Visualizing Planes To A Normal Vector

The Equation for a Plane in 3D Space - $\mathbb{R}^3$

$$\vec{A} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = a_1 x + a_2 y + a_3 z = d$$

where $\vec{A}$ is normal to the plane! Remember that this equation holds for all values of x, y, and z.

Now, IF you do NOT know A, but you know 3 points in the plane, you can solve for A this way ...

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} d \\ d \\ d \end{bmatrix}$$

We need at least 3 points in $\mathbb{R}^3$ to develop this equation for a plane. Each row in the left matrix represents a point. Once we have solved for $\vec{A}$ given a value for d, we can create a linearly spaced grid of points in x and y and solve for z.

$$z = (d - a_1 x - a_2 y)/a_3$$

NOTE that you could also solve for x using a grid of y and z values, or for y using a grid of x and z values.

Visualizing One Plane in 3D

Let's start with a simple and obvious equation of a plane problem.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

which yields,

$$\vec{A} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Now we create our grid of points in the x - y plane, and our equation for z simplifies to

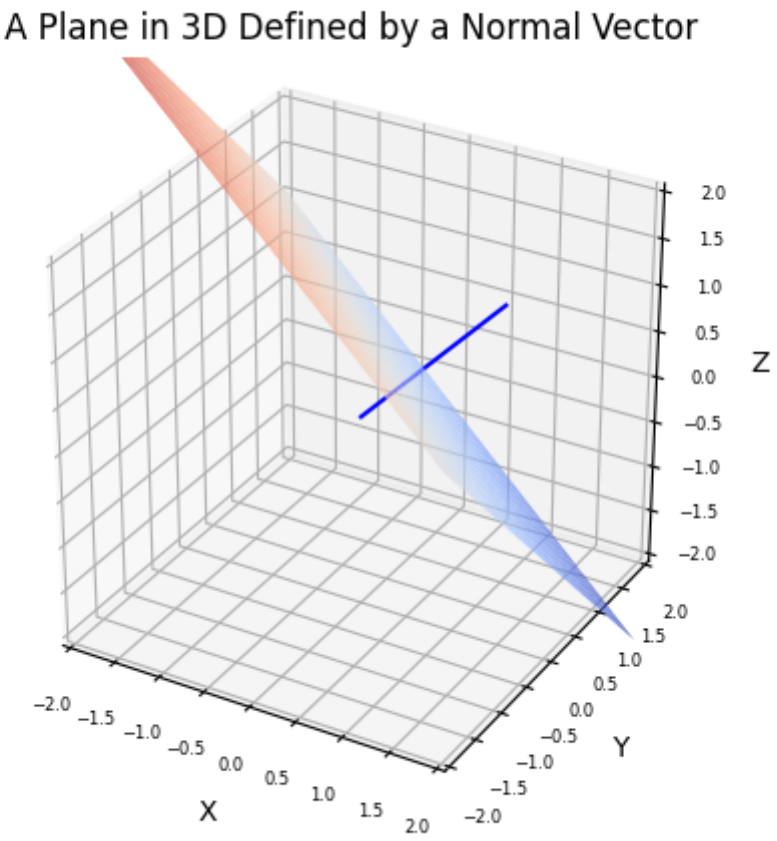
$$z = (1 - x - y)$$

Let's plot this in 3D.

First, the code that we need. The code below is in my Linear Algebra In Python DagsHub Repo. It's in the `MatPlotLib` subdirectory and is named `A_Plane_Normal_To_Vector.py` . I hope you will clone it, study it carefully, and seek to understand it line by line.

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xy = np.mgrid[-2:2:21j, -2:2:21j]
z = np.zeros((21, 21))
x = xy[0]; y = xy[1]


d = 1.0
z = d - x - y
ax.plot_surface(x, y, z, cmap=cm.coolwarm, linewidth=0, alpha=.7)


a = [0, 1]; b = [0, 1]; c = [0, 1]
ax.plot(a, b, c, color="blue")  # Normal vector - why a, b, and c 2D?


ax.set_title('A Plane in 3D Defined by a Normal Vector')
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
ax.set_xlim(-2, 2); ax.set_ylim(-2, 2); ax.set_zlim(-2, 2)


