# 1.11.18

Kotha Pratheek Reddy
AI24BTECH11019

November 6, 2024

1. Problem
2. Solution
3. Plot
   - C-Code
   - Python Code

# Problem Statement

Find the direction cosines of the line joining points **P** $\left(4, 3, -5\right)$ and **Q** $\left(-2, 1, 8\right)$.

# Solution

| Point | Coordinate |
|---|---|
| **P** | $(4, 3, -5)$ |
| **Q** | $(-2, 1, 8)$ |

Table: Coordinates

# Solution

Let the unit vector in the direction of the vector $\mathbf{PQ}$ be $\hat{a}$.Then

$$\hat{a} = \frac{\mathbf{Q} - \mathbf{P}}{||\mathbf{Q} - \mathbf{P}||} \tag{1}$$

$$\mathbf{P} = \begin{pmatrix} 4 \\ 3 \\ -5 \end{pmatrix} \tag{2}$$

$$\mathbf{Q} = \begin{pmatrix} -2 \\ 1 \\ 8 \end{pmatrix} \tag{3}$$

$$\mathbf{Q} - \mathbf{P} = \begin{pmatrix} -6 \\ -2 \\ 13 \end{pmatrix} \tag{4}$$

$$\begin{aligned} ||\mathbf{Q} - \mathbf{P}|| &= \sqrt{(-6)^2 + (-2)^2 + 13^2} \\ &= \sqrt{209} \end{aligned} \tag{5}$$

# Solution

From the above equations,

$$\hat{a} = \begin{pmatrix} \frac{-6}{\sqrt{209}} \\ \frac{-2}{\sqrt{209}} \\ \frac{13}{\sqrt{209}} \end{pmatrix} \tag{6}$$

The direction cosines of the the line joining **A** and **B** are the components of $\hat{a}$ i.e. $\frac{-6}{\sqrt{209}}$ , $\frac{-2}{\sqrt{209}}$, $\frac{13}{\sqrt{209}}$

# C-Code

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

typedef struct {
    double x;
    double y;
    double z;
} Vector;
```

```
            // Function to calculate the direction cosines of
            ↪  the line joining two points
Vector* calculate_cosines(Vector* P, Vector* Q) {
    Vector* result = (Vector*)malloc(sizeof(Vector));

    // Calculate the direction vector
    double dx = Q->x - P->x;
    double dy = Q->y - P->y;
    double dz = Q->z - P->z;
```

```
        // Calculate the magnitude of the direction vector
    double magnitude = sqrt(dx * dx + dy * dy + dz * dz);



    // Calculate direction cosines
    result->x = dx / magnitude;  // cos(alpha)
    result->y = dy / magnitude;  // cos(beta)
    result->z = dz / magnitude;  // cos(gamma)



    return result;
}

// Function to free the allocated vector
void free_vector(Vector* vec) {
    free(vec);
}
```

# Python Code

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

# Load the shared object file
lib = ctypes.CDLL('./code.so')

# Define the Point struct in Python
class Point(ctypes.Structure):
    _fields_ = [("x", ctypes.c_double),
                ("y", ctypes.c_double),
                ("z", ctypes.c_double)]
```

```
        # Define the Vector struct in Python
class Vector(ctypes.Structure):
    _fields_ = [("x", ctypes.c_double),
                ("y", ctypes.c_double),
                ("z", ctypes.c_double)]
# Specify the return type and argument types for the
↪  calculate_cosines function
lib.calculate_cosines.restype = ctypes.POINTER(Vector)
lib.calculate_cosines.argtypes = [ctypes.POINTER(Point),
↪  ctypes.POINTER(Point)]
```

```
  # Function to draw angle between two vectors
def draw_angle_between_vectors(v1, v2, ax, text_offset=0):
    # Normalize the vectors
    v1 = v1 / np.linalg.norm(v1)
    v2 = v2 / np.linalg.norm(v2)

    # Compute the normal vector to the plane defined by v1 and
    ↪  v2
    normal = np.cross(v1, v2)
    normal = normal / np.linalg.norm(normal)  # Normalize the
    ↪  normal vector

    # Calculate the angle between the vectors
    angle_rad = np.arccos(np.dot(v1, v2))
    angle_deg = np.degrees(angle_rad)  # Convert to degrees

    # Parametrize the arc
    theta = np.linspace(0, angle_rad, 100)
    arc_points = np.array([np.cos(t) * v1 + np.sin(t) *
    ↪  np.cross(normal, v1) for t in theta]) / 2
```

```
    # Plot the arc
    ax.plot(arc_points[:, 0], arc_points[:, 1], arc_points[:,
    ↪  2], 'g', label='Angle Arc')

    # Label the angle in the middle of the arc
    mid_arc_point = arc_points[len(arc_points) // 2] + (v1 + v2)
    ↪  / 5
    ax.text(mid_arc_point[0] + text_offset, mid_arc_point[1],
    ↪  mid_arc_point[2], f'{angle_deg:.0f}°', color='purple',
    ↪  fontsize=9)

# Create Point structs for P and Q
P = Point(4, 3, -5)
Q = Point(-2, 1, 8)
```

```
        # Call the C function to get direction cosines
vector_ptr = lib.calculate_cosines(ctypes.byref(P),
↪  ctypes.byref(Q))
origin = np.array([0, 0, 0])
vector = np.array([vector_ptr.contents.x, vector_ptr.contents.y,
↪  vector_ptr.contents.z]) * 2  # Scale for clarity

print("Direction cosines:")
print("Cos alpha:", vector_ptr.contents.x)
print("Cos beta:", vector_ptr.contents.y)
print("Cos gamma:", vector_ptr.contents.z)

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

