Matgeo Presentation - Problem 2.9.7

ee25btech11056 - Suraj.N

August 30, 2025

Problem Statement

Question :

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

then find $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$

Symbol	Value	Description
$\mathbf{a}$	$\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$	vector
$\mathbf{b}$	$\begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}$	vector
$\mathbf{c}$	$\begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$	vector

Table : vectors

Solution

$$\mathbf{b} \times \mathbf{c} = \begin{pmatrix} |\mathbf{B_{23}} & \mathbf{C_{23}}| \\ |\mathbf{B_{31}} & \mathbf{C_{31}}| \\ |\mathbf{B_{12}} & \mathbf{C_{12}}| \end{pmatrix}$$

$$|\mathbf{B_{23}} \quad \mathbf{C_{23}}| = \begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix} = 3$$

$$|\mathbf{B_{31}} \quad \mathbf{C_{31}}| = \begin{vmatrix} 1 & 2 \\ -1 & 3 \end{vmatrix} = 5$$

$$|\mathbf{B_{12}} \quad \mathbf{C_{12}}| = \begin{vmatrix} -1 & 3 \\ 2 & 1 \end{vmatrix} = -7$$

$$\mathbf{b} \times \mathbf{c} = \begin{pmatrix} 3 \\ 5 \\ -7 \end{pmatrix}$$

$$\text{the value of } \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{a}^{\top}(\mathbf{b} \times \mathbf{c}) = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \begin{pmatrix} 3 \\ 5 \\ -7 \end{pmatrix}$$

$$= (2)(3) + (1)(5) + (3)(\text{-}7) = 6 + 5 - 21$$

$$= \text{-}10$$

$$\textbf{Final Answer} : \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \text{-}10$$

