MatGeo Assignment - Problem 2.8.5

EE25BTECH11024

IIT Hyderabad

October 2, 2025

Problem Statement

A line makes angles α, β, γ and δ with the diagonals of a cube, prove that

$$\cos^2 \alpha + \cos^2 \beta + \cos^2 \gamma + \cos^2 \delta = \frac{4}{3} \tag{2.8.5.1}$$

Solution:

Symbol	Value	Description
$\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4$	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \dots$	Column vectors for the four cube diagonals
$\mathbf{L}$	$\begin{pmatrix} l \\ m \\ n \end{pmatrix}$	Line's unit direction vector, where $\mathbf{L}^\top \mathbf{L} = 1$

For angle θ_i between the line $\mathbf{L}$ and a diagonal $\mathbf{D}_i$,

$$\cos\theta_i = \frac{\mathbf{L}^\top \mathbf{D}_i}{\|\mathbf{L}\| \|\mathbf{D}_i\|} = \frac{\mathbf{L}^\top \mathbf{D}_i}{\sqrt{3}} \tag{2.8.5.2}$$

Solution:

Since $\mathbf{L}^\top \mathbf{D}_i$ is a scalar, it equals its own transpose, so

$$(\mathbf{L}^\top \mathbf{D}_i)^2 = (\mathbf{L}^\top \mathbf{D}_i)(\mathbf{D}_i^\top \mathbf{L}). \tag{2.8.5.3}$$

using (**??**) and (**??**) to find S i.e sum of squares,

$$S = \sum_{i=1}^{4} \cos^2 \theta_i = \sum_{i=1}^{4} \frac{(\mathbf{L}^\top \mathbf{D}_i)(\mathbf{D}_i^\top \mathbf{L})}{3} = \frac{1}{3}\mathbf{L}^\top \left(\sum_{i=1}^{4} \mathbf{D}_i \mathbf{D}_i^\top \right) \mathbf{L} \tag{2.8.5.4}$$

The expression $\mathbf{D}_i \mathbf{D}_i^\top$ is the **outer product** of the vector with itself. Let's calculate the matrix $M = \sum_{i=1}^{4} \mathbf{D}_i \mathbf{D}_i^\top$.

$$\mathbf{D}_1 \mathbf{D}_1^\top = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^\top = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Solution:

$$\mathbf{D}_2\mathbf{D}_2^\top = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}^\top = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{D}_3\mathbf{D}_3^\top = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}^\top = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

$$\mathbf{D}_4\mathbf{D}_4^\top = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}^\top = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}$$

Solution:

By adding these four matrices we get,

$$M = \sum_{i=1}^{4} \mathbf{D}_i \mathbf{D}_i^\top = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix} = 4I \tag{2.8.5.5}$$

Substituting(**??**) in (**??**) we get,

$$S = \frac{1}{3}\mathbf{L}^\top(4I)\mathbf{L} = \frac{4}{3}\mathbf{L}^\top I \mathbf{L} = \frac{4}{3}\mathbf{L}^\top \mathbf{L} \tag{2.8.5.6}$$

Since $\mathbf{L}$ is a unit vector, $\mathbf{L}^\top\mathbf{L} = \|\mathbf{L}\|^2 = 1$.

$$S = \frac{4}{3}(1) = \frac{4}{3} \tag{2.8.5.7}$$

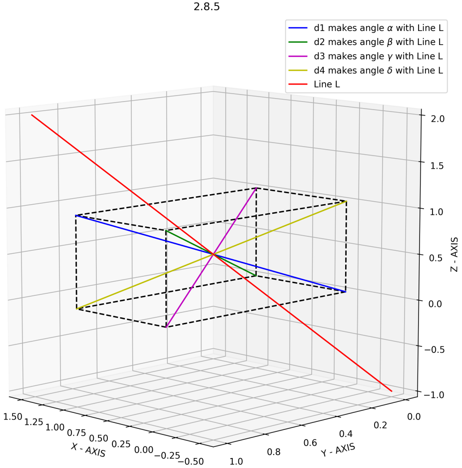
Final Answer

Thus, it is proven that

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma + \cos^2\delta = \frac{4}{3}.$$

See Figure **??**.

Figure



2.8.5
d1 makes angle α with Line L
d2 makes angle β with Line L
d3 makes angle γ with Line L
d4 makes angle δ with Line L
Line L
X - AXIS
Y - AXIS
Z - AXIS

Python Code: plot.py (Native)

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

p1 = np.array([0,0,0])
p2 = np.array([1,0,0])
p3 = np.array([1,0,1])
p4 = np.array([0,0,1])
p5 = np.array([0,1,0])
p6 = np.array([1,1,0])
p7 = np.array([1,1,1])
p8 = np.array([0,1,1])

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

Python Code (Native Implementation – plot.py)

```
# Bottom face
ax.plot([p1[0], p2[0]], [p1[1], p2[1]], [p1[2], p2[2]], 'k--')
ax.plot([p2[0], p3[0]], [p2[1], p3[1]], [p2[2], p3[2]], 'k--')
ax.plot([p3[0], p4[0]], [p3[1], p4[1]], [p3[2], p4[2]], 'k--')
ax.plot([p4[0], p1[0]], [p4[1], p1[1]], [p4[2], p1[2]], 'k--')

# Top face
ax.plot([p5[0], p6[0]], [p5[1], p6[1]], [p5[2], p6[2]], 'k--')
ax.plot([p6[0], p7[0]], [p6[1], p7[1]], [p6[2], p7[2]], 'k--')
ax.plot([p7[0], p8[0]], [p7[1], p8[1]], [p7[2], p8[2]], 'k--')
ax.plot([p8[0], p5[0]], [p8[1], p5[1]], [p8[2], p5[2]], 'k--')

# Vertical edges
ax.plot([p1[0], p5[0]], [p1[1], p5[1]], [p1[2], p5[2]], 'k--')
ax.plot([p2[0], p6[0]], [p2[1], p6[1]], [p2[2], p6[2]], 'k--')
ax.plot([p3[0], p7[0]], [p3[1], p7[1]], [p3[2], p7[2]], 'k--')
ax.plot([p4[0], p8[0]], [p4[1], p8[1]], [p4[2], p8[2]], 'k--')
```

