Presentation - Matgeo

Aryansingh Sonaye
AI25BTECH11032
EE1030 - Matrix Theory

September 11, 2025

Problem Statement

Problem 2.10.21 : Let **a** and **b** be two non-collinear unit vectors. If

$$\mathbf{u} = \mathbf{a} - (\mathbf{a} \cdot \mathbf{b})\mathbf{b}, \quad \mathbf{v} = \mathbf{a} \times \mathbf{b}, \tag{1.1}$$

find $\|\mathbf{v}\|$.
(a) $\|\mathbf{u}\|$
(b) $\|\mathbf{u}\| + |\mathbf{u} \cdot \mathbf{a}|$
(c) $\|\mathbf{u}\| + |\mathbf{u} \cdot \mathbf{b}|$
(d) $\|\mathbf{u}\| + \mathbf{u} \cdot (\mathbf{a} + \mathbf{b})$

Theoretical Solution

$$
\begin{aligned}
\|\mathbf{u}\|^2 &= \mathbf{u}^T\mathbf{u} \\
&= \big(\mathbf{a} - (\mathbf{a}\cdot\mathbf{b})\mathbf{b}\big)^T\big(\mathbf{a} - (\mathbf{a}\cdot\mathbf{b})\mathbf{b}\big) \\
&= \mathbf{a}^T\mathbf{a} - 2(\mathbf{a}\cdot\mathbf{b})^2 + (\mathbf{a}\cdot\mathbf{b})^2\,\mathbf{b}^T\mathbf{b} \\
&= \|\mathbf{a}\|^2 - (\mathbf{a}\cdot\mathbf{b})^2 \qquad (\text{since } \|\mathbf{a}\| = \|\mathbf{b}\| = 1) \\
&= 1 - (\mathbf{a}\cdot\mathbf{b})^2. \qquad (2.1)
\end{aligned}
$$

$$
\begin{aligned}
\|\mathbf{v}\|^2 &= \|\mathbf{a}\times\mathbf{b}\|^2 \\
&= \|\mathbf{a}\|^2\|\mathbf{b}\|^2 - (\mathbf{a}\cdot\mathbf{b})^2 \qquad (\text{vector identity}) \\
&= 1 - (\mathbf{a}\cdot\mathbf{b})^2. \qquad (2.2)
\end{aligned}
$$

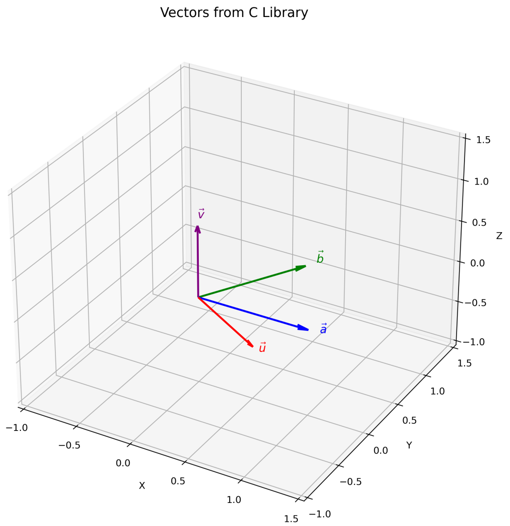
Theoretical Solution

$$(2) \text{ and } (3) \quad \Longrightarrow \quad \|\mathbf{v}\|^2 = \|\mathbf{u}\|^2 \quad \Rightarrow \quad \|\mathbf{v}\| = \|\mathbf{u}\|. \tag{2.3}$$

$$\boxed{\|\mathbf{v}\| = \|\mathbf{u}\|} \tag{2.4}$$

Option A is correct

Plot



Vectors from C Library
$\vec{v}$
$\vec{b}$
$\vec{a}$
$\vec{u}$
X
Y
Z
−1.0
−0.5
0.0
0.5
1.0
1.5

Code - C

```
#include <stdio.h>

// Function to compute dot product
double dot(double a[3], double b[3]) {
    return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
}

// Function to compute cross product
void cross(double a[3], double b[3], double res[3]) {
    res[0] = a[1]*b[2] - a[2]*b[1];
    res[1] = a[2]*b[0] - a[0]*b[2];
    res[2] = a[0]*b[1] - a[1]*b[0];
}
```

