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 ,

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

Theoretical Solution

To compute the equation of the diagonals from the direction vectors, we can use the normal form of the equation,which is given by

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

where,

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

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

$\mathbf{P}$=A point which lies along the vector

Equation

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

$$\mathbf{n}^T\mathbf{r} = \mathbf{n}^T\mathbf{P}$$

where,

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

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

$\mathbf{P}$=A point which lies along the vector

Theoretical Solution

For diagnol $\mathbf{c} - \mathbf{a}$,

$$\mathbf{n} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{d}$$

where $\mathbf{d}$ is the direction vector of diagonal.

$$\therefore \mathbf{n} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right)$$

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

$$\mathbf{n} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad and \quad \mathbf{P} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

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

Theoretical Solution

Similarly, for diagnol $\mathbf{d} - \mathbf{b}$,

$$\therefore \mathbf{n} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right)$$

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

$$\mathbf{n} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad and \quad \mathbf{P} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

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

C Code -Finding Diagonals of a square

```c
    #include <stdio.h>

typedef struct {
    double A, B, C; // Line coefficients Ax + By + C = 0
} Line;

// Function to compute line equation given two points (x1,y1), (
    x2,y2)
Line line_from_points(double x1, double y1, double x2, double y2)
     {
    Line l;
    l.A = y1 - y2;
    l.B = x2 - x1;
    l.C = (x1 * y2) - (x2 * y1);
    return l;
}
```

C Code -Finding Diagonals of a square

```c
// Function to compute diagonals of a square
void diagonals_of_square(double vertices[4][2], Line *diag1, Line
     *diag2) {
    // vertices order: A, B, C, D
    // Diagonals: AC and BD
    *diag1 = line_from_points(vertices[0][0], vertices[0][1],
                              vertices[2][0], vertices[2][1]);
    *diag2 = line_from_points(vertices[1][0], vertices[1][1],
                              vertices[3][0], vertices[3][1]);
}
```

C Code -Finding Diagonals of a square

```c
// Export function for Python
__attribute__((visibility("default")))
void get_square_diagonals(double vertices[4][2], double *out) {
    Line d1, d2;
    diagonals_of_square(vertices, &d1, &d2);


    // Store results in array: [A1, B1, C1, A2, B2, C2]
    out[0] = d1.A;
    out[1] = d1.B;
    out[2] = d1.C;
    out[3] = d2.A;
    out[4] = d2.B;
    out[5] = d2.C;
}
```

Python+C code

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

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

# Define function signature for C function
lib.get_square_diagonals.argtypes = [ctypes.c_double * 8, ctypes.
    c_double * 6]
```

Python+C code

```
def get_diagonals(vertices):
    """Call C function to compute diagonals of a square"""
    verts = (ctypes.c_double * 8)(*np.array(vertices).flatten())
    out = (ctypes.c_double * 6)()
    lib.get_square_diagonals(verts, out)
    return np.array(out[:]).reshape(2,3) # [[A1,B1,C1],[A2,B2,C2
        ]]
```

Python+C code

```
def format_equation(A, B, C):
    """
    Beautify line equation into readable string.
    Example: -1x + 1y + 0 -> -x + y = 0
    """
    terms = []

    # Ax term
    if A != 0:
        if A == 1:
            terms.append("x")
        elif A == -1:
            terms.append("-x")
        else:
            terms.append(f"{A:g}x")
```

Python+C code

```
# By term
   if B != 0:
       sign = "+" if B > 0 and terms else ""
       if B == 1:
           terms.append(f"{sign}y")
       elif B == -1:
           terms.append(f"{sign}-y")
       else:
           terms.append(f"{sign}{B:g}y")
 # Constant term
   if C != 0:
       sign = "+" if C > 0 and terms else ""
       terms.append(f"{sign}{C:g}")
   if not terms:
       return "0 = 0"
   return " ".join(terms) + " = 0"
```

Python+C code

```
# Example: Square vertices (A,B,C,D)
vertices = [(0,0), (1,0), (1,1), (0,1)]
lines = get_diagonals(vertices)

eq1 = format_equation(*lines[0])
eq2 = format_equation(*lines[1])

print("Diagonal AC:", eq1)
print("Diagonal BD:", eq2)

# Plotting
square_x, square_y = zip(*vertices, vertices[0])
plt.plot(square_x, square_y, "b-", label="Square")
```

Python+C code

```
# Diagonal AC
plt.plot([vertices[0][0], vertices[2][0]], [vertices[0][1],
    vertices[2][1]], "r--", label=f"AC: {eq1}")

# Diagonal BD
plt.plot([vertices[1][0], vertices[3][0]], [vertices[1][1],
    vertices[3][1]], "g--", label=f"BD: {eq2}")

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

Python code

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

def line_equation_cartesian(point, direction):
    """
    Returns line equation in Cartesian form: Ax + By + C = 0
    given a point and a direction vector.
    """
    x0, y0 = point
    a, b = direction

    # Normal vector
    A, B = -b, a
    C = -(A*x0 + B*y0)
    return A, B, C
```

Python code

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

    # Diagonals are AC and BD
    AC_dir = (C[0]-A[0], C[1]-A[1])
    BD_dir = (D[0]-B[0], D[1]-B[1])

    line1 = line_equation_cartesian(A, AC_dir)
    line2 = line_equation_cartesian(B, BD_dir)

    return line1, line2
```

Python code

```python
def format_equation(A, B, C):
    """
    Beautify line equation into readable string.
    Example: -1x + 1y + 0 -> -x + y = 0
    """
    terms = []

    # Handle Ax term
    if A != 0:
        if A == 1:
            terms.append("x")
        elif A == -1:
            terms.append("-x")
        else:
            terms.append(f"{A}x")
```

Python code

```
    # Handle By term
    if B != 0:
        sign = "+" if B > 0 and terms else ""
        if B == 1:
            terms.append(f"{sign}y")
        elif B == -1:
            terms.append(f"{sign}-y")
        else:
            terms.append(f"{sign}{B}y")
            # Handle C constant term
    if C != 0:
        sign = "+" if C > 0 and terms else ""
        terms.append(f"{sign}{C}")

    # In case all are zero
    if not terms:
        return "0 = 0"
    return " ".join(terms) + " = 0"
```

Python code

```
def plot_square_and_diagonals(vertices, line1, line2):
    """
    Plot square and its diagonals with equations shown 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
    eq1 = format_equation(*line1)
    eq2 = format_equation(*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 with slight offsets to avoid overlap
  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()
```

Plot

Figure: Plot of square and its diagonals