Lab 1: Curve Fitting & Numerical Integration

Name: Shreyansh Agarwal
Roll: B23PH1024

Aim:- Take a dataset & fit a curve to it. Then find area under that curve by numerical integration.

Theory:-

I. **Curve fitting:-** It is process of finding mathematical function that best describes relation between datapts.

II. **Numerical Integration:-** Method of approximating value of definite integral when an exact solution is not known.

Dataset:- Simple Pendulum data from Kaggle.

Procedure:- 1. Curve Fitting

1. Linear Fit by known solution:-

a) $f(x) = y = ax + b$ where we need to find a, b s.t. error,

$$e = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

is minimum, i.e

$$\frac{\partial e}{\partial a} = 0 \quad \frac{\partial e}{\partial b} = 0$$

b) $\frac{\partial e}{\partial a} = -2\sum_i (y_i - ax_i - b)x_i = 0$

$\frac{\partial e}{\partial b} = -2\sum_i (y_i - ax_i - b) = 0$

c) $\therefore a\sum x_i^2 + b\sum x_i = \sum y_i x_i$

$a\sum x_i + bn = \sum y_i$

d) Solution of these 2 eqn gives,

$$a = \frac{n\sum x_i y_i - \sum y_i \sum x_i}{n\sum x_i^2 - \left[\sum x_i\right]^2}$$

$$b = \frac{\sum y_i - a\sum x_i}{n}$$

11. Quadratic Fit by Matrices:-

a) $f(x) = ax^2 + bx + c$, $e = \sum_i (y_i - ax_i^2 - bx_i$

-c). We need to solve for a, b, c s.t.

$$\frac{\partial e}{\partial a} = 0 \quad \frac{\partial e}{\partial b} = 0 \quad \frac{\partial e}{\partial c} = 0$$

b) $\frac{\partial e}{\partial a} = -2\sum_i y_i x_i^2 - ax_i^4 - bx_i^3 - cx_i^2 = 0$

$\frac{\partial e}{\partial b} = -2\sum_i y_i x_i - ax_i^3 - bx_i^2 - cx_i = 0$

$\frac{\partial e}{\partial c} = -2\sum_i y_i - ax_i^2 - bx_i - c = 0$

c) $\therefore \begin{bmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum y_i x_i^2 \\ \sum y_i x_i \\ \sum y_i \end{bmatrix}$

d) Let us take this as: $AX = B$

$\therefore X = A^{-1}B$

iii. Exponential Fit using Omptimization :-

a) $f(x) = Ae^{Bx} + C$

$E = \sum_i (y_i - Ae^{Bx_i} - C)^2$

We need A,B,C s.t. E is minim

b) Take some initial value of A,B,C

c) Compute error at this A,B,C

d) Iterate & update.

$A = A - \eta \frac{\partial E}{\partial A}$

$B = B - \eta \frac{\partial E}{\partial B}$

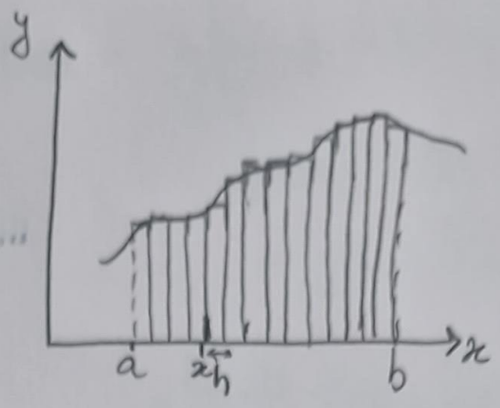
$C = C - \eta \frac{\partial E}{\partial C}$

e) Stop after a number of iterations or when $E <$ Threshold or E is not changing much.

2. Numerical Integration

1. Rectangular Integration :-

a) Given any curve divide it in steps of $h = \frac{b-a}{n}$ i.e. n rectangle as shown.



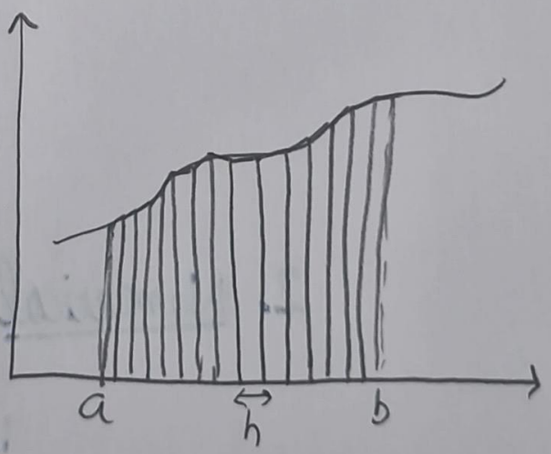
b) Evaluate f at mid of each division i.e.,

