2.6.9

AI25BTECH11003 - Bhavesh Gaikwad

August 26,2025

Question

The area of a triangle with vertices A(3,0), B(7,0) and C(8,4) is?

Theoretical Solution

Given: $A(3,0),\ B(7,0),\ C(8,4).$

$$\mathbf{B}-\mathbf{A}=\begin{pmatrix}7-3\\0-0\end{pmatrix}=\begin{pmatrix}4\\0\end{pmatrix},\qquad \mathbf{C}-\mathbf{A}=\begin{pmatrix}8-3\\4-0\end{pmatrix}=\begin{pmatrix}5\\4\end{pmatrix}.$$

$$|(\mathbf{B}-\mathbf{A})\times(\mathbf{C}-\mathbf{A})|=\left|\begin{pmatrix}|\mathbf{A}_{23} & \mathbf{B}_{23}|\\|\mathbf{A}_{31} & \mathbf{B}_{31}|\\|\mathbf{A}_{12} & \mathbf{B}_{12}|\end{pmatrix}\right|=16$$

Area $=\frac{1}{2}||(\mathbf{B}-\mathbf{A})\times(\mathbf{C}-\mathbf{A})||=8$

$$\boxed{Area\ of\ \ Triangle\ ABC\ \ =\ 8\,sq.units}\tag{1}$$

Python + C Code

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

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif


#include "matfun.h"
#include "geofun.h"

int main(void) {
    // Allocate 2x1 matrices for points
    double **A = createMat(2,1);
    double **B = createMat(2,1);
    double **C = createMat(2,1);
```

Python + C Code

```c
        // A(3,0), B(7,0), C(8,4) - correct matrix indexing
    A[0][0] = 3.0; A[1][0] = 0.0;
    B[0][0] = 7.0; B[1][0] = 0.0;
    C[0][0] = 8.0; C[1][0] = 4.0;


    // Vectors B-A and C-A as 2x1 matrices
    double **BA = Matsub(B, A, 2, 1);
    double **CA = Matsub(C, A, 2, 1);

    // Extract components for clarity - correct matrix indexing
    double BAx = BA[0][0], BAy = BA[1][0];
    double CAx = CA[0][0], CAy = CA[1][0];
```

Python + C Code

```c
    // 2D cross product magnitude |(B-A) x (C-A)| = |BAx*CAy - BAy*CAx|
    double cp = fabs(BAx*CAy - BAy*CAx);
    double area = 0.5 * cp;

    // Save to points.dat
    FILE *fp = fopen("points.dat", "w");
    if (!fp) {
        perror("points.dat");
        // Clean up on error
        freeMat(BA, 2); freeMat(CA, 2);
        freeMat(A, 2); freeMat(B, 2); freeMat(C, 2);
        return 1;
    }
```

Python + C Code

```
    fprintf(fp, "# Point_Name X Y\n");
    fprintf(fp, "A %.1f %.1f\n", A[0][0], A[1][0]);
    fprintf(fp, "B %.1f %.1f\n", B[0][0], B[1][0]);
    fprintf(fp, "C %.1f %.1f\n", C[0][0], C[1][0]);
    fclose(fp);
    printf("Wrote points.dat\n");

    // Free memory
    freeMat(BA, 2); freeMat(CA, 2);
    freeMat(A, 2); freeMat(B, 2); freeMat(C, 2);

    return 0;
}
```

Python + C Code

```python
import numpy as np
import matplotlib.pyplot as plt
import sys
import os
# Add the triangle folder to the path to import funcs.py
sys.path.append('triangle')
try:
    from funcs import *
except ImportError:
    print("Warning: Could not import from triangle/funcs.py. Using local functions.")
    # Define basic functions if funcs.py is not available
    def tri_sides(A, B, C):
        a = np.linalg.norm(B - C)
        b = np.linalg.norm(C - A)
        c = np.linalg.norm(A - B)
        return c, a, b
```

Python + C Code

```
def read_points_from_file(filename="points.dat"):
  """Read triangle vertices from points.dat file"""
  points = {}

  try:
      with open(filename, 'r') as file:
          for line in file:
              # Skip comments and empty lines
              if line.startswith('#') or line.strip() == '':
                  continue

              # Parse point data: Point_Name X Y
              parts = line.strip().split()
              if len(parts) == 3:
                  point_name = parts[0]
                  x = float(parts[1])
                  y = float(parts[2])
                  points[point_name] = np.array([x, y])
```

Python + C Code

```
      except FileNotFoundError:
        print(f"Error: {filename} not found!")
        return None

    return points

def calculate_triangle_area(A, B, C):
    """Calculate triangle area using cross product method"""
    BA = B - A
    CA = C - A

    # Cross product magnitude for 2D vectors
    cross_product = abs(BA[0] * CA[1] - BA[1] * CA[0])
    area = 0.5 * cross_product

    return area
```

Python + C Code

```python
def plot_triangle(A, B, C, area):
    """Plot the triangle with vertices and area annotation"""
    fig, ax = plt.subplots(1, 1, figsize=(10, 8))

    # Create triangle vertices array for plotting
    triangle_x = [A[0], B[0], C[0], A[0]] # Close the triangle
    triangle_y = [A[1], B[1], C[1], A[1]]

    # Plot the triangle with orange color
    ax.plot(triangle_x, triangle_y, 'orange', linewidth=2)
    ax.fill(triangle_x, triangle_y, alpha=0.3, color='orange')

