2.10.42

EE25BTECH11013 - Bhargav

September 5, 2025

Question

If $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$ are unit coplanar vectors, evaluate the scalar triple product:
$[2\mathbf{a}-\mathbf{b} \quad 2\mathbf{b}-\mathbf{c} \quad 2\mathbf{c}-\mathbf{a}]$

Solution

Given Condition: Since the vectors are coplanar, their scalar triple product is zero.

$$\mathbf{B} = \begin{pmatrix} 2\mathbf{a} - \mathbf{b} & 2\mathbf{b} - \mathbf{c} & 2\mathbf{c} - \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix} \begin{pmatrix} 2 & 0 & -1 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{pmatrix} \tag{1}$$

Let

$$\mathbf{M} = \begin{pmatrix} 2 & 0 & -1 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{pmatrix} \tag{2}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix} \tag{3}$$

Solution

$$\therefore \det(\mathbf{B}) = \det(\mathbf{A}) \cdot \det(\mathbf{M}) \tag{4}$$

Since $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are coplanar,

$$\begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{bmatrix} = 0. \tag{5}$$

$$\Rightarrow \begin{bmatrix}2\mathbf{a} - \mathbf{b} & 2\mathbf{b} - \mathbf{c} & 2\mathbf{c} - \mathbf{a}\end{bmatrix} = \det(\mathbf{M}) \cdot \begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{bmatrix} = 0. \tag{6}$$

Verification

This can be verified by taking an example of 3 coplanar unit vectors.

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

$$\mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{8}$$

$$\mathbf{c} = \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} \tag{9}$$

$$\mathbf{X} = \begin{bmatrix} 2\mathbf{a} - \mathbf{b} & 2\mathbf{b} - c & 2\mathbf{c} - \mathbf{a} \end{bmatrix} \tag{10}$$

From the code, it is clear that the value of $\mathbf{X}$ is 0

C Code

```
#include <stdio.h>

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


Vector createVector(double x, double y, double z) {
    Vector v = {x, y, z};
    return v;
}

Vector subtract(Vector u, Vector v) {
    return createVector(u.x - v.x, u.y - v.y, u.z - v.z);
}
```

C Code

```
Vector scale(Vector u, double k) {
    return createVector(k*u.x, k*u.y, k*u.z);
}

Vector cross(Vector u, Vector v) {
    return createVector(
        u.y*v.z - u.z*v.y,
        u.z*v.x - u.x*v.z,
        u.x*v.y - u.y*v.x
    );
}

double dot(Vector u, Vector v) {
    return u.x*v.x + u.y*v.y + u.z*v.z;
}
```

C Code

```
double triple(Vector u, Vector v, Vector w) {
    return dot(u, cross(v, w));
}

Vector twominus(Vector a, Vector b) {
    return subtract(scale(a, 2), b);
}


double computeX(Vector a, Vector b, Vector c) {
    Vector v1 = twominus(a, b);
    Vector v2 = twominus(b, c);
    Vector v3 = twominus(c, a);
    return triple(v1, v2, v3);
}
```

C Code

```
__attribute__((visibility("default")))
double computeX_py(double ax, double ay, double az,
                   double bx, double by, double bz,
                   double cx, double cy, double cz) {
    Vector a = createVector(ax, ay, az);
    Vector b = createVector(bx, by, bz);
    Vector c = createVector(cx, cy, cz);
    return computeX(a, b, c);
}
```

Python + C Code

```
import ctypes
lib = ctypes.CDLL("./libscalartp.so")
lib.computeX_py.argtypes = [ctypes.c_double, ctypes.c_double, ctypes.
    c_double,
                            ctypes.c_double, ctypes.c_double, ctypes.
                                c_double,
                            ctypes.c_double, ctypes.c_double, ctypes.
                                c_double]
lib.computeX_py.restype = ctypes.c_double
a = (1.0, 0.0, 0.0)
b = (0.0, 1.0, 0.0)
c = (0.6, 0.8, 0.0)
X = lib.computeX_py(*a, *b, *c)
print("X =", X)
```

Python Code

```
import numpy as np



a = np.array([1, 0, 0])
b = np.array([0,1,0])
c = np.array([0.6, 0.8, 0])

x = np.dot(2*a-b, np.cross(2*b-c,2*c-a))
print("Value of x: ", x)
```