4.13.45

Naman Kumar-EE25BTECH11041

25 September,2025

Question b)

Find the equation of the line which bisects the obtuse angle between the lines $x - 2y + 4 = 0$ and $4x - 3y + 2 = 0$.

Solution

Given,

$\mathbf{n_1}$	Normal vector of line 1
$\mathbf{n_2}$	Normal vector of line 2
c_1	constant of line 1
c_2	constant of line 2
$\mathbf{B}$	vector on bisector
$\mathbf{n_{B_1}}$	Normal vector Bisector of line 1 and 2
$\mathbf{n_{B_2}}$	Normal vector Bisector of line 1 and 2
θ_1	angle between line 1 and bisector 1
θ_2	angle between line 1 and bisector 2

Table:

$$\mathbf{n_1}^T\mathbf{x} = c_1, \mathbf{n_2}^T\mathbf{x} \tag{1}$$

Solution

Where,

$$\mathbf{n_1} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \mathbf{n_2} = \begin{pmatrix} 4 \\ -3 \end{pmatrix}, c_1 = -4 \text{ and } c_2 = -2 \tag{2}$$

Equation for bisectors is

$$\left|\frac{\mathbf{n_1}^T\mathbf{B} - c_1}{\| \mathbf{n_1} \|}\right| = \left|\frac{\mathbf{n_2}^T\mathbf{B} - c_2}{\| \mathbf{n_2} \|}\right| \tag{3}$$

$$\frac{\mathbf{n_1}^T\mathbf{B} - c_1}{\| \mathbf{n_1} \|} = \pm\frac{\mathbf{n_2}^T\mathbf{B} - c_2}{\| \mathbf{n_2} \|} \tag{4}$$

$$\frac{\mathbf{n_1}^T\mathbf{B_1} - c_1}{\| \mathbf{n_1} \|} = \frac{\mathbf{n_2}^T\mathbf{B_1} - c_2}{\| \mathbf{n_2} \|}, \text{ and } \frac{\mathbf{n_1}^T\mathbf{B_2} - c_1}{\| \mathbf{n_1} \|} = -\frac{\mathbf{n_2}^T\mathbf{B_2} - c_2}{\| \mathbf{n_2} \|} \tag{5}$$

Solution

Can be written as

$$\left(\frac{\mathbf{n_1}^T}{\| \mathbf{n_1} \|} - \frac{\mathbf{n_2}^T}{\| \mathbf{n_2} \|}\right)\mathbf{B_1} = \left(\frac{c_1}{\| \mathbf{n_1} \|} - \frac{c_2}{\| \mathbf{n_2} \|}\right) \tag{6}$$

$$\mathbf{n_{B_1}}^T B_1 = c_{B_1} \tag{7}$$

and

$$\left(\frac{\mathbf{n_1}^T}{\| \mathbf{n_1} \|} + \frac{\mathbf{n_2}^T}{\| \mathbf{n_2} \|}\right)\mathbf{B_2} = \left(\frac{c_2}{\| \mathbf{n_2} \|} + \frac{c_1}{\| \mathbf{n_1} \|}\right) b2 \tag{8}$$

$$\mathbf{n_{B_2}}^T B_2 = c_{B_2} \tag{9}$$

Now for obtuse angle bisector

$$\cos\theta_1 = \frac{\mathbf{n_{B_1}}^T \mathbf{n_1}}{\| \mathbf{n_1} \| \| \mathbf{n_{B_1}} \|} \tag{10}$$

$$\cos\theta_2 = \frac{\mathbf{n_{B_2}}^T \mathbf{n_1}}{\| \mathbf{n_1} \| \| \mathbf{n_{B_2}} \|} \tag{11}$$

