

4.7.12

Harsha-EE25BTECH11026

August 29,2025

Question

Find the distance of the line $4x - y = 0$ from the point $P(4, 1)$ measured along the line making an angle of 135° with the positive x-axis.

Theoretical Solution

According to the question,

$$\text{Equation of target line: } \begin{pmatrix} 4 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0 \quad (1)$$

and

$$\mathbf{P} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \quad (2)$$

As the direction of line makes an angle of 135° with the $+x$ axis, the unit direction vector of the line is given by

$$\mathbf{m}_0 = \begin{pmatrix} \cos 135^\circ \\ \sin 135^\circ \end{pmatrix} = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad (3)$$

Equation

To calculate the distance κ of a vector \mathbf{P} from the target line $\mathbf{n}^\top \mathbf{x} = c$ along a line with direction vector \mathbf{m}_0 ,

$$\text{Parametric form: } \mathbf{x} = \mathbf{P} + \kappa \mathbf{m}_0 \quad (4)$$

$$\implies \mathbf{n}^\top (\mathbf{P} + \kappa \mathbf{m}_0) = c \quad (5)$$

$$\therefore \kappa = \frac{c - \mathbf{n}^\top \mathbf{P}}{\mathbf{n}^\top \mathbf{m}_0} \quad (6)$$

$$(7)$$

Theoretical Solution

$$\Rightarrow \kappa = \frac{-\begin{pmatrix} 4 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix}}{\begin{pmatrix} 4 & -1 \end{pmatrix} \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}} \quad (8)$$

$$\Rightarrow \kappa = 3\sqrt{2} \text{ units} \quad (9)$$

C Code -Finding Equations of diagonals of a square

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

// Function to compute intersection point Q
// Inputs: Px, Py = point P
// Outputs: Qx, Qy (via pointers)
void find_intersection(double Px, double Py, double *Qx, double *
    Qy) {
    // direction vector for 135 = (-1, 1)
    double dx = -1.0, dy = 1.0;
```

C Code -Finding Equations of diagonals of a square

```
// Parametric line: (x,y) = (Px + t dx, Py + t dy)
// Line: 4x - y = 0 -> y = 4x
// Substitute: Py + t dy = 4(Px + t dx)
// => Py + t = 4Px + 4t dx
// => Py + t = 4Px + 4t(-1)
// => Py + t = 4Px - 4t
// => t + 4t = 4Px - Py
// => 5t = 4Px - Py
double t = (4*Px - Py) / 5.0;

*Qx = Px + t*dx;
*Qy = Py + t*dy;
}

// Function to compute distance between two points
double distance(double x1, double y1, double x2, double y2) {
    return sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
}
```

```
import ctypes
import math
import matplotlib.pyplot as plt
import matplotlib as mp
mp.use("TkAgg")
# Load shared library
lib = ctypes.CDLL("./libnorm.so")
# Define function signatures
lib.find_intersection.argtypes = [ctypes.c_double, ctypes.c_double,
                                   ctypes.POINTER(ctypes.c_double),
                                   ctypes.POINTER(ctypes.c_double)]
lib.distance.argtypes = [ctypes.c_double, ctypes.c_double, ctypes.c_double, ctypes.c_double]
lib.distance.restype = ctypes.c_double
```



```
# Given point P
Px, Py = 4.0, 1.0

# Call C function to get Q
Qx = ctypes.c_double()
Qy = ctypes.c_double()
lib.find_intersection(Px, Py, ctypes.byref(Qx), ctypes.byref(Qy))
Qx, Qy = Qx.value, Qy.value

# Distance from C
dist = lib.distance(Px, Py, Qx, Qy)
print(f"distance: {dist:.3f} units")
```

```
# -----  
# Plot same as before  
# -----  
fig, ax = plt.subplots(figsize=(6, 6))  
  
# Line  $y = 4x$   
x_line = [-1, 5]  
y_line = [4*x for x in x_line]  
ax.plot(x_line, y_line, label="Line:  $4x - y = 0$  ( $y=4x$ )")  
  
