2.8.24

Nipun Dasari - EE25BTECH11042

September 19, 2025

Question

The $\mathbf{a} + \mathbf{b}$ bisects the angle between $\mathbf{a}$ and $\mathbf{b}$ if ____________

Theoretical Solution

Theorem: The $\mathbf{a}+\mathbf{b}$ bisects the angle between $\mathbf{a}$ and $\mathbf{b}$ if and only if $\|\mathbf{a}\|$ $= \|\mathbf{b}\|$
We prove the above in two parts:
Assume a $\mathbf{c}$ such that

$$\mathbf{c} = \mathbf{a} + \mathbf{b} \tag{1}$$

Let α and β be angles made by $\mathbf{c}$ with $\mathbf{a}$ and $\mathbf{b}$ respectively. **Part 1** Given :

$$\|\mathbf{a}\| = \|\mathbf{b}\| \tag{2}$$

Theoretical Solution

To prove : $\mathbf{a}+\mathbf{b}$ bisects the angle between $\mathbf{a}$ and $\mathbf{b}$
Proof:
The angle θ between $\mathbf{p}$ and $\mathbf{q}$ is given by:

$$\cos\theta = \frac{\mathbf{p}^\top \mathbf{q}}{\|\mathbf{p}\| \, \|\mathbf{q}\|} \tag{3}$$

By (3) and (1)

$$\implies \cos\alpha = \frac{\mathbf{a}^\top (\mathbf{a}+\mathbf{b})}{\|\mathbf{a}\| \, \|\mathbf{a}+\mathbf{b}\|} \tag{4}$$

$$\implies \cos\beta = \frac{\mathbf{b}^\top (\mathbf{a}+\mathbf{b})}{\|\mathbf{b}\| \, \|\mathbf{a}+\mathbf{b}\|} \tag{5}$$

Theoretical Solution

By (2)

$$\mathbf{a}^\top \mathbf{a} = \mathbf{b}^\top \mathbf{b} \tag{6}$$

$$\mathbf{a}^\top \mathbf{a} + \mathbf{a}^\top \mathbf{b} = \mathbf{b}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a} \tag{7}$$

$$\frac{\mathbf{a}^\top \mathbf{a} + \mathbf{a}^\top \mathbf{b}}{\left\|\mathbf{a}\right\| \left\|\mathbf{a}+\mathbf{b}\right\|} = \frac{\mathbf{b}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a}}{\left\|\mathbf{b}\right\| \left\|\mathbf{a}+\mathbf{b}\right\|} \tag{8}$$

$$\therefore \cos\alpha = \cos\beta \tag{9}$$

$$\therefore \alpha = \beta \tag{10}$$

Part 2

Part 2

Given:

$$\alpha = \beta \tag{11}$$

To prove:

$$\|\mathbf{a}\| = \|\mathbf{b}\| \tag{12}$$

Proof:

By (9)

$$\cos\alpha = \cos\beta \tag{13}$$

$$\frac{\mathbf{a}^\top(\mathbf{a}+\mathbf{b})}{\|\mathbf{a}\|\,\|\mathbf{a}+\mathbf{b}\|} = \frac{\mathbf{b}^\top(\mathbf{a}+\mathbf{b})}{\|\mathbf{b}\|\,\|\mathbf{a}+\mathbf{b}\|} \tag{14}$$

$$\|\mathbf{b}\|\left(\|\mathbf{a}\|^2 + \mathbf{a}^\top\mathbf{b}\right) = \|\mathbf{a}\|\left(\|\mathbf{b}\|^2 + \mathbf{a}^\top\mathbf{b}\right) \tag{15}$$

$$\|\mathbf{b}\|\,\|\mathbf{a}\|^2 + \|\mathbf{b}\|\left(\mathbf{a}^\top\mathbf{b}\right) = \|\mathbf{a}\|\,\|\mathbf{b}\|^2 + \|\mathbf{a}\|\left(\mathbf{a}^\top\mathbf{b}\right) \tag{16}$$

