International Islamic University Chittagong

Department of Computer Science and Engineering

Final Term Examination Autumn 2022

Project Report

Submitted by	Team_Integration
Semester	7th
Section	7BM
Course Code	CSE-4746
Course Title	Numerical Method Sessional
Project Topic	Numerical Integration
Team members	Md Jubair Hossain_ C193051 Mohammad Sakib Chowdhury_C193055 **Mainuddin Hasan_C193070(Leader)** Jahirul Islam_C193083 Mohammad Mezbah Uddin_C193082
Submitted to	Prof. Mohammed Shamsul Alam
Date of Submission	09th June 2023

Introduction

Numerical methods are mathematical techniques used to solve problems that cannot be solved analytically. They are an essential tool in many fields, including engineering, science, and finance. In this project, calculate the approximating work of an object done by a variable force using numerical integration. In this scenario, we are considering an object that is being moved along a straight line by a force that varies with position. The force applied to the object is given by the function $F(x) = 3x^2 - 2x + 1$, where x represents the position of the object in meters and the force is measured in Newtons. The task at hand is to calculate the work done by the force as the object is moved from the initial position of $x = 1$ to $x = 5$ meters. To achieve this, we will employ numerical integration methods.

Features

1. Position-Dependent Force: The force acting on the object varies with its position along the straight line. This allows for a more realistic representation of how forces can change depending on the object's location.

2. Numerical Integration: To calculate the work done by the force, we will utilize numerical integration techniques. These methods approximate the area under the force-position curve, which corresponds to the work done by the force.

Tools

To perform the numerical integration, we can make use of various programming languages and libraries that provide numerical computation capabilities. Some commonly used tools include:

1. Codeblocks: Code::Blocks is an open-source integrated development environment (IDE) that supports multiple programming languages. It provides a user-friendly interface and powerful features for code editing, building, and debugging. Code::Blocks is available for Windows, macOS, and Linux platforms.

2. C++: C++ is a popular programming language for scientific and engineering computations and offers built-in numerical integration functions.

Implementation

1. Define the Force Function: Start by defining the force function $F(x) = 3x^2 - 2x + 1$, which represents the force applied to the object at a given position x.

2. Choose Integration Method: Select three appropriate numerical integration methods for calculating the work done. Examples include the trapezoidal rule, Simpson's rule ⅓, and Simpson's rule ⅜.

3. Discretize the Interval: Divide the interval between the initial and final positions (1 to 5 meters) into smaller subintervals or (10 to 15 meters). The more subintervals used, the more accurate the integration result.

4. Evaluate the Force at Each Subinterval: Calculate the force at each subinterval by substituting the corresponding position values into the force function F(x).

5. Perform Numerical Integration: Apply the chosen integration method to approximate the work done by summing up the contributions from each subinterval.

6. Obtain the Work Done: The final result of the numerical integration will provide an estimate of the work done by the force as the object is moved from x = 1 to x = 5 meters or any interval.

Description of choosing method

Trapezoidal Rule

The trapezoidal rule can be written as

$I = h/2 [y_0 + 2 (y_1 + y_2 + \ldots\ldots\ldots + y_{n-1}) + y_n]$

The above formula is known as the *trapezoidal rule* for numerical integration.

- ☐ The geometrical significance of this rule is that the curve $y = f(x)$ is replaced by n straight line joining the points (x_0, y_0) and (x_1, y_1), (x_1, y_1) and (x_2, y_2), …. , (x_{n-1}, y_{n-1}) and (x_n, y_n).
- ☐ The area bounded by the curve $y = f(x)$, the ordinates $x = x_0$ and $x = x_n$ and the x axis is then approximately equivalent to the sum of the areas of the n trapezium obtained.

Simpson's ⅓ Rule

$I = h/3 [y_0 + 4 (y_1 + y_3 + y_5 + \ldots + y_{n-1}) + 2 (y_2 + y_4 + \ldots + y_{n-2}) + y_n]$

The above formula is known as Simpson's one-third rule on simply Simpson's rule.

