Area of Parallelogram

EE25BTECH11008 - Anirudh M Abhilash

September 16, 2025

Problem Statement

The two adjacent sides of a parallelogram are

$$\mathbf{a} = \begin{pmatrix} 2 \\ -4 \\ -5 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 2 \\ 2 \\ 3 \end{pmatrix}$$

Find the two unit vectors parallel to its diagonals. Using the diagonal vectors, find the area of the parallelogram.

Solution

Diagonals:

$$\mathbf{d}_1 = \mathbf{a} + \mathbf{b} \tag{1}$$

$$\mathbf{d}_2 = \mathbf{a} - \mathbf{b} \tag{2}$$

$$\mathbf{d}_1 = \begin{pmatrix} 4 \\ -2 \\ -2 \end{pmatrix} \tag{3}$$

$$\mathbf{d}_2 = \begin{pmatrix} 0 \\ -6 \\ -8 \end{pmatrix} \tag{4}$$

Norms:

$$\|\mathbf{d}_1\|^2 = \mathbf{d}_1^\top \mathbf{d}_1 = 24 \tag{5}$$

$$\|\mathbf{d}_1\| = 2\sqrt{6} \tag{6}$$

$$\|\mathbf{d}_2\|^2 = \mathbf{d}_2^\top \mathbf{d}_2 = 100 \tag{7}$$

$$\|\mathbf{d}_2\| = 10 \tag{8}$$

Solution (cont..)

Unit vectors along diagonals:

$$\mathbf{u}_1 = \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|} = \frac{1}{2\sqrt{6}} \begin{pmatrix} 4 \\ -2 \\ -2 \end{pmatrix} \tag{9}$$

$$\mathbf{u}_2 = \frac{\mathbf{d}_2}{\|\mathbf{d}_2\|} = \frac{1}{10} \begin{pmatrix} 0 \\ -6 \\ -8 \end{pmatrix}. \tag{10}$$

Area of the parallelogram:

$$\|\mathbf{a} \times \mathbf{b}\|^2 = (\mathbf{a}^\top \mathbf{a})(\mathbf{b}^\top \mathbf{b}) - (\mathbf{a}^\top \mathbf{b})^2. \tag{11}$$

$$\mathbf{a}^\top \mathbf{a} = 45, \quad \mathbf{b}^\top \mathbf{b} = 17, \quad \mathbf{a}^\top \mathbf{b} = -19. \tag{12}$$

$$\|\mathbf{a} \times \mathbf{b}\|^2 = 45 \cdot 17 - (-19)^2 = 404, \tag{13}$$

$$\text{Area} = \sqrt{404} = 2\sqrt{101}. \tag{14}$$

Solution (cont..)

$$\mathbf{u}_1 = \frac{1}{2\sqrt{6}} \begin{pmatrix} 4 \\ -2 \\ -2 \end{pmatrix}, \quad \mathbf{u}_2 = \frac{1}{10} \begin{pmatrix} 0 \\ -6 \\ -8 \end{pmatrix}, \quad \text{Area} = 2\sqrt{101}$$

Python Code (Plotting Parallelogram)

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

A = np.array([2, -4, -5])
B = np.array([2, 2, 3])
O = np.array([0, 0, 0])
C = A + B
points = np.array([O, A, C, B])

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
verts = [[O, A, C, B]]
```

Python Code (cont..)

