

Problem 2.4.18

ee25btech11023-Venkata Sai

August 26, 2025

Problem Statement

Find the values of **p** so that the lines $\frac{1-x}{3} = \frac{7y-14}{2p} = \frac{z-3}{2}$ and $\frac{7-7x}{3p} = \frac{y-5}{1} = \frac{6-z}{5}$ are at right angles.

| Variable | Description |
|----------|----------------------------|
| m_1 | Direction vector of Line 1 |
| m_2 | Direction vector of line 2 |

Table: Variables given

Requirement

To show that two lines are at right angles

$$(m_1)^\top (m_2) = 0 \quad (3.1)$$

Transformation of given lines

Line 1:

$$\frac{1-x}{3} = \frac{7y-14}{2p} = \frac{z-3}{2} \implies \frac{x-1}{-3} = \frac{y-2}{\frac{2p}{7}} = \frac{z-3}{2} \quad (3.2)$$

Line 2:

$$\frac{7-7x}{3p} = \frac{y-5}{1} = \frac{6-z}{5} \implies \frac{x-1}{-\frac{3p}{7}} = \frac{y-5}{1} = \frac{z-6}{-5} \quad (3.3)$$

Direction Vectors

Direction vector for line 1:

$$m_1 = \begin{pmatrix} -3 \\ \frac{2p}{7} \\ 2 \end{pmatrix} \quad (3.4)$$

Direction vector for line 2:

$$m_2 = \begin{pmatrix} -\frac{3p}{7} \\ 1 \\ -5 \end{pmatrix} \quad (3.5)$$

Answer

$$(m_1)^\top (m_2) = 0 \quad (3.6)$$

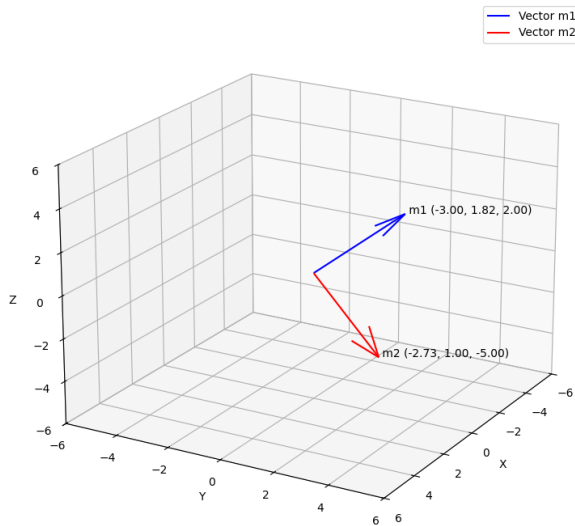
$$\left(-3 \quad \frac{2p}{7} \quad 2\right) \begin{pmatrix} -\frac{3p}{7} \\ 1 \\ -5 \end{pmatrix} = 0 \quad (3.7)$$

$$(-3)\left(-\frac{3p}{7}\right) + \left(\frac{2p}{7}\right)(1) + (2)(-5) = 0 \quad (3.8)$$

$$p = \frac{70}{11} \quad (3.9)$$

Hence the value of **p** is $\frac{70}{11}$

Plot



C Code for Finding p value

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

int main() {

    double p;
    double m1_p_coeffs[3] = {0.0, 2.0/7.0, 0.0};
    double m1_consts[3] = {-3.0, 0.0, 2.0};

    double m2_p_coeffs[3] = {-3.0/7.0, 0.0, 0.0};
    double m2_consts[3] = {0.0, 1.0, -5.0};

    // Calculate A (the total coefficient for 'p') from the dot
    // product expansion.
    double p_coefficient = (m1_consts[0] * m2_p_coeffs[0]) + (
        m1_p_coeffs[0] * m2_consts[0]) + (m1_consts[1] *
        m2_p_coeffs[1]) + (m1_p_coeffs[1] * m2_consts[1]) + (
        m1_consts[2] * m2_p_coeffs[2]) + (m1_p_coeffs[2] *
```

C Code for Finding p value

```
m2_consts[2]);  
// Calculate B (the total constant term) from the dot product  
expansion.  
double constant_term = (m1_consts[0] * m2_consts[0]) +  
                        (m1_consts[1] * m2_consts[1]) +  
                        (m1_consts[2] * m2_consts[2]);  
  
p = -constant_term / p_coefficient;  
  
double m1_final[3];  
double m2_final[3];  
  
m1_final[0] = m1_consts[0] + m1_p_coeffs[0] * p;  
m1_final[1] = m1_consts[1] + m1_p_coeffs[1] * p;
```

C Code for Finding p value

```
m1_final[2] = m1_consts[2] + m1_p_coeffs[2] * p;  
m2_final[0] = m2_consts[0] + m2_p_coeffs[0] * p;  
m2_final[1] = m2_consts[1] + m2_p_coeffs[1] * p;  
m2_final[2] = m2_consts[2] + m2_p_coeffs[2] * p;  
  
FILE *file = fopen("values.dat", "w");  
if (file == NULL) {  
    printf("Error opening file!\n");  
    return 1;  
}
```

C Code for Finding p value

```
fprintf(file, "%.4f %.4f %.4f\n", m1_final[0], m1_final[1],  
      m1_final[2]);  
fprintf(file, "%.4f %.4f %.4f\n", m2_final[0], m2_final[1],  
      m2_final[2]);  
  
fclose(file);  
printf("Direction vectors calculated and written to values.  
      dat\n");  
printf("(The calculated value of p was: %.4f)\n", p);  
  
return 0;  
}
```

Python Code for Plotting

```
# Code by /sdcard/github/matgeo/codes/CoordGeoVV Sharma  
# September 12, 2023  
# Revised July 21, 2024  
# Released under GNU GPL  
# Section Formula  
  
import sys  
sys.path.insert(0, '/workspaces/urban-potato/matgeo/codes/  
    CoordGeo/') # path to my scripts  
  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
# Local imports  
from line.funcs import *  
from triangle.funcs import *  
from conics.funcs import circ_gen
```

Python Code for Plotting

```
# Read data
data = np.loadtxt("values.dat", skiprows=1)

xc = data[0] # Extract x-coordinate (e.g., -1)
yc = data[1] # Extract y-coordinate (e.g., 4.5)

# Given points
A = np.array([-6, 7]).reshape(-1, 1)
B = np.array([-1, -5]).reshape(-1, 1)
P = np.array([xc, yc]).reshape(-1, 1)

# Generating line AB
x_AB = line_gen(A, B)

# Plotting
plt.plot(x_AB[0, :], x_AB[1, :], label='$AB$')
```

Python Code for Plotting

```
# Labeling the coordinates
tri_coords = np.block([[A, B, P]])
plt.scatter(tri_coords[0, :], tri_coords[1, :])

vert_labels = ['A', 'B', 'P']

# Helper function: format number with decimal only if needed
def fmt(val):
    return f"{val:.1f}" if abs(val - round(val)) > 1e-6 else f"{int(val)}"

for i, txt in enumerate(vert_labels):
    x = tri_coords[0, i].item()
    y = tri_coords[1, i].item()
    plt.annotate(f'{txt}\n({fmt(x)}, {fmt(y)})',
                (x, y),
                textcoords="offset points",
                xytext=(20, -10),
                ha='center')
```

Python Code for Plotting

```
ax = plt.gca()
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)

plt.legend(loc='best')
plt.grid()

# Increase y-axis from -8 to 8 to show full range
plt.ylim(-7, 8)
plt.xlim(-7,7)

# Save and open
plt.show()
plt.savefig('../figs/fig1.png')
```