2.2.6

Varun-ai25btech11016

September 1, 2025

Question

Find the angle between the vectors

$$\mathbf{a} = 2\hat{i} - \hat{j} + \hat{k}, \quad \mathbf{b} = 3\hat{i} + 4\hat{j} - \hat{k}$$

Theoretical Solution

$$\mathbf{a} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \tag{1}$$

$$\mathbf{b} = \begin{pmatrix} 3 \\ 4 \\ -1 \end{pmatrix} \tag{2}$$

From the formula,

$$\cos\theta = \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|} \tag{3}$$

Theoretical Solution

Substituting,

$$\cos\theta = \frac{1}{\sqrt{6}\,\sqrt{26}} \tag{4}$$

$$= \frac{1}{\sqrt{156}}$$

Therefore,

$$\theta = \cos^{-1}\left(\frac{1}{\sqrt{156}}\right) \tag{5}$$

The angle between the given two vectors is $\cos^{-1}\left(\frac{1}{\sqrt{156}}\right)$

C code

```c

#include <math.h>

// Function to compute angle (in radians) between two 3D vectors
double angle_between(double ax, double ay, double az,
                     double bx, double by, double bz) {
    double dot = ax*bx + ay*by + az*bz;

    double mag_a = sqrt(ax*ax + ay*ay + az*az);
    double mag_b = sqrt(bx*bx + by*by + bz*bz);

    if (mag_a == 0.0 || mag_b == 0.0) {
        return -1.0; // invalid
    }
```

C code

```
    double cos_theta = dot / (mag_a * mag_b);
    if (cos_theta > 1.0) cos_theta = 1.0;
    if (cos_theta < -1.0) cos_theta = -1.0;

    return acos(cos_theta);
}
```

C plus Python code

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

# Load .so
lib = ctypes.CDLL(2.2.6fuction.so)
lib.angle_between.argtypes = [ctypes.c_double, ctypes.c_double,
    ctypes.c_double,
                              ctypes.c_double, ctypes.c_double,
                                  ctypes.c_double]
lib.angle_between.restype = ctypes.c_double

# Original 3D vectors
a = np.array([2, -1, 1])
b = np.array([3, 4, -1])
```

C plus Python code

```
# Call C function
theta = lib.angle_between(*a, *b)
print(Angle (radians):, theta)
print(Angle (degrees):, np.degrees(theta))

# ---- Project to 2D plane ----
# Orthonormal basis from vector a
u = a / np.linalg.norm(a) # first basis vector
v = b - np.dot(b, u) * u # make b orthogonal to u
v = v / np.linalg.norm(v) # second basis vector

# Coordinates of a and b in this 2D plane
a2d = np.array([np.dot(a, u), np.dot(a, v)])
b2d = np.array([np.dot(b, u), np.dot(b, v)])
```

C plus Python code

```
# ---- Plot in 2D ----
plt.figure(figsize=(6,6))
plt.axhline(0, color='gray', lw=0.5)
plt.axvline(0, color='gray', lw=0.5)

plt.quiver(0, 0, a2d[0], a2d[1], angles='xy', scale_units='xy', scale=1, color=r, label=a = (2,-1,1))
plt.quiver(0, 0, b2d[0], b2d[1], angles='xy', scale_units='xy', scale=1, color=b, label=b = (3,4,-1))

plt.xlim(-5, 5)
plt.ylim(-5, 5)
plt.gca().set_aspect(equal)

plt.legend()
plt.savefig(/sdcard/Matrix/ee1030-2025/ai25btech11016/Matgeo/2.2.6/figs/2.2.6.png)
plt.show()
```

Python plot code

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

# Vectors in 3D
a = np.array([2, -1, 1])
b = np.array([3, 4, -1])

# Normalize a  treat as new x-axis
u = a / np.linalg.norm(a)

# Remove component of b along u  gives orthogonal direction in
    plane
b_proj = b - np.dot(b, u) * u
v = b_proj / np.linalg.norm(b_proj) # normalize  new y-axis
```

Python plot code

```
# 2D coordinates in this new basis
a_2d = np.array([np.dot(a, u), np.dot(a, v)])
b_2d = np.array([np.dot(b, u), np.dot(b, v)])

# Plot in 2D
plt.figure(figsize=(6,6))
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)

plt.quiver(0, 0, a_2d[0], a_2d[1], angles='xy', scale_units='xy',
     scale=1, color='r', label=a = (2,-1,1))
plt.quiver(0, 0, b_2d[0], b_2d[1], angles='xy', scale_units='xy',
     scale=1, color='b', label=b = (3,4,-1))
```

Python plot code

```
plt.legend()
plt.grid(True, linestyle=--, alpha=0.6)
plt.gca().set_aspect('equal', adjustable='box')
plt.savefig(/sdcard/Matrix/ee1030-2025/ai25btech11016/Matgeo
    /2.2.6/figs/2.2.6.png)
plt.show()
```

Plot