

Title: ***Applications of Definite Integrals in Real-Life Problems***

**Calculus Project:**

**Group Members:**

**231663\_Syed Ali Adan Naqvi**

**231666\_khawaja Ghayour Mumtaz**

**231591\_Ahmed Ameer**

**230957\_Tayyab Liaquat**

**231543\_Azeem Asghar**

**231567\_Muhib Khan**

# **Project Report: Applications of Definite Integrals in Real-Life Problems**

Abstract

In this project, we dive into the real-world applications of definite integrals through two practical scenarios: calculating fuel consumption and estimating consumer surplus. Using Python programming and mathematical models, we show how definite integrals can be applied to analyze and optimize fuel usage and to understand economic behavior. Key outcomes include total fuel consumption over time and consumer surplus derived from a demand curve, both supported by graphs for better understanding.

**Contents**

**1. Introduction**

**2. Purpose of the Project**

**3. Methodology** *3.1 Fuel Consumption Analysis  
 3.2 Consumer Surplus Estimation*

**4. Code Explanation** *1. Fuel Consumption Functions  
 2. Consumer Surplus Functions  
 3. Graph Plotting Functions  
 4. Integration with SciPy*

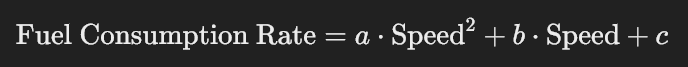
**5. Results and Discussion** *5.1 Fuel Consumption Analysis  
 5.2 Consumer Surplus Analysis*

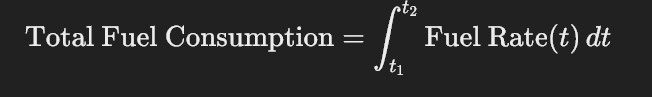
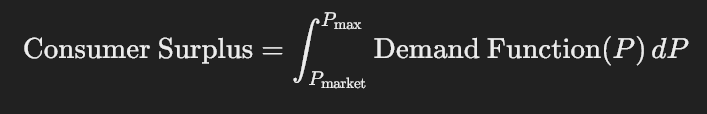
**6. Applications of Definite Integrals**

**7. Conclusion**

**8. References**

Introduction  
Definite integrals are a cornerstone of calculus, offering tools to measure accumulated quantities such as area, volume, and changes over a period. This project focuses on two distinct real-world applications: optimizing fuel consumption for vehicles and analyzing consumer surplus in economics. By modeling these problems mathematically and solving them with Python, we showcase the practicality and power of calculus.

1. Purpose of the Project  
   The main objectives of this project are:
   * To demonstrate how definite integrals can solve real-world problems.
   * To use Python programming for performing calculations and creating visualizations.
   * To highlight the relevance of calculus in fields like transportation and economics
2. Methodology  
   3.1 Fuel Consumption Calculation  
   Fuel consumption is modeled as a function of speed over time. The quadratic formula for the fuel consumption rate is:  
   ***`***The speed of a vehicle varies with time and is represented as a sinusoidal function:  
   ******

Using these functions, we calculate the total fuel consumption over a specific time interval by integrating the fuel rate:  
   
  
3.2 Consumer Surplus Estimation  
Consumer surplus is the difference between what consumers are willing to pay and what they actually pay. It is calculated as the area under the demand curve above the market price:  


The demand function used in this project is linear.

