4.7.54

Rushil Shanmukha Srinivas
EE25BTECH11057
Electrical Enggineering ,
IIT Hyderabad.

September 15, 2025

Problem Statement

Question : Find the vector equations of the line passing through the point (1,2,-4) and perpendicular to the two lines $\frac{x-8}{3} = \frac{y+19}{-16} = \frac{z-10}{7}$ and $\frac{x-15}{3} = \frac{y-29}{8} = \frac{z-5}{-5}$.

Equation of Line in Vector form

Solution : Given the line passes through the point

$$\mathbf{A} = \begin{pmatrix} 1 \\ 2 \\ -4 \end{pmatrix} \tag{3.1}$$

The line is also perpendicular to

$$\frac{x-8}{3} = \frac{y+19}{-16} = \frac{z-10}{7}, \frac{x-15}{3} = \frac{y-29}{8} = \frac{z-5}{-5}. \tag{3.2}$$

The direction vector of the line is given by

$$\begin{pmatrix} 3 & -16 & 7 \\ 3 & 8 & -5 \end{pmatrix} \mathbf{m} = 0 \xleftrightarrow{R_2 \longrightarrow R_2 - R_1} \begin{pmatrix} 3 & -16 & 7 \\ 0 & 24 & -12 \end{pmatrix} \tag{3.3}$$

$$\begin{pmatrix} 3 & -16 & 7 \\ 0 & 24 & -12 \end{pmatrix} \xleftrightarrow{R_1 \longrightarrow 3R_1 + 2R_2} \begin{pmatrix} 9 & 0 & -3 \\ 0 & 24 & -12 \end{pmatrix} \tag{3.4}$$

$$\begin{pmatrix} 9 & 0 & -3 \\ 0 & 24 & -12 \end{pmatrix} \xleftrightarrow{R_1 \longrightarrow \frac{R_1}{9}} \begin{pmatrix} 1 & 0 & \frac{-1}{3} \\ 0 & 24 & -12 \end{pmatrix} \tag{3.5}$$

$$\begin{pmatrix} 1 & 0 & \frac{-1}{3} \\ 0 & 24 & -12 \end{pmatrix} \overleftrightarrow{R_2 \longrightarrow \frac{R_2}{24}} \begin{pmatrix} 1 & 0 & \frac{-1}{3} \\ 0 & 1 & \frac{-1}{2} \end{pmatrix} \tag{3.6}$$