- It should be noted that this rule requires the division the whole range into an even number of subintervals of width h.

Simpson's ⅜ Rule

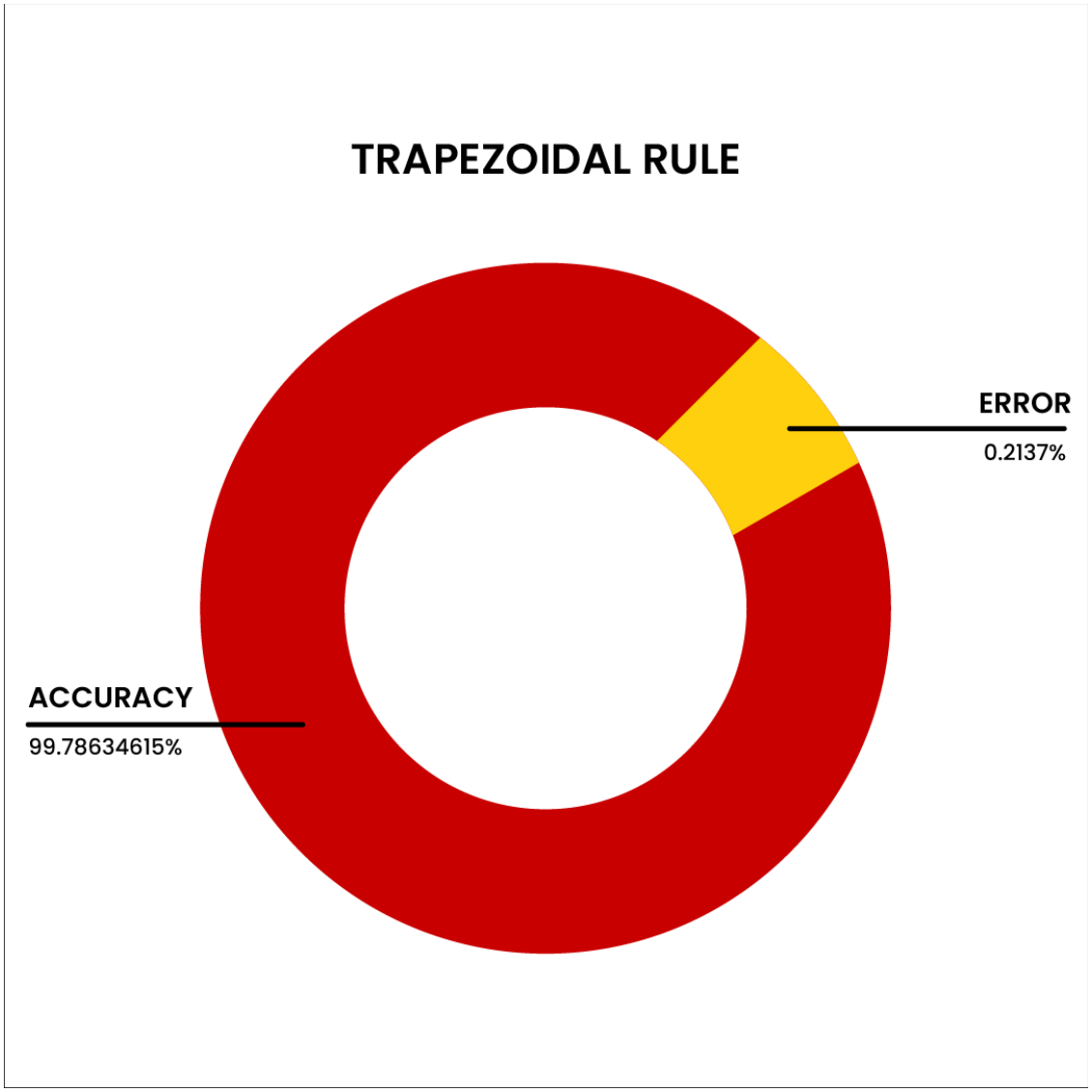
$$I = 3h/8 [(y_0 + y_n) + 3 (y_1 + y_2 + y_4 + y_5 + ...+ y_{n-2} + y_{n-1}) + 2 (y_3 + y_6 + ... + y_{n-3})]$$

