Presentation By

EE24BTECH11021 - ESHAN RAY

November 5, 2024

1. Problem
2. Input Parameters
3. Solution
4. Plot
5. C Code
6. Python Code

Problem Statement

If the distances of $\mathbf{P} = (x, y)$ from $\mathbf{A} = (5, 1)$ and $\mathbf{B} = (-1, 5)$ are equal, then prove that $3x = 2y$.

Input Parameters

Variable	Description
$\mathbf{A}(5,1)$	coordinates of first point
$\mathbf{B}(-1,5)$	coordinates of second point
$\mathbf{P}(x,y)$	Equidistant point of **A** and **B**

Solution

$$\|\mathbf{B}-\mathbf{P}\|^2 = \|\mathbf{A}-\mathbf{P}\|^2 \tag{4.1}$$

$$\implies (\mathbf{B}-\mathbf{P})^\top (\mathbf{B}-\mathbf{P}) = (\mathbf{A}-\mathbf{P})^\top (\mathbf{A}-\mathbf{P}) \tag{4.2}$$

$$\implies \mathbf{B}^2 + \mathbf{P}^2 - 2\mathbf{P}\mathbf{B}^\top = \mathbf{A}^2 + \mathbf{P}^2 - 2\mathbf{P}\mathbf{A}^\top \tag{4.3}$$

$$\implies \mathbf{P}\left(\mathbf{A}^\top - \mathbf{B}^\top\right) = \frac{\mathbf{A}^2 - \mathbf{B}^2}{2} \tag{4.4}$$

Solution

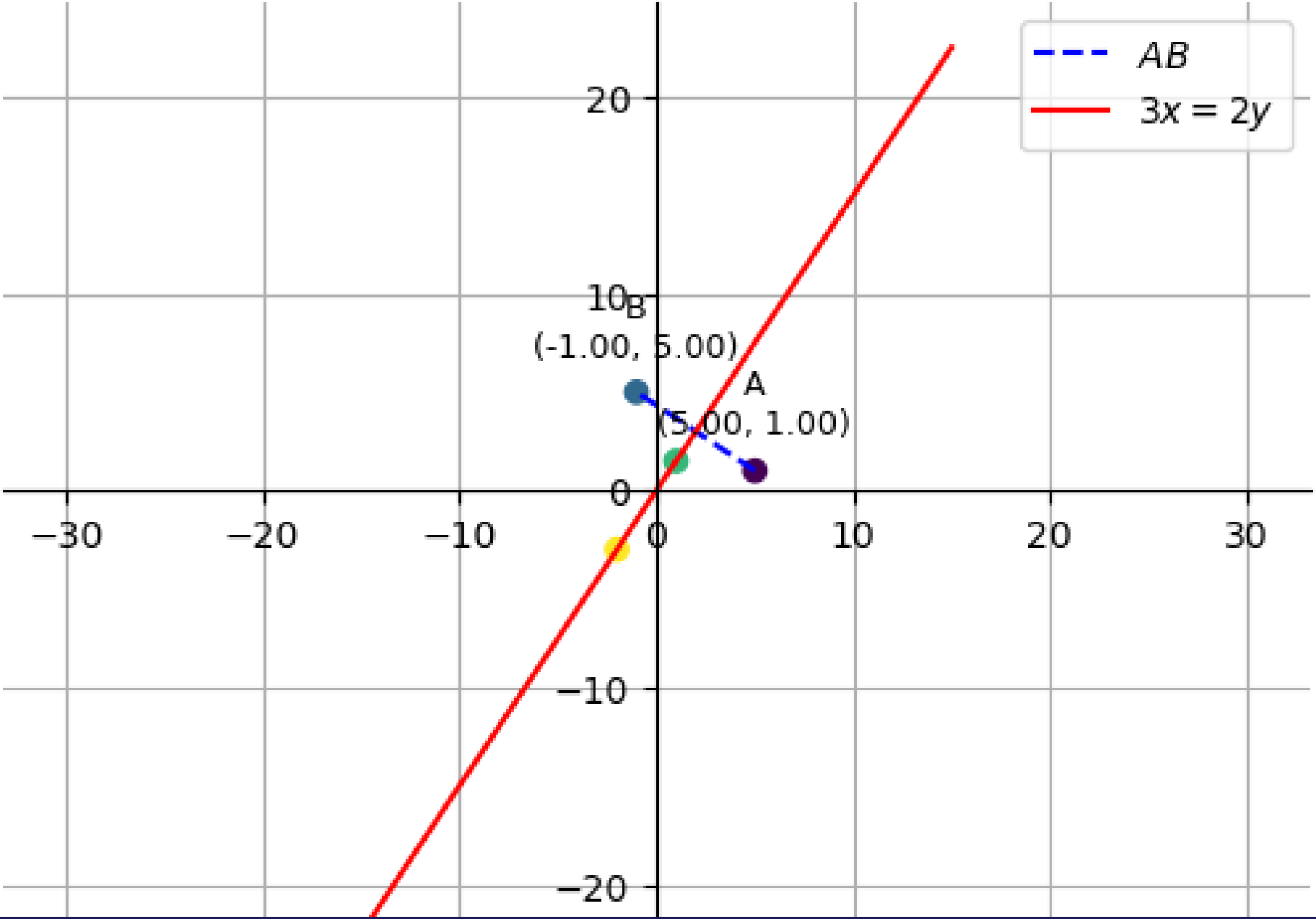
$$\implies \mathbf{P}\left(\begin{pmatrix}5 & 1\end{pmatrix} - \begin{pmatrix}-1 & 5\end{pmatrix}\right) = \frac{26-26}{2} \tag{4.5}$$

