MatGeo Presentation - Problem 1.11.1

EE25BTECH11064 - Yojit Manral

Question

Find a vector $\mathbf{r}$ that is equally inclined to the three axes and whose magnitude is $3\sqrt{3}$ units.

Solution

$\rightarrow$ A vector that subtends equal angles to all three axes will have equal components. Let the scaling factor be c. Then,

$$\text{Given that, } \|\mathbf{r}\| = 3\sqrt{3} \text{ and } \mathbf{r} = c \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{0.1}$$

$$\implies \|\mathbf{r}\| = |c| \, \| \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \| \tag{0.2}$$

$$\implies \|\mathbf{r}\| = |c| \sqrt{3} \tag{0.3}$$

$$\implies 3\sqrt{3} = |c| \sqrt{3} \tag{0.4}$$

$$\implies |c| = 3 \tag{0.5}$$

$$\implies \mathbf{r} = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix} \text{ or } \mathbf{r} = \begin{pmatrix} -3 \\ -3 \\ -3 \end{pmatrix} \tag{0.6}$$

Plot

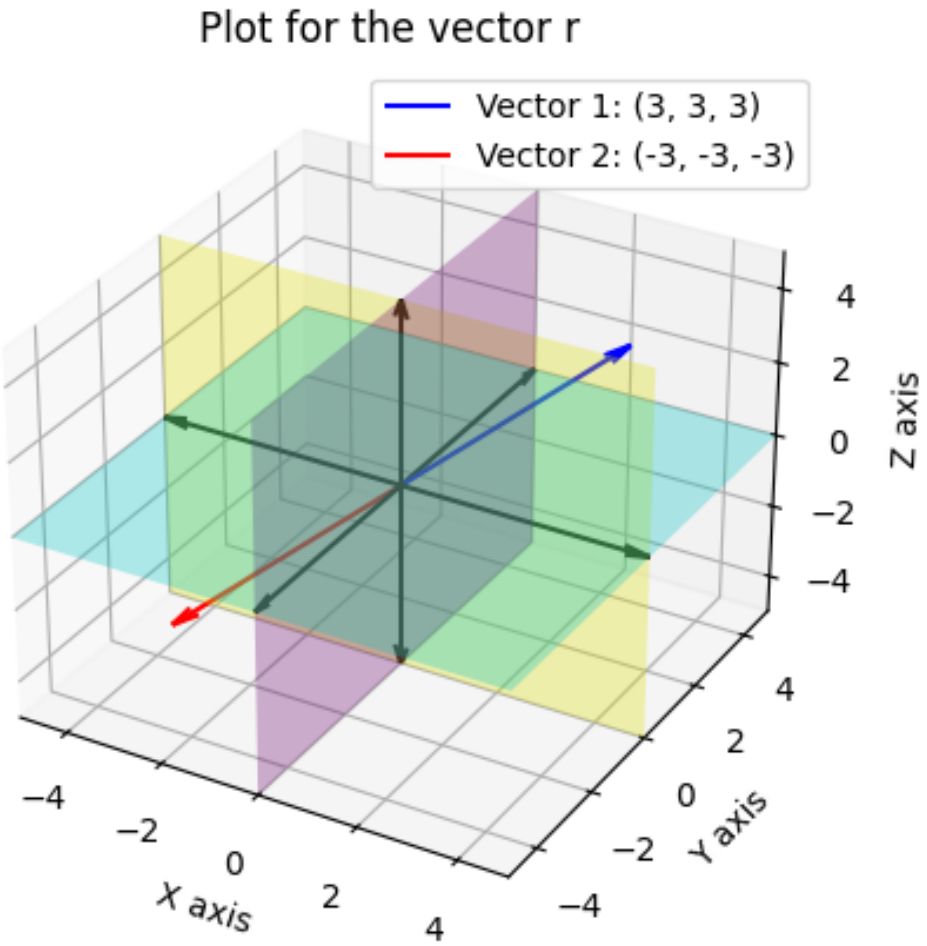

Figure: Plot of the vector **r**

File: points.c

```
#include <stdio.h>

int main() {
  FILE *fp;

  // -------------------
  // Question 1.11.1
  // -------------------


  fp = fopen("points.dat", "w");
  fprintf(fp, "%d,%d,%d\n", 3, 3, 3); // 1
  fprintf(fp, "%d,%d,%d\n", -3, -3, -3); // 2
  fclose(fp);
  return 0;
  }
```

File: call_c.py

```
import subprocess

# Compile the C program
subprocess.run(["gcc", "points.c", "-o", "points"])

# Run the compiled C program
result = subprocess.run(["./points"], capture_output=True, text=True)

# Print the output from the C program
print(result.stdout)
```

File: plot.py

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

# Define the points directly
point1 = np.array([3, 3, 3], dtype=float) # First point (3, 3, 3)
point2 = np.array([-3, -3, -3], dtype=float) # Second point (-3, -3, -3)

# Create the figure and 3D axis
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Calculate vector components for the two paths
vector1 = point1 # Vector from origin to (3, 3, 3)
vector2 = point2 # Vector from origin to (-3, -3, -3)

# Define unit vectors for the x, y, and z axes
x_axis = np.array([1, 0, 0], dtype=float)
y_axis = np.array([0, 1, 0], dtype=float)
z_axis = np.array([0, 0, 1], dtype=float)

# Plot the vectors as arrows using quiver
# Vector 1
ax.quiver(0, 0, 0, vector1[0], vector1[1], vector1[2],
          color='b', label='Vector␣1:␣(3,␣3,␣3)', arrow_length_ratio=0.1)

# Vector 2
ax.quiver(0, 0, 0, vector2[0], vector2[1], vector2[2],
          color='r', label='Vector␣2:␣(-3,␣-3,␣-3)', arrow_length_ratio=0.1)
```

File: plot.py

```
# Draw the x, y, z axes
ax.quiver(0, 0, 0, 5, 0, 0, color='black', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, 0, 5, 0, color='black', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, 0, 0, 5, color='black', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, -5, 0, 0, color='black', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, 0, -5, 0, color='black', arrow_length_ratio=0.1)
ax.quiver(0, 0, 0, 0, 0, -5, color='black', arrow_length_ratio=0.1)

# Plot the XY, YZ, and ZX planes with transparency
# Define the grid
scale = 5
xx, yy = np.meshgrid(np.linspace(-scale, scale, 10),
                     np.linspace(-scale, scale, 10))
zz = np.zeros_like(xx)
ax.plot_surface(xx, yy, zz, color='cyan', alpha=0.3, rstride=100, cstride=100)

yy, zz = np.meshgrid(np.linspace(-scale, scale, 10),
                     np.linspace(-scale, scale, 10))
xx = np.zeros_like(yy)
ax.plot_surface(xx, yy, zz, color='magenta', alpha=0.3, rstride=100, cstride=100)

xx, zz = np.meshgrid(np.linspace(-scale, scale, 10),
                     np.linspace(-scale, scale, 10))
yy = np.zeros_like(xx)
ax.plot_surface(xx, yy, zz, color='yellow', alpha=0.3, rstride=100, cstride=100)

origin = np.array([0, 0, 0])
```

File: plot.py

```
# Set the limits of the plot
ax.set_xlim([-scale, scale])
ax.set_ylim([-scale, scale])
ax.set_zlim([-scale, scale])

# Add labels for the axes
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')

# Add a title
ax.set_title('Plot for the vector r')

# Show the label for the vector
ax.legend()

# Display the plot
plt.show()
```