2.10.70

Nipun Dasari - EE25BTECH11042

September 16, 2025

Question

In a $\triangle$ABC, D and E are points on BC and AC respectively, such that $BD = 2DC$ and $AE = 3EC$. Let P be the point of intersection of AD and BE. Find $\frac{BP}{PE}$ using vector methods.

Theoretical Solution

Let vertex A be the origin. The position vectors are:

$$\mathbf{a} = \mathbf{0}, \quad \mathbf{b} = \text{Position vector of B}, \quad \mathbf{c} = \text{Position vector of C} \tag{1}$$

Position vector of point D, which divides BC in the ratio 2 : 1:

$$\mathbf{d} = \frac{1\mathbf{b} + 2\mathbf{c}}{2+1} = \frac{\mathbf{b} + 2\mathbf{c}}{3} \tag{2}$$

Position vector of point E, which divides AC in the ratio 3 : 1:

$$\mathbf{e} = \frac{1\mathbf{a} + 3\mathbf{c}}{3+1} = \frac{3\mathbf{c}}{4} \tag{3}$$

Let P divide AD in the ratio $AP : PD = \lambda : 1$. The position vector of P is:

Theoretical Solution

$$\mathbf{p} = \frac{1\mathbf{a} + \lambda\mathbf{d}}{\lambda + 1} = \frac{\lambda}{\lambda + 1}\mathbf{d} \tag{4}$$

Substituting for **d**:

$$\mathbf{p} = \left(\frac{\lambda}{\lambda + 1}\right)\left(\frac{\mathbf{b} + 2\mathbf{c}}{3}\right) = \frac{\lambda}{3\left(\lambda + 1\right)}\mathbf{b} + \frac{2\lambda}{3\left(\lambda + 1\right)}\mathbf{c} \tag{5}$$

Let P divide BE in the ratio $BP : PE = \mu : 1$. The position vector of P is:

$$\mathbf{p} = \frac{1\mathbf{b} + \mu\mathbf{e}}{\mu + 1} \tag{6}$$

Substituting for **e**:

Theoretical Solution

$$\mathbf{p} = \frac{1}{\mu+1}\mathbf{b} + \frac{\mu}{\mu+1}\mathbf{e} = \frac{1}{\mu+1}\mathbf{b} + \frac{\mu}{\mu+1}\left(\frac{3\mathbf{c}}{4}\right) = \frac{1}{\mu+1}\mathbf{b} + \frac{3\mu}{4\left(\mu+1\right)}\mathbf{c} \tag{7}$$

Since **b** and **c** are non-collinear, we equate their coefficients from (5) and (7).
Coefficients of **b**:

$$\frac{\lambda}{3\left(\lambda+1\right)} = \frac{1}{\mu+1} \tag{8}$$

Coefficients of **c**:

$$\frac{2\lambda}{3\left(\lambda+1\right)} = \frac{3\mu}{4\left(\mu+1\right)} \tag{9}$$

Theoretical Solution

From (8) and (9), we can see that the LHS of (9) is twice the LHS of (8).

$$2\left(\frac{1}{\mu+1}\right) = \frac{3\mu}{4\left(\mu+1\right)} \tag{10}$$

Multiplying both sides by $4\left(\mu+1\right)$:

$$8 = 3\mu \implies \mu = \frac{8}{3} \tag{11}$$

The required ratio is $BP : PE = \mu : 1$.

$$\therefore \frac{BP}{PE} = \mu = \frac{8}{3} \tag{12}$$

C Code - formula function

```c
void calculate_points_from_arrays(
double* input_A, // Pointer to a 2-element array [
    Ax, Ay]
double* input_B, // Pointer to a 2-element array [
    Bx, By]
double* input_C, // Pointer to a 2-element array [
    Cx, Cy]
double* output_points // Pointer to a 12-element
    array to be filled
) {
        // Unpack input points for clarity
        double Ax = input_A[0], Ay = input_A[1];
        double Bx = input_B[0], By = input_B[1];
        double Cx = input_C[0], Cy = input_C[1];
```

C Code - formula function

```c
// Calculate Point D
double Dx = (1.0 * Bx + 2.0 * Cx) / 3.0;
double Dy = (1.0 * By + 2.0 * Cy) / 3.0;

// Calculate Point E
double Ex = (1.0 * Ax + 3.0 * Cx) / 4.0;
double Ey = (1.0 * Ay + 3.0 * Cy) / 4.0;

// Calculate Point P
double mu = 8.0 / 11.0;
double Px = (1.0 - mu) * Bx + mu * Ex;
double Py = (1.0 - mu) * By + mu * Ey;
```

C Code - formula function

```c
// Fill the output array (12 elements: Ax, Ay, Bx, By,
    ...)
output_points[0] = Ax; output_points[1] = Ay;
output_points[2] = Bx; output_points[3] = By;
output_points[4] = Cx; output_points[5] = Cy;
output_points[6] = Dx; output_points[7] = Dy;
output_points[8] = Ex; output_points[9] = Ey;
output_points[10] = Px; output_points[11] = Py;
}
```

Python Code through shared output

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

# --- Step 1: Load the shared library ---
lib = ctypes.CDLL('./2.10.70.so')

# --- Step 2: Define the C function signature using NumPy
    -aware pointers ---
calculate_points = lib.calculate_points_from_arrays

# Define the argument types. 'ndpointer' creates a ctypes
    -compatible type for NumPy arrays.
calculate_points.argtypes = [
np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags='
    C_CONTIGUOUS'), # input_A
np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags='
    C_CONTIGUOUS'), # input_B
np.ctypeslib.ndpointer(dtype=np.double, ndim=1, flags='
```

Python Code through shared output

```python
# The function has a 'void' return type in C
calculate_points.restype = None

# --- Step 3: Prepare NumPy arrays and call the C function ---
# Define the vertices of the triangle as NumPy arrays
A = np.array([0.0, 0.0], dtype=np.double)
B = np.array([6.0, 1.0], dtype=np.double)
C = np.array([2.0, 5.0], dtype=np.double)

# Create an empty NumPy array for the C function to fill
# It needs to have space for 6 points (6 * 2 = 12 doubles)
output_data = np.zeros(12, dtype=np.double)

# Call the C function. NumPy arrays are passed directly.
calculate_points(A, B, C, output_data)
```

Python Code through shared output

```
# --- Step 4: Reshape the output and plot ---
# Reshape the flat output array into a 6x2 array (6
    points, 2 coords each)
points_array = output_data.reshape(6, 2)

point_names = ['A', 'B', 'C', 'D', 'E', 'P']
points = {name: coord for name, coord in zip(point_names,
     points_array)}

print(Coordinates calculated by C library and loaded into
     NumPy:)
for name, coords in points.items():
print(f Point {name}: ({coords[0]:.4f}, {coords[1]:.4f}))

# Plotting logic remains the same
fig, ax = plt.subplots(figsize=(10, 8))
ax.set_aspect('equal', adjustable='box')
ax.grid(True, linestyle=':', alpha=0.7)
```

Python Code through shared output

```
for name, coords in points.items():
color = 'red' if name == 'P' else 'black'
size = 12 if name == 'P' else 8
ax.plot(coords[0], coords[1], 'o',
    markersize=size, color=color, label=f'
    Point {name}')
ax.text(coords[0] + 0.15, coords[1] + 0.15,
     name, fontsize=14, fontweight='bold',
    color=color)

ax.set_title('Plot from C Library using
    NumPy and ctypes', fontsize=16)
ax.legend(loc=upper left)

plt.figure()
plt.savefig('numpy_ctypes_plot.png')

plt.show()
```

Python code : Direct

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

def solve_and_plot_with_numpy():

Calculates and plots the triangle
    intersection using NumPy.

# --- Step 1: Define vertices as NumPy
    arrays ---
# We choose A as the origin, consistent
    with the vector solution.
A = np.array([0.0, 0.0])
B = np.array([6.0, 1.0])
C = np.array([2.0, 5.0])

# --- Step 2: Calculate points D and E
    using vector arithmetic ---
# Point D on BC such that BD:DC = 2:1
```

```
# --- Step 3: Calculate the intersection point P
    ---
# From the vector solution, we found the ratio mu
    for the line BE is 8/11.
# Vector formula: P = (1-mu)*B + mu*E
mu = 8.0 / 11.0
P = (1 - mu) * B + mu * E

# --- Step 4: Plot the results ---
points = {'A': A, 'B': B, 'C': C, 'D': D, 'E': E,
    'P': P}

print(Coordinates calculated by NumPy:)
for name, coords in points.items():
print(f Point {name}: ({coords[0]:.4f}, {coords
    [1]:.4f}))

# Setup plot
fig, ax = plt.subplots(figsize=(10, 8))
```

```python
ax.plot([A[0], D[0]], [A[1], D[1]], 'r--', label='
    Line AD')
ax.plot([B[0], E[0]], [B[1], E[1]], 'g--', label='
    Line BE')

for name, coords in points.items():
color = 'red' if name == 'P' else 'black'
size = 12 if name == 'P' else 8
ax.plot(coords[0], coords[1], 'o', markersize=size
    , color=color, label=f'Point {name}')
ax.text(coords[0] + 0.15, coords[1] + 0.15, name,
    fontsize=14, fontweight='bold', color=color)

ax.set_title('Geometric Solution using Python and
    NumPy', fontsize=16)
ax.set_xlabel('X-axis', fontsize=12)
ax.set_ylabel('Y-axis', fontsize=12)
ax.legend(loc=upper left)
```

Plot by python using shared output from c

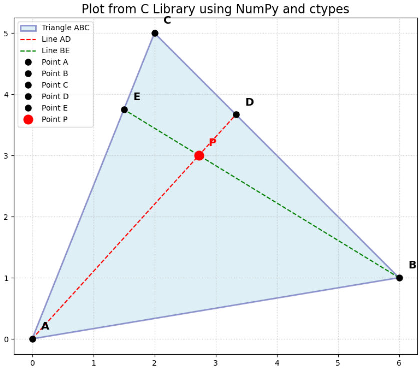

Figure: *

Plot by python only

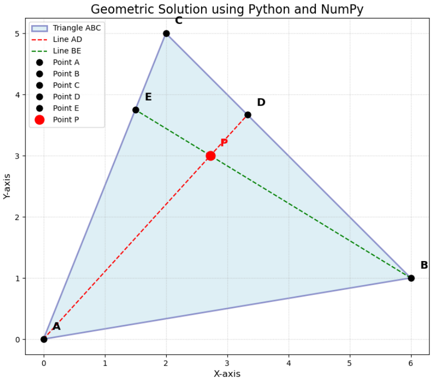

Figure: *