2.10.41

BEERAM MADHURI - EE25BTECH11012

September 2025

Question

Let the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{d}$ be such that $(\mathbf{a} \times \mathbf{b}) \times (\mathbf{c} \times \mathbf{d}) = \mathbf{0}$. Let A and B be planes determined by the pairs of vectors $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}, \mathbf{d}$ respectively. Then the angle between A and B is

a) 0 b) $\frac{\pi}{4}$ c) $\frac{\pi}{3}$ d) $\frac{\pi}{2}$

given data

$(\mathbf{a} \times \mathbf{b}) \times (\mathbf{c} \times \mathbf{d}) = \mathbf{0}$

A : span of a, b

B : span of c, d

finding Angle between Planes A and B

Cross product of 2 vectors can be written using a skew-symmetric matrix:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} \quad \text{where} \quad [\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{1}$$

Thus,

$$(\mathbf{a} \times \mathbf{b}) \times (\mathbf{c} \times \mathbf{d}) = [\mathbf{a} \times \mathbf{b}]_\times (\mathbf{c} \times \mathbf{d}) = 0 \tag{2}$$

$$[\mathbf{a} \times \mathbf{b}]_\times (\mathbf{c} \times \mathbf{d}) = 0 \iff (\mathbf{c} \times \mathbf{d}) \parallel (\mathbf{a} \times \mathbf{b}) \tag{3}$$

$$(\mathbf{a} \times \mathbf{b}) = \lambda (\mathbf{c} \times \mathbf{d}) \tag{4}$$

normals to planes A and B:

$$n_A = \mathbf{a} \times \mathbf{b} \tag{5}$$

$$n_B = \mathbf{c} \times \mathbf{d} \tag{6}$$

$$n_A = \lambda n_B \tag{7}$$

Angle between Planes A and B = Angle between Normals n_A and n_B

Angle between planes A and B:

$$\theta = \cos^{-1}\left(\frac{\mathbf{n}_A^\top \mathbf{n}_B}{\|\mathbf{n}_A\| \, \|\mathbf{n}_B\|}\right) \tag{8}$$

$$= \cos^{-1}\left(\frac{\lambda \|\mathbf{n}_B\|^2}{|\lambda| \|\mathbf{n}_B\|^2}\right) \tag{9}$$

$$= \cos^{-1}(\pm 1) \tag{10}$$

$$\tag{11}$$

Considering acute angle,

$$\theta = 0 \tag{12}$$

$$\therefore n_A \parallel n_B \tag{13}$$

$$\therefore planeA \parallel planeB \tag{14}$$

Hence, Angle between the planes is 0
option (a).

Python Code

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

#Example vectors that satisfy (ab)(cd)=0

a = np.array([1, 0, 0])
b = np.array([0, 1, 0])
c = np.array([2, 0, 0])
d = np.array([0, 2, 0])
```

Python Code

```
#Normals of planes A and B

n1 = np.cross(a, b)
n2 = np.cross(c, d)

print("n1:", n1, " n2:", n2) # normals
print("Cross of normals:", np.cross(n1, n2)) # should be zero
```

Python Code

```
#Mesh grid for plotting
xx, yy = np.meshgrid(np.linspace(-2, 2, 20), np.linspace(-2, 2,
    20))
def plane_z(normal, X, Y):
# n(x,y,z) = 0 => z = (-n_x X - n_y Y)/n_z if n_z != 0
return (-normal[0]*X - normal[1]*Y)/normal[2] if normal[2] != 0
    else np.zeros_like(X)
z1 = plane_z(n1, xx, yy)
z2 = plane_z(n2, xx, yy)
```

Python Code

```
fig = plt.figure(figsize=(9,7))
ax = fig.add_subplot(111, projection='3d')

#Plot planes

surf1 = ax.plot_surface(xx, yy, z1, color='cyan', alpha=0.5)
surf2 = ax.plot_surface(xx, yy, z2 + 0.1, color='magenta', alpha
    =0.5) # small offset for visibility
```

Python Code

```python
#Mark plane names
ax.text(0, 0, 0.05, "Plane A", color='blue', fontsize=12, ha='
    center')
ax.text(0, 0, 0.15, "Plane B", color='purple', fontsize=12, ha='
    center')

#Plot and label normals
origin = np.array([0, 0, 0])
ax.quiver(*origin, *n1, color='blue', length=1.0,
    arrow_length_ratio=0.1)
ax.quiver(*origin, *n2, color='red', length=1.0,
    arrow_length_ratio=0.1)
```

Python Code

```
#Add text at the arrow tips for normals
ax.text((n11.1), "n", color='blue', fontsize=12)
ax.text((n21.1), "n", color='red', fontsize=12)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Planes A & B with Normals Marked')
#Make the axes equal for a better view
ax.set_box_aspect([1,1,0.5])
plt.show()
```

C Code

```c
#include <stdio.h>
#include <math.h>
// cross product of two 3D vectors
void cross(double u[3], double v[3], double result[3]) {
    result[0] = u[1]*v[2] - u[2]*v[1];
    result[1] = u[2]*v[0] - u[0]*v[2];
    result[2] = u[0]*v[1] - u[1]*v[0];}
// dot product
double dot(double u[3], double v[3]) {
    return u[0]*v[0] + u[1]*v[1] + u[2]*v[2];
}
```

C Code

