4.7.15

Jnanesh Sathisha karmar - EE25BTECH11029

September 28,2025

Question

Find the vector equation of the plane which is at a distance of $\frac{6}{\sqrt{29}}$ from the origin and its normal vector from the origin is $2\hat{i} - 3\hat{j} + 4\hat{k}$..

Equation

Given details

$$\text{distance of plane from the origin} = d = \frac{6}{\sqrt{29}} \tag{1}$$

$$\text{normal vector} = \mathbf{n} = \begin{pmatrix} 2 \\ -3 \\ 4 \end{pmatrix} \tag{2}$$

Theoretical Solution

Generally a plane can be represented as:

$$\hat{\mathbf{n}}^{\top}\left(\mathbf{r}-\mathbf{r_o}\right)=0 \tag{3}$$

For our convience we can choose $\mathbf{r_o}$ to be the point closest to origin.Therefore:

$$\mathbf{r_o}=d\hat{\mathbf{n}} \tag{4}$$

Theoretical Solution

Substituting this in the plane equation:

$$\hat{\mathbf{n}}^\top (\mathbf{r} - d\hat{\mathbf{n}}) = 0 \tag{5}$$

$$\left(\hat{\mathbf{n}}^\top \mathbf{r}\right) - \left(d\hat{\mathbf{n}}^\top \hat{\mathbf{n}}\right) = 0 \tag{6}$$

$$\because \quad \hat{\mathbf{n}}^\top \hat{\mathbf{n}} = 1 \tag{7}$$

$$\hat{\mathbf{n}}^\top \mathbf{r} = d \tag{8}$$

Substituting it in the plane equation:

$$\frac{1}{\sqrt{29}} \begin{pmatrix} 2 & -3 & 4 \end{pmatrix}^\top \mathbf{r} = \frac{6}{\sqrt{29}} \tag{9}$$

The final plane equation is:

$$\begin{pmatrix} 2 & -3 & 4 \end{pmatrix}^\top \mathbf{r} = 6 \tag{10}$$

C Code (1) - Function to store the points

```c
#include <stdlib.h>


float* generate_plane_points(float x_min, float x_max, float
    y_min, float y_max, int num_steps) {
    if (num_steps <= 1) {
        return NULL;
    }

    int total_points = num_steps * num_steps;
    float* points = (float*)malloc(total_points * 3 * sizeof(
        float));
    if (points == NULL) {
        return NULL;
    }
```

C Code (1) - Function to store the points

```
float x_step_size = (x_max - x_min) / (num_steps - 1);
float y_step_size = (y_max - y_min) / (num_steps - 1);

int index = 0;

for (int i = 0; i < num_steps; i++) {
    float x = x_min + i * x_step_size;

    for (int j = 0; j < num_steps; j++) {
        float y = y_min + j * y_step_size;


        float z = (3.0f/2.0f) - (1.0f/2.0f) * x + (3.0f / 4.0f
            ) * y;
```

C Code (1) - Function to store the points

```
            points[index++] = x;
            points[index++] = y;
            points[index++] = z;
        }
    }

    return points;
}


void free_points(float* points) {
    if (points != NULL) {
        free(points);
    }
}
```

Python Code - Using Shared Object

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

lib = ctypes.CDLL("./plane.so")

lib.generate_plane_points.argtypes = [
    ctypes.c_float, ctypes.c_float,
    ctypes.c_float, ctypes.c_float,
    ctypes.c_int
]
lib.generate_plane_points.restype = ctypes.POINTER(ctypes.c_float
    )

