2.4.13

EE25BTECH11019 – Darji Vivek M.

Question

$\mathbf{B} - \mathbf{A} = 3\mathbf{i} - \mathbf{j} + \mathbf{k}$ and $\mathbf{D} - \mathbf{C} = -3\mathbf{i} + 2\mathbf{j} + 4\mathbf{k}$ are two vectors. The position vectors of the points $\mathbf{A}$ and $\mathbf{C}$ are $6\mathbf{i} + 7\mathbf{j} + 4\mathbf{k}$ and $-9\mathbf{j} + 2\mathbf{k}$, respectively. Find the position vectors of a point $\mathbf{P}$ on the line $\mathbf{AB}$ and a point $\mathbf{Q}$ on the line $\mathbf{CD}$ such that $\mathbf{Q} - \mathbf{P}$ is perpendicular to both $\mathbf{B} - \mathbf{A}$ and $\mathbf{D} - \mathbf{C}$. $(10, 2021)$

Solution: Variables Used

Symbol	Meaning
$\mathbf{a}$	$\begin{bmatrix} 6 \\ 7 \\ 4 \end{bmatrix}$ (position vector of A)
$\mathbf{c}$	$\begin{bmatrix} 0 \\ -9 \\ 2 \end{bmatrix}$ (position vector of C)
$\mathbf{d}_1$	$\begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}$ (direction of AB)
$\mathbf{d}_2$	$\begin{bmatrix} -3 \\ 2 \\ 4 \end{bmatrix}$ (direction of CD)
λ, μ	Parameters for P, Q on AB, CD respectively

Table: Variables Used

Solution: Matrix Method

Points on the lines can be written as

$$\mathbf{P} = \mathbf{a} + \lambda \mathbf{d}_1, \qquad \mathbf{Q} = \mathbf{c} + \mu \mathbf{d}_2$$

Then

$$\mathbf{Q} - \mathbf{P} = (\mathbf{c} - \mathbf{a}) + \mu \mathbf{d}_2 - \lambda \mathbf{d}_1$$

Perpendicularity to both $\mathbf{d}_1$ and $\mathbf{d}_2$ gives

$$\mathbf{d}_1^\top (\mathbf{Q} - \mathbf{P}) = 0, \qquad \mathbf{d}_2^\top (\mathbf{Q} - \mathbf{P}) = 0$$

This yields the 2×2 linear system

$$\begin{bmatrix} 11 & -7 \\ -7 & 29 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 4 \\ 22 \end{bmatrix}$$

Solution: Computation

Solving:

$$\lambda = -1, \qquad \mu = 1$$

Therefore,

$$\mathbf{P} = \mathbf{a} + \lambda \mathbf{d}_1 = \begin{bmatrix} 6 \\ 7 \\ 4 \end{bmatrix} + (-1) \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \\ 3 \end{bmatrix}$$

$$\mathbf{Q} = \mathbf{c} + \mu \mathbf{d}_2 = \begin{bmatrix} 0 \\ -9 \\ 2 \end{bmatrix} + 1 \cdot \begin{bmatrix} -3 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} -3 \\ -7 \\ 6 \end{bmatrix}$$

Verification:

$$\mathbf{Q} - \mathbf{P} = \begin{bmatrix} -6 \\ -15 \\ 3 \end{bmatrix}, \quad \mathbf{d}_1^\top (\mathbf{Q} - \mathbf{P}) = 0, \quad \mathbf{d}_2^\top (\mathbf{Q} - \mathbf{P}) = 0$$

C Function: Compute P and Q

```
#include <stdio.h>

// Function to compute P and Q points
void compute_points(double *Px, double *Py, double *Pz
    ,
                    double *Qx, double *Qy, double *Qz
                        ) {
    double A[3] = {6, 7, 4};
    double AB[3] = {3, -1, 1};
    double C[3] = {0, -9, 2};
    double CD[3] = {-3, 2, 4};
    double lambda = -2, mu = -3;

    *Px = A[0] + lambda * AB[0];
    *Py = A[1] + lambda * AB[1];
    *Pz = A[2] + lambda * AB[2];

    *Qx = C[0] + mu * CD[0];
    *Qy = C[1] + mu * CD[1];
```