Code - C

```
// Function to compute u = a − (a.b)b
void compute_u(double a[3], double b[3], double u[3]) {
    double c = dot(a, b);
    for(int i=0;i<3;i++)
        u[i] = a[i] - c*b[i];
}
```

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 shared library (Linux/Mac)
lib = ctypes.CDLL("./libvector.so")

# Set argument and return types
lib.compute_u.argtypes = [ctypes.POINTER(ctypes.c_double),
                          ctypes.POINTER(ctypes.c_double),
                          ctypes.POINTER(ctypes.c_double)]
lib.cross.argtypes = [ctypes.POINTER(ctypes.c_double),
                      ctypes.POINTER(ctypes.c_double),
                      ctypes.POINTER(ctypes.c_double)]
```

Code - Python(with shared C code)

```
# Helper function to call C compute_u
def compute_u(a, b):
    a_c = (ctypes.c_double * 3)(*a)
    b_c = (ctypes.c_double * 3)(*b)
    u_c = (ctypes.c_double * 3)()
    lib.compute_u(a_c, b_c, u_c)
    return np.array([u_c[0], u_c[1], u_c[2]])

# Helper function to call C cross
def compute_cross(a, b):
    a_c = (ctypes.c_double * 3)(*a)
    b_c = (ctypes.c_double * 3)(*b)
    v_c = (ctypes.c_double * 3)()
    lib.cross(a_c, b_c, v_c)
    return np.array([v_c[0], v_c[1], v_c[2]])
```

Code - Python(with shared C code)

```
# Define vectors
a = np.array([1, 0, 0], dtype=float)
b = np.array([0.5, np.sqrt(3)/2, 0], dtype=float)

# Compute using C functions
u = compute_u(a, b)
v = compute_cross(a, b)

print("u =", u)
print("v =", v)

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

Code - Python(with shared C code)

```
def draw_vec(ax, vec, color, label):
    ax.quiver(0, 0, 0, vec[0], vec[1], vec[2],
              color=color, arrow_length_ratio=0.1, linewidth=2)
    ax.text(vec[0]*1.1, vec[1]*1.1, vec[2]*1.1, label, color=color, fontsize
        =12)

draw_vec(ax, a, "blue", r"$\vec{a}$")
draw_vec(ax, b, "green", r"$\vec{b}$")
draw_vec(ax, u, "red", r"$\vec{u}$")
draw_vec(ax, v, "purple", r"$\vec{v}$")

ax.set_xlim([-1, 1.5])
ax.set_ylim([-1, 1.5])
ax.set_zlim([-1, 1.5])
```

Code - Python(with shared C code)

```
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title("Vectors from C Library", fontsize=14)

plt.tight_layout()

# Save figure to file
plt.savefig("vector_plot.png", dpi=300)
plt.show()
```

Code - Python only

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

# Define vectors
a = np.array([1, 0, 0])
b = np.array([0.5, np.sqrt(3)/2, 0])

# Compute u = a − (a.b)b
u = a − np.dot(a, b) * b

# Compute v = a x b
v = np.cross(a, b)

print("a =", a)
print("b =", b)
print("u =", u)
print("v =", v)
```

Code - Python only

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

def draw_vec(ax, vec, color, label):
    ax.quiver(0, 0, 0, vec[0], vec[1], vec[2],
              color=color, arrow_length_ratio=0.1, linewidth=2)
    ax.text(vec[0]*1.1, vec[1]*1.1, vec[2]*1.1,
            label, color=color, fontsize=12)

# Draw vectors
draw_vec(ax, a, "blue", r"$\vec{a}$")
draw_vec(ax, b, "green", r"$\vec{b}$")
draw_vec(ax, u, "red", r"$\vec{u}$")
draw_vec(ax, v, "purple", r"$\vec{v}$")
```

Code - Python only

```
# Set limits
ax.set_xlim([−1, 1.5])
ax.set_ylim([−1, 1.5])
ax.set_zlim([−1, 1.5])

# Labels
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
ax.set_title("3D-Vectors-a,-b,-u,-v", fontsize=14)

plt.tight_layout()

# Save figure
plt.savefig("vector_plot_3D.png", dpi=300)
plt.show()
```