

4.11.4

Harsha-EE25BTECH11026

August 31,2025

Question

Find the equation of the plane which contains the line of intersection of the planes $\mathbf{r} \cdot (\hat{i} + 2\hat{j} + 3\hat{k}) - 4 = 0$ and $\mathbf{r} \cdot (2\hat{i} + \hat{j} - \hat{k}) + 5 = 0$ and which is perpendicular to the plane $\mathbf{r} \cdot (5\hat{i} + 3\hat{j} - 6\hat{k}) + 8 = 0$.

Theoretical Solution

According to the question,

$$\mathbf{n}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \mathbf{n}_2 = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} \quad c_1 = 4 \quad c_2 = -5 \quad (1)$$

The equation of intersection of planes is given by

$$\mathbf{n}_1^T \mathbf{x} - c_1 + \lambda (\mathbf{n}_2^T \mathbf{x} - c_2) = 0 \quad (2)$$

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

Theoretical Solution

Let the direction vector of the plane perpendicular to intersection of planes be \mathbf{n}_3

$$\therefore (\mathbf{n}_1^\top + \lambda \mathbf{n}_2^\top) \mathbf{n}_3 = 0 \quad (4)$$

$$\implies \lambda = -\frac{\mathbf{n}_1^\top \mathbf{n}_3}{\mathbf{n}_2^\top \mathbf{n}_3} \quad (5)$$

Theoretical Solution

$$\therefore \lambda = - \frac{\begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 5 \\ 3 \\ -6 \end{pmatrix}}{\begin{pmatrix} 2 & 1 & -1 \end{pmatrix} \begin{pmatrix} 5 \\ 3 \\ -6 \end{pmatrix}} = \frac{7}{19} \quad (6)$$

$$\Rightarrow \text{equation of the plane: } \begin{pmatrix} 33 & 45 & 50 \end{pmatrix} \mathbf{x} = 41 \quad (7)$$

C Code -Finding Equation of the plane

```
#include <stdio.h>

// Function to compute required plane coefficients
void required_plane(double n1[3], double d1,
                   double n2[3], double d2,
                   double n3[3], double d3,
                   double n_req[3], double *d_req) {

    // Find  $\lambda$  such that  $(n1 + \lambda n2) \cdot n3 = 0$ 
    double dot_n1n3 = n1[0]*n3[0] + n1[1]*n3[1] + n1[2]*n3[2];
    double dot_n2n3 = n2[0]*n3[0] + n2[1]*n3[1] + n2[2]*n3[2];
    double lam = -dot_n1n3 / dot_n2n3;
    // Required normal =  $n1 + \lambda n2$ 
    for(int i=0; i<3; i++) {
        n_req[i] = n1[i] + lam * n2[i];
    }
    // Required constant term
    *d_req = d1 + lam * d2;
}
```

Python+C code

```
import ctypes
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
import matplotlib as mp
from fractions import Fraction
import math
mp.use("TkAgg")

lib = ctypes.CDLL("./libplane.so")
# Define argtypes and restype
lib.required_plane.argtypes = [
    ctypes.POINTER(ctypes.c_double), ctypes.c_double,
    ctypes.POINTER(ctypes.c_double), ctypes.c_double,
    ctypes.POINTER(ctypes.c_double), ctypes.c_double,
    ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes.
        c_double)]
lib.required_plane.restype = None
```



```
# Input planes
n1 = (ctypes.c_double * 3)(1, 2, 3)
d1 = -4.0
n2 = (ctypes.c_double * 3)(2, 1, -1)
d2 = 5.0
n3 = (ctypes.c_double * 3)(5, 3, -6)
d3 = 8.0

# Output storage
n_req = (ctypes.c_double * 3)()
d_req = ctypes.c_double()

# Call C function
lib.required_plane(n1, d1, n2, d2, n3, d3, n_req, ctypes.byref(
    d_req))

# Convert to numpy + sympy
n_req_np = np.array([n_req[i] for i in range(3)], dtype=float)
d_req_val = float(d_req.value)
n_vec = sp.Matrix(n_req_np)
```

```
# Formatting helpers
def format_plane_integer(n, d):
    # Fractions for exactness
    n_frac = [Fraction(str(float(v))).limit_denominator() for v
               in n]
    d_frac = Fraction(str(float(d))).limit_denominator()
    # LCM of denominators
    denoms = [f.denominator for f in n_frac] + [d_frac.
                                                  denominator]
    lcm = math.lcm(*denoms)
    n_int = [int(f * lcm) for f in n_frac]
    d_int = int(d_frac * lcm)
    # Fix sign convention
    # Equation is  $n \cdot r + d = 0 \rightarrow n \cdot r = -d$ 
    rhs = -d_int
```

```
# Make gcd simplification
g = math.gcd(math.gcd(abs(n_int[0]), abs(n_int[1])), math.gcd(abs(
    (n_int[2]), abs(rhs)))
n_int = [ni // g for ni in n_int]
rhs //= g
return f"r ·({n_int[0]}, {n_int[1]}, {n_int[2]}) = {rhs}"
x, y, z = sp.symbols('x y z')
plane_eq = n_vec[0]*x + n_vec[1]*y + n_vec[2]*z + d_req_val
print("\nEquation of required plane:")
print(format_plane_integer(n_req_np, d_req_val))
```

