Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 19, 2025

Problem Statement

Problem Statement

Find the equation of the plane passing through the intersection of the planes

$$\mathbf{n}_1^\top \mathbf{x} - c_1 = 0, \qquad \mathbf{n}_2^\top \mathbf{x} - c_2 = 0 \tag{1.1}$$

which is parallel to the x-axis, and compute the perpendicular distance of this plane from the x-axis.

Description of Variables used

Quantity	Value
$\mathbf{n}_1$	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
c_1	1
$\mathbf{n}_2$	$\begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}$
c_2	-4
$\mathbf{e}_1$	$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

Table: Input data for the problem

Theoretical Solution

Step 1. General plane through the intersection

$$\left(\mathbf{n}_1 + k\mathbf{n}_2\right)^{\top}\mathbf{x} - \left(c_1 + kc_2\right) = 0 \tag{2.1}$$

$$\mathbf{n} = \mathbf{n}_1 + k\mathbf{n}_2 \tag{2.2}$$

$$c = c_1 + kc_2 \tag{2.3}$$

Step 2. Condition for parallelism with x-axis

$$\mathbf{e}_1^{\top}\mathbf{n} = 0 \tag{2.4}$$

$$\mathbf{e}_1^{\top}\mathbf{n}_1 + k\,\mathbf{e}_1^{\top}\mathbf{n}_2 = 0 \tag{2.5}$$

$$k = -\frac{\mathbf{e}_1^{\top}\mathbf{n}_1}{\mathbf{e}_1^{\top}\mathbf{n}_2} \tag{2.6}$$

Theoretical Solution

Step 3. Required plane

$$\mathbf{n} = \mathbf{n}_1 - \frac{\mathbf{e}_1^\top \mathbf{n}_1}{\mathbf{e}_1^\top \mathbf{n}_2}\,\mathbf{n}_2 \tag{2.7}$$

$$c = c_1 - \frac{\mathbf{e}_1^\top \mathbf{n}_1}{\mathbf{e}_1^\top \mathbf{n}_2}\,c_2 \tag{2.8}$$

$$\mathbf{n}^\top \mathbf{x} = c \tag{2.9}$$

Step 4. Distance from the x-axis

Let a point on the x-axis be

$$\mathbf{P} = t\mathbf{e}_1 \tag{2.10}$$

The perpendicular distance is

$$d = \frac{\left|\mathbf{n}^\top \mathbf{P} - c\right|}{\|\mathbf{n}\|} \tag{2.11}$$

$$= \frac{|c|}{\|\mathbf{n}\|}, \quad \text{since } \mathbf{n}^\top \mathbf{e}_1 = 0 \tag{2.12}$$

Theoretical Solution

Step 5. Substitution of values

$$\mathbf{e}_1^\top \mathbf{n}_1 = 1, \qquad \mathbf{e}_1^\top \mathbf{n}_2 = 2 \tag{2.13}$$

$$k = -\tfrac{1}{2} \tag{2.14}$$

$$\mathbf{n} = \mathbf{n}_1 - \tfrac{1}{2}\mathbf{n}_2 = \begin{pmatrix} 0 \\ -\frac{1}{2} \\ \frac{3}{2} \end{pmatrix} \tag{2.15}$$

$$c = c_1 - \tfrac{1}{2}c_2 = 3 \tag{2.16}$$

Norm:

$$\|\mathbf{n}\|^2 = \mathbf{n}_1^\top \mathbf{n}_1 - \mathbf{n}_1^\top \mathbf{n}_2 + \tfrac{1}{4}\mathbf{n}_2^\top \mathbf{n}_2 \tag{2.17}$$

$$= 3 - 4 + \tfrac{1}{4}(14) = \tfrac{5}{2} \tag{2.18}$$

$$\|\mathbf{n}\| = \tfrac{\sqrt{10}}{2} \tag{2.19}$$

Theoretical Solution

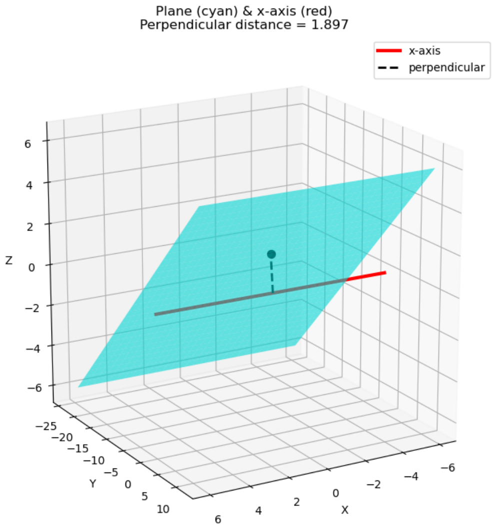
Final Results

$$\text{Equation of Plane:} \quad \left(\mathbf{n}_1 - \tfrac{1}{2}\mathbf{n}_2\right)^\top \mathbf{x} = c_1 - \tfrac{1}{2}c_2 \tag{2.20}$$

