1.8.26

Anshu kumar ram-EE25BTECH11009

September 05, 2025

Question

Find a point on the y-axis which is equidistant from the points $A(6, 5)$ and $B(-4, 3)$.

Input Parameters

The input parameters for this problem are available in the table below.

Symbol	Value	Description
A	$\begin{pmatrix} 6 \\ 5 \end{pmatrix}$	First point
B	$\begin{pmatrix} -4 \\ 3 \end{pmatrix}$	Second point
O	$\begin{pmatrix} 0 \\ y \end{pmatrix}$	Desired point

Table: Parameters for the problem

Table:

Solution Step 1

If $\mathbf{O}$ lies on the y-axis and is equidistant from the points $\mathbf{A}$ and $\mathbf{B}$,

$$\|\mathbf{O} - \mathbf{A}\| = \|\mathbf{O} - \mathbf{B}\| \tag{1}$$

$$\implies \|\mathbf{O} - \mathbf{A}\|^2 = \|\mathbf{O} - \mathbf{B}\|^2 \tag{2}$$

Expanding both sides,

$$\|\mathbf{O}\|^2 - 2\mathbf{O}^\top\mathbf{A} + \|\mathbf{A}\|^2 = \|\mathbf{O}\|^2 - 2\mathbf{O}^\top\mathbf{B} + \|\mathbf{B}\|^2 \tag{3}$$

Solution Step 2

Simplifying,

$$(\mathbf{A} - \mathbf{B})^\top \mathbf{O} = \frac{\|\mathbf{A}\|^2 - \|\mathbf{B}\|^2}{2} \tag{4}$$

Since $\mathbf{O}$ lies on the y-axis,

$$\mathbf{O} = y\mathbf{e}_2 \tag{5}$$

Thus,

$$y = \frac{\|\mathbf{A}\|^2 - \|\mathbf{B}\|^2}{2(\mathbf{A} - \mathbf{B})^\top \mathbf{e}_2} \tag{6}$$

Solution Step 3

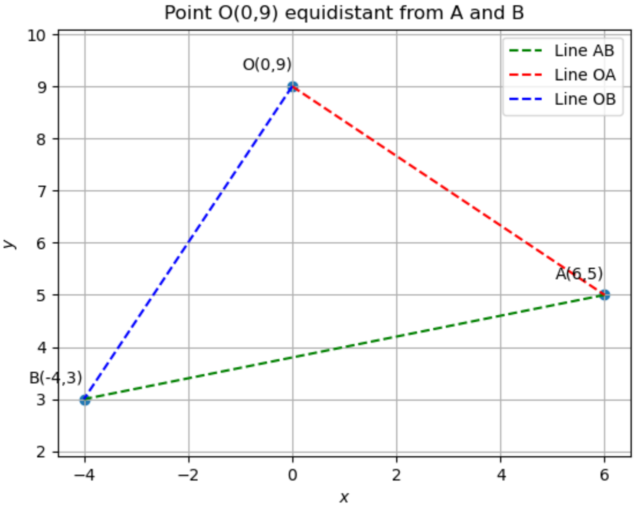
Substituting from the table we get,

$$y = 9 \tag{7}$$

So the required point is,

$$\boxed{\mathbf{O} = \begin{pmatrix} 0 \\ 9 \end{pmatrix}}$$

Plot


Point O(0,9) equidistant from A and B
Line AB
Line OA
Line OB
O(0,9)
A(6,5)
B(-4,3)
10
9
8
7
6
5
4
3
2
y
−4
−2
0
2
4
6
x

Pure Python (Part 1)

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

# path to your external scripts
sys.path.insert(0, '/home/anshu-ram/matgeo/codes/CoordGeo')

# local imports
from line.funcs import line_gen_num

# ========== Given vectors ==========
A = np.array([6, 5]).reshape(-1,1)
B = np.array([-4, 3]).reshape(-1,1)
O = np.array([0, 9]).reshape(-1,1)
```

Pure Python (Part 2)

```python
# ========== Lines ==========
x_AB = line_gen_num(A, B, 20)
x_OA = line_gen_num(O, A, 20)
x_OB = line_gen_num(O, B, 20)

# ========== Plotting ==========
plt.plot(x_AB[0,:], x_AB[1,:], "g--", label="Line AB")
plt.plot(x_OA[0,:], x_OA[1,:], "r--", label="Line OA")
plt.plot(x_OB[0,:], x_OB[1,:], "b--", label="Line OB")
```

Pure Python (Part 3)

```
# Points
tri_coords = np.hstack((A,B,O)) # stack column vectors
plt.scatter(tri_coords[0,:], tri_coords[1,:])

