Numerical Methods for Integration and Differentiation

Course : CPCS-212 Applied Math for Computing

Group Members :

- Hassan B Al-Ghamdi
- Faisal M Al-Suhaimi
- Yazan W Al-Karmo

Submission Date : 26-Sep-2024

Contents

Abstract

This project is discussing the use of numerical methods for Differentiation and Integration. Using Forward, Backward and Central Difference methods, and Trapezoidal, Simpson's Rule and Monte Carlo Integration. We evaluated the performance of these methods on some mathematical functions. Results shows that Central Difference is the most accurate differentiation, While Simpson's Rule outperforms other integration methods in terms of accuracy.

Introduction

The goal of this project is to implement and compare numerical methods for differentiation and integration, evaluating their accuracy against analytical solutions. Numerical methods are essential when exact solutions are difficult or impossible to find, or when solving them is too time-consuming.

Differentiation methods, such as Forward, Backward, and Central Difference, approximate derivatives using discrete points. Integration methods, including Trapezoidal Rule, Simpson's Rule, and Monte Carlo, estimate areas under curves. These techniques are widely used in fields like physics, engineering, and data analysis for solving complex problems.

In this project, we analyze these methods by applying them to various functions, testing their performance, and identifying their strengths and limitations.

Numerical Differenetiation

Functions

These are the functions we will use as an example for both numerical differentiation and integration.

$F(x) = x^4 - 6x^2 + x + 2$

$G(x) = \sin^3(x) - \cos^2(x)$

$H(x) = e^{-x^2}$

These functions allow us to test the effectiveness of numerical methods under different conditions.

Numerical Differentiation

Numerical Differentiation methods are used for approximating derivatives when analytical solutions are unavailable or impractical.

There are Three well-known numerical methods for differentiation :

The **Forward Difference** method estimates the derivative of a function at a point xxx by looking at how the function value changes between x and a point slightly ahead of it (x+h).

$$\text{Forward Difference} \approx \frac{f(x+h)-f(x)}{h}$$

The **Backward Difference** method estimates the derivative by looking at how the function value changes between x and a point slightly behind it (x−h).

$$\text{Backward Difference} \approx \frac{f(x)-f(x-h)}{h}$$

The **Central Difference** method estimates the derivative by averaging the forward and backward differences. It compares the function values at points equally spaced ahead (x+ h) and behind (x−h).

$$\text{Central Difference} \approx \frac{f(x+h)-f(x-h)}{2h}$$

The Implementation of each method in MATLAB:

Forward Difference

```
function r = Forward_difference(func, x_val, step_size)
    % func : the function to be diffrentiated
    % x_val : array for x values
    % step_size : array for h values
    func_handle = str2func(['@(x)' char(func)]); % Convert string to function handle
    fx_plus_h = func_handle(x_val + step_size); % calculate f(x+h)
    fx = func_handle(x_val); % calculate f(x)
    r = (fx_plus_h - fx) ./ step_size; % evaluate the forward difference and return the value
end
```

Backward Difference

```
function r = Backward_difference(func,x_val, step_size)
    % func : the function to be diffrentiated
    % x_val : array for x values
    % step_size : array for h values
    func_handle = str2func(['@(x)' char(func)]); % Convert string to function handle
    fx_minus_h = func_handle(x_val - step_size); % calculate f(x-h)
    fx = func_handle(x_val); % calculate f(x)
    r = (fx - fx_minus_h) ./ step_size; % evaluate the backward difference and return the value
end
```

Central Difference

```
function r = Central_difference(func, x_val , step_size)
    % func : the function to be diffrentiated
    % x_val : array for x values
    % step_size : array for h values
    func_handle = str2func(['@(x)' char(func)]); % Convert string to function handle
    fx_plus_h = func_handle(x_val + step_size); % calculate f(x+h)
    fx_minus_h = func_handle(x_val - step_size); % calculate f(x-h)
    r = (fx_plus_h - fx_minus_h) ./ (2.*step_size); % evaluate the central difference and return the value
end
```

Results of Differentiation

For $F(x) = x^4 - 6x^2 + x + 2$

x values were 100 points from -2,2

h values were [0.5, 0.05, 0.005]

Here are the results :

```
For the Function :x.^4 - 6*x.^2 + x + 2
at x = -2 and h = 0.5000
The Forward Difference = 0.125000
The Backward Difference = -18.125000
The Central Differeance = -9.000000
The actual Value = -7.000000

at x = -2 and h = 0.0500
The Forward Difference = -6.119875
The Backward Difference = -7.920125
The Central Differeance = -7.020000
The actual Value = -7.000000

at x = -2 and h = 0.0050
The Forward Difference = -6.910200
The Backward Difference = -7.090200
The Central Differeance = -7.000200
The actual Value = -7.000000
```

For $G(x) = \sin^3(x) - \cos^2(x)$

X values were 100 points from 0 to π.

