1.5.21

Kavin B-EE25BTECH11033

August 23,2025

Question

Find the ratio in which $\mathbf{P}(4, m)$ divides the line segment joining the points $\mathbf{A}(2, 3)$ and $\mathbf{B}(6, -3)$. Hence, find m.

Theoretical Solution

Let the vector $\mathbf{P}$ be

$$\mathbf{P} = \begin{pmatrix} 4 \\ m \end{pmatrix} , \tag{1}$$

Given the points,

$$\mathbf{A} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \mathbf{B} = \begin{pmatrix} 6 \\ -3 \end{pmatrix} \tag{2}$$

we can use the section formula to find the ratio first and then we can compute the value of m.

Equation

Section formula for a vector P which divides the line formed by vectors A and B in the ratio k:1 is given by

$$\mathbf{P} = \frac{k\mathbf{B} + \mathbf{A}}{k+1} \tag{3}$$

Theoretical Solution

Using section formula,

$$\begin{pmatrix} 4 \\ m \end{pmatrix} = \frac{\begin{pmatrix} 2 \\ 3 \end{pmatrix} + k\begin{pmatrix} 6 \\ -3 \end{pmatrix}}{1+k} \tag{4}$$

$$\implies \begin{pmatrix} 4 \\ m \end{pmatrix} + k\begin{pmatrix} 4 \\ m \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} + k\begin{pmatrix} 6 \\ -3 \end{pmatrix} \tag{5}$$

$$\implies k\begin{pmatrix} 2 \\ -3-m \end{pmatrix} = \begin{pmatrix} 2 \\ m-3 \end{pmatrix} \tag{6}$$

$$\text{or, } k = \frac{1}{1}. \tag{7}$$

$$\implies m = 0. \tag{8}$$

C Code - A function to find the value of m

```c
#include <stdio.h>

float findM(float Ax, float Ay, float Bx, float By, float Px) {
    float k = (Px - Ax) / (Bx - Px);
    float m = (k * By + Ay) / (k + 1);
    return m;
}
```

Python Code

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

c_lib=ctypes.CDLL('./code.so')

# Define the argument types for the findM function
c_lib.findM.argtypes = [ctypes.POINTER(ctypes.c_float), ctypes.
    POINTER(ctypes.c_float),ctypes.POINTER(ctypes.c_float),
    ctypes.POINTER(ctypes.c_float),ctypes.POINTER(ctypes.c_float)
    ]
# Define the return type of the findM function
c_lib.findM.restype = ctypes.c_float
```

Python Code

```
# --- Define Points and Calculate 'm' using C function ---

# Define the coordinates for the endpoints A and B
A = np.array([2.0, 3.0])
B = np.array([6.0, -3.0])
# Define the known x-coordinate for the dividing point P
Px = 4.0
# Call the C function to get the value of m
m_value = c_lib.findM(
    ctypes.c_float(A[0]), # Ax
    ctypes.c_float(A[1]), # Ay
    ctypes.c_float(B[0]), # Bx
    ctypes.c_float(B[1]), # By
    ctypes.c_float(Px) # Px
)
```

Python Code

```
# Create the dividing point P with the calculated 'm'
P_dividing = np.array([Px, m_value])
def find_ratio(point_A, point_B, dividing_point):

    # Ensure all inputs are numpy arrays for vector operations
    A_vec = np.array(point_A)
    B_vec = np.array(point_B)
    P_vec = np.array(dividing_point)

    # Calculate the ratio vector. If the points are collinear,
    # the ratio will be consistent for both x and y components.
    # We add a small epsilon to avoid division by zero if P
        coincides with B.
    epsilon = 1e-9
    ratio_vector = (P_vec - A_vec) / (B_vec - P_vec + epsilon)
    return ratio_vector
```

Python Code

```
# Calculate and print the ratio
ratio = find_ratio(A, B, P_dividing)
print(f'Point {tuple(P_dividing)} divides the line AB in the
    ratio: {ratio[0]}:{ratio[1]}')

def generate_line_segment(point1, point2, num_points=10):
    """Generates points to plot a line segment between two points
        ."""
    dim = point1.shape[0]
    line_segment = np.zeros((dim, num_points))
    lambda_vals = np.linspace(0, 1, num_points)
    for i in range(num_points):
        temp = point1 + lambda_vals[i] * (point2 - point1)
        line_segment[:, i] = temp.T
    return line_segment

# Generate the line segment for plotting
x_AB = generate_line_segment(A, B)
```

Python Code

```
# --- Plotting ---
plt.plot(x_AB[0, :], x_AB[1, :], label='$AB$')

# Plot the points A, B, and the dividing point P
all_points = np.vstack((A, B, P_dividing)).T
plt.scatter(all_points[0, :], all_points[1, :])