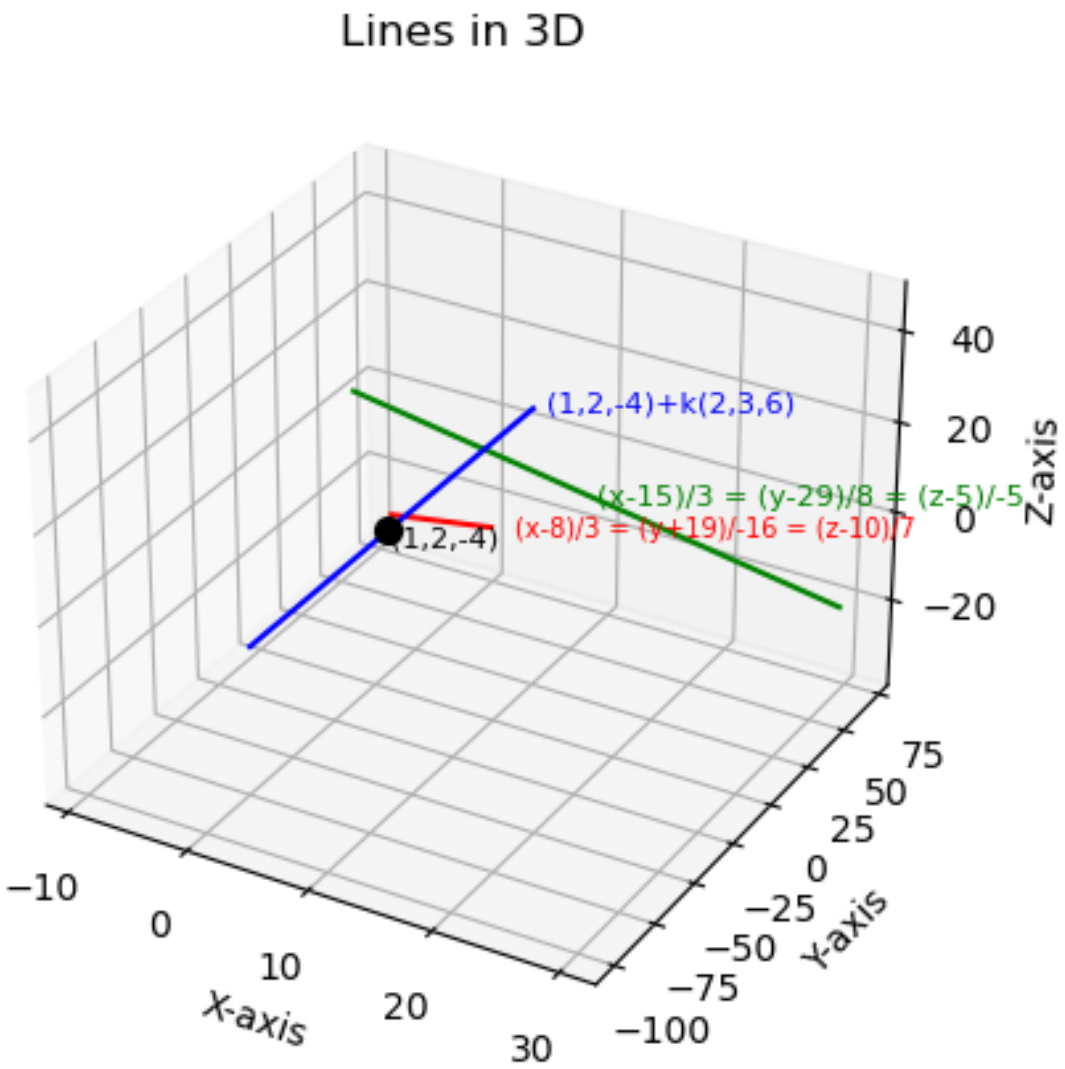
So

$$\mathbf{m} = \begin{pmatrix} 2 \\ 3 \\ 6 \end{pmatrix} = \textit{Direction vector of line} \tag{3.7}$$

The vector equation of the line is

$$\mathbf{L_1} = \mathbf{A} + k\mathbf{m} = \begin{pmatrix} 1 \\ 2 \\ -4 \end{pmatrix} + k \begin{pmatrix} 2 \\ 3 \\ 6 \end{pmatrix} \tag{3.8}$$

Plots


Lines in 3D
(1,2,-4)+k(2,3,6)
(x-15)/3 = (y-29)/8 = (z-5)/-5
(x-8)/3 = (y+19)/-16 = (z-10)/7
(1,2,-4)
40
20
0
−20
Z-axis
75
50
25
0
−25
−50
−75
−100
Y-axis
−10
0
10
20
30
X-axis