H values were [0.5, 0.05, 0.005].

Here are the results:

```
For the Function :sin(x).^3 - cos(x).^2
at x = 0 and h = 0.5000
The Forward Difference = 0.680089
The Backward Difference = -0.239307
The Central Differeance = 0.220391
The actual Value = 0.000000

at x = 0 and h = 0.0500
The Forward Difference = 0.052455
The Backward Difference = -0.047461
The Central Differeance = 0.002497
The actual Value = 0.000000

at x = 0 and h = 0.0050
The Forward Difference = 0.005025
The Backward Difference = -0.004975
The Central Differeance = 0.000025
The actual Value = 0.000000
```

For H(x) = e^{-x^2}

X values were 100 points from -3,3.

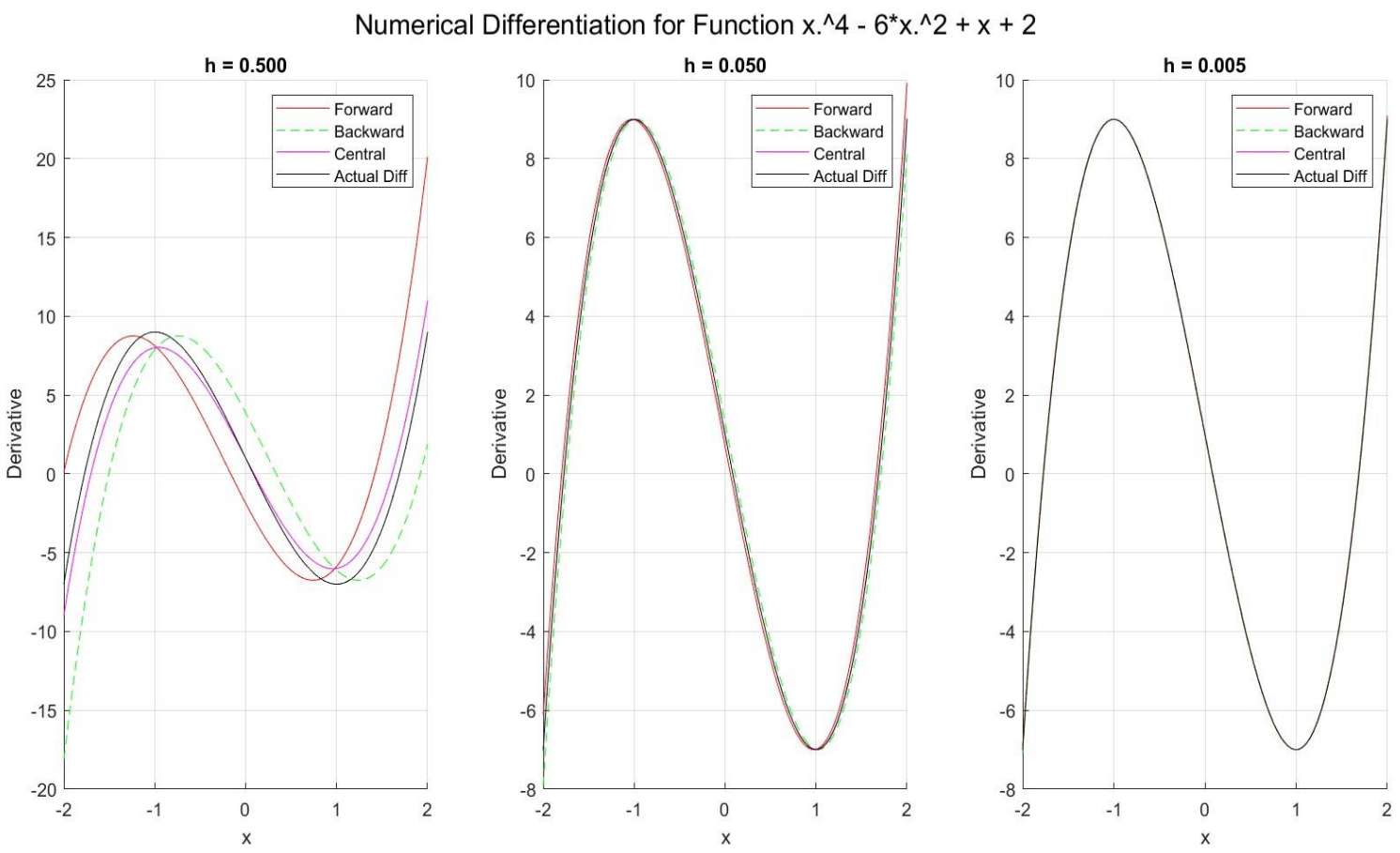
H values were [0.5, 0.05, 0.005].