    # Plot and label vertices
    ax.plot(A[0], A[1], 'ro', markersize=8)
    ax.plot(B[0], B[1], 'go', markersize=8)
    ax.plot(C[0], C[1], 'mo', markersize=8)
```

Python + C Code

```
# Add vertex labels
  ax.annotate(f'A({A[0]:.0f}, {A[1]:.0f})', (A[0], A[1]),
             xytext=(5, 5), textcoords='offset points',
                 fontsize=12, fontweight='bold')
  ax.annotate(f'B({B[0]:.0f}, {B[1]:.0f})', (B[0], B[1]),
             xytext=(5, -15), textcoords='offset points',
                 fontsize=12, fontweight='bold')
  ax.annotate(f'C({C[0]:.0f}, {C[1]:.0f})', (C[0], C[1]),
             xytext=(5, 5), textcoords='offset points',
                 fontsize=12, fontweight='bold')

  # Set grid and labels
  ax.grid(True, alpha=0.3)
  ax.set_xlabel('X', fontsize=12)
  ax.set_ylabel('Y', fontsize=12)
  ax.set_title('Triangle ABC', fontsize=14, fontweight='bold')
```

Python + C Code

```
    # Set axis limits with some padding
    x_min, x_max = min(A[0], B[0], C[0]) - 1, max(A[0], B[0], C[0]) + 1
    y_min, y_max = min(A[1], B[1], C[1]) - 1, max(A[1], B[1], C[1]) + 1
    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)

    # Equal aspect ratio
    ax.set_aspect('equal', adjustable='box')

    # Save the plot
    plt.tight_layout()
    plt.savefig('triangle_plot.png', dpi=300, bbox_inches='tight')
    plt.show()
```

Python + C Code

```
def main():
  print("Triangle Area Calculator and Plotter")
  print("=" * 40)

  # Read points from file
  points = read_points_from_file("points.dat")

  if points is None:
      print("Failed to read points from file.")
      return

  # Extract vertices
  try:
      A = points['A']
      B = points['B']
      C = points['C']
```

Python + C Code

```
        print(f"Points read from file:")
        print(f"A = ({A[0]:.1f}, {A[1]:.1f})")
        print(f"B = ({B[0]:.1f}, {B[1]:.1f})")
        print(f"C = ({C[0]:.1f}, {C[1]:.1f})")
        print()

    except KeyError as e:
        print(f"Error: Missing point {e} in points.dat")
        return

    # Calculate area using the cross product method
    area = calculate_triangle_area(A, B, C)

    # Display calculation steps (similar to the image)
    BA = B - A
    CA = C - A
```

Python + C Code

```python

print("Solution:")
print(f"Given: A({A[0]:.0f},{A[1]:.0f}), B({B[0]:.0f},{B[1]:.0f}), C({C[0]:.0f},{C[1]:.0f})")
print()
print(f"B - A = ({BA[0]:.0f}, {BA[1]:.0f})")
print(f"C - A = ({CA[0]:.0f}, {CA[1]:.0f})")
print()

cross_product = abs(BA[0] * CA[1] - BA[1] * CA[0])
print(f"|(B - A) * (C - A)| = |{BA[0]:.0f} * {CA[1]:.0f} - {BA[1]:.0f} * {CA[0]:.0f}| = {cross_product:.0f}")
print()
print(f"Area = (1/2) * |(B - A) * (C - A)| = (1/2) * {cross_product:.0f} = {area:.0f}")
print()
print(f"Area of Triangle ABC = {area:.2f} sq.units")
print()
```

Python + C Code

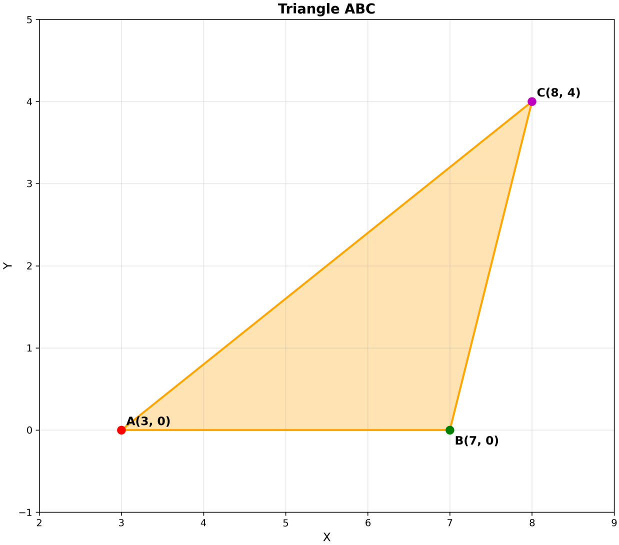
```
      # Try to calculate side lengths if funcs.py is available
    try:
        sides = tri_sides(A, B, C)
        print(f"Triangle side lengths:")
        print(f"AB (c) = {sides[0]:.2f}")
        print(f"BC (a) = {sides[1]:.2f}")
        print(f"CA (b) = {sides[2]:.2f}")
        print()
    except NameError:
        print("tri_sides function not available")

    # Plot the triangle
    plot_triangle(A, B, C, area)

    print("Triangle plot saved as 'triangle_plot.png'")

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

Graph


Triangle ABC
C(8, 4)
A(3, 0)
B(7, 0)
Y
X