2.7.3

Shivam Sawarkar
AI25BTECH11031

September 19, 2025

Question (2.7.3)

If $\mathbf{a}$ and $\mathbf{b}$ are two vectors such that $\mathbf{a} = \hat{i} - \hat{j} + \hat{k}$, $\mathbf{b} = 2\hat{i} - \hat{j} - 3\hat{k}$, then find the vector $\mathbf{c}$, given that $\mathbf{a} \times \mathbf{c} = \mathbf{b}$, $\mathbf{a} \cdot \mathbf{c} = 4$.

Solution

$$\mathbf{a} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 2 \\ -1 \\ -3 \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \tag{1}$$

$$\mathbf{a} \times \mathbf{c} = \mathbf{b} \tag{2}$$

$$\implies \mathbf{c} \perp \mathbf{b} \tag{3}$$

$$\therefore \mathbf{b}^\top \mathbf{c} = 0 \tag{4}$$

Solution

and $\mathbf{a}^\top \mathbf{c} = 4$ is given

$$\begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix}^\top \mathbf{c} = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} 1 & -1 & 1 \\ 2 & -1 & -3 \end{pmatrix} \mathbf{c} = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \tag{6}$$

$$\begin{pmatrix} 1 & -1 & 1 \\ 2 & -1 & -3 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \tag{7}$$

Let, $c_3 = \lambda$
Then

$$c_1 = -4 + 4\lambda \tag{8}$$

$$c_2 = -8 + 5\lambda \tag{9}$$

Solution

$$\mathbf{c} = \begin{pmatrix} -4 \\ -8 \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 4 \\ 5 \\ 1 \end{pmatrix} \tag{10}$$

This satisfies all the given conditions when $\lambda = 1$
Thus,

$$\mathbf{c} = \begin{pmatrix} 0 \\ -3 \\ 1 \end{pmatrix} \tag{11}$$

Plot

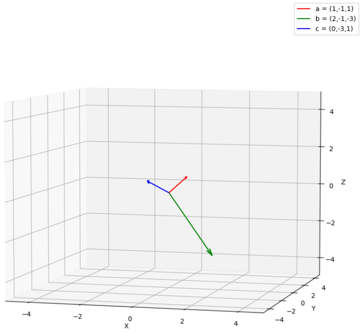

Figure:

C Code

```
#ifndef VECTOR_UTILS_H
#define VECTOR_UTILS_H

// Cross product of 3D vectors
void cross_product(double a[3], double b[3], double result[3]
    result[0] = a[1]*b[2] - a[2]*b[1];
    result[1] = a[2]*b[0] - a[0]*b[2];
    result[2] = a[0]*b[1] - a[1]*b[0];
}

// Dot product of 3D vectors
double dot_product(double a[3], double b[3]) {
    return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
}
```

C Code

```
// Compute vector c = (a*k - (a×b)) / (||a||^2)
void compute_c(double a[3], double b[3], double k, double c[3
    double cross[3];
    cross_product(a, b, cross);

    double norm_sq = dot_product(a, a); // |a|^2

    for (int i = 0; i < 3; i++) {
        c[i] = (a[i]*k - cross[i]) / norm_sq;
    }
}

#endif
```

C Code

```
#include <stdio.h>
#include "solution.h"

int main() {
    double a[3], b[3], c[3], k;
    printf("Enter vector a (3 values): ");
    scanf("%lf %lf %lf", &a[0], &a[1], &a[2]);
    printf("Enter vector b (3 values): ");
    scanf("%lf %lf %lf", &b[0], &b[1], &b[2]);
    printf("Enter scalar k (a·c): ");
    scanf("%lf", &k);
    compute_c(a, b, k, c);
    printf("\nVector c = (%.2lf, %.2lf, %.2lf)\n", c[0], c[1],
    return 0;
}
```

Python + C Code

```
import ctypes
import numpy as np

# Load shared library
lib = ctypes.CDLL("./solution.so")

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

Python + C Code

```
def compute_c(a, b, k):
    a_arr = (ctypes.c_double * 3)(*a)
    b_arr = (ctypes.c_double * 3)(*b)
    c_arr = (ctypes.c_double * 3)()
    lib.compute_c(a_arr, b_arr, ctypes.c_double(k), c_arr)
    return np.array([c_arr[0], c_arr[1], c_arr[2]])

a = list(map(float, input("Enter vector a (3 numbers separate
b = list(map(float, input("Enter vector b (3 numbers separate
k = float(input("Enter scalar k: "))
c = compute_c(a, b, k)

print("Computed c =", c)
```

Python Code

```
import numpy as np

def compute_c_from_a_b_k(a, b, k, tol=1e-9):
    a = np.array(a, dtype=float)
    b = np.array(b, dtype=float)

    if np.linalg.norm(a) < tol:
        raise ValueError("Vector a is (nearly) zero | no uniq
    # For b = a x c, b must be perpendicular to a:
    if abs(np.dot(a, b)) > 1e-8:
        raise ValueError("Inconsistent input: a·b must be 0 fo

    a_cross_b = np.cross(a, b)
    denom = np.dot(a, a)   # |a|^2, guaranteed > 0 here
    c = (a * k - a_cross_b) / denom
    return c
```

Python Code

```
def parse_vec(s):
    vals = list(map(float, s.strip().split()))
    if len(vals) != 3:
        raise ValueError("Please enter exactly 3 numbers for a
    return vals
def main():
    try:
        a = parse_vec(input("Enter vector a (3 values separate
        b = parse_vec(input("Enter vector b (3 values separate
        k = float(input("Enter the value of a·c (scalar k): ")
        c = compute_c_from_a_b_k(a, b, k)
    except Exception as e:
        print("Error:", e)
        return
    print("\nComputed c =", c)
if __name__ == "__main__":
    main()
```