4.4.38

EE25BTECH11002 - Achat Parth Kalpesh

September 30,2025

Question

Find the equation of the line which bisects the line segment joining points $\mathbf{A}\,(2,3,4)$ and $\mathbf{B}\,(4,5,8)$ and is perpendicular to the lines:

Line 1: $\frac{x-8}{3} = \frac{y+19}{-16} = \frac{z-10}{7}$

Line 2: $\frac{x-15}{3} = \frac{y-29}{8} = \frac{z-5}{-5}$

Solution:

Let the equation of the required line be

$$\mathbf{x} = \mathbf{h} + \kappa \mathbf{m} \tag{1}$$

where $\mathbf{h}$ is any point on the line and $\mathbf{m}$ is the direction vector of the line
Let the direction vectors of the given lines be $\mathbf{m_1}$ and $\mathbf{m_2}$

$$\mathbf{m_1} = \begin{pmatrix} 3 \\ -16 \\ 7 \end{pmatrix} \tag{2}$$

$$\mathbf{m_2} = \begin{pmatrix} 3 \\ 8 \\ -5 \end{pmatrix} \tag{3}$$

Solution:

The required line bisects the line segment joining **A** and **B**. Therefore, the point **h** on the line is the midpoint of **A** and **B**

$$\mathbf{h} = \frac{\mathbf{A} + \mathbf{B}}{2} \tag{4}$$

Solution:

By the given condition,

$$\mathbf{m_1}^\top \mathbf{m} = \mathbf{0} \tag{5}$$

$$\mathbf{m_2}^\top \mathbf{m} = \mathbf{0} \tag{6}$$

$$\begin{pmatrix} \mathbf{m_1}^\top \\ \mathbf{m_2}^\top \end{pmatrix} \mathbf{m} = \mathbf{0} \tag{7}$$

Solution:

$$\begin{pmatrix} 3 & -16 & 7 \\ 3 & 8 & -5 \end{pmatrix} \mathbf{m} = \mathbf{0} \xleftrightarrow{R_2 \to R_2 - R_1} \begin{pmatrix} 3 & -16 & 7 \\ 0 & 24 & -12 \end{pmatrix} \tag{8}$$

$$\xleftrightarrow{R_1 \leftarrow R_1 + \frac{2}{3} R_2} \begin{pmatrix} 3 & 0 & -1 \\ 0 & 24 & -12 \end{pmatrix} \xleftrightarrow{R_2 \leftarrow \frac{R_2}{12}} \begin{pmatrix} 3 & 0 & -1 \\ 0 & 2 & -1 \end{pmatrix} \tag{9}$$

This yeilds

$$\mathbf{m} = \begin{pmatrix} 2 \\ 3 \\ 6 \end{pmatrix} \tag{10}$$

Solution:

Hence, the vector equation of the line passing through $\mathbf{h}$ is

$$\mathbf{x} = \mathbf{h} + \kappa\mathbf{m} \tag{11}$$

$$\mathbf{x} = \left(\frac{\mathbf{A}+\mathbf{B}}{2}\right) + \kappa\mathbf{m} \tag{12}$$

$$\mathbf{x} = \left(\frac{\begin{pmatrix}2\\3\\4\end{pmatrix} + \begin{pmatrix}4\\5\\8\end{pmatrix}}{2}\right) + \kappa\mathbf{m} \tag{13}$$

$$\mathbf{x} = \begin{pmatrix}3\\4\\6\end{pmatrix} + \kappa\begin{pmatrix}2\\3\\6\end{pmatrix} \tag{14}$$

C Code

```c
#include <stdio.h>
void midpoint(double *A, double *B, double *h, int n)
{
    for(int i=0; i<n; i++)
    {
        h[i] = (A[i] + B[i]) / 2;
    }
}

void cross_product(double *m1, double *m2, double *m)
{
    m[0] = m1[1]*m2[2] - m1[2]*m2[1];
    m[1] = m1[2]*m2[0] - m1[0]*m2[2];
    m[2] = m1[0]*m2[1] - m1[1]*m2[0];
}
```

C Code

```c
void compute_line(double *x, double *y, double *z,double *h,
    double *m, int n)
{
    for(int i=0; i<n; i++)
    {
        double k = (i - n/2);
        x[i] = h[0] + k*m[0];
        y[i] = h[1] + k*m[1];
        z[i] = h[2] + k*m[2];
    }
}
```

Python Code

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

    mylib = ctypes.CDLL(r"D:/Matgeo/4.4.38/codes/mylib.so")
# Define function prototypes for type safety
ND_POINTER = np.ctypeslib.ndpointer(dtype=np.float64, flags="
    C_CONTIGUOUS")

mylib.midpoint.argtypes = [ND_POINTER, ND_POINTER, ND_POINTER,
    ctypes.c_int]
mylib.midpoint.restype = None

mylib.compute_line.argtypes = [ND_POINTER, ND_POINTER, ND_POINTER
    ,
                               ND_POINTER, ND_POINTER, ctypes.c_int]
mylib.compute_line.restype = None
```

Python Code

```
# --- Input Data based on the problem ---
A = np.array([2.0, 3.0, 4.0])
B = np.array([4.0, 5.0, 8.0])

p1 = np.array([8.0, -19.0, 10.0])
m1 = np.array([3.0, -16.0, 7.0])

p2 = np.array([15.0, 29.0, 5.0])
m2 = np.array([3.0, 8.0, -5.0])

# --- Calculations ---
# Calculate the midpoint h using the C function
h = np.zeros(3, dtype=np.float64)
mylib.midpoint(A, B, h, 3) # h is now [3., 4., 6.]

