Matgeo Presentation - Problem 2.10.84

ee25btech11056 - Suraj.N

September 5, 2025

Problem Statement

Let Q be the cube with the set of vertices

$$\{(x_1, x_2, x_3) \mid x_1, x_2, x_3 \in \{0, 1\}\} \subset \mathbb{R}^3$$

Let F be the set of all twelve lines containing the diagonals of the six faces of the cube Q. Let S be the set of all four lines containing the main diagonals of the cube Q; for instance, the line passing through the vertices $(0, 0, 0)$ and $(1, 1, 1)$ is in S.
For lines λ_1 and λ_2, let $d(\lambda_1, \lambda_2)$ denote the shortest distance between them. Then the maximum value of $d(\lambda_1, \lambda_2)$, as λ_1 varies over F and λ_2 varies over S, is

Data

The diagonals of the cube can be written as

Line	Equation
Body diagonal	$\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + k_1 \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$
Face diagonal	$\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + k_2 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$

Table : diagonals

Solution

$$\mathbf{x} = \mathbf{A} + k_1\mathbf{m}_1, \quad \mathbf{A} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \ \mathbf{m}_1 = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \tag{0.1}$$

$$\mathbf{x} = \mathbf{B} + k_2\mathbf{m}_2, \quad \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \ \mathbf{m}_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \tag{0.2}$$

$$\mathbf{M} = \begin{pmatrix} \mathbf{m_1} & \mathbf{m_2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix} \tag{0.3}$$

$$\left(\mathbf{B} - \mathbf{A}\right) = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{0.4}$$

$$\begin{pmatrix}\mathbf{M} & \mathbf{B}-\mathbf{A}\end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ -1 & 1 & -1 \end{pmatrix} \tag{0.5}$$

Performing row operations:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ -1 & 1 & -1 \end{pmatrix} \xleftrightarrow{R_2 \to R_2 - R_1} \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 1 & -1 \end{pmatrix} \xleftrightarrow{R_3 \to R_3 + R_1} \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 2 & -1 \end{pmatrix} \tag{0.6}$$

$$\xleftrightarrow{R_3 \to R_3 + 2R_2} \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{0.7}$$

Clearly, the rank of this matrix is 3, and therefore, the lines are skew.

From the least squares formulation,

$$\mathbf{M}^\top \mathbf{M} \begin{pmatrix} k_1 \\ -k_2 \end{pmatrix} = \mathbf{M}^\top (\mathbf{B} - \mathbf{A}) \tag{0.8}$$

Thus,

$$\mathbf{M}^\top \mathbf{M} = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \tag{0.9}$$

$$\mathbf{M}^\top (\mathbf{B} - \mathbf{A}) = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{0.10}$$

Therefore,

$$\begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} k_1 \\ -k_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{0.11}$$

Solving,

$$\begin{pmatrix} k_1 \\ -k_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ -\frac{1}{2} \end{pmatrix} \tag{0.12}$$

Hence the closest points are

$$\mathbf{P} = \mathbf{A} + k_1\mathbf{m}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{pmatrix} \tag{0.13}$$

$$\mathbf{Q} = \mathbf{B} + k_2\mathbf{m}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{pmatrix} \tag{0.14}$$

The shortest distance is

$$\|\mathbf{P} - \mathbf{Q}\| = \left\| \begin{pmatrix} -\frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{6} \end{pmatrix} \right\| = \frac{1}{\sqrt{6}} \tag{0.15}$$

Plot

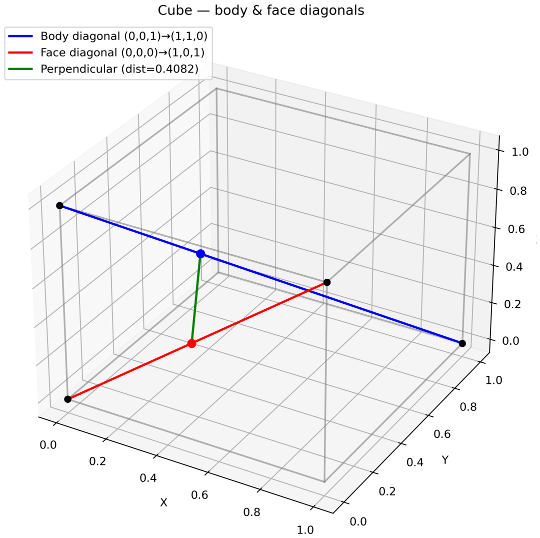

Fig : diagonals

C Code: points.c

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

/* Compute shortest distance between two lines:
   Line1: point P1=(0,0,1), direction d1=(1,1,-1)
   Line2: point P2=(0,0,0), direction d2=(1,0,1)
*/
void compute_distance(double *result) {
  /* Points */
  double p1x = 0.0, p1y = 0.0, p1z = 1.0;
  double p2x = 0.0, p2y = 0.0, p2z = 0.0;