Here are the results:

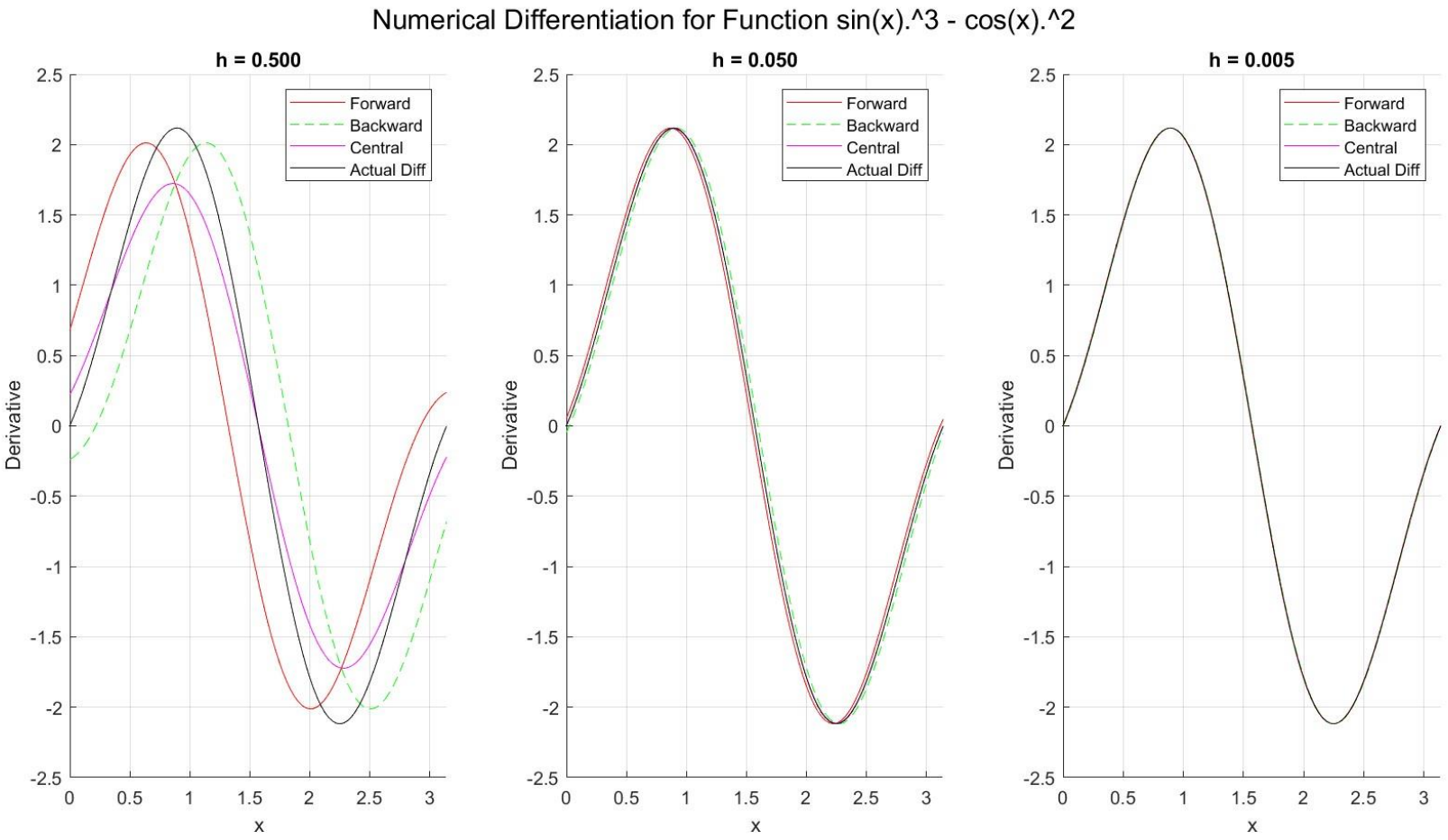
```
For the Function :exp(-x.^2)
at x = -3 and h = 0.5000
The Forward Difference = 0.003614
The Backward Difference = 0.000237
The Centra Differeance = 0.001926
The actual Value = 0.000740

at x = -3 and h = 0.0500
The Forward Difference = 0.000855
The Backward Difference = 0.000644
The Central Differeance = 0.000750
The actual Value = 0.000740

at x = -3 and h = 0.0050
The Forward Difference = 0.000751
The Backward Difference = 0.000730
The Central Differeance = 0.000741
The actual Value = 0.000740
```

