# MATGEO PRESENTATION

DWARAK A
EE24BTECH11019

November 6, 2024

## 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 **X** is equidistant from points **A** and **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:

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

## Perpendicular Bisector Equation

The equation (3.3) simplifies to

$$(\mathbf{A} - \mathbf{B})^\top \mathbf{X} = \frac{\|\mathbf{A}\|^2 - \|\mathbf{B}\|^2}{2} \tag{3.4}$$

Substituting the **A** and **B** values, (3.4) can be derived as follows

$$\begin{pmatrix} 3 - (-3) \\ 6 - 4 \end{pmatrix}^\top \mathbf{X} = \frac{\left\|\begin{pmatrix} 3 \\ 6 \end{pmatrix}\right\|^2 - \left\|\begin{pmatrix} -3 \\ 4 \end{pmatrix}\right\|^2}{2} \tag{3.5}$$

$$\begin{pmatrix} 6 \\ 2 \end{pmatrix}^\top \mathbf{X} = 10 \tag{3.6}$$

$$\implies \begin{pmatrix} 3 \\ 1 \end{pmatrix}^\top \mathbf{X} = 5 \tag{3.7}$$

## Final Line Equation

Thus, from (3.7) the line equation representing points equidistant from $\mathbf{A}(3,6)$ and $\mathbf{B}(-3,4)$ is:

$$3x + y = 5 \tag{3.8}$$

The code below verifies (3.7)

## 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 line_gen(FILE *fptr, double **A, double **dir_vector, int no_rows,
    int no_cols, double num_units_1, double num_units_2, int num_points)
     {
    double **output;
    double **unit_dir_vector = Matscale(dir_vector, no_rows, no_cols, 1/
    Matnorm(dir_vector, no_rows));
    for (double i = -num_units_1; i <= num_units_2; i += (num_units_1 +
    num_units_2)/num_points) {
```

## C Code II

```c
        output = Matadd(A, Matscale(unit_dir_vector,no_rows,no_cols, i),
      no_rows, no_cols);
        fprintf(fptr, "%lf %lf\n", output[0][0], output[1][0]);
        freeMat(output, no_rows);
    }
}

int main(){
    double x1 = 3.0, y1 = 6.0, x2 = -3.0, y2 = 4.0;
    int m = 2, n = 1;
    int k1 = 4, k2 = 4, res = 20;
    double **A, **B, **mid_point, **m_ab, **n_ab;

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

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

## C Code III

```
    B[1][0] = y2;

    // Calculate the midpoint of AB
    mid_point = Matscale(Matadd(A, B, m, n), m, n, 0.5);

    // Calculate the vector AB and then the perpendicular bisector
     vector
    m_ab = Matsub(B, A, m, n);
    n_ab = normVec(m_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);
```

## C Code IV

```c
    fprintf(fptr, "%lf %lf\n", mid_point[0][0], mid_point[1][0]);

    // Generate points on the perpendicular bisector
    line_gen(fptr, mid_point, n_ab, m, n, k1, k2, res);

    // Close the file
    fclose(fptr);

    // Free all allocated memory
    freeMat(A,m);
    freeMat(B,m);
    freeMat(mid_point,m);
    freeMat(m_ab,m);
    freeMat(n_ab,m);
    return 0;
}
```

## Python Code I

```python
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

```python
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()
```

## Python Code III

```python
plt.tight_layout()

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

## Plot

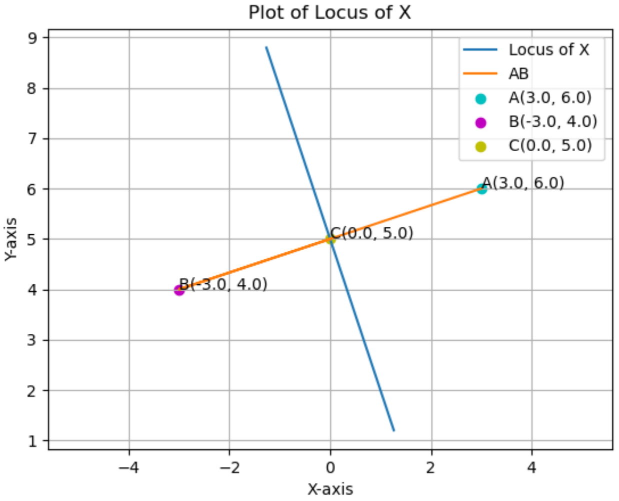

Figure: Locus of point X, equidistant from **A** and **B**