8.4.15

BEERAM MADHURI - EE25BTECH11012

October 2025

Question

In a triangle ABC with fixed base BC, the vertex A moves such that

$$\cos B + \cos C = 4 \sin^2 \frac{A}{2}.$$

If a, b and c denote the lengths of the sides of the triangle opposite to the angles A, B and C, respectively, then

a) $b + c = 4a$

b) $b + c = 2a$

c) locus of the point A is an ellipse

d) locus of the point A is a pair of straight lines

finding the locus of A:

Given,

$$\cos B + \cos C = 4 \sin^2 \frac{A}{2} \tag{1}$$

$$\cos B + \cos C = 4 \frac{(1 - \cos A)}{2} \tag{2}$$

$$2 \cos A + \cos B + \cos C = 2 \tag{3}$$

By Projection rule:

$$c \cos B + b \cos C = a \tag{4}$$

$$c \cos A + a \cos C = b \tag{5}$$

$$b \cos A + a \cos B = c \tag{6}$$

Combining all these into a Matrix:

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & c & b \\ c & 0 & a \\ b & a & 0 \end{bmatrix} \begin{bmatrix} \cos A \\ \cos B \\ \cos C \end{bmatrix} = \begin{bmatrix} 2 \\ a \\ b \\ c \end{bmatrix} \tag{7}$$

$$AX = b \tag{8}$$

for this system to be consistent

$$\text{rank}(A) = \text{rank}([A|b]) \leq 3 \tag{9}$$

if rank$([A|b]) \leq 3$
its columns are linearly dependent. $\therefore$ det$([A|b]) = 0$

$$(2a - b - c)(-a^2 + (b - c)a + (b - c)^2) = 0 \tag{10}$$

$$\text{as, a, b, c are sides of triangle} \tag{11}$$

$$-a^2 + (b - c)a + (b - c)^2 \neq 0 \tag{12}$$

$$\therefore 2a - b - c = 0 \tag{13}$$

$$\therefore b + c = 2a \tag{14}$$

$$b + c = 2a \tag{15}$$

$$\|A - C\| + \|A - B\| = 2\|B - C\| \tag{16}$$

given B and C are fixed.
∴ Locus of 'A' is an ellipse,
as, the sum of its distances from 2 fixed points is constant (represents an ellipse).
∴ Options b and c are correct.

Python Code

```python
import numpy as np
import matplotlib.pyplot as plt
# --- 1. Define the fixed base and derive ellipse parameters ---
# Let the fixed base BC be on the x-axis, centered at the origin.
# B = (-k, 0), C = (k, 0). We can choose k=2 for a clear visual.
k = 2
B = np.array([-k, 0])
C = np.array([k, 0])
# The length of the base 'a' is the distance between B and C.
# Note: 'a' here is the side length, not the semi-major axis of
    the ellipse.
side_a = np.linalg.norm(C - B) # side_a = 2k = 4
```

Python Code

```python
# --- 2. Use the derived condition to find the ellipse's properties ---
# From the solution, we found b + c = 2a.
# b = distance(A, C) and c = distance(A, B).
# So, distance(A, C) + distance(A, B) = 2 * side_a = 2 * (4) = 8.
# This is the definition of an ellipse with foci at B and C.
# The constant sum of distances is 2 * a_ellipse (semi-major axis).
constant_sum = 2 * side_a
a_ellipse = constant_sum / 2 # a_ellipse = side_a = 4
```

Python Code

```
# The distance from the center to a focus is c_ellipse.
c_ellipse = k # c_ellipse = 2
# For an ellipse, a_ellipse^2 = b_ellipse^2 + c_ellipse^2
b_ellipse = np.sqrt(a_ellipse**2 - c_ellipse**2) # Semi-minor axis
# --- 3. Generate points for the ellipse (locus of A) ---
t = np.linspace(0, 2 * np.pi, 300)
x_ellipse = a_ellipse * np.cos(t)
y_ellipse = b_ellipse * np.sin(t)
```

