

Problem 4.6.11

ee25btech11023-Venkata Sai

September 23, 2025

1 Problem

2 Solution

- Finding Direction Vector
- Conclusion
- Plot

3 C Code

4 Python Code

Problem

Find the equation of the line passing through the point $(1, -3, 2)$ and parallel to the line

$$\mathbf{r} = (2 + \lambda)\hat{i} + \lambda\hat{j} + (2\lambda - 1)\hat{k} \quad (1.1)$$

Finding Direction Vector

Given line is

$$\mathbf{r} = \begin{pmatrix} 2 + \lambda \\ \lambda \\ 2\lambda - 1 \end{pmatrix} \quad (2.1)$$

The vector equation of given line is given by

$$\mathbf{r} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad (2.2)$$

The direction vectors of given line are

$$\mathbf{m} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad (2.3)$$

Conclusion

The lines with direction vectors **m** and **n** are parallel if

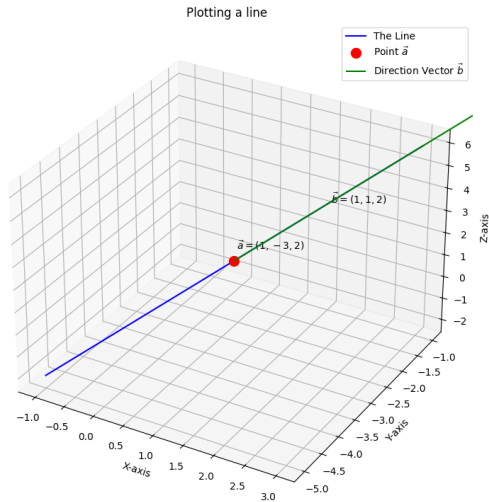
$$\mathbf{m} = \mathbf{n} \implies \mathbf{n} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad (2.4)$$

The equation of a line is given by

$$\mathbf{r} = \mathbf{a} + t\mathbf{n} \quad (2.5)$$

$$\mathbf{r} = \begin{pmatrix} 1 \\ -3 \\ 2 \end{pmatrix} + t \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad (2.6)$$

Plot



C Code

```
void get_line_vectors(double* out_data) {  
  
    double point_a[3] = {1.0, -3.0, 2.0};  
    double dir_b[3] = {1.0, 1.0, 2.0};  
  
    out_data[0] = point_a[0];  
    out_data[1] = point_a[1];  
    out_data[2] = point_a[2];  
  
    out_data[3] = dir_b[0];  
    out_data[4] = dir_b[1];  
    out_data[5] = dir_b[2];  
}
```

Calling C Function

```
import ctypes
import numpy as np

def get_vectors_from_c():
    lib = ctypes.CDLL('./vector.so')

    # The C function expects a pointer to a C double array of
    size 6
    double_array_6 = ctypes.c_double * 6
    lib.get_line_vectors.argtypes = [ctypes.POINTER(ctypes.
        c_double)]

    # Create the C-style array to receive the output data
    out_data_c = double_array_6()

    # Call the C function, which will fill the array
    lib.get_line_vectors(out_data_c)
```


Calling C Function

```
# Convert the C array back into a NumPy array
all_data = np.array(out_data_c)

# Split the data into the point vector and direction vector
point_a = all_data[:3]
dir_b = all_data[3:]

return point_a, dir_b
```

Python Code for Plotting

```
#Code by GVV Sharma  
#September 12, 2023  
#Revised July 21, 2024  
#released under GNU GPL  
  
import sys  
import matplotlib.pyplot as plt  
import numpy as np  
from mpl_toolkits.mplot3d import Axes3D  
sys.path.insert(0, '/workspaces/urban-potato/matgeo/codes/  
    CoordGeo/')  
from call import get_vectors_from_c  
hat_symbol = '\u0302'  
from line.funcs import *  
from triangle.funcs import *  
from conics.funcs import circ_gen  
  
point_a, dir_b = get_vectors_from_c()
```

Python Code for Plotting

```
from call import get_vectors_from_c

point_a, dir_b = get_vectors_from_c()

lambda_vals = np.array([-2, 2])
line_points = point_a + lambda_vals.reshape(-1,1) * dir_b

# --- Plotting ---
fig = plt.figure(figsize=(9, 9))
ax = fig.add_subplot(111, projection='3d')

# Plot the line segment itself
ax.plot(line_points[:, 0], line_points[:, 1], line_points[:, 2],
        color='blue', label='The Line')

# Plot the point 'a' on the line
ax.scatter(point_a[0], point_a[1], point_a[2], color='red', s
           =100, label='Point  $\vec{a}$ ')
```

Python Code for Plotting

```
# Plot the direction vector 'd' starting from point 'a'
ax.quiver(point_a[0], point_a[1], point_a[2],
          dir_b[0], dir_b[1], dir_b[2],
          color='green', label='Direction Vector  $\vec{b}$ ',
          length=5, arrow_length_ratio=0.3)

# Add text labels for the point and vectors
ax.text(point_a[0], point_a[1], point_a[2] + 0.5, f'  $\vec{a}$ '
        = ({point_a[0]:.0f}, {point_a[1]:.0f}, {point_a[2]:.0f}))$')
ax.text(point_a[0] + dir_b[0], point_a[1] + dir_b[1], point_a[2]
        + dir_b[2], '  $\vec{b}$ ')
```

Python Code for Plotting

```
# --- Formatting ---  
ax.set_title('Vector Equation of the Line')  
ax.set_xlabel('X-axis')  
ax.set_ylabel('Y-axis')  
ax.set_zlabel('Z-axis')  
  
ax.grid(True)  
ax.legend()  
plt.show()  
plt.savefig('fig.png')
```