$$f\left(\frac{x + x+h}{2}\right)$$

c) Area $= h \times \sum_{i=1}^{n} f\left(\frac{x+x+h}{2}\right) = I$

11. Trapezoidal Integration :-

a) Divide curve in trapezoids as follows :-



$$h = \frac{b-a}{n}$$

b) Evaluate f at both pts. of trapezoid

$f(x)$ & $f(x+h)$

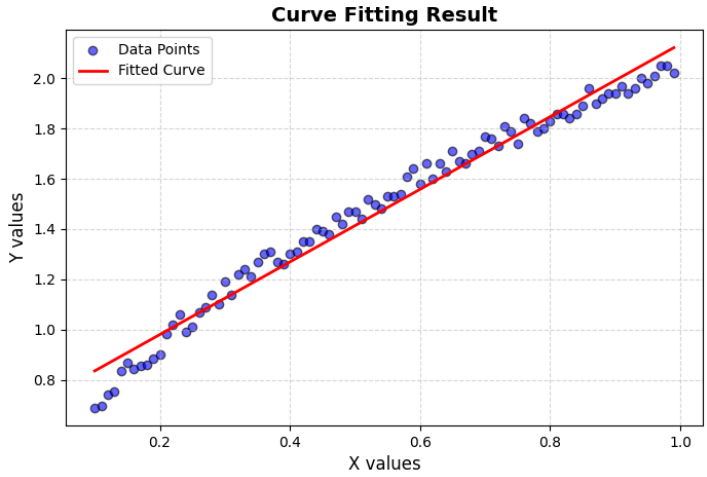
c) $I = \text{Area} = \frac{h}{2}\left[\left[f(a) + f(a+h)\right] + \left[f(a+h) + f(a+2h) + \cdots + \left[f(b-h) + f(b)\right]\right]\right.$

$$= \frac{h}{2} \sum_{i=1}^{n-1} f(x) + f(x+h)$$

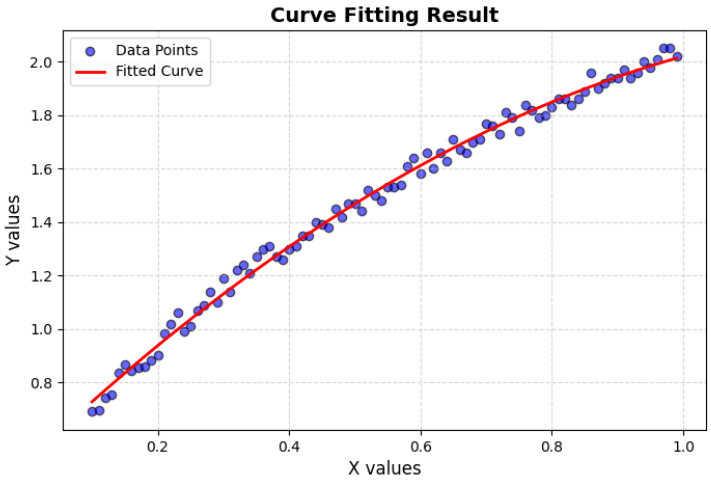
Result :- I. ~~Integration~~ Quadratic curve for data was best fit.

II. Integration of the quadratic curve by rectangular = 1.32

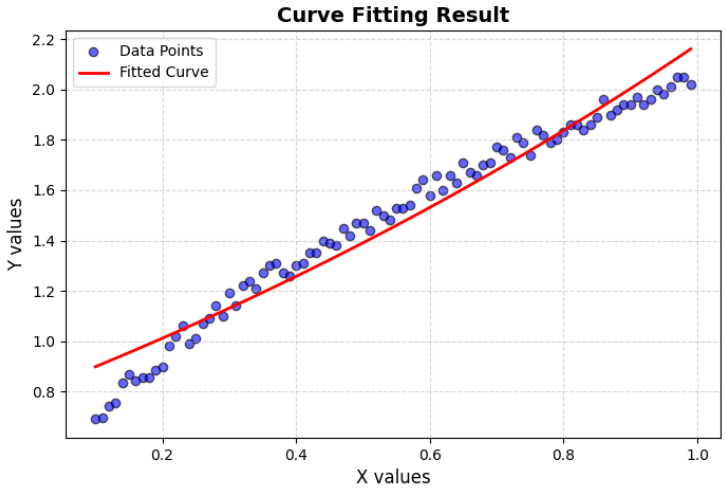
III. Integration of curve by trapezoidal = 1.32



a) Linear Fit



b) Quadratic Fit



c) Exponential Fit

Dataset: https://www.kaggle.com/datasets/abhi8923shriv/simple-pendulum-dataset-length-vs-time/data

Lab1-CurveFitting.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def linear_fit(x,y):
    n = len(x)
    xi = yi = xiyi = xi_sq = 0

    for i in range(n):
        xi += x[i]
        yi += y[i]
        xiyi += x[i]*y[i]
        xi_sq += x[i]**2

    a = ((n*xiyi) - (xi*yi))/((n*xi_sq) - (xi**2))
    b = (yi - (a*xi))/n

    return a,b

def quadratic_fit(x,y):
    n = len(x)
    xi = yi = xiyi = xi_sq = xi_3 = xi_4 = yixi_2 = 0

    for i in range(n):
        xi += x[i]
        yi += y[i]
        xiyi += x[i]*y[i]
        xi_sq += x[i]**2
        xi_3  += x[i]**3
        xi_4 += x[i]**4
        yixi_2 += y[i]*(x[i]**2)

