2.4.37

Naman Kumar-EE25BTECH11041

6 September,2025

Question

Find a vector of magnitude 6, which is perpendicular to both the vectors $2\hat{\imath} - \hat{\jmath} + 2\hat{k}$ and $4\hat{\imath} - \hat{\jmath} + 3\hat{k}$

Equation Used

Inner Product,

$$\mathbf{A} \cdot \mathbf{B} \tag{1}$$

Also Rank of matrix

Solution

Given Vectors

$$\mathbf{A} = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 4 \\ -1 \\ 3 \end{pmatrix} \tag{2}$$

Let required vector be,

$$\mathbf{C} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3}$$

Solution

Using Inner Product,

$$\mathbf{C}^T \cdot \mathbf{A} = 0 \text{ and } \mathbf{C}^T \cdot \mathbf{B} = 0 \tag{4}$$

$$\mathbf{C}^T \cdot \mathbf{A} = 2x - y + 2z = 0 \tag{5}$$

$$\mathbf{C}^T \cdot \mathbf{B} = 4x - y + 3z = 0 \tag{6}$$

Using Row Transformations to Get Row Reduced echelon Form

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 2 \\ 4 & -1 & 3 \end{pmatrix} \xrightarrow{R_2 \to R_2 - 2R_1} \begin{pmatrix} 2 & -1 & 2 \\ 0 & 1 & -1 \end{pmatrix} \tag{7}$$

$$\xrightarrow{R_1 \to \frac{1}{2} R_1} \begin{pmatrix} 1 & \frac{-1}{2} & 1 \\ 0 & 1 & -1 \end{pmatrix} \tag{8}$$

$$\xrightarrow{R_1 \to R_1 + \frac{1}{2} R_2} \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ 0 & 1 & -1 \end{pmatrix} \tag{9}$$

Solution

$$\mathbf{A} = (\mathbf{I}\mathbf{X}), \mathbf{I} \text{ is identity matrix} \tag{10}$$

And, **X** is

$$\begin{pmatrix} \frac{1}{2} \\ -1 \end{pmatrix} \tag{11}$$

Since rank of matrix is 2($\leq$ 3), their are infinite many solutions $R^3 \rightarrow R^2$
From the Row Reduced Echelon form(RREF),we can write the new system of equation:

$$x + \frac{1}{2}z = 0 \tag{12}$$

$$y - z = 0 \tag{13}$$

Solution

Therefore vector **C** using equations (12) and (13) is

$$\mathbf{C} = \begin{pmatrix} x \\ -2x \\ -2x \end{pmatrix} = x \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix} \tag{14}$$

Now getting vector with magnitude 6

$$\|C\| = 6 \tag{15}$$

$$\|x\| \sqrt{(1)^2 + (-2)^2 + (-2)^2} = 6 \tag{16}$$

$$\|x\| \sqrt{1 + 4 + 4} = 6 \tag{17}$$

$$\|x\| \sqrt{9} = 6 \tag{18}$$

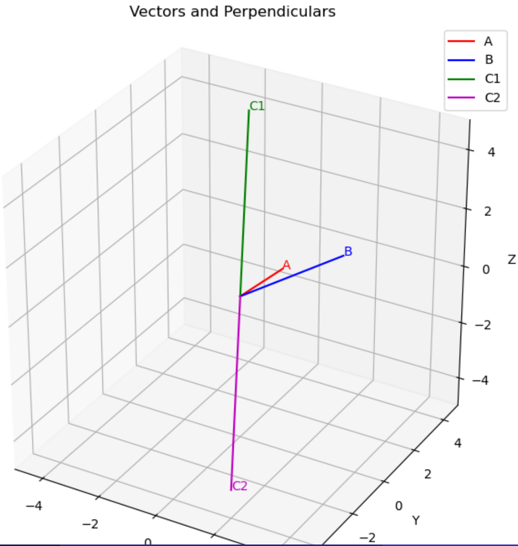
$$\|x\| = 2 \tag{19}$$

Solution

Therefore final vectors are

$$C_1 = \begin{pmatrix} -2 \\ 4 \\ 4 \end{pmatrix}, C_2 = \begin{pmatrix} 2 \\ -4 \\ -4 \end{pmatrix} \tag{20}$$