Theoretical Solution

Rearrange the terms to group common factors:

$$\|\mathbf{b}\| \|\mathbf{a}\|^2 - \|\mathbf{a}\| \|\mathbf{b}\|^2 = \|\mathbf{a}\| (\mathbf{a}^\top \mathbf{b}) - \|\mathbf{b}\| (\mathbf{a}^\top \mathbf{b}) \tag{17}$$

$$\|\mathbf{a}\| \|\mathbf{b}\| (\|\mathbf{a}\| - \|\mathbf{b}\|) = (\mathbf{a}^\top \mathbf{b})(\|\mathbf{a}\| - \|\mathbf{b}\|) \tag{18}$$

$$(\|\mathbf{a}\| - \|\mathbf{b}\|)(\|\mathbf{a}\| \|\mathbf{b}\| - \mathbf{a}^\top \mathbf{b}) = 0 \tag{19}$$

Theoretical Solution

This equation gives two possibilities:

$$\|\mathbf{a}\| - \|\mathbf{b}\| = 0 \implies \|\mathbf{a}\| = \|\mathbf{b}\| \tag{20}$$

$$\|\mathbf{a}\| \|\mathbf{b}\| - \mathbf{a}^\top \mathbf{b} = 0 \implies \mathbf{a}^\top \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \tag{21}$$

(21) is incorrect as parallel vectors are not being assumed.
Thus proved

C Code

```c
#include <math.h> // Required for sqrt() and fabs()

// Define a small constant for floating-point comparisons
#define EPSILON 1e-9

// Structure to represent a 3D vector
typedef struct {
        double x;
        double y;
        double z;
} Vector3D;

/**
* @brief Calculates the magnitude (length) of a 3D vector.
* @param v The input Vector3D.
* @return The magnitude of the vector.
```

C Code

```c
double vector_magnitude(Vector3D v)
    {
        return sqrt(v.x * v.x + v.y *
            v.y + v.z * v.z);
}
```

C Code

```
int does_sum_bisect_angle(Vector3D a, Vector3D b) {
double mag_a = vector_magnitude(a);
double mag_b = vector_magnitude(b);

// Compare magnitudes with a small tolerance for floating-point
    numbers.
// If the absolute difference is less than EPSILON, consider them
    equal.
if (fabs(mag_a - mag_b) < EPSILON) {
return 1; // Magnitudes are approximately equal.
} else {
return 0; // Magnitudes are not equal.
```

Python Code using shared output

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import ctypes
import os
# --- Ctypes Setup ---
# Define the Vector3D structure to match the C definition
class Vector3D(ctypes.Structure):
_fields_ = [
(x, ctypes.c_double),
(y, ctypes.c_double),
(z, ctypes.c_double)
                        ]
```

Python Code using shared output

```
# Load the shared library
lib = ctypes.CDLL('./2.8.24.so')
# Define the argument types and return types for the C functions
# For vector_magnitude: takes Vector3D, returns double
lib.vector_magnitude.argtypes = [Vector3D]
lib.vector_magnitude.restype = ctypes.c_double
# For does_sum_bisect_angle: takes two Vector3D, returns int
lib.does_sum_bisect_angle.argtypes = [Vector3D, Vector3D]
lib.does_sum_bisect_angle.restype = ctypes.c_int
# --- Python Helper Functions (using numpy where appropriate for
    plotting) ---
```

Python Code using shared output

```
def angle_between_vectors_np(v1_np, v2_np):
Calculates the angle in degrees between two numpy vectors.
mag1 = np.linalg.norm(v1_np)
mag2 = np.linalg.norm(v2_np)
if mag1 == 0 or mag2 == 0:
return np.nan # Undefined angle with a zero vector
dot_product = np.dot(v1_np, v2_np)
cos_theta = np.clip(dot_product / (mag1 * mag2), -1.0, 1.0)
angle_rad = np.arccos(cos_theta)
return np.degrees(angle_rad)
def plot_vector_bisection_ctypes(a_np, b_np, title_suffix=):

Plots vectors a, b, and a+b in 3D and displays angle bisection
    information.
Uses C functions via ctypes for magnitude calculation and
    bisection check.

```

Python Code using shared output