```
ax.add_collection3d(Poly3DCollection(verts, alpha=0.3, facecolor=
    'cyan'))
edges = [(O, A), (O, B), (A, C), (B, C)]
for p1, p2 in edges:
    ax.plot(*zip(p1, p2), color='blue')
ax.plot(*zip(O, C), color='red', linestyle='--')
ax.plot(*zip(A, B), color='green', linestyle='--')
labels = {"O": O, "A": A, "B": B, "C": C}
for name, coord in labels.items():
    ax.scatter(*coord, color='black', s=10)
    ax.text(coord[0], coord[1], coord[2], f'{name}{tuple(coord)}",
        fontsize=9)
```

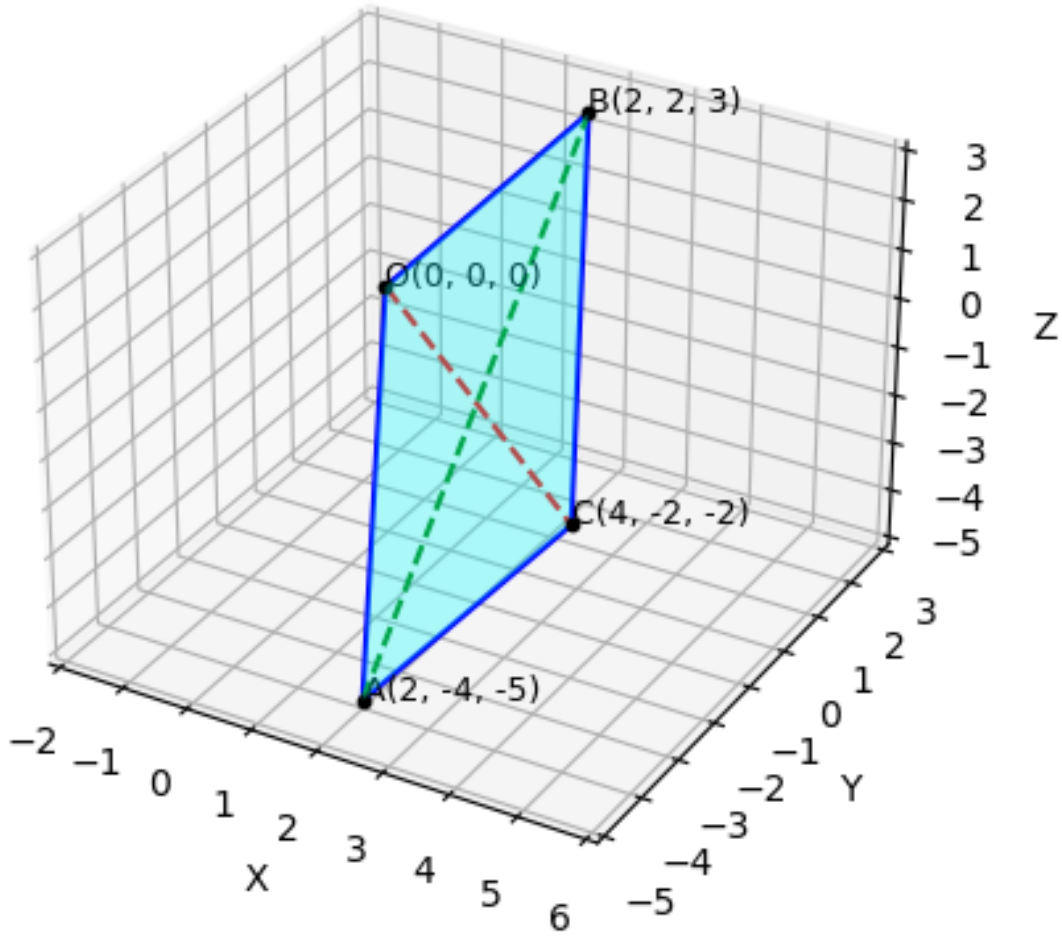
Python Code (cont..)

```
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

max_range = np.array([points[:,0].max()-points[:,0].min(),
                      points[:,1].max()-points[:,1].min(),
                      points[:,2].max()-points[:,2].min()]).max
                          () / 2.0
mid_x = (points[:,0].max()+points[:,0].min()) * 0.5
mid_y = (points[:,1].max()+points[:,1].min()) * 0.5
mid_z = (points[:,2].max()+points[:,2].min()) * 0.5
ax.set_xlim(mid_x - max_range, mid_x + max_range)
ax.set_ylim(mid_y - max_range, mid_y + max_range)
ax.set_zlim(mid_z - max_range, mid_z + max_range)

plt.show()
```

Plot



C Code (Computations)

```
#include <math.h>
void diagonals(double A[3], double B[3], double d1[3], double
    d2[3]) {
    d1[0] = A[0] - B[0];
    d1[1] = A[1] - B[1];
    d1[2] = A[2] - B[2];

    d2[0] = A[0] + B[0];
    d2[1] = A[1] + B[1];
    d2[2] = A[2] + B[2];
}
double dot(double a[3], double b[3]) {
    return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
}
```

C Code (Cont..)

```
void unit(double v[3], double u[3]) {
    double mag = sqrt(dot(v, v));
    u[0] = v[0]/mag;
    u[1] = v[1]/mag;
    u[2] = v[2]/mag;
}
double cross_via_dot(double a[3], double b[3]) {
    double mag_a2 = dot(a, a);
    double mag_b2 = dot(b, b);
    double dot_ab = dot(a, b);
    return sqrt(mag_a2 * mag_b2 - dot_ab * dot_ab);
}
```

Python Code (Calling C)

```
import ctypes

lib = ctypes.CDLL("./computations.so")
DoubleArray3 = ctypes.c_double * 3

lib.diagonals.argtypes = [DoubleArray3, DoubleArray3,
    DoubleArray3, DoubleArray3]
lib.unit.argtypes = [DoubleArray3, DoubleArray3]
lib.dot.argtypes = [DoubleArray3, DoubleArray3]
lib.dot.restype = ctypes.c_double
lib.cross_via_dot.argtypes = [DoubleArray3, DoubleArray3]
lib.cross_via_dot.restype = ctypes.c_double
```

Python Code (cont..)

```
A = DoubleArray3(2, -4, -5)
B = DoubleArray3(2, 2, 3)

d1, d2 = DoubleArray3(), DoubleArray3()
lib.diagonals(A, B, d1, d2)

u1, u2 = DoubleArray3(), DoubleArray3()
lib.unit(d1, u1)
lib.unit(d2, u2)

area = 0.5 * lib.cross_via_dot(d1, d2)

print("u1:", list(u1))
print("u2:", list(u2))
print("Area:", area)
```