4.13.85-Beamer

Varun-ai25btech11016

September 12, 2025

Question

If the lines $\frac{x-1}{2} = \frac{y+1}{3} = \frac{z-1}{4}$ and $\frac{x-3}{1} = \frac{y-k}{2} = \frac{z}{1}$ intersect, then the value of k is

Theoretical solution

The lines $A + K_1 m_1, B + K_2 m_2$ will intersect if

$$\text{rank}(\mathbf{M} \quad \mathbf{B} - \mathbf{A}) = 2 \tag{1}$$

$$\mathbf{M} = \begin{pmatrix} m_1 & m_2 \end{pmatrix} \tag{2}$$

$$\tag{3}$$

Here,

$$\mathbf{m_1} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix} \mathbf{m_2} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \tag{4}$$

Theoretical solution

$$\mathbf{M} = \begin{pmatrix} 2 & 1 \\ 3 & 2 \\ 4 & 1 \end{pmatrix} \tag{5}$$

$$\mathbf{A} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 3 \\ k \\ 0 \end{pmatrix} \tag{6}$$

$$\mathbf{B} - \mathbf{A} = \begin{pmatrix} 3-1 \\ k-(-1) \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ k+1 \\ -1 \end{pmatrix} \tag{7}$$

$$\text{rank}\left(\begin{pmatrix} 2 & 1 & 2 \\ 3 & 2 & k+1 \\ 4 & 1 & -1 \end{pmatrix}\right) = 2 \tag{8}$$

Theoretical solution

$$\begin{pmatrix} 2 & 1 & 2 \\ 3 & 2 & k+1 \\ 4 & 1 & -1 \end{pmatrix} \xrightarrow[R_3 \to 2R_3 - 4R_1]{R_2 \to 2R_2 - 3R_1} \begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 2k-4 \\ 0 & -2 & -10 \end{pmatrix} \tag{9}$$

$$\tag{10}$$

$$\xrightarrow{R_3 \to R_3 + 2R_2} \begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 2k-4 \\ 0 & 0 & 4k-18 \end{pmatrix} \tag{11}$$

For therank($\mathbf{M} \quad \mathbf{B} - \mathbf{A}$) to be 2
the last row must be all zero implies

$$4k - 18 = 0 \tag{12}$$

$$k = \frac{9}{2} \tag{13}$$

C code

```c
    #include <stdlib.h>

// Fill arrays with coordinates of Line 1: (1+2t, -1+3t, 1+4t)
// and Line 2: (3+t, k+2t, t)
// Inputs: t_min, t_max, n_points, k
// Outputs: arrays (x1,y1,z1,x2,y2,z2)
void generate_lines(double t_min, double t_max, int n_points,
    double k,
                    double *x1, double *y1, double *z1,
                    double *x2, double *y2, double *z2) {
    double step = (t_max - t_min) / (n_points - 1);

    for (int i = 0; i < n_points; i++) {
        double t = t_min + i * step;
```

C code

```
        // Line 1
        x1[i] = 1 + 2*t;
        y1[i] = -1 + 3*t;
        z1[i] = 1 + 4*t;

        // Line 2
        x2[i] = 3 + t;
        y2[i] = k + 2*t;
        z2[i] = t;
    }
}
```

Py with C code

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

# Load the shared library
lib = ctypes.CDLL(./liblines.so)

# Define function signature
lib.generate_lines.argtypes = [
    ctypes.c_double, ctypes.c_double, ctypes.c_int, ctypes.
        c_double,
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
        C_CONTIGUOUS),
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
        C_CONTIGUOUS),
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
        C_CONTIGUOUS),
```

Py with C code

```
np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
    C_CONTIGUOUS),
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
        C_CONTIGUOUS),
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags=
        C_CONTIGUOUS),
]
lib.generate_lines.restype = None

# Parameters
n_points = 200
t_min, t_max = -10, 10
k = 2.0 # change this as needed
```

Py with C code

```
# Allocate numpy arrays
x1 = np.zeros(n_points, dtype=np.float64)
y1 = np.zeros(n_points, dtype=np.float64)
z1 = np.zeros(n_points, dtype=np.float64)
x2 = np.zeros(n_points, dtype=np.float64)
y2 = np.zeros(n_points, dtype=np.float64)
z2 = np.zeros(n_points, dtype=np.float64)
```

Py with C code