```
        fig = plt.figure(figsize=(10, 8))
        ax = fig.add_subplot(111, projection='3d')
        origin = np.array([0, 0, 0])
# Calculate sum vector using numpy
Sum_vec_np = a_np + b_np
# Plot vectors
ax.quiver(*origin, *a_np, color='r', arrow_length_ratio=0.1,
    label='Vector a')
ax.quiver(*origin, *b_np, color='g', arrow_length_ratio=0.1,
    label='Vector b')
ax.quiver(*origin, *sum_vec_np, color='b', arrow_length_ratio
    =0.1, label='Vector a + b')
# Convert numpy arrays to ctypes Vector3D structures for C
    function calls
ctypes_a = Vector3D(x=a_np[0], y=a_np[1], z=a_np[2])
ctypes_b = Vector3D(x=b_np[0], y=b_np[1], z=b_np[2])
ctypes_sum_vec = Vector3D(x=sum_vec_np[0], y=sum_vec_np[1], z=
    sum_vec_np[2])
```

Python Code using shared output

```
# Calculate magnitudes using the C function
mag_a_c = lib.vector_magnitude(ctypes_a)
mag_b_c = lib.vector_magnitude(ctypes_b)
mag_sum_c = lib.vector_magnitude(ctypes_sum_vec)
# Calculate angles using numpy for clarity in visualization
angle_a_sum = angle_between_vectors_np(a_np, sum_vec_np)
angle_b_sum = angle_between_vectors_np(b_np, sum_vec_np)
angle_a_b = angle_between_vectors_np(a_np, b_np)
# Set plot limits
max_coord = np.max(np.abs([a_np, b_np, sum_vec_np])) * 1.2
ax.set_xlim([-max_coord, max_coord])
ax.set_ylim([-max_coord, max_coord])
ax.set_zlim([-max_coord, max_coord])
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
```

Python Code using shared output

```
# Add text for magnitudes and angles
info_text = fMagnitudes (from C):\n||a|| = {mag_a_c:.6f}\n||b|| = {mag_b_c:.6f}\n
info_text += fAngles (deg, from Python):\nAngle(a, a+b) = {angle_a_sum:.2f}\nAngle(b, a+b) = {angle_b_sum:.2f}\n
info_text += fAngle(a, b) = {angle_a_b:.2f}\n
# Check for bisection condition using the C function
is_bisected_c = lib.does_sum_bisect_angle(ctypes_a, ctypes_b)
if is_bisected_c:
info_text += \nResult (from C): Magnitudes are equal (within EPSILON),\n so a+b bisects the angle (alpha ~ beta).
fig_title = Angle Bisected (Rhombus Case)
else:
info_text += \nResult (from C): Magnitudes are NOT equal,\n so a+b does NOT bisect the angle.
fig_title = Angle Not Bisected (Parallelogram Case)
```

Python Code using shared output

```
ax.set_title(f{fig_title} {title_suffix}\n{info_text}, loc='left
    ', fontsize=10)
ax.legend()
plt.tight_layout()
plt.show()
# --- Test Cases ---
print(--- Case 1: Magnitudes are equal (Expected from C: Bisects)
    ---)
a1 = np.array([1.0, 2.0, 0.0])
b1 = np.array([2.0, -1.0, 0.0])
# ||a1|| = sqrt(1^2 + 2^2) = sqrt(5)
# ||b1|| = sqrt(2^2 + (-1)^2) = sqrt(5)
plot_vector_bisection_ctypes(a1, b1, (a=[1,2,0], b=[2,-1,0]))
print(\n--- Case 2: Magnitudes are different (Expected from C:
    Does NOT Bisect) ---)
a2 = np.array([3.0, 0.0, 0.0]) # Mag = 3
b2 = np.array([1.0, 1.0, 0.0]) # Mag = sqrt(2) approx 1.414
plot_vector_bisection_ctypes(a2, b2, (a=[3,0,0], b=[1,1,0]))
```

Plot by python using shared output from c