Solution

Solving with (7) and (9)

$$\cos\theta_1 = \frac{\left(\frac{\mathbf{n_1}^T}{\|\mathbf{n_1}\|} - \frac{\mathbf{n_2}^T}{\|\mathbf{n_2}\|}\right)\mathbf{n_1}}{\| \mathbf{n_1} \|\| \mathbf{n_{B_1}} \|} \tag{12}$$

$$\cos\theta_1 = \frac{\left(\frac{\mathbf{n_1}^T}{\|\mathbf{n_1}\|}\mathbf{n_1} - \frac{\mathbf{n_2}^T}{\|\mathbf{n_2}\|}\mathbf{n_1}\right)}{\| \mathbf{n_1} \|\| \mathbf{n_{B_1}} \|} \tag{13}$$

$$\cos\theta_1 = \frac{\left(\sqrt{5} - \frac{10}{5}\right)}{\| \mathbf{n_1} \|\| \mathbf{n_{B_1}} \|} \tag{14}$$

$$\cos\theta_1 = \frac{\sqrt{5} - 2}{\sqrt{5}\sqrt{\left(\frac{1}{\sqrt{5}} - \frac{4}{5}\right)^2 + \left(\frac{3}{5} - \frac{2}{\sqrt{5}}\right)^2}} \approx 0.22 \tag{15}$$

Solution

Similarly

$$\cos\theta_2 = \frac{\sqrt{5}+2}{\sqrt{5}\sqrt{\left(\frac{1}{\sqrt{5}}+\frac{4}{5}\right)^2+\left(\frac{3}{5}+\frac{2}{\sqrt{5}}\right)^2}} \approx 0.97 \tag{16}$$

by comparing (15) and (16)

$$\theta_1 > \theta_2 \tag{17}$$

Solution

So B_1 is obtuse angle bisector

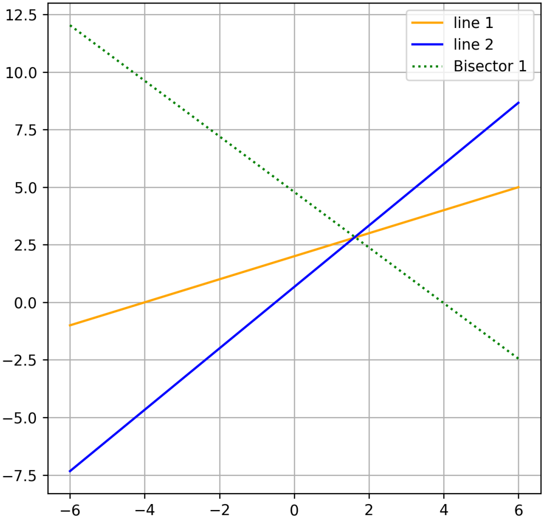
$$\mathbf{n_{B_1}}^T B_1 = c_{B_1} \tag{18}$$

$$\mathbf{n_{B_1}} = \begin{pmatrix} \frac{1}{\sqrt{5}} - \frac{4}{5} \\ \frac{3}{5} - \frac{2}{\sqrt{5}} \end{pmatrix} \tag{19}$$

and

$$c_{B_1} = \frac{2}{5} + \frac{-4}{\sqrt{5}} \tag{20}$$

Figure



C code

```
// main.c
// Compile with: gcc -shared -o libbisector.so -fPIC main.c -lm

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

typedef struct {
    double a;
    double b;
    double c;
} Line;

// helper: clamp for acos
static double clamp(double v, double lo, double hi){
    if(v < lo) return lo;
    if(v > hi) return hi;
    return v;
}
```

C code

```c
// compute direction vector (dx,dy) for line ax+by+c=0
// direction vector = (b, -a)
static void dir_vector(double a, double b, double *dx, double *dy
    ){
    *dx = b;
    *dy = -a;
}

// angle between directions (dx1,dy1) and (dx2,dy2) in [0, pi]
static double angle_between(double dx1, double dy1, double dx2,
    double dy2){
    double dot = dx1*dx2 + dy1*dy2;
    double n1 = sqrt(dx1*dx1 + dy1*dy1);
    double n2 = sqrt(dx2*dx2 + dy2*dy2);
    if(n1==0 || n2==0) return 0.0;
    double cosv = clamp(dot/(n1*n2), -1.0, 1.0);
    return acos(cosv);
}
```

C code

```
// Solve intersection of two lines. returns 1 if solvable, 0 if
    parallel.
static int intersect_point(double a1,double b1,double c1,double
    a2,double b2,double c2, double *x, double *y){
    double D = a1*b2 - a2*b1;
    if(fabs(D) < 1e-12) return 0;
    *x = (b1*c2 - b2*c1) / D;
    *y = (a2*c1 - a1*c2) / D;
    return 1;
}

// API: compute the obtuse-angle bisector line coefficients into
    result
Line obtuse_bisector(double a1, double b1, double c1, double a2,
    double b2, double c2){
    Line res = {0,0,0};
```