Python Code

```python
# --- 4. Create the plot ---
plt.style.use('seaborn-v0_8-whitegrid')
fig, ax = plt.subplots(figsize=(10, 8))
# Plot the locus of A
ax.plot(x_ellipse, y_ellipse, label='Locus of Vertex A (Ellipse)', color='dodgerblue', linewidth=2)
# Plot the fixed base BC and foci
ax.plot([B[0], C[0]], [B[1], C[1]], 'o', markersize=8, color='red', label='Foci (Fixed Base BC)')
ax.text(B[0], B[1] - 0.5, f'B({B[0]}, {B[1]})', ha='center', fontsize=12)
ax.text(C[0], C[1] - 0.5, f'C({C[0]}, {C[1]})', ha='center', fontsize=12)
```

Python Code

```python
# --- 5. Illustrate with an example triangle ---
# Choose an example point A on the ellipse (e.g., at t = 2/5)
t_example = 2 * np.pi / 5
A_example = np.array([a_ellipse * np.cos(t_example), b_ellipse * np.sin(t_example)])
# Draw the triangle ABC
triangle_x = [A_example[0], B[0], C[0], A_example[0]]
triangle_y = [A_example[1], B[1], C[1], A_example[1]]
ax.plot(triangle_x, triangle_y, 'g--', label='Example Triangle ABC', linewidth=1.5)
```

Python Code

```python
ax.plot(A_example[0], A_example[1], 'go', markersize=6)
ax.text(A_example[0], A_example[1] + 0.3, 'A(x, y)', ha='center',
     fontsize=12)

# Verify the condition for the example point and display it
side_b = np.linalg.norm(A_example - C)
side_c = np.linalg.norm(A_example - B)
info_text = (
    f"Condition: $b + c = 2a$\n\n"
    f"Side $a$ (distance BC) = {side_a:.2f}\n"
    f"Side $b$ (distance AC) = {side_b:.2f}\n"
    f"Side $c$ (distance AB) = {side_c:.2f}\n\n"
```

Python Code

```
    f"Check: ${side_b:.2f} + {side_c:.2f} = {side_b + side_c:.2f}$\n"
    f"Required: $2 \\times a = 2 \\times {side_a:.2f} = {2 * side_a:.2f}$"
)
ax.text(0.95, 0.05, info_text, transform=ax.transAxes, fontsize=11,
       verticalalignment='bottom', horizontalalignment='right',
       bbox=dict(boxstyle='round,pad=0.5', fc='aliceblue', alpha=0.9))
```

Python Code

```python
# --- 6. Finalize the plot ---
ax.set_aspect('equal', adjustable='box')
ax.set_title('Locus of Vertex A is an Ellipse', fontsize=16)
ax.set_xlabel('x-axis', fontsize=12)
ax.set_ylabel('y-axis', fontsize=12)
ax.legend(loc='upper left')
plt.show()
```

C Code

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

// Define PI for trigonometric calculations if not already defined
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

double to_radians(double degrees) {
    return degrees * M_PI / 180.0;
}
```

C Code

```c
void check_triangle_properties(double angle_A_deg) {
    printf("--- Checking for A = %.2f degrees ---\n", angle_A_deg);

    // For a valid triangle, the simplified condition cos((B-C)/2) = 2*sin(A/2) must hold.
    // Since the maximum value of cosine is 1, we must have 2*sin(A/2) <= 1.
    // This implies sin(A/2) <= 0.5, which means A/2 <= 30 degrees, so A <= 60 degrees.
```

C Code

```c
    if (angle_A_deg > 60.0 || angle_A_deg <= 0) {
        printf("A triangle with this condition cannot exist for A > 60 degrees or A <= 0.\n");
        printf("The expression 2*sin(A/2) would be > 1, which is impossible for a cosine value.\n\n");
        return;
    }
    // Convert angle A to radians for use in math functions
    double A_rad = to_radians(angle_A_deg);
```

C Code

```c
    // --- Step 1: Find angles B and C that satisfy the condition ---
    // From the simplified relation: cos((B-C)/2) = 2*sin(A/2)
    double val_for_acos = 2.0 * sin(A_rad / 2.0);
    double B_minus_C_half_rad = acos(val_for_acos);
    // We also know A + B + C = PI radians, so (B+C)/2 = PI/2 - A/2
    double B_plus_C_half_rad = M_PI / 2.0 - A_rad / 2.0;
    // Solve for B and C
    double B_rad = B_plus_C_half_rad + B_minus_C_half_rad;
    double C_rad = B_plus_C_half_rad - B_minus_C_half_rad;
```

C Code