Plot for Differentiation methods

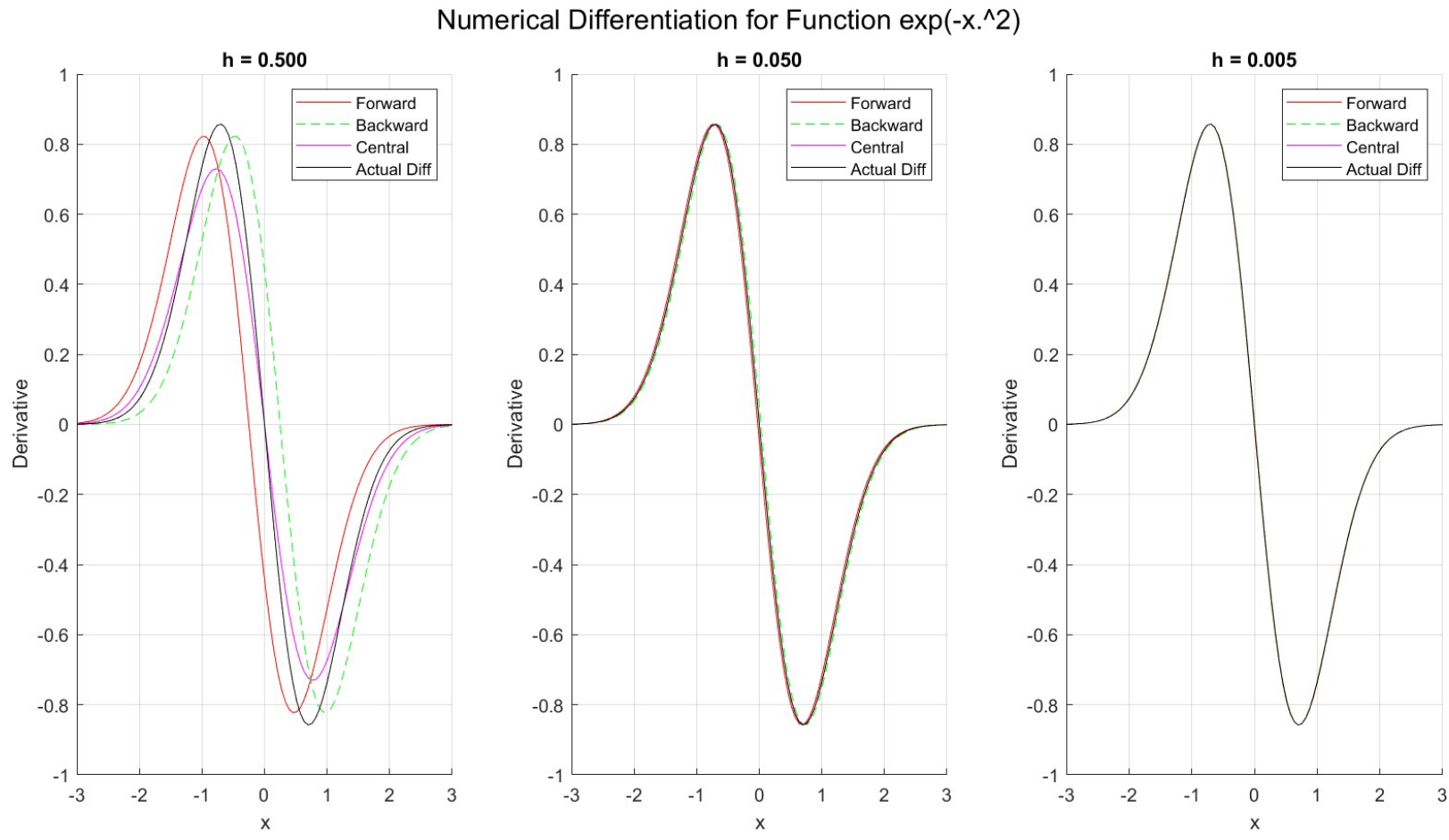
For F(x)