$$\implies \begin{pmatrix} 0 & -\frac{1}{2} & \frac{3}{2} \end{pmatrix} \mathbf{x} = 3 \tag{2.21}$$

$$\text{Distance from } x\text{-axis:} \quad d = \frac{|3|}{\sqrt{10}/2} = \frac{6}{\sqrt{10}} = \frac{3\sqrt{10}}{5} \tag{2.22}$$

Plot



Plane (cyan) & x-axis (red)
Perpendicular distance = 1.897
x-axis
perpendicular
Z
6
4
2
0
−2
−4
−6
Y
−25
−20
−15
−10
−5
0
5
10
X
6
4
2
0
−2
−4
−6

Figure

Code - C

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

// dot product
double dot(const double *a, const double *b, int n) {
    double s = 0.0;
    for (int i = 0; i < n; i++) s += a[i] * b[i];
    return s;
}

// compute n = n1 + k n2
void compute_n(const double *n1, const double *n2, double k, double
    *res, int n) {
    for (int i = 0; i < n; i++) res[i] = n1[i] + k*n2[i];
}
```

Code - C

```
// compute constant C = c1 + k c2
double compute_C(double c1, double c2, double k) {
    return c1 + k*c2;
}

// compute k = -(ex.n1)/(ex.n2)
double find_k(const double *ex, const double *n1, const double *n2,
    int n) {
    double num = dot(ex,n1,n);
    double den = dot(ex,n2,n);
    return -num/den;
}

// norm of vector
double norm(const double *a, int n) {
    return sqrt(dot(a,a,n));
}
```

Code - C

```
// distance = |C| / ||n||
double plane_distance(const double *n, double C, int nlen) {
    return fabs(C)/norm(n,nlen);
}

// foot of perpendicular from origin to plane: (C/||n||^2) * n
void foot_point(const double *n, double C, double *res, int nlen) {
    double factor = C/dot(n,n,nlen);
    for (int i = 0; i < nlen; i++) res[i] = factor*n[i];
}
```

Code - Python(with shared C code)

The code to obtain the required plot is

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

# --- load shared library ---
lib = ctypes.CDLL("./libplane3d.so")

# define argtypes/restypes
lib.find_k.restype = ctypes.c_double
lib.find_k.argtypes = [ctypes.POINTER(ctypes.c_double),
                       ctypes.POINTER(ctypes.c_double),
                       ctypes.POINTER(ctypes.c_double),
                       ctypes.c_int]
```

Code - Python(with shared C code)

```
lib.compute_n.argtypes = [ctypes.POINTER(ctypes.c_double),
                          ctypes.POINTER(ctypes.c_double),
                          ctypes.c_double,
                          ctypes.POINTER(ctypes.c_double),
                          ctypes.c_int]

lib.compute_C.restype = ctypes.c_double
lib.compute_C.argtypes = [ctypes.c_double, ctypes.c_double, ctypes.
    c_double]

lib.plane_distance.restype = ctypes.c_double
lib.plane_distance.argtypes = [ctypes.POINTER(ctypes.c_double), ctypes.
    c_double, ctypes.c_int]
```

Code - Python(with shared C code)

```
lib.foot_point.argtypes = [ctypes.POINTER(ctypes.c_double), ctypes.
    c_double,
                                   ctypes.POINTER(ctypes.c_double), ctypes.
                                        c_int]

# helper to convert numpy to C array
def c_arr(arr):
    return (ctypes.c_double * len(arr))(*arr.tolist())

# --- input data ---
n1 = np.array([1.0,1.0,1.0])
n2 = np.array([2.0,3.0,-1.0])
c1, c2 = 1.0, -4.0
ex = np.array([1.0,0.0,0.0])

n1_c, n2_c, ex_c = c_arr(n1), c_arr(n2), c_arr(ex)
```

