4.11.14

M Chanakya Srinivas- EE25BTECH11036

Problem Statement

Find the value of λ for which the following lines are perpendicular. Determine whether the lines intersect or not.

$$\frac{x-5}{5\lambda+2} = \frac{2-y}{5} = \frac{1-z}{-1}, \tag{1}$$

$$\frac{x}{1} = \frac{y+\frac{1}{2}}{2\lambda} = \frac{z-1}{3}. \tag{2}$$

Step 1: Vector form of lines

Choose points and direction vectors:

$$\mathbf{A}_1 = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}, \qquad \mathbf{m}_1 = \begin{pmatrix} 5\lambda + 2 \\ -5 \\ 1 \end{pmatrix}, \tag{3}$$

$$\mathbf{A}_2 = \begin{pmatrix} 0 \\ -1/2 \\ 1 \end{pmatrix}, \qquad \mathbf{m}_2 = \begin{pmatrix} 1 \\ 2\lambda \\ 3 \end{pmatrix}. \tag{4}$$

Lines in parametric form:

$$\mathbf{r}_1 = \mathbf{A}_1 + \kappa_1 \mathbf{m}_1, \tag{5}$$

$$\mathbf{r}_2 = \mathbf{A}_2 + \kappa_2 \mathbf{m}_2. \tag{6}$$

Step 2: Perpendicularity condition

$$\mathbf{m}_1^\top \mathbf{m}_2 = 0$$

Compute dot product:

$$\begin{aligned}
\mathbf{m}_1^\top \mathbf{m}_2 &= \begin{pmatrix} 5\lambda + 2 & -5 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2\lambda \\ 3 \end{pmatrix} && (7) \\
&= 5\lambda + 2 - 10\lambda + 3 && (8) \\
&= -5\lambda + 5 && (9)
\end{aligned}$$

$$-5\lambda + 5 = 0 \quad \Rightarrow \quad \boxed{\lambda = 1}$$

Step 3: Intersection condition

Lines intersect if

$$\mathbf{r}_1 = \mathbf{r}_2 \quad \Rightarrow \quad \kappa_1 \mathbf{m}_1 - \kappa_2 \mathbf{m}_2 = \mathbf{A}_2 - \mathbf{A}_1$$

Define

$$M(\lambda) = \begin{bmatrix} \mathbf{m}_1 & -\mathbf{m}_2 \end{bmatrix} = \begin{pmatrix} 5\lambda + 2 & -1 \\ -5 & -2\lambda \\ 1 & -3 \end{pmatrix}, \tag{10}$$

$$\mathbf{z} = \begin{pmatrix} \kappa_1 \\ \kappa_2 \end{pmatrix}, \quad \mathbf{b} = \mathbf{A}_2 - \mathbf{A}_1 = \begin{pmatrix} -5 \\ -5/2 \\ 0 \end{pmatrix}. \tag{11}$$

Then

$$M(\lambda)\mathbf{z} = \mathbf{b}$$

Step 4: Augmented matrix and row reduction

Augmented matrix:

$$\left[\begin{array}{cc|c} 5\lambda+2 & -1 & -5 \\ -5 & -2\lambda & -5/2 \\ 1 & -3 & 0 \end{array}\right]$$

Use row 3 to eliminate κ_1:

$$1 \cdot \kappa_1 - 3 \cdot \kappa_2 = 0 \quad \Rightarrow \quad \kappa_1 = 3\kappa_2$$

Substitute $\kappa_1 = 3\kappa_2$:

$$(15\lambda + 5)\kappa_2 = -5, \tag{12}$$

$$(-15 - 2\lambda)\kappa_2 = -5/2 \tag{13}$$

Step 5: Consistency condition

System consistent if:

$$\frac{-5}{15\lambda + 5} = \frac{-5/2}{-15 - 2\lambda} \quad \Rightarrow \quad -\frac{1}{3\lambda + 1} = \frac{5}{30 + 4\lambda}$$

Solve:

$$-(30 + 4\lambda) = 5(3\lambda + 1) \tag{14}$$

$$-30 - 4\lambda = 15\lambda + 5 \tag{15}$$

$$19\lambda = -35 \tag{16}$$

$$\boxed{\lambda} = -\frac{35}{19} \tag{17}$$

Step 6: Conclusions (Chapter 4 style)

- Perpendicularity occurs at $\lambda = 1$. At this value, $\text{rank}[M(1) \mid \mathbf{b}] > \text{rank}(M(1))$, so lines are *skew*.
- Intersection occurs at $\lambda = -35/19$. At this value, $\text{rank}[M(\lambda) \mid \mathbf{b}] = \text{rank}(M(\lambda))$, so lines intersect (but are not perpendicular).

