Matgeo Presentation - Problem 4.7.53

ee25btech11056 - Suraj.N

September 8, 2025

Problem Statement

If $\mathbf{O}$ is the origin and $\mathbf{P} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$, then find the equation of the plane passing through $\mathbf{P}$ and perpendicular to OP.

Data

Description	**Value**
Normal vector	$\mathbf{n} = \mathbf{P} - \mathbf{O} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$
Point on plane	$\mathbf{h} = \mathbf{P} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$

Table : Plane

Solution

The normal vector to the plane is

$$\mathbf{n} = \mathbf{P} - \mathbf{O} = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix} \tag{0.1}$$

The plane equation is written in the form

$$\mathbf{n}^\top \mathbf{x} = \mathbf{n}^\top \mathbf{h} \tag{0.2}$$

where $\mathbf{h}$ is a point on the plane. Here $\mathbf{h} = \mathbf{P}$

$$\mathbf{n}^\top \mathbf{x} = \begin{pmatrix} 1 & 2 & -3 \end{pmatrix} \mathbf{P} \tag{0.3}$$

$$\mathbf{n}^\top \mathbf{x} = \begin{pmatrix} 1 & 2 & -3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix} \tag{0.4}$$

Hence, the equation of the plane is

$$\mathbf{n}^\top \mathbf{x} = 14 \tag{0.5}$$

$$\begin{pmatrix} 1 & 2 & -3 \end{pmatrix} \mathbf{x} = 14 \tag{0.6}$$

Plot

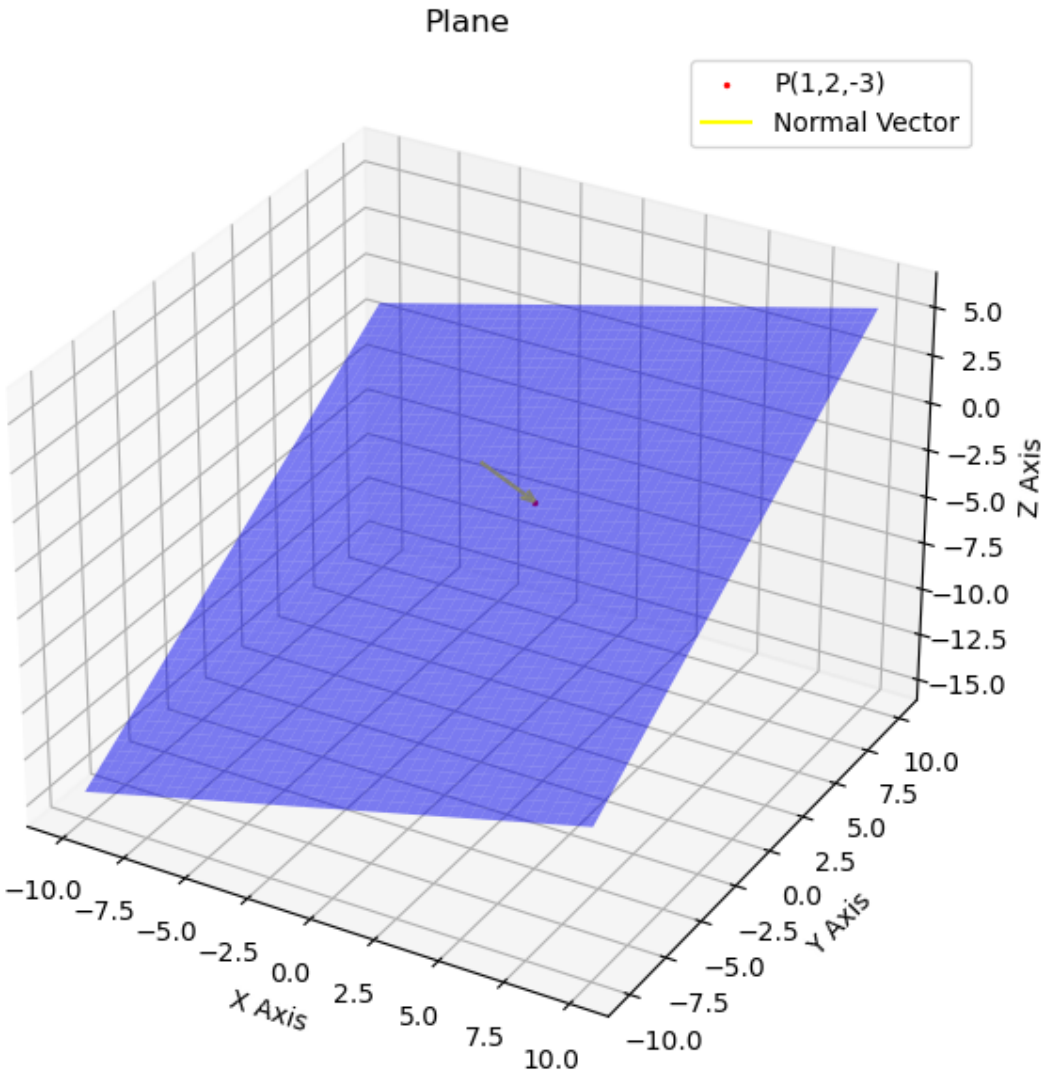

Fig : Plane

C Code: points.c

```
#include <stdio.h>
#include <stdlib.h>

double *normal(double p[]) {

  double o[3] = {0, 0, 0};

  double *n = malloc(3 * sizeof(double));

  if (n == NULL)
    return NULL;

  n[0] = p[0] - o[0];
  n[1] = p[1] - o[1];
  n[2] = p[2] - o[2];

  return n;
}
```

Python: call_c.py

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

#for generating figure in figs folder
figs_folder= os.path.join("..","figs")

#loading shared object
lib = ctypes.CDLL("./points.so")

#tell ctypes about the function signature
lib.normal.restype = ctypes.POINTER(ctypes.c_double) #returns pointer to double
lib.normal.argtypes = [ctypes.POINTER(ctypes.c_double)] #takes pointer to array

#defining point P as a list(array) to pass it as input to c function
P = (ctypes.c_double * 3)(1,2,-3)

#calling c function