```c
// --- Step 2: Verify the original trigonometric identity ---
double lhs_identity = cos(B_rad) + cos(C_rad);
double rhs_identity = 4.0 * pow(sin(A_rad / 2.0), 2);
printf("Verification of given identity:\n");
printf(" LHS (cos B + cos C) = %f\n", lhs_identity);
printf(" RHS (4 * sin^2(A/2)) = %f\n", rhs_identity);
// --- Step 3: Verify the derived side relationship b + c = 2 a ---
```

C Code

```c
    // Using the Sine Rule, we can use the sines of the angles as relative side lengths.
    double a = sin(A_rad);
    double b = sin(B_rad);
    double c = sin(C_rad);

    double b_plus_c = b + c;
    double two_a = 2.0 * a;
```

C Code

```c
    printf("\nVerification of the side relationship (b + c = 2a):\n");
    printf(" Relative side lengths (a=sinA, b=sinB, c=sinC):\n");
    printf(" b + c = %f\n", b_plus_c);
    printf(" 2 * a = %f\n", two_a);

    // Check for equality using a small tolerance for floating-point errors
```

C Code

```c
    if (fabs(b_plus_c - two_a) < 1e-9) {
        printf("\nResult: The relationship b + c = 2a holds true.
             \n\n");
    } else {
        printf("\nResult: The relationship b + c = 2a does NOT
            hold. \n\n");
    }
}
```

C Code

```c
int main() {
    printf("## Solution Verification for Triangle Problem ##\n\n");
    printf("This program verifies the two correct conclusions from the problem:\n");
    printf(" b) b + c = 2a\n");
    printf(" c) locus of the point A is an ellipse\n\n");
    // --- Part 1: Numerical Verification of b + c = 2a ---
    printf("### Part 1: Verifying the side relationship b + c = 2a ###\n\n");
    printf("The code will now test the derived relationship for various valid angles of A.\n\n");
```

C Code

```
    // Check for a few valid angles of A (where A <= 60 degrees)
    check_triangle_properties(60.0); // Special case: equilateral triangle
    check_triangle_properties(45.0);
    check_triangle_properties(30.0);
    // Check an invalid angle to show the constraint
    check_triangle_properties(90.0);
    // --- Part 2: Explanation of the Locus of A ---
    printf("### Part 2: Determining the Locus of Point A ###\n\n");
    printf("1. The problem states that the base BC is fixed. Let its length be 'a'.\n");
```

C Code

```c
    printf("2. From the derivation, we found the relationship b + c = 2a.\n");
    printf("3. The side 'b' is the distance AC, and 'c' is the distance AB.\n");
    printf("4. Therefore, the condition is AB + AC = 2a.\n");
    printf("5. Since 'a' is a fixed length, '2a' is a constant value.\n\n");
    printf("This is the geometric definition of an ellipse: the set of all points (A) for which the sum of the distances to two fixed points (the foci, B and C) is a constant (2a).\n\n");
    printf("Conclusion: The locus of point A is an ellipse. \n");
    return 0;
}
```

Python and C Code

```python
import math

# Define PI for trigonometric calculations
PI = math.pi

def to_radians(degrees: float) -> float:
    return degrees * PI / 180.0

def check_triangle_properties(angle_A_deg: float):
    print(f"--- Checking for A = {angle_A_deg:.2f} degrees ---")
```

Python and C Code

```
    if angle_A_deg > 60.0 or angle_A_deg <= 0:
        print("A triangle with this condition cannot exist for A > 60 degrees or A <= 0.")
        print("The expression 2*sin(A/2) would be > 1, which is impossible for a cosine value.\n")
        return

    # Convert angle A to radians
    A_rad = to_radians(angle_A_deg)
```

Python and C Code

```
# Step 1: Find angles B and C that satisfy the condition
val_for_acos = 2.0 * math.sin(A_rad / 2.0)

# Check if acos input is in valid domain
if val_for_acos > 1.0:
    print("acos argument exceeds 1. Invalid configuration.\n")
    return

B_minus_C_half_rad = math.acos(val_for_acos)
B_plus_C_half_rad = PI / 2.0 - A_rad / 2.0
```

Python and C Code