Plot

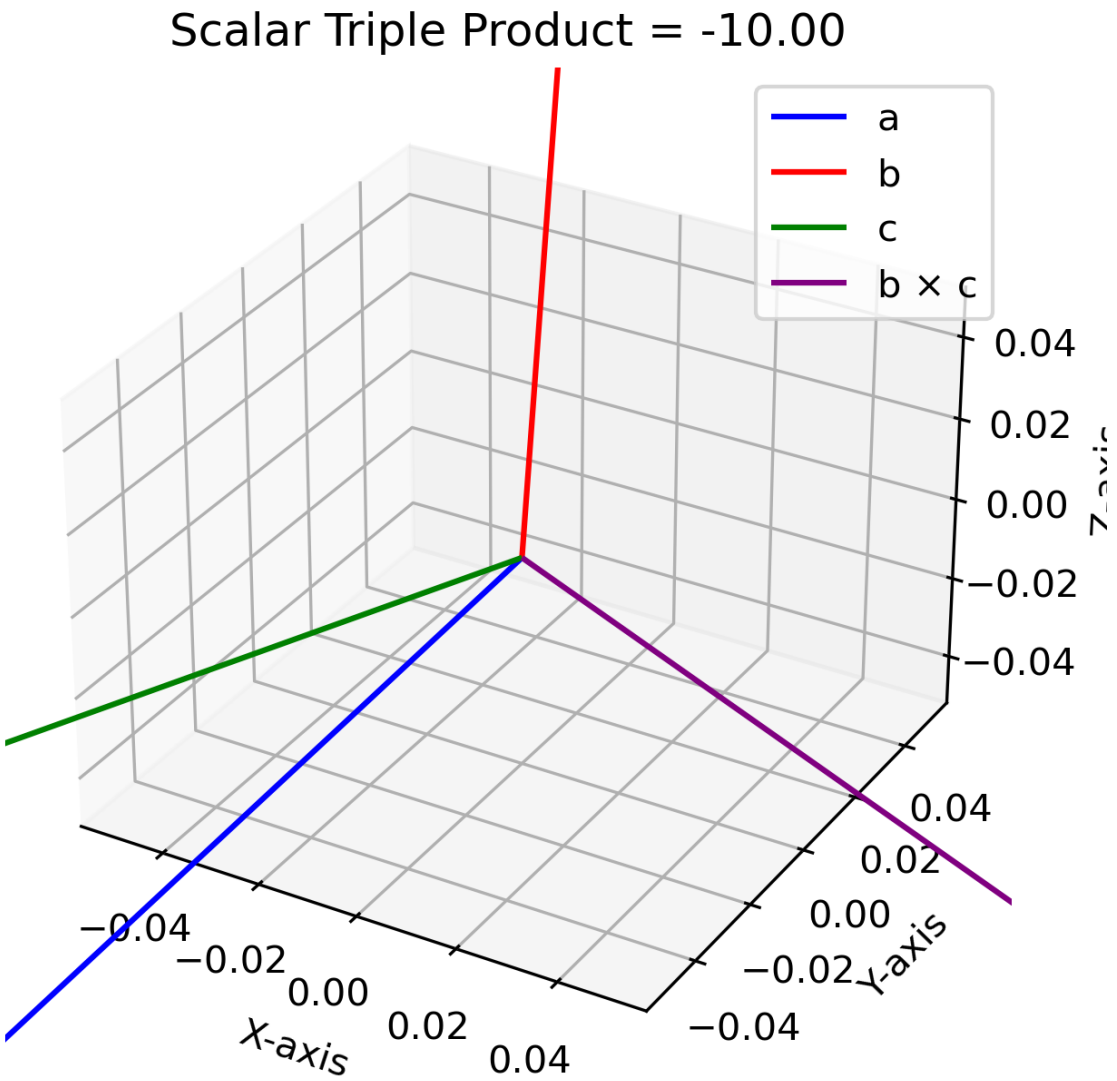

Fig : Vectors

C Code: points.c

```c
#include <stdio.h>

// Function to compute scalar triple product: a  (b  c)
double triple_product(double a[3], double b[3], double c[3]) {
  double cross[3];

  // Cross product b  c
  cross[0] = b[1] * c[2] - b[2] * c[1];
  cross[1] = b[2] * c[0] - b[0] * c[2];
  cross[2] = b[0] * c[1] - b[1] * c[0];

  // Dot product a  (b  c)
  double result = a[0] * cross[0] + a[1] * cross[1] + a[2] * cross[2];
  return result;
}
```

Python: call_c.py

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

# Load shared library
lib = ctypes.CDLL("./libpoints.so")
lib.triple_product.restype = ctypes.c_double
lib.triple_product.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]

# Default vectors (if no args provided)
a = np.array([2.0, 1.0, 3.0], dtype=np.double)
b = np.array([-1.0, 2.0, 1.0], dtype=np.double)
c = np.array([3.0, 1.0, 2.0], dtype=np.double)

# If user provides command-line arguments  override
if len(sys.argv) == 10: # script + 9 numbers
    a = np.array([float(sys.argv[1]), float(sys.argv[2]), float(sys.argv[3])], dtype=np.double)
    b = np.array([float(sys.argv[4]), float(sys.argv[5]), float(sys.argv[6])], dtype=np.double)
    c = np.array([float(sys.argv[7]), float(sys.argv[8]), float(sys.argv[9])], dtype=np.double)

# Call C function
res = lib.triple_product(a.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
                         b.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
                         c.ctypes.data_as(ctypes.POINTER(ctypes.c_double)))

print(f"Scalar␣Triple␣Product␣(from␣C):␣{res}")
```

Python: call_c.py

```
# ---- Visualization using matplotlib ----
cross_bc = np.cross(b, c)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.quiver(0, 0, 0, a[0], a[1], a[2], color='blue', label='a', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, b[0], b[1], b[2], color='red', label='b', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, c[0], c[1], c[2], color='green', label='c', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, cross_bc[0], cross_bc[1], cross_bc[2],
          color='purple', label='b␣␣c', arrow_length_ratio=0.1)

ax.set_title(f"Scalar␣Triple␣Product␣=␣{res:.2f}")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.legend()

plt.savefig("vectors.png", dpi=300, bbox_inches="tight")
plt.show()
```

Python: plot.py

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

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

# Scalar triple product
res = np.dot(a, np.cross(b, c))
print(f"Scalar Triple Product (using NumPy): {res}")

# Cross product b  c
cross_bc = np.cross(b, c)

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

ax.quiver(0, 0, 0, a[0], a[1], a[2], color='blue', label='a', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, b[0], b[1], b[2], color='red', label='b', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, c[0], c[1], c[2], color='green', label='c', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, cross_bc[0], cross_bc[1], cross_bc[2],
          color='purple', label='b  c', arrow_length_ratio=0.1)

ax.set_title(f"Scalar Triple Product = {res:.2f}")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.legend()

plt.savefig("vectors.png", dpi=300, bbox_inches="tight")
plt.show()
```