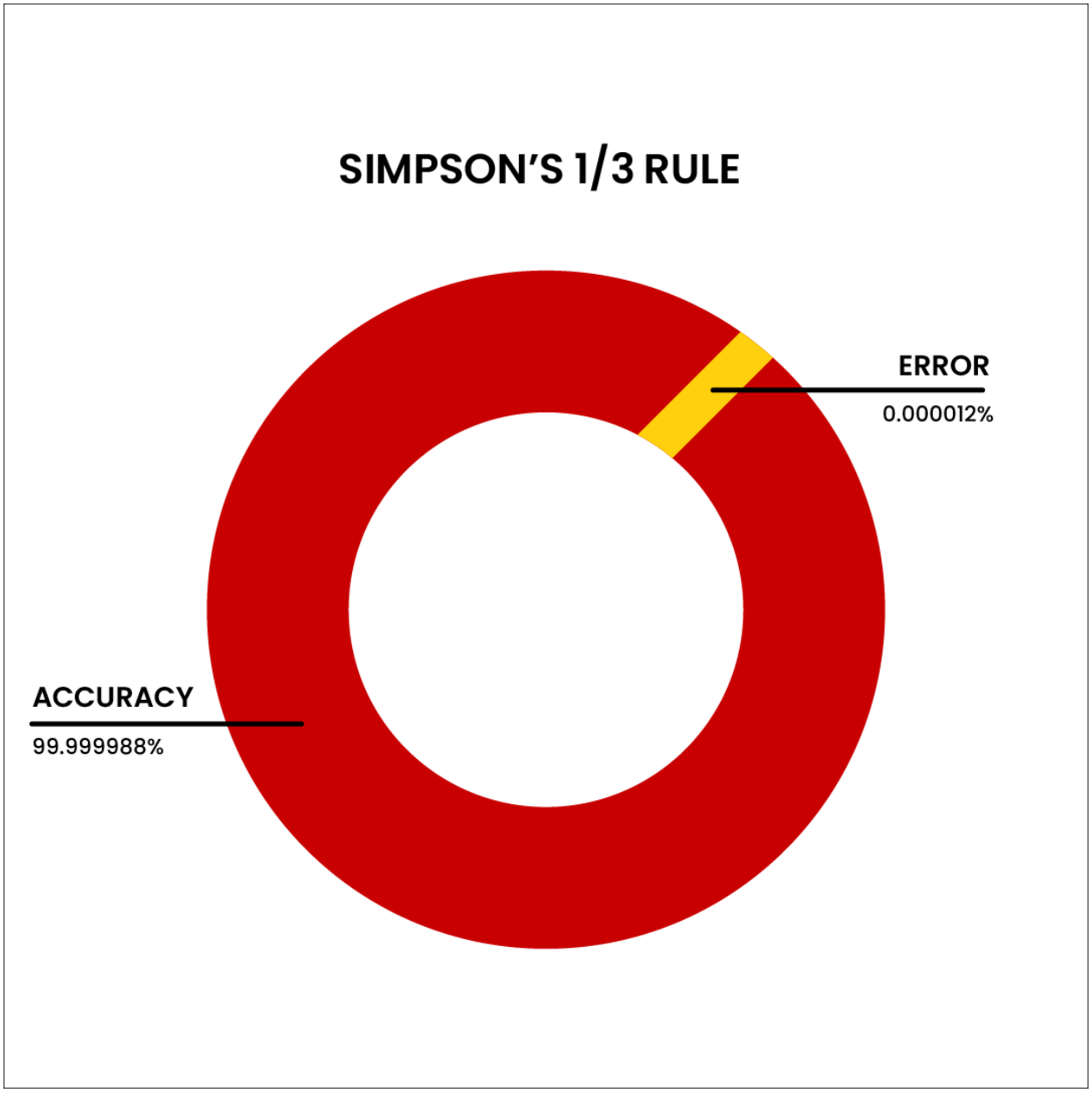
The above formula is known as Simpson's Three-by-Eight rule on simply Simpson's rule.