# Add labels for the points
point_labels = ['A (2,3)', 'B (6,-3)', 'P (4,0)']
for i, txt in enumerate(point_labels):
    plt.annotate(txt, # text to display
                 (all_points[0, i], all_points[1, i]), # point to
                     label
                 textcoords="offset points", # position of the
                     text
                 xytext=(10, 5), # distance from text to points (x
                     ,y)
                 ha='center') # horizontal alignment
```

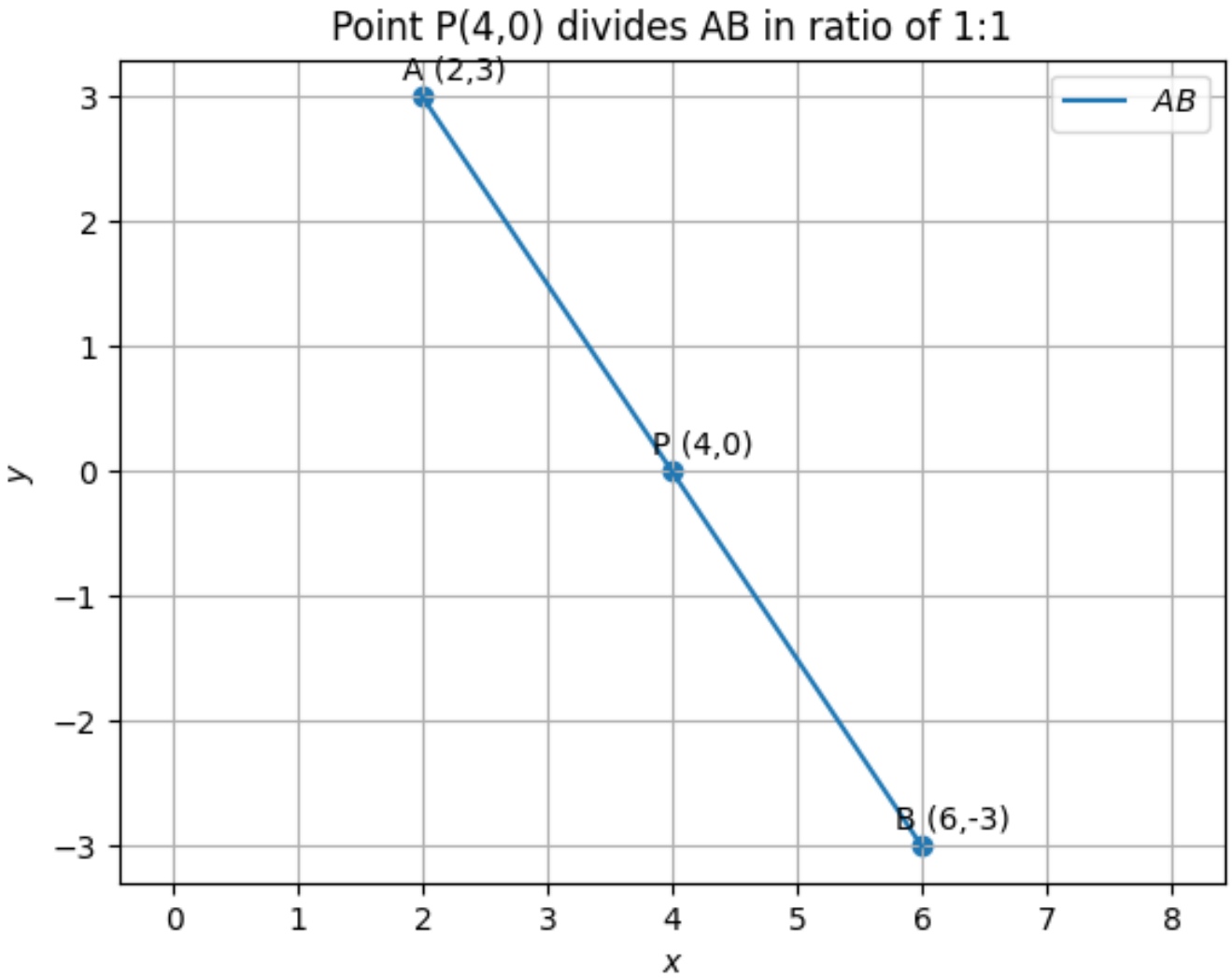
Python Code

```
# Set plot details
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.title('Point P(4,0) divides AB in ratio of 1:1')
plt.legend(loc='best')
plt.grid(True)
plt.axis('equal')

# Save the plot to a file
plt.savefig('../figs/fig.png')

# Display the plot
plt.show()
```

Plot


Point P(4,0) divides AB in ratio of 1:1
A (2,3)
P (4,0)
B (6,-3)
AB
x
y
0
1
2
3
4
5
6
7
8
3
2
1
0
−1
−2
−3