  /* Directions */
  double d1x = 1.0, d1y = 1.0, d1z = -1.0;
  double d2x = 1.0, d2y = 0.0, d2z = 1.0;

  /* P2 - P1 */
  double rx = p2x - p1x;
  double ry = p2y - p1y;
  double rz = p2z - p1z;

  /* cross = d1 x d2 */
  double cx = d1y * d2z - d1z * d2y;
  double cy = d1z * d2x - d1x * d2z;
  double cz = d1x * d2y - d1y * d2x;

  double numer = fabs(rx * cx + ry * cy + rz * cz);
  double denom = sqrt(cx * cx + cy * cy + cz * cz);

  *result = numer / denom;
}
```

Python: call_c.py

```
# call_c.py
import ctypes
import sys
import numpy as np
import matplotlib.pyplot as plt

# Load C shared lib
lib = ctypes.CDLL("./libpoints.so")
lib.compute_distance.restype = None
lib.compute_distance.argtypes = [ctypes.POINTER(ctypes.c_double)]

# Call C function to get distance
dist = ctypes.c_double()
lib.compute_distance(ctypes.byref(dist))
print("Shortest distance (from C):", dist.value)

# Define endpoints exactly
P1 = np.array([0.0, 0.0, 1.0]) # start of body diagonal
D1 = np.array([1.0, 1.0, -1.0]) # direction -> endpoint P1 + D1 = (1,1,0)
P2 = np.array([0.0, 0.0, 0.0]) # start of face diagonal
D2 = np.array([1.0, 0.0, 1.0]) # direction -> endpoint P2 + D2 = (1,0,1)

# Compute closest points on the two infinite lines (analytic)
w0 = P1 - P2
A = np.dot(D1, D1)
B = np.dot(D1, D2)
C = np.dot(D2, D2)
rhs = np.array([-np.dot(D1, w0), -np.dot(D2, w0)])
M = np.array([[A, -B],[B, -C]])
s, t = np.linalg.solve(M, rhs) # s for line1, t for line2
```

Python: call_c.py

```
closest1 = P1 + s * D1
closest2 = P2 + t * D2

print("Closest␣point␣on␣body␣diagonal:", closest1)
print("Closest␣point␣on␣face␣diagonal:", closest2)
print("Distance␣between␣them␣(computed):", np.linalg.norm(closest1 - closest2))

# --- plotting (neat cube wireframe, diagonals precise, perpendicular segment) ---
def set_axes_equal(ax):
    # make 3D axes equal
    x_limits = ax.get_xlim3d()
    y_limits = ax.get_ylim3d()
    z_limits = ax.get_zlim3d()
    x_range = abs(x_limits[1] - x_limits[0])
    y_range = abs(y_limits[1] - y_limits[0])
    z_range = abs(z_limits[1] - z_limits[0])
    max_range = max(x_range, y_range, z_range)
    x_mid = np.mean(x_limits)
    y_mid = np.mean(y_limits)
    z_mid = np.mean(z_limits)
    half = max_range / 2
    ax.set_xlim(x_mid - half, x_mid + half)
    ax.set_ylim(y_mid - half, y_mid + half)
    ax.set_zlim(z_mid - half, z_mid + half)

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

Python: call_c.py

```
# cube vertices and edges
vertices = np.array([[0,0,0],[1,0,0],[1,1,0],[0,1,0],
                     [0,0,1],[1,0,1],[1,1,1],[0,1,1]])
edges = [(0,1),(1,2),(2,3),(3,0),
         (4,5),(5,6),(6,7),(7,4),
         (0,4),(1,5),(2,6),(3,7)]
for e in edges:
    a = vertices[e[0]]
    b = vertices[e[1]]
    ax.plot([a[0],b[0]],[a[1],b[1]],[a[2],b[2]], color='gray', alpha=0.6)

# precise diagonals (vertex to vertex lines, no arrows)
ax.plot([P1[0], (P1 + D1)[0] ], [P1[1], (P1 + D1)[1] ], [P1[2], (P1 + D1)[2] ],
        color='blue', linewidth=2, label='Body diagonal (0,0,1)(1,1,0)')
ax.plot([P2[0], (P2 + D2)[0] ], [P2[1], (P2 + D2)[1] ], [P2[2], (P2 + D2)[2] ],
        color='red', linewidth=2, label='Face diagonal (0,0,0)(1,0,1)')

# perpendicular shortest segment between the two lines
ax.plot([closest1[0], closest2[0]],
        [closest1[1], closest2[1]],
        [closest1[2], closest2[2]],
        color='green', linewidth=2, label=f'Perpendicular (dist={np.linalg.norm(closest1-closest2):.4f})')

# mark endpoints and closest points
ax.scatter(*(P1), color='black', s=30)
ax.scatter(*((P1 + D1)), color='black', s=30)
ax.scatter(*(P2), color='black', s=30)
ax.scatter(*((P2 + D2)), color='black', s=30)
ax.scatter(*closest1, color='blue', s=50) # closest on body diag
ax.scatter(*closest2, color='red', s=50) # closest on face diag
```