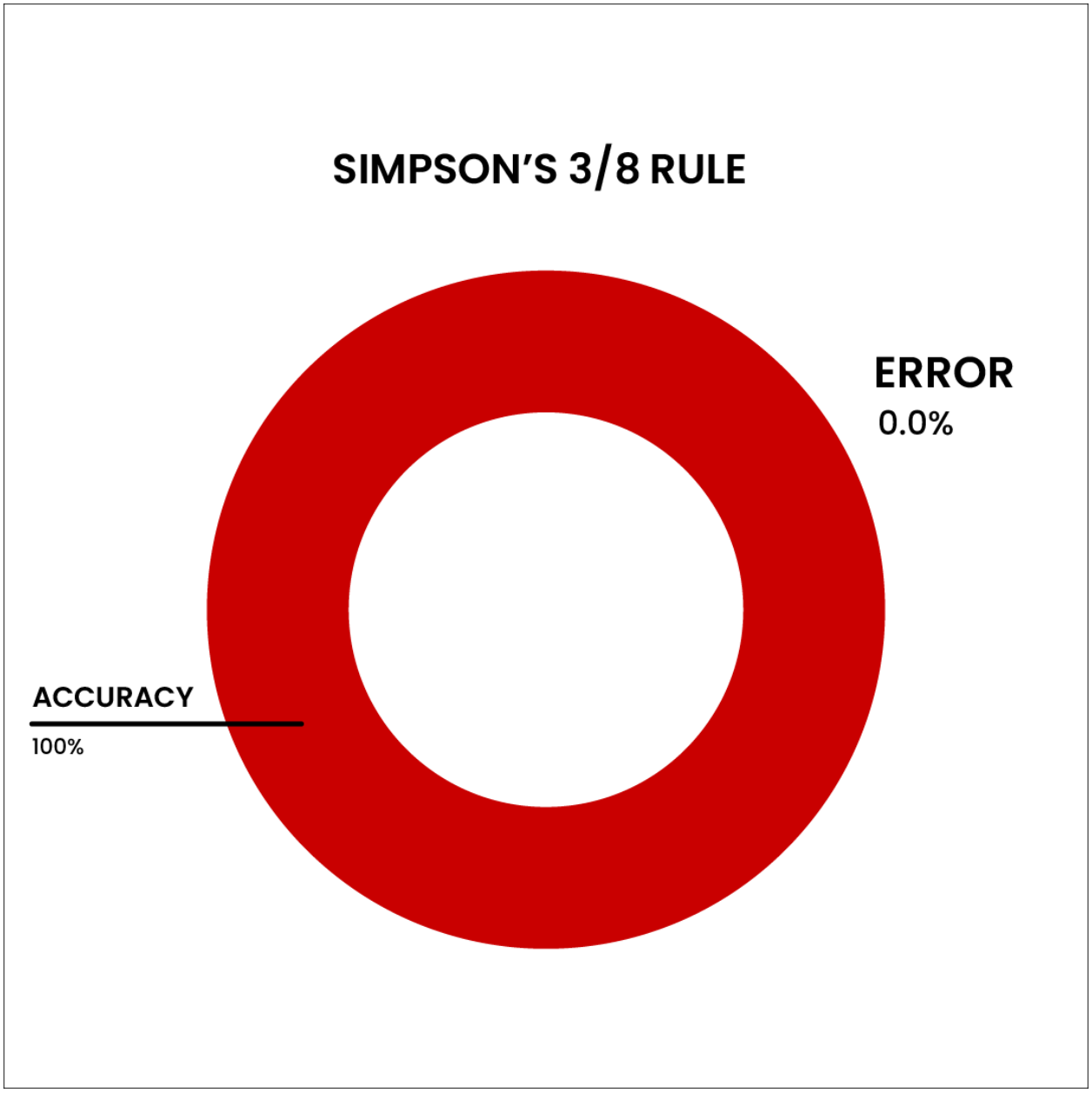
- Simpson's 3/8 rule can be applied when the range [a, b] is divided into a number of subintervals, which must be a multiple of 3.