# Direction line through P (slope -1)  
x_dir = [Px - 4, Px + 4]  
y_dir = [Py + 4, Py - 4]  
ax.plot(x_dir, y_dir, linestyle="--", label="Direction 135 (slope  
-1)")
```

```
# Points P and Q
ax.plot(Px, Py, 'ro')
ax.annotate(f'P({Px:.0f},{Py:.0f})', xy=(Px,Py), xytext=(Px+0.2,
    Py-0.5))
ax.plot(Qx, Qy, 'mo')
ax.annotate(f'Q({Qx:.0f},{Qy:.0f})', xy=(Qx,Qy), xytext=(Qx+0.2,
    Qy+0.2))

# Segment PQ
ax.plot([Px, Qx], [Py, Qy], 'r-', linewidth=2)

# Distance label
mid = ((Px+Qx)/2, (Py+Qy)/2)
ax.text(mid[0], mid[1]+0.3, f'distance = {dist:.3f}', ha='center',
    , color='red')
```

```
# Formatting
ax.set_aspect('equal', 'box')
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.grid(True, alpha=0.4)
ax.legend(loc="upper right")

plt.savefig("/home/user/Matrix/Matgeo_assignments/4.7.12/figs/
Figure_1.png")
plt.show()
```

Python code

```
import math
import matplotlib.pyplot as plt
import matplotlib as mp
mp.use("TkAgg")
# -----
# Data
# -----
P = (4.0, 1.0) # given point
m_dir = -1.0 # slope for 135 direction (tan 135 = -1)
d = (-1.0, 1.0) # unit direction up to scale
# Given line:  $4x - y = 0 \rightarrow y = 4x$ 
# Parametric line through P in direction d:  $(x, y) = P + t d = (4 - t, 1 + t)$ 
# Intersect with  $y = 4x$ :
#  $1 + t = 4(4 - t) \rightarrow 1 + t = 16 - 4t \rightarrow 5t = 15 \rightarrow t = 3$ 
t = 3.0
Q = (P[0] + t*d[0], P[1] + t*d[1]) # (1, 4)
```

Python code

```
# Distance along the 135 direction
DX = Q[0] - P[0]
DY = Q[1] - P[1]
distance = math.hypot(DX, DY) # 3*sqrt(2)

print(f"distance: {distance:.3f} units")

# -----
# Plot
# -----
fig, ax = plt.subplots(figsize=(6, 6))

# Plot given line y = 4x
x_line = [ -1, 5 ]
y_line = [ 4*x for x in x_line ]
ax.plot(x_line, y_line, label='Line: 4x - y = 0 (y=4x)')
```

Python code

```
# Plot direction line through P (slope -1)
x_dir = [P[0] - 4, P[0] + 4]
y_dir = [P[1] + 4, P[1] - 4]
ax.plot(x_dir, y_dir, linestyle='--', label='Direction 135 (slope
-1)')

# Points P and Q
ax.plot(P[0], P[1], 'ro')
ax.annotate('P(4,1)', xy=P, xytext=(P[0]+0.2, P[1]-0.5))
ax.plot(Q[0], Q[1], 'mo')
ax.annotate(f'Q{Q}', xy=Q, xytext=(Q[0]+0.2, Q[1]+0.2))
# Segment PQ (the measured distance)
ax.plot([P[0], Q[0]], [P[1], Q[1]], 'r-', linewidth=2)
# Annotate distance directly above the segment
mid = ((P[0]+Q[0])/2, (P[1]+Q[1])/2)
ax.text(mid[0], mid[1]+0.3, f'distance = {distance:.3f}', ha='
center', va='bottom', fontsize=10, color='red')
```

Python code

```
# Axes formatting
ax.set_aspect('equal', 'box')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.grid(True, alpha=0.4)
ax.legend(loc='upper right')

# Set sensible limits around all geometry
xs = [-1, 5, P[0], Q[0]]
ys = [min(y_line), max(y_line), P[1], Q[1]]
ax.set_xlim(min(xs)-1, max(xs)+1)
ax.set_ylim(min(ys)-1, max(ys)+1)

plt.savefig("/home/user/Matrix/Matgeo_assignments/4.7.12/figs/
            Figure_1.png")
plt.show()
```


Plot

