# 2.4.21

Harsha-EE25BTECH11026

August 24,2025

# Question

The number of vectors of unit length perpendicular to the vectors $a = 2\hat{i} + \hat{j} + 2\hat{k}$ and $b = \hat{j} + \hat{k}$ is

# Theoretical Solution

Given the two vectors,

$$\mathbf{a} = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \tag{1}$$

we need to find the unit vector which is perpendicular to the vectors **a** and **b**.The vector perpendicular to **a** and **b** is given by their cross-product.

# Theoretical Solution

Let the perpendicular vector be $\mathbf{x}^T = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}$

$$\because \mathbf{a}^T\mathbf{x} = 0 \tag{2}$$

$$\mathbf{b}^T\mathbf{x} = 0\ , \tag{3}$$

$$\therefore \begin{pmatrix} \mathbf{a}^T \\ \mathbf{b}^T \end{pmatrix} \mathbf{x} = 0 \tag{4}$$

# Theoretical Solution

$$\begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = 0 \tag{5}$$

This can be represented as,

$$\begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix} \xleftrightarrow{R_1 \leftarrow R_1 - R_2} \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{6}$$

yielding,

$$2x_1 + x_3 = 0 \tag{7}$$

$$x_2 + x_3 = 0 \tag{8}$$

$$\mathbf{x} = x_3 \begin{pmatrix} \frac{-1}{2} \\ -1 \\ 1 \end{pmatrix} = \frac{x_3}{2} \begin{pmatrix} -1 \\ -2 \\ 2 \end{pmatrix} \tag{9}$$

# Theoretical solution

As we know that the vector can be in both the directions i.e, into and out of the plane containing $\mathbf{a}$ and $\mathbf{b}$, so the vector perpendicular to vectors $\mathbf{a}$ and $\mathbf{b}$ would be $\pm(\mathbf{a} \times \mathbf{b})$.

The desired output is

$$\mathbf{x} = \pm\frac{1}{3}\begin{pmatrix} -1 \\ -2 \\ 2 \end{pmatrix} \tag{10}$$

# C Code - Cross product and magnitude of vector

```
#include<stdio.h>
#include<math.h>

double find_magnitude(double *result)
{
        double mag;
        mag=sqrt(pow(result[0],2)+pow(result[1],2)+pow(result
            [2],2));
        return mag;

}
```

# C Code - Cross product and magnitude of vector

```c
double find_cross_product(double *a, double *b, double *result)
{
    float A[2][3];
    float x1, x2, x3;
    // Build the 2x3 matrix [a; b]
    for(int j = 0; j < 3; j++) {
        A[0][j] = a[j];
        A[1][j] = b[j];
    }
    // Row reduction
    if (fabs(A[0][0]) < 1e-6) {
        for(int j = 0; j < 3; j++) {
            float tmp = A[0][j];
            A[0][j] = A[1][j];
            A[1][j] = tmp;
        }
    }
```

# C Code - Cross product and magnitude of vector

```c
float factor = A[1][0] / A[0][0];
for(int j = 0; j < 3; j++) {
    A[1][j] -= factor * A[0][j];
}

// Solve system A * x = 0 with free variable x3 = 1
x3 = 1;
if (fabs(A[1][1]) > 1e-6) {
    x2 = -(A[1][2] * x3) / A[1][1];
} else {
    x2 = 0;
}
x1 = -(A[0][1]*x2 + A[0][2]*x3) / A[0][0];

result[0] = x1;
result[1] = x2;
result[2] = x3;
```

# Python+C Code

```
import numpy as np
import matplotlib as mp
mp.use("TkAgg")
import matplotlib.pyplot as plt
import ctypes

# Load shared library
lib = ctypes.CDLL("./libcrsproduct_mag.so") # <-- change name if
    your .so file is different

# Define argument and return types for the C functions
lib.find_cross_product.argtypes = [
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
lib.find_cross_product.restype = None # because result is
    returned via array
```

# Python+C Code

```python
lib.find_magnitude.argtypes = [ctypes.POINTER(ctypes.c_double)]
lib.find_magnitude.restype = ctypes.c_double

def cross_via_c(a, b):
    a_arr = (ctypes.c_double * 3)(*a)
    b_arr = (ctypes.c_double * 3)(*b)
    result = (ctypes.c_double * 3)()

    lib.find_cross_product(a_arr, b_arr, result)

    return np.array([result[0], result[1], result[2]], dtype=
        float)

a = np.array([2, 1, 2], dtype=np.int32)
b = np.array([0, 1, 1], dtype=np.int32)
```

# Python+C Code

```python
# Cross product from C
x = cross_via_c(a, b)
print("Cross product :", x)

# Magnitude from C
x_ctypes = (ctypes.c_double * 3)(*x)
mag = lib.find_magnitude(x_ctypes)

# Unit vector
u = x / mag

print("Unit vector perpendicular to vectors a and b is \u00B1 ["
    + ", ".join(f"{val:.2f}" for val in u) + "]")
print("That is,")
print("+u =", [format(val, ".2f") for val in u])
print("-u =", [format(val, ".2f") for val in -u])
```