Result Analysis

Initial, End position (Meter)	Work Done - Real integration (Joule)	Work Done - Trapezoidal rule (Joule)	Work Done - Simpson's ⅓ rule (Joule)	Work Done - Simpson's ⅜ rule (Joule)
1 - 5	104J	104.222J	104.00012J	104.000J
5 - 15	3060J	3067.81J	3060.00032J	3060.00J
10 - 50	121640J	122140.3241J	121640.00034J	121640.00J


TRAPEZOIDAL RULE
ERROR
0.2137%
ACCURACY
99.78634615%


SIMPSON'S 1/3 RULE
ERROR
0.000012%
ACCURACY
99.999988%


SIMPSON'S 3/8 RULE
ERROR
0.0%
ACCURACY
100%

Code:

```
#include<bits/stdc++.h>
using namespace std;
double force(double x)
{
    return (3*x*x) - (2*x) + 1;
}
double trapezoidalRule(double a, double b, int n)
{
    double h = (b - a) / (double)n;
    double sum = force(a) + force(b);

    for (int i = 1; i < n; i++)
    {
        double x = a + (i * h);
        sum += 2.0 * force(x);
    }

    return (h /2.0) * sum;
}

double simpsons13Rule(double a, double b, int n)
{
    double h = (b - a) /(double)n;
    double sum = force(a) + force(b);
    for (int i = 1; i < n; i ++)
    {
        double x = a + (i * h);
        if(i%2!=0)
        {
            sum += 4.0 * force(x);
        }
        else
```

```
        {
            sum += 2.0 * force(x);
        }
    }
    return (h / 3.0) * sum;
}

double simpsons38Rule(double a, double b, int n)
{
    double h = (b - a) / (double)n;
    double sum = force(a) + force(b);

    for (int i = 1; i < n; i++)
    {
        double x = a + (i * h);
        if(i%3==0)
        {
            sum += 2.0 * force(x);
        }
        else
        {
            sum += 3.0 * force(x);
        }
    }
    return ((3.0 * h) / 8.0) * sum;
}
int main()
{
    double a = 1.0;
    double b = 5.0;
    int n1 = 9;
    int n2 = 14;
    int n3 = 18;
```

```
    double workTrapezoidal = trapezoidalRule(a, b, n1);
    double workSimpsons13 = simpsons13Rule(a, b, n2);
    double workSimpsons38 = simpsons38Rule(a, b, n3);
    cout << "Work using Trapezoidal rule   : "<<setprecision(12) << fixed
<< workTrapezoidal  << " Joule"  << endl;
    cout << "Work using Simpson's 1/3 rule : "<<setprecision(12) << fixed
<< workSimpsons13 << " Joule"  << endl;
    cout << "Work using Simpson's 3/8 rule : "<<setprecision(12) << fixed
<< workSimpsons38 << " Joule"  << endl;
    return 0;
}
```

Future Plans & Possible Extensions

1. Analysis of Energy: Explore the relationship between work done and the energy of the system. Investigate how the work done by the force affects the kinetic and potential energy of the object. By doing so, we can observe the oil consumption of a company's daily vehicles.

2. Optimization of Integration Methods: Investigate and compare the accuracy and efficiency of different numerical integration methods for calculating work done by forces. Optimize the choice of the method based on specific scenarios.

Conclusion

By utilizing numerical integration methods, we can calculate the work done by a position-dependent force as an object is moved from one position to another. In this case, we examined a force function $F(x) = 3x^2 - 2x + 1$ and evaluated the work done as the object moved from $x = 1$ to $x = 5$ meters. The application of numerical integration provides a practical approach for approximating the work done by the force, allowing for a deeper understanding of the system dynamics and energy transformations.