Step 6: Plot planes

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

def plot_plane(ax, n, d, color, alpha=0.4):
    xx, yy = np.meshgrid(np.linspace(-5,5,15), np.linspace
        (-5,5,15))
    zz = (-n[0]*xx - n[1]*yy - d) / n[2]
    ax.plot_surface(xx, yy, zz, color=color, alpha=alpha)

# Convert sympy / ctypes →float numpy
n1f, d1f = np.array([1,2,3], dtype=float), -4.0
n2f, d2f = np.array([2,1,-1], dtype=float), 5.0
n3f, d3f = np.array([5,3,-6], dtype=float), 8.0
nrf = n_req_np
drf = d_req_val
```

```
# Plot given planes and required plane
plot_plane(ax, n1f, d1f, "red", 0.2) # 1
plot_plane(ax, n2f, d2f, "blue", 0.2) # 2
plot_plane(ax, n3f, d3f, "green", 0.2) # 3
plot_plane(ax, nrf, drf, "purple", 0.6) # Required plane

ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title(r"Required Plane through Intersection, $\perp$ to  
Given Plane")

plt.savefig("/home/user/Matrix/Matgeo_assignments/4.11.4/figs/  
Figure_1", dpi=300, bbox_inches="tight")
plt.show()
```

Python code

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mp
mp.use("TkAgg")
# Step 1: Define symbols
lam = sp.symbols('lam')
# Given plane normals and constants
n1, d1 = sp.Matrix([1,2,3]), -4
n2, d2 = sp.Matrix([2,1,-1]), 5
n3, d3 = sp.Matrix([5,3,-6]), 8
# Step 2: General plane through intersection
n = n1 + lam*n2 # normal vector depends on  $\lambda$ 
d = d1 + lam*d2 # constant term
# Step 3: Perpendicular condition
eq = sp.Eq(n.dot(n3), 0)
lam_val = sp.solve(eq, lam)[0]
```

```
# Substitute  $\lambda$  into plane equation
n_req = (n.subs(lam, lam_val))
d_req = d.subs(lam, lam_val)

# Step 4: Scale to integers (fix applied here)
all_vals = list(n_req) + [d_req] # avoid .tolist()
mult = sp.lcm([sp.denom(val) for val in all_vals])
n_req = (n_req * mult).applyfunc(sp.simplify)
d_req = sp.simplify(d_req * mult)

# Step 5: Vector form
x, y, z = sp.symbols('x y z')
plane_eq = n_req[0]*x + n_req[1]*y + n_req[2]*z + d_req
point_on_plane = sp.solve(plane_eq.subs({y:0, z:0}), x)
```

Python code

Step 6: Plot planes

```
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')
def plot_plane(ax, n, d, color, alpha=0.4):
    xx, yy = np.meshgrid(np.linspace(-5,5,15), np.linspace
        (-5,5,15))
    zz = (-n[0]*xx - n[1]*yy - d) / n[2]
    ax.plot_surface(xx, yy, zz, color=color, alpha=alpha)
# Convert sympy →float numpy
n1f, d1f = np.array(n1, dtype=float), float(d1)
n2f, d2f = np.array(n2, dtype=float), float(d2)
n3f, d3f = np.array(n3, dtype=float), float(d3)
nrf, drf = np.array([float(v) for v in n_req]), float(d_req)
# Plot given planes and required plane
plot_plane(ax, n1f, d1f, "red", 0.2)
plot_plane(ax, n2f, d2f, "blue", 0.2)
plot_plane(ax, n3f, d3f, "green", 0.2)
plot_plane(ax, nrf, drf, "purple", 0.6) # Required plane
```



```
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title(r"Required Plane through Intersection, $\perp$ to  
Given Plane")
plt.savefig("/home/user/Matrix/Matgeo_assignments/4.11.4/figs/  
Figure_1")
plt.show()
```

Required Plane through Intersection, \perp to Given Plane