Figure



C code

```c
#include <math.h>

// Compute cross product of two 3D vectors
void cross_product(double *a, double *b, double *result) {
    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];
}

// Compute magnitude of a 3D vector
double magnitude(double *v) {
    return sqrt(v[0]*v[0] + v[1]*v[1] + v[2]*v[2]);
}
```

C code

```
// Given a, b, and desired magnitude, compute two perpendicular
    vectors of that magnitude
// Output: v1[3], v2[3]
void perpendicular_vectors(double *a, double *b, double mag,
    double *v1, double *v2) {
    double c[3];
    cross_product(a, b, c);

    double norm_c = magnitude(c);

    if (norm_c == 0.0) {
        // Parallel vectors  cross product is zero
        v1[0] = v1[1] = v1[2] = 0.0;
        v2[0] = v2[1] = v2[2] = 0.0;
        return;
    }
```

C code

```
    // Normalize c
    double c_hat[3];
    c_hat[0] = c[0] / norm_c;
    c_hat[1] = c[1] / norm_c;
    c_hat[2] = c[2] / norm_c;

    // Scale to desired magnitude
    v1[0] = mag * c_hat[0];
    v1[1] = mag * c_hat[1];
    v1[2] = mag * c_hat[2];

    v2[0] = -v1[0];
    v2[1] = -v1[1];
    v2[2] = -v1[2];
}
```

Python Code through SO file

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

# Load C shared object
lib = ctypes.CDLL("./main.so")

# Define argument and return types
lib.perpendicular_vectors.argtypes = [
    ctypes.POINTER(ctypes.c_double), # a
    ctypes.POINTER(ctypes.c_double), # b
    ctypes.c_double, # magnitude
    ctypes.POINTER(ctypes.c_double), # v1
    ctypes.POINTER(ctypes.c_double) # v2
]
lib.perpendicular_vectors.restype = None
```

Python Code through SO file

```
# Given vectors
a = np.array([2.0, -1.0, 2.0], dtype=np.double)
b = np.array([4.0, -1.0, 3.0], dtype=np.double)
magnitude = 6.0

# Prepare outputs
v1 = np.zeros(3, dtype=np.double)
v2 = np.zeros(3, dtype=np.double)

# Call C function
lib.perpendicular_vectors(
    a.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    b.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    magnitude,
    v1.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    v2.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
)
```

```
print(f"Vector a: {a}")
print(f"Vector b: {b}")
print(f"First perpendicular vector v1 (|v1|={np.linalg.norm(v1)
    :.2f}): {v1}")
print(f"Second perpendicular vector v2 (|v2|={np.linalg.norm(v2)
    :.2f}): {v2}")

# --- Plotting ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
origin = [0, 0, 0]

ax.plot([0,a[0]], [0,a[1]], [0,a[2]], color='r', label='A')
ax.plot([0,b[0]], [0,b[1]], [0,b[2]], color='b', label='B')
```

Python Code through SO file

```
ax.plot([0,v1[0]], [0,v1[1]], [0,v1[2]], color='g', label='V1')
ax.plot([0,v2[0]], [0,v2[1]], [0,v2[2]], color='m', label='V2')

ax.text(a[0], a[1], a[2], 'A', color='r')
ax.text(b[0], b[1], b[2], 'B', color='b')
ax.text(v1[0], v1[1], v1[2], 'C1', color='g')
ax.text(v2[0], v2[1], v2[2], 'C2', color='m')

max_val = np.max(np.abs(np.vstack((a, b, v1, v2)))) * 1.2
ax.set_xlim([-max_val, max_val])
ax.set_ylim([-max_val, max_val])
ax.set_zlim([-max_val, max_val])
ax.set_xlabel('X')
```

Python Code through SO file

```
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Vectors and Perpendiculars')
ax.legend()
ax.grid(True)
ax.set_box_aspect([1,1,1])

plt.savefig('vector_plot.png')

plt.show()
```

Direct Python Code

```
# Inspired by code from GVV Sharma
# September 5, 2024
# released under GNU GPL
# Find two vectors of magnitude 6 perpendicular to two given
    vectors and plot them.