lib.free_points.argtypes = [ctypes.POINTER(ctypes.c_float)]
lib.free_points.restype = None
NUM_STEPS = 50
total_points = NUM_STEPS * NUM_STEPS
points_ptr = None
```

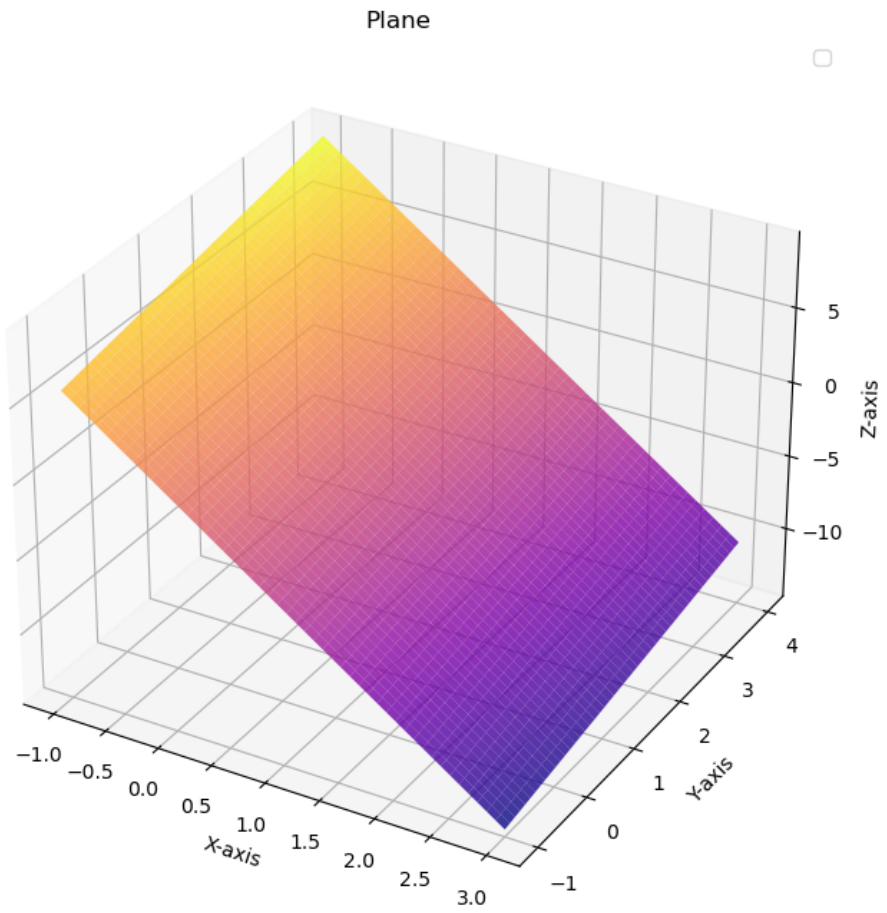
Python Code - Using Shared Object

```
try:
    points_ptr = lib.generate_plane_points(-1.0, 3.0,-1.0, 4.0,
        NUM_STEPS)
    if not points_ptr:
        raise MemoryError("C function failed to allocate memory."
            )

    points_np = np.ctypeslib.as_array(points_ptr, shape=(
        total_points, 3))
    points_data = np.copy(points_np)
finally:
    if points_ptr:
        lib.free_points(points_ptr)
X = points_data[:, 0].reshape(NUM_STEPS, NUM_STEPS)
Y = points_data[:, 1].reshape(NUM_STEPS, NUM_STEPS)
Z = points_data[:, 2].reshape(NUM_STEPS, NUM_STEPS)
```

Python Code - Using Shared Object

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


ax.plot_surface(X, Y, Z, cmap='plasma', alpha=0.8)
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Simplified Plane Generation')
ax.legend()
plt.savefig("./figs/plane.png")
subprocess.run(shlex.split('termux-open ../figs/parallelogram.png
    '))
plt.show()
```

Plot-Using Both C and Python



Python Code

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

def plot_plane_from_points():
    x_range = np.linspace(-1, 4, 20)
    y_range = np.linspace(-1, 4, 20)
    X, Y = np.meshgrid(x_range, y_range)

    Z = (3/2) - (1/2) * X + (3 / 4) * Y

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

Python Code

```
    ax.plot_surface(X, Y, Z, alpha=0.7, cmap='plasma', edgecolor=
        'none')

    ax.scatter(p1[0], p1[1], p1[2], color='red', s=120, label='
        (2,0,0)', depthshade=False)
    ax.scatter(p2[0], p2[1], p2[2], color='red', s=120, label='
        (0,3,0)', depthshade=False)
    ax.scatter(p3[0], p3[1], p3[2], color='red', s=120, label='
        (0,0,4)', depthshade=False)
```

Python Code

```
    ax.set_xlabel('X-axis')
    ax.set_ylabel('Y-axis')
    ax.set_zlabel('Z-axis')
    ax.set_title('Plane Passing Through Three Points')
    ax.legend()
    ax.set_xlim([-1, 4])
    ax.set_ylim([-1, 4])
    ax.set_zlim([-1, 6])
    plt.savefig("./figs/plane2.png")
    plt.show()
    subprocess.run(shlex.split('termux-open ../figs/parallelogram
        .png'))

if __name__ == '__main__':
    plot_plane_from_points()
```

Plot-Using only Python