C code

```
#include <math.h>
// Return lambda for which the two lines are perpendicular.
double perpendicular_lambda(void) {
    return 1.0;
}
// Return lambda for which the two lines intersect.
double intersection_lambda(void) {
    return -35.0/19.0;
}
// For a given lambda, check whether the two lines intersect.
// Writes s (parameter of line2) and t (parameter of line1) to the output pointers.
// Returns 1 if intersection exists (within tolerance), otherwise 0.
int lines_intersection_params(double lambda, double *s_out, double *t_out) {
    const double EPS = 1e-9;
    double denom1 = 1.0 - 3.0*(5.0*lambda + 2.0);
    if (fabs(denom1) < EPS) return 0;
```

C code

```c
    double s1 = 5.0/denom1;
    double denom2 = 30.0 + 4.0*lambda;
    if (fabs(denom2) < EPS) return 0;
    double s2 = 5.0/denom2;
    if (fabs(s1 - s2) > 1e-6) return 0;
    double s = 0.5*(s1 + s2);
    double t = 3.0*s;
    if (s_out) *s_out = s;
    if (t_out) *t_out = t;
    return 1;
}
```

Python code through shared output

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

# Load shared library
lib = ctypes.CDLL('./liblines.so')
lib.perpendicular_lambda.restype = ctypes.c_double
lib.intersection_lambda.restype = ctypes.c_double
lib.lines_intersection_params.restype = ctypes.c_int
lib.lines_intersection_params.argtypes = [
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
```

Python code through shared output

```
# 1. Get lambdas
lam_perp = lib.perpendicular_lambda()
lam_inter = lib.intersection_lambda()

print(fLambda for perpendicular lines: {lam_perp} = {Fraction(
    lam_perp).limit_denominator()})
print(fLambda for intersection: {lam_inter} = {Fraction(lam_inter
    ).limit_denominator()})

# 2. Check intersection at perpendicular lambda
s1 = ctypes.c_double()
t1 = ctypes.c_double()
intersects_perp = lib.lines_intersection_params(lam_perp, ctypes.
    byref(s1), ctypes.byref(t1))

# 3. Check intersection at intersection lambda
s2 = ctypes.c_double()
t2 = ctypes.c_double()
intersects = lib.lines_intersection_params(lam_inter, ctypes.
```

Python code through shared output

```
if intersects_perp:
    print(Lines intersect when perpendicular.)
else:
    print( Lines do NOT intersect when they are perpendicular.)

if intersects:
    print( Lines intersect when  =, lam_inter)
    print(Intersection parameters: s =, s2.value, , t =, t2.value
        )
    inter_point = np.array([
        s2.value,
        -0.5 + 2*lam_inter*s2.value,
        1 + 3*s2.value
    ])
    print(Intersection point:, inter_point)
else:
    inter_point = None
    print( Lines do NOT intersect for intersection .)
```

Python code through shared output

```
# 4. Plot
t_vals = np.linspace(-2, 2, 100)
x1 = 5 + (5*lam_inter + 2)*t_vals
y1 = 2 - 5*t_vals
z1 = 1 + t_vals

s_vals = np.linspace(-2, 2, 100)
x2 = s_vals
y2 = -0.5 + 2*lam_inter*s_vals
z2 = 1 + 3*s_vals

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot lines
ax.plot(x1, y1, z1, label=Line 1, color='blue', linewidth=2)
ax.plot(x2, y2, z2, label=Line 2, color='red', linewidth=2)
```

```
# Plot intersection
if inter_point is not None:
    ax.scatter(*inter_point, color='green', s=80, edgecolors='black', label=Intersection Point)
    ax.text(*inter_point + 0.3,
            f({inter_point[0]:.2f}, {inter_point[1]:.2f}, {inter_point[2]:.2f}),
            fontsize=10, color='green')
```

Python code through shared output

```
# Labels and legend
ax.set_xlabel(X Axis, fontsize=12)
ax.set_ylabel(Y Axis, fontsize=12)
ax.set_zlabel(Z Axis, fontsize=12)
ax.set_title(3D Plot: Intersection of Two Lines, fontsize=14)
ax.legend()
ax.grid(True)
ax.view_init(elev=25, azim=135)

plt.tight_layout()
plt.show()
```

Only Python code

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

# Local imports (assuming these scripts are available and in
    PYTHONPATH)
from line.funcs import *
from triangle.funcs import *
from conics.funcs import circ_gen

# Given intersection parameter lambda (from your C-types code or
    calculation)
lam = -35/19 # approx -1.8421

# Calculate intersection parameters s and t for the lines
# The two parametric lines from your problem:
```

Only Python code

```
# Line 1:
# x = 5 + (5*lam + 2)*t
# y = 2 - 5*t
# z = 1 + t

# Line 2:
# x = s
# y = -0.5 + 2*lam*s
# z = 1 + 3*s

# We want to find s, t such that both lines intersect

# From parametric equality:
# 5 + (5*lam + 2)*t = s ...(1)
# 2 - 5*t = -0.5 + 2*lam*s ...(2)
# 1 + t = 1 + 3*s ...(3)

# Solve (3):
# t = 3*s
```

Only Python code

```
# Substitute into (1):
# 5 + (5*lam + 2)*3*s = s
# 5 + 3*(5*lam + 2)*s = s
# 5 = s - 3*(5*lam + 2)*s = s*(1 - 3*(5*lam + 2))
# s = 5 / (1 - 3*(5*lam + 2))

denom = 1 - 3*(5*lam + 2)
s = 5 / denom

# Then t = 3*s
t = 3*s

# Calculate intersection point from line 2 (or line 1)
inter_point = np.array([
    s,
    -0.5 + 2*lam*s,
    1 + 3*s
])
```