# Trying plotting with and without the next line to appreciate
#     what it is doing for us. It doesn't work in colab
ax.set_box_aspect((np.ptp(x), np.ptp(y), np.ptp(x)))

ax.tick_params(axis='both', which='major', labelsize=6)


plt.show()
```

Let's view the plot generated by this code.



I hope you run this on your computer so that you can rotate the plot and view it from different angles.

Visualizing Multiple Planes Normal To A Vector

Let's keep that same normal vector of

$$\vec{A} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

But now, we will create several planes that are normal to this vector. How? We will simply change the value for d to create a new planes that are parallel to our original plane. Thus, the equation from above

$$z = (1 - x - y)$$

becomes

$$z = (d - x - y)$$

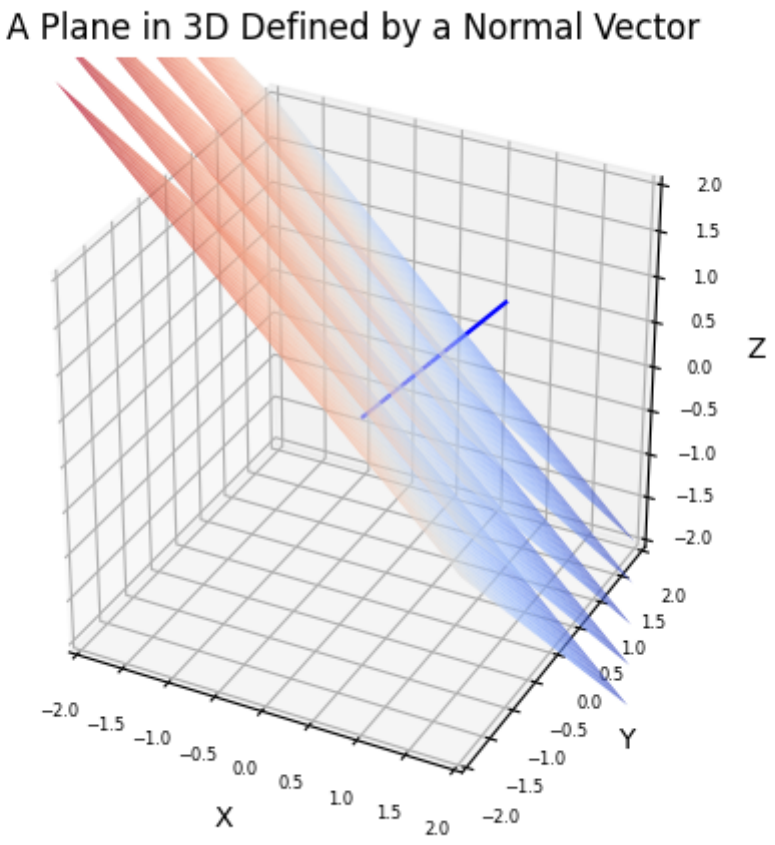
Let's plot planes using this equation in 3D for several values of d. The code below is in the same subdirectory of the same repo, but it is stored in the file named `Multi_Planes_Along_Normal.py` .

Please note the following.

1. This is VERY similar to the previous file.
2. We are careful to only run the solutions for `z` and `ax.plot_surface` multiple times. These are the only things we need to repeat for each value of d to create planes parallel to our original plane and have all planes normal to our normal vector $\vec{A}$.

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xy = np.mgrid[-2:2:21j, -2:2:21j]
z = np.zeros((21, 21))
x = xy[0]; y = xy[1]


for d in (0.0, 0.5, 1, 1.5, 2.0):
    z = d - x - y
    ax.plot_surface(x, y, z, cmap=cm.coolwarm, linewidth=0, alpha=.7)

a = [0, 1]; b = [0, 1]; c = [0, 1]
ax.plot(a, b, c, color="blue")


ax.set_title('A Plane in 3D Defined by a Normal Vector')
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
ax.set_xlim(-2, 2); ax.set_ylim(-2, 2); ax.set_zlim(-2, 2)
ax.set_box_aspect((np.ptp(x), np.ptp(y), np.ptp(x)))
ax.tick_params(axis='both', which='major', labelsize=6)


plt.show()
```

Let's view the plot produced by this code.



Again, I hope you run this on your computer so that you can rotate the plot and view it from different angles.

What's left for you to do now? Play! Try different normal vectors and start to leverage this code to experiment off into new directions.

Enjoy!

Thom Ives