Python: call_c.py

```
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
ax.set_title('Cube␣␣body␣&␣face␣diagonals')
ax.legend(loc='upper␣left')
set_axes_equal(ax)

plt.tight_layout()
plt.savefig("cube_lines.png", dpi=300, bbox_inches='tight')
plt.show()
```

Python: plot.py

```
# plot.py
import numpy as np
import matplotlib.pyplot as plt

# Line definitions (exact)
P1 = np.array([0.0, 0.0, 1.0]) # body diagonal start
D1 = np.array([1.0, 1.0, -1.0]) # body diagonal direction -> end (1,1,0)
P2 = np.array([0.0, 0.0, 0.0]) # face diagonal start
D2 = np.array([1.0, 0.0, 1.0]) # face diagonal direction -> end (1,0,1)

# exact shortest distance using cross product
cross = np.cross(D1, D2)
distance = abs(np.dot(P2 - P1, cross)) / np.linalg.norm(cross)
print("Shortest distance (NumPy):", distance)

# compute closest points (solve 2x2 linear system)
w0 = P1 - P2
A = np.dot(D1, D1)
B = np.dot(D1, D2)
C = np.dot(D2, D2)
rhs = np.array([-np.dot(D1, w0), -np.dot(D2, w0)])
M = np.array([[A, -B],[B, -C]])
s, t = np.linalg.solve(M, rhs)
closest1 = P1 + s * D1
closest2 = P2 + t * D2
print("Closest points:", closest1, closest2)
print("Check distance:", np.linalg.norm(closest1 - closest2))
```

Python: plot.py

```
# plotting
def set_axes_equal(ax):
    x_limits = ax.get_xlim3d()
    y_limits = ax.get_ylim3d()
    z_limits = ax.get_zlim3d()
    x_range = abs(x_limits[1] - x_limits[0])
    y_range = abs(y_limits[1] - y_limits[0])
    z_range = abs(z_limits[1] - z_limits[0])
    max_range = max(x_range, y_range, z_range)
    x_mid = np.mean(x_limits)
    y_mid = np.mean(y_limits)
    z_mid = np.mean(z_limits)
    half = max_range / 2
    ax.set_xlim(x_mid - half, x_mid + half)
    ax.set_ylim(y_mid - half, y_mid + half)
    ax.set_zlim(z_mid - half, z_mid + half)

fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(111, projection='3d')

vertices = np.array([[0,0,0],[1,0,0],[1,1,0],[0,1,0],
                     [0,0,1],[1,0,1],[1,1,1],[0,1,1]])
edges = [(0,1),(1,2),(2,3),(3,0),
         (4,5),(5,6),(6,7),(7,4),
         (0,4),(1,5),(2,6),(3,7)]
for e in edges:
    a = vertices[e[0]]
    b = vertices[e[1]]
    ax.plot([a[0],b[0]],[a[1],b[1]],[a[2],b[2]], color='gray', alpha=0.6)
```

Python: plot.py

```
# diagonals as precise lines from vertex to vertex (no arrowheads)
ax.plot([P1[0], P1[0]+D1[0]],[P1[1], P1[1]+D1[1]],[P1[2], P1[2]+D1[2]],
       color='blue', linewidth=2, label='Body diagonal (0,0,1)(1,1,0)')
ax.plot([P2[0], P2[0]+D2[0]],[P2[1], P2[1]+D2[1]],[P2[2], P2[2]+D2[2]],
       color='red', linewidth=2, label='Face diagonal (0,0,0)(1,0,1)')

# exact perpendicular segment
ax.plot([closest1[0], closest2[0]],
       [closest1[1], closest2[1]],
       [closest1[2], closest2[2]],
       color='green', linewidth=2, label=f'Perpendicular (dist={distance:.4f})')

# markers for vertices and closest points
ax.scatter(*P1, color='black', s=30)
ax.scatter(*(P1 + D1), color='black', s=30)
ax.scatter(*P2, color='black', s=30)
ax.scatter(*(P2 + D2), color='black', s=30)
ax.scatter(*closest1, color='blue', s=50)
ax.scatter(*closest2, color='red', s=50)

ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
ax.set_title('Cube  body & face diagonals')
ax.legend(loc='upper left')
set_axes_equal(ax)

plt.tight_layout()
plt.savefig("cube_lines.png", dpi=300, bbox_inches='tight')
plt.show()
```