    A = np.array([[xi_4, xi_3, xi_sq],
                  [xi_3, xi_sq, xi],
                  [xi_sq, xi, n]])
    A_inv = np.linalg.inv(A)
    B = np.array([yixi_2, xiyi, yi])

    X = A_inv @ B

    return X[0], X[1], X[2]

def exp_fit(x,y, eta = 0.1, iterations = 10000):
    A,B,C = 1,1,0
    n = len(x)
    E_prev = np.inf

    for it in range(iterations):
```

```
        exp_term = np.exp(np.clip(B * x, -1e6, 1e6))
        y_pred = A * exp_term + C
        error = y - y_pred
        E = np.mean(error**2)

        dA = -2 * np.mean(error * exp_term)
        dB = -2 * np.mean(error * A * exp_term * x)
        dC = -2 * np.mean(error)

        A -= eta * dA
        B -= eta * dB
        C -= eta * dC

        if np.abs(E - E_prev) <= 1e-8:
            break

        E_prev = E

    return A, B, C

def rectangular_integration(low, high, a,b,c=0, curve="linear"):
    if curve=="linear":
        def f(x):
            return a*x+b
    elif curve=="quadratic":
        def f(x):
            return a*(x**2) + b*x + c
    if curve=="exp":
        def f(x):
            return a*np.exp(b*x)+c

    h = 0.1
    n = int((high-low)/h)
    h = (high-low)/n

    y = 0
    x = low
    for _ in range(n):
        y += f((x+x+h)/2)
        x+=h

    return h*y

def trapezoidal_integration(low, high, a,b,c=0, curve="linear"):
    if curve=="linear":
        def f(x):
            return a*x+b
    elif curve=="quadratic":
        def f(x):
            return a*(x**2) + b*x + c
```

```
    if curve=="exp":
        def f(x):
            return a*np.exp(b*x)+c

    h = 0.1
    n = int((high-low)/h)
    h = (high-low)/n

    y = 0
    x = low
    for _ in range(n-1):
        y += f(x)+f(x+h)
        x+=h

    return h*y/2

data = pd.read_csv("Simple pendulum data.csv")

x = np.array(data["length(l)"])
y = np.array(data["time(t)"])

result = {"linear":{"a": 0, "b": 0}, "quadratic":{"a": 0, "b": 0, "c": 0}, "exp":{"a": 0, "b": 0, "c": 0}}

a,b = linear_fit(x,y)
result["linear"]["a"] = a
result["linear"]["b"] = b
y_pred = a*x+b

plt.figure(figsize=(8, 5))
plt.scatter(x, y, color='blue', alpha=0.6, edgecolor='k', label='Data Points')
plt.plot(x, y_pred, color='red', linewidth=2, label='Fitted Curve')
plt.xlabel("X values", fontsize=12)
plt.ylabel("Y values", fontsize=12)
plt.title("Curve Fitting Result", fontsize=14, fontweight='bold')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()

a,b,c = quadratic_fit(x,y)
result["quadratic"]["a"] = a
result["quadratic"]["b"] = b
result["quadratic"]["c"] = c
y_pred = a*(x**2) + b*x + c

plt.figure(figsize=(8, 5))
plt.scatter(x, y, color='blue', alpha=0.6, edgecolor='k', label='Data Points')
plt.plot(x, y_pred, color='red', linewidth=2, label='Fitted Curve')
plt.xlabel("X values", fontsize=12)
plt.ylabel("Y values", fontsize=12)
```

```
plt.title("Curve Fitting Result", fontsize=14, fontweight='bold')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()

a,b,c = exp_fit(x,y)
result["exp"]["a"] = a
result["exp"]["b"] = b
result["exp"]["c"] = c
y_pred = a*np.exp(b*x) + c

plt.figure(figsize=(8, 5))
plt.scatter(x, y, color='blue', alpha=0.6, edgecolor='k', label='Data Points')
plt.plot(x, y_pred, color='red', linewidth=2, label='Fitted Curve')
plt.xlabel("X values", fontsize=12)
plt.ylabel("Y values", fontsize=12)
plt.title("Curve Fitting Result", fontsize=14, fontweight='bold')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.show()

print(f"Area under curve using rectangular integration: {rectangular_integration(min(x), max(x), result['quadratic']['a'], result['quadratic']['b'], result['quadratic']['c'], 'quadratic')}")
print(f"Area under curve trapezoidal  integration: {rectangular_integration(min(x), max(x), result['quadratic']['a'], result['quadratic']['b'], result['quadratic']['c'], 'quadratic')}")
```