```
        # Plot the vector and axes
ax.quiver(*origin, *vector, length=1, color='r',
↪  label='Direction Vector')
ax.quiver(*origin, 0, 0, 2, length=1, color='k', label='Y-axis')
ax.quiver(*origin, 0, 2, 0, length=1, color='k', label='Z-axis')
ax.quiver(*origin, 2, 0, 0, length=1, color='k', label='X-axis')

# Draw angle arcs
draw_angle_between_vectors(np.array([1, 0, 0]), vector, ax)  #
↪  Angle with X-axis
draw_angle_between_vectors(np.array([0, 1, 0]), vector, ax)  #
↪  Angle with Y-axis
draw_angle_between_vectors(np.array([0, 0, 1]), vector, ax)  #
↪  Angle with Z-axis
```

```
        # Set limits and labels
ax.set_xlim([-2, 2])
ax.set_ylim([-2, 2])
ax.set_zlim([-2, 2])
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

# Add axis labels
ax.text(1.2, 0, 0, "X", color='k')
ax.text(0, 1.2, 0, "Y", color='k')
ax.text(0, 0, 1.2, "Z", color='k')

plt.grid(True)
plt.legend()
plt.show()

# Free the C pointer
lib.free_vector(vector_ptr)
```

# Plot

The codes in

`https://github.com/Pratheek39/EE1030/tree/c703931a5fffd529b14ab319f`

plot the following figure

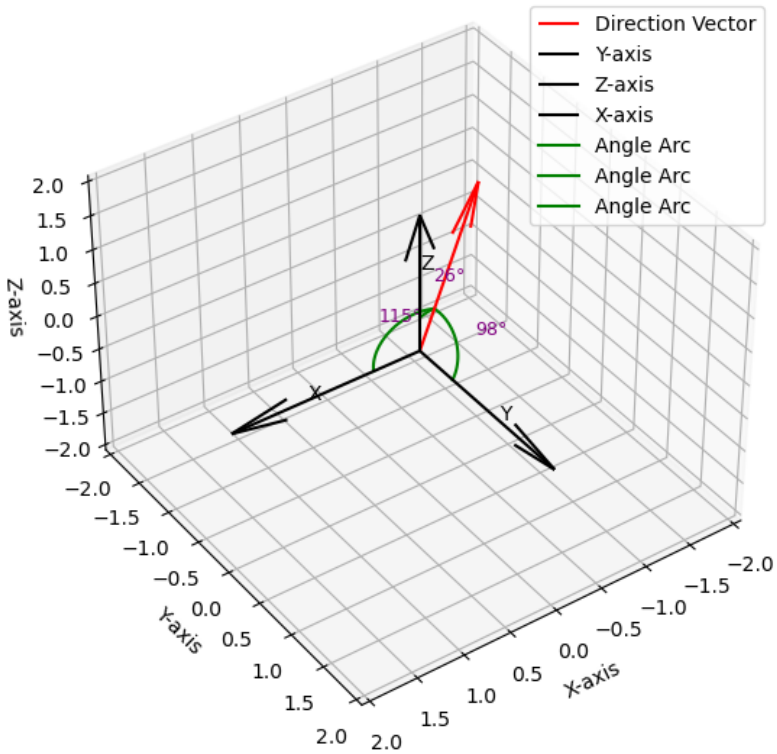


Figure: Line joining **P** and **Q**