MATGEO Presentation

Dhawal
ee24btech11015,
IIT Hyderabad.

November 6, 2024

Problem Statement

Find the area enclosed by the parabola $4y = 3x^2$ and the line $2y = 3x + 12$.

Variable	**Description**	**Values**
P	Parabola	$4y = 3x^2$
L	Line	$2y = 3x + 12$
A	Point of intersection	To find
B	Point of intersection	To find

Table: Variables given

Matrix Equation

Parabola P in terms of matrix:

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

Where:

$$\mathbf{V} = \begin{pmatrix} 3 & 0 \\ 0 & 0 \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} 0 \\ -2 \end{pmatrix} \qquad f = 0 \tag{3.2}$$

Line L

$$L: \quad \mathbf{x} = \mathbf{h} + \kappa\mathbf{m} \quad \kappa \in \mathbb{R} \tag{3.3}$$

Where:

$$\mathbf{h} = \begin{pmatrix} 0 \\ 6 \end{pmatrix} \qquad \mathbf{m} = \begin{pmatrix} 1 \\ \frac{3}{2} \end{pmatrix} \tag{3.4}$$

Point of Intersection

Point of intersection of line L and parabola P:

$$\mathbf{x}_i = \mathbf{h} + \kappa_i \mathbf{m} \tag{3.5}$$

Where:

$$\kappa_i = \frac{1}{\mathbf{m}^\top \mathbf{V}\mathbf{m}} \left(-\mathbf{m}^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) \pm \sqrt{\left[\mathbf{m}^\top \left(\mathbf{V}\mathbf{h} + \mathbf{u} \right) \right]^2 - \mathrm{g}\left(\mathbf{h} \right) \left(\mathbf{m}^\top \mathbf{V}\mathbf{m} \right)} \right) \tag{3.6}$$

Finding $\mathrm{g}\left(\mathbf{h}\right)$:

$$\mathrm{g}\left(\mathbf{h} \right) = -24 \tag{3.7}$$

Finding κ_i:

$$\kappa_i = 4 \text{ and } -2 \tag{3.8}$$

So Points of intersection are:

$$\mathbf{A} = \begin{pmatrix} 4 \\ 12 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} -2 \\ 3 \end{pmatrix} \tag{3.9}$$

Area

Area between the curves:

$$\int_{-2}^{4} \left|1.5x + 6 - 0.75x^2\right| \, dx = 27 \tag{3.10}$$

So Area between the graphs is 27.

Plot

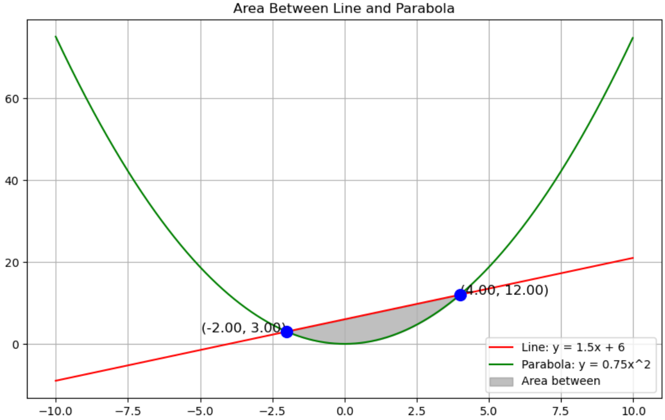

Figure: Area Enclosed by parabola and line.