```c
  // magnitude of a 3D vector
double magnitude(double v[3]) {
    return sqrt(dot(v, v));
}
int main(void) {
    // Example vectors satisfying (ab)(cd)=0
    double a[3] = {1, 0, 0};
    double b[3] = {0, 1, 0};
    double c[3] = {2, 0, 0};
    double d[3] = {0, 2, 0};
```

C Code

```c
     double n1[3], n2[3];
    cross(a, b, n1);
    cross(c, d, n2);
    double angle = acos(dot(n1, n2) / (magnitude(n1) * magnitude(
        n2)));
    printf("Angle between the two planes (radians): %f\n", angle)
        ;
    printf("Angle between the two planes (degrees): %f\n", angle
        * 180.0 / M_PI);
    return 0;
}
```

Python and C Code

```python
import ctypes
import platform
import math
# --- 1. Load the shared library ---
if platform.system() == "Windows":
    lib_path = "./libvector.dll"
else:
    lib_path = "./libvector.so"
try:
    lib = ctypes.CDLL(lib_path)
except OSError as e:
    print(f"Error loading library: {e}")
    print("Have you compiled vector_ops.c?")
    exit()
```

Python and C Code

```
# --- 2. Define types and function signatures ---
# Define a pointer to a C double
c_double_p = ctypes.POINTER(ctypes.c_double)
# Define a Python type for a C array of 3 doubles
Vector3D = ctypes.c_double * 3

# Signature for cross() -> void cross(double*, double*, double*)
lib.cross.argtypes = [c_double_p, c_double_p, c_double_p]
lib.cross.restype = None
```

Python and C Code

```
# Signature for dot() -> double dot(double*, double*)
lib.dot.argtypes = [c_double_p, c_double_p]
lib.dot.restype = ctypes.c_double

# Signature for magnitude() -> double magnitude(double*)
lib.magnitude.argtypes = [c_double_p]
lib.magnitude.restype = ctypes.c_double
```

Python and C Code

```python
# --- 3. Re-create the logic from the C main() function in Python
    ---
# Initialize the same vectors
a = Vector3D(1, 0, 0)
b = Vector3D(0, 1, 0)
c = Vector3D(2, 0, 0)
d = Vector3D(0, 2, 0)

# Create empty vectors to hold the results (output buffers)
n1 = Vector3D()
n2 = Vector3D()
```

```python
# Call the C 'cross' function twice
lib.cross(a, b, n1)
lib.cross(c, d, n2)

# Call the C 'dot' and 'magnitude' functions
dot_product = lib.dot(n1, n2)
mag_n1 = lib.magnitude(n1)
mag_n2 = lib.magnitude(n2)
```

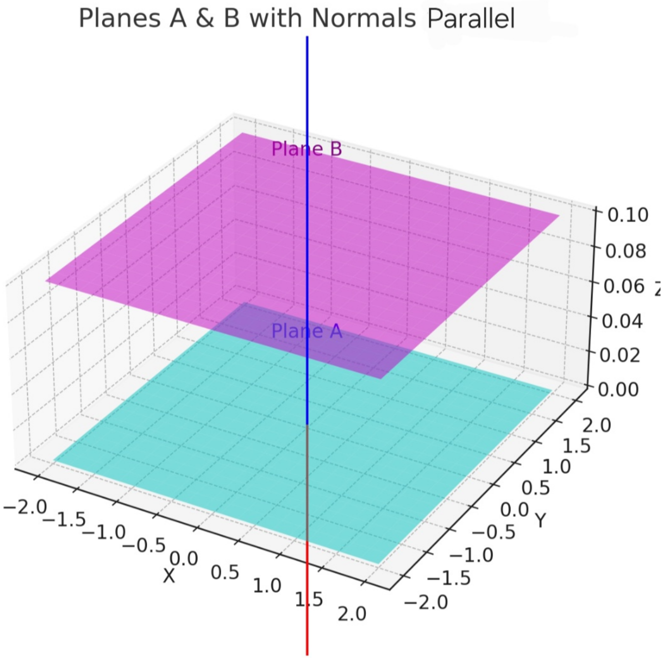
Python and C Code

```python
# Perform the final calculation in Python
if mag_n1 == 0 or mag_n2 == 0:
    angle_radians = 0.0
else:
    # Clamp value to avoid math domain errors with acos
    cos_theta = max(-1.0, min(1.0, dot_product / (mag_n1 * mag_n2
        )))
    angle_radians = math.acos(cos_theta)

angle_degrees = angle_radians * 180 / math.pi
```

Python and C Code

```python
# --- 4. Print the final results ---
print("--- Logic from main() recreated in Python, using C
    functions for math ---")
print(f"Angle between the two planes (radians): {angle_radians}")
print(f"Angle between the two planes (degrees): {angle_degrees}")
```


Planes A & B with Normals Parallel
Plane B
Plane A
0.10
0.08
0.06
0.04
0.02
0.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
X
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Y