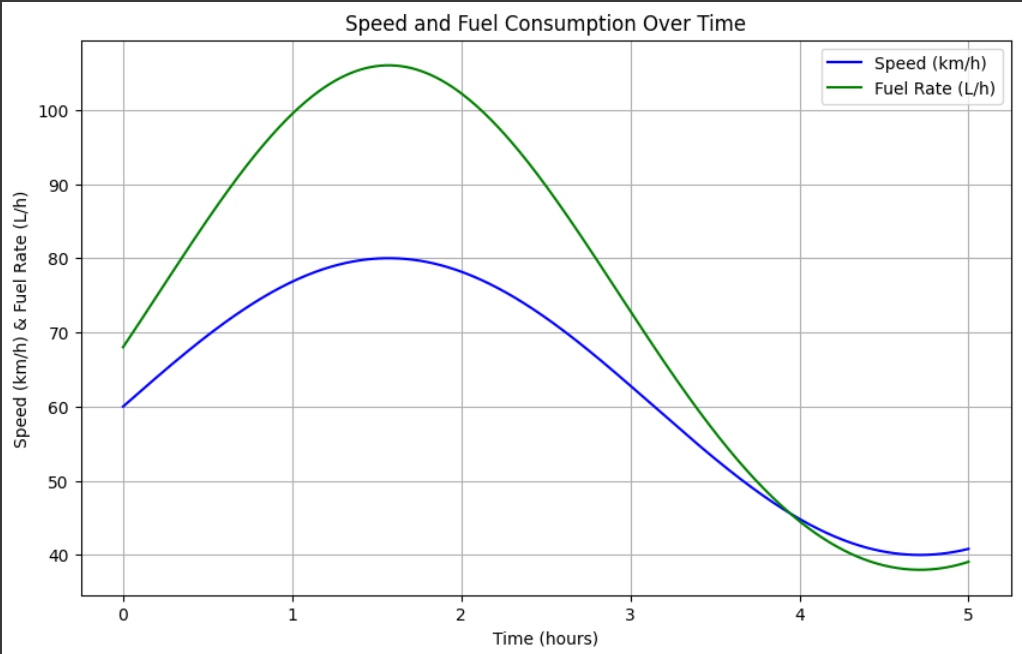
A number on a black background

Description automatically generated

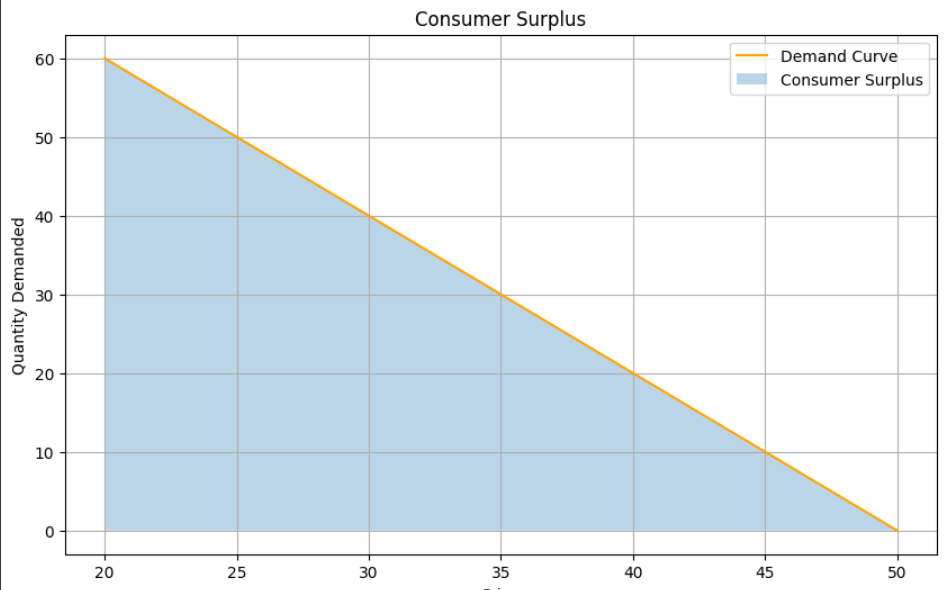
Here, **Q** represents the quantity demanded, and **P** is the price.

1. Code Explanation  
   The Python code developed for this project is broken down into key components:  
   1. Fuel Consumption Functions:  
    **• fuel\_consumption\_rate:** This function calculates fuel  
    consumption based on speed using a quadratic equation.  
    **•** **speed\_over\_time:** It models speed as a sinusoidal function  
    of time.  
    **•** **fuel\_rate\_over\_time:** Combine the above functions to compute  
    fuel rate over time.  
   2. Consumer Surplus Functions:  
    **•** **demand\_function:** Defines a linear demand curve.  
   3. Graph Plotting Functions:  
    **• plot\_fuel\_consumption:** Plots speed and fuel consumption rate  
    over time.  
    **•** **plot\_consumer\_surplus:** Visualizes the demand curve and highlight  
    the consumer surplus.  
   4. Integration with SciPy:  
     **•** The **quad** function from **scipy.integrate** is used to compute  
    integrals for fuel consumption and consumer surplus.
2. Results and Discussion  
   5.1 Fuel Consumption Analysis  
   Example Inputs for fuel Consumption Model:
   * **Start Time:** 0 hours
   * **End Time:** 5 hours
   * **Base speed:** 60 km/h
   * **Speed Variation Amplitude:** 20 km/h
   * **Speed Variation Frequency:** 1 cycle/hour
   * **Coefficient a for Fuel Rate:** 0.01
   * **Coefficient b for Fuel Rate:** 0.5
   * **Coefficient c for Fuel Rate:** 2

The total fuel consumed over the given time interval was calculated to be  
 approximately **374.9 liters.** The graph below shows the **speed** and **fuel** **consumption rate** varied over time.

  
 5.2 Consumer Surplus Analysis  
 Example Inputs for fuel Consumption Model:

* + **Market price:** 20$
  + **Max price:** 50$

The consumer surplus was calculated to be **900 units.** The graph below  
 illustrate the **demand curve** and the shaded area representing the surplus:  
 

6. **Applications of Definite Integrals:**

1. **Transportation:** Definite integrals help optimize fuel efficiency by analyzing consumption patterns over time.
2. **Economics:** Calculating consumer surplus enables businesses to better understand market dynamics and pricing strategies.

7. **Conclusion** This project demonstrated how definite integrals can solve real-world problems effectively. By integrating Python programming with calculus, we computed and visualized total fuel consumption and consumer surplus. These applications emphasize the importance of calculus as a practical tool for analysis and decision-making in various fields.

