2.8.25

EE25BTECH11043 - Nishid Khandagre

September 19, 2025

Question

If $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are mutually perpendicular vectors of equal magnitudes, show that $\mathbf{A} + \mathbf{B} + \mathbf{C}$ is equally inclined to $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

Theoretical Solution

Given:

$$\mathbf{A}^\top \mathbf{B} = 0 \tag{1}$$

$$\mathbf{B}^\top \mathbf{C} = 0 \tag{2}$$

$$\mathbf{C}^\top \mathbf{A} = 0 \tag{3}$$

$$\|\mathbf{A}\| = \|\mathbf{B}\| = \|\mathbf{C}\| = k \tag{4}$$

Theoretical Solution

This implies:

$$\mathbf{A}^\top \mathbf{A} = \left\|\mathbf{A}\right\|^2 = k^2 \tag{5}$$

$$\mathbf{B}^\top \mathbf{B} = \left\|\mathbf{B}\right\|^2 = k^2 \tag{6}$$

$$\mathbf{C}^\top \mathbf{C} = \left\|\mathbf{C}\right\|^2 = k^2 \tag{7}$$

Let

$$\mathbf{R} = \left(\mathbf{A} + \mathbf{B} + \mathbf{C}\right) \tag{8}$$

The cosine of the angle θ between two vectors $\mathbf{X}$ and $\mathbf{Y}$ is given by:

$$\cos\theta = \frac{\mathbf{X}^\top \mathbf{Y}}{\left\|\mathbf{X}\right\| \left\|\mathbf{Y}\right\|} \tag{9}$$

Theoretical Solution

$$\|\mathbf{R}\|^2 = \mathbf{R}^\top\mathbf{R} \tag{10}$$

$$= \left(\mathbf{A}+\mathbf{B}+\mathbf{C}\right)^\top\left(\mathbf{A}+\mathbf{B}+\mathbf{C}\right) \tag{11}$$

$$= \mathbf{A}^\top\mathbf{A}+\mathbf{A}^\top\mathbf{B}+\mathbf{A}^\top\mathbf{C}+\mathbf{B}^\top\mathbf{A}+\mathbf{B}^\top\mathbf{B}+\mathbf{B}^\top\mathbf{C}+\mathbf{C}^\top\mathbf{A}+\mathbf{C}^\top\mathbf{B}+\mathbf{C}^\top\mathbf{C} \tag{12}$$

$$= \|\mathbf{A}\|^2+0+0+0+\|\mathbf{B}\|^2+0+0+0+\|\mathbf{C}\|^2 \tag{13}$$

$$= k^2+k^2+k^2 \tag{14}$$

$$= 3k^2 \tag{15}$$

Therefore, $\|\mathbf{R}\| = \sqrt{3}k$.

Now, let α be the angle between $\mathbf{R}$ and $\mathbf{A}$. using (9)

$$\cos\alpha = \frac{\mathbf{R}^\top \mathbf{A}}{\|\mathbf{R}\| \, \|\mathbf{A}\|} \tag{16}$$

$$= \frac{\left(\mathbf{A} + \mathbf{B} + \mathbf{C}\right)^\top \mathbf{A}}{\|\mathbf{R}\| \, \|\mathbf{A}\|} \tag{17}$$

$$= \frac{\mathbf{A}^\top \mathbf{A} + \mathbf{B}^\top \mathbf{A} + \mathbf{C}^\top \mathbf{A}}{\|\mathbf{R}\| \, \|\mathbf{A}\|} \tag{18}$$

$$= \frac{\|\mathbf{A}\|^2 + 0 + 0}{\|\mathbf{R}\| \, \|\mathbf{A}\|} \tag{19}$$

Theoretical Solution

$$= \frac{k^2}{(\sqrt{3}k)(k)} \tag{20}$$

$$= \frac{k^2}{\sqrt{3}k^2} \tag{21}$$

$$= \frac{1}{\sqrt{3}} \tag{22}$$

Let β be the angle between $\mathbf{R}$ and $\mathbf{B}$. using (9)

$$\cos\beta = \frac{\mathbf{R}^\top \mathbf{B}}{\|\mathbf{R}\| \, \|\mathbf{B}\|} \tag{23}$$

$$= \frac{\left(\mathbf{A} + \mathbf{B} + \mathbf{C}\right)^\top \mathbf{B}}{\|\mathbf{R}\| \, \|\mathbf{B}\|} \tag{24}$$

$$= \frac{\mathbf{A}^\top \mathbf{B} + \mathbf{B}^\top \mathbf{B} + \mathbf{C}^\top \mathbf{B}}{\|\mathbf{R}\| \, \|\mathbf{B}\|} \tag{25}$$

