Question

Problem

If the line

$$\frac{x}{a} + \frac{y}{b} = 1$$

passes through the points $(2, -3)$ and $(4, -5)$, find (a, b).

Solution: Step 1

The given points are

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 4 \\ -5 \end{bmatrix}.$$

The direction vector of the line is

$$\mathbf{m} = \mathbf{x}_2 - \mathbf{x}_1 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}.$$

Solution: Step 2

The normal vector $\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$ must satisfy

$$\mathbf{n}^T \mathbf{m} = 0.$$

$$\begin{bmatrix} n_1 & n_2 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \end{bmatrix} = 0 \quad \Rightarrow \quad n_1 = n_2.$$

So we can take

$$\mathbf{n} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Solution: Step 3

The line equation is

$$\mathbf{n}^T\mathbf{x} = c \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = c.$$

Substitute point $\mathbf{x}_1 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$:

$$c = 2 - 3 = -1.$$

Final Equation:

$$x + y = -1.$$

C Code (Part 1)

```c
#include <stdio.h>

int main() {
    double x1 = 2, y1 = -3;
    double x2 = 4, y2 = -5;

    // Direction vector
    double m1 = x2 - x1;
    double m2 = y2 - y1;

    // Normal vector (perpendicular)
    double n1 = m2;
    double n2 = -m1;
```

```c
    // Constant c
    double c = n1*x1 + n2*y1;

    // Line equation
    printf("Equation of line: %.2lf*x + %.2lf*y = %.2lf\n",
           n1, n2, c);

    return 0;
}
```

Python Code (Part 1)

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

# Load the shared library
lib = ctypes.CDLL("./c.so")

# Define the function signature for points
lib.points.argtypes = [
    ctypes.c_float, # x_0
    ctypes.c_float, # y_0
    ctypes.c_float, # x_end
    ctypes.c_float, # h
    np.ctypeslib.ndpointer(dtype=np.float32, ndim=1),
    np.ctypeslib.ndpointer(dtype=np.float32, ndim=1),
    ctypes.c_int # steps
]
```

Python Code (Part 2)

```python
# Parameters for simulation
x_0, y_0 = 0.0, 2.0
x_end, step_size = 1.0, 0.001
steps = int((x_end - x_0) / step_size) + 1

x_points = np.zeros(steps, dtype=np.float32)
y_points = np.zeros(steps, dtype=np.float32)

# Call the points function
lib.points(x_0, y_0, x_end, step_size,
           x_points, y_points, steps)

# Theoretical solution (C = -2)
def theoretical_solution(x):
    return (-x + 4 - 2*np.exp(x))
```

```python
# Generate theory curve
x_theory = np.linspace(x_0, x_end, 1000)
y_theory = theoretical_solution(x_theory)

# Plot results
plt.plot(x_points, y_points, 'ro-',
        markersize=2, linewidth=4, label="sim")
plt.plot(x_theory, y_theory, 'b-',
        linewidth=2, label="theory")

plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True, linestyle="--")
plt.show()
```

Plot of the Line