n_ptr = lib.normal(P) #storing the return value in n_ptr that points to memory location of [1,2,-3]

#extracting values into a python list using for loop
n = [n_ptr[i] for i in range(3)]

#conveting to numpy for plotting

n = np.array(n)
point = np.array([1,2,-3]) #point P that lies on the plane
```

Python: call_c.py

```
#Plane equation
x = np.linspace(-10,10,100)
y = np.linspace(-10,10,100)

x,y = np.meshgrid(x,y)

d = n[0]*point[0] + n[1]*point[1] + n[2]*point[2]

z = (d - n[0]*x -n[1]*y)/n[2]

#plotting

fig = plt.figure(figsize=(8,6))

ax = fig.add_subplot(111,projection="3d")

ax.plot_surface(x,y,z,alpha=0.5,color='blue',edgecolor='none')

ax.scatter(*point,color='red',s=2,label="P(1,2,-3)")


#plotting normal vector OP
ax.quiver(0,0,0,n[0],n[1],n[2],color='yellow', arrow_length_ratio=0.2,label="Normal␣Vector")

ax.set_xlabel("X␣Axis")
ax.set_ylabel("Y␣Axis")
ax.set_zlabel("Z␣Axis")
ax.set_title("Plane")
ax.legend()
ax.grid(True)
```

Python: call_c.py

```
plt.tight_layout()

#saving the figure in figs folder
fig.savefig(os.path.join(figs_folder,"plane.png"))

plt.show()
```

Python: plot.py

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

#for generating figure in figs folder
figs_folder= os.path.join("..","figs")

#normal vector (same as returned by C code)
n = np.array([1,2,-3])
point = np.array([1,2,-3]) #point P that lies on the plane

#Plane equation
x = np.linspace(-10,10,100)
y = np.linspace(-10,10,100)

x,y = np.meshgrid(x,y)


d = n[0]*point[0] + n[1]*point[1] + n[2]*point[2]

z = (d - n[0]*x -n[1]*y)/n[2]

#plotting

fig = plt.figure(figsize=(8,6))

ax = fig.add_subplot(111,projection="3d")

ax.plot_surface(x,y,z,alpha=0.5,color='blue',edgecolor='none')

ax.scatter(*point,color='red',s=2,label="P(1,2,-3)")
```

Python: plot.py

```
#plotting normal vector OP
ax.quiver(0,0,0,n[0],n[1],n[2],color='yellow', arrow_length_ratio=0.2,label="Normal Vector")

ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")
ax.set_zlabel("Z Axis")
ax.set_title("Plane")
ax.legend()
ax.grid(True)

plt.tight_layout()

#saving the figure in figs folder
fig.savefig(os.path.join(figs_folder,"plane.png"))

plt.show()
```