Code - Python(with shared C code)

```
# Step 1: find k
k = lib.find_k(ex_c, n1_c, n2_c, 3)

# Step 2: compute plane normal n = n1 + k n2
n_c = (ctypes.c_double * 3)()
lib.compute_n(n1_c, n2_c, ctypes.c_double(k), n_c, 3)
n = np.array([n_c[i] for i in range(3)])

# Step 3: compute constant C
C = lib.compute_C(c1, c2, k)

# Step 4: distance
dist = lib.plane_distance(n_c, C, 3)
```

Code - Python(with shared C code)

```
# Step 5: foot of perpendicular from origin to plane
foot_c = (ctypes.c_double * 3)()
lib.foot_point(n_c, C, foot_c, 3)
foot = np.array([foot_c[i] for i in range(3)])

print("k =", k)
print("plane normal n =", n)
print("C =", C)
print("distance =", dist)
print("foot of perpendicular =", foot)

# --- Plotting ---
fig = plt.figure(figsize=(8,7))
ax = fig.add_subplot(111, projection='3d')
```

Code - Python(with shared C code)

```
# x-axis
x_line = np.linspace(-6,6,50)
ax.plot(x_line, np.zeros_like(x_line), np.zeros_like(x_line), color='r', lw=3,
    label="x-axis")

# plane surface
xx, zz = np.meshgrid(np.linspace(-6,6,30), np.linspace(-6,6,30))
yy = (C - n[0]*xx - n[2]*zz)/n[1]
ax.plot_surface(xx,yy,zz,alpha=0.6,color='cyan')

# perpendicular
ax.plot([0, foot[0]], [0, foot[1]], [0, foot[2]], 'k--', lw=2, label="
    perpendicular")
ax.scatter(*foot, color='k', s=50)
```

Code - Python(with shared C code)

```
# labels & view
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title(f"Plane (cyan) & x-axis (red)\nPerpendicular distance = {dist
    :.3f}")
ax.legend()
ax.view_init(elev=18, azim=60)
ax.set_box_aspect([1,1,1])

plt.tight_layout()
plt.savefig("plane.png")
plt.show()
```

Code - Python only

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

# --- Input data ---
n1 = np.array([1.0, 1.0, 1.0])
n2 = np.array([2.0, 3.0, -1.0])
c1, c2 = 1.0, -4.0
ex = np.array([1.0, 0.0, 0.0]) # x-axis direction

# --- Step 1: find k ---
k = -np.dot(ex, n1) / np.dot(ex, n2)

# --- Step 2: compute plane normal and constant ---
n = n1 + k*n2
C = c1 + k*c2
```

Code - Python only

```
# --- Step 3: distance formula ---
dist = abs(C)/np.linalg.norm(n)

# --- Step 4: foot of perpendicular from origin to plane ---
foot = (C/np.dot(n, n)) * n

print("k =", k)
print("plane normal =", n)
print("C =", C)
print("distance from x-axis =", dist)
print("foot of perpendicular =", foot)

# --- Plotting ---
fig = plt.figure(figsize=(8,7))
ax = fig.add_subplot(111, projection='3d')
```

Code - Python only

```
# Plot the x-axis (red line)
x_line = np.linspace(-6,6,50)
ax.plot(x_line, np.zeros_like(x_line), np.zeros_like(x_line),
        color='r', lw=3, label="x-axis")

# Plot the plane (cyan surface)
xx, zz = np.meshgrid(np.linspace(-6,6,30), np.linspace(-6,6,30))
yy = (C - n[0]*xx - n[2]*zz)/n[1]
ax.plot_surface(xx, yy, zz, alpha=0.6, color='cyan')

# Plot perpendicular line (black dashed) and foot point (black dot)
ax.plot([0, foot[0]], [0, foot[1]], [0, foot[2]], 'k--', lw=2, label="
    perpendicular")
ax.scatter(*foot, color='k', s=50)
```

Code - Python only

```
# Styling
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title(f"Plane (cyan) & x-axis (red)\nPerpendicular distance = {dist
    :.3f}")
ax.legend()
ax.view_init(elev=18, azim=60)
ax.set_box_aspect([1,1,1]) # equal aspect ratio

plt.tight_layout()
plt.savefig("new_plane.png")
plt.show()
```