For G(x)


Numerical Differentiation for Function sin(x).^3 - cos(x).^2
h = 0.500
h = 0.050
h = 0.005
Forward
Backward
Central
Actual Diff
Derivative
x

For H(x)


Numerical Differentiation for Function exp(-x.^2)
h = 0.500
h = 0.050
h = 0.005
Forward
Backward
Central
Actual Diff
Derivative
x

Discuss Differentiation Resluts

From the values and plots, we observe that the Central Difference method provides more accurate results compared to the Forward and Backward Difference methods. This is because the Central Difference uses information from both sides of the point of interest, leading to reduced error.

Additionally, as the step size (h) decreases, the error becomes smaller. This indicates that smaller (h) values result in better approximations of

the derivative. However, it is important to note that (h) should not be too small, as extremely small values can lead to numerical instability due to floating-point precision limitations.

Numerical Integration

Numerical Integration

Numerical Integration methods are used for approximating area under carves when analytical solutions are unavailable or impractical.

For this project we will use Three famous methods

1. Trapezoidal Rule

The Trapezoidal Rule approximates the area under a curve by dividing the interval into smaller subintervals and treating each segment as a trapezoid. The area of each trapezoid is computed and summed to estimate the integral.

- **Formula**:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[\left(f(a)+f(b)\right)+2\sum_{i=1}^{n-1} f(x_i)\right]$$

  Where $h = \frac{b-a}{n}$ is the step size

- **How It Works**:
  - Divide the interval [a,b] into n subintervals.
  - For each subinterval, approximate the curve using a straight line.
  - Calculate the area of each trapezoid and sum them.

- **Advantages**:

- Simple and quick to implement.
- Works well for smooth functions.

- **Disadvantages**:
  - Less accurate for functions with significant curvature unless the interval is finely divided.

- **Implementation**

```
function I = trapozodial(F,lower_limit, upper_limit, n)
    % F : is function to be integrated
    % n : number of intervals
    h = (upper_limit - lower_limit) / n;
    s = 0.5*(F(lower_limit) + F(upper_limit));
    for i = 1:n - 1
        s = s + (F(lower_limit + i*h));
    end

    T_I = s * h;
    I = T_I;
    return;

end
```

2. Simpson's Rule

Simpson's Rule improves accuracy by approximating the curve with parabolic segments instead of straight lines. This approach fits a parabola to every pair of subintervals, making it more precise for smooth functions.

- **Formula**:

  $\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + f(b) + 4\sum_{odd\ i} f(x_{i)} + 2\sum_{even\ i} f(x_i)]$

  where $h = \frac{b-a}{n}$ and n must be even.

- **How It Works**:
  - Divide the interval [a,b] into n subintervals, ensuring n is even.
  - Fit a parabola to each pair of subintervals.
  - Use the formula to calculate the integral.
- **Advantages**:
  - More accurate than the Trapezoidal Rule for smooth functions.
  - Converges faster with fewer subintervals.

- **Disadvantages**:
  - Requires an even number of subintervals.
  - May struggle with functions that oscillate rapidly or have discontinuities.
- **Implementation**

```
function I = simpson(F,lower_limit, upper_limit,n)
    % F : is function to be integrated
    % n : number of intervals "must be even"
    % finding the width
    h = (upper_limit - lower_limit) / n;
    % calculating the first and last terms
    s = F(lower_limit) + F(upper_limit);

    % loop over the odd values
    for i=1:2:n-1
        s = s + 4*F(lower_limit + i*h);
    end
    % loop over the even values
    for i = 2: 2: n-1
        s = s + 2*F(lower_limit + i*h);
    end
    % calculate the simpson integration
    S_I = h/3 * s;
    I = S_I;
    return;
end
```

3. Monte Carlo Integration

Monte Carlo Integration estimates the integral using random sampling. It is particularly useful for higher-dimensional problems or functions with irregular behavior.

