9.2.11

Sai Krishna Bakki - EE25BTECH11049

Question

Find the area of the region bounded by the curve $y^2 = 9x$ and the lines $x = 2$ and $x = 4$ and the x-axis in the first quadrant.

Theoretical Solution

The general equation of a conic section is given by $\mathbf{x}^\top \mathbf{V}\mathbf{x} + 2\mathbf{u}^\top \mathbf{x} + f = 0$, where $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$. The parameters of the conic are

$$\mathbf{V} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} -9/2 \\ 0 \end{pmatrix}, \quad f = 0 \tag{1}$$

For the line x - 2 = 0, the parameters are

$$\mathbf{h}_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \quad \mathbf{m}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

The parameter κ for the points of intersection is found using the formula:

$$\kappa = \frac{1}{\mathbf{m}^\top \mathbf{V}\mathbf{m}} \left(-\mathbf{m}^\top (\mathbf{V}\mathbf{h} + \mathbf{u}) \pm \sqrt{[\mathbf{m}^\top (\mathbf{V}\mathbf{h} + \mathbf{u})]^2 - g(\mathbf{h})(\mathbf{m}^\top \mathbf{V}\mathbf{m})} \right) \tag{3}$$

where $g(\mathbf{h}) = \mathbf{h}^\top \mathbf{V}\mathbf{h} + 2\mathbf{u}^\top \mathbf{h} + f$.

Substituting the values into the formula for κ:

$$\kappa = \left(-0 \pm \sqrt{0^2 - (-18)(1)}\right) = 3\sqrt{2}, -3\sqrt{2} \tag{4}$$

yielding the points of intersection

$$\mathbf{a}_0 = \begin{pmatrix} 2 \\ 3\sqrt{2} \end{pmatrix}, \mathbf{a}_1 = \begin{pmatrix} 2 \\ -3\sqrt{2} \end{pmatrix} \tag{5}$$

For the line x - 4 = 0, the parameters are:

$$\mathbf{h}_2 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{6}$$

$$\kappa = \frac{1}{1}\left(-0 \pm \sqrt{0^2 - (-36)(1)}\right) = 6, -6 \tag{7}$$

yielding the points of intersection

$$\mathbf{a}_2 = \begin{pmatrix} 4 \\ 6 \end{pmatrix}, \mathbf{a}_3 = \begin{pmatrix} 4 \\ -6 \end{pmatrix} \tag{8}$$

Theoretical Solution

Thus, the area of the parabola in between the lines x = 2 and x = 4 is given by

$$A = \int_0^4 3\sqrt{x}\,dx - \int_0^2 3\sqrt{x}\,dx \tag{9}$$

$$= 16 - 4\sqrt{2} \tag{10}$$

Thus, the area of the specified region is 16 - 4 $\sqrt{2}$ square units.

C Code

```c
#include <math.h>

/**
 * @brief Defines the parabola y = 3*sqrt(x) for the first
     quadrant.
 * * @param x The x-coordinate.
 * @return The corresponding y-coordinate.
 */
double parabola_func(double x) {
    return 3.0 * sqrt(x);
}

/**
 * @brief Calculates the definite integral of the parabola
     function
 * using the trapezoidal rule.
 * * @param a The lower limit of integration.
 * @param b The upper limit of integration.
```

C Code

```c
 * @param n The number of trapezoids (steps) to use for the
     approximation.
 * @return The calculated area under the curve in the first
     quadrant.
 */
double trapezoidal_area(double a, double b, int n) {
    double h = (b - a) / n;
    // Initialize sum with the first and last terms of the
        trapezoidal rule
    double sum = 0.5 * (parabola_func(a) + parabola_func(b));

    // Add the intermediate terms
    for (int i = 1; i < n; i++) {
        sum += parabola_func(a + i * h);
    }

    return h * sum;
}
```

