2.4.40

EE25BTECH11043 - Nishid Khandagre

September 17, 2025

Question

Find the angle between the lines $-\sqrt{3}x + y - 5 = 0$ and $-x + \sqrt{3}y + 6 = 0$.

Theoretical Solution

Given lines:

$$L_1 : -\sqrt{3}x + y - 5 = 0 \tag{1}$$

$$L_2 : -x + \sqrt{3}y + 6 = 0 \tag{2}$$

The matrix form of a line can be written as

$$\mathbf{n}^\top \mathbf{x} = c \tag{3}$$

Where $\mathbf{n}$ is the normal vector and $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ is the position vector.

Theoretical Solution

$$L_1 : \mathbf{n_1}^\top \mathbf{x} = c_1 \tag{4}$$

$$L_2 : \mathbf{n_2}^\top \mathbf{x} = c_2 \tag{5}$$

Where $\mathbf{n_1}$ and $\mathbf{n_2}$ are the normal vectors to the lines L_1 and L_2 respectively.

$$\mathbf{n_1} = \begin{pmatrix} -\sqrt{3} \\ 1 \end{pmatrix} \tag{6}$$

$$\mathbf{n_2} = \begin{pmatrix} -1 \\ \sqrt{3} \end{pmatrix} \tag{7}$$

Theoretical Solution

The angle θ between the lines is the angle between their normal vectors.

$$\cos\theta = \frac{\mathbf{n_1}^\top \mathbf{n_2}}{\left\|\mathbf{n_1}\right\| \left\|\mathbf{n_2}\right\|} \tag{8}$$

$$\begin{aligned} \mathbf{n_1}^\top \mathbf{n_2} &= \begin{pmatrix} -\sqrt{3} & 1 \end{pmatrix} \begin{pmatrix} -1 \\ \sqrt{3} \end{pmatrix} && (9) \\ &= (-\sqrt{3})(-1) + (1)(\sqrt{3}) && (10) \\ &= 2\sqrt{3} && (11) \end{aligned}$$

Theoretical Solution

$$
\begin{aligned}
\|\mathbf{n_1}\| &= \sqrt{\mathbf{n_1}^\top \mathbf{n_1}} && (12)\\
&= \sqrt{(-\sqrt{3})^2 + (1)^2} && (13)\\
&= \sqrt{4} && (14)\\
&= 2 && (15)
\end{aligned}
$$

$$
\begin{aligned}
\|\mathbf{n_2}\| &= \sqrt{\mathbf{n_2}^\top \mathbf{n_2}} && (16)\\
&= \sqrt{(-1)^2 + (\sqrt{3})^2} && (17)\\
&= \sqrt{4} && (18)\\
&= 2 && (19)
\end{aligned}
$$

Theoretical Solution

Now, substitute these values into the formula (8)

$$\cos\theta = \frac{2\sqrt{3}}{(2)(2)} \tag{20}$$

$$= \frac{\sqrt{3}}{2} \tag{21}$$

$$\theta = \frac{\pi}{6} \text{ radians} \tag{22}$$

C Code

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

// Function to calculate the angle between two lines in degrees
// given their slopes m1 and m2
void findAngleBetweenLines(double m1, double m2, double *
    angle_degrees) {
    // Calculate the tangent of the angle
    double tan_theta = (m1 - m2) / (1 + m1 * m2);

    // Calculate the angle in radians using atan
    double angle_radians = atan(tan_theta);

    // Convert the angle from radians to degrees
    *angle_degrees = angle_radians * (180.0 / M_PI);
```

C Code

```
    // Ensure the angle is positive
    if (*angle_degrees < 0) {
        *angle_degrees += 180.0;
    }
}

int main() {
    double m1, m2; // Slopes of the two lines
    double calculated_angle_degrees; // Variable to store the
        calculated angle
```

C Code