Only Python code

```
print(fIntersection parameters: s = {s:.5f}, t = {t:.5f})
print(fIntersection Point: ({inter_point[0]:.5f}, {inter_point
    [1]:.5f}, {inter_point[2]:.5f}))

# Define parametric lines for plotting near intersection point

def line1(t_vals):
    x = 5 + (5*lam + 2)*t_vals
    y = 2 - 5*t_vals
    z = 1 + t_vals
    return x, y, z

def line2(s_vals):
    x = s_vals
    y = -0.5 + 2*lam*s_vals
    z = 1 + 3*s_vals
    return x, y, z
```

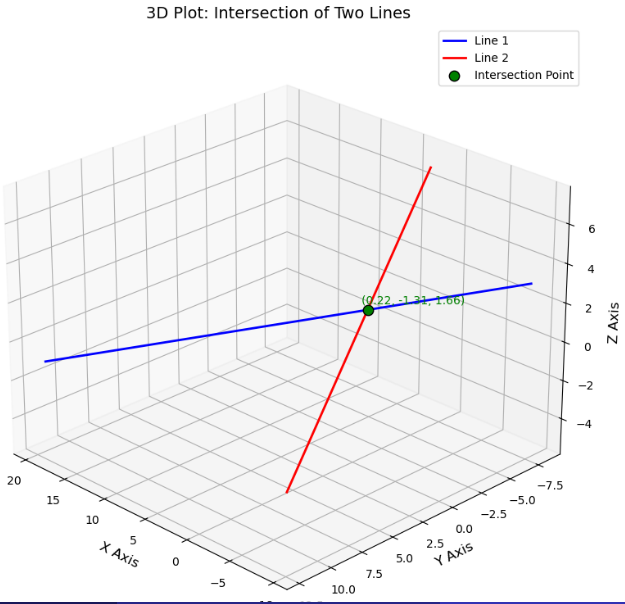
Only Python code

```
# Plot ranges close to intersection
t_vals = np.linspace(t - 1, t + 1, 100)
s_vals = np.linspace(s - 1, s + 1, 100)

x1, y1, z1 = line1(t_vals)
x2, y2, z2 = line2(s_vals)

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

ax.plot(x1, y1, z1, label='Line 1', color='blue', linewidth=2)
ax.plot(x2, y2, z2, label='Line 2', color='red', linewidth=2)

# Mark intersection
ax.scatter(*inter_point, color='green', s=80, label='Intersection
    Point')
```

Only Python code

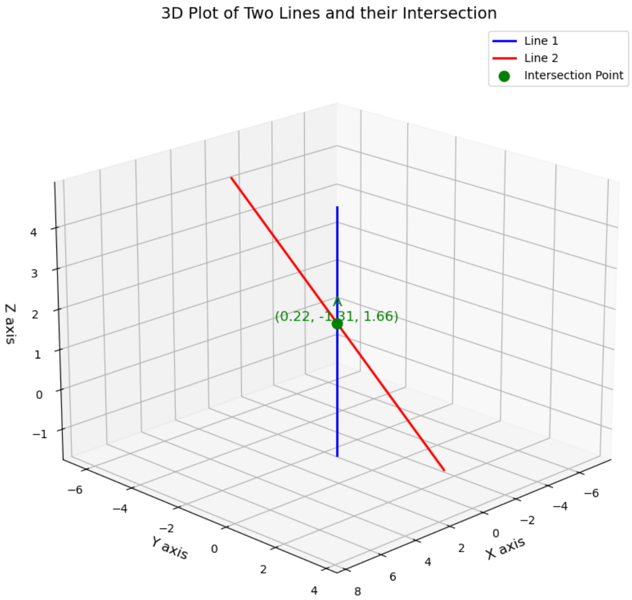
```
# Annotate intersection point
ax.text(inter_point[0], inter_point[1], inter_point[2],
       fA\n({inter_point[0]:.2f}, {inter_point[1]:.2f}, {
           inter_point[2]:.2f}),
       color='green', fontsize=12, ha='center', va='bottom')

# Axis labels and title
ax.set_xlabel(X axis, fontsize=12)
ax.set_ylabel(Y axis, fontsize=12)
ax.set_zlabel(Z axis, fontsize=12)
ax.set_title(3D Plot of Two Lines and their Intersection,
    fontsize=14)

ax.legend()
ax.grid(True)
ax.view_init(elev=20, azim=45)

plt.tight_layout()
plt.show()
```

PLOTS



3D Plot: Intersection of Two Lines
Line 1
Line 2
Intersection Point
(0.22, -1.31, 1.66)
X Axis
Y Axis
Z Axis
20
15
10
5
0
−5
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
6
4
2
0
−2
−4

PLOTS



3D Plot of Two Lines and their Intersection
Line 1
Line 2
Intersection Point
A
(0.22, -1.31, 1.66)
Z axis
Y axis
X axis