Matgeo Presentation - Problem 1.10.31

ee25btech11063 - Vejith

August 28, 2025

Question

A vector **r** has magnitude 14 and direction ratios 2,3,-6.Find the direction cosines and components of **r**, given that **r** makes an acute angle with X axis

Description

Solution:

Symbol	Description
$\mathbf{r}$	given vector with magnitude=14
$\mathbf{r}_X$	component of r along X axis
$\mathbf{r}_Y$	component of r along Y axis
$\mathbf{r}_Z$	component of r along Z axis
k	scaling factor

Table: Variables Used

Solution

$$\mathbf{r} = k \begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix} \tag{0.1}$$

$$\|\mathbf{r}\| = |k| \, \| \begin{pmatrix} 2 \\ 3 \\ -6 \end{pmatrix} \| \tag{0.2}$$

$$\|\mathbf{r}\| = |k| \, 7 \tag{0.3}$$

$$14 = |k| \, 7 \tag{0.4}$$

$$|k| = 2 \tag{0.5}$$

$$\implies \mathbf{r} = \begin{pmatrix} 4 \\ 6 \\ -12 \end{pmatrix} \tag{0.6}$$

(but k=2 not -2 because given that vector r makes acute angle with X axis)

conclusion

The unit vector in the direction of **r** is

$$\frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{1}{14}\begin{pmatrix} 4 \\ 6 \\ -12 \end{pmatrix} = \begin{pmatrix} \frac{2}{7} \\ \frac{3}{7} \\ \frac{-6}{7} \end{pmatrix} \tag{0.7}$$

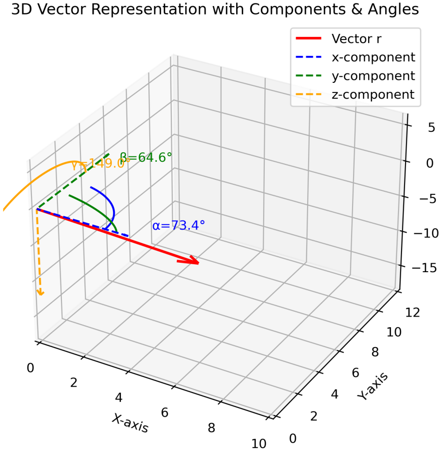
$$\tag{0.8}$$

The component of **r** along X axis $=\mathbf{r}_X = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}$

The component of **r** along Y axis $=\mathbf{r}_Y = \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix}$

The component of **r** along Z axis $=\mathbf{r}_Z = \begin{pmatrix} 0 \\ 0 \\ -12 \end{pmatrix}$

Plot



3D Vector Representation with Components & Angles
Vector r
x-component
y-component
z-component
β=64.6°
γ=149.0°
α=73.4°
X-axis
Y-axis
0
2
4
6
8
10
12
5
−5
−10
−15

Figure

C Code: code.c

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

int main() {
    FILE *fp;
    double magnitude, l, m, n;
    double dr1, dr2, dr3; // direction ratios
    double k; // normalization factor
    double comp_x, comp_y, comp_z;

    // Open file vector.dat
    fp = fopen("vector.dat", "r");
    if (fp == NULL) {
        printf("Error!␣Could␣not␣open␣file.\n");
        return 1;
    }

    // Reading magnitude and direction ratios
    fscanf(fp, "%lf␣%lf␣%lf␣%lf", &magnitude, &dr1, &dr2, &dr3);
    fclose(fp);

    // Calculate normalization factor
    k = sqrt(dr1*dr1 + dr2*dr2 + dr3*dr3);

    // Direction cosines
    l = dr1 / k;
    m = dr2 / k;
    n = dr3 / k;

    // Verify that angle with X-axis is acute
    if (l <= 0) {
        printf("Error:␣The␣vector␣does␣not␣make␣an␣acute␣angle␣with␣the␣X-axis.\n");
        return 1;}
```

C Code: code.c

```
    // Components of vector
    comp_x = magnitude * l;
    comp_y = magnitude * m;
    comp_z = magnitude * n;

    // Display results
    printf("Direction Cosines: l = %.3f, m = %.3f, n = %.3f\n", l, m, n);
    printf("Components of vector r: (%.3f, %.3f, %.3f)\n", comp_x, comp_y, comp_z);
 return 0;
}
```

Python: call.py

```
import subprocess

# Step 1: Create vector.dat automatically
with open("vector.dat", "w") as f:
    f.write("14␣2␣3␣-6\n") # You can change values here if needed

