1.11.2

Yoshita J - EE25BTECH11065

August 28,2025

Question

Unit vector along PQ, where coordinates of P and Q respectively are $\begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 4 \\ 4 \\ -7 \end{pmatrix}$ is.

Table

Let the coordinates of the points be $\mathbf{P}(2, 1, -1)$ and $\mathbf{Q}(4, 4, -7)$.

Point	Name
$\begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}$	P
$\begin{pmatrix} 4 \\ 4 \\ -7 \end{pmatrix}$	Q

Table: Vectors

Theoretical Solution

To find the vector **PQ**, we subtract the matrix for P from the matrix for Q:

$$\mathbf{PQ} = \mathbf{Q} - \mathbf{P} \tag{1}$$

$$= \begin{pmatrix} 4 \\ 4 \\ -7 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \\ -2 \end{pmatrix} \tag{2}$$

$$= \begin{pmatrix} 4-2 \\ 4-1 \\ -7-(-1) \end{pmatrix} \tag{3}$$

$$= \begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix} \tag{4}$$

This resulting vector can also be written in the standard basis notation shown in the image:

$$\mathbf{PQ} = 2\mathbf{i} + 3\mathbf{j} - 6\mathbf{k}$$

Theoretical Solution

If we represent the vector **PQ** as a column vector **a**:

$$\mathbf{a} = \begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix}$$

The norm is the square root of the dot product of the vector with itself, which can be expressed as the matrix product of its transpose $\mathbf{a}^T$ and **a**.

$$\|\mathbf{a}\| = \sqrt{\mathbf{a}^T\mathbf{a}} \tag{5}$$

$$= \sqrt{\begin{pmatrix} 2 & 3 & -6 \end{pmatrix}\begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix}} \tag{6}$$

$$= \sqrt{49} \tag{7}$$

$$= 7 \tag{8}$$

Theoretical Solution

The unit vector in the direction of **PQ**, denoted as $\hat{\mathbf{u}}$, is found by dividing the vector by its magnitude.

$$\hat{\mathbf{u}} = \frac{\mathbf{PQ}}{\|\mathbf{PQ}\|} \tag{9}$$

$$= \frac{1}{7}(2\mathbf{i} + 3\mathbf{j} - 6\mathbf{k}) \tag{10}$$

$$= \frac{2}{7}\mathbf{i} + \frac{3}{7}\mathbf{j} - \frac{6}{7}\mathbf{k} \tag{11}$$

Thus, the unit vector along PQ is $\begin{pmatrix} 2/7 \\ 3/7 \\ -6/7 \end{pmatrix}$ or $\frac{1}{7}(\mathbf{2i} + \mathbf{3j} - \mathbf{6k})$.

C Code

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

typedef struct {
    float x, y, z;
} Vector3D;

Vector3D find_unit_vector(Vector3D P, Vector3D Q) {

    Vector3D PQ;
    PQ.x = Q.x - P.x;
    PQ.y = Q.y - P.y;
    PQ.z = Q.z - P.z;
```

C Code

```
    float magnitude = sqrtf(PQ.x * PQ.x + PQ.y * PQ.y + PQ.z * PQ
        .z);

   Vector3D unit_vector = {0.0f, 0.0f, 0.0f};


    if (magnitude > 0) {
        unit_vector.x = PQ.x / magnitude;
        unit_vector.y = PQ.y / magnitude;
        unit_vector.z = PQ.z / magnitude;
    }

    return unit_vector;
}
```

Python Code

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

class Vector3D(ctypes.Structure):
    _fields_ = [(x, ctypes.c_float),
                (y, ctypes.c_float),
                (z, ctypes.c_float)]


C_SOURCE_FILE = 'c_unit_vector_code.c'
SHARED_LIBRARY = './unit_vector.so'
```

Python Code

```
if not os.path.exists(SHARED_LIBRARY):
    print(fShared library '{SHARED_LIBRARY}' not found.)

    if os.path.exists(C_SOURCE_FILE):
        print(fAttempting to compile '{C_SOURCE_FILE}'...)

        compile_command = fgcc -shared -o {SHARED_LIBRARY} -fPIC
            {C_SOURCE_FILE}
        print(fRunning: {compile_command})
        exit_code = os.system(compile_command)
        if exit_code != 0:
            print(\n--- COMPILATION FAILED ---)
            print(Please ensure you have a C compiler (like gcc)
                installed.)
            print(You may need to compile the C code from the
                Canvas manually.)
            exit(1)
        print(Compilation successful.)
```

Python Code

```
 else:
        print(f\nError: C source file '{C_SOURCE_FILE}' not found
            .)
        print(Please save the C code from the Canvas as '
            c_unit_vector_code.c' in the same directory.)
        exit(1)



try:if not os.path.exists(SHARED_LIBRARY):
    print(fShared library '{SHARED_LIBRARY}' not found.)
```

Python Code

```
if os.path.exists(C_SOURCE_FILE):
        print(fAttempting to compile '{C_SOURCE_FILE}'...)

        compile_command = fgcc -shared -o {SHARED_LIBRARY} -fPIC
            {C_SOURCE_FILE}
        print(fRunning: {compile_command})
        exit_code = os.system(compile_command)
        if exit_code != 0:
            print(\n--- COMPILATION FAILED ---)
            print(Please ensure you have a C compiler (like gcc)
                installed.)
            print(You may need to compile the C code from the
                Canvas manually.)
            exit(1)
        print(Compilation successful.)
```