Theoretical Solution

$$= \frac{0 + \left\|\mathbf{B}\right\|^2 + 0}{\left\|\mathbf{R}\right\| \left\|\mathbf{B}\right\|} \tag{26}$$

$$= \frac{k^2}{(\sqrt{3}k)(k)} \tag{27}$$

$$= \frac{k^2}{\sqrt{3}k^2} \tag{28}$$

$$= \frac{1}{\sqrt{3}} \tag{29}$$

Theoretical Solution

Let γ be the angle between $\mathbf{R}$ and $\mathbf{C}$. using (9)

$$\cos\gamma = \frac{\mathbf{R}^\top \mathbf{C}}{\|\mathbf{R}\| \, \|\mathbf{C}\|} \tag{30}$$

$$= \frac{\left(\mathbf{A} + \mathbf{B} + \mathbf{C}\right)^\top \mathbf{C}}{\|\mathbf{R}\| \, \|\mathbf{C}\|} \tag{31}$$

$$= \frac{\mathbf{A}^\top \mathbf{C} + \mathbf{B}^\top \mathbf{C} + \mathbf{C}^\top \mathbf{C}}{\|\mathbf{R}\| \, \|\mathbf{C}\|} \tag{32}$$

Theoretical Solution

$$= \frac{0 + 0 + \|\mathbf{C}\|^2}{\|\mathbf{R}\| \, \|\mathbf{C}\|} \tag{33}$$

$$= \frac{k^2}{(\sqrt{3}k)(k)} \tag{34}$$

$$= \frac{k^2}{\sqrt{3}k^2} \tag{35}$$

$$= \frac{1}{\sqrt{3}} \tag{36}$$

Since $\cos\alpha = \cos\beta = \cos\gamma = \frac{1}{\sqrt{3}}$, it implies $\alpha = \beta = \gamma$.
Thus, $\mathbf{A} + \mathbf{B} + \mathbf{C}$ is equally inclined to $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.

C Code

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

// Function to calculate the dot product of two 3D vectors
double dot_product(double v1x, double v1y, double v1z,
                   double v2x, double v2y, double v2z) {
    return v1x * v2x + v1y * v2y + v1z * v2z;
}

// Function to calculate the magnitude of a 3D vector
double magnitude(double vx, double vy, double vz) {
    return sqrt(vx * vx + vy * vy + vz * vz);
}
```

C Code

```c
// Function to calculate the cosines of angles between (A+B+C)
    and A, B, C
// Arguments:
// ax, ay, az: Components of vector A
// bx, by, bz: Components of vector B
// cx, cy, cz: Components of vector C
// cos_angle_result: Pointer to an array to store the results
void calculate_angles_cosines(double ax, double ay, double az,
                              double bx, double by, double bz,
                              double cx, double cy, double cz,
                              double* cos_angle_result) {
   // Calculate the resultant vector R = A + B + C
   double rx = ax + bx + cx;
   double ry = ay + by + cy;
   double rz = az + bz + cz;
```

C Code

```c
    // Calculate magnitudes
    double mag_A = magnitude(ax, ay, az);
    double mag_B = magnitude(bx, by, bz);
    double mag_C = magnitude(cx, cy, cz);
    double mag_R = magnitude(rx, ry, rz);

    // Calculate dot products
    double dot_R_A = dot_product(rx, ry, rz, ax, ay, az);
    double dot_R_B = dot_product(rx, ry, rz, bx, by, bz);
    double dot_R_C = dot_product(rx, ry, rz, cx, cy, cz);

    cos_angle_result[0] = dot_R_A / (mag_R * mag_A);
    cos_angle_result[1] = dot_R_B / (mag_R * mag_B);
    cos_angle_result[2] = dot_R_C / (mag_R * mag_C);
}
```

Python Code through shared output

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

# Load the shared library
lib_angles = ctypes.CDLL(./code4.so)

# Define the argument types and return type for the C function
lib_angles.calculate_angles_cosines.argtypes = [
    ctypes.c_double, ctypes.c_double, ctypes.c_double, # A_x, A_y
        , A_z
    ctypes.c_double, ctypes.c_double, ctypes.c_double, # B_x, B_y
        , B_z
    ctypes.c_double, ctypes.c_double, ctypes.c_double, # C_x, C_y
        , C_z
    ctypes.POINTER(ctypes.c_double * 3) # Pointer to an array of
        3 doubles for results
]
```

