4.3.10

Aniket-EE25BTECH11007

October 4, 2025

Question

Find the direction and normal vectors of the line $x - y = 2$.

Theoretical Solution

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ c \end{pmatrix} + x \begin{pmatrix} 1 \\ m \end{pmatrix} \tag{1}$$

where $\begin{pmatrix} 1 \\ m \end{pmatrix}$ is the direction vector and m is the slope.

$$\mathbf{n}^\top x = c \tag{2}$$

where $\mathbf{n}$ is the normal vector of the line.

$$\mathbf{n}^\top \begin{pmatrix} 1 \\ m \end{pmatrix} = 0 \tag{3}$$

From $x - y = 2$,the slope is $m = 1$. Hence, using (1),

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -2 \end{pmatrix} + x \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{4}$$

Theoretical Solution

Let $\begin{pmatrix} x \\ y \end{pmatrix}$ be a normal vector. Then, from (3),

$$\begin{pmatrix} x \\ y \end{pmatrix}^{\top} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0 \quad \Longrightarrow \quad x + y = 0 \quad \Longrightarrow \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{5}$$

Line in normal form using (2)

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}^{\top} \begin{pmatrix} x \\ y \end{pmatrix} = 2$$

Hence, the direction vector is $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and the normal vector is $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

C Code

```c
// mg4.c
#include <math.h>

double arr_dot(const double *u, const double *v, int n) {
    double sum = 0.0;
    for (int i = 0; i < n; ++i) sum += u[i] * v[i];
    return sum;
}

double arr_norm(const double *u, int n) {
    double s = arr_dot(u, u, n);
    if (s <= 0.0) return 0.0;
    return sqrt(s);
}

void arr_scale(const double *u, double k, double *out, int n) {
    for (int i = 0; i < n; ++i) out[i] = k * u[i];
}
```

C Code

```c
void arr_normalize(const double *u, double *out, int n) {
    double r = arr_norm(u, n);
    if (r == 0.0) {
        for (int i = 0; i < n; ++i) out[i] = 0.0;
    } else {
        for (int i = 0; i < n; ++i) out[i] = u[i] / r;
    }
}

/* Line: ax + by = c
   normal = (a, b)
   direction = (-b, a) (perpendicular to normal)
*/
void line_direction_normal(double a, double b, double c,
                           double *dir_out, /* len 2 */
                           double *normal_out) { /* len 2 */
    (void)c; /* c not needed for vectors */
    normal_out[0] = a; normal_out[1] = b;
    dir_out[0] = -b; dir_out[1] = a;
```

Python + C

```
import ctypes
from ctypes import c_double, c_int, POINTER
import numpy as np
import matplotlib.pyplot as plt

lib = ctypes.CDLL("./mg4.so")

lib.arr_dot.argtypes = [POINTER(c_double), POINTER(c_double),
    c_int]
lib.arr_dot.restype = c_double
lib.arr_norm.argtypes = [POINTER(c_double), c_int]
lib.arr_norm.restype = c_double
lib.arr_normalize.argtypes = [POINTER(c_double), POINTER(c_double
    ), c_int]
lib.arr_normalize.restype = None
lib.line_direction_normal.argtypes = [c_double, c_double,
    c_double,
                                      POINTER(c_double), POINTER(
                                          c_double)]
```

Python + C

```
def cvec(arr):
    arr = np.asarray(arr, dtype=float)
    return (c_double * arr.size)(*arr.tolist()), arr.size

def dot(u, v):
    cu, n = cvec(u); cv, _ = cvec(v)
    return lib.arr_dot(cu, cv, n)

def normalize(u):
    cu, n = cvec(u)
    out = (c_double * n)()
    lib.arr_normalize(cu, out, n)
    return np.array([out[i] for i in range(n)], dtype=float)

# Solve x - y = 2 (a=1, b=-1, c=2)
a, b, c = 1.0, -1.0, 2.0
d = (c_double * 2)()
n = (c_double * 2)()
lib.line_direction_normal(a, b, c, d, n)
```

Python + C

```
print("Direction (raw):", d_np.tolist()) # [1.0, 1.0]
print("Normal (raw):", n_np.tolist()) # [1.0, -1.0]
print("dot(direction, normal) =", dot(d_np, n_np))

x = np.linspace(-2, 6, 200)
y = x - 2

x0, y0 = 2.0, 0.0
scale = 2.0

plt.figure()
plt.plot(x, y, label="x - y = 2")
plt.quiver([x0], [y0], [ud[0]*scale], [ud[1]*scale],
           angles='xy', scale_units='xy', scale=1, label="
               direction")
plt.quiver([x0], [y0], [un[0]*scale], [un[1]*scale],
           angles='xy', scale_units='xy', scale=1, label="normal")
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel("x"); plt.ylabel("y")
```

Python

```
import numpy as np
import matplotlib.pyplot as plt

def dot(u, v):
    u = np.asarray(u, dtype=float)
    v = np.asarray(v, dtype=float)
    if u.shape != v.shape:
        raise ValueError("dot: shapes must match")
    return float(np.dot(u, v))

def norm(u):
    u = np.asarray(u, dtype=float)
    return float(np.sqrt(np.dot(u, u)))

def normalize(u):
    u = np.asarray(u, dtype=float)
    r = norm(u)
    return u / r if r != 0.0 else np.zeros_like(u)
```

Python

```
def line_direction_normal(a, b, c):
    d = np.array([-b, a], dtype=float) # direction
    n = np.array([a, b], dtype=float) # normal
    return d, n

a, b, c = 1.0, -1.0, 2.0
d, n = line_direction_normal(a, b, c)
ud, un = normalize(d), normalize(n)

print("Direction (raw):", d.tolist()) # [1.0, 1.0]
print("Normal (raw):", n.tolist()) # [1.0, -1.0]
print("dot(direction, normal) =", dot(d, n)) # 0.0

x = np.linspace(-2, 6, 200)
y = x - 2
x0, y0 = 2.0, 0.0
scale = 2.0

plt.figure()
```

Plot