4.3.37

RAVULA SHASHANK REDDY - EE25BTECH11047

September 28, 2025

Question

Find the equation of the line passing through the points

$$\mathbf{A} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 3 \\ 6 \end{pmatrix}.$$

Equation

Equation of a line is:

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

Theoretical Solution

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix}^T \mathbf{n} = \begin{pmatrix} c \\ c \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix}^T \mathbf{n} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix} \mathbf{n} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} 1 & 2 & 1 \\ 3 & 6 & 1 \end{pmatrix} \xrightarrow{R_2 - 3R_1} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & -2 \end{pmatrix} \tag{4}$$

$$0 = -2 \tag{5}$$

Inconsistent.Hence c=0.

$$\begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix} \mathbf{n} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{6}$$

Theoretical Solution

$$\begin{pmatrix} 1 & 2 & 0 \\ 3 & 6 & 0 \end{pmatrix} \xrightarrow{R_2 - 3R_1} \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{8}$$

$$\mathbf{n} = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \tag{9}$$

Equation of a Line is

$$\mathbf{n}^T \mathbf{x} = c \tag{10}$$

$$\begin{pmatrix} -2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = c \tag{11}$$

$$c = 0 \tag{12}$$

$$\begin{pmatrix} -2 & 1 \end{pmatrix} \mathbf{x} = 0 \tag{13}$$

$$-2x + y = 0 \tag{14}$$

$$\boxed{y = 2x} \tag{15}$$

C Code

```c
#include <stdio.h>

int main() {
    // Points A and B
    int Ax = 1, Ay = 2;
    int Bx = 3, By = 6;

    // Direction vector of AB
    int dx = Bx - Ax;
    int dy = By - Ay;

    // Normal vector (perpendicular to direction)
    int nx = -dy;
    int ny = dx;

    // Equation: n^T * x = c
    // Compute c using point A
    int c = nx * Ax + ny * Ay;
```

C Code

```
    printf(Normal vector n = (%d, %d)\n, nx, ny);
    printf(Equation of line: %d*x + %d*y = %d\n, nx, ny, c);

    // Convert to slope-intercept form if ny != 0
    if (ny != 0) {
        double slope = -(double)nx / ny;
        double intercept = (double)c / ny;
        printf(Line equation (y = mx + c): y = %.2fx + %.2f\n,
            slope, intercept);
    }

    return 0;
}
```

Python Direct

```
import numpy as np
import matplotlib.pyplot as plt
# local imports
from libs.line.funcs import *
from libs.triangle.funcs import *
from libs.conics.funcs import circ_gen

# Points
A = np.array([1, 2]).reshape(-1,1)
B = np.array([3, 6]).reshape(-1,1)

# Direction vector AB
d = B - A

# Normal vector n (perpendicular to AB)
n = np.array([-d[1,0], d[0,0]]).reshape(-1,1)
```

Python Direct

```
# Compute c using point A
c = (n.T @ A)[0,0]

# Line equation: y = (c - n1*x)/n2
x_vals = np.linspace(0, 5, 100)
y_vals = (c - n[0,0]*x_vals)/n[1,0]

# Plot
plt.figure(figsize=(6,6))

# Line
plt.plot(x_vals, y_vals, 'b-', label='Line AB')

# Points with coordinates labeled
plt.plot(A[0,0], A[1,0], 'ro')
plt.text(A[0,0]+0.1, A[1,0]+0.1, f'A({A[0,0]}, {A[1,0]})',
    fontsize=12, color='red')
```

Python Direct

```
plt.plot(B[0,0], B[1,0], 'go')
plt.text(B[0,0]+0.1, B[1,0]+0.1, f'B({B[0,0]}, {B[1,0]})',
    fontsize=12, color='green')

# Axes and grid
plt.xlabel('x')
plt.ylabel('y')
plt.title('Line through points A and B')
plt.grid(True)
plt.axis('equal')
plt.legend()
plt.show()
```

Python Shared Output

```
    import ctypes
import numpy as np
import matplotlib.pyplot as plt
# local imports
from libs.line.funcs import *
from libs.triangle.funcs import *
from libs.conics.funcs import circ_gen

# Load C shared library
lib = ctypes.CDLL('./libline.so')

# Define argument and return types
lib.line_normal.argtypes = [ctypes.c_double, ctypes.c_double,
                            ctypes.c_double, ctypes.c_double,
                            ctypes.POINTER(ctypes.c_double),
                            ctypes.POINTER(ctypes.c_double),
                            ctypes.POINTER(ctypes.c_double)]
```

Python Shared Output

```
# Points
Ax, Ay = 1.0, 2.0
Bx, By = 3.0, 6.0

# Prepare output variables
n1 = ctypes.c_double()
n2 = ctypes.c_double()
c = ctypes.c_double()

# Call the C function
lib.line_normal(Ax, Ay, Bx, By, ctypes.byref(n1), ctypes.byref(n2
    ), ctypes.byref(c))
```

Python Shared Output

```
print(Normal vector n =, n1.value, n2.value)
print(Scalar c =, c.value)

# Line: y = (c - n1*x)/n2
x_vals = np.linspace(0, 5, 100)
y_vals = (c.value - n1.value * x_vals)/n2.value

# Plot
plt.figure(figsize=(6,6))
plt.plot(x_vals, y_vals, 'b-', label='Line AB')
```

Python Shared Output

```
# Points labeled with coordinates
plt.plot(Ax, Ay, 'ro')
plt.text(Ax+0.1, Ay+0.1, f'A({Ax}, {Ay})', fontsize=12, color='
    red')

plt.plot(Bx, By, 'go')
plt.text(Bx+0.1, By+0.1, f'B({Bx}, {By})', fontsize=12, color='
    green')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Line through points A and B (computed via C + ctypes)'
    )
plt.grid(True)
plt.axis('equal')
plt.legend()
plt.show()
```

Plot