2.5.25

Vaishnavi - EE25BTECH11059

September 9, 2025

Question

If $\mathbf{a} = 2\hat{i} - \hat{j} - 2\hat{k}$ and $\mathbf{b} = 7\hat{i} + 2\hat{j} - 3\hat{k}$, then express $\mathbf{b}$ in the form $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where $\mathbf{b}_1$ is parallel to $\mathbf{a}$ and $\mathbf{b}_2$ is perpendicular to $\mathbf{a}$.

Solution

Variable	Value
a	$2\hat{i} - \hat{j} - 2\hat{k}$
b	$7\hat{i} + 2\hat{j} - 3\hat{k}$

Table: Variables Used

Solution

$$\mathbf{a} = \begin{pmatrix} 2 \\ -1 \\ -2 \end{pmatrix} \tag{1}$$

$$\mathbf{b} = \begin{pmatrix} 7 \\ 2 \\ -3 \end{pmatrix} \tag{2}$$

solution

Using the Gram-Schmidt approach
$\mathbf{b_1}$ is the projection of $\mathbf{b}$ on $\mathbf{a}$

$$\mathbf{b_1} = \frac{\mathbf{a^T b}}{\mathbf{a^T a}}\mathbf{a} \tag{3}$$

$$\mathbf{b_1} = \frac{18}{9}\mathbf{a} \tag{4}$$

$$\mathbf{b_1} = 2\mathbf{a} \tag{5}$$

Solution

$$\mathbf{b}_2 = \mathbf{b} - \mathbf{b}_1 = \begin{pmatrix} 7 \\ 2 \\ -3 \end{pmatrix} - \begin{pmatrix} 4 \\ -2 \\ -4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 1 \end{pmatrix} \tag{6}$$

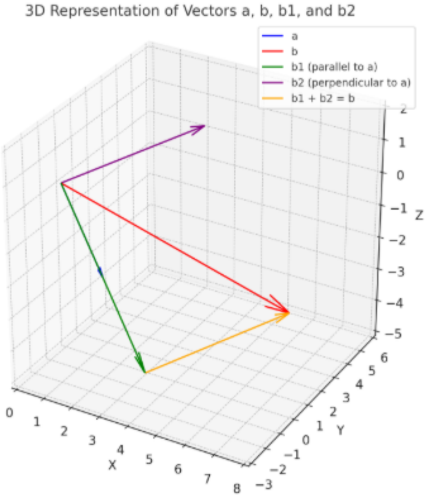
$$\begin{pmatrix} 7 \\ 2 \\ -3 \end{pmatrix} = \begin{pmatrix} 4 \\ -2 \\ -4 \end{pmatrix} + \begin{pmatrix} 3 \\ 4 \\ 1 \end{pmatrix} \tag{7}$$

Therefore,

$$\mathbf{b_1} = \begin{pmatrix} 4 \\ -2 \\ -4 \end{pmatrix} \tag{8}$$

$$\mathbf{b_2} = \begin{pmatrix} 3 \\ 4 \\ 1 \end{pmatrix} \tag{9}$$

Refer to Figure



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

# Define vectors
a = np.array([2, -1, -2])
b = np.array([7, 2, -3])
b1 = np.array([4, -2, -4])
b2 = np.array([3, 4, 1])

# Function to draw vectors
def draw_vector(ax, start, vec, color, label):
    ax.quiver(*start, *vec, color=color, label=label,
        arrow_length_ratio=0.1)

# Create 3D plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection='3d')
```

Python Code

```
# Draw from origin
origin = np.array([0,0,0])
draw_vector(ax, origin, a, 'blue', 'a')
draw_vector(ax, origin, b, 'red', 'b')
draw_vector(ax, origin, b1, 'green', 'b1 (parallel to
    a)')
draw_vector(ax, origin, b2, 'purple', 'b2 (
    perpendicular to a)')

# Show b as b1 + b2 (parallelogram completion)
draw_vector(ax, b1, b2, 'orange', 'b1 + b2 = b')
```

Python Code

```
# Labels and title
ax.set_xlabel('X', fontsize=12)
ax.set_ylabel('Y', fontsize=12)
ax.set_zlabel('Z', fontsize=12)
ax.set_title( 3D Representation of Vectors a, b, b1, and b2 , fontsize=14)
ax.legend()

# Grid and aspect ratio
ax.grid(True)
ax.set_box_aspect([1,1,1])

# Axis limits
ax.set_xlim(0,8)
ax.set_ylim(-3,6)
ax.set_zlim(-5,2)

# Save figure
plt.savefig( Graph3.png , dpi=300, bbox_inches='tight'
```

C Code

```c
#include <stdio.h>

int dotProduct(int a[], int b[], int size) {
    int dot = 0;
    for (int i = 0; i < size; i++) {
        dot += a[i] * b[i];
    }
    return dot;
}

void scalarMultiply(int vector[], int scalar, int
   result[], int size) {
    for (int i = 0; i < size; i++) {
        result[i] = scalar * vector[i];
    }
}
```

C Code

```c
void vectorSubtract(int a[], int b[], int result[],
    int size) {
    for (int i = 0; i < size; i++) {
        result[i] = a[i] - b[i];
    }
}

void solve_vectors() {
    int a[3] = {2, -1, -2};
    int b[3] = {7, 2, -3};

    int a_dot_b = dotProduct(a, b, 3);
    int a_dot_a = dotProduct(a, a, 3);

    int k = a_dot_b / a_dot_a;
```

```c
    int b1[3];
    scalarMultiply(a, k, b1, 3);

    int b2[3];
    vectorSubtract(b, b1, b2, 3);

    printf( Vector a: [%d, %d, %d]\n , a[0], a[1], a[2]);
    printf( Vector b: [%d, %d, %d]\n , b[0], b[1], b[2]);
    printf( Scalar k: %d\n , k);
    printf( Vector b1 (parallel to a): [%d, %d, %d]\n , b1[0], b1[1], b1[2]);
    printf( Vector b2 (perpendicular to a): [%d, %d, %d]\n , b2[0], b2[1], b2[2]);
}
```

Python and C Code

```python
import ctypes

# Load the shared object file
lib = ctypes.CDLL('./code.so')

# Call the solve_vectors function (no args, no return)
lib.solve_vectors()
```