Matgeo Presentation

Pushkar Gudla
AI24BTECH11012
IIT Hyderabad.

November 6, 2024

Contents

Problem Statement

The area of the region bounded by the curve $x^2 = 4y$ and the straight line $x = 4y - 2$ is

Setup and Variable Definitions

Variable	**Description**
$g(x)$	Equation of Conic
L	Equation of line
$\mathbf{h}$	A point on line L
$\mathbf{m}$	Direction vector of line L
$\mathbf{x_1}$ and $\mathbf{x_2}$	Points of intersection of L and $g(x)$

Table: Variables and given data

Converting the equations to Matrix form

To simplify the calculation of points of intersection, the curve $x^2 = 4y$ is represented in standard conic matrix form:

$$g(x) = x^\top \mathbf{V} x + 2\mathbf{u}^\top x + f = 0 \tag{3.1}$$

$$\mathbf{V} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{3.2}$$

$$\mathbf{u} = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \tag{3.3}$$

$$f = 0 \tag{3.4}$$

Parametric Form of the Line

$$L : \mathbf{x} = \mathbf{h} + k\mathbf{m} \tag{3.5}$$

where k is a scalar constant

$$\mathbf{h} = \begin{pmatrix} -2 \\ 0 \end{pmatrix} \tag{3.6}$$

$$\mathbf{m} = \begin{pmatrix} 1 \\ \frac{1}{4} \end{pmatrix} \tag{3.7}$$

$$\mathbf{x_i} = \mathbf{h} + k_i\mathbf{m} \tag{3.8}$$

Finding Points of Intersection

On substituting L in $g(x)$ we get a quadratic equation in k. We find that the discriminant is not equal to zero and thus we get two values k_1 and k_2. The point obtained by substituting k_1 in L is $\mathbf{x_1}$ and similarly on substituting k_2 in L we get $\mathbf{x_2}$.

$$k_1 = \frac{1}{\mathbf{m}^\top \mathbf{V}\mathbf{m}} \left(-m^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) + \sqrt{\left[\mathbf{m}^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) \right]^2 - g(\mathbf{h}) \left(\mathbf{m}^\top \mathbf{V}\mathbf{m} \right)} \right) \tag{3.9}$$

$$k_2 = \frac{1}{\mathbf{m}^\top \mathbf{V}\mathbf{m}} \left(-m^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) - \sqrt{\left[\mathbf{m}^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) \right]^2 - g(\mathbf{h}) \left(\mathbf{m}^\top \mathbf{V}\mathbf{m} \right)} \right) \tag{3.10}$$

On solving for k_1 and k_2 we get $k_1 = 4$ and $k_2 = 1$. Thus

$$\mathbf{x_1} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \tag{3.11}$$

$$\mathbf{x_2} = \begin{pmatrix} -1 \\ \frac{1}{4} \end{pmatrix} \tag{3.12}$$

Finding Area through Integration

The area between the curve and the line from $x = -1$ to $x = 2$ is calculated using the following integration:
Area $= \int_{-1}^{2} \left(\frac{x}{4} + \frac{1}{2} - \frac{x^2}{4} \right) dx$
Solving this integral, we get Area$=\frac{9}{8}$.
Hence, the area bound between the curve $x^2 = 4y$ and the line $x = 4y - 2$ is $\frac{9}{8}$.

C code to verify the Area I

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

// Define the functions for the curve and the line
double curve(double x) {
    return x * x / 4.0;  // y = x^2 / 4
}

double line(double x) {
    return (x + 2) / 4.0;  // y = (x + 2) / 4
}

// Numerical integration using the trapezoidal rule
double integrate(double (*f1)(double), double (*f2)(double), double a, double b, int n) {
    double h = (b - a) / n;  // Step size
    double area = 0.0;

    for (int i = 0; i < n; i++) {
        double x1 = a + i * h;
```

C code to verify the Area II

```c
        double x2 = a + (i + 1) * h;

        // Area of trapezoid between the functions
        double y1 = f2(x1) - f1(x1);
        double y2 = f2(x2) - f1(x2);

        area += 0.5 * (y1 + y2) * h;
    }

    return area;
}

int main() {
    // Define integration bounds
    double a = -1.0;
    double b = 2.0;
    int n = 10000;  // Number of intervals for better accuracy

    // Calculate area
    double area = integrate(curve, line, a, b, n);
```

C code to verify the Area III

```c

    // Write result to area.txt
    FILE *file = fopen("area.txt", "w");
    if (file != NULL) {
        fprintf(file, "Area between the curve and the line: %f\n", area);
        fclose(file);
        printf("Area has been written to area.txt\n");
    } else {
        printf("Error opening file!\n");
    }

    return 0;
}

```

Generating Parabola using C I

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

int main() {
    FILE *file;
    file = fopen("parabola_points.txt", "w");

    if (file == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Generate points for the parabola x^2 = 4y
    float x, y;
    float step = 0.1;  // Step size for x
    float x_max = 5;  // Maximum value of x (you can adjust as needed)

    for (x = -x_max; x <= x_max; x += step) {
        y = (x * x) / 4.0;
        fprintf(file, "%f %f\n", x, y);
```

Generating Parabola using C II

```
    }

    fclose(file);
    printf("Points have been written to parabola_points.txt\n");
    return 0;
}
}
```

Plotting the figure using Python I

```
import sys
sys.path.insert(0, '/home/pushkar/matgeo/codes/CoordGeo')
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from line.funcs import *
from conics.funcs import *
from triangle.funcs import *
import params
A = np.array([2,1])
B = np.array([-1, 0.25])
import matplotlib.pyplot as plt

# Initialize lists for x and y coordinates
x_vals = []
y_vals = []

# Read points from the file
```

Plotting the figure using Python II

```
with open("parabola_points.txt", "r") as file:
    for line in file:
        x, y = map(float, line.split())
        x_vals.append(x)
        y_vals.append(y)

# Plot the points
plt.plot(x_vals, y_vals, label="$x^2 = 4y$")
plt.xlabel('x')
plt.ylabel('y')
plt.title('Parabola: $x^2 = 4y$')

# use set_position
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')

#Plotting all line
```

Plotting the figure using Python III

```

# Define the range of y values
y = np.linspace(-1, 2, 100)

# Define the equation x = 4y - 2
x = 4 * y - 2

# Create the plot
plt.plot(x, y, label='x = 4y - 2')



#Labeling the coordinates
tri_coords = np.vstack((A,B)).T
plt.scatter(tri_coords[0,:], tri_coords[1,:])
vert_labels = ['A(2,1)','B(-1,0.25)']
for i, txt in enumerate(vert_labels):
       plt.annotate(txt,      # this is the text
                (tri_coords[0,i], tri_coords[1,i]), # this is the point
                ↪  to label
```

Plotting the figure using Python IV

```
                     textcoords="offset points",   # how to position the text
                     xytext=(0,10),      # distance from text to points (x,y)
                     ha='center')      # horizontal alignment can be left, right or center


#plt.fill_between(x1,x2,0,color='green', alpha=.2)
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid(True) # minor
plt.axis('equal')
plt.savefig('plot')

```

Figure