MATGEO PRESENTATION

DWARAK A
EE24BTECH11019

November 6, 2024

1. Problem Statement

2. Solution
   - Variables Used
   - Equidistant Point Condition
   - Perpendicular Bisector Equation
   - Final Equation

3. C Code

4. Python Code

5. Plot

Problem Statement

Find a relation between x and y such that the point (x, y) is equidistant from the points $(3, 6)$ and $(-3, 4)$.

Variables Used

Variable	Description	Value
A	First point	$\begin{pmatrix} 3 \\ 6 \end{pmatrix}$
B	Second point	$\begin{pmatrix} -3 \\ 4 \end{pmatrix}$
C	Mid-point of **A** and **B**	$\left(\frac{\mathbf{A}+\mathbf{B}}{2}\right)$
X	Set of points equidistant from **A** and **B**	$\begin{pmatrix} x \\ y \end{pmatrix}$

Equidistant Point Condition

If $\mathbf{X}$ is equidistant from points $\mathbf{A}$ and $\mathbf{B}$

$$\|\mathbf{A}-\mathbf{X}\| = \|\mathbf{B}-\mathbf{X}\| \tag{3.1}$$

$$\|\mathbf{A}-\mathbf{X}\|^2 = \|\mathbf{B}-\mathbf{X}\|^2 \tag{3.2}$$

Expanding the squared norms:

$$\|\mathbf{A}\|^2 - 2\mathbf{A}^\top\mathbf{X} + \|\mathbf{X}\|^2 = \|\mathbf{B}\|^2 - 2\mathbf{B}^\top\mathbf{X} + \|\mathbf{X}\|^2 \tag{3.3}$$

(3.4)

Perpendicular Bisector Equation

The equation simplifies to

$$(\mathbf{A}-\mathbf{B})^{\top}\mathbf{X} = \frac{\|\mathbf{A}\|^2 - \|\mathbf{B}\|^2}{2} \tag{3.5}$$

Substituting the **A** and **B** values, (3.5) can be derived as follows

$$\begin{pmatrix}6\\2\end{pmatrix}^{\top}\mathbf{X} = \frac{\left\|\begin{pmatrix}3\\6\end{pmatrix}\right\|^2 - \left\|\begin{pmatrix}-3\\4\end{pmatrix}\right\|^2}{2} \tag{3.6}$$

$$\begin{pmatrix}6\\2\end{pmatrix}^{\top}\mathbf{X} = 10 \tag{3.7}$$

$$\implies \begin{pmatrix}3\\1\end{pmatrix}^{\top}\mathbf{X} = 5 \tag{3.8}$$

Final Line Equation

Thus, from (3.8) the line equation representing points equidistant from $\mathbf{A}(3, 6)$ and $\mathbf{B}(-3, 4)$ is:

$$3x + y = 5 \tag{3.9}$$

The code below verifies (3.8)

C Code I

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

#include "libs/matfun.h"
#include "libs/geofun.h"

void point_gen(FILE *fptr, double **A, double **direction_vector, int
    no_rows, int no_col, int num_points) {
    double **output;
    for (double i = -num_points/2; i <= num_points/2; i++) {
        output = Matadd(A, Matscale(direction_vector,no_rows,no_col,(
    double) i/num_points), no_rows, no_col);
        fprintf(fptr, "%lf %lf\n", output[0][0], output[1][0]);
        freeMat(output, no_rows);
```

C Code II

```
    }
}

int main(){
    double x1 = 3, y1 = 6, x2 = -3, y2 = 4;
    int m = 2, n = 1;
    double **A, **B, **midpoint, **s_ab, **bisectorABMidpoint;

    A = createMat(m, n);
    B = createMat(m, n);
    midpoint = createMat(m, n);

    A[0][0] = x1;
    A[1][0] = y1;
    B[0][0] = x2;
    B[1][0] = y2;

    // Calculate the midpoint of AB
    midpoint = Matadd(A, B, m, n);
    midpoint = Matscale(midpoint, m, n, 1.0/2);
```

C Code III

```c

    // Calculate the vector AB and then the perpendicular bisector vector
    s_ab = Matsub(B, A, m, n);
    bisectorABMidpoint = normVec(s_ab);

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

    fprintf(fptr, "%lf %lf\n", x1, y1);
    fprintf(fptr, "%lf %lf\n", x2, y2);
    fprintf(fptr, "%lf %lf\n", midpoint[0][0], midpoint[1][0]);

    // Generate points on the perpendicular bisector
    point_gen(fptr, midpoint, bisectorABMidpoint, m, n, 20);
```

C Code IV

```
    // Close the file
    fclose(fptr);

    // Free all allocated memory
    freeMat(A,m);
    freeMat(B,m);
    freeMat(midpoint,m);
    freeMat(s_ab,m);
    freeMat(bisectorABMidpoint,m);
    return 0;
}
```

Python Code I

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

# Load the points from the .dat file
points = np.loadtxt('points.dat', max_rows = 3)
data = np.loadtxt('points.dat', skiprows = 3)

# Extract the x and y coordinates
x = data[:, 0]
y = data[:, 1]

# Define the points A, B, and (0,9)
[A, B, C] = np.array(points)
[txtA, txtB, txtC] =
    ['A'+str(tuple(A)),'B'+str(tuple(B)),'C'+str(tuple(C))]

# Plot the locus of X
plt.figure()
```

Python Code II

```
plt.plot(x, y, label = 'Locus of X')
plt.plot(points[:, 0], points[:, 1], label = 'AB')
plt.scatter(A[0], A[1], c = 'c', label = txtA)
plt.scatter(B[0], B[1], c = 'm', label = txtB)
plt.scatter(C[0], C[1], c = 'y', label = txtC)

# Annotate the points
plt.annotate(txtA, xy = A)
plt.annotate(txtB, xy = B)
plt.annotate(txtC, xy = C)

# Plot specs
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Plot of Locus of X')
plt.axis('equal')
plt.grid(True)
plt.legend(loc = 'upper right')
```

Python Code III

```

# Save and Display plot
plt.savefig("../figs/plot.png")
plt.show()
```

Plot

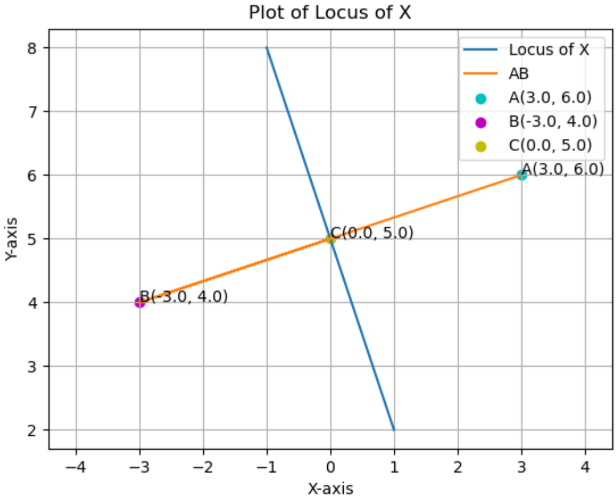

Figure: Locus of point X, equidistant from **A** and **B**