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

# Origin
origin = np.zeros(3)

# Plot a, b, and cross product directions
ax.quiver(*origin, *a, color='r', label='a', arrow_length_ratio=0.1)
ax.quiver(*origin, *b, color='g', label='b', arrow_length_ratio=0.1)
ax.quiver(*origin, *u, color='c', label='(a  b)', arrow_length_ratio=0.1)
ax.quiver(*origin, *-u, color='b', label='-(a  b)', arrow_length_ratio=0.1)
```

# Python+C Code

```
ax.set_xlim([min(a[0], b[0], u[0], -u[0], 0),
             max(a[0], b[0], u[0], -u[0], 0)])

ax.set_ylim([min(a[1], b[1], u[1], -u[1], 0),
             max(a[1], b[1], u[1], -u[1], 0)])

ax.set_zlim([min(a[2], b[2], u[2], -u[2], 0),
             max(a[2], b[2], u[2], -u[2], 0)])

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.savefig("/home/user/Matrix/Matgeo_assignments/2.4.21/figs/
    Figure_1.png")
plt.show()
```

```
import numpy as np
import matplotlib as mp
mp.use("TkAgg")
import matplotlib.pyplot as plt

def cross_via_row_reduction(a, b):
     A = np.array([a, b], dtype=float) # 2x3 system

    # Row reduction manually
    # Step 1: make pivot in first column
     if A[0,0] == 0:
         A[[0,1]] = A[[1,0]] # swap rows if needed

    # Eliminate below
     factor = A[1,0] / A[0,0]
     A[1] = A[1] - factor*A[0]
```

# Python Code

```
# Now we have 2 equations in 3 variables => free variable (say x3 = t)
   # Solve system Ax=0
   # Extract coefficients
    eq1 = A[0]
    eq2 = A[1]
   # Free variable x3 = t
    t = 1 # choose t=1 for direction
   # Solve eq2 for x2 in terms of t
    if abs(eq2[1]) > 1e-12:
       x2 = -(eq2[2]/eq2[1])*t
    else:
       x2 = 0
   # Solve eq1 for x1
    x1 = -(eq1[1]*x2 + eq1[2]*t) / eq1[0]
    return np.array([x1, x2, t])
```

# Python Code

```
# Given vectors
a = np.array([2, 1, 2], dtype=np.int32)
b = np.array([0, 1, 1], dtype=np.int32)

x = cross_via_row_reduction(a, b)
print("Cross product :", x)
mag = np.linalg.norm(x)

u=x/mag

print("Unit vector perpendicular to vectors a and b is \u00B1 ["
    + ", ".join(f"{val:.2f}" for val in u) + "]")
print("That is,")
print("+u =", [format(val, ".2f") for val in u])
print("-u =", [format(val, ".2f") for val in -u])
```

# Python Code

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

# Origin
origin = np.zeros(3)

# Plot a, b, and cross product
ax.quiver(*origin, *a, color='r', label='a', arrow_length_ratio
    =0.1)
ax.quiver(*origin, *b, color='g', label='b', arrow_length_ratio
    =0.1)
ax.quiver(*origin, *-u, color='b', label='-(a  b)',
    arrow_length_ratio=0.1)
ax.quiver(*origin, *u, color='c', label='(a  b)' ,
    arrow_length_ratio=0.1)
```

# Python Code

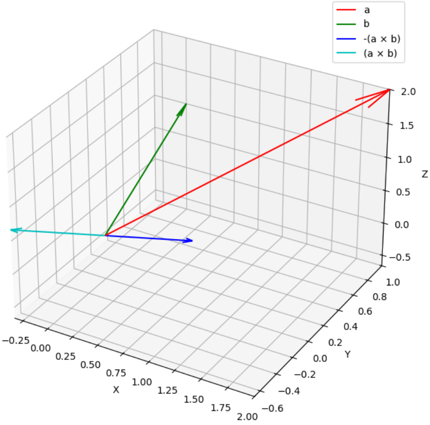
```
ax.set_xlim([min(a[0], b[0], u[0], -u[0], 0),
             max(a[0], b[0], u[0], -u[0], 0)])

ax.set_ylim([min(a[1], b[1], u[1], -u[1], 0),
             max(a[1], b[1], u[1], -u[1], 0)])

ax.set_zlim([min(a[2], b[2], u[2], -u[2], 0),
             max(a[2], b[2], u[2], -u[2], 0)])


ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()
plt.savefig("/home/user/Matrix/Matgeo_assignments/2.4.21/figs/
    Figure_1.png")
plt.show()
```

# Plot


a
b
-(a × b)
(a × b)
2.0
1.5
1.0
0.5
0.0
−0.5
Z
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Y
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
X