Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 8, 2025

Problem Statement

If

$$\mathbf{a} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \tag{1.1}$$

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

Description of Variables used

Input variable	**Value**
a	$\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$
b	$\begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}$
c	$\begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$

Table

Theoretical Solution

Write the vectors in component form:

$$\mathbf{a} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}. \tag{2.1}$$

Using the minor notation from the problem statement, where

$$\mathbf{B_{ij}} = \begin{pmatrix} b_i \\ b_j \end{pmatrix}, \qquad \mathbf{C_{ij}} = \begin{pmatrix} c_i \\ c_j \end{pmatrix}, \tag{2.2}$$

we write the cross product :

$$\mathbf{b} \times \mathbf{c} \ = \ \begin{pmatrix} |\mathbf{B_{23}} \ \mathbf{C_{23}}| \\ |\mathbf{B_{31}} \ \mathbf{C_{31}}| \\ |\mathbf{B_{12}} \ \mathbf{C_{12}}| \end{pmatrix}. \tag{2.3}$$

Theoretical Solution

Substituting the components gives

$$\mathbf{b} \times \mathbf{c} = \begin{pmatrix} 3 \\ 5 \\ -7 \end{pmatrix}. \tag{2.4}$$

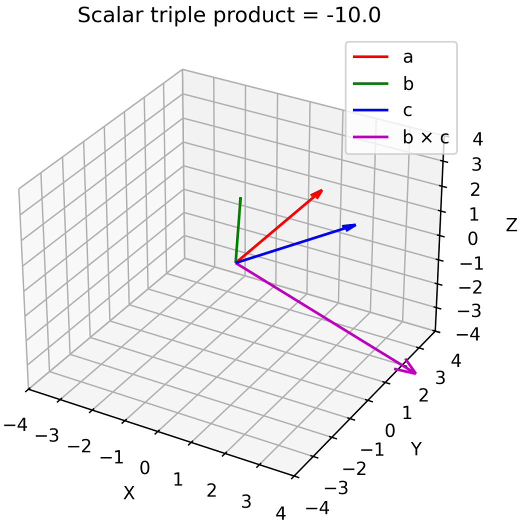
Now use the transpose (row-vector) method for the dot product:

$$\mathbf{a}^T(\mathbf{b} \times \mathbf{c}) = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}^T \begin{pmatrix} 3 \\ 5 \\ -7 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \begin{pmatrix} 3 \\ 5 \\ -7 \end{pmatrix} = 2 \cdot 3 + 1 \cdot 5 + 3 \cdot (-7) = \tag{2.5}$$

Thus

$$\boxed{\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = -10}. \tag{2.6}$$

Plot



Code - C

```
#include <stdio.h>

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

Code - Python(with shared C code)

The code to obtain the required plot is

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

# Load compiled C library
lib = ctypes.CDLL('./libvecops.so')

# Argument/return types
lib.cross_product.argtypes = [ctypes.POINTER(ctypes.c_double),
                              ctypes.POINTER(ctypes.c_double),
                              ctypes.POINTER(ctypes.c_double)]
lib.dot_product.argtypes = [ctypes.POINTER(ctypes.c_double),
                            ctypes.POINTER(ctypes.c_double)]
lib.dot_product.restype = ctypes.c_double
```

Code - Python(with shared C code)

```
# Helper type
DoubleArray3 = ctypes.c_double * 3

# Define vectors
a = np.array([2.0, 1.0, 3.0])
b = np.array([-1.0, 2.0, 1.0])
c = np.array([3.0, 1.0, 2.0])

# Cross product (via C)
cross_res = DoubleArray3()
lib.cross_product(DoubleArray3(*b), DoubleArray3(*c), cross_res)
bx_c = np.array([cross_res[i] for i in range(3)])

# Dot product (via C)
scalar_triple = lib.dot_product(DoubleArray3(*a), cross_res)
```

Code - Python(with shared C code)

```
print("b x c =", bx_c)
print("a . (b x c) =", scalar_triple)

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

def draw_vec(v, color, label):
    ax.quiver(0, 0, 0, v[0], v[1], v[2],
              color=color, arrow_length_ratio=0.1, label=label)

draw_vec(a, 'r', 'a')
draw_vec(b, 'g', 'b')
draw_vec(c, 'b', 'c')
draw_vec(bx_c, 'm', 'b x c')
```

Code - Python(with shared C code)

```
lim = 4
ax.set_xlim([-lim, lim])
ax.set_ylim([-lim, lim])
ax.set_zlim([-lim, lim])

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()

plt.title(f"Scalar triple product = {scalar_triple}")

# Save the image instead of just showing
plt.savefig("/sdcard/ee1030-2025/ai25btech11032/Matgeo/2.7.4/figs/
    triple_product.png", dpi=300)
plt.show()
```

Code - Python only

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

# Define vectors
a = np.array([2.0, 1.0, 3.0])
b = np.array([-1.0, 2.0, 1.0])
c = np.array([3.0, 1.0, 2.0])

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

# Scalar triple product
scalar_triple = np.dot(a, bx_c)
```

Code - Python only

```
# Print results
print("b␣x␣c␣=", bx_c)
print("a␣.␣(b␣x␣c)␣=", scalar_triple)

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

def draw_vec(v, color, label):
    ax.quiver(0, 0, 0, v[0], v[1], v[2],
              color=color, arrow_length_ratio=0.1, label=label)

# Draw vectors
draw_vec(a, 'r', 'a')
draw_vec(b, 'g', 'b')
draw_vec(c, 'b', 'c')
draw_vec(bx_c, 'm', 'b␣x␣c')
```

Code - Python only

```
# Axis limits
lim = 4
ax.set_xlim([-lim, lim])
ax.set_ylim([-lim, lim])
ax.set_zlim([-lim, lim])

# Labels and title
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()

plt.title(f"Scalar triple product = {scalar_triple}")

# Save image
plt.savefig("triple_product_python.png", dpi=300)
plt.show()
```