# Step 2: Compile the C code with -lm (math library)
compile_result = subprocess.run(["gcc", "code.c", "-o", "vector", "-lm"], capture_output=True, text=True)

if compile_result.returncode != 0:
    print("Compilation␣failed:\n", compile_result.stderr)
else:
    print("Compilation␣successful.␣Running␣program...\n")
    # Step 3: Run the compiled program
    run_result = subprocess.run(["./vector"], capture_output=True, text=True)

    # Step 4: Print program output (results or error from C code)
    if run_result.stdout.strip():
        print(run_result.stdout.strip())
    if run_result.stderr.strip():
        print("Runtime␣error:\n", run_result.stderr.strip())
```

Python: plot.py

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

# ----------------------------
# Read magnitude and direction ratios from file
# vector.dat should contain: 14 2 3 -6
# ----------------------------
data = np.loadtxt("vector.dat")
magnitude, a, b, c = data

# Normalize direction ratios  direction cosines
d = np.sqrt(a**2 + b**2 + c**2)
l, m, n = a/d, b/d, c/d

# Vector components (scaled by magnitude)
x, y, z = magnitude * np.array([l, m, n])

# Angles with axes
alpha = np.degrees(np.arccos(l)) # angle with x-axis
beta = np.degrees(np.arccos(m)) # angle with y-axis
gamma = np.degrees(np.arccos(n)) # angle with z-axis

# ----------------------------
# Verify condition: r makes acute angle with x-axis
# ----------------------------
if alpha >= 90:
    print(f"Vector␣r␣does␣NOT␣make␣an␣acute␣angle␣with␣x-axis␣(␣=␣{alpha:.2f}).␣No␣figure␣generated.")
else:
    print(f"Vector␣r␣makes␣an␣acute␣angle␣with␣x-axis␣(␣=␣{alpha:.2f}).␣Generating␣figure...")

    # ----------------------------
    # Plot setup (one figure only)
```

Python: plot.py

```python
# ----------------------------
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the main vector
ax.quiver(0, 0, 0, x, y, z, color='r', arrow_length_ratio=0.1, linewidth=2, label="Vector r")

# Plot projections on axes
ax.quiver(0, 0, 0, x, 0, 0, color='b', linestyle='dashed', arrow_length_ratio=0.05, label="x-component"
      )
ax.quiver(0, 0, 0, 0, y, 0, color='g', linestyle='dashed', arrow_length_ratio=0.05, label="y-component"
      )
ax.quiver(0, 0, 0, 0, 0, z, color='orange', linestyle='dashed', arrow_length_ratio=0.05, label="z-
      component")

# Function to draw arc in 3D
def plot_arc(ax, radius, angle, axis='x', color='k'):
    t = np.linspace(0, np.radians(angle), 50)
    if axis == 'x':
        xs, ys, zs = radius*np.cos(t), radius*np.sin(t), 0*t
    elif axis == 'y':
        xs, ys, zs = radius*np.cos(t), 0*t, radius*np.sin(t)
    else: # z-axis
        xs, ys, zs = 0*t, radius*np.cos(t), radius*np.sin(t)
    ax.plot(xs, ys, zs, color=color, linewidth=1.5)

# Draw arcs (different radii to avoid overlap)
plot_arc(ax, 3, alpha, axis='x', color='b') #  at radius 3
plot_arc(ax, 3.5, beta, axis='y', color='g') #  at radius 3.5
plot_arc(ax, 4, gamma, axis='z', color='orange') #  at radius 4

# ----------------------------
# Angle labels (separated properly)
```

Python: plot.py

```
# ----------------------------
ax.text(4.5, 1, 0, f"={alpha:.1f}", color='b') #  label far on +x
ax.text(1, 5.0, 1, f"={beta:.1f}", color='g') #  label farther on +y
ax.text(1, 1, 5.2, f"={gamma:.1f}", color='orange') #  label higher on +z

# ----------------------------
# Axes labels & limits
# ----------------------------
ax.set_xlim([0, max(0, x) + 6])
ax.set_ylim([0, max(0, y) + 6])
ax.set_zlim([min(0, z) - 6, max(0, z) + 6])

ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("3D Vector Representation with Components & Angles")

ax.legend()

# Save the figure and show it
plt.savefig("vector_plot.png", dpi=300, bbox_inches='tight')
plt.show()
```