```c

    // Call the function to calculate the angle
    findAngleBetweenLines(m1, m2, &calculated_angle_degrees);

    return 0;
}
```

Python Code through shared output

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

# Load the shared library
lib_angle = ctypes.CDLL(./code3.so)

# Define the argument types and return type for the C function
lib_angle.findAngleBetweenLines.argtypes = [
    ctypes.c_double, # m1
    ctypes.c_double, # m2
    ctypes.POINTER(ctypes.c_double) # angle_degrees
]
lib_angle.findAngleBetweenLines.restype = None

# Line 1: y - sqrt(3)x - 5 = 0 => y = sqrt(3)x + 5
# Slope m1 = sqrt(3)
```

Python Code through shared output

```
m1_given = math.sqrt(3)

# Line 2: sqrt(3)y - x + 6 = 0 => sqrt(3)y = x - 6 => y = (1/sqrt
    (3))x - 6/sqrt(3)

# Slope m2 = 1 / math.sqrt(3)
m2_given = 1 / math.sqrt(3)

# Create a ctypes double to hold the result
angle_result = ctypes.c_double()

# Call the C function to find the angle
lib_angle.findAngleBetweenLines(
    m1_given, m2_given, ctypes.byref(angle_result)
)

angle_found = angle_result.value
```

Python Code through shared output

```
print(fThe angle between the lines is {angle_found:.2f} degrees)

# --- Plotting the lines ---
# Generate x values
x_vals = np.linspace(-10, 10, 400)

# Calculate y values for Line 1: y = sqrt(3)x + 5
y1_vals = m1_given * x_vals + 5

# Calculate y values for Line 2: y = (1/sqrt(3))x - 6/sqrt(3)
y2_vals = m2_given * x_vals - 6 / math.sqrt(3)

plt.figure(figsize=(10, 8))

# Plot Line 1
plt.plot(x_vals, y1_vals, label=f'Line 1: y - $\\sqrt{{3}}$x - 5 = 0 (m={m1_given:.2f})', color='blue')
```

```
# Plot Line 2
plt.plot(x_vals, y2_vals, label=f'Line 2: $\\sqrt{{3}}$y - x + 6
    = 0 (m={m2_given:.2f})', color='red')

# Find intersection point for plotting
# Intersection of y = m1*x + c1 and y = m2*x + c2
# m1*x + c1 = m2*x + c2 => x(m1 - m2) = c2 - c1
# x = (c2 - c1) / (m1 - m2)
c1 = 5
c2 = -6 / math.sqrt(3)
x_intersect = (c2 - c1) / (m1_given - m2_given)
y_intersect = m1_given * x_intersect + c1 # Using Line 1 equation
     to find y

plt.scatter(x_intersect, y_intersect, color='green', s=100,
    zorder=5, label='Intersection')
```

Python Code through shared output

```
plt.annotate(
    f'({x_intersect:.2f}, {y_intersect:.2f})',
    xy=(x_intersect, y_intersect),
    xytext=(x_intersect + 1, y_intersect + 1), # Offset text for
        better readability
    fontsize=10,
    color='black'
)


plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title(f'Lines and the Angle Between Them ({angle_found:.2f}
    degrees)')
```

Python Code through shared output

```
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(True)
plt.legend()
plt.ylim(-10, 10) # Adjust y-limits for better viewing
plt.xlim(-10, 10) # Adjust x-limits for better viewing
plt.gca().set_aspect('equal', adjustable='box')
plt.savefig(fig1.png) % Save the plot
plt.show()
```

Python Code: Direct

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

def find_angle_between_lines_and_plot():
    # Line 1: y - sqrt(3)x - 5 = 0
    # Slope m1 = sqrt(3)
    m1 = np.sqrt(3)

