2.10.57

J.NAVYASRI- EE25BTECH11028

september 2025

Question

If $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{d}$ are unit vectors such that

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = 1 \quad \text{and} \quad \mathbf{a} \cdot \mathbf{c} = \frac{1}{2},$$

then

(a) $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are non-coplanar

(b) $\mathbf{b}, \mathbf{c}, \mathbf{d}$ are non-coplanar

(c) $\mathbf{b}, \mathbf{d}$ are non-parallel

(d) $\mathbf{a}, \mathbf{d}$ are parallel and $\mathbf{b}, \mathbf{c}$ are parallel

solution:

We are given that

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = 1, \qquad \mathbf{a} \cdot \mathbf{c} = \frac{1}{2}. \tag{1}$$

Step 1: Vector Identity

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}). \tag{2}$$

Step 2: Substitution Since $\mathbf{a} \cdot \mathbf{c} = \frac{1}{2}$,

$$1 = \frac{1}{2}(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}). \tag{3}$$

Solution:

Step 3: Assume $\mathbf{b} \parallel \mathbf{d}$ If $\mathbf{b} \cdot \mathbf{d} = 1$, then

$$1 = \tfrac{1}{2}(1) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}). \tag{4}$$

$$1 = \tfrac{1}{2} - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}). \tag{5}$$

$$(\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c}) = -\tfrac{1}{2}. \tag{6}$$

Step 4: Conclusion Thus, the condition is satisfied when

$$\mathbf{a} \parallel \mathbf{d}, \qquad \mathbf{b} \parallel \mathbf{c}. \tag{7}$$

Option (D) is correct.

Python Code

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

# Define base vectors for parallelism
a = np.array([1.0, 0.0, 0.0]) # a along x-axis
b = np.array([0.5, np.sqrt(3)/2, 0.0]) # b at 60 in xy-plane
c = b.copy() # c || b
d = a.copy() # d || a

# Small offsets so the arrows don't overlap visually
offset_c = np.array([0.0, 0.0, 0.05]) # shift c slightly up in z
offset_d = np.array([0.0, 0.0, -0.05]) # shift d slightly down in
    z
```

Python Code

```
# Compute values
val = np.dot(np.cross(a, b), np.cross(c, d))
dot_ac = np.dot(a, c)

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

origin = np.zeros(3)

# Assign distinct colors for clarity
vectors = {
    'a': (origin, a, 'r'),
    'd': (offset_d, d, 'b'), # Blue
    'b': (origin, b, 'g'),
    'c': (offset_c, c, 'm') # Magenta
}
```

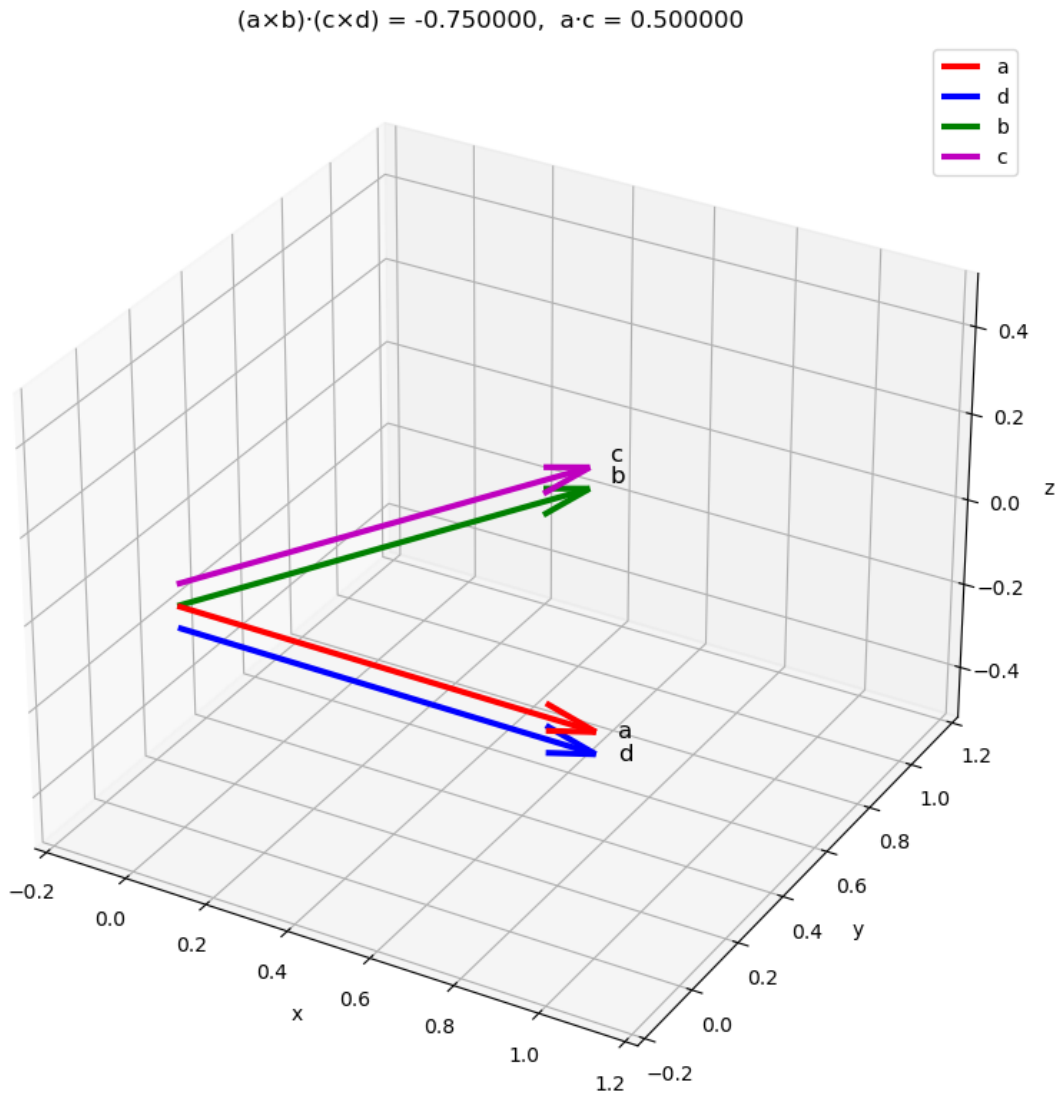
Python Code

