2.4.29

Kavin B-EE25BTECH11033

August 26,2025

Question

The points $\mathbf{A}(2,9), \mathbf{B}(a,5)$ and $\mathbf{C}(5,5)$ are the vertices of a triangle **ABC** right angled at **B**. Find the values of a and hence the area of $\triangle\mathbf{ABC}$.

Theoretical Solution

Given the points,

$$\mathbf{A} = \begin{pmatrix} 2 \\ 9 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} a \\ 5 \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \tag{1}$$

Also it is given that the triangle **ABC** right angled at **B**.

$\therefore$ The vectors **BA** and **BC** are perpendicular.

Formulae

The angle θ between $\mathbf{BA}, \mathbf{BC}$, is given by

$$\cos\theta = \frac{(\mathbf{BA})^\top (\mathbf{BC})}{\|\mathbf{BA}\| \, \|\mathbf{BC}\|} \tag{2}$$

Theoretical Solution

Here $\theta = 90$.

$$\implies (\mathbf{BA})^\top (\mathbf{BC}) = 0 \tag{3}$$

$$\mathbf{BA} = \mathbf{A} \text{ - } \mathbf{B} = \begin{pmatrix} 2 - a \\ 4 \end{pmatrix}$$

$$\mathbf{BC} = \mathbf{C} \text{ - } \mathbf{B} = \begin{pmatrix} 5 - a \\ 0 \end{pmatrix}$$

Theoretical Solution

$$\implies \begin{pmatrix} 2-a \\ 4 \end{pmatrix}^{\top} \begin{pmatrix} 5-a \\ 0 \end{pmatrix} = 0 \tag{4}$$

$$\implies \begin{pmatrix} 2-a & 4 \end{pmatrix} \begin{pmatrix} 5-a \\ 0 \end{pmatrix} = 0 \tag{5}$$

$$\implies (2-a)(5-a) + (4 \times 0) = 0 \tag{6}$$

$$\implies (2-a)(5-a) = 0 \tag{7}$$

$$\implies a = 2 \tag{8}$$

Here $a = 5$ is not considered because when $a = 5$, the points **B** and **C** will be the same and hence a triangle cannot be formed.

$$\mathbf{B} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

Formulae

The area of $\triangle ABC$ is given by

$$Area = \frac{1}{2}\left\|(\mathbf{A}-\mathbf{B})\times(\mathbf{A}-\mathbf{C})\right\| \tag{9}$$

Theoretical Solution

$$(\mathbf{A}-\mathbf{B}) = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$$

$$(\mathbf{A}-\mathbf{C}) = \begin{pmatrix} -3 \\ 4 \end{pmatrix}$$

$$\implies Area = \frac{1}{2}\left\| \begin{pmatrix} 0 \\ 4 \end{pmatrix} \times \begin{pmatrix} -3 \\ 4 \end{pmatrix} \right\| \tag{10}$$

$$\implies Area = \frac{1}{2}\left\| 0 + 12 \right\| \tag{11}$$

$$\implies Area = 6 \tag{12}$$

Hence the area of $\triangle\mathbf{ABC}$ is 6 sq.units.

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

typedef struct {
    double x;
    double y;
} Point;

typedef struct {
    int count;
    double solution1;
    double solution2;
} Solutions;
```

C Code - A function to find the value of a

```c
Solutions solveForA(Point A, double B_y, Point C) {
    Solutions result = {0, 0.0, 0.0};

    double P = 1.0;
    double Q = -(A.x + C.x);
    double R = (A.x * C.x) + (A.y - B_y) * (C.y - B_y);
    double discriminant = Q*Q - 4*P*R;

    if (discriminant < 0) {
        return result;
    }

    double a1 = (-Q + sqrt(discriminant)) / (2*P);
    double a2 = (-Q - sqrt(discriminant)) / (2*P);
```

C Code - A function to find the value of a

```c
    int a1_is_valid = !(a1 == A.x && B_y == A.y) && !(a1 == C.x
        && B_y == C.y);
    int a2_is_valid = !(a2 == A.x && B_y == A.y) && !(a2 == C.x
        && B_y == C.y);
    if (a1_is_valid) {
        result.solution1 = a1;
        result.count++;
    }
    if (discriminant > 1e-9 && a2_is_valid) {
        if (result.count == 0) {
            result.solution1 = a2;
        } else {
            result.solution2 = a2;
        }
        result.count++;
    }
    return result;
}
```

C Code - A function to find the value of a