$$\implies \begin{pmatrix}x\\y\end{pmatrix}\begin{pmatrix}6 & -4\end{pmatrix} = 0 \tag{4.6}$$

$$\implies 6x - 4y = 0 \tag{4.7}$$

$$\implies 3x = 2y \tag{4.8}$$

Plot



C Code

```
#include <stdio.h>

void print_points_to_file(const char *filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        perror("Error opening file");
        return;
    }

    // Define points A and B
    fprintf(file, "5 1\n"); // Point A
    fprintf(file, "-1 -5\n"); // Point B

    // Define two points on the line 3x = 2y
    double x1 = 2, y1 = (3 * x1) / 2;
    double x2 = -2, y2 = (3 * x2) / 2;
```

C Code

```
    fprintf(file, "%.2lf-%.2lf\n", x1, y1); // Point on the line
    fprintf(file, "%.2lf-%.2lf\n", x2, y2); // Another point on the line

    fclose(file);
}

int main() {
    print_points_to_file("output.txt");
    return 0;
}
```

Python Code for Plotting

```
import sys
sys.path.insert(0, '/home/eshan/matgeo/codes/CoordGeo')
import numpy as np
import matplotlib.pyplot as plt

#local imports
from line.funcs import *

# Function to read points from a file
def read_points_from_file(filename):
    # Load the data from the file directly into a NumPy array
    points = np.loadtxt(filename)
    return points
# Read points from file
points = read_points_from_file('output.txt')
```

Python Code for Plotting

```
# Flatten the array if necessary
if points.ndim == 3:
    points = points.reshape(-1, 2)

# Extracting points A, B, C, and D
A, B, C, D = points[0], points[1], points[2], points[3]

# Define a range of values for plotting infinitely
x_range = np.linspace(-15, 15, 100) # Adjust as necessary to ensure
    lines extend sufficiently
# Generating all lines
x_AB = line_gen(A, B)

plt.plot(x_range,x_AB[0, :], x_AB[1, :], 'b--', label='$AB$')
# Line CD
slope_CD = (D[1] - C[1]) / (D[0] - C[0])
intercept_CD = C[1] - slope_CD * C[0]
```

Python Code for Plotting

```
plt.plot(x_range, slope_CD * x_range + intercept_CD, label='$3x=2y$',
     color='red')

# Plotting points
colors = np.arange(1, 5) # 4 points
plt.scatter(points[:, 0], points[:, 1], c=colors, label=None)

# Annotate the vertices
def annotate_point(point, label):
     plt.annotate(f'{label}\n({point[0]:.2f},-{point[1]:.2f})',
                  point,
                  textcoords="offset points",
                  xytext=(0, 10), # Position above the point
                  ha='center',
                  fontsize=9)
annotate_point(A, 'A')
annotate_point(B, 'B')
```

Python Code for Plotting

```
# Customize the plot
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')
plt.grid() # minor
plt.axis('equal')
plt.legend(loc='best')
plt.savefig('../plots/plot.png', format='png', bbox_inches='tight')
```