Python Code Uses ctypes

```
# Program to plot the area under a parabola using a C backend for
    calculation.
# Based on code by GVV Sharma
# Python script by user, C integration by Gemini

import numpy as np
import matplotlib.pyplot as plt
import ctypes
import os
import sys

# --- Local Imports Setup ---
# Update this path to the location of your 'CoordGeo' scripts

try:
    from libs.line.funcs import *
    from libs.triangle.funcs import *
    from libs.conics.funcs import *
```

Python Code Uses ctypes

```
except ImportError:
    print( )
# --- End Local Imports Setup ---

# --- 1. Compile and Load the C Library ---

# Define file names
c_source = area_lib.c
lib_name = area_lib.so

# Compilation command (for Linux/macOS). For Windows, this would
    be different.


# Load the compiled shared library
try:
    area_lib = ctypes.CDLL(os.path.abspath(lib_name))
except OSError as e:
```

Python Code Uses ctypes

```
    print(fError loading shared library: {e})
    sys.exit(1)


# --- 2. Define the C function signature for Python ---

# Get the function from the library
trapezoidal_area_c = area_lib.trapezoidal_area

# Specify the argument types (double, double, int)
trapezoidal_area_c.argtypes = [ctypes.c_double, ctypes.c_double,
    ctypes.c_int]
# Specify the return type (double)
trapezoidal_area_c.restype = ctypes.c_double


# --- 3. Define Parabola, Boundaries and Calculate Area ---
# The curve is y^2 = 9x
```

Python Code Uses ctypes

```
def parabola_x(y):
    Returns the x-coordinate for a given y on the parabola y^2 = 9x.
    return (y**2) / 9
# Boundaries
x_min = 2
x_max = 4
# Call the C function to get the area for the first quadrant
area_first_quadrant = trapezoidal_area_c(ctypes.c_double(x_min), ctypes.c_double(x_max), ctypes.c_int(1000))
# The total area is symmetric, so we double the result
total_area = 2 * area_first_quadrant
print(fThe calculated total area (from C function) is: {total_area})
# --- 4. Find Intersection Points for Plotting ---
y1 = np.sqrt(9 * x_min)
y2 = np.sqrt(9 * x_max)
a2 = np.array([x_max, y2])
a1 = np.array([x_min, y1])
```

```
a0 = np.array([x_min, -y1])
a3 = np.array([x_max, -y2])
points = np.vstack((a0, a1, a2, a3)).T
point_labels = ['$\\mathbf{a}_0$', '$\\mathbf{a}_1$', '$\\mathbf{
    a}_2$', '$\\mathbf{a}_3$']
# --- 5. Set up the Plot ---
fig = plt.figure()
ax = fig.add_subplot(111)

# Generate data for plotting
y_curve = np.linspace(-7, 7, 400)
x_curve = parabola_x(y_curve)
x_fill = np.linspace(x_min, x_max, 100)
y_fill_pos = np.sqrt(9 * x_fill)
y_fill_neg = -np.sqrt(9 * x_fill)
# Plot the elements
ax.plot(x_curve, y_curve, 'r', label='Parabola')
ax.plot([a0[0], a1[0]], [a0[1], a1[1]], color='dodgerblue', label
    ='Chord')
```

Python Code Uses ctypes

```
ax.plot([a3[0], a2[0]], [a3[1], a2[1]], color='darkorange', label
    ='Chord')
ax.fill_between(x_fill, y_fill_pos, y_fill_neg, color='cyan',
    label=f'Area $\\approx$ {total_area:.4f}')
ax.scatter(points[0, :], points[1, :], s=30, color='dimgray')
for i, txt in enumerate(point_labels):
    ax.annotate(txt, (points[0, i], points[1, i]), textcoords=
        offset points, xytext=(5,5), ha='center')
# --- 6. Formatting and Display ---
ax.spines['left'].set_position('zero')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
plt.xlim(-1, 7)
plt.ylim(-7, 7)
ax.grid(True)
ax.legend(loc='upper left') plt.show()
```

