Matgeo Presentation - Problem 2.5.19

ee25btech11056 - Suraj.N

August 27, 2025

Problem Statement

Question : Find the value of p for which the lines $\dfrac{1-x}{3} = \dfrac{2y-14}{2p} = \dfrac{z-3}{2}$ and $\dfrac{1-x}{3p} = \dfrac{y-5}{1} = \dfrac{6-z}{5}$ are perpendicular.

Symbol	**Line**
A	$\dfrac{1-x}{3} = \dfrac{2y-14}{2p} = \dfrac{z-3}{2}$
B	$\dfrac{1-x}{3p} = \dfrac{y-5}{1} = \dfrac{6-z}{5}$

Table : Lines

Solution

These lines can also be written in the vector form $\mathbf{x} = \mathbf{h} + k\mathbf{m}$.

$$\text{Line A:} \quad \mathbf{x} = \begin{pmatrix} 1 \\ 7 \\ 3 \end{pmatrix} + k_1 \begin{pmatrix} -3 \\ p \\ 2 \end{pmatrix}$$

$$\text{Line B:} \quad \mathbf{x} = \begin{pmatrix} 1 \\ 5 \\ 6 \end{pmatrix} + k_2 \begin{pmatrix} -3p \\ 1 \\ -5 \end{pmatrix}$$

Hence, the direction vectors are

$$\mathbf{m_1} = \begin{pmatrix} -3 \\ p \\ 2 \end{pmatrix}, \qquad \mathbf{m_2} = \begin{pmatrix} -3p \\ 1 \\ -5 \end{pmatrix}.$$

For the lines to be perpendicular, we require $\mathbf{m_1}^\top \mathbf{m_2} = 0$.

$$\begin{aligned}
\mathbf{m_1}^\top \mathbf{m_2} &= \begin{pmatrix} -3 & p & 2 \end{pmatrix} \begin{pmatrix} -3p \\ 1 \\ -5 \end{pmatrix} \\
&= (-3)(-3p) + p(1) + 2(-5) \\
&= 9p + p - 10 \\
&= 10p - 10.
\end{aligned}$$

Thus,

$$10p - 10 = 0 \;\Rightarrow\; p = 1.$$

Final answer: $p = 1$.

Plot

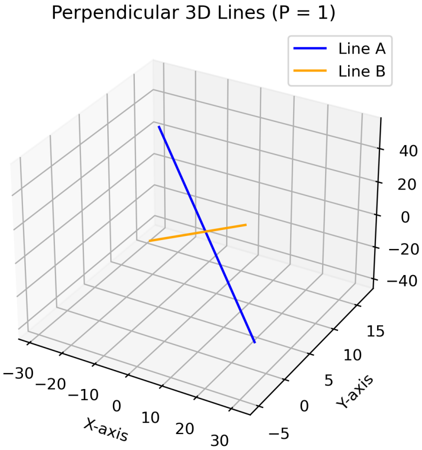

Fig : Lines A and B

C Code: points.c

```
#include <math.h>
#include <stdio.h>

// Return the dot product instead of p
double product(double p) {
  double dot = 0;
  double a[3] = {-3, p, 2};
  double b[3] = {-3 * p, 1, -5};

  for (int i = 0; i < 3; i++) {
    dot += a[i] * b[i];
  }
  return dot; // return dot product
}
```

Python: call_c.py

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

# Load shared library
lib = ctypes.CDLL("./points.so")
lib.product.restype = ctypes.c_double
lib.product.argtypes = [ctypes.c_double]

solution_p = None
for p in range(-10, 11):
    dot = lib.product(ctypes.c_double(p))
    if abs(dot) < 1e-6: # check near zero
        solution_p = p
        print(f"Solution found: p = {p}")
        break

if solution_p is None:
    print("No solution found")
    sys.exit(0)

p = solution_p

# Parametric equations
t = np.linspace(-10, 10, 100)
x1 = 1 - 3*t
y1 = 7 + p*t
z1 = 3 + 2*t
```

Python: call_c.py

```
s = np.linspace(-10, 10, 100)
x2 = 1 - 3*p*s
y2 = 5 + s
z2 = 6 - 5*s

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

ax.plot(x1, y1, z1, label="Line␣A", color="blue")
ax.plot(x2, y2, z2, label="Line␣B", color="orange")

ax.set_title("Perpendicular␣3D␣Lines␣(P␣=␣1)")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.legend()

# Save the figure
plt.savefig("perpendicular_lines.png", dpi=300, bbox_inches="tight")

# Show the plot
plt.show()
```

Python: plot.py

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

# Directly use the analytical solution
p = 1

# Parametric equations for Line 1 and Line 2
t = np.linspace(-10, 10, 200)
x1 = 1 - 3*t
y1 = 7 + p*t
z1 = 3 + 2*t

s = np.linspace(-10, 10, 200)
x2 = 1 - 3*p*s
y2 = 5 + s
z2 = 6 - 5*s

# Plotting
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')

ax.plot(x1, y1, z1, label="Line␣A", color="blue", linewidth=2)
ax.plot(x2, y2, z2, label="Line␣B", color="orange", linewidth=2)

ax.set_title("Perpendicular␣3D␣Lines␣(p␣=␣1)")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.legend()

# Save the figure
plt.savefig("perpendicular_lines.png", dpi=300, bbox_inches="tight")
```