4.3.13

Harsha-EE25BTECH11026

August 27,2025

Question

Equations of the diagonals of the square formed by the lines $x = 0$, $y = 0$, $x = 1$ and $y = 1$ are ____________.

Theoretical Solution

According to the question,
The vertices of the square are ,

$$\mathbf{a} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \mathbf{d} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Equation

To compute the equation of the diagnols , we can use the normal form of the equation,which is given by

$$\mathbf{n}^T\mathbf{x} = 0 \text{ } \textit{for the lines passing through the origin}$$

$$\mathbf{n}^T\mathbf{x} = 1 \text{ } \textit{for the lines not passing through the origin}$$

where,

$$\mathbf{n} - \textit{vector orthogonal to the direction vector}$$

$$\mathbf{x} = \begin{pmatrix} x & y \end{pmatrix}^T$$

Theoretical Solution

For diagonal $\mathbf{c} - \mathbf{a}$, as it passes through the origin,

$$\therefore \mathbf{n}^T \mathbf{x} = 0$$

By substituting the vector through which it passes through,

$$\mathbf{n}^T \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0$$

$$\implies \mathbf{n} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

$$\therefore \begin{pmatrix} -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0$$

Theoretical Solution

But, for diagonal $\mathbf{d} - \mathbf{b}$, as the diagonal doesn't pass through the origin,

$$\mathbf{n}^T \mathbf{x} = 1$$

$$\therefore \mathbf{n}^T \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\implies \mathbf{n} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\therefore \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 1$$

C Code -Finding Equations of diagonals of a square

```
#include <stdio.h>

void diagonal_equations(double vertices[4][2], double n1[2],
    double *c1, double n2[2], double *c2) {
    // A, B, C, D are the vertices
    double *A = vertices[0];
    double *B = vertices[1];
    double *C = vertices[2];
    double *D = vertices[3];

    // Diagonal AC
    double dx1 = C[0] - A[0];
    double dy1 = C[1] - A[1];
    n1[0] = dy1;
    n1[1] = -dx1;
    *c1 = n1[0]*A[0] + n1[1]*A[1];
```

C Code -Finding Equations of diagonals of a square

```
    // Diagonal BD
    double dx2 = D[0] - B[0];
    double dy2 = D[1] - B[1];
    n2[0] = dy2;
    n2[1] = -dx2;
    *c2 = n2[0]*B[0] + n2[1]*B[1];
}
```

Python+C code

```
import ctypes
import numpy as np
import matplotlib as mp
mp.use("TkAgg")
import matplotlib.pyplot as plt

# Load C shared library
lib = ctypes.CDLL("./libdiagonals.so")

# Define argument types for the function
lib.diagonal_equations.argtypes = [
    (ctypes.c_double * 2) * 4,
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]
```

Python+C code

```
# Define square vertices
vertices = [(0.0,0.0), (1.0,0.0), (1.0,1.0), (0.0,1.0)]
vert_array = ((ctypes.c_double*2)*4)(*[(ctypes.c_double*2)(*v)
    for v in vertices])

# Output containers
n1 = (ctypes.c_double*2)()
c1 = ctypes.c_double()
n2 = (ctypes.c_double*2)()
c2 = ctypes.c_double()

# Call C function
lib.diagonal_equations(vert_array, n1, ctypes.byref(c1), n2,
    ctypes.byref(c2))

n1 = np.array([n1[0], n1[1]])
n2 = np.array([n2[0], n2[1]])
```

Python+C code

```
# Convert to Cartesian equation: ax + by + d = 0
def cartesian_eq(n, c):
    a, b = n
    d = -c
    eq = []
    if a != 0:
        eq.append(f"{a}x")
    if b != 0:
        sign = "+" if b > 0 and eq else ""
        eq.append(f"{sign}{b}y")
    if d != 0:
        sign = "+" if d > 0 and eq else ""
        eq.append(f"{sign}{d}")
    return " ".join(eq) + " = 0"

eq1 = cartesian_eq(n1, c1.value)
eq2 = cartesian_eq(n2, c2.value)
```