    # Line 2: sqrt(3)y - x + 6 = 0
    # Slope m2 = 1 / np.sqrt(3)
    m2 = 1 / np.sqrt(3)

    # Calculate the angle using the formula: tan(theta) = (m1 -
        m2) / (1 + m1 * m2)
    # We use arctan to get the angle in radians.
    # The result will be in the range [-pi/2, pi/2].
    angle_radians = np.arctan((m1 - m2) / (1 + m1 * m2))
```

Python Code : Direct

```
# Convert the angle from radians to degrees
angle_degrees = np.degrees(angle_radians)

# The formula gives an angle between -90 and 90 degrees.
# To get the acute angle between lines, we take the absolute
    value.
angle_degrees = abs(angle_degrees)

# If the angle is obtuse, we subtract it from 180 to get the
    acute angle.
if angle_degrees > 90:
    angle_degrees = 180 - angle_degrees

print(fThe angle between the lines is {angle_degrees:.2f}
    degrees)
```

Python Code : Direct

```
# --- Plotting the lines ---

plt.figure(figsize=(10, 8))

# Generate x values
x_vals = np.linspace(-10, 10, 400)

# Calculate y values for Line 1: y = sqrt(3)x + 5
y1_vals = m1 * x_vals + 5

# Calculate y values for Line 2: y = (1/sqrt(3))x - 6/np.sqrt
    (3)
y2_vals = m2 * x_vals - 6 / np.sqrt(3)

# Plot Line 1
plt.plot(x_vals, y1_vals, label=f'Line 1: y - $\\sqrt{{3}}$x
    - 5 = 0 (m={m1:.2f})', color='blue')
```

Python Code : Direct

```
# Plot Line 2
plt.plot(x_vals, y2_vals, label=f'Line 2: $\\sqrt{{3}}$y - x
    + 6 = 0 (m={m2:.2f})', color='red')

# Find the intersection point for labeling and drawing the
    angle
c1 = 5
c2 = -6 / np.sqrt(3)

# Check for parallel lines (m1 == m2) to avoid division by
    zero
if abs(m1 - m2) > 1e-9:
    x_intersect = (c2 - c1) / (m1 - m2)
    y_intersect = m1 * x_intersect + c1
    plt.scatter(x_intersect, y_intersect, color='green', s
        =100, zorder=5, label='Intersection Point')
```

```
        plt.annotate(f'({x_intersect:.2f}, {y_intersect:.2f})',
                     (x_intersect, y_intersect),
                     textcoords=offset points, xytext=(5,5), ha='
                         left')
    else:
        print(The lines are parallel.)
        x_intersect = None
        y_intersect = None

    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title(f'Lines and the Angle Between Them ({angle_degrees
        :.2f} degrees)')
    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)
```

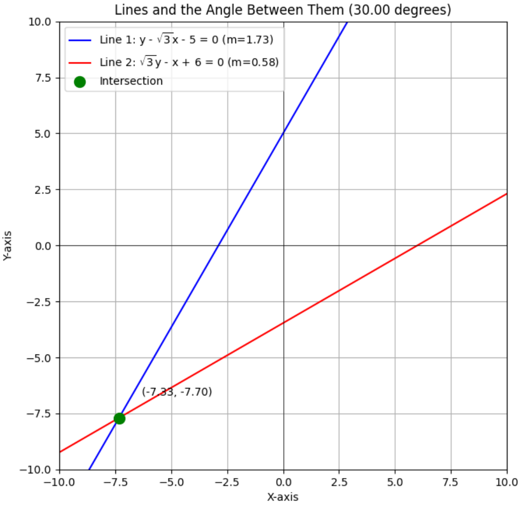
Python Code : Direct

```
    plt.grid(True)
    plt.legend()
    plt.ylim(-10, 10)
    plt.xlim(-10, 10)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.savefig(fig2.png) % Save the plot
    plt.show()

    print(Figure saved as fig2.png)

# Call the function to execute the code and generate the plot
find_angle_between_lines_and_plot()
```

Plot by Python using shared output from C



Plot by Python only