```python
    B_rad = B_plus_C_half_rad + B_minus_C_half_rad
    C_rad = B_plus_C_half_rad - B_minus_C_half_rad

    # Step 2: Verify the original trigonometric identity
    lhs_identity = math.cos(B_rad) + math.cos(C_rad)
    rhs_identity = 4.0 * (math.sin(A_rad / 2.0) ** 2)

    print("Verification of given identity:")
    print(f" LHS (cos B + cos C) = {lhs_identity}")
    print(f" RHS (4 * sin^2(A/2)) = {rhs_identity}")
```

Python and C Code

```python
# Step 3: Verify the derived side relationship b + c = 2a
a = math.sin(A_rad)
b = math.sin(B_rad)
c = math.sin(C_rad)

b_plus_c = b + c
two_a = 2.0 * a
print("\nVerification of the side relationship (b + c = 2a):")
print(" Relative side lengths (a=sinA, b=sinB, c=sinC):")
print(f" b + c = {b_plus_c}")
print(f" 2 * a = {two_a}")
```

Python and C Code

```python
    # Allow small numerical tolerance
    if abs(b_plus_c - two_a) < 1e-9:
        print("\nResult: The relationship b + c = 2a holds true.\n")
    else:
        print("\nResult: The relationship b + c = 2a does NOT hold.\n")
```

Python and C Code

```
def main():
    print("## Solution Verification for Triangle Problem ##\n")
    print("This program verifies the two correct conclusions from the problem:")
    print(" b) b + c = 2a")
    print(" c) locus of the point A is an ellipse\n")
    # Part 1: Numerical Verification
    print("### Part 1: Verifying the side relationship b + c = 2a ###\n")
    print("The code will now test the derived relationship for various valid angles of A.\n")
```

Python and C Code

```
    # Test various angles
    check_triangle_properties(60.0) # Equilateral triangle
    check_triangle_properties(45.0)
    check_triangle_properties(30.0)

    # Invalid case
    check_triangle_properties(90.0)
```

Python and C Code

```python
# Part 2: Locus Explanation
print("### Part 2: Determining the Locus of Point A ###\n")
print("1. The problem states that the base BC is fixed. Let its length be 'a'.")
print("2. From the derivation, we found the relationship b + c = 2a.")
print("3. The side 'b' is the distance AC, and 'c' is the distance AB.")
print("4. Therefore, the condition is AB + AC = 2a.")
```

Python and C Code

```python
    print("5. Since 'a' is a fixed length, '2a' is a constant value.\n")
    print("This is the geometric definition of an ellipse:")
    print("The set of all points (A) for which the sum of the distances to two fixed points (the foci, B and C) is a constant (2a).\n")
    print("Conclusion: The locus of point A is an ellipse.\n")

if __name__ == "__main__":
    main()
```

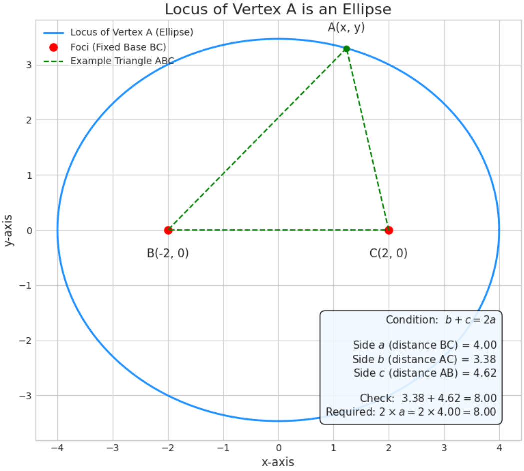
Locus of Vertex A is an Ellipse
Locus of Vertex A (Ellipse)
Foci (Fixed Base BC)
Example Triangle ABC
A(x, y)
B(-2, 0)
C(2, 0)
Condition: $b + c = 2a$
Side a (distance BC) = 4.00
Side b (distance AC) = 3.38
Side c (distance AB) = 4.62
Check: $3.38 + 4.62 = 8.00$
Required: $2 \times a = 2 \times 4.00 = 8.00$
x-axis
y-axis
−4
−3
−2
−1
0
1
2
3
4
3
2
1
0
−1
−2
−3

Figure: Plot