C Code

```c
#include <stdio.h>
// Function to compute cross product of two 3D vectors
void cross_product(double a[3], double b[3], double result[3]) {
    result[0] = a[1]*b[2] - a[2]*b[1];
    result[1] = a[2]*b[0] - a[0]*b[2];
    result[2] = a[0]*b[1] - a[1]*b[0];}

// Function to compute line equation point for given t
void line_equation(double t, double *x, double *y, double *z) {
    // Given point (1,2,-4)
    double p[3] = {1, 2, -4};
    // Given direction vectors of the two lines
    double d1[3] = {3, -16, 7};
    double d2[3] = {3, 8, -5};

    // Direction of required line = cross product of d1 and d2
    double d[3];
```

```
    cross_product(d1, d2, d);
    // Simplify direction vector by dividing by GCD factor if desired
    // For this case, (24,36,72)  (2,3,6)
    d[0] = 2; d[1] = 3; d[2] = 6;

    // Parametric equation
    *x = p[0] + d[0]*t;
    *y = p[1] + d[1]*t;
    *z = p[2] + d[2]*t;}

// Convenience function for printing
void print_line(double t) {
    double x, y, z;
    line_equation(t, &x, &y, &z);
    printf("For t=%.2f -> (x, y, z) = (%.2f, %.2f, %.2f)\n", t, x, y, z);
}
```

```
// Main function (for standalone execution)
int main() {
    printf("Line passing through (1,2,-4) and perpendicular to given lines
        :\n");
    printf("Vector form: r(t) = (1,2,-4) + t(2,3,6)\n\n");

    // Print some sample points
    for (int i = 0; i <= 5; i++) {
        print_line((double)i);
    }
    return 0;
}
```

Python : call_c.py

```
import ctypes
# Load the shared library compiled from line.c
# Make sure libline.so is in the same folder
lib = ctypes.CDLL("./libline.so")
# Setup the function signatures
# void line_equation(double t, double *x, double *y, double *z)
lib.line_equation.argtypes = [
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)]
lib.line_equation.restype = None

# void print_line(double t)
lib.print_line.argtypes = [ctypes.c_double]
lib.print_line.restype = None
```

```
# Example usage of the functions
if __name__ == "__main__":
    # Use line_equation() to get coordinates back into Python
    t = 2.0
    x = ctypes.c_double()
    y = ctypes.c_double()
    z = ctypes.c_double()
    lib.line_equation(t, ctypes.byref(x), ctypes.byref(y), ctypes.byref(z))
    print(f"Using line_equation from C: t={t} -> (x, y, z) = ({x.value},
        {y.value}, {z.value})")

    # Use print_line() to let C handle printing
    print("\nCalling print_line from C:")
    lib.print_line(3.0)
    # Generate multiple points
    print("\nGenerating multiple points using print_line:")
    for i in range(6):
        lib.print_line(float(i))
```

Python Code for Plotting

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Given lines direction vectors
d1 = np.array([3, -16, 7])
d2 = np.array([3, 8, -5])
p1 = np.array([8, -19, 10]) # point on line 1
p2 = np.array([15, 29, 5]) # point on line 2
# New line
p0 = np.array([1, 2, -4])
d = np.array([2, 3, 6])
# Parameter ranges
t = np.linspace(-5, 5, 100)

# Line equations
line1 = p1[:, None] + d1[:, None]*t
line2 = p2[:, None] + d2[:, None]*t
```

```
line3 = p0[:, None] + d[:, None]*t
# Plotting
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(line1[0], line1[1], line1[2], color='r')
ax.plot(line2[0], line2[1], line2[2], color='g')
ax.plot(line3[0], line3[1], line3[2], color='b')

# Mark the point (1,2,-4)
ax.scatter(p0[0], p0[1], p0[2], color='k', s=50)
ax.text(p0[0], p0[1], p0[2]-4, "(1,2,-4)", color='k', fontsize=8) # below
    the point
# Labels beside the lines
# Red line label -> shifted outward in x, but moved a bit downward
ax.text(line1[0][-1]+2, line1[1][-1]-2, line1[2][-1],
        "(x-8)/3 = (y+19)/-16 = (z-10)/7", color='r', fontsize=7)
# Green line label -> shifted upward compared to last version
mid_idx = len(t)//2
```

```
ax.text(line2[0][mid_idx], line2[1][mid_idx]-3, line2[2][mid_idx],
        "(x-15)/3 = (y-29)/8 = (z-5)/-5", color='g', fontsize=8)

# Blue line label
ax.text(line3[0][-1]+1, line3[1][-1], line3[2][-1],
        "(1,2,-4)+k(2,3,6)", color='b', fontsize=8)
# Formatting
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("Lines in 3D")

plt.savefig("../figs/fig7.png")
plt.show()
```