C Code for generating points on line

```c
#include <stdio.h>

void generate_line_points(double m, double c, double x_start, double
    x_end, double step, double *x_vals, double *y_vals, int *n) {
    int i = 0;
    for (double x = x_start; x <= x_end; x += step) {
        x_vals[i] = x;
        y_vals[i] = m * x + c; // Equation: y = mx + c
        i++;
    }
    *n = i; // Number of points generated
}
```

C Code for generating points on parabola

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

// Function to generate points on the parabola y = a * x^2 + b * x + c
void generate_parabola_points(double a, double b, double c, double
    x_start, double x_end, double step, double* x_points, double*
    y_points) {
    int index = 0;
    for (double x = x_start; x <= x_end; x += step) {
        x_points[index] = x;
        y_points[index] = a * x * x + b * x + c;
        index++;
    }
}
}
```

Python Code for Plotting

```
import numpy as np
import matplotlib.pyplot as plt
import ctypes
from scipy.optimize import fsolve
from scipy.integrate import quad

line_lib = ctypes.CDLL('./line_points.so')
parabola_lib = ctypes.CDLL('./parabola_points.so')

line_lib.generate_line_points.argtypes = [ctypes.c_double, ctypes.c_double,
    ctypes.c_double, ctypes.c_double, ctypes.c_double, ctypes.POINTER(
    ctypes.c_double), ctypes.POINTER(ctypes.c_double)]
parabola_lib.generate_parabola_points.argtypes = [ctypes.c_double, ctypes.
    c_double, ctypes.c_double, ctypes.c_double, ctypes.c_double, ctypes.
    c_double, ctypes.POINTER(ctypes.c_double), ctypes.POINTER(ctypes
    .c_double)]
```

Python Code for Plotting

```
n_points = 1000
x_points = np.linspace(-10, 10, n_points)
y_line_points = np.zeros(n_points, dtype=np.float64)
y_parabola_points = np.zeros(n_points, dtype=np.float64)

x_start, x_end = -10.0, 10.0
step = (x_end - x_start) / n_points

line_lib.generate_line_points(1.5, 6.0, x_start, x_end, step, x_points.ctypes.
    data_as(ctypes.POINTER(ctypes.c_double)), y_line_points.ctypes.
    data_as(ctypes.POINTER(ctypes.c_double)))
parabola_lib.generate_parabola_points(0.75, 0, 0, x_start, x_end, step,
    x_points.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
    y_parabola_points.ctypes.data_as(ctypes.POINTER(ctypes.c_double)))
def line_eq(x):
    return 1.5 * x + 6 # Line equation: y = 1.5x + 6
```

Python Code for Plotting

```
def parabola_eq(x):
    return 0.75 * x ** 2 # Parabola equation: y = 0.75x^2
def equations(x):
    return parabola_eq(x) - line_eq(x)

# Find the intersection points using fsolve
x_intersect_1 = fsolve(equations, -5)[0] # First intersection, close to -5
x_intersect_2 = fsolve(equations, 5)[0] # Second intersection, close to 5

y_intersect_1 = line_eq(x_intersect_1) # y-coordinate of first intersection
y_intersect_2 = line_eq(x_intersect_2) # y-coordinate of second
    intersection
def integrand(x):
    return abs(parabola_eq(x) - line_eq(x))
area, _ = quad(integrand, x_intersect_1, x_intersect_2)
print(f"Area between the line and the parabola: {area:.4f}")
```

Python Code for Plotting

```
plt.figure(figsize=(10, 6))
plt.plot(x_points, y_line_points, 'r', label='Line: y = 1.5x + 6')
plt.plot(x_points, y_parabola_points, 'g', label='Parabola: y = 0.75x^2')
plt.fill_between(x_points, y_line_points, y_parabola_points,
                 where=((x_points >= x_intersect_1) & (x_points <=
                     x_intersect_2)),
                 color='gray', alpha=0.5, label='Area between')
plt.scatter([x_intersect_1, x_intersect_2], [y_intersect_1, y_intersect_2], color
    ='blue', s=100, zorder=5)
plt.text(x_intersect_1, y_intersect_1, f'({x_intersect_1:.2f}, {y_intersect_1:.2f
    })', fontsize=12, ha='right')
plt.text(x_intersect_2, y_intersect_2, f'({x_intersect_2:.2f}, {y_intersect_2:.2f
    })', fontsize=12, ha='left')
plt.legend()
plt.title('Area Between Line and Parabola')
plt.grid(True)
plt.show()
```