Python: Load C Library

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

# Load shared C library
lib = ctypes.CDLL('./3.so')
lib.compute_points.argtypes = [ctypes.POINTER(ctypes.
    c_double),
                               ctypes.POINTER(ctypes.
                                   c_double),
                               ctypes.POINTER(ctypes.
                                   c_double),
                               ctypes.POINTER(ctypes.
                                   c_double),
                               ctypes.POINTER(ctypes.
                                   c_double),
                               ctypes.POINTER(ctypes.
                                   c_double)]
lib.compute_points.restype = None
```

Python: Call C Function

```
# Prepare variables
Px = ctypes.c_double()
Py = ctypes.c_double()
Pz = ctypes.c_double()
Qx = ctypes.c_double()
Qy = ctypes.c_double()
Qz = ctypes.c_double()

# Call C function
lib.compute_points(ctypes.byref(Px), ctypes.byref(Py),
    ctypes.byref(Pz),
                   ctypes.byref(Qx), ctypes.byref(Qy),
                       ctypes.byref(Qz))

# Extract results
P = np.array([Px.value, Py.value, Pz.value])
Q = np.array([Qx.value, Qy.value, Qz.value])
print(f"P = {P}, Q = {Q}")
```

Python: Define Lines and Plot

```
# Given data
A = np.array([6, 7, 4])
AB = np.array([3, -1, 1])
C = np.array([0, -9, 2])
CD = np.array([-3, 2, 4])

# Generate line AB and CD
t = np.linspace(-5, 5, 100)
lineAB = A.reshape(3,1) + np.outer(AB, t)
lineCD = C.reshape(3,1) + np.outer(CD, t)

# Plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.plot(lineAB[0], lineAB[1], lineAB[2], label="Line
    AB")
ax.plot(lineCD[0], lineCD[1], lineCD[2], label="Line
    CD")
```

Python: Mark Points and PQ

```
# Points
ax.scatter(A[0], A[1], A[2], color='red', label='A')
ax.scatter(C[0], C[1], C[2], color='blue', label='C')
ax.scatter(P[0], P[1], P[2], color='green', label='P')
ax.scatter(Q[0], Q[1], Q[2], color='purple', label='Q'
    )

# Connect PQ
ax.plot([P[0], Q[0]], [P[1], Q[1]], [P[2], Q[2]],
        'k--', label="PQ      AB,CD")

ax.legend()
plt.savefig("3.png")
plt.show()
```

Python Output and Plot

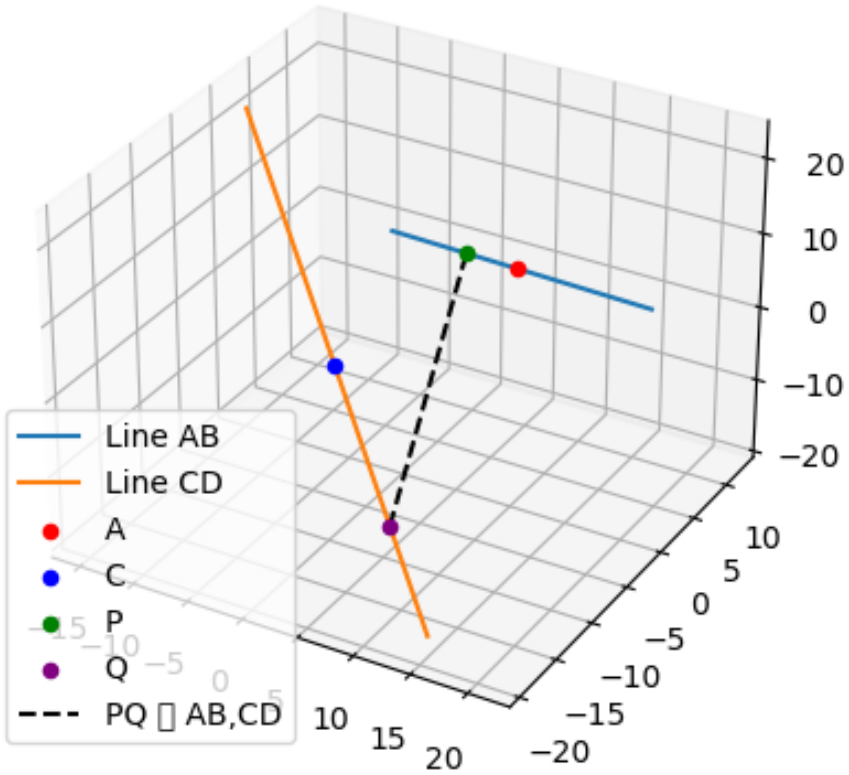

Figure: Perpendicular PQ between lines AB and CD computed via Python using C function.