C code

```c
// normalization factors
double s1 = sqrt(a1*a1 + b1*b1);
double s2 = sqrt(a2*a2 + b2*b2);
if(s1 == 0 || s2 == 0){
    return res;
}

// Build the two candidate bisector lines:
// (+) => (a1/s1 - a2/s2) x + (b1/s1 - b2/s2) y + (c1/s1 - c2
    /s2) = 0
// (-) => (a1/s1 + a2/s2) x + (b1/s1 + b2/s2) y + (c1/s1 + c2
    /s2) = 0
Line Lplus, Lminus;
Lplus.a = a1/s1 - a2/s2;
Lplus.b = b1/s1 - b2/s2;
Lplus.c = c1/s1 - c2/s2;
```

C code

```
    Lminus.a = a1/s1 + a2/s2;
    Lminus.b = b1/s1 + b2/s2;
    Lminus.c = c1/s1 + c2/s2;

    // We need to choose which of Lplus or Lminus is the bisector
         of the obtuse angle.
    // Strategy:
    // - compute direction vectors of original lines and the two
        bisectors
    // - compute the small angle between L1 and L2 (in [0, pi/2])
    // - for each bisector compute its angle with L1 (in [0, pi])
    // - the bisector corresponding to the obtuse angle will make
         an angle > pi/4 with L1
    // This numeric test is stable for non-degenerate cases.
```

C code

```c
double dx1, dy1;
dir_vector(a1, b1, &dx1, &dy1);
double dx2, dy2;
dir_vector(a2, b2, &dx2, &dy2);

double theta = angle_between(dx1, dy1, dx2, dy2);
if(theta > M_PI_2) theta = M_PI - theta; // small angle in
    [0, pi/2]

// bisector directions
double dpx, dpy, dmx, dmy;
dir_vector(Lplus.a, Lplus.b, &dpx, &dpy);
dir_vector(Lminus.a, Lminus.b, &dmx, &dmy);

double alpha_plus = angle_between(dx1, dy1, dpx, dpy);
double alpha_minus = angle_between(dx1, dy1, dmx, dmy);
```

C code

```c
// The bisector of the obtuse angle will have alpha > pi/4 (
    because obtuse half-angle > pi/4).
if(alpha_plus > alpha_minus){
    // Choose plus if it gives larger angle
    res = Lplus;
} else {
    res = Lminus;
}
// Optionally normalize so that sqrt(a^2+b^2)=1 and keep sign
     consistent
double norm = sqrt(res.a*res.a + res.b*res.b);
if(norm > 1e-12){
    res.a /= norm;
    res.b /= norm;
    res.c /= norm;
}
```

C code

```c
    // ensure consistent sign: make a >= 0 or if a==0 ensure b>=0
    if(res.a < -1e-12 || (fabs(res.a) < 1e-12 && res.b < -1e-12))
        {
        res.a = -res.a; res.b = -res.b; res.c = -res.c;
    }
    return res;
}
// If used from command line for quick test
#ifdef TEST_C
int main(){
    // Given example: x - 2y + 4 = 0 and 4x - 3y + 2 = 0
    double a1=1, b1=-2, c1=4;
    double a2=4, b2=-3, c2=2;
    Line obt = obtuse_bisector(a1,b1,c1,a2,b2,c2);
    printf("Obtuse bisector: %.6f x + %.6f y + %.6f = 0\n", obt.a
        , obt.b, obt.c);
    return 0;
}
#endif
```

Python code shared object

```
# main.py
# Usage:
# 1. Compile C: gcc -shared -o libbisector.so -fPIC main.c -lm
# 2. Run: python3 main.py

import ctypes
from ctypes import Structure, c_double
import numpy as np
import matplotlib.pyplot as plt
import math
import os

# define Line struct to match C
class Line(Structure):
    _fields_ = [("a", c_double), ("b", c_double), ("c", c_double)
        ]
```