```
# Call C function
lib.generate_lines(t_min, t_max, n_points, k, x1, y1, z1, x2, y2,
    z2)

# Plot 3D
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection=3d)

ax.plot(x1, y1, z1, label=Line 1, color=blue)
ax.plot(x2, y2, z2, label=fLine 2 (k={k}), color=orange)
```

Py with C code

```
ax.scatter(x1[0], y1[0], z1[0], color=red, s=50, label=Start Line
    1)
ax.scatter(x2[0], y2[0], z2[0], color=green, s=50, label=Start
    Line 2)

ax.set_xlabel(X-axis)
ax.set_ylabel(Y-axis)
ax.set_zlabel(Z-axis)
ax.set_title(3D Plot of Two Lines (from C library))
ax.legend()
plt.show()
```

Py code

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

# Parameter range
t = np.linspace(-10, 10, 200)

# First line: (x, y, z) = (1+2, -1+3, 1+4)
x1 = 1 + 2*t
y1 = -1 + 3*t
z1 = 1 + 4*t

# Second line: (x, y, z) = (3+, k+2, )
# Example: set k = 2 (you can change it to the value you solve
    for)
k = 2
x2 = 3 + t
y2 = k + 2*t
z2 = t
```

Py code

```
# Create 3D plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Plot lines
ax.plot(x1, y1, z1, label=Line 1, color=blue)
ax.plot(x2, y2, z2, label=fLine 2 (k={k}), color=orange)

# Mark reference points
ax.scatter(1, -1, 1, color='red', s=50, label=Point on Line 1)
ax.scatter(3, k, 0, color='green', s=50, label=Point on Line 2)
```

Py code

```
# Labels & title
ax.set_xlabel(X-axis)
ax.set_ylabel(Y-axis)
ax.set_zlabel(Z-axis)
ax.set_title(3D Plot of Two Lines)
ax.legend()
plt.show()
```

plot

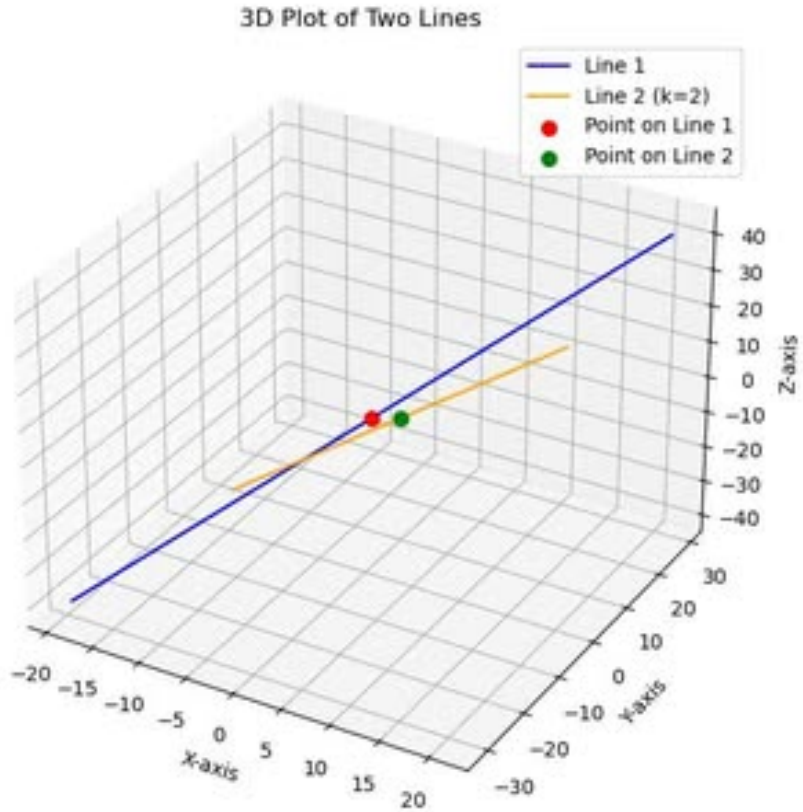
3D Plot of Two Lines
Line 1
Line 2 (k=2)
Point on Line 1
Point on Line 2
X-axis
Y-axis
Z-axis

Figure: