1.4.16

EE25BTECH11001 - Aarush Dilawri

August 29, 2025

Question

Find the coordinates of the points which trisect the line segment joining the points

$$\mathbf{P}(4, 2, -6) \quad \text{and} \quad \mathbf{Q}(10, -16, 6).$$

Let the vectors be

$$\mathbf{P} = \begin{pmatrix} 4 \\ 2 \\ -6 \end{pmatrix}, \tag{1}$$

$$\mathbf{Q} = \begin{pmatrix} 10 \\ -16 \\ 6 \end{pmatrix}. \tag{2}$$

We want to find the points which divide PQ in the ratio $2:1$ and $1:2$.

Section Formula

Section formula: If a point divides the line joining $\mathbf{A}$ and $\mathbf{B}$ in the ratio $k:1$, then

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

First Trisection Point

Using section formula for ratio 2 : 1,

$$\mathbf{S} = \frac{2\mathbf{Q} + \mathbf{P}}{3} \tag{3}$$

$$= \frac{\begin{pmatrix} 20 \\ -32 \\ 12 \end{pmatrix} + \begin{pmatrix} 4 \\ 2 \\ -6 \end{pmatrix}}{3} \tag{4}$$

$$= \frac{\begin{pmatrix} 24 \\ -30 \\ 6 \end{pmatrix}}{3} \tag{5}$$

$$= \begin{pmatrix} 8 \\ -10 \\ 2 \end{pmatrix}. \tag{6}$$

Second Trisection Point

Using section formula for ratio 1 : 2,

$$\mathbf{R} = \frac{\mathbf{Q} + 2\mathbf{P}}{3} \tag{7}$$

$$= \frac{\begin{pmatrix} 10 \\ -16 \\ 6 \end{pmatrix} + \begin{pmatrix} 8 \\ 4 \\ -12 \end{pmatrix}}{3} \tag{8}$$

$$= \frac{\begin{pmatrix} 18 \\ -12 \\ -6 \end{pmatrix}}{3} \tag{9}$$

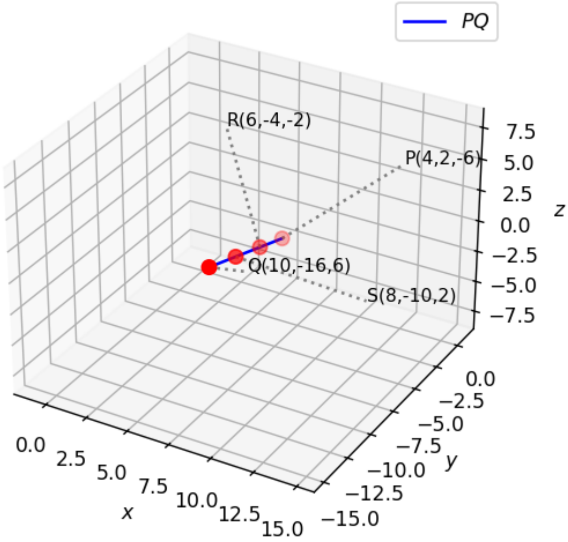
$$= \begin{pmatrix} 6 \\ -4 \\ -2 \end{pmatrix}. \tag{10}$$

Final Answer

Therefore, the points of trisection of PQ are

$$\mathbf{S} = \begin{pmatrix} 8 \\ -10 \\ 2 \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} 6 \\ -4 \\ -2 \end{pmatrix}.$$

Plot



PQ
R(6,-4,-2)
P(4,2,-6)
Q(10,-16,6)
S(8,-10,2)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
z
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
y
0.0
2.5
5.0
7.5
10.0
12.5
15.0
x

C Code (code.c)

```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 (code.py)

```
# Code by Aarush
# August 28, 2025
# Released under GNU GPL
# Section Formula - Trisection of a Line Segment in 3D

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Given points
P = np.array(([4, 2, -6])).reshape(-1,1)
Q = np.array(([10, -16, 6])).reshape(-1,1)

# Points dividing PQ in 1:2 and 2:1 (Trisection points)
R = (2*P + Q)/3 # Point closer to P
S = (P + 2*Q)/3 # Point closer to Q
```

```
# Plotting
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Line PQ
x_vals = [P[0,0], Q[0,0]]
y_vals = [P[1,0], Q[1,0]]
z_vals = [P[2,0], Q[2,0]]
ax.plot(x_vals, y_vals, z_vals, label='$PQ$', color="blue")

# All points
points = np.block([[P,Q,R,S]])
ax.scatter(points[0,:], points[1,:], points[2,:], color="red", s=50)
```

Python Code (code.py)

```
# Labels with offsets
labels = ['P','Q','R','S']
offsets = [(5,5,5), (5,-5,5), (-5,5,5), (5,5,-5)]

for i, txt in enumerate(labels):
    dx, dy, dz = offsets[i]
    ax.text(points[0,i]+dx, points[1,i]+dy, points[2,i]+dz,
            f'{txt}({points[0,i]:.0f},{points[1,i]:.0f},{points[2,i]:.0f})',
            fontsize=9, color="black")
    ax.plot([points[0,i], points[0,i]+dx],
            [points[1,i], points[1,i]+dy],
            [points[2,i], points[2,i]+dz],
            color="gray", linestyle="dotted")
```

Python Code (code.py)

```
# Equal axis scaling
max_range = np.array([points[0,:].max()-points[0,:].min(),
                      points[1,:].max()-points[1,:].min(),
                      points[2,:].max()-points[2,:].min()]).max() / 2.0
mid_x = (points[0,:].max()+points[0,:].min()) * 0.5
mid_y = (points[1,:].max()+points[1,:].min()) * 0.5
mid_z = (points[2,:].max()+points[2,:].min()) * 0.5
ax.set_xlim(mid_x - max_range, mid_x + max_range)
ax.set_ylim(mid_y - max_range, mid_y + max_range)
ax.set_zlim(mid_z - max_range, mid_z + max_range)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_zlabel('$z$')
ax.legend()
ax.grid(True)
plt.show()
```

Python Code (nativecode.py)

```
import ctypes
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# --- Load shared library ---
lib = ctypes.CDLL("./code.so")

# Configure ctypes function signature
lib.findM.argtypes = [ctypes.c_float, ctypes.c_float,
                      ctypes.c_float, ctypes.c_float, ctypes.c_float]
lib.findM.restype = ctypes.c_float
```

Python Code (nativecode.py)

```
def point_on_line(A, B, Px):
    """Compute (x,y,z) for given Px using the C function findM"""
    Ax, Ay, Az = A
    Bx, By, Bz = B
    y = lib.findM(Ax, Ay, Bx, By, Px)
    z = lib.findM(Ax, Az, Bx, Bz, Px)
    return np.array([Px, y, z], dtype=float)

# --- Given points ---
P = np.array([4.0, 2.0, -6.0])
Q = np.array([10.0, -16.0, 6.0])

# Trisection (1:2 and 2:1 ratios)
Rx = (2*P[0] + Q[0]) / 3
Sx = (P[0] + 2*Q[0]) / 3

R = point_on_line(P, Q, Rx)
```

Python Code (nativecode.py)

```
# --- Plotting ---
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Line PQ
ax.plot([P[0],Q[0]], [P[1],Q[1]], [P[2],Q[2]], label='$PQ$', color="blue")

# Collect points
points = np.stack([P,Q,R,S], axis=1)
ax.scatter(points[0,:], points[1,:], points[2,:], color="red", s=50)

# Labels with offsets + dotted connectors
labels = ['P','Q','R','S']
offsets = [(5,5,5), (5,-5,5), (-5,5,5), (5,5,-5)]
```

Python Code (nativecode.py)

```python
for i, txt in enumerate(labels):
    dx, dy, dz = offsets[i]
    ax.text(points[0,i]+dx, points[1,i]+dy, points[2,i]+dz,
            f'{txt}({points[0,i]:.0f},{points[1,i]:.0f},{points[2,i]:.0f})',
            fontsize=9, color="black")
    ax.plot([points[0,i], points[0,i]+dx],
            [points[1,i], points[1,i]+dy],
            [points[2,i], points[2,i]+dz],
            color="gray", linestyle="dotted")

# Equal axis scaling
max_range = np.array([points[0,:].max()-points[0,:].min(),
                      points[1,:].max()-points[1,:].min(),
                      points[2,:].max()-points[2,:].min()]).max() / 2.0
```

Python Code (nativecode.py)

```
mid_x = (points[0,:].max()+points[0,:].min()) * 0.5
mid_y = (points[1,:].max()+points[1,:].min()) * 0.5
mid_z = (points[2,:].max()+points[2,:].min()) * 0.5

ax.set_xlim(mid_x - max_range, mid_x + max_range)
ax.set_ylim(mid_y - max_range, mid_y + max_range)
ax.set_zlim(mid_z - max_range, mid_z + max_range)

ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_zlabel('$z$')
ax.legend()
ax.grid(True)

plt.show()
```