Matgeo: 4-4.2-21

J. Adishesh Balaji
AI24BTECH11016

November 6, 2024

1. Problem

2. Solution
   - Direction Vector
   - Normal Vector

Problem Statement

Find the direction and normal vectors of the line.

$$F = \frac{9}{5}C + 32 \tag{3.1}$$

Direction Vector

The equation of line is given by

$$\begin{pmatrix} C \\ F \end{pmatrix} = \begin{pmatrix} C \\ \frac{9}{5}C + 32 \end{pmatrix}$$
$$\begin{pmatrix} C \\ F \end{pmatrix} = \begin{pmatrix} 0 \\ 32 \end{pmatrix} + C \begin{pmatrix} 1 \\ \frac{9}{5} \end{pmatrix}$$

which can be compared with

$$\mathbf{x} = \mathbf{h} + k\mathbf{m} \tag{3.2}$$

Where **h** is any point on the line and

$$\mathbf{m} = \begin{pmatrix} 1 \\ \frac{9}{5} \end{pmatrix} \tag{3.3}$$

is the direction vector

Normal Vector

The normal vector can be found as follows

$$\mathbf{m}^T\mathbf{n} = 0 \tag{3.4}$$

$$\mathbf{n}^T\mathbf{x} = \mathbf{n}^T\mathbf{h} + k\mathbf{n}^T\mathbf{m} \tag{3.5}$$

$$\mathbf{n} = \begin{pmatrix} -m \\ 1 \end{pmatrix} \tag{3.6}$$

Hence, the normal vector

$$\mathbf{n} = \begin{pmatrix} -\frac{9}{5} \\ 1 \end{pmatrix} \tag{3.7}$$

The code in /bmrasgn/asgn1/codes/line.py verifies (3.3) and (3.7)

C-code to generate Data I

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

// Function to generate points on the line F = (9/5)C + 32
void point_gen(FILE *fptr, double c1, double f1, double c2, double f2, int num_points) {
    for (int i = 0; i <= num_points; i++) {
        double t = (double)i / num_points;
        double c = c1 + t * (c2 - c1); // Linear interpolation for C
        double f = f1 + t * (f2 - f1); // Linear interpolation for F
        fprintf(fptr, "%lf,%lf\n", c, f);
    }
}

int main() {
    // Define two points on the line F = (9/5)C + 32
    double c1 = -50.0, f1 = (9.0 / 5.0) * c1 + 32.0;  // First point (C=-50, F)
    double c2 = 100.0, f2 = (9.0 / 5.0) * c2 + 32.0;  // Second point (C=100, F)
```

C-code to generate Data II

```c

    // Open the file to save the points
    FILE *fptr = fopen("line_points.txt", "w");
    if (fptr == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Generate points on the line
    point_gen(fptr, c1, f1, c2, f2, 63);  // Generate 63 points on the line

    // Normal vector generation
    // The slope of the line is 9/5, so the slope of the normal is -5/9.
    double c_normal = c1, f_normal = f1;  // Take the first point for normal vector
    double norm_slope = -5.0 / 9.0;       // Slope of the normal line
    double norm_length = 50.0;            // Arbitrary length for the normal line

```

C-code to generate Data III

```c
    double c_norm_end = c_normal + norm_length;
    double f_norm_end = f_normal + norm_slope * (c_norm_end - c_normal);

    // Generate points on the normal line
    point_gen(fptr, c_normal, f_normal, c_norm_end, f_norm_end, 20);

    // Close the file
    fclose(fptr);

    printf("Points on the line and normal vector saved to line_points.txt\n");

    return 0;
}


```

Python code to plot graph I

```
        import sys  # for path to external scripts
sys.path.insert(0, '/home/adishesh-balaji/github/matgeo/codes/CoordGeo')  # path to my scripts
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt

# local imports
from line.funcs import *
from triangle.funcs import *
from conics.funcs import circ_gen

# Function to read points from the txt file
def read_points_from_file(filename):
    c_values = []
    f_values = []

    with open(filename, 'r') as file:
        for line in file:
            # Split the line by comma and convert to float
```

Python code to plot graph II

```python
            c, f = map(float, line.strip().split(','))
            c_values.append(c)
            f_values.append(f)

    return np.array(c_values), np.array(f_values)

# Function to plot the line and its normal vector
def plot_line_and_normal(c_values, f_values):
    # Plot the line as a dotted line
    plt.plot(c_values, f_values, label=r'$F = \frac{9}{5}C + 32$', color='blue', linestyle='dotted')

    # Calculate the midpoint of the line
    midpoint_index = len(c_values) // 2
    c_mid = c_values[midpoint_index]
    f_mid = f_values[midpoint_index]

    # Midpoint as the starting point for the normal vector
    A_mid = np.array([c_mid, f_mid])

```

Python code to plot graph III

```
    # Direction vector (slope = 9/5)
    m = np.array([1, 9/5])

    # Normal vector (slope = -5/9)
    n = np.array([-9/5, 1])

    # Plot the main line using parametric form (starting at A = [0, 32])
    main_line_points = line_dir_pt(m, np.array([0, 32]), -50, 50)
    plt.plot(main_line_points[0, :], main_line_points[1, :], color='blue',
    ↪  linestyle='dotted')

    # Plot the normal vector emerging from the midpoint
    normal_points = line_dir_pt(n, A_mid, -20, 20)  # Adjust the range of
    ↪  the normal vector as needed
    plt.plot(normal_points[0, :], normal_points[1, :], label='Normal
    ↪  Vector', color='red', linestyle = 'dotted')

    # Set labels
    plt.xlabel('C (Celsius)')
    plt.ylabel('F (Fahrenheit)')
```

Python code to plot graph IV

```python
    plt.legend()

    # Set limits
    plt.xlim(-60, 110)
    plt.ylim(-70, 220)

    # Add grid
    plt.grid(True)
    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)

    # Show the plot
    plt.gca().set_aspect('equal', adjustable='box')
    plt.show()

# Main code execution
if __name__ == "__main__":
    filename = "line_points.txt"
    c_values, f_values = read_points_from_file(filename)
    plot_line_and_normal(c_values, f_values)
```

Python code to plot graph V

```

```