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Given vectors
a = np.array([2, -1, 2])
b = np.array([4, -1, 3])

# To find a vector perpendicular to both a and b, we compute
    their cross product.
c = np.cross(a, b)
```

Direct Python Code

```
print(f"Vector 'a': {a}")
print(f"Vector 'b': {b}")
print(f"Vector perpendicular to a and b (a x b = c): {c}")

# Now, we need a vector of magnitude 6 in the direction of c.
# First, find the unit vector in the direction of c.
# Magnitude of c
norm_c = np.linalg.norm(c)
# Unit vector c_hat
c_hat = c / norm_c

print(f"Magnitude of c: {norm_c:.2f}")
print(f"Unit vector in the direction of c: {c_hat}")
```

Direct Python Code

```python
# Define the desired magnitude
magnitude = 6

# Calculate the final vector v and its opposite
v1 = magnitude * c_hat
v2 = -v1

print(f"The first required vector 'v1' of magnitude {magnitude}
    is: {v1}")
print(f"The second required vector 'v2' of magnitude {magnitude}
    is: {v2}")
print(f"Verification: Magnitude of v1 is {np.linalg.norm(v1):.2f}
    ")
print(f"Verification: Magnitude of v2 is {np.linalg.norm(v2):.2f}
    ")
```

Direct Python Code

```python

# --- Plotting the vectors ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Origin point
origin = [0, 0, 0]

# Plotting the vectors as lines from origin
ax.plot([origin[0], a[0]], [origin[1], a[1]], [origin[2], a[2]],
    color='r', label='A = [2, -1, 2]')
ax.plot([origin[0], b[0]], [origin[1], b[1]], [origin[2], b[2]],
    color='b', label='B = [4, -1, 3]')
ax.plot([origin[0], v1[0]], [origin[1], v1[1]], [origin[2], v1
    [2]], color='g', label=f'Result C1 = [{v1[0]:.0f}, {v1[1]:.0f
    }, {v1[2]:.0f}]')
```

Direct Python Code

```
ax.plot([origin[0], v2[0]], [origin[1], v2[1]], [origin[2], v2[2]], color='m', label=f'Result C2 = [{v2[0]:.0f}, {v2[1]:.0f}, {v2[2]:.0f}]')

# Adding text labels at the end of each vector
ax.text(a[0]*1.1, a[1]*1.1, a[2]*1.1, 'A', color='r', fontsize=12)
ax.text(b[0]*1.1, b[1]*1.1, b[2]*1.1, 'B', color='b', fontsize=12)
ax.text(v1[0]*1.1, v1[1]*1.1, v1[2]*1.1, 'C1', color='g', fontsize=12)
ax.text(v2[0]*1.1, v2[1]*1.1, v2[2]*1.1, 'C2', color='m', fontsize=12)
```

Direct Python Code

```
# Setting the plot limits to be symmetric and encompass all
    vectors
max_val = np.max(np.abs(np.vstack((a, b, v1)))) * 1.2
ax.set_xlim([-max_val, max_val])
ax.set_ylim([-max_val, max_val])
ax.set_zlim([-max_val, max_val])

# Adding labels and title
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
ax.set_title('Perpendicular Vector Visualization')
ax.legend()
ax.grid(True)
```

Direct Python Code

```
# Change the viewing angle (elevation, azimuth)
ax.view_init(elev=25, azim=-45)

# To make the aspect ratio equal
ax.set_box_aspect([1,1,1])

# Save the figure as a PNG file
plt.savefig('vector_plot.png')

# Show the plot
plt.show()
```