Python Code

```
    c_lib = ctypes.CDLL(SHARED_LIBRARY)
except OSError as e:
    print(fError loading shared library: {e})
    exit(1)

c_lib.find_unit_vector.argtypes = [Vector3D, Vector3D]

c_lib.find_unit_vector.restype = Vector3D

P_coords = Vector3D(x=2.0, y=1.0, z=-1.0)
Q_coords = Vector3D(x=4.0, y=4.0, z=-7.0)

unit_vector_result = c_lib.find_unit_vector(P_coords, Q_coords)

print(--- Results from C Function ---)
```

Python Code

```
   print(fUnit Vector x: {unit_vector_result.x:.6f})
print(fUnit Vector y: {unit_vector_result.y:.6f})
print(fUnit Vector z: {unit_vector_result.z:.6f})
print(------------------------------\n)

print(Generating 3D plot...)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

p_np = np.array([P_coords.x, P_coords.y, P_coords.z])
q_np = np.array([Q_coords.x, Q_coords.y, Q_coords.z])
```

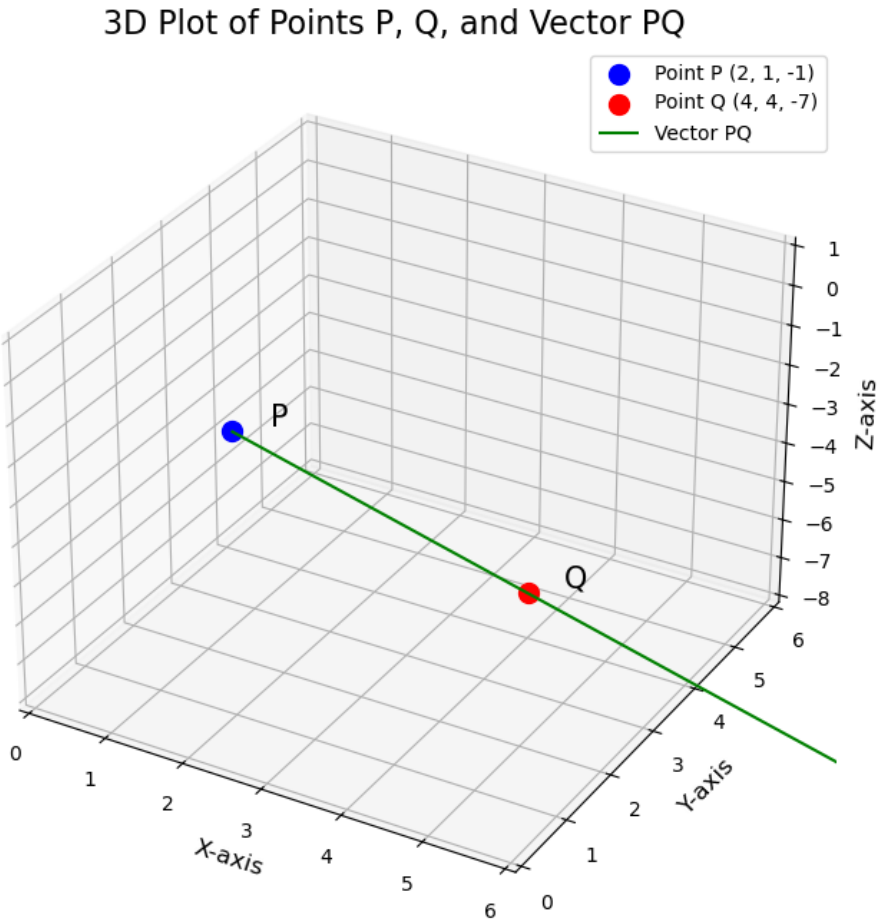
Python Code

```
  ax.scatter(*p_np, color='blue', s=100, label='Point P (2, 1,
      -1)')
ax.scatter(*q_np, color='red', s=100, label='Point Q (4, 4, -7)')
ax.text(*(p_np + 0.3), 'P', size=15, color='k')
ax.text(*(q_np + 0.3), 'Q', size=15, color='k')

vector_pq = q_np - p_np
ax.quiver(*p_np, *vector_pq, color='gray', linestyle='dashed',
    arrow_length_ratio=0.1, label='Vector PQ')

unit_vec_np = np.array([unit_vector_result.x, unit_vector_result.
    y, unit_vector_result.z])
ax.quiver(0, 0, 0, *unit_vec_np, color='green', length=1.0,
    arrow_length_ratio=0.2, label='Unit Vector (from C)')
```

Python Code

```
 ax.set_xlabel('X-axis', fontsize=12)
ax.set_ylabel('Y-axis', fontsize=12)
ax.set_zlabel('Z-axis', fontsize=12)
ax.set_title('Visualization of Vector and its Unit Vector (
    Calculated in C)', fontsize=14)
ax.legend()
ax.grid(True)
ax.set_xlim([0, 5])
ax.set_ylim([0, 5])
ax.set_zlim([-8, 2])
ax.view_init(elev=20., azim=-50) # Set a nice viewing angle

plt.tight_layout()
plt.show()
```

Plot


3D Plot of Points P, Q, and Vector PQ
Point P (2, 1, -1)
Point Q (4, 4, -7)
Vector PQ
P
Q
X-axis
Y-axis
Z-axis