Python code shared object

```python
# load shared library (adjust path if necessary)
libpath = "./libbisector.so"
if not os.path.exists(libpath):
    raise RuntimeError(f"Shared object {libpath} not found.
        Compile main.c first.")

lib = ctypes.CDLL(libpath)
lib.obtuse_bisector.restype = Line
lib.obtuse_bisector.argtypes = [c_double,c_double,c_double,
    c_double,c_double,c_double]

# Given lines
# L1: x - 2y + 4 = 0 => a1=1, b1=-2, c1=4
# L2: 4x - 3y + 2 = 0 => a2=4, b2=-3, c2=2
a1, b1, c1 = 1.0, -2.0, 4.0
a2, b2, c2 = 4.0, -3.0, 2.0
```

Python code shared object

```
# call C function
res = lib.obtuse_bisector(a1,b1,c1,a2,b2,c2)
A, B, C = res.a, res.b, res.c

# print equation in nicer form (scale back to readable)
# We'll scale so that coefficients are not tiny; find max abs
scale = max(abs(A), abs(B), abs(C))
if scale < 1e-12: scale = 1.0
A_s, B_s, C_s = A/scale, B/scale, C/scale

print("Computed (normalized) obtuse-angle bisector coefficients:"
    )
print(f"A = {A:.6f}, B = {B:.6f}, C = {C:.6f}")
print(f"Equation: ({A_s:.6f}) x + ({B_s:.6f}) y + ({C_s:.6f}) = 0
     (scaled)")
```

Python code shared object

```python
# Compute intersection point of L1 and L2 for plotting center
D = a1*b2 - a2*b1
if abs(D) < 1e-12:
    print("Given lines are parallel or nearly parallel; cannot
        find intersection.")
    Px = Py = 0.0
else:
    Px = (b1*c2 - b2*c1) / D
    Py = (a2*c1 - a1*c2) / D
    print(f"Intersection point: P = ({Px:.6f}, {Py:.6f})")
```

Python code shared object

```
# Prepare plotting
# produce a range around intersection
rng = 10
xs = np.linspace(Px - rng, Px + rng, 400)

# function to produce y from ax+by+c=0 -> y = (-a x - c)/b if b
    !=0 else None
def line_y(a,b,c, xvals):
    if abs(b) < 1e-12:
        return None
    return [(-a*x - c)/b for x in xvals]
```

Python code shared object

```
y1 = line_y(a1,b1,c1, xs)
y2 = line_y(a2,b2,c2, xs)
yB = line_y(A,B,C, xs)

plt.figure(figsize=(8,8))
if y1 is not None:
    plt.plot(xs, y1, label="L1: x - 2y + 4 = 0", linewidth=2)
else:
    # vertical line x = -c/a
    xv = -c1/a1
    plt.axvline(x=xv, label="L1 (vertical)")
```

Python code shared object

```
if y2 is not None:
    plt.plot(xs, y2, label="L2: 4x - 3y + 2 = 0", linewidth=2)
else:
    xv = -c2/a2
    plt.axvline(x=xv, label="L2 (vertical)")

if yB is not None:
    plt.plot(xs, yB, '--', label="Obtuse-angle bisector",
        linewidth=2)
else:
    xv = -C/A
    plt.axvline(x=xv, linestyle='--', label="Obtuse bisector (
        vertical)")
```

Python code shared object

```
# plot intersection point
plt.scatter([Px],[Py], color='k')
plt.text(Px, Py, ' P(intersection)', fontsize=9)

plt.axhline(0, color='gray', linewidth=0.5)
plt.axvline(0, color='gray', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Lines and the Obtuse-angle Bisector')
plt.axis('equal')
plt.show()
```

Direct Python

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

plt.figure(figsize=(6,6), dpi=200)

x1= np.linspace(-6,6,100)
y1=(x1+4)/2

x2= np.linspace(-6,6,100)
y2=(4*x2+2)/3
```

Direct Python

```
bx=np.linspace(-6,6,100)
by=(1.39-0.35*bx)/0.29

plt.plot(x1,y1, color='orange', label="line 1")
plt.plot(x2,y2, color='blue', label="line 2")
plt.plot(bx,by,':', color='green', label="Bisector 1")
plt.legend()
plt.grid()
plt.savefig("figure.png", dpi=250)
plt.show()
```