# The direction vector from our theoretical solution
m_solution_simplified = np.array([2.0, 3.0, 6.0])
```

Python Code

```
# --- Generate Line Points using C function ---
n = 100
# (Scaling factors are used to control visual line length)
scale_solution = 7.0 / (n / 2.0)
scale_given = 1.5 / (n / 2.0)

m_solution_scaled = m_solution_simplified * scale_solution
m1_scaled = m1 * scale_given
m2_scaled = m2 * scale_given

x_sol, y_sol, z_sol = [np.zeros(n) for _ in range(3)]
x1, y1, z1 = [np.zeros(n) for _ in range(3)]
x2, y2, z2 = [np.zeros(n) for _ in range(3)]

mylib.compute_line(x_sol, y_sol, z_sol, h, m_solution_scaled, n)
mylib.compute_line(x1, y1, z1, p1, m1_scaled, n)
mylib.compute_line(x2, y2, z2, p2, m2_scaled, n)
```

Python Code

```
# --- Plotting ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection="3d")

# Plot the elements
ax.scatter(h[0], h[1], h[2], color='green', s=150,
           label='Midpoint h (3, 4, 6)', zorder=5)
ax.plot(x1, y1, z1, color='orange', linewidth=2, label='Given
    Line 1')
ax.plot(x2, y2, z2, color='purple', linewidth=2, label='Given
    Line 2')
ax.plot(x_sol, y_sol, z_sol, color='red', linewidth=3,
        label='Required Line (Solution)')
```

Python Code

```
# --- Formatting the plot ---
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("Line Bisecting a Segment and Perpendicular to Two
    Lines")
ax.set_xlim([-15, 25]); ax.set_ylim([-45, 45]); ax.set_zlim([-40,
     50])
ax.legend(loc='upper left')
ax.grid(True)
ax.set_box_aspect([1, 1, 1]) # Equal aspect ratio

plt.show()
```

Plot

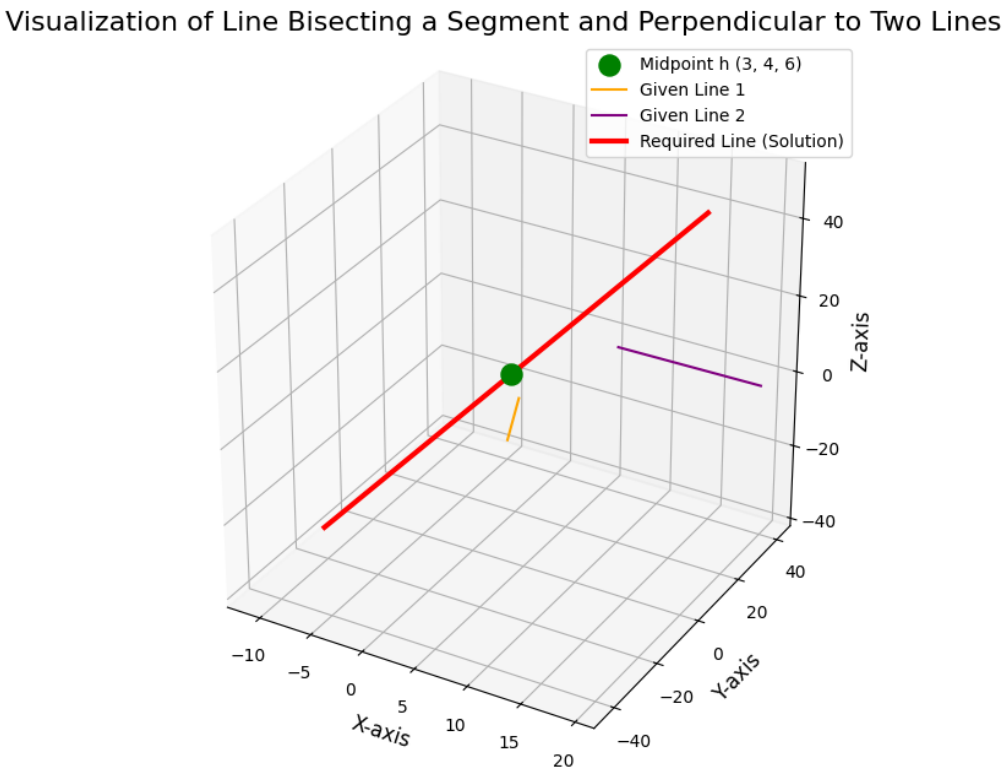

Figure: Visualization of the solution.