4.7.56

Vaishnavi - EE25BTECH11059

October 2, 2025

Question

Find the equation of the line whose perpendicular distance from the origin is 4 units and the angle which the normal makes with positive direction of x-axis is 15°

Solution

Variable	**Value**
d	4
m	$-cot15^\circ$

Table: Variables Used

Solution

Let eq of line be

$$\mathbf{n}^{\mathbf{T}}\mathbf{x} = c \tag{1}$$

where,

$$\mathbf{n} = \begin{pmatrix} cos15^\circ \\ sin15^\circ \end{pmatrix} \tag{2}$$

eq of line is

$$\begin{pmatrix} cos15^\circ & sin15^\circ \end{pmatrix} \mathbf{x} = c \tag{3}$$

solution

As distance from origin (d)=4 units

$$\frac{|c|}{\|n\|} = 4 \tag{4}$$

$$\frac{|c|}{1} = 4 \tag{5}$$

$$c = \pm 4 \tag{6}$$

Hence eq of line is

$$\begin{pmatrix} cos15^\circ & sin15^\circ \end{pmatrix} \mathbf{x} = \pm 4 \tag{7}$$

Refer to Figure

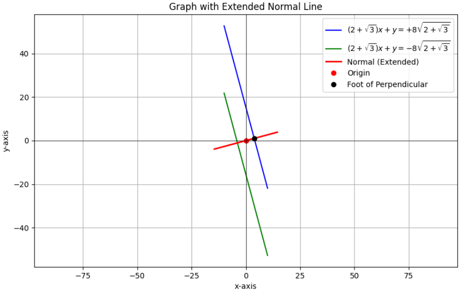

Figure:

Python Code

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

# Constants
sqrt3 = np.sqrt(3)
A = 2 + sqrt3
B = 1
C = 8 * np.sqrt(2 + sqrt3)

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

# Two lines: one for +C and one for -C
y1 = -A * x + C  # Line 1
y2 = -A * x - C  # Line 2

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x, y1, label=r'$(2+\sqrt{3})x +y = +8\sqrt
```

Python Code

```python
# === Calculate the foot of the perpendicular from origin to line1 ===
x0, y0 = 0, 0
A_val = A
B_val = 1
C_val = -C  # Rewrite as Ax + By + C = 0

denominator = A_val**2 + B_val**2
x_foot = (B_val * (B_val * x0 - A_val * y0) - A_val * C_val) / denominator
y_foot = (A_val * (A_val * y0 - B_val * x0) - B_val * C_val) / denominator

# === Draw extended solid normal line through origin and foot ===
# Direction vector of normal = (A, B)
```

Python Code

```
normal_length = 15  # increase to make longer
unit_normal = np.array([A_val, B_val]) / np.sqrt(
    denominator)
start_point = -normal_length * unit_normal
end_point = normal_length * unit_normal

plt.plot([start_point[0], end_point[0]], [start_point
    [1], end_point[1]],
         color='red', linewidth=2, label='Normal (
             Extended)')

# Plot origin and foot point
plt.plot(0, 0, 'ro', label='Origin')
plt.plot(x_foot, y_foot, 'ko', label='Foot of
    Perpendicular')

# Axes
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
```

Python Code

```
# Formatting
plt.title('Graph with Extended Normal Line')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid(True)
plt.axis('equal')
plt.legend()

# Save and show plot
plt.savefig( graph7_with_extended_normal.png )
plt.show()
```

C Code

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

// Function to compute normal vector, magnitude, and constants
void compute_line_params(double m, double d, double* A, double* B, double* C1, double* C2) {
    // Normal vector n = (-m, 1)
    *A = -m;
    *B = 1.0;

    // Norm of the vector
    double norm = sqrt((*A) * (*A) + (*B) * (*B));

    // |c| = d * ||n||
    double abs_c = d * norm;

    //  c  values
    *C1 = abs_c;
```

C Code

```c
int main() {
    double m = -2.0 - sqrt(3.0);  // Given slope
    double d = 4.0;               // Perpendicular distance from origin

    double A, B, C1, C2;

    // Compute parameters
    compute_line_params(m, d, &A, &B, &C1, &C2);

    // Display the results
    printf( Equation of the lines:\n );
    printf( %.4fx + %.4fy = %.4f\n , A, B, C1);
    printf( %.4fx + %.4fy = %.4f\n , A, B, C2);

    return 0;
}
```

Python and C Code

```python
import ctypes
from ctypes import c_double, POINTER

# Load the compiled shared object
lib = ctypes.CDLL( ./code.so )

# Define argument and return types for the function
lib.compute_line_params.argtypes = [c_double, c_double,
                                    POINTER(c_double), POINTER(
                                    c_double),
                                    POINTER(c_double), POINTER(
                                    c_double)]

# Inputs
m = -2 - 3**0.5  # given slope
d = 4.0          # distance from origin
```

Python and C Code

```
# Outputs
A = c_double()
B = c_double()
C1 = c_double()
C2 = c_double()

# Call the C function
lib.compute_line_params(m, d, ctypes.byref(A), ctypes.byref(B), ctypes.byref(C1), ctypes.byref(C2))

# Print the result
print( Equation of the line(s): )
print(f {A.value:.4f}x + {B.value:.4f}y = {C1.value:.4f} )
print(f {A.value:.4f}x + {B.value:.4f}y = {C2.value:.4f} )
```