8. References:

1. Python Libraries:
   1. **Matplotlib (for plotting graphs)**
   2. **SciPy (for numerical integration)**
   3. **NumPy (for mathematical computations)**

.

**Code #::**

import math

from scipy.integrate import quad

import matplotlib.pyplot as plt

import numpy as np

# 1. Fuel Consumption Calculation

def fuel\_consumption\_rate(speed, a, b, c):

    return a \* speed\*\*2 + b \* speed + c

def speed\_over\_time(time, base\_speed, amplitude, frequency):

    return base\_speed + amplitude \* np.sin(frequency \* time)

def fuel\_rate\_over\_time(time, base\_speed, amplitude, frequency, a, b, c):

    speed = speed\_over\_time(time, base\_speed, amplitude, frequency)

    return fuel\_consumption\_rate(speed, a, b, c)

# 2. Consumer Surplus Calculation

def demand\_function(price):

    return 100 - 2 \* price  # Example demand curve: Q = 100 - 2P

# Graph Functions

def plot\_fuel\_consumption(base\_speed, amplitude, frequency, a, b, c, start\_time, end\_time):

    times = np.linspace(start\_time, end\_time, 500)

    speeds = speed\_over\_time(times, base\_speed, amplitude, frequency)

    fuel\_rates = fuel\_consumption\_rate(speeds, a, b, c)

    plt.figure(figsize=(10, 6))

    plt.plot(times, speeds, label="Speed (km/h)", color="blue")

    plt.plot(times, fuel\_rates, label="Fuel Rate (L/h)", color="green")

    plt.title("Speed and Fuel Consumption Over Time")

    plt.xlabel("Time (hours)")

    plt.ylabel("Speed (km/h) & Fuel Rate (L/h)")

    plt.legend()

    plt.grid()

    plt.show()

def plot\_consumer\_surplus(price\_market, price\_max):

    prices = np.linspace(price\_market, price\_max, 500)

    quantities = demand\_function(prices)

    plt.figure(figsize=(10, 6))

    plt.plot(prices, quantities, label="Demand Curve", color="orange")

    plt.fill\_between(prices, quantities, alpha=0.3, label="Consumer Surplus")

    plt.title("Consumer Surplus")

    plt.xlabel("Price")

    plt.ylabel("Quantity Demanded")

    plt.legend()

    plt.grid()

    plt.show()

# Main function to calculate both applications

def main():

    while True:

        print("\nWhat do you want to calculate?")

        print("1. Fuel Consumption")

        print("2. Consumer Surplus")

        print("3. Both")

        print("4. Exit")

        choice = int(input("Enter your choice (1, 2, 3, or 4): "))

        if choice == 1 or choice == 3:

            # Fuel Consumption Inputs

            print("\n=== Fuel Consumption Calculation ===")

            start\_time = float(input("Enter start time (hours): "))

            end\_time = float(input("Enter end time (hours): "))

            base\_speed = float(input("Enter base speed (km/h): "))

            amplitude = float(input("Enter speed variation amplitude: "))

            frequency = float(input("Enter speed variation frequency: "))

            a = float(input("Enter coefficient a for fuel rate: "))

            b = float(input("Enter coefficient b for fuel rate: "))

            c = float(input("Enter coefficient c for fuel rate: "))

            # Calculate total fuel consumption

            total\_fuel, \_ = quad(

                fuel\_rate\_over\_time, start\_time, end\_time,

                args=(base\_speed, amplitude, frequency, a, b, c)

            )

            print(f"\nTotal fuel consumed over {end\_time - start\_time} hours: {total\_fuel:.2f} liters")

            # Plot fuel consumption graph

            plot\_fuel\_consumption(base\_speed, amplitude, frequency, a, b, c, start\_time, end\_time)

        if choice == 2 or choice == 3:

            # Consumer Surplus Inputs

            print("\n=== Consumer Surplus Calculation ===")

            price\_market = float(input("Enter market price: "))

            price\_max = float(input("Enter maximum price consumers are willing to pay: "))

            # Check if market price exceeds maximum price

            if price\_market > price\_max:

                print("Consumer Surplus: 0 units (No transaction occurs as market price is higher than what consumers are willing to pay).")

            else:

                # Calculate consumer surplus

                consumer\_surplus, \_ = quad(demand\_function, price\_market, price\_max)

                # Ensure consumer surplus is not negative

                if consumer\_surplus < 0:

                    consumer\_surplus = 0

                    print("Consumer Surplus is adjusted to 0 units (Negative surplus isn't possible).")

                print(f"Consumer Surplus: {consumer\_surplus:.2f} units")

                # Plot consumer surplus graph

                plot\_consumer\_surplus(price\_market, price\_max)

        if choice == 4:

            print("Exiting program. Goodbye!")

            break

        if choice not in [1, 2, 3, 4]:

            print("Invalid choice. Please select 1, 2, 3, or 4.")

# Run the main function

if \_\_name\_\_ == "\_\_main\_\_":

    main()