- **Steps**:
  1. Define a rectangle over the interval [a,b][a, b][a,b] with a height greater than the maximum value of the function.
  2. Generate random points within this rectangle.
  3. Count how many points fall below the curve.
  4. Estimate the integral as:

$$Integral \approx \frac{Points\ under\ the\ curve}{Total\ points} \times Area\ of\ the\ shape$$

- **Advantages**:
  - Robust for complex, discontinuous, or irregular functions.
  - Scales well for high-dimensional integrals.
- **Disadvantages**:
  - Slow convergence compared to deterministic methods.
  - Requires many random points for high accuracy in 1D problems.

- **Implementation**

```
function I = monte_carlo(F,lower_limit,upper_limit, N)
    % F : is function to be integrated
    % N : number of points
    if nargin < 4
        N = 10000;
    end
    M = 1.4 * max(F(linspace(lower_limit,upper_limit)));
    for i = 1:N
        % generate random points
        x_val = rand(1) * (upper_limit - lower_limit) + lower_limit;
        y_val = rand(1) * M;
        % compare random against the curve
        fx = F(x_val);
        if y_val < fx
            under(i,1) = x_val;
            under(i,2) = y_val;
        else
            above(i,1) = x_val;
            above(i,2) = y_val;
        end
    end
    % remove the zeros from the array
    under2(:,1) = nonzeros(under(:,1));
    under2(:,2) = nonzeros(under(:,2));
    above2(:,1) = nonzeros(above(:,1));
    above2(:,2) = nonzeros(above(:,2));
    % calculate the Monte carol integration
    M_I = length(under2) / N * (M*(upper_limit - lower_limit));
```

Results of Integration

For $F(x) = x^4 - 6x^2 + x + 2$

Interval = [0,1], N = 16

Here are the results :

```
For the Function: x.^4-6*x.^2+x+2
The analytical solution is: 0.700000
The Trapozodial solution is: 0.697395
The Error using Trapozodial is: 0.002605
The Simpson solution is: 0.700002
The Error using Simpson is: 0.000002
The Monte Carlo solution is: 0.974122
The Error using Monte Carlo is: 0.274122
```

For $G(x) = \sin^3(x) - \cos^2(x)$

Interval = $[0,\pi]$, N = 16

Here are the results :

```
For the Function: sin(x).^3-cos(x).^2
The analytical solution is: -0.237463
The Trapozodial solution is: -0.237438
The Error using Trapozodial is: 0.000025
The Simpson solution is: -0.237567
The Error using Simpson is: 0.000104
The Monte Carlo solution is: 0.901070
The Error using Monte Carlo is: 1.138533
```

For H(x) = e^{-x^2}

Interval = [-1,1], N = 16

Here are the results:

```
For the Function: exp(-x.^2)
The analytical solution is: 1.493648
The Trapozodial solution is: 1.491731
The Error using Trapozodial is: 0.001917
The Simpson solution is: 1.493652
The Error using Simpson is: 0.000004
The Monte Carlo solution is: 1.485248
The Error using Monte Carlo is: 0.008400
```

Discuss Integration Resluts

From the values and plots, we observe that **Simpson's Rule** consistently provides the most accurate results, closely matching the analytical solutions. This is due to its use of parabolic approximations, which are highly effective for smooth functions. The **Trapezoidal Rule** also produces reasonable results but is less accurate compared to Simpson's Rule, especially for functions with significant curvature.

The **Monte Carlo Method**, while versatile, exhibits the highest error compared to the other methods. This is expected due to its reliance on random sampling, which requires a very large number of points (N) to achieve high accuracy. For 1D integrals, deterministic methods like Simpson's Rule and Trapezoidal Rule are more efficient and accurate.

Additionally, as the number of intervals (n) or sample points increases:

- The error for Simpson's Rule and Trapezoidal Rule decreases significantly, demonstrating faster convergence for these methods.
- The Monte Carlo Method shows a slower error reduction, as its convergence rate is proportional to $\frac{1}{\sqrt{N}}$.

In summary:

- Simpson's Rule is the most effective method for accurate integration of smooth functions.
- Trapezoidal Rule is a simpler alternative with reasonable accuracy.
- Monte Carlo Method is less suitable for 1D problems but remains valuable for higher-dimensional or irregular functions.