```
for name, (start, vec, col) in vectors.items():
    ax.quiver(start[0], start[1], start[2],
              vec[0], vec[1], vec[2],
              length=1.0, linewidth=3, arrow_length_ratio=0.12,
              color=col, label=name)
    ax.text(start[0]+vec[0]*1.05, start[1]+vec[1]*1.05, start[2]+
        vec[2]*1.05,
            name, fontsize=12)

# Add legend in top-right corner
ax.legend(loc='upper right')
```

Python Code

```
ax.set_xlim(-0.2, 1.2)
ax.set_ylim(-0.2, 1.2)
ax.set_zlim(-0.5, 0.5)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title(f"(ab)(cd) = {val:.6f}, ac = {dot_ac:.6f}")

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

Plot-Using by Python



C Code

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

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

double dot(Vector a, Vector b) {
    return a.x*b.x + a.y*b.y + a.z*b.z;
}

Vector cross(Vector a, Vector b) {
    Vector r;
    r.x = a.y*b.z - a.z*b.y;
    r.y = a.z*b.x - a.x*b.z;
    r.z = a.x*b.y - a.y*b.x;
    return r;
}
```

C Code

```c
int main() {
    // Choose simple unit vectors
    Vector a = {sqrt(3)/2, 0.5, 0}; // unit vector
    Vector d = {sqrt(3)/2, 0.5, 0}; // parallel to a
    Vector b = {0, 1, 0}; // unit vector
    Vector c = {0, 1, 0}; // parallel to b

    // Compute
    Vector axb = cross(a, b);
    Vector cxd = cross(c, d);
    double lhs = dot(axb, cxd);
    double ac = dot(a, c);
```

C Code

```
printf("(a  b)  (c  d) = %.2f\n", lhs);
    printf("a  c = %.2f\n", ac);

    if (fabs(lhs - 1.0) < 1e-6 && fabs(ac - 0.5) < 1e-6) {
        printf("Condition satisfied: a || d and b || c\n");
    }

    return 0;
}
```

Python and C Code

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

# Load the compiled C library
lib = ctypes.CDLL("./libvectors.so") # use "vectors.dll" on
    Windows

# Define argument types
lib.check.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
```

Python and C Code

```
# Define vectors (unit vectors)
a = (ctypes.c_double * 3)(1, 0, 0) # along x
d = (ctypes.c_double * 3)(1, 0, 0) # parallel to a
b = (ctypes.c_double * 3)(0, 1, 0) # along y
c = (ctypes.c_double * 3)(0, 1, 0) # parallel to b

val1 = ctypes.c_double()
val2 = ctypes.c_double()

# Call C function
lib.check(a, b, c, d, ctypes.byref(val1), ctypes.byref(val2))
```

Python and C Code

```
print("(ab)(cd) =", val1.value)
print("ac =", val2.value)

# Plot in Python
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(111, projection='3d')

origin = [0,0,0]

ax.quiver(*origin, *a, color='r', label='a')
ax.quiver(*origin, *b, color='g', label='b')
ax.quiver(*origin, *c, color='b', label='c')
ax.quiver(*origin, *d, color='m', label='d')
```

Python and C Code

```
ax.set_xlim([0,1.5])
ax.set_ylim([0,1.5])
ax.set_zlim([0,1.5])
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title("C code in Python: a || d, b || c")

ax.legend()
plt.savefig("fig5.1.png")
plt.show()
```

Plot-Using by both C and Python