Python Code (Native Implementation – plot.py)

```
# Body diagonal from p1 to p7
ax.plot([p1[0], p7[0]], [p1[1], p7[1]], [p1[2], p7[2]], 'b', label = r"
    d1 makes angle $\alpha$ with Line L")

# Body diagonal from p2 to p8
ax.plot([p2[0], p8[0]], [p2[1], p8[1]], [p2[2], p8[2]], 'g', label = r"
    d2 makes angle $\beta$ with Line L")

# Body diagonal from p3 to p5
ax.plot([p3[0], p5[0]], [p3[1], p5[1]], [p3[2], p5[2]], 'm', label = r"
    d3 makes angle $\gamma$ with Line L")

# Body diagonal from p4 to p6
ax.plot([p4[0], p6[0]], [p4[1], p6[1]], [p4[2], p6[2]], 'y', label = r"
    d4 makes angle $\delta$ with Line L")

# Center of the cube
center = np.array([0.5, 0.5, 0.5])
```

Python Code (Native Implementation – plot.py)

```
v = np.array([2, 1, 3]) # Direction vector of the line

k = 0.5 # Parameter for length of the line

start = center - k * v
end = center + k * v

ax.plot([start[0], end[0]], [start[1], end[1]], [start[2], end[2]], 'r',
     label = 'Line L')

ax.set_xlabel("X - AXIS")
ax.set_ylabel("Y - AXIS")
ax.set_zlabel("Z - AXIS")
ax.set_title("2.8.5")

ax.legend()
ax.set_box_aspect([1, 1, 1])
ax.view_init(elev=10, azim=135)
plt.savefig("fig4.png", dpi=300)
plt.show()
```

C Code (Shared Library – findcube.c)

```c
#include <stdio.h>

void find_cube(double n, double *dir_vec, double len_par, double *
     cube_pts, double *start, double *end) {
    double pts[8][3] = {
        {0, 0, 0}, {n, 0, 0}, {n, 0, n}, {0, 0, n}, // bottom face
        {0, n, 0}, {n, n, 0}, {n, n, n}, {0, n, n} // top face
    };

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 3; j++) {
            cube_pts[i*3 + j] = pts[i][j];
        }
    }

    double center[3] = { n/2, n/2, n/2 };
```

C Code (Shared Library – findprojection.c)

```
    for (int i = 0; i < 3; i++) {
        start[i] = center[i] - len_par * dir_vec[i];
        end[i] = center[i] + len_par * dir_vec[i];
    }
}
```

Python Code (C Integrated – call.py)

```python
import ctypes
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

so = ctypes.CDLL("./find_cube.so")

so.find_cube.argtypes = [
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
so.find_cube.restype = None

n = 1.0
dir_vec = np.array([2, 1, 3], dtype=np.double)
dir_vec_ptr = dir_vec.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
len_par = 0.5
```

Python Code (C Integrated – call.py)

```
cube_pts = (ctypes.c_double*24)() # 8 points  3 coords
start_arr = (ctypes.c_double*3)()
end_arr = (ctypes.c_double*3)()

so.find_cube(n, dir_vec_ptr, len_par, cube_pts, start_arr, end_arr)

cube_pts_np = np.array(cube_pts).reshape(8,3)
start = np.array(start_arr)
end = np.array(end_arr)

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

edges_idx = [
    (0,1),(1,2),(2,3),(3,0), # bottom
    (4,5),(5,6),(6,7),(7,4), # top
    (0,4),(1,5),(2,6),(3,7) # vertical
]
```

Python Code (C Integrated – call.py)

```
for i,j in edges_idx:
    s = cube_pts_np[i]
    e = cube_pts_np[j]
    ax.plot([s[0],e[0]], [s[1],e[1]], [s[2],e[2]], 'k--')

body_diags_idx = [(0,6),(1,7),(2,4),(3,5)]
colors = ['b','g','m','y']
labels = ['d1','d2','d3','d4']

for (i,j),color,label in zip(body_diags_idx,colors,labels):
    s = cube_pts_np[i]
    e = cube_pts_np[j]
    ax.plot([s[0],e[0]], [s[1],e[1]], [s[2],e[2]], color=color, label=
        label)

ax.plot([start[0],end[0]], [start[1],end[1]], [start[2],end[2]], 'r',
    label='Line L')
```

Python Code (C Integrated – call.py)

```
ax.set_xlabel("X - AXIS")
ax.set_ylabel("Y - AXIS")
ax.set_zlabel("Z - AXIS")
ax.set_title(r"Line L making angles $\alpha, \beta, \gamma, \delta$ with
     the body diagonals of a cube")
ax.legend()
ax.set_box_aspect([1,1,1])
ax.view_init(elev=10, azim=135)

plt.savefig("fig_cube_line.png", dpi=300)
plt.show()
```