# Labels
vert_labels = [
    f'A({int(A[0,0])},{int(A[1,0])})',
    f'B({int(B[0,0])},{int(B[1,0])})',
    f'O({int(O[0,0])},{int(O[1,0])})'
]

for i, txt in enumerate(vert_labels):
    plt.annotate(txt, (tri_coords[0,i], tri_coords[1,i]),
                 textcoords="offset points", xytext=(0,10), ha='
                     right')
```

Pure Python (Part 4)

```python
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.title("Point O(0,9) equidistant from A and B")
plt.axis('equal')

# Save & show
plt.savefig("../figs/equidistant_point.png")
plt.show()
```

C Code (Part 1)

```c
#include <stdio.h>

/* Function to compute O on y-axis equidistant from A and B */
void equidistant_yaxis(const double* A, const double* B, double*
    O) {
    // A = (x1,y1), B = (x2,y2), O = (0,y)
    double x1 = A[0], y1 = A[1];
    double x2 = B[0], y2 = B[1];

    double num = (x1*x1 + y1*y1) - (x2*x2 + y2*y2);
    double den = 2*(y1 - y2);

    O[0] = 0.0;
    O[1] = num/den;
}
```

C Code (Part 2)

```c
/* Generate n points on line AB */
void line_gen(double* X, double* Y, const double* A, const double
    * B, int n, int m) {
    double temp[2];

    for (int i = 0; i < 2; i++) {
        temp[i] = (B[i] - A[i]) / (double)(n-1);
    }

    for (int i = 0; i < n; i++) {
        X[i] = A[0] + temp[0] * i;
        Y[i] = A[1] + temp[1] * i;
    }
}
```

Python + C Integration (Part 1)

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

# Load shared library
handc = ctypes.CDLL("./func.so")

# Define equidistant_yaxis prototype
handc.equidistant_yaxis.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
]
handc.equidistant_yaxis.restype = None
```

Python + C Integration (Part 2)

```
# Define line_gen prototype
handc.line_gen.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.c_int,
    ctypes.c_int
]
handc.line_gen.restype = None

# Dimension
m = 2

# Points A and B
A = np.array([6.0, 5.0], dtype=np.float64)
B = np.array([-4.0, 3.0], dtype=np.float64)

# Placeholder for O
```

Python + C Integration (Part 3)

```
# Call C function to compute O
handc.equidistant_yaxis(
    A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    O.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
)

print("A =", A)
print("B =", B)
print("O =", O)

# Generate lines
n = 20
X_l = np.zeros(n, dtype=np.float64)
Y_l = np.zeros(n, dtype=np.float64)
```

Python + C Integration (Part 4)

```
# AB
handc.line_gen(X_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double)
    ),
              Y_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
                  ,
              A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
              B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
              n, m)
x_AB, y_AB = X_l.copy(), Y_l.copy()

# OA
handc.line_gen(X_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double)
    ),
              Y_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
                  ,
              O.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
              A.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
              n, m)
x_OA, y_OA = X_l.copy(), Y_l.copy()
```

Python + C Integration (Part 5)

```
# OB
handc.line_gen(X_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double)
    ),
               Y_l.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
                   ,
               O.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
               B.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
               n, m)
x_OB, y_OB = X_l.copy(), Y_l.copy()

# Plotting
plt.figure()
plt.plot(x_AB, y_AB, "g--", label="Line AB")
plt.plot(x_OA, y_OA, "r--", label="Line OA")
plt.plot(x_OB, y_OB, "b--", label="Line OB")
```

Python + C Integration (Part 6)

```python
# Points
plt.scatter(A[0], A[1], color="blue", s=50)
plt.scatter(B[0], B[1], color="red", s=50)
plt.scatter(O[0], O[1], color="green", s=50)

# Labels
plt.annotate(f"A({A[0]:.0f},{A[1]:.0f})", (A[0], A[1]),
    textcoords="offset points", xytext=(-20,10))
plt.annotate(f"B({B[0]:.0f},{B[1]:.0f})", (B[0], B[1]),
    textcoords="offset points", xytext=(10,-15))
plt.annotate(f"O({O[0]:.0f},{O[1]:.0f})", (O[0], O[1]),
    textcoords="offset points", xytext=(10,10))
```

Python + C Integration (Part 7)

```
# Equal aspect ratio
plt.gca().set_aspect("equal", adjustable="box")
plt.xlim([-8,8])
plt.ylim([0,12])

plt.xlabel("X")
plt.ylabel("Y")
plt.title("Point O on y-axis equidistant from A and B")
plt.legend(loc="upper left")
plt.grid(True)

# Save & show
plt.savefig("../figs/equidistant_graph.png")
plt.show()
```