Python+C code

```
print("Diagonal AC (normal form):", f"[{n1[0]} {n1[1]}]  [x y]^T
    = {c1.value}")
print("Diagonal BD (normal form):", f"[{n2[0]} {n2[1]}]  [x y]^T
    = {c2.value}")

# ---- PLOT ----
A, B, C, D = vertices
square_x = [A[0], B[0], C[0], D[0], A[0]]
square_y = [A[1], B[1], C[1], D[1], A[1]]

plt.plot(square_x, square_y, 'b-', label='Square')

# Plot diagonals
plt.plot([A[0], C[0]], [A[1], C[1]], 'r--', label=eq1)
plt.plot([B[0], D[0]], [B[1], D[1]], 'g--', label=eq2)
```

Python+C code

```
plt.gca().set_aspect('equal', adjustable='box')
plt.legend(loc="upper right")
plt.grid(True)
plt.savefig("/home/user/Matrix/Matgeo_assignments/4.3.13/figs/
    Figure_1.png")
plt.show()
```

Python code

```
import matplotlib as mp
mp.use("TkAgg")
import numpy as np
import matplotlib.pyplot as plt

def line_equation_normal(point1, point2):
    """Returns line equation in normal form: n^T x = c
       where n is the normal vector and x = [x y]^T."""
    x1, y1 = point1
    x2, y2 = point2
    # Direction vector
    dx, dy = x2 - x1, y2 - y1
    # Normal vector
    n = np.array([dy, -dx])
    # Constant term
    c = n @ np.array([x1, y1])
    return n, c
```

```python
def diagonals_of_square(vertices):
    """
    Given 4 vertices of a square (in order), compute equations of
        diagonals.
    """
    A, B, C, D = vertices

    # Diagonals are AC and BD
    line1 = line_equation_normal(A, C)
    line2 = line_equation_normal(B, D)

    return line1, line2

def format_normal_form(n, c):
    """Format equation in normal form."""
    return f"[{n[0]} {n[1]}]  [x y]^T = {c}"
```

Python code

```python
def format_cartesian(n, c):
    """Convert n^T x = c into Cartesian form ax + by + d = 0
    where n = [a b]."""
    a, b = n
    d = -c
    terms = []
    if a != 0:
        terms.append(f"{'' if a == 1 else '-' if a == -1 else a}x")
    if b != 0:
        sign = "+" if b > 0 and terms else ""
        terms.append(f"{sign}{'' if abs(b) == 1 else b}y" if b not in [1, -1] else f"{sign}{'y' if b == 1 else '-y'}")
    if d != 0:
        sign = "+" if d > 0 and terms else ""
        terms.append(f"{sign}{d}")
    return " ".join(terms) + " = 0"
```

Python code

```python
def plot_square_and_diagonals(vertices, line1, line2):
"""Plot square and its diagonals with Cartesian equations on
    the plot."""
A, B, C, D = vertices
square_x = [A[0], B[0], C[0], D[0], A[0]]
square_y = [A[1], B[1], C[1], D[1], A[1]]

plt.plot(square_x, square_y, 'b-', label='Square')

# Plot diagonals
plt.plot([A[0], C[0]], [A[1], C[1]], 'r--', label='Diagonal
    AC')
plt.plot([B[0], D[0]], [B[1], D[1]], 'g--', label='Diagonal
    BD')
# Equations in Cartesian form for plot
eq1 = format_cartesian(*line1)
eq2 = format_cartesian(*line2)
```

Python code

```
# Midpoints of diagonals
   mid_AC = ((A[0]+C[0])/2, (A[1]+C[1])/2)
   mid_BD = ((B[0]+D[0])/2, (B[1]+D[1])/2)
   # Place texts
   plt.text(mid_AC[0]+0.05, mid_AC[1]+0.05, eq1, color='red',
       fontsize=10, ha='left')
   plt.text(mid_BD[0]-0.15, mid_BD[1]-0.1, eq2, color='green',
       fontsize=10, ha='right')

   plt.gca().set_aspect('equal', adjustable='box')
   plt.legend(loc="upper right")
   plt.grid(True)
   plt.savefig("/home/user/Matrix/Matgeo_assignments/4.3.13/figs
       /Figure_1")
   plt.show()
```

Python code

```
vertices = [(0,0), (1,0), (1,1), (0,1)]

# Compute diagonal equations
line1, line2 = diagonals_of_square(vertices)

# Print normal forms
print("Diagonal AC equation (normal form):", format_normal_form(*
    line1))
print("Diagonal BD equation (normal form):", format_normal_form(*
    line2))

# Plot with Cartesian equations
plot_square_and_diagonals(vertices, line1, line2)
```

Plot

Figure: Plot of square and its diagonals