Python Code through shared output

```
lib_angles.calculate_angles_cosines.restype = None
# Define mutually perpendicular vectors of equal magnitude
magnitude = 5.0
A = np.array([magnitude, 0.0, 0.0])
B = np.array([0.0, magnitude, 0.0])
C = np.array([0.0, 0.0, magnitude])

# Resultant vector R = A + B + C
R = A + B + C
# Create a C array to hold the three cosine results
cos_angles_c_array = (ctypes.c_double * 3)()

# Call the C function
lib_angles.calculate_angles_cosines(
    A[0], A[1], A[2],
    B[0], B[1], B[2],
    C[0], C[1], C[2],
    ctypes.byref(cos_angles_c_array)
)
```

Python Code through shared output

```python
# Retrieve the results from the C array
cos_theta_RA = cos_angles_c_array[0]
cos_theta_RB = cos_angles_c_array[1]
cos_theta_RC = cos_angles_c_array[2]

# Convert cosines to angles in degrees
theta_RA_deg = np.degrees(np.arccos(cos_theta_RA))
theta_RB_deg = np.degrees(np.arccos(cos_theta_RB))
theta_RC_deg = np.degrees(np.arccos(cos_theta_RC))

print(fVectors A = {A}, B = {B}, C = {C})
print(fResultant vector R = A + B + C = {R})
```

```
print(fCosine of angle between R and A: {cos_theta_RA:.6f})
print(fAngle between R and A: {theta_RA_deg:.2f} degrees)
print(fCosine of angle between R and B: {cos_theta_RB:.6f})
print(fAngle between R and B: {theta_RB_deg:.2f} degrees)
print(fCosine of angle between R and C: {cos_theta_RC:.6f})
print(fAngle between R and C: {theta_RC_deg:.2f} degrees)

if np.isclose(theta_RA_deg, theta_RB_deg) and np.isclose(
    theta_RB_deg, theta_RC_deg):
    print(\nConclusion: The angles are approximately equal,
        showing that A+B+C is equally inclined to A, B, and C.)
else:
    print(\nConclusion: The angles are not equal. There might be
        an issue with the input vectors or calculation.)
```

```python
# --- Visualization ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

origin = [0, 0, 0]

# Plot vectors A, B, C
ax.quiver(*origin, *A, color='r', linewidth=2, arrow_length_ratio
    =0.1, label='Vector A')
ax.quiver(*origin, *B, color='g', linewidth=2, arrow_length_ratio
    =0.1, label='Vector B')
ax.quiver(*origin, *C, color='b', linewidth=2, arrow_length_ratio
    =0.1, label='Vector C')

# Plot resultant vector R
ax.quiver(*origin, *R, color='purple', linewidth=3,
    arrow_length_ratio=0.08, label='Vector A+B+C')
```

Python Code through shared output

```
# Set labels and title
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Mutually Perpendicular Vectors and Their Sum')
ax.legend()

# Set limits for a better view
max_coord = max(np.max(np.abs(A)), np.max(np.abs(B)), np.max(np.
    abs(C)), np.max(np.abs(R))) * 1.2
ax.set_xlim([-max_coord, max_coord])
ax.set_ylim([-max_coord, max_coord])
ax.set_zlim([-max_coord, max_coord])

# Add grid
ax.grid(True)
plt.savefig(fig1.png) # Save the plot
plt.show()
```

Python Code

```
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # For 3D plotting

# --- 1. Define vectors A, B, C ---
# Mutually perpendicular vectors of equal magnitude
magnitude = 3.0 # You can change the magnitude
A = np.array([magnitude, 0.0, 0.0])
B = np.array([0.0, magnitude, 0.0])
C = np.array([0.0, 0.0, magnitude])

print(fGiven Vectors:)
print(fA = {A})
print(fB = {B})
print(fC = {C})
```

```python
# --- 2. Calculate the resultant vector R = A + B + C ---
R = A + B + C
print(f\nResultant vector R = A + B + C = {R})

# --- 3. Calculate angles using dot products ---
def angle_between_vectors(v1, v2):
    dot_product = np.dot(v1, v2)
    magnitude_v1 = LA.norm(v1)
    magnitude_v2 = LA.norm(v2)

    # Handle potential division by zero for zero vectors
    if magnitude_v1 < 1e-9 or magnitude_v2 < 1e-9:
        if magnitude_v1 < 1e-9 and magnitude_v2 < 1e-9:
            return 0.0 # Both are zero, angle is 0
        else:
            return np.pi / 2 # One is zero, other non-zero, angle is 90 degrees (pi/2 radians)
```