```c

double getValidA() {
    Point A = {2.0, 9.0};
    Point C = {5.0, 5.0};
    double B_y = 5.0;

    Solutions solutions = solveForA(A, B_y, C);

    if (solutions.count > 0) {
        return solutions.solution1;
    }

    return 2.0;
}
```

Python Code

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

# Load the shared library
lib = ctypes.CDLL('./code.so')

# Define C types matching the exact structure
class Point(ctypes.Structure):
    _fields_ = [("x", ctypes.c_double),
                ("y", ctypes.c_double)]

class Solutions(ctypes.Structure):
    _fields_ = [("count", ctypes.c_int),
                ("solution1", ctypes.c_double),
                ("solution2", ctypes.c_double)]
```

Python Code

```
# Set up function prototypes exactly as in C
lib.solveForA.argtypes = [Point, ctypes.c_double, Point]
lib.solveForA.restype = Solutions

lib.getValidA.argtypes = []
lib.getValidA.restype = ctypes.c_double

# Get the value of a from C library using the exact function
a_value = lib.getValidA()
print(f"Value of a from C library: {a_value}")
# Define points
A = np.array([2, 9])
B = np.array([a_value, 5])
C = np.array([5, 5])
print(f"Coordinates:")
print(f"A({A[0]}, {A[1]})")
print(f"B({B[0]}, {B[1]})")
print(f"C({C[0]}, {C[1]})")
```

Python Code

```python
# Function to generate line points
def line_gen(P, Q):
    return np.column_stack((P, Q))

# Calculate triangle properties
c = LA.norm(A - B)
a = LA.norm(B - C)
b = LA.norm(C - A)
print(f"\nSide lengths:")
print(f"AB = {c:.2f}")
print(f"BC = {a:.2f}")
print(f"CA = {b:.2f}")
```

Python Code

```
# Calculate area (since it's right-angled at B)
area = 0.5 * a * c
print(f"\nArea of triangle ABC: {area:.2f}")

# Generate lines
x_AB = line_gen(A, B)
x_BC = line_gen(B, C)
x_CA = line_gen(C, A)

# Plotting
plt.figure(figsize=(10, 8))
plt.plot(x_AB[0,:], x_AB[1,:], label='$AB$', linewidth=3, color='
    blue')
plt.plot(x_BC[0,:], x_BC[1,:], label='$BC$', linewidth=3, color='
    green')
plt.plot(x_CA[0,:], x_CA[1,:], label='$CA$', linewidth=3, color='
    red')
```

Python Code

```
# Labeling the coordinates
tri_coords = np.column_stack((A, B, C))
plt.scatter(tri_coords[0,:], tri_coords[1,:], color='black', s
    =150, zorder=5)

vert_labels = ['A','B','C']
for i, txt in enumerate(vert_labels):
    plt.annotate(txt,
                 (tri_coords[0,i], tri_coords[1,i]),
                 textcoords="offset points",
                 xytext=(0,15),
                 ha='center',
                 fontsize=14,
                 fontweight='bold',
                 bbox=dict(boxstyle="round,pad=0.3", facecolor="
                     yellow", alpha=0.7))
```

Python Code

```python
plt.xlabel('$x$', fontsize=14)
plt.ylabel('$y$', fontsize=14)
plt.legend(loc='upper right', fontsize=12)
plt.grid(True, alpha=0.3, linestyle='--')
plt.axis('equal')

# Set appropriate limits with some padding
plt.xlim(min(tri_coords[0,:]) - 1, max(tri_coords[0,:]) + 1)
plt.ylim(min(tri_coords[1,:]) - 1, max(tri_coords[1,:]) + 1)
```

Python Code

```
# Add right angle marker at B
angle_x = B[0] + 0.5
angle_y = B[1] + 0.5
plt.plot([B[0], angle_x], [B[1], B[1]], 'k--', alpha=0.5)
plt.plot([B[0], B[0]], [B[1], angle_y], 'k--', alpha=0.5)
plt.text(B[0] + 0.3, B[1] + 0.3, '90', fontsize=12, fontweight='
    bold')

plt.tight_layout()
plt.savefig('../figs/fig.png')
plt.show()
```

Plot