Python Code

```
# Program to plot the area under a parabola
# Based on code by GVV Sharma
# Revised to match a specific plot style.

import numpy as np
import matplotlib.pyplot as plt

# --- Local Imports Setup ---
import sys
# Update this path to the location of your 'CoordGeo' scripts
sys.path.insert(0, '/sdcard/github/matgeo/codes/CoordGeo')

# Local imports from your custom geometry library
# Note: These specific functions are not used in this area
    calculation,
# but the structure is included for consistency with your other
    projects.
try:
    from libs.line.funcs import *
```

Python Code

```
    from libs.triangle.funcs import *
    from libs.conics.funcs import *
except ImportError:
    print(Generating Plot)
# --- End Local Imports Setup ---
# --- 1. Define the Parabola and Bounding Lines ---
# The curve is y^2 = 9x
def parabola_x(y):
    Returns the x-coordinate for a given y on the parabola y^2 =
        9x.
    return (y**2) / 9
# Boundaries
x_min = 2
x_max = 4
# --- 2. Find Intersection Points ---
y1 = np.sqrt(9 * x_min) # y-coordinate at x=2
y2 = np.sqrt(9 * x_max) # y-coordinate at x=4
```

Python Code

```
# Points are labeled counter-clockwise from the top right
a2 = np.array([x_max, y2])
a1 = np.array([x_min, y1])
a0 = np.array([x_min, -y1])
a3 = np.array([x_max, -y2])

points = np.vstack((a0, a1, a2, a3)).T
point_labels = ['$\\mathbf{a}_0$', '$\\mathbf{a}_1$', '$\\mathbf{
    a}_2$', '$\\mathbf{a}_3$']

# --- 3. Set up the Plot ---
fig = plt.figure()
ax = fig.add_subplot(111)

# Generate y values for a smooth parabola curve
y_curve = np.linspace(-7, 7, 400)
x_curve = parabola_x(y_curve)
```

Python Code

```
# Generate x values for the shaded area
x_fill = np.linspace(x_min, x_max, 100)
y_fill_pos = np.sqrt(9 * x_fill)
y_fill_neg = -np.sqrt(9 * x_fill)

# --- 4. Plot the Elements ---

# Plot the parabola
ax.plot(x_curve, y_curve, 'r', label='Parabola')

# Plot the chords (vertical lines)
ax.plot([a0[0], a1[0]], [a0[1], a1[1]], color='dodgerblue', label
    ='Chord')
ax.plot([a3[0], a2[0]], [a3[1], a2[1]], color='darkorange', label
    ='Chord') # Second label is for legend entry

# Shade the area between the curves
ax.fill_between(x_fill, y_fill_pos, y_fill_neg, color='cyan',
    label='Area')
```

Python Code

```
# Plot and label the intersection points
ax.scatter(points[0, :], points[1, :], s=30, color='dimgray')
for i, txt in enumerate(point_labels):
    ax.annotate(txt, (points[0, i], points[1, i]), textcoords=
        offset points, xytext=(5,5), ha='center')
# --- 5. Formatting and Display ---
# Center the axes at (0,0)
ax.spines['left'].set_position('zero')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
# Set axis limits
plt.xlim(-1, 7)
plt.ylim(-7, 7)
ax.grid(True)
ax.legend(loc='upper left')
plt.show()
```

Plot By C code and Python Code

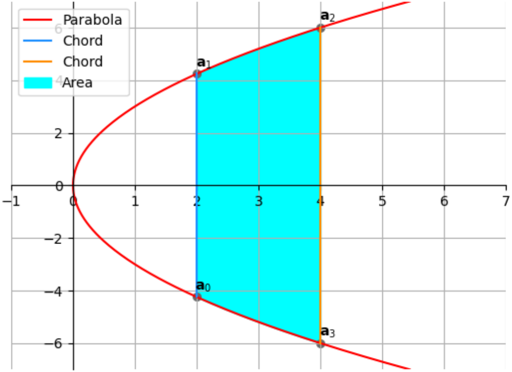

Figure: 1