Python Code

```
    cosine_angle = dot_product / (magnitude_v1 * magnitude_v2)
    # Ensure cosine_angle is within [-1, 1] due to potential
        floating point inaccuracies
    cosine_angle = np.clip(cosine_angle, -1.0, 1.0)
    return np.arccos(cosine_angle) # Returns angle in radians

# Calculate angles
angle_RA_rad = angle_between_vectors(R, A)
angle_RB_rad = angle_between_vectors(R, B)
angle_RC_rad = angle_between_vectors(R, C)

angle_RA_deg = np.degrees(angle_RA_rad)
angle_RB_deg = np.degrees(angle_RB_rad)
angle_RC_deg = np.degrees(angle_RC_rad)

print(\n--- Calculated Angles ---)
print(fAngle between R and A: {angle_RA_deg:.2f} degrees)
print(fAngle between R and B: {angle_RB_deg:.2f} degrees)
print(fAngle between R and C: {angle_RC_deg:.2f} degrees)
```

Python Code

```
# Conclusion
if np.isclose(angle_RA_deg, angle_RB_deg) and np.isclose(
    angle_RB_deg, angle_RC_deg):
    print(\nConclusion: The angles are approximately equal. This
        shows that A+B+C is equally inclined to A, B, and C.)
else:
    print(\nConclusion: The angles are not equal. Please check
        the input vectors to ensure they are mutually
        perpendicular and have equal magnitudes.)

# --- 5. Generate a 3D plot to visualize these vectors ---
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

origin = [0, 0, 0]
```

Python Code

```
# Plot vectors A, B, C
ax.quiver(*origin, *A, color='r', linewidth=2, arrow_length_ratio
    =0.1, label='Vector A')
ax.quiver(*origin, *B, color='g', linewidth=2, arrow_length_ratio
    =0.1, label='Vector B')
ax.quiver(*origin, *C, color='b', linewidth=2, arrow_length_ratio
    =0.1, label='Vector C')

# Plot resultant vector R
ax.quiver(*origin, *R, color='purple', linewidth=3,
    arrow_length_ratio=0.08, label='Vector A+B+C')

# Set labels and title
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Mutually Perpendicular Vectors and Their Sum')
ax.legend()
```

Python Code (Direct) - Visualization

```
# Set limits for a better view
all_coords = np.concatenate((A, B, C, R))
max_coord = np.max(np.abs(all_coords)) * 1.2 # Add some padding
ax.set_xlim([-max_coord, max_coord])
ax.set_ylim([-max_coord, max_coord])
ax.set_zlim([-max_coord, max_coord])

# Add grid
ax.grid(True)

plt.savefig(fig2.png)
plt.show()

print(\nFigure saved as fig2.png)
```

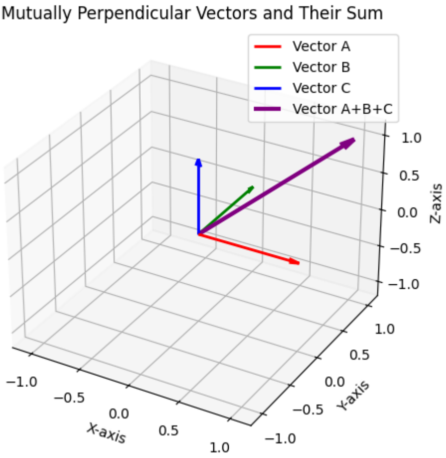
Mutually Perpendicular Vectors and Their Sum
Vector A
Vector B
Vector C
Vector A+B+C
X-axis
Y-axis
Z-axis
−1.0
−0.5
0.0
0.5
1.0

Figure:

Plot by Python only

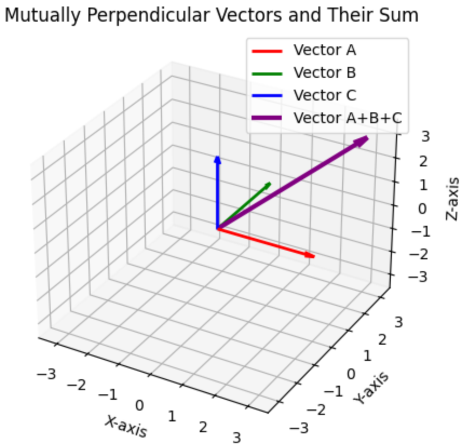

Figure: