4.7.62

EE25BTECH11065-Yoshita J

September 14,2025

Question

Find the equation of the plane which passes through the point (5, 2, -4) and perpendicular to the line with direction ratios 2, 3, -1.

Theoretical Solution

The plane passes through the point

$$\mathbf{A} = \begin{pmatrix} 5 \\ 2 \\ -4 \end{pmatrix}$$

with normal vector

$$\mathbf{n} = \begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}.$$

The equation of the plane can be written as

$$\mathbf{n}^T(\mathbf{x} - \mathbf{A}) = 0.$$

Equivalently,

$$\mathbf{n}^T\mathbf{x} = \mathbf{n}^T\mathbf{A}.$$

Theoretical Solution

Substituting the values,

$$\begin{pmatrix} 2 & 3 & -1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 2 & 3 & -1 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -4 \end{pmatrix} \tag{1}$$

$$\implies \begin{pmatrix} 2 & 3 & -1 \end{pmatrix} \mathbf{x} = 20. \tag{2}$$

Hence, the equation of the plane is

$$\mathbf{n}^T \mathbf{x} = 20,$$

where

$$\mathbf{n} = \begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}.$$

C Code

```c
#include<stdio.h>
typedef struct {
    double x, y, z;
} Vector;

typedef struct {
    double a, b, c, d;
} Plane;

Plane find_plane_from_point_and_normal(Vector point, Vector
    normal) {
    Plane result;
    result.a = normal.x;
    result.b = normal.y;
    result.c = normal.z;
    result.d = (normal.x * point.x) + (normal.y * point.y) + (
        normal.z * point.z);
    return result;
}
```

Python Code

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

point_A = np.array([5.0, 2.0, -4.0])
normal_n = np.array([2.0, 3.0, -1.0])

d = np.dot(normal_n, point_A)
plane_eq_str = f{normal_n[0]:.0f}x + {normal_n[1]:.0f}y + {
    normal_n[2]:.0f}z = {d:.0f}
print(fPlane equation: {plane_eq_str})

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

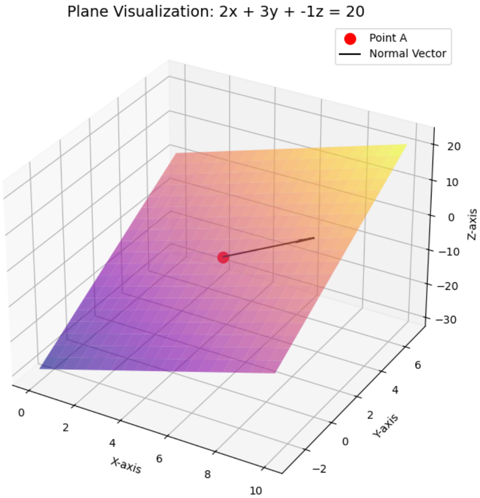
Python Code

```
x_range = np.linspace(point_A[0] - 5, point_A[0] + 5, 20)
y_range = np.linspace(point_A[1] - 5, point_A[1] + 5, 20)
x_grid, y_grid = np.meshgrid(x_range, y_range)

z_plane = (d - normal_n[0] * x_grid - normal_n[1] * y_grid) /
    normal_n[2]

ax.plot_surface(x_grid, y_grid, z_plane, alpha=0.6, cmap='plasma'
    , edgecolor='none')

ax.scatter([point_A[0]], [point_A[1]], [point_A[2]], color='red',
     s=100, label=f'Point A {tuple(point_A)}')
```

Python Code

```python
ax.quiver(
    point_A[0], point_A[1], point_A[2],
    normal_n[0], normal_n[1], normal_n[2],
    length=4, normalize=True, color='black', arrow_length_ratio
        =0.2,
    label=f'Normal Vector n={tuple(normal_n)}'
)

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title(f'Plane Visualization: {plane_eq_str}', fontsize=14)
ax.legend()
plt.grid(True)
plt.show()
```

Plot


Plane Visualization: 2x + 3y + -1z = 20
● Point A
Normal Vector
0
2
4
6
8
10
X-axis
−2
0
2
4
6
Y-axis
20
10
0
−10
−20
−30
Z-axis