1 = (-B + np.sqrt(disc)) / (2*A)
lambda2 = (-B - np.sqrt(disc)) / (2*A)

# Normals in n^T x = 1 form
n_case1 = (n1 + lambda1*n2) / (c1 + lambda1*c2)
n_case2 = (n1 + lambda2*n2) / (c1 + lambda2*c2)

# -------- Shared Output Style --------
print(=== Shared Output with ctypes ===)
```

Python Shared

```
print(Solutions for :)
print(1 =, ctypes.c_double(lambda1).value)
print(2 =, ctypes.c_double(lambda2).value)

print(\nPlane equations in n^T x = 1 form:)
print(1) , [ctypes.c_double(val).value for val in n_case1],  x =
    1)
print(2) , [ctypes.c_double(val).value for val in n_case2],  x =
    1)
```

Python Shared

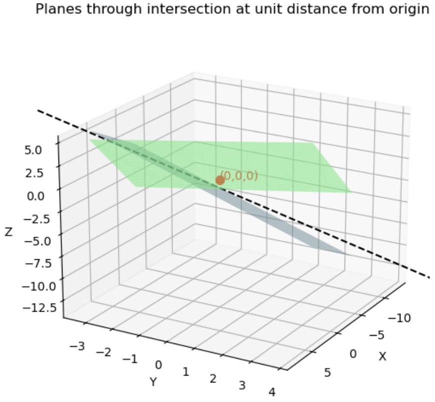
```
# -------- Plotting --------
x = np.linspace(-4, 4, 20)
y = np.linspace(-4, 4, 20)
X, Y = np.meshgrid(x, y)

def plane_eq(normal, X, Y):
    a, b, c = normal
    if abs(c) < 1e-6:
        return np.full_like(X, np.nan)
    return (1 - a*X - b*Y) / c

Z1 = plane_eq(n_case1, X, Y)
Z2 = plane_eq(n_case2, X, Y)

fig = plt.figur
```

Plot



Planes through intersection at unit distance from origin
(0,0,0)
Z
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−3
